UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE BIOMEDICINA

ANÁLISIS COMPARATIVO DEL DESEMPEÑO DE UN MODELO CNN PERSONALIZADO FRENTE A MODELOS PREENTRENADOS PARA LA CLASIFICACIÓN MULTIETAPA DE LA RETINOPATÍA DIABÉTICA

Trabajo de titulación previo a la obtención del
Título de Ingeniero Biomédico

AUTORES: Edith Patricia Córdova Moreira
Cristina Carolina Novillo Jara
TUTOR: Ing. Roberto Gerardo Bayas Toro, Mgs.

Guayaquil - Ecuador
2026

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, **Edith Patricia Córdova Moreira** con documento de identificación N° **0926194002** y **Cristina Carolina Novillo Jara** con documento de identificación N° **0927694224**; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo.

Guayaquil, 2 de marzo del año 2026

Atentamente,

Edith Patricia Córdova Moreira
0926194002

Cristina Carolina Novillo Jara
0927694224

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, **Edith Patricia Córdova Moreira** con documento de identificación Nº **0926194002** y **Cristina Carolina Novillo Jara** con documento de identificación Nº **0927694224**, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del **Proyecto Técnico: ANÁLISIS COMPARATIVO DEL DESEMPEÑO DE UN MODELO CNN PERSONALIZADO FRENTE A MODELOS PREENTRENADOS PARA LA CLASIFICACIÓN MULTIETAPA DE LA RETINOPATÍA DIABÉTICA**, el cual ha sido desarrollado para optar por el título de: Ingeniero Biomedico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 2 de marzo del año 2026

Atentamente,

Edith Patricia Córdova Moreira
0926194002

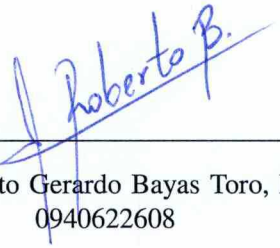
Cristina Carolina Novillo Jara
0927694224

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, **Roberto Gerardo Bayas Toro**, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **ANÁLISIS COMPARATIVO DEL DESEMPEÑO DE UN MODELO CNN PERSONALIZADO FRENTE A MODELOS PREENTRENADOS PARA LA CLASIFICACIÓN MULTI-ETAPA DE LA RETINOPATÍA DIABÉTICA**, realizado por **Edith Patricia Córdova Moreira** con documento de identificación N° **0926194002** y por **Cristina Carolina Novillo Jara** con documento de identificación N° **0927694224**, obteniendo como resultado final el trabajo de titulación bajo la opción **Proyecto técnico** que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Guayaquil, 2 de marzo del año 2026

Atentamente,



Ing. Roberto Gerardo Bayas Toro, Mgs.
0940622608

DEDICATORIA

Este trabajo de titulación está dedicado con profundo amor, respeto y gratitud a la persona que ha sido mi mayor fortaleza, inspiración y ejemplo de vida: A mi padre, Manuel Córdova, por enseñarme el valor del trabajo duro, la responsabilidad y la perseverancia. Gracias por su apoyo constante, por ser un referente de disciplina y por demostrarme que los sueños se cumplen cuando se enfrentan con valentía, dedicación y compromiso. Sus valores y su esfuerzo por vernos crecer, han marcado mi vida y han sido fundamentales para mi formación personal y profesional.

A mi madre, Edith Moreira, por su entrega absoluta, por creer en mí incluso en los días más difíciles y por nunca dejarme sola a lo largo de este camino. Cada palabra de aliento, cada sacrificio silencioso y cada lección que me brindó con amor han sido el motor que me permitió llegar hasta aquí. Todo lo que soy y lo que he logrado, lleva su esencia, su esfuerzo y su amor infinito.

Con especial cariño y nostalgia, dedico este logro a la memoria de mi amado abuelo, Norman Rosales, quien partió de este mundo pero jamás de mi corazón. Su fuerza, sabiduría, consejos y amor dejaron una huella imborrable en mí. A pesar de su ausencia, sé que desde el cielo celebra conmigo cada meta alcanzada. Su ejemplo de lucha y dignidad es el legado más bello que me pudo dejar, y prometo honrarlo en cada paso de mi vida.

Finalmente, a mis amigos: Mauricio E, Daniela M, Christopher N, Jerry O, Danna L, Gregory A, Christian A, Kevin G, Rafael V y Mike G, por ser parte de mi crecimiento, por su lealtad, por sus ánimos constantes y por demostrarnos que la verdadera amistad sostiene, acompaña y motiva. Gracias por cada consejo, risa, aprendizaje y conversación que hicieron este camino más ligero y significativo.

Finalmente, dedico este logro a todas las personas que formaron parte de mi desarrollo, que compartieron palabras de fe, apoyo y motivación durante este proceso académico. Cada gesto, por pequeño que parezca, sumó en mi vida y en la culminación de esta meta. Este triunfo no solo me pertenece, también es para todos quienes estuvieron y creyeron en mí.

Edith Patricia Córdova Moreira

Este trabajo de titulación está dedicado con amor, respeto y gratitud a mi padre Christian Novillo por su ejemplo de esfuerzo, disciplina, y resiliencia, enseñándome que los retos son oportunidades para crecer y que la constancia siempre tiene su recompensa, a mi madre Margarita Jara, por su amor incondicional, su paciencia infinita y su apoyo silencioso pero constante, recordándome siempre que no hay meta imposible cuando se trabaja con pasión y dedicación; y a mi hermana Arianna Novillo, por se mi inspiración y mi compañía en cada paso, por su capacidad de transformar los desafíos en momentos de aprendizaje y por hacer que cada logro se sienta más valioso con su apoyo y cariño.

También dedicó este logro en honor a la memoria de mi abuelo Nicolas Novillo, que en vida me enseño que la fortaleza y la honestidad son las verdaderas guías del camino, su ejemplo permanece vivo en mi corazón y continúa inspirando cada decisión, esfuerzo, y paso que doy, dejando un legado que siempre me acompañara.

Asimismo, extiendo este reconocimiento a todas las personas que estuvieron presentes en el proceso, que no dudaron en apoyarme, brindar su compañía, y compartir sus conocimientos, haciendo cada desafío más llevadero y cada logro más significativo.

Cristina Carolina Novillo Jara

AGRADECIMIENTO

Agradezco a Dios por ser la luz que guía mi vida, por darme fortaleza en los momentos de cansancio, claridad en medio de los desafíos y esperanza cuando el camino parecía difícil. Gracias por acompañarme en cada paso, por sostener mis sueños y recordarme que todo propósito tiene un tiempo perfecto. Esta meta alcanzada es también el reflejo de su gracia y su amor infinito.

A mis padres, mi madre Edith Moreira y mi padre Manuel Córdova, quienes han sido mi mayor bendición y apoyo incondicional. Gracias por su amor, por sus sacrificios silenciosos, por enseñarme valores como la humildad, la responsabilidad y la perseverancia; y por creer en mí incluso cuando las dudas intentaban hacerme retroceder. Cada esfuerzo que hicieron por darme educación, consejos y ánimo constante se convirtió en la base sólida que me permitió construir este logro. Nada de esto habría sido posible sin su dedicación, paciencia y amor inquebrantable.

A mis hermanos, Ligia Aviles y Ángel Aviles, por ser mi compañía y soporte incondicional en este camino. Gracias por cada palabra de aliento, por su paciencia en mis días de ausencia mientras me dedicaba a este proyecto, y por celebrar conmigo cada pequeño avance como si fuera propio. Su presencia, cariño y confianza en mí fueron un motor silencioso pero poderoso que me impulsó a seguir adelante. Este logro también es suyo, porque en cada paso sentí su apoyo y su amor. Gracias por estar siempre, por creer en mí y por ser parte fundamental de mi vida.

Agradezco también a todos los docentes, profesionales y personas que compartieron conmigo sus conocimientos o me brindaron apoyo de alguna manera durante mi formación académica. Gracias por guiarme, enseñarme, corregirme con paciencia y aportar a mi crecimiento con profesionalismo y dedicación. Cada enseñanza recibida se convirtió en un ladrillo más de esta meta culminada.

Por último, agradezco a cada persona que desde una oración, un consejo, un mensaje o un gesto me motivó a seguir adelante. Gracias por creer en mí cuando yo también estaba aprendiendo a hacerlo. Hoy cierro esta etapa con gratitud, pero también con el compromiso firme de continuar honrando cada enseñanza y cada apoyo recibido.

Edith Patricia Córdova Moreira

Agradezco a Dios, por ser mi guía, y mi fortaleza a lo largo de este proceso, por sostenerme en los momentos de incertidumbre, brindarme paz y darme la motivación necesaria para no rendirme y continuar hasta alcanzar mis metas.

A mis padres Christian Novillo y Margarita Jara, quienes, con su amor inmenso, y cada sacrificio realizado en silencio, me han apoyado, aconsejado, enseñándome a perseverar y a enfrentar los desafíos con determinación; y a mi hermana Arianna Novillo, por ser quien siempre estuvo a mi lado, escuchándome, animándome y creyendo en mí, incluso cuando dudaba y quería retroceder. Su dedicación, cariño y respaldo fueron los cimientos sólidos sobre los que pude construir cada paso de este logro, permitiéndome avanzar con gratitud, fortaleza y la certeza de que nunca caminé sola.

Mi más sincera gratitud a mi tutor de tesis el Ing. Roberto Bayas, cuya paciencia, disposición y compromiso, fueron fundamentales para cumplir esta meta, y a todos los docentes que, con sus conocimientos, guía y orientación, enriquecieron esta investigación y contribuyeron a mi formación profesional.

Asimismo, extiendo mi gratitud, hacia mis compañeros y amigos, por su apoyo, compañía y palabras de animo han hecho esta experiencia un camino más llevadero y significativo.

Cristina Carolina Novillo Jara

RESUMEN

El presente trabajo desarrolla y evalúa un sistema de aprendizaje profundo orientado a la clasificación multietapa de la retinopatía diabética mediante imágenes de fondo de ojo. El estudio se enmarca en un enfoque comparativo, cuyo propósito es analizar el desempeño de un modelo CNN propuesto y contrastarlo con distintas arquitecturas preentrenadas bajo condiciones homogéneas de entrenamiento y evaluación, con el fin de identificar diferencias atribuibles a la estructura de cada modelo.

Para llevar a cabo este análisis, se consolidó un conjunto de datos balanceado en varias categorías diagnósticas, construido mediante la integración de repositorios públicos de retinografías y organizado mediante partición estratificada en entrenamiento, validación y prueba. Las imágenes fueron sometidas a un proceso de estandarización y preprocesamiento para asegurar consistencia en las entradas del sistema, y los modelos se entrenaron bajo una configuración experimental controlada. La evaluación final se realizó sobre un conjunto de prueba independiente, empleando métricas multiclase y herramientas de análisis como matriz de confusión y curvas ROC bajo un esquema One-vs-Rest.

Los resultados evidenciaron que el modelo CNN propuesto alcanzó un desempeño global competitivo frente a las arquitecturas preentrenadas, destacando un buen equilibrio entre capacidad discriminativa y estabilidad durante el entrenamiento. De manera general, las mayores confusiones se concentraron en las clases intermedias, lo cual es coherente con el solapamiento visual entre estadios contiguos de la enfermedad. Asimismo, el análisis de generalización mostró la presencia de sobreajuste moderado en los modelos evaluados, lo que sugiere la necesidad de estrategias adicionales de regularización y validación externa para fortalecer la robustez del sistema ante datos no vistos.

Como validación operativa, se implementó una plataforma web interactiva que integra el modelo seleccionado y permite ejecutar inferencia a partir de la carga de retinografías, aplicando preprocesamiento automático y presentando la predicción multiclase de forma accesible. En conjunto, el trabajo aporta un marco reproducible para la comparación de arquitecturas de aprendizaje profundo en la clasificación multietapa de retinopatía diabética y sienta bases para su integración en entornos digitales de apoyo al análisis.

Palabras clave: Retinopatía diabética, clasificación multietapa, aprendizaje profundo, redes neuronales convolucionales, modelos preentrenados, métricas de evaluación, AUC-ROC, matriz de confusión, plataforma web.

ABSTRACT

This work develops and evaluates a deep learning system aimed at multi-stage classification of diabetic retinopathy from fundus images. The study follows a comparative approach, whose purpose is to analyze the performance of a proposed CNN model and contrast it with different pretrained architectures under homogeneous training and evaluation conditions, in order to identify differences attributable to the structure of each model.

To conduct this analysis, a balanced dataset across several diagnostic categories was consolidated by integrating public fundus image repositories and organizing it through a stratified split into training, validation, and test sets. The images underwent a standardization and preprocessing procedure to ensure consistent inputs, and the models were trained under a controlled experimental configuration. Final evaluation was performed on an independent test set using multiclass metrics and analysis tools such as the confusion matrix and ROC curves under a One-vs-Rest scheme.

The results showed that the proposed CNN model achieved competitive overall performance compared to the pretrained architectures, highlighting a good balance between discriminative capability and training stability. In general, most confusions occurred in intermediate classes, which is consistent with the visual overlap between adjacent disease stages. Likewise, the generalization analysis indicated the presence of moderate overfitting in the evaluated models, suggesting the need for additional regularization strategies and external validation to strengthen the system's robustness on unseen data.

As an operational validation, an interactive web platform was implemented that integrates the selected model and enables inference by uploading fundus images, applying automatic preprocessing, and presenting the multiclass prediction in an accessible manner. Overall, this work provides a reproducible framework for comparing deep learning architectures for multi-stage diabetic retinopathy classification and lays the groundwork for integration into digital decision-support environments.

Keywords: diabetic retinopathy, multi-stage classification, deep learning, convolutional neural networks, pretrained models, evaluation metrics, ROC-AUC, confusion matrix, web platform.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I. Introducción

La retinopatía diabética (RD) es una complicación microvascular progresiva asociada a la diabetes mellitus y constituye una de las principales causas de pérdida visual prevenible a nivel mundial. Su evolución se relaciona con el daño crónico de la microvasculatura retiniana, lo que genera alteraciones estructurales y funcionales en la retina. Clínicamente, la RD se manifiesta en diferentes etapas de severidad desde ausencia de retinopatía hasta formas avanzadas y cada una implica decisiones diagnósticas y terapéuticas distintas, por lo que la clasificación multietapa resulta relevante para el tamizaje, el seguimiento y la priorización de pacientes.

El diagnóstico oportuno de la RD es fundamental para reducir el riesgo de deterioro visual irreversible; sin embargo, la evaluación manual de retinografías requiere tiempo, experiencia y disponibilidad de especialistas en oftalmología. Además, la interpretación puede presentar variabilidad interobservador, especialmente en estadios intermedios donde los signos clínicos son sutiles y se superponen entre categorías vecinas. Estas limitaciones motivan el desarrollo de sistemas de apoyo basados en inteligencia artificial que contribuyan a mejorar la estandarización, la rapidez y la accesibilidad del proceso de clasificación.

En este contexto, las redes neuronales convolucionales (CNN) han mostrado eficacia en tareas de clasificación de imágenes médicas por su capacidad para extraer características jerárquicas y reconocer patrones complejos. A partir de esta base, el uso de modelos preentrenados mediante transferencia de aprendizaje permite aprovechar representaciones aprendidas en grandes corpus, mientras que la personalización de arquitecturas busca adaptar dichas representaciones a señales específicas del dominio clínico. De forma complementaria, modelos basados en atención, como Vision Transformer (ViT), permiten modelar relaciones globales entre regiones de la imagen, lo que puede ser ventajoso cuando la evidencia clínica se distribuye a lo largo del fondo de ojo. Asimismo, arquitecturas optimizadas para clasificación como YOLOv8-cls aportan alternativas eficientes con alto desempeño discriminativo.

Bajo esta motivación, la presente investigación propone un análisis comparativo del desempeño de un modelo CNN personalizado basado en EfficientNet-B0 frente a modelos preentrenados para la clasificación multietapa de retinopatía diabética a partir de imágenes de fondo de ojo. El modelo propuesto incorpora capas de refinamiento y un módulo de *External Attention*, y su desempeño se contrasta con ResNet-50, DenseNet-121, ViT-B/16 y YOLOv8-cls utilizando métricas homogéneas de evaluación. La comparación se realiza mediante exactitud, precisión, sensibilidad (*recall*), especificidad, F1-score, AUC-ROC (one-vs-rest) y matriz de confusión, con el fin de cuantificar tanto el rendimiento global como el comportamiento por clase y la tendencia al sobreajuste.

Como componente de validación operativa, el modelo seleccionado se integra en una plataforma web interactiva (RetinaVisionAI) que permite cargar retinografías, ejecutar preprocesamiento automático y visualizar la predicción multiclase junto con probabilidades asociadas. De este modo, el estudio busca aportar evidencia comparativa sobre la selección de arquitecturas para clasificación multietapa de RD y establecer una base funcional para su utilización como herramienta de apoyo en entornos digitales de tamizaje y análisis.

II. Problema

La retinopatía diabética se clasifica según su severidad en cuatro etapas principales: leve, moderada, severa y proliferativa. La magnitud de esta condición crítica afecta el 34.6 % de las personas con diabetes, el 6.96 % desarrolla su forma proliferativa, mientras que paralelo a ello el 6.81 % padece de edema macular y más del 10.2 % sufre una variante que amenaza la visión. En España la prevalencia es del 15.28 %, con 1.92 % en sus etapas severas, una tasa menor a comparación de otros países, lo que señala la importancia del soluciones que aporten ala detección temprana mediante programas de cribado [1].

Estudios en Polonia han demostrado que uno de los principales retos existentes de esta patología radica en que su diagnóstico depende completamente de la interpretación de cada oftalmólogo, pues, no existe un programa estándar y universal que diagnostique de manera eficiente la retinopatía diabética. Esta dependencia humana dificulta la estandarización, lo que puede generar confusiones para poder establecer el grado de afección del paciente. También se ha demostrado mayor dificultad para diferenciar lesiones en sus etapas moderadas y severas, en consecuencia, el tiempo de espera para determinar el diagnóstico puede extenderse, lo que atrasa el análisis de otros expedientes [2].

Numerosos estudios han demostrado que los modelos de redes neuronales profundas presentan altas tasas de desempeño en la detección de retinopatía diabética, con sensibilidades superiores al 95 % y especificidades cercanas al 95 %. En un estudio reciente realizado en un programa nacional de cribado en India, un algoritmo de aprendizaje profundo alcanzó una sensibilidad del 95,9 % y una especificidad del 94,9 %, superando el rendimiento promedio de los especialistas humanos en la interpretación de retinografía [3].

En este contexto, realizar un análisis de diferentes redes neuronales permitiría identificar la arquitectura más efectiva para reconocer patrones de interés, optimizando el aprendizaje con grandes volúmenes de datos y automatizando tareas clave. Además, este trabajo podría reducir los errores diagnósticos, acelerando la detección temprana y mejorando la atención visual preventiva. De esta manera, se establece un marco eficiente y estandarizado, que sienta bases sólidas para futuras investigaciones en inteligencia artificial aplicada a la salud visual.

III. JUSTIFICACIÓN

El presente proyecto, propone el análisis comparativo de un algoritmo perzonalizado para la detección de la retinopatía diabética, mediante un método organizado que involucra la recopilación y el tratamiento de un conjunto de datos representativo. Estos datos contendrán imágenes etiquetadas, cubriendo todas las fases de la enfermedad según la clasificación ETDRS.

La validación del rendimiento del algoritmo construido se realizará mediante una comparación directa con modelos de redes neuronales existentes y preentrenadas. Esta evaluación se llevará a cabo utilizando métricas estadísticas sólidas que incluyen sensibilidad, especificidad, valor predictivo positivo, valor predictivo negativo y el área bajo la curva ROC, con la finalidad de demostrar su fiabilidad diagnóstica [4].

El modelo se centrará en el procesamiento jerárquico de las imágenes, analizando la información a través de múltiples capas, en las primeras etapas de la red, se identifican elementos básicos como líneas, bordes y formas simples; posteriormente, mediante procesos de agrupación y ajustes no lineales, se reconocen patrones más complejos asociados a lesiones características de la retinopatía diabética; conforme la red avanza hacia capas más profundas, logra distinguir estructuras médicas específicas como microaneurismas, exudados y hemorragias, permitiendo una interpretación más precisa de la imagen. Esta progresión jerárquica, junto con la visualización de los mapas de activación generados por el modelo, proporciona una caracterización detallada de los distintos grados de severidad de la enfermedad y contribuye a que el algoritmo clasifique de forma confiable cada imagen según el estudio evolutivo de la retinopatía.

Además, este trabajo contribuye a consolidar un marco de estudio más preciso y sistemático para la identificación de patrones complejos, fortaleciendo el entrenamiento del modelo y la capacidad de análisis de la información recopilada. Asímismo, se integrará el algoritmo mejor evaluado en un entorno digital, optimizando el tiempo para el procesamiento de los datos, mejorando la gestión y trazabilidad de la información, y la automatización de tareas del preprocesamiento, redimensionamiento, normalización y entrenamiento de la base de datos, ampliando su accesibilidad y contribuyendo a la estandarización de procesos de análisis de imágenes en distintos contextos académicos y de investigación. Este enfoque también se alinea con la tendencia mundial hacia la democratización de las tecnologías de salud digital aplicada a la oftalmología, promoviendo la equidad en el acceso a servicios de diagnóstico avanzado.

IV. OBJETIVOS

IV-A. Objetivo general

Analizar el desempeño de una CNN personalizada, mediante la comparación con modelos preentrenados para la clasificación multietapa de la Retinopatía Diabética.

IV-B. Objetivos específicos

- Establecer la base de datos, aplicando técnicas de curación, preprocesamiento y aumento de datos para garantizar la calidad del conjunto de entrenamiento.
- Desarrollar un modelo personalizado de Redes Neuronales Convolucionales (CNN) para la clasificación automática de la severidad de la Retinopatía Diabética.
- Comparar el desempeño del modelo CNN desarrollado con modelos preentrenados mediante métricas estadísticas.
- Validar el desempeño del algoritmo personalizado mediante su integración en la plataforma web interactiva y la ejecución de pruebas controladas para asegurar su fiabilidad operativa y usabilidad en el entorno digital.

V. Marco Teórico

V-A. *Fundamentos de la Retinopatía Diabética*

V-A1. Definición: Según National Eye Institute, la retinopatía diabética es una afección del ojo que puede causar pérdida de visión y ceguera en personas con diabetes. Afecta los vasos sanguíneos de la retina (la capa de tejido sensible a la luz en la parte de atrás del ojo). Esta enfermedad se desarrolla cuando la hiperglucemia prolongada causa daño a los capilares retinianos, debilitando sus paredes, aumentando su permeabilidad y provocando obstrucción vascular, lo que resulta en isquemia retiniana y posteriormente en la creación de conductos sanguíneos irregulares (neovascularización) [5], tal como se ilustra en la Figura 1.

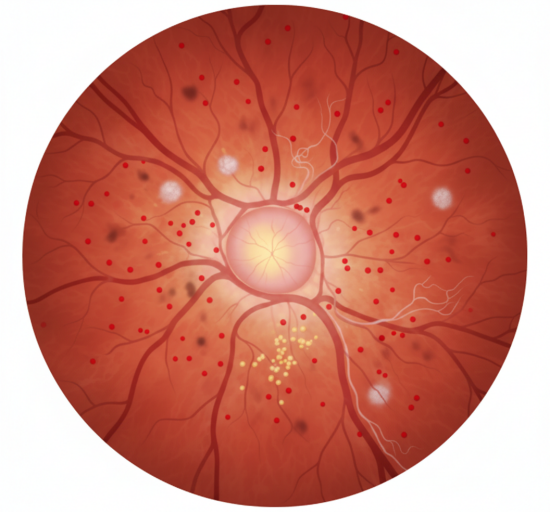


Figura 1. Ilustración de la Retinopatía Diabética (RD). Fuente: Autor Propio.

V-A2. Etiopatogenia de la retinopatía diabética: La teoría sobre el daño a las neuronas en la retinopatía diabética, o RD, sugiere que es por el exceso de glucosa la cual provoca estrés oxidativo, además de la activación de la proteína C quinasa y la formación de productos de glicosilación. Esto afecta la barrera que protege el ojo, lo que a su vez causa estrés en el retículo endoplásmico, lo cual lleva a la muerte celular programada de las células ganglionares y los pericitos. Un ciclo vicioso de hipoxia se establece, con incremento en citoquinas proinflamatorias como VEGF e IGF-1, promoviendo edema macular y neovascularización. Las pruebas que miden la función visual demuestran cambios antes que se vean afectados los micro vasos, revelando que el daño neuronal antecede al vascular en la Retinopatía Diabética [6].

V-A3. Prevalencia Global de la retinopatía diabética: De acuerdo con la Organización Mundial de la Salud, la cantidad mundial de adultos diabéticos se cuadriplicó, aumentando de 108 millones en 1980 a unos asombrosos 422 millones para el 2014. El problema de la diabetes a nivel global prácticamente se duplicó, alzándose del 4.7 % al 8.5 % durante ese mismo lapso de tiempo; este incremento fue aún más vertiginoso en países con rentas bajas y medianas. Solo en América, las estimaciones calculaban unos 35 millones de individuos afectados allá por el 2010. En Ecuador, los registros de casos en 2010 alcanzaron los 92.629, a pesar de que la cantidad real probablemente excede ampliamente esta debido al subdiagnóstico persistente. Actualmente, se estima que más de 103 millones de personas viven con algún grado de esta enfermedad, y que aumentará a los 130 millones para el año 2030 y superará los 161 millones para el año 2045, debido al aumento constante de la diabetes [7].

V-B. *Etapas de la Retinopatía Diabética (RD)*

La clasificación de imágenes de RD se realiza en función de la gravedad de las lesiones visibles en la retina, siguiendo un esquema clínico estandarizado.

V-B1. Retinopatía no Proliferativa (NPDR) Leve: Etapa inicial caracterizada únicamente por la presencia de microaneurismas. Los microaneurismas son pequeñas protuberancias en las paredes de los vasos sanguíneos que aparecen como puntos rojos diminutos. Los modelos de DL deben estar calibrados para detectar estas lesiones sutiles, al ser el primer signo inequívoco de la enfermedad [8], tal como se ilustra en la Figura 2.

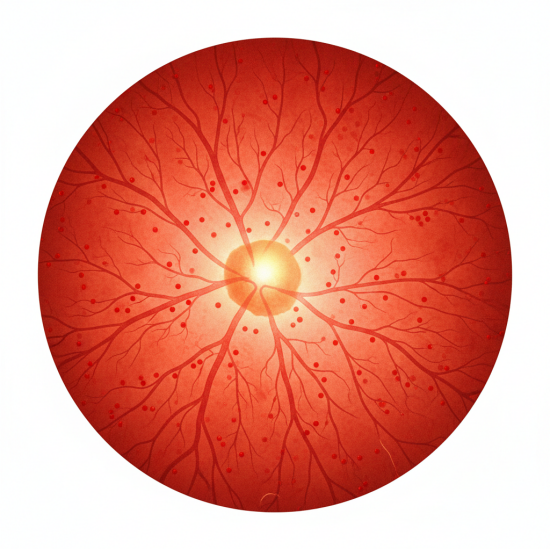


Figura 2. Manifestación de la Retinopatía no Proliferativa (NPDR) Leve. Fuente: Autor Propio.

V-B2. Retinopatía no Proliferativa (NPDR) Moderada: Se identifica por la presencia de microaneurismas más numerosos, además de hemorragias en un cuadrante o más de la retina. También puede haber exudados duros (depósitos de lípidos) y/o exudados algodonosos (infartos de capa de fibras nerviosas). La detección en esta etapa es crítica porque el aumento de lesiones indica un riesgo significativamente mayor de progresión [8], tal como se ilustra en la Figura 3.

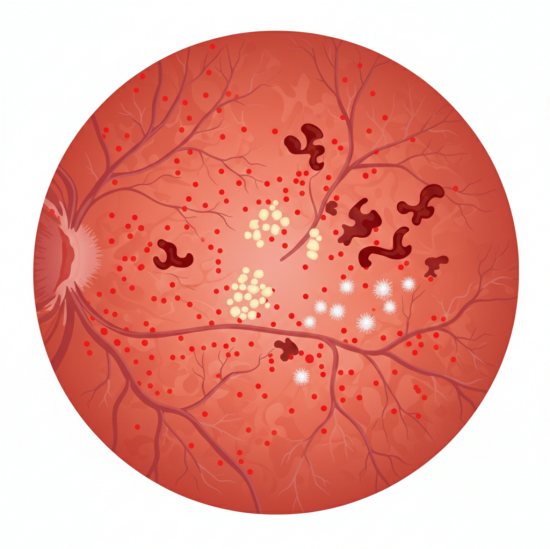


Figura 3. Características de la Retinopatía no Proliferativa (NPDR) Moderada. Fuente: Autor Propio.

V-B3. Retinopatía no Proliferativa (NPDR) Severa: Esta etapa avanzada de NPDR se diagnostica cuando se observan hemorragias retinianas graves en los cuatro cuadrantes, o un número elevado de microaneurismas venosas o anomalías microvasculares intrarretinianas (AMIR). La presencia de estas lesiones prefigura la progresión inminente a la forma más grave y representa una alta probabilidad de pérdida de visión [8], tal como se ilustra en la Figura 4.

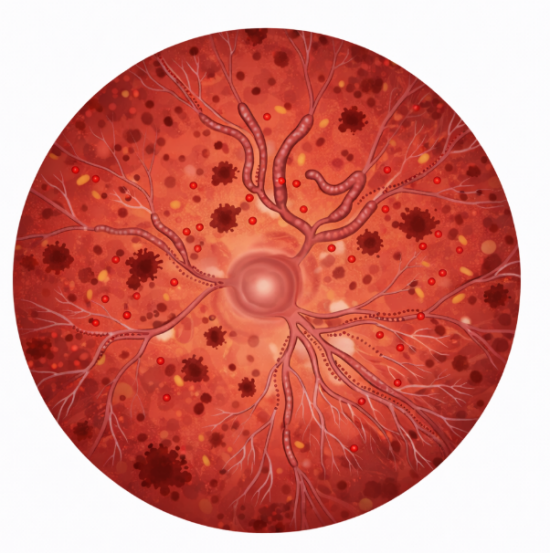


Figura 4. Características de la Retinopatía no Proliferativa (NPDR) Severa. Fuente: Autor Propio.

V-B4. Retinopatía Proliferativa (PDR): La etapa más avanzada y peligrosa, caracterizada por la aparición de neovascularización (formación de nuevos vasos sanguíneos anormales) sobre la superficie de la retina o el disco óptico. Estos vasos son frágiles y pueden causar hemorragias vítreas graves o fibrosis, llevando a un desprendimiento de retina por tracción y, en consecuencia, a la ceguera permanente si no se trata con urgencia [8], tal como se ilustra en la Figura 5.

Figura 5. Características de la Retinopatía no Proliferativa (NPDR) Proliferativa. Fuente: Autor Propio.

V-C. Clasificación Multietapa de la Retinopatía Diabética

La Tabla I, basada en el esquema mostrado en la imagen de referencia, presenta la clasificación multietapa de la Retinopatía Diabética (ETDRS) según su severidad, relacionando cada etapa clínica con sus principales características diagnósticas, facilitando la comprensión del proceso patológico. Asimismo, sirve como base para el desarrollo de modelos de inteligencia artificial aplicados al análisis automatizado de imágenes oftalmológicas.

Tabla I
ESQUEMA DE CLASIFICACIÓN MULTIETAPA DE LA RETINOPATÍA DIABÉTICA PARA MODELOS DE DEEP LEARNING. FUENTE: AUTOR PROPIO.

Etapa	Descripción	Características Clínicas
Etapa 0 (No-DR)	**Descripción:** Ausencia de Retinopatía Diabética.	**Características clínicas:** - Retina completamente normal - Ausencia total de microaneurismas - Ausencia de hemorragias - Ausencia de exudados - Integridad vascular completa
Etapa 1 (Mild-NPDR)	**Descripción:** Retinopatía Diabética Leve.	**Características clínicas:** - Presencia de microaneurismas únicamente - Lesiones microvasculares sutiles - Sin hemorragias significativas - Sin exudados
Etapa 2 (Moderate-NPDR)	**Descripción:** Retinopatía Diabética Moderada.	**Características clínicas:** - Microaneurismas múltiples - Hemorragias en uno o más cuadrantes - Posibles exudados duros (depósitos lípidos) - Posibles exudados algodonosos (infartos nerviosos)
Etapa 3 (Severe-NPDR)	**Descripción:** Retinopatía Diabética Severa.	**Características clínicas:** - Hemorragias en los cuatro cuadrantes - Número elevado de microaneurismas venosos - Anomalías microvasculares intrarretinianas (AMIR) - Lesiones que indican isquemia retiniana grave
Etapa 4 (PDR)	**Descripción:** Retinopatía Diabética Proliferativa.	**Características clínicas:** - Neovascularización en retina o disco óptico - Vasos sanguíneos anormales y frágiles - Riesgo de hemorragia vítrea - Riesgo de desprendimiento de retina por tracción

La Figura 6 ilustra un sistema de clasificación comprehensivo que va más allá del abordaje oftalmológico tradicional de la retinopatía diabética, reconociendo que se trata de una enfermedad sistémica con manifestaciones multiorgánicas.

Figura 6. Clasificación Multietapa de la Retinopatía Diabética. Fuente: Autor Propio.

V-D. Importancia de la Clasificación Multietapa para Modelos de IA

La clasificación multietapa es superior a enfoques binarios y ternarios porque:

- **Granularidad Clínica:** Permite identificar exactamente en qué etapa se encuentra el paciente, facilitando decisiones terapéuticas específicas.
- **Detección Temprana:** El modelo puede identificar la transición de una etapa a otra, permitiendo intervención preventiva antes del avance de la enfermedad.
- **Evaluación de Desempeño Precisa:** Las métricas de precisión, sensibilidad y especificidad pueden evaluarse por etapa, identificando en cuál el modelo tiene mayor dificultad.
- **Validación Clínica:** Facilita la comparación con diagnósticos oftalmológicos estándar, que también utilizan este esquema de clasificación.
- **Curva ROC Multietapa:** Permite generar curvas ROC individuales para cada etapa, evaluando la capacidad discriminativa del modelo en transiciones críticas (ej: leve vs moderada).

V-E. Tratamiento de la retinopatía diabética

Se sustenta primordialmente en el control riguroso de la glucemia, además de la presión arterial, con el propósito de mitigar su avance. Para el edema macular, se prefiere como terapia inicial la administración de inyecciones intraoculares de agentes anti-VEGF, aunque se podría emplear, alternativamente, implantes de corticosteroides o láser focalizado. En cuanto a la retinopatía proliferativa, el tratamiento clave es la fotocoagulación láser panretiniana, que en ocasiones se ve suplementada por anti-VEGF. Por último, ante cuadros clínicos más severos, como hemorragias vítreas persistentes o desprendimiento de retina por tracción, se implementa la vitrectomía, esto con la finalidad de conservar o recuperar la visión del paciente [9], tal como se ilustra en la Figura 7, mientras que los principales métodos de tratamiento, indicaciones específicas y objetivos clínicos se presentan en la Tabla II.

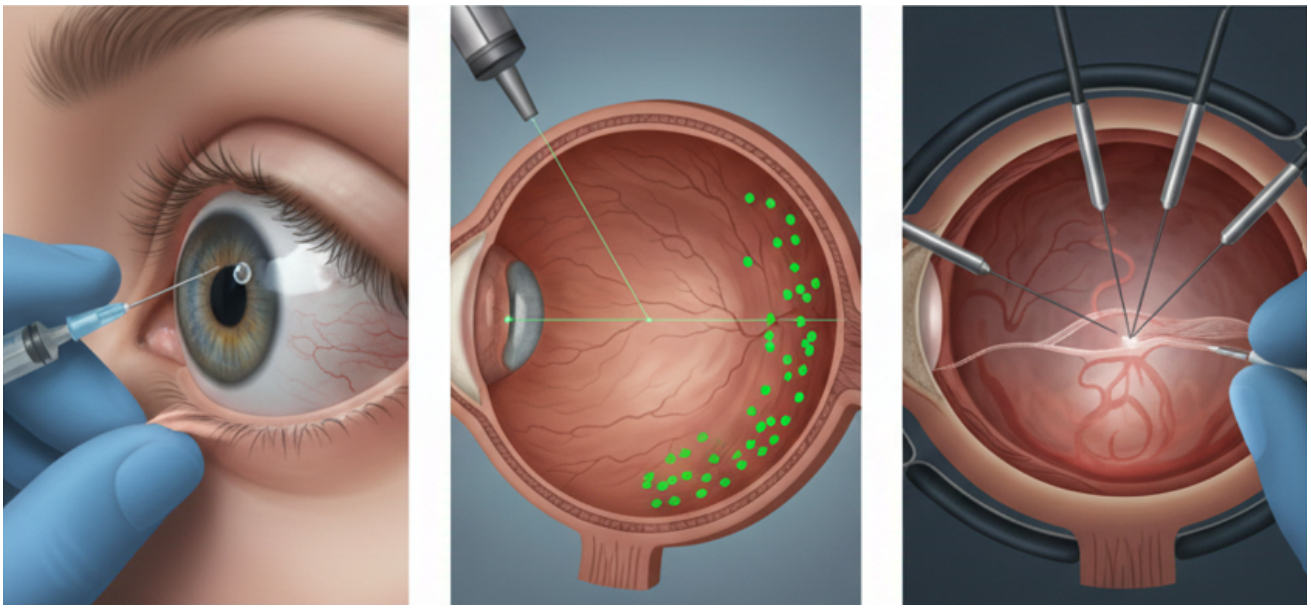


Figura 7. Estrategias terapéuticas en retinopatía diabética. Fuente: Autor Propio.

Tabla II
PRINCIPALES MÉTODOS DE TRATAMIENTO PARA LA RETINOPATÍA DIABÉTICA. FUENTE: AUTOR PROPIO.

Método de Tratamiento	**Indicación Específica**	**Objetivo Primordial**
Control Sistémico	Estado general del paciente	Control riguroso de glucemia y presión arterial para mitigar el avance.
Inyecciones Anti-VEGF	Edema Macular (Terapia inicial)	Bloquear el crecimiento de vasos y reducir la fuga de líquido.
Implantes / Láser Focal	Edema Macular (Alternativo)	Opción secundaria a las inyecciones de agentes anti-VEGF.
Fotocoagulación Láser	Retinopatía Proliferativa	Tratamiento panretiniano para detener la neovascularización.
Vitrectomía	Cuadros severos (Hemorragias o desprendimiento)	Intervención quirúrgica para conservar o recuperar la visión.

V-F. Fundamentos de la Inteligencia Artificial

La IA revoluciona la medicina modernizando el diagnóstico y personalizando tratamientos. A través de algoritmos avanzados de aprendizaje automático, analiza grandes conjuntos de datos tales como imágenes médicas y expedientes clínicos, descubriendo patrones intrincados para detectar dolencias con más premura y exactitud. Por otro lado, permite una medicina más específica, considerando el perfil genético y antecedentes de cada individuo para proponer los tratamientos más adecuados. También predice la posibilidad de afecciones crónicas, facilitando intervenciones preventivas. Sin embargo, su despliegue demanda abordar problemas significativos relacionados con la transparencia algorítmica, protección de datos y una cooperación más cercana entre médicos y tecnólogos [10], tal como se ilustra en la Figura 8, y se complementa con la Tabla III, donde sintetiza las aplicaciones principales de la inteligencia artificial en el ámbito médico.

Figura 8. Fundamentos de la Inteligencia Artificial. Fuente: Autor Propio.

Tabla III
APLICACIONES DE LA INTELIGENCIA ARTIFICIAL EN EL ÁMBITO MÉDICO. FUENTE: AUTOR PROPIO.

Aplicación	Descripción	Impacto
Diagnóstico Revolucionado	Análisis de imágenes médicas y expedientes clínicos.	Mayor precisión, exactitud y detección temprana de dolencias.
Medicina Personalizada	Consideración del perfil genético y antecedentes individuales.	Propuesta de tratamientos específicos para cada individuo.
Predicción de Enfermedades	Análisis de tendencias en afecciones crónicas.	Facilita intervenciones preventivas y medicina proactiva.

V-G. Redes Neuronales Convolucionales

V-G1. Definición: Las CNNs, constituyen modelos de aprendizaje profundo notablemente competentes en el procesamiento de datos estructurados en forma de cuadrícula. Estas arquitecturas, claramente inspiradas en el sistema visual humano, exhiben la capacidad de aprender automáticamente una jerarquía de características espaciales. Esto se logra mediante la implementación estratégica de capas convolucionales y de agrupación, que se encargan de la extracción de patrones clave, además de capas completamente conectadas destinadas a la clasificación. Finalmente, el entrenamiento del modelo optimiza delicadamente los parámetros, como los kernels, utilizando algoritmos de vanguardia tales como la retropropagación, minimizando los errores de predicción [11], tal como se ilustra en la Figura 9, la cual se complementa con la descripción de las etapas del proceso representada en la Tabla IV, en conjunto con las principales características y aplicaciones presentadas en la Tabla V.

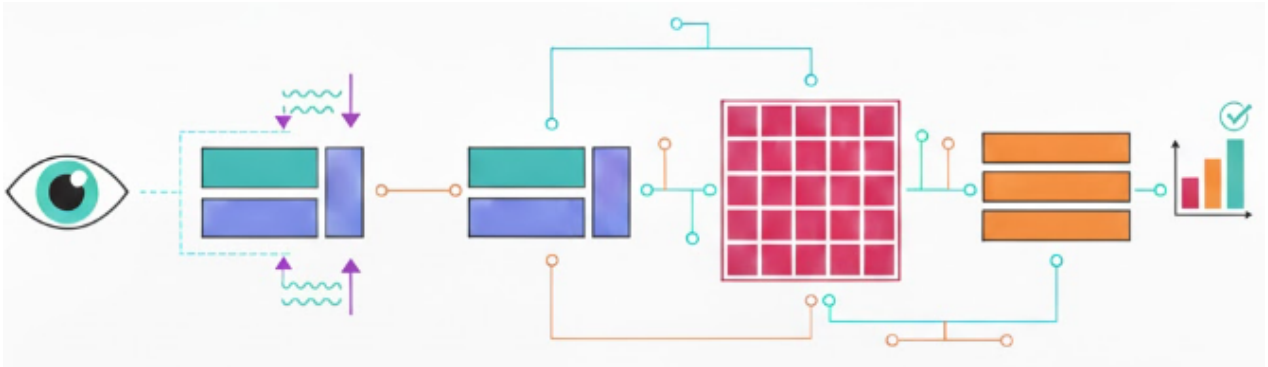


Figura 9. Diagrama de una CNN aplicada al diagnóstico por imágenes de la retina, mostrando el flujo de datos desde la retinografía de entrada hasta la clasificación del estadio de la Retinopatía Diabética. Fuente: Autor Propio.

Tabla IV
ETAPAS DEL PROCESO DE UNA RED NEURONAL CONVOLUCIONAL (CNN). FUENTE: AUTOR PROPIO.

Etapa del Proceso	**Acción Realizada**	**Resultado Esperado**
Imagen de Entrada	Recepción de datos iniciales	Información base para el análisis.
Capas Convolucionales	Extracción de características	Identificación de patrones espaciales mediante *kernels*.
Agrupamiento (Pooling)	Reducción dimensional	Optimización de los datos y reducción de complejidad.
Capas Conectadas	Clasificación final	Interpretación jerárquica de la información procesada.
Resultado Final	Validación y salida	Clasificación o detección exitosa del objeto o patrón.

Tabla V
CARACTERÍSTICAS Y APLICACIONES CLAVE DE LAS REDES NEURONALES CONVOLUCIONALES (CNN). FUENTE: AUTOR PROPIO.

Categoría	**Detalle Informativo**	**Observaciones Clave**
Características Espaciales	Inspiradas en el sistema visual humano.	Aprenden jerarquías: desde bordes y texturas hasta objetos completos.
Entrenamiento	Optimización mediante retropropagación.	Ajuste de parámetros y refinamiento iterativo para minimizar errores.
Aplicaciones Principales	Reconocimiento de rostros, diagnóstico médico (patologías) y vehículos autónomos.	Versatilidad en reconocimiento de patrones y navegación visual.
Ventaja Clave	Aprendizaje automático de características.	No requieren ingeniería manual; la red aprende representaciones óptimas por sí sola.

V-G2. La integración de la IA en la oftalmología: La inteligencia artificial, está transformando la oftalmología de manera muy notable, impulsada por un cúmulo significativo de datos objetivos e imágenes médicas, entre ellas, las retinografías y las tomografías de coherencia óptica, OCT. Su integración ofrece la promesa de diagnósticos más precoces, así como una optimización en el uso de recursos, algo fundamental considerando el incremento de patologías oculares ligado al envejecimiento demográfico. Logros como la autorización del IDx-DR para la retinopatía diabética, ejemplifican su inmenso potencial. Asimismo, estas tecnologías mejoran la detección automatizada de enfermedades oculares, reducen la variabilidad de diagnósticos entre especialistas accediendo a evaluaciones oftalmológicas que apoyen el criterio oftalmológico. No obstante, aun existen dificultades importantes, incluyendo la fiabilidad que depende de la calidad de los datos, la opacidad inherente a los algoritmos y los desafíos relacionados con su efectiva incorporación en el ámbito clínico [12]. Desde esta perspectiva la Figura 10, presenta una visión general de la IA en el análisis de retinografías, en conjunto con la Tabla VI que presenta las principales aplicaciones de la inteligencia artificial en el campo oftalmológico, destacando su impacto en el diagnóstico temprano, la predicción clínicos, el tratamiento personalizado y los aspectos éticos asociados.

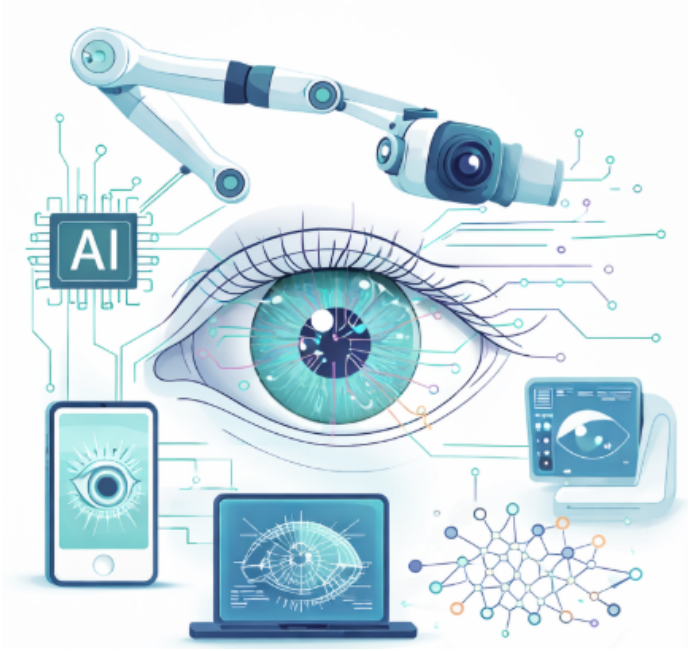


Figura 10. Diagrama ilustrativo de la Inteligencia Artificial (IA) en Oftalmología, mostrando su rol en el análisis de imágenes médicas para el diagnóstico rápido y la gestión optimizada de enfermedades oculares. Fuente: Autor Propio.

Tabla VI
INTEGRACIÓN DE LA INTELIGENCIA ARTIFICIAL EN EL MANEJO DE LA RETINOPATÍA DIABÉTICA Y LA OFTALMOLOGÍA. FUENTE: AUTOR PROPIO.

Categoría	**Aplicación de la IA y Tecnología**	**Impacto en la Retinopatía Diabética**
Diagnóstico y Detección	Análisis automatizado de retinografías y Tomografía de Coherencia Óptica (OCT).	Diagnósticos más precoces y precisos mediante la detección de patrones clínicos no perceptibles al ojo humano.
Automatización y Riesgo	Implementación del sistema IDx-DR para la clasificación automatizada.	Estratificación de riesgo y detección oportuna, optimizando recursos ante el incremento de casos.
Tratamiento Personalizado	Uso de algoritmos de Deep Learning y Redes Neuronales Convolucionales (CNN).	Selección del tratamiento óptimo (Anti-VEGF, láser o vitrectomía) según el perfil clínico del paciente.
Predicción y Prevención	Modelos predictivos basados en antecedentes individuales y control sistémico.	Medicina proactiva mediante el análisis de tendencias de glucemia y presión arterial para evitar la progresión.
Desafíos Técnicos	Entrenamiento de modelos mediante retropropagación.	Necesidad de datos clínicos de alta calidad para garantizar fiabilidad en entornos reales.
Ética y Gobernanza	Transparencia algorítmica y protección de datos del paciente.	Permite que el personal médico comprenda y confíe en los diagnósticos generados por la IA.

V-H. Métricas de evaluación de modelos de clasificación

Permiten cuantificar el desempeño de un modelo de clasificación, evaluando su capacidad para predecir correctamente las clases reales. No basta un porcentaje de aciertos general: con estas métricas se analizan detalladamente aciertos, errores, sensibilidad, precisión y capacidad de discriminar clases, lo que asegura que el modelo funcione de manera robusta, comparable y confiable especialmente en tareas críticas como diagnóstico médico o detección de fraudes. Esta necesidad de evaluación rigurosa ha sido destacada en revisiones recientes de la literatura [13], tal

como se ilustra en la Figura 11.

Figura 11. Métricas clave de evaluación de modelos de clasificación de IA: Precisión (exactitud general), Sensibilidad (detección de positivos) y Especificidad (detección de negativos). Fuente: Autor Propio.

V-H1. Accuracy (Exactitud): Mide la proporción de predicciones correctas (positivas y negativas) sobre el total de muestras evaluadas. Es útil como métrica general cuando las clases están balanceadas, pero se vuelve poco informativa si existe desbalance de clases: un clasificador puede acertar muchas veces prediciendo siempre la clase mayoritaria. Por esta razón, su uso debe complementarse con otras métricas más sensibles a desequilibrios [14], expresada por la Ecuación (1).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Variables:

- $\boldsymbol{TP}$**:** Verdaderos positivos.
- $\boldsymbol{TN}$**:** Verdaderos negativos.
- $\boldsymbol{FP}$**:** Falsos positivos.
- $\boldsymbol{FN}$**:** Falsos negativos.

V-H2. Precision (Precisión): Evalúa la proporción de predicciones positivas que realmente son correctas. Es especialmente valiosa cuando un falso positivo tiene un costo alto como el diagnóstico erróneo de una enfermedad, genera una alarma innecesaria. Alta precisión indica que la mayoría de las predicciones positivas convienen confiar, reduciendo positivos falsos [14], expresada por la Ecuación (2).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

Variables:

- $\boldsymbol{TP}$**:** predicciones correctas de los verdaderos positivo.
- $\boldsymbol{FP}$**:** predicciones incorrectas de los falsos positivos.
- $\boldsymbol{TP + FP}$**:** total de predicciones positivas realizadas por el modelo.

V-H3. Recall (Sensibilidad / Exhaustividad): Mide la proporción de casos positivos reales que el modelo detecta correctamente. Un recall alto implica que la mayoría de positivos reales son capturados, reduciendo falsos negativos. Su optimización es indispensable en sistemas donde no perder un positivo real es prioritario [14], expresada por la Ecuación (3).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

Variables:

- $\boldsymbol{TP}$**:** casos positivos correctamente detectados.
- $\boldsymbol{FN}$**:** casos positivos no identificados por el modelo.
- $\boldsymbol{TP + FN}$**:** total real de casos positivos.

V-H4. F1-Score (Media armónica entre Precision y Recall): El F1-Score equilibra Precisión y Recall en una sola métrica, mediante su media armónica, es ideal cuando existe desbalance de clases o cuando ambos falsos positivos y falsos negativos son costosos. El F1 alto implica buen balance entre detectar muchos positivos reales sin generar tantos falsos positivos. En la práctica, suele ser la métrica recomendada para comparar modelos en tareas críticas [15], expresada por la Ecuación (4).

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

Variables:

- **Precision:** proporción de predicciones positivas que fueron correctas.
- **Recall:** proporción de casos positivos correctamente detectados.
- **F1:** medida balanceada que combina Precision y Recall.

V-H5. Specificity (Especificidad / Tasa de verdaderos negativos): Mide la proporción de verdaderos negativos correctamente identificados como negativos. Es especialmente útil donde los falsos positivos deben evitarse, lo que resulta en pruebas masivas de detección, cuando un diagnóstico erróneo puede generar consecuencias graves o costos innecesarios. Combinada con Recall, da una visión más completa del desempeño del modelo [16], expresada por la Ecuación (5).

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{5}$$

Variables:

- $\mathbf{TN}$: casos negativos correctamente clasificados.
- $\mathbf{FP}$: casos negativos predichos como positivos.
- $\mathbf{TN} + \mathbf{FP}$: total real de casos negativos.

V-H6. Matriz de confusión: Es una tabla estructural que resume cuántas predicciones fueron correctas o erróneas por clase: verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos. Es una herramienta esencial para visualizar errores de clasificación, identificar clases confundidas y determinar qué tipo de error domina (FP o FN). A partir de ella se derivan todas las métricas anteriores. Su uso está ampliamente recomendado en la literatura [17].

V-H7. AUC – ROC (Área bajo la Curva ROC): La métrica AUC-ROC evalúa la capacidad de un modelo para distinguir entre clases positivas y negativas considerando todos los posibles umbrales de decisión. La curva ROC relaciona la tasa de verdaderos positivos (TPR) con la tasa de falsos positivos (FPR). Un AUC cercano a 1 indica un modelo con excelente capacidad discriminativa; un valor cercano a 0.5 sugiere un desempeño equivalente al azar. A diferencia de Accuracy, AUC-ROC es más robusta frente a desequilibrios de clase y es ampliamente usada en evaluación clínica y de riesgo [18], expresada por la Ecuación (6).

$$\text{AUC} = \int_0^1 TPR(FPR)\, d(\text{FPR}) \tag{6}$$

Variables:

- $\mathbf{TPR} = \frac{\mathbf{TP}}{\mathbf{TP+FN}}$: tasa de verdaderos positivos.
- $\mathbf{FPR} = \frac{\mathbf{FP}}{\mathbf{FP+TN}}$: tasa de falsos positivos.
- $\mathbf{AUC}$: área bajo la curva ROC.

V-I. Fundamentos del Aprendizaje Profundo y su Aplicación en la Clasificación de Imágenes Médicas

V-I1. Definición: El análisis de la inteligencia artificial contemporánea; en particular, dentro del campo de la visión por computadora, se fundamenta en la comprensión de tres enfoques esenciales: el aprendizaje automático (Machine Learning), el aprendizaje profundo (Deep Learning) y el aprendizaje por transferencia (Transfer Learning). Cada uno de estos enfoques refleja un avance en la habilidad de las máquinas para identificar patrones y obtener conocimiento a partir de grandes volúmenes de datos [19], como se ilustra en la Figura 12.

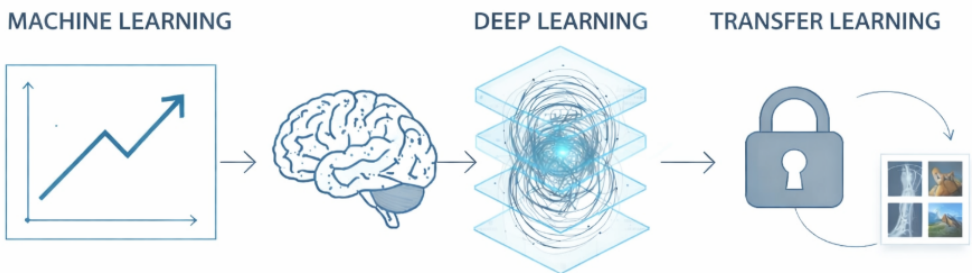


Figura 12. Diagrama conceptual que representa la relación entre el Aprendizaje Automático (Machine Learning), el Aprendizaje Profundo (Deep Learning) y el Aprendizaje por Transferencia (Transfer Learning). Fuente: Autor Propio.

V-I2. Aprendizaje Automático (Machine Learning - ML): El aprendizaje automático (ML) es una subdisciplina de la Inteligencia Artificial centrada en el desarrollo de algoritmos que permiten a las computadoras aprender directamente de los datos sin necesidad de estar programadas explícitamente para una tarea específica. A partir de un conjunto de entrenamiento con entradas y salidas deseadas, estos modelos identifican patrones y relaciones estadísticas que luego utilizan para hacer predicciones o tomar decisiones sobre nuevos datos no vistos. Las técnicas tradicionales de Machine Learning incluyen la regresión lineal, las máquinas de soporte vectorial y los árboles de decisión, las cuales han sido muy útiles históricamente en tareas de clasificación de datos, especialmente en aquellos de tipo tabular o de baja dimensionalidad [15], como se ilustra en la Figura 13.

Figura 13. Diagrama conceptual que ilustra el proceso del Aprendizaje Automático (Machine Learning), desde los datos de entrada hasta la identificación de patrones y la generación de predicciones). Fuente: Autor Propio.

V-I3. Aprendizaje Supervisado: El aprendizaje supervisado se distingue por los datos utilizados durante el proceso de entrenamiento vengan acompañados de etiquetas claramente definidas, tanto para las variables de entrada como para las de salida. Estas etiquetas proporcionan el conocimiento previo al modelo, sirviendo de guía para que el algoritmo pueda aprender la relación entre las características de los datos y los resultados esperados. En la práctica, la asignación de estas etiquetas generalmente es realizada por un equipo especializado, durante una fase de preparación cuidadosa previo al entrenamiento del modelo. Este proceso consiste en etiquetar y garantizar la coherencia y calidad de la información, asegurando que los datos utilizados sean representativos y estén libres de errores que puedan afectar el desempeño del modelo. Además, el aprendizaje supervisado es fundamental para tareas de predicción, debido a que permite anticipar tendencias, estimar resultados futuros e incluso identificar comportamientos anómalos, gracias a su capacidad para detectar patrones subyacentes en los datos. Esta versatilidad lo convierte en una herramienta esencial en el desarrollo de soluciones inteligentes en diversas áreas del conocimiento y la industria [20].

V-I4. Aprendizaje Profundo (Deep Learning): El Deep Learning [DL] representa una categoría avanzada de Machine Learning que utiliza arquitecturas de Redes Neuronales Artificiales [RNA] compuestas por múltiples capas ocultas (tres o más) para modelar abstracciones de alto nivel en los datos. A diferencia del ML tradicional, el DL elimina la necesidad de la ingeniería manual de características, en virtud de que las redes pueden aprender automáticamente las representaciones de las características más relevantes directamente desde los datos sin procesar, como los píxeles de una imagen. Esta capacidad de extracción jerárquica de características permite a los modelos DL manejar la complejidad inherente a los problemas de percepción, como la detección y clasificación de objetos en imágenes [21].

El aprendizaje profundo es una rama del aprendizaje automático que emplea redes neuronales de capas múltiples con el fin de obtener patrones jerárquicos directamente a partir de los datos. Estas redes, con el uso de procedimientos de retropropagación, modifican millones de parámetros que posibilitan el reconocimiento de estructuras complejas sin necesidad del contacto humano directo. Según Solano, en el campo de la medicina, el Deep Learning ha

transformado el diagnóstico por imágenes, brindando una exactitud que es similar o incluso mayor a la que tiene un ojo humano entrenado, sobre todo en enfermedades visuales como la retinopatía diabética (RD). Su uso ha hecho posible la creación de sistemas capaces de identificar lesiones en la retina en etapas tempranas, lo cual mejora el tiempo de diagnóstico y disminuye los errores clínicos [22].

Asimismo, la naturaleza jerárquica del Deep Learning le permite identificar desde rasgos básicos como bordes y colores, hasta estructuras más complejas incluyendo microaneurimas, exudados, hemorragias y alteraciones de los vasos sanguíneos asociados a la RD. Según Bryn Mawr, esta capacidad de representación multiescala ha posibilitado la detección automática de lesiones en imágenes ultra-widefield con niveles de sensibilidad superiores al 90 %, reforzando su potencial en entornos clínicos reales. Por tanto, el Deep Learning no solo mejora la exactitud diagnóstica, sino que también promueve un cambio de paradigma hacia la medicina predictiva y personalizada [23], como se ilustra en la Figura 14.

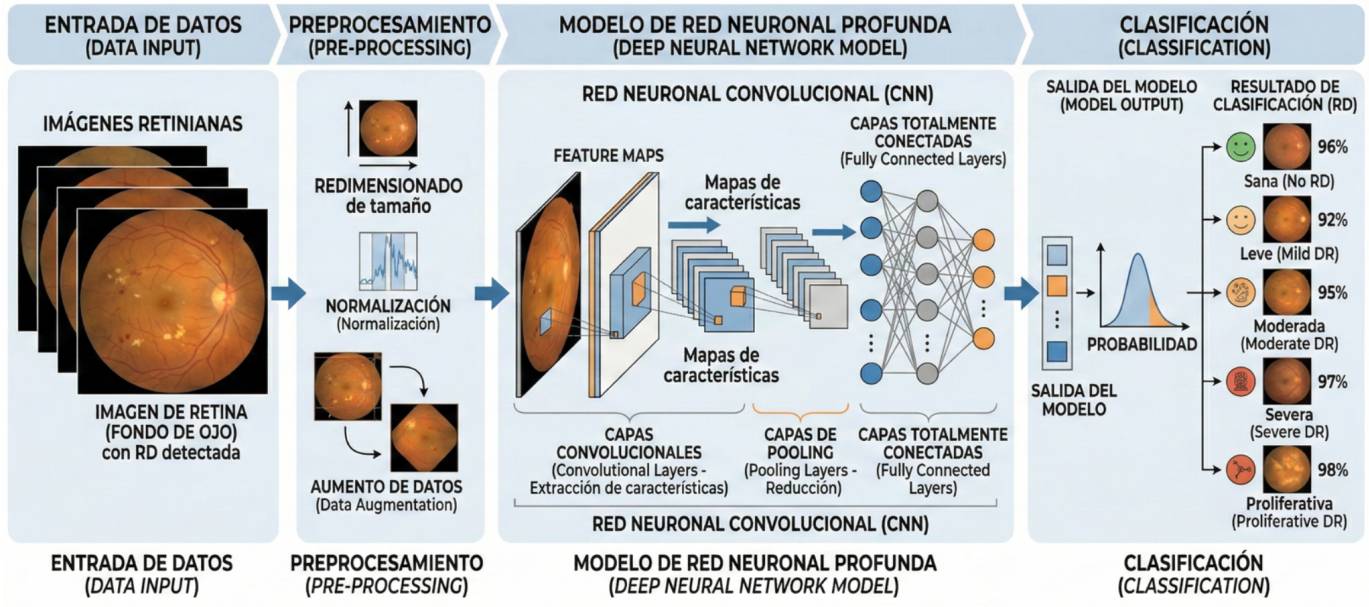


Figura 14. Diagrama del uso de Redes Neuronales Profundas (Deep Learning) para clasificar la retinopatía diabética (RD) a partir de imágenes de retina. Fuente: Autor Propio.

V-I5. Principios Operativos del Aprendizaje Profundo (Deep Learning): El Deep Learning, es el mecanismo matemático-estructural, mediante el cual una red neuronal profunda procesa datos, aprende de ellos y generaliza a nuevos ejemplos, este proceso incluye la representación jerárquica de los datos, las transformaciones capa por capa, el ajuste de parámetros mediante optimización, la gestión del error, y las técnicas que permiten estabilidad, eficiencia y generalización. De forma integrada, definen cómo una red piensa internamente.

El funcionamiento del Deep Learning se cimienta en tres principios operativos esenciales que lo distinguen de sus predecesores:

- **Extracción Jerárquica Características:** Cada capa oculta de la red aprende a representar características con un nivel de abstracción creciente. Las capas iniciales identifican patrones simples (líneas y curvas), mientras que las capas posteriores combinan estos patrones para reconocer conceptos complejos (caras, ojos, microaneurismas) [24].

- **Propagación hacia Atrás (Backpropagation):** Es el algoritmo clave que ajusta los pesos de la red. Este proceso calcula el gradiente de la función de pérdida con respecto a cada peso mediante la regla de la cadena y lo propaga hacia atrás para minimizar el error de predicción [24].

- **Aprendizaje de Extremo a Extremo (End-to-End Learning):** Permite que el modelo tome la entrada cruda y produzca directamente la salida final, optimizando todas las etapas intermedias de forma unificada [24].

V-J. Componentes fundamentales del Aprendizaje Profundo (Deep Learning)

V-J1. Neurona artificial y capas feed-forward: Una unidad básica procesa entradas numéricas mediante una combinación lineal y luego aplica una función no lineal (activación), produciendo una salida. Muchas neuronas se agrupan en capas, y muchas capas se encadenan: así se obtiene una red neuronal profunda. Esta estructura permite aproximar funciones muy complejas. Esta descripción es válida para redes feed-forward profundas [25], se define en la Ecuación (7).

$$y = \phi\left(\sum_i w_i x_i + b\right) \tag{7}$$

- $\mathbf{x_i}$: entradas a la neurona (pueden ser píxeles, características o salidas de capas previas).

- $\mathbf{w_i}$: pesos, parámetros aprendibles que ponderan cada entrada.

- $\mathbf{b}$: bias (sesgo), permite desplazar la función lineal.

- ϕ: función de activación no lineal (ReLU, *tanh*, *sigmoid*, etc.), esencial para modelar relaciones no lineales.

Esta unidad básica, repetida en muchas capas, es la que otorga el poder expresivo al modelo.

V-J2. Propagación hacia delante (forward pass): Cuando la red recibe una muestra de datos, estos se propagan desde la capa de entrada hasta la salida, transformándose capa por capa. Cada capa aplica su transformación (matriz de pesos + bias + activación), generando un vector de activaciones intermedias. Como resultados de este mecanismo, las activaciones internas o representaciones jerárquicas van capturando características crecientemente abstractas del dato: de detalles simples hasta patrones complejos. Este proceso constituye el fundamento del principio de Extracción de Características [25], se define en la Ecuación (8).

$$h^{(l)} = \phi\left(W^{(l)} h^{(l-1)} + b^{(l)}\right) \tag{8}$$

- $\mathbf{h}^{(l)}$: vector de activaciones de la capa l (o entrada si $l = 0$).

- $\mathbf{W}^{(l)}$: matriz de pesos de la capa l.

- $\mathbf{b}^{(l)}$: vector de bias de la capa l.

- ϕ: función de activación no lineal.

V-J3. Criterio de aprendizaje-la función de pérdida (Loss): Para que la red aprenda, necesita una función que cuantifique qué tan equivocada está su predicción en cada ejemplo. Esa función de pérdida compara la salida predicha con la salida real (etiqueta) y da un valor escalar que indica error, ese valor será lo que la red intente minimizar [25], se define en la Ecuación (9).

$$L = -\sum_{c} y_c \log(\hat{p}_c), \qquad \hat{p} = \text{softmax}(z) \tag{9}$$

- $\mathbf{y_c}$: vector *one-hot* que indica la clase verdadera.
- $\mathbf{z}$: *logits*, salida lineal de la última capa antes del *softmax*.
- $\hat{p}_c$: probabilidad predicha para la clase c.
- $\mathbf{L}$: pérdida del ejemplo.

Minimizar este valor promedio sobre muchos ejemplos guía el ajuste de los parámetros de la red.

V-J4. Backpropagation y Optimización: Ajusta los pesos y biases $(w_i,\, b)$, empleando el algoritmo de retropropagación (backpropagation) que calcula el gradiente, es decir, las derivadas parciales de la pérdida respecto a cada parámetro, y luego aplica un optimizador como el Descenso de Gradiente o Adam, modificando dichos parámetros en la dirección que minimize el valor de la pérdida. El ciclo de aprendizaje, propagación hacia adelante, evaluación de la pérdida, retropropagación y actualización de parámetros se repite múltiples veces (epochs) hasta alcanzar la convergencia. Este mecanismo representa el segundo principio fundamental Backpropagation [25], se define en la Ecuación (10).

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z} \cdot \frac{\partial z}{\partial w} \tag{10}$$

- $\mathbf{L}$: pérdida.
- $\mathbf{y}$: salida de la neurona (post-activación).
- $\mathbf{z}$: entrada lineal de la neurona (suma ponderada + bias).
- $\mathbf{w}$: peso cuyo gradiente se evalúa.

Este mecanismo permite el aprendizaje efectivo: La red ajusta sus parámetros gradualmente para mejorar su predicción.

V-J5. Redes convolucionales (CNN): Cuando los datos tienen estructura espacial imágenes, las redes feed-forward densas resultan poco eficientes: demasiados parámetros y no explotan correlaciones locales. Por lo tanto, se utilizan redes convolucionales (CNN), que emplean filtros (kernels) que se deslizan sobre la imagen para detectar patrones locales (bordes, texturas), compartir pesos y reducir la complejidad. Posteriormente, capas de pooling y convolución alternadas construyen representaciones jerárquicas: de lo local a lo global. Este diseño ejemplifica el principio de extracción de características jerárquicas aplicado a datos estructurados [26], se define en la Ecuación (11).

$$y_{i,j,k} = \sum_{m,n,c} W_{m,n,c,k}\, X_{i+m,\, j+n,\, c} + b_k \tag{11}$$

- $X_{i+m,\,j+n,\,c}$: valor de entrada en la posición espacial $(i+m, j+n)$, canal c.
- $W_{m,n,c,k}$: peso del kernel para desplazamiento (m, n), canal de entrada c y canal de salida k.
- b_k: *bias* correspondiente al canal de salida k.
- $y_{i,j,k}$: activación resultante en la posición (i, j) para el canal k.

Este diseño aprovecha la estructura espacial y resulta más eficiente en parámetros y procesamiento que las redes completamente conectadas.

Finalmente, para asegurar que el aprendizaje sea efectivo y generalizable, se incorporan mecanismos de regularización, tales como:

- **Dropout:** Consiste en apagar aleatoriamente una fracción de neuronas en capas ocultas durante cada paso de entrenamiento; de este modo la red no puede depender excesivamente de rutas particulares de activación, lo que reduce la co-adaptación de neuronas y fuerza a que las representaciones internas sean más robustas. Esta técnica disminuye el sobreajuste, especialmente en redes profundas con muchos parámetros y cuando los datos son limitados [27], se define en la Ecuación (12).

$$h' = m \odot h \tag{12}$$

  - $\mathbf{h}$: activaciones originales.
  - $\mathbf{m}$: máscara binaria aleatoria con probabilidad p de mantenerse activa.
  - $\mathbf{h}'$: activaciones resultantes.

- **Weight Decay:** Penaliza el tamaño de los pesos del modelo agregando un término extra a la función de pérdida que crece con la magnitud de los pesos. De este modo, la red prefiere soluciones con parámetros pequeños, menos complejas, lo que evita que ajuste excesivamente a datos de entrenamiento ruidosos, promoviendo generalización y reduciendo overfitting [28], se define en la Ecuación (13,14).

$$\mathcal{L}(\theta) = \mathcal{L}_{\text{original}}(\theta) + \lambda \|\theta\|_2^2 \tag{13}$$

$$\|\theta\|_2^2 = \sum_{i=1}^{n} \theta_i^2 \tag{14}$$

  - $\boldsymbol{\theta}$: pesos del modelo.
  - $\boldsymbol{\lambda}$: fuerza de penalización.
  - $\|\boldsymbol{\theta}\|_2^2$: norma cuadrática que suma los pesos al cuadrado.

- **Batch Normalization (BatchNorm):** Normaliza las activaciones de una capa usando la media y la varianza calculadas sobre cada mini-batch, recentrándolas y reescalándolas. Como resultado, estabiliza y acelera el entrenamiento, permite usar tasas de aprendizaje más altas y reduce problemas de desvanecimiento o explosión de gradientes. Además actúa como regularizador: al homogenizar activaciones evita que la red dependa

demasiado de valores extremos o de configuraciones inestables [29], se define en la Ecuación (15,16).

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{15}$$

$$y = \gamma \hat{x} + \beta \tag{16}$$

- $\mathbf{x}$: activaciones originales.
- $\boldsymbol{\mu_B}$, $\boldsymbol{\sigma_B^2}$: media y varianza del *batch*.
- $\boldsymbol{\epsilon}$: valor pequeño añadido para estabilidad numérica.
- $\boldsymbol{\gamma}$, $\boldsymbol{\beta}$: parámetros aprendibles de escala y desplazamiento.

V-J6. Aprendizaje por Transferencia (Transfer Learning): El aprendizaje por transferencia, aunque forma parte del aprendizaje profundo, se diferencia por su punto de partida, mientras que el Deep Learning puro entrena un modelo desde cero, el aprendizaje por transferencia parte de un modelo previamente entrenado con datos generales y lo ajusta para una tarea médica concreta. Está técnica se basa en la reutilización de las representaciones aprendidas en las capas iniciales, que normalmente capturan características universales como formas o texturas [30].

En la práctica, el aprendizaje por transferencia resulta especialmente útil en el campo médico, donde los conjuntos de datos suelen ser limitados y el etiquetado clínico es costoso. La implementación de Shift Window Transformers preentrenados permite la clasificación de la retinopatía diabética, lo que reduce el riesgo de sobreajuste y acelera el entrenamiento. De esta manera, el Transfer Learning se convierte en una alternativa eficiente que conserva la capacidad de generalización y aprovecha el conocimiento previo de redes entrenadas con millones de imágenes [31].

V-J7. Principios Operativos del Transfer Learning: El Transfer Learning se rige por un conjunto diferente de principios operativos que maximizan la eficiencia y el rendimiento con datos limitados.

1. Reutilización de parámetros preentrenados: El principio fundamental es la Reutilización del Conocimiento, donde los pesos preentrenados actúan como un punto de partida optimizado, evitando la inicialización aleatoria de pesos. En consecuencia, acelera la convergencia y estabiliza el entrenamiento [32].

2. Congelación de Capas (Layer Freezing): Implica mantener los pesos de las capas iniciales (las que aprenden características genéricas) fijos, lo que previene que el pequeño dataset del dominio objetivo sobreescriba el conocimiento general adquirido [33].

3. Feature extraction y Linear probing: Si la *prueba* demuestra que las representaciones z ya separan adecuadamente las clases del *target*, la estrategia más eficiente consiste en congelar los parámetros θ_0 (es decir, no actualizarlos) y entrenar únicamente la cabeza clasificadora sobre dichas características. Este enfoque es rápido, robusto cuando existen pocos datos disponibles y evita distorsionar representaciones previamente aprendidas.En consecuencia, se emplea como *baseline* de bajo costo antes de aplicar cualquier estrategia de *fine-tuning* más invasiva [34], tal como se define en la Ecuación (17, 18).

$$\hat{p} = \text{softmax}(Wz + b) \tag{17}$$

$$z = f_{\theta_0}^{enc}(x) \tag{18}$$

Variables:

- z: *embedding* generado por el encoder preentrenado al procesar una entrada x del dominio *target*.
- W: matriz de pesos entrenables de la cabeza de clasificación.
- b: vector *bias* entrenable correspondiente al clasificador.
- $\hat{p}$: vector de probabilidades predichas para cada clase del problema objetivo.
- θ_0: parámetros congelados del encoder preentrenado (contienen el conocimiento aprendido en el dominio fuente).

Busca obtener un punto de referencia rápido y barato para verificar la calidad de las representaciones transferidas y decidir si es viable realizar un fine-tuning más profundo; lo que evita gasto computacional innecesario.

4. Fine Tuning (Ajuste fino): Si la tarea objetivo exige cambios profundos (clases distintas, dominio distante), se realiza fine tuning: partir de θ_0 y optimizar θ sobre D_{target} para minimizar la pérdida objetivo. Este procedimiento adapta representaciones generales a las particularidades del target, pero puede distorsionar buenas features y empeorar la generalización fuera de distribución (OOD) si no se hace con cuidado. De este modo, su aplicación se decide tras un diagnóstico previo [35], se define en la Ecuación (19).

$$\theta^* = \arg\min_{\theta} \ \mathbb{E}_{(x,y)\sim D_{target}} \left[L(f_\theta(x), y) \right] \tag{19}$$

Variables:

- θ: parámetros actualizables durante el *fine-tuning*.
- D_{target}: distribución de datos del dominio objetivo.
- L: función de pérdida apropiada (cross-entropy, MSE, etc.).
- $f_\theta(x)$: salida del modelo parametrizado por θ.

5. Adaptación de parámetros eficiente (PEFT — Parameter-Efficient Fine-Tuning): Modifica solo una fracción pequeña del modelo (adapters, matrices de bajo rango, nuevas capas ligeras), mientras la mayoría permanece congelada, lo cual reduce memoria, cómputo, permite múltiples adaptaciones con un mismo backbone, y mantiene robustez. En escenarios con recursos limitados o múltiples tareas, es una estrategia muy valiosa [36], se define en la Ecuación (20).

$$W = W_0 + BA \tag{20}$$

Variables:

- W_0: matriz de pesos original (congelada).
- $A \in \mathbb{R}^{r \times d_{in}}$, $B \in \mathbb{R}^{d_{out} \times r}$: matrices entrenables de bajo rango r, con $r \ll d_{in}$.

- r: rango bajo que controla cuántos parámetros nuevos se aprenden.

Adapta la red con mínimo costo, preserva la base aprendida, permite adaptación rápida a nuevas tareas o dominios, y reduce sobreajuste en escenarios con pocos datos.

V-K. Redes Neuronales en la Detección de Retinopatía Diabética (RD)

El uso de Redes Neuronales, particularmente las Redes Neuronales Convolucionales (CNN), ha revolucionado la detección automatizada de la RD, logrando un rendimiento comparable al de los oftalmólogos [37]:

V-K1. Concepto y Funcionamiento de las Redes Neuronales (NN): Una Red Neuronal Artificial es un modelo de cómputo inspirado en la estructura y función del cerebro biológico, compuesto por unidades interconectadas llamadas neuronas o nodos. Cada neurona recibe entradas, aplica una suma ponderada a estas entradas y utiliza una función de activación no lineal para producir una salida que se transmite a la siguiente capa.

V-L. Capas Clave en Deep Learning y Transfer Learning

Las arquitecturas de Deep Learning (como VGG o ResNet, cruciales para TL) están compuestas por bloques de capas que realizan funciones específicas:

V-L1. Capa de Convolución (Convolutional Layer): Esta es la capa fundamental de una CNN y su función es extraer características espaciales de la imagen de entrada mediante el uso de filtros (o kernels). Cada filtro desliza (convoluciona) sobre la imagen para calcular la suma ponderada de sus píxeles, generando un mapa de características (feature map) que destaca patrones específicos como bordes, colores o, en el caso de la RD, lesiones como microaneurismas y hemorragias. El principio de peso compartido en el filtro reduce drásticamente el número de parámetros a entrenar, haciendo que la red sea eficiente [26], se define en la Ecuación (21).

$$y_{i,j}^{(k)} = (X * W^{(k)})_{i,j} + b^{(k)} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X_{i+m,\, j+n} \cdot W_{m,n}^{(k)} + b^{(k)}. \tag{21}$$

Variables:

- $y_{i,j}^{(k)}$: valor de la salida en la posición (i, j) del mapa de características k.
- X: imagen de entrada o mapa de características previo.
- $W^{(k)}$: filtro o kernel de la capa convolucional k.
- $b^{(k)}$: sesgo asociado al filtro k.
- M, N: dimensiones del filtro convolucional.

Esta operación destaca patrones espaciales de la imagen como bordes, texturas o lesiones.

V-L2. Capa de Agrupamiento (Pooling Layer): Su propósito principal es reducir la dimensionalidad espacial de los mapas de características, lo que disminuye la cantidad de cálculos posteriores y minimiza el riesgo de overfitting (demasiado ajuste a los datos de entrenamiento). La técnica más común es el Max Pooling, donde se toma el valor máximo dentro de una región definida del mapa de características, conservando la información más relevante de las características extraídas por la capa convolucional anterior. Esta capa ayuda a

que el modelo sea más robusto a pequeñas traslaciones del objeto en la imagen [26], se define en la Ecuación (22).

$$y_{i,j}^{(k)} = \max \left\{ x_{m,n}^{(k)} \;\middle|\; m \in R_i,\ n \in R_j \right\}. \tag{22}$$

Variables:

- $y_{i,j}^{(k)}$: valor de salida del mapa de características k en la posición (i, j).
- $x_{m,n}^{(k)}$: valores de entrada de la región $R_i \times R_j$ correspondiente a la posición (i, j).
- R_i, R_j: región del mapa sobre la cual se aplica el max pooling.

V-L3. Capa Rectificadora (ReLU - Rectified Linear Unit): Esta capa aplica una función de activación no lineal a la salida de la capa convolucional, siendo la función $f(x) = \max(0, x)$ la más común. Su función es introducir la no linealidad necesaria en el modelo, permitiéndole aprender relaciones complejas y no lineales en los datos, lo cual es esencial para discriminar patrones complejos como los de las lesiones retinianas. Si las redes solo tuvieran activaciones lineales, la composición de múltiples capas siempre resultaría en otra función lineal [38], se define en la Ecuación (23).

$$f(x) = \max(0, x) \tag{23}$$

Variables:

- x: valor de entrada de la neurona.
- $f(x)$: valor de salida después de aplicar la activación.

Introducir esta no linealidad permite que la red aprenda relaciones complejas no lineales.

V-L4. Capa Totalmente Conectada (Fully Connected Layer - FC): Ubicada típicamente al final de la arquitectura CNN, esta capa toma las características extraídas y aplanadas por las capas convolucionales y de pooling y las mapea al espacio de la clasificación final (por ejemplo, las diferentes etapas de la RD). Cada neurona en la capa FC está conectada con cada neurona de la capa anterior, actuando como un clasificador basado en las características de alto nivel aprendidas previamente. Esta capa suele ser la que se reemplaza completamente en las arquitecturas de Transfer Learning [26], se define en la Ecuación (24).

$$y = W \cdot x + b \tag{24}$$

Variables:

- y: vector de salida de la capa FC.
- W: matriz de pesos que conecta cada neurona de la capa previa con cada neurona de la FC.
- x: vector de entrada (generalmente la salida aplanada de la capa convolucional).
- b: vector de sesgos aplicado a la capa FC.

Esta capa actúa como un clasificador utilizando las características de alto nivel aprendidas por la red.

V-L5. Capa de Salida (Output Layer - Softmax): Esta capa final utiliza la función Softmax para convertir las salidas de la capa FC en probabilidades de pertenencia para cada una de las clases de clasificación (por ejemplo, PDR, NPDR severa, etc.). La suma de las probabilidades de todas las clases es igual a uno, proporcionando una puntuación probabilística de la confianza del modelo en su predicción para cada una de las etapas de la Retinopatía Diabética [39], se define en la Ecuación (25).

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{25}$$

Variables:

- $\sigma(z)_i$: probabilidad asignada a la clase i.
- z_i: salida de la neurona i de la capa FC antes de la normalización.
- K: número total de clases del problema.

V-M. Arquitecturas Convolucionales Avanzadas por Visión de Computadora

V-M1. EfficientNet-B0: Es una familia de redes neuronales convolucionales profundas diseñadas para visión por computadora, desarrollada por Google AI. Su innovación principal radica en el concepto de escalado compuesto, una técnica que ajusta tres dimensiones simultáneamente: profundidad, anchura y resolución de entrada. Con un único coeficiente de escala, EfficientNet maximiza la precisión y la eficiencia del modelo, optimizando el uso de recursos computacionales. En particular, EfficientNet-B0 es el modelo base de esta familia, entrenado originalmente con Imagenet utilizando imágenes de tamaño 224×224 píxeles, y es capaz de clasificar 1000 categorías [40].

El principio operativo de EfficientNet-B0 se centra en la utilización de capas convolucionales tradicionales, pero con un escalado uniforme de estas tres dimensiones (profundidad, anchura y resolución), lo que mejora la eficiencia computacional sin perder precisión. Esta estructura lo convierte en una opción ideal para tareas donde se requiere un balance entre precisión y velocidad, como en aplicaciones médicas. En el ámbito de la retinopatía diabética, EfficientNet-B0 ha mostrado un desempeño excelente, siendo útil para la clasificación de imágenes retinianas, facilitando la detección temprana y precisa de la enfermedad, tal y como se ilustra en la Figura 15.

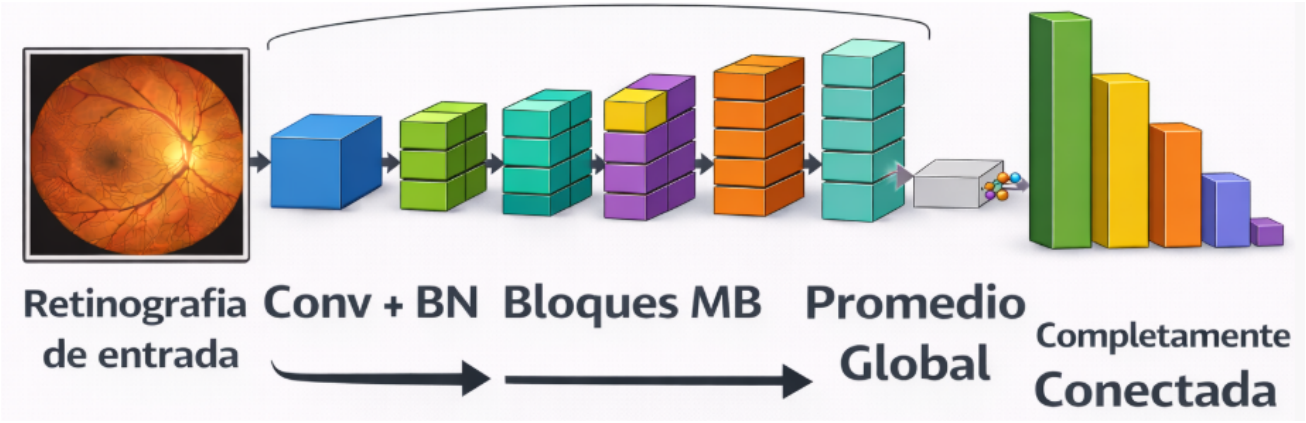


Figura 15. Diagrama de la arquitectura EfficientNet-B0 aplicada al análisis de retinografías para la clasificación multietapa de RD. Fuente: Autor Propio.

V-M2. ResNet-50: Es una arquitectura de red neuronal profunda que utiliza bloques residuales para abordar el problema del desvanecimiento del gradiente en redes muy profundas. Introducida por Microsoft Research en 2015, ResNet propone una innovación clave: las conexiones de salto (skip connections). En lugar de aprender una función de mapeo directo entre la entrada y la salida, los bloques residuales permiten que la red aprenda residuos: la diferencia entre la entrada y la salida deseada. De este modo, se añade un atajo que salta una o más capas de convolución, facilitando que la información fluya sin problemas a través de la red [41], tal como se ilustra en la Figura 16.

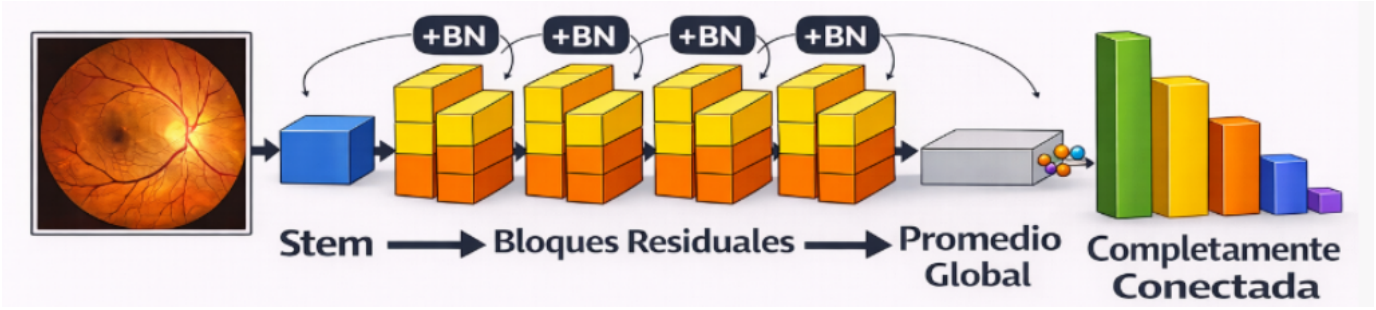


Figura 16. Diagrama del uso de la arquitectura ResNet-50 (Deep Learning) para la clasificación Multietapa de la RD a partir de retinografías de entrada. Fuente: Autor Propio.

El principio operativo de ResNet-50 consiste en usar conexiones residuales que ayudan a la red a entrenarse de manera más eficaz y estable, incluso en modelos con muchas capas. ResNet-50, que contiene 50 capas, se ha utilizado ampliamente en el aprendizaje por transferencia (Transfer Learning), debido a que, facilita el entrenamiento de redes profundas sin la pérdida de información que podría ocurrir con arquitecturas tradicionales. Esta capacidad para aprender representaciones jerárquicas complejas es útil en tareas de visión por computadora como la clasificación y detección de retinopatía diabética, donde ResNet-50 ha demostrado ser eficaz en identificar características clave en imágenes retinianas.

V-M3. DenseNet-121: Es una arquitectura de red neuronal convolucional que sobresale por sus conexiones densas entre capas. A diferencia de redes tradicionales como ResNet, que utilizan conexiones residuales, DenseNet concatena las salidas de todas las capas anteriores como entradas para las capas subsiguientes, lo que favorece una propagación más eficiente de la información. Esta conexión densa ayuda a que la red reciba un flujo continuo de características de las capas previas, reduciendo la redundancia y mejorando la reutilización de características [42], tal y como se ilustra en la Figura 17.

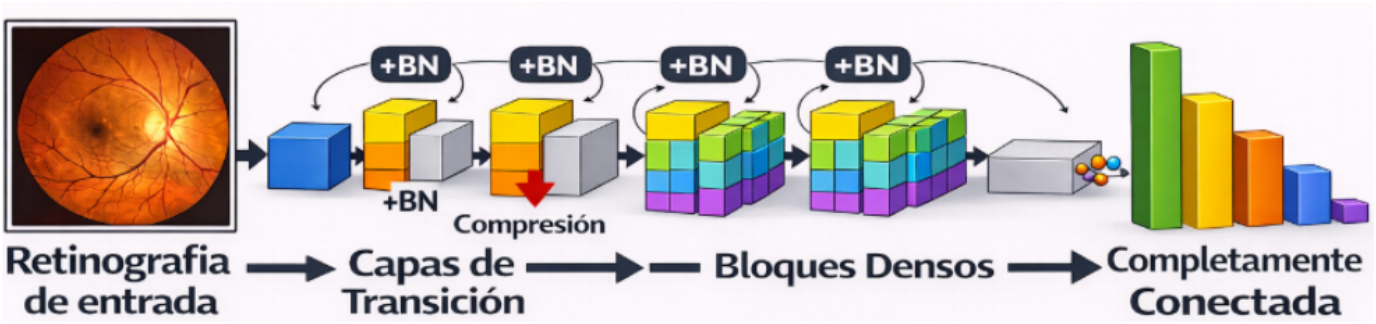


Figura 17. Diagrama del uso de la red neuronal profunda DenseNet-121 (Deep Learning) empleando bloques densos y capas de transición para el análisis de retinografías y clasificación Multietapa de la RD. Fuente: Autor Propio.

El principio operativo de DenseNet-121 consiste en aplicar bloques densos donde cada capa no solo recibe información de la capa anterior, sino también de todas las capas previas, lo que permite aprender representaciones más ricas y variadas. Esto optimiza el rendimiento, particularmente en redes profundas, sin aumentar de manera significativa la cantidad de parámetros. En el ámbito de la retinopatía diabética, DenseNet-121 es útil en tareas de segmentación de imágenes retinianas, especialmente en la identificación de lesiones pequeñas, como los microaneurismas, que son fundamentales para el diagnóstico temprano de la enfermedad.

V-M4. YOLOv8: You Only Look Once v8 es una arquitectura de detección de objetos en tiempo real que se ha diseñado para ser extremadamente rápida y eficiente, desarrollada por Ultralytics en 2023, representa una de las versiones más avanzadas para la detección y clasificación de imágenes. A diferencia de otros métodos de detección de objetos que operan en múltiples etapas, YOLOv8 trata la detección como un problema de regresión única, realizando la localización y clasificación de objetos en una sola pasada. En consecuencia, permite que el modelo prediga tanto la clase del objeto como sus cuadros delimitadores (bounding boxes) en tiempo real, lo que es crucial en aplicaciones que requieren resultados rápidos y precisos. Esta arquitectura combina rapidez, precisión y eficiencia computacional gracias al uso de bloques C2f y estrategias de fusión de características (feature fusion) que optimizan la reutilización de información [43], tal como se ilustra en la Figura 18.

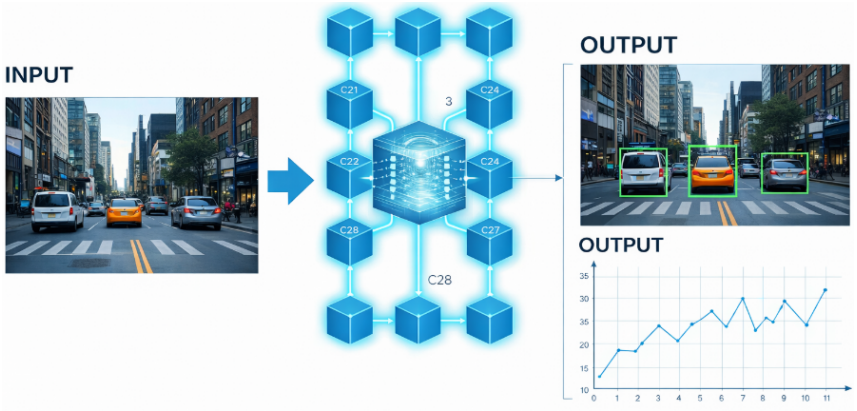


Figura 18. Representación conceptual de la arquitectura YOLOv8, ilustrando sus componentes clave para la detección y clasificación eficiente de objetos en imágenes. Fuente: Autor Propio.

En el ámbito médico, YOLOv8 ha demostrado una gran capacidad para detectar microaneurismas, exudados y hemorragias retinianas. Estas características permite que sea adaptable a tareas de clasificación en múltiples etapas. Por lo que, está arquitectura puede segmentar características asociadas a la retinopatía diabética sin necesidad de una codificación compleja, lo que permite que pueda ser utilizada incluso por personal clínico con conocimientos limitados en programación [44].

El principio operativo de YOLOv8 se basa en un backbone especializado y una cabeza de detección desacoplada que divide las tareas de clasificación y regresión de objetos, mejorando tanto la velocidad como la precisión. El uso de un enfoque anchor-free y la inclusión de módulos C2f permiten una extracción de características más eficiente y una detección multiescala que se adapta a objetos de diferentes tamaños y complejidades. En el caso de la retinopatía diabética, YOLOv8 se aplica en la detección de anomalías en las imágenes retinianas, identificando lesiones o áreas críticas de la retina de forma rápida, lo que ayuda a los médicos a realizar diagnósticos inmediatos y precisos [45].

V-M5. Vision Transformer ViT-B/16: Las arquitecturas Transformer surgieron originalmente en el procesamiento del lenguaje natural, pero desde 2021 se han consolidado como un paradigma dominante también

en visión por computadora. Estos modelos reemplazan las convoluciones por mecanismos de autoatención (self-attention), los cuales permiten al sistema identificar relaciones espaciales globales dentro de la imagen. En lugar de analizar regiones locales, el Transformer evalúa el contexto completo, mejorando la capacidad de detectar patrones sutiles en imágenes médicas [46], tal como se ilustra en la Figura 19.

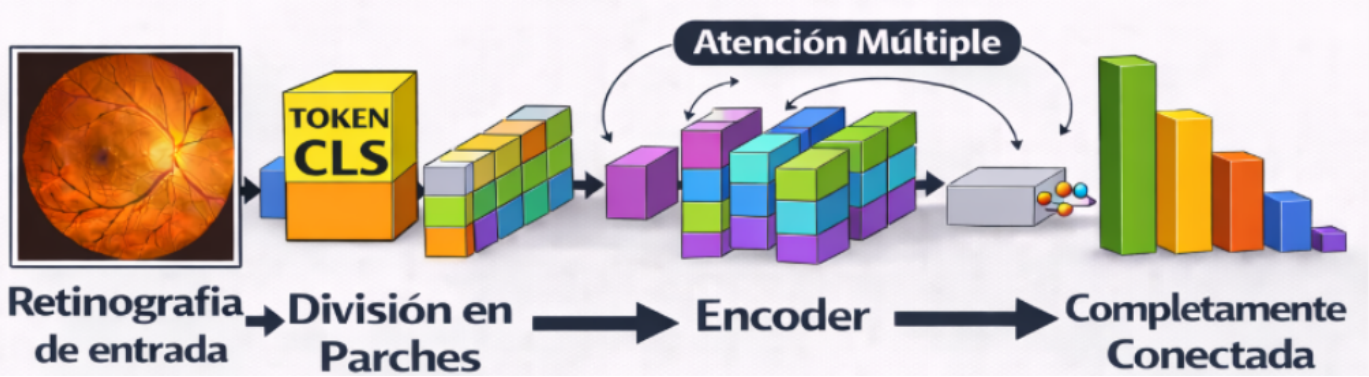


Figura 19. Diagrama del uso de la arquitectura Vision Transformer ViT-B/16 (Deep Learning) mediante división en parches y mecanismos de atención para el análisis de imágenes de retina. Fuente: Autor Propio.

A diferencia de las CNNs, que operan localmente, ViT captura información a gran escala, lo que lo hace adecuado para tareas complejas de visión por computadora. En la clasificación de imágenes médicas, como la retinopatía diabética, ViT-B16 ha mostrado ser muy eficaz al identificar patrones complejos en las imágenes retinianas sin las limitaciones de las arquitecturas convolucionales [47].

V-N. Desafíos en la Clasificación de Imágenes mediante Aprendizaje Profundo

La clasificación automática de imágenes, aun con los avances del Deep Learning enfrenta varios desafíos técnicos y prácticos, especialmente en el ámbito médico, donde la precisión es crítica [48].
Estos desafíos requieren consideraciones específicas en la arquitectura y el entrenamiento de las Redes Neuronales, tal como se ilustra en la Figura 20.

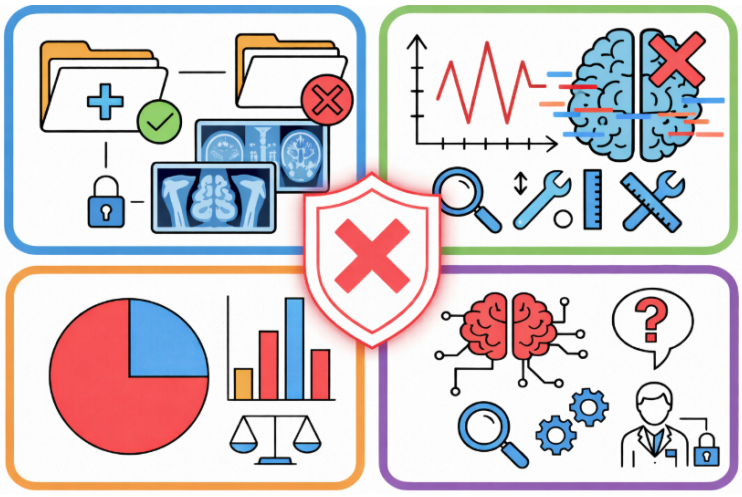


Figura 20. Desafíos clave en la clasificación de imágenes médicas con Deep Learning: falta de datos, variabilidad, desequilibrio de clases e interpretabilidad). Fuente: Autor Propio.

V-N1. Variación Intra-Clase e Iluminación: Este desafío se refiere a la gran variabilidad de apariencia dentro de una misma clase, como una microaneurisma que puede manifestarse en diferentes tamaños, formas y tonalidades en distintas imágenes de fondo de ojo, haciendo que el modelo deba ser robusto a estas diferencias. Además, se presenta la inconsistencia de la calidad de la imagen debido a variaciones en las condiciones de captura, iluminación (brillo o sombras) y artefactos de ruido (polvo en el lente), lo que dificulta la extracción de características estables [49].

V-N2. Similitud Inter-Clase y Oclusión Parcial: Ocurre cuando objetos de clases distintas comparten una gran similitud visual, por ejemplo, distinguir un exudado duro de un artefacto de brillo en la imagen retiniana, lo que puede llevar al modelo a errores de clasificación [50]. La oclusión, donde una parte del objeto de interés (lesión) está cubierta por vasos sanguíneos o artefactos de la imagen, fuerza al modelo a generalizar la identificación basándose en características parciales.

V-N3. Tamaño y Balance del Conjunto de Datos (Dataset): Los modelos de Deep Learning requieren una enorme cantidad de datos etiquetados (millones de imágenes) para entrenar con éxito sus millones de parámetros, lo cual es inalcanzable en la mayoría de los escenarios clínicos por la escasez de expertos para la anotación [51]. Además, los datasets médicos suelen ser desequilibrados (la mayoría de las imágenes son sanas), lo que hace que la red tienda a clasificar todo como la clase mayoritaria (sana), fallando en la detección de patologías raras.

V-N4. Transferibilidad y Generalización del Modelo: El desafío de la transferibilidad se centra en asegurar que un modelo entrenado en un conjunto de datos por ejemplo, un hospital específico mantenga su alto rendimiento cuando se aplica a imágenes capturadas con diferentes dispositivos o en otras regiones geográficas. Si el modelo memoriza patrones específicos del hardware o población de entrenamiento, fallará al generalizar nuevas muestras, lo que disminuye la utilidad clínica y requiere el uso estricto de técnicas de Transfer Learning y fine-tuning [33].

V-Ñ. Regulación de normas y protección de datos clínicos utilizados para el entrenamiento del algoritmo:
Proteger los datos dentro de un algoritmo es crucial porque la privacidad, entendida como el ámbito íntimo de la persona y esencial para la dignidad humana, está gravemente amenazada por la inteligencia artificial. La IA posee una capacidad sin precedentes para recopilar, procesar y explotar masivos volúmenes de información personal en tiempo real, exponiendo detalles íntimos de la vida de las personas sin un consentimiento verdaderamente informado. Esta concentración de datos en pocas corporaciones debilita al Estado-Nación en su rol de garante de derechos humanos. La preservación de la privacidad no se limita a un interés individual, sino que integra un bien social fundamental para la libertad y la democracia, debido a que, evita que el poder tecnológico se utilice para manipulaciones masivas o decisiones discriminatorias, como lo demuestran lecciones históricas [52], tal como se ilustra en la Figura 21.

Figura 21. Regulación de normas y protección de datos clínicos utilizados para el entrenamiento del algoritmo. Fuente: Autor Propio.

V-Ñ1. Normativa ecuatoriana aplicable a la protección de datos digitales: En Ecuador, el principal instrumento legal es la Ley Orgánica de Protección de Datos Personales (LOPDP), la cual establece principios fundamentales como la licitud, finalidad, proporcionalidad, seguridad y responsabilidad proactiva en el tratamiento de datos personales, incluyendo explícitamente los datos de salud, considerados datos sensibles. Esta ley exige la implementación de medidas técnicas y organizativas para prevenir accesos no autorizados, pérdidas, alteraciones o divulgaciones indebidas, lo que resulta crítico en plataformas de diagnóstico asistido por inteligencia artificial [53].

El Instituto Nacional de Estadística y Censos (INEC), aunque no regula directamente sistemas clínicos, establece lineamientos sobre manejo ético de bases de datos, anonimización y agregación de información, los cuales son relevantes para el uso de datasets médicos en procesos de entrenamiento de algoritmos de aprendizaje profundo [54].

Por su parte, la Agencia Nacional de Regulación, Control y Vigilancia Sanitaria (ARCSA) regula los sistemas de información en salud y exige que las plataformas digitales que manejen información clínica cumplan criterios de seguridad informática, control de accesos, auditoría y respaldo de datos, especialmente cuando se trata de aplicaciones con fines diagnósticos o de apoyo clínico [55].

V-Ñ2. Estándares internacionales ISO para seguridad de la información en salud: A nivel internacional, los estándares de la International Organization for Standardization (ISO), constituyen el principal marco técnico para la protección de información digital:

- **ISO/IEC 27001:** Establece un Sistema de Gestión de Seguridad de la Información (SGSI), aplicable a plataformas que almacenan datos clínicos en servidores locales o en la nube. Define controles para gestión de accesos, cifrado, continuidad operativa y gestión de incidentes [56].

- **ISO/IEC 27002:** Es un estándar de controles específicos para la seguridad y protección de la privacidad diseñado para orientar a las organizaciones en la gestión de riesgos de ciberseguridad, protección de datos y defensa de sus activos de información frente a accesos no autorizados o pérdidas de datos [57].

- **ISO 27799:** Orientada específicamente a la seguridad de la información en salud, regula la protección de historias clínicas electrónicas, imágenes médicas y datasets utilizados para entrenamiento de modelos de IA [57].

- **ISO/IEC 27701:** Amplía la 27001 para la gestión de información personal (Privacy Information Management), alineándose con principios de privacidad desde el diseño (privacy by design) [58].

Estos estándares son especialmente relevantes cuando los datos clínicos se procesan en servicios de computación en la nube, ya que exigen mecanismos como cifrado de extremo a extremo, autenticación multifactor, segregación de datos y auditorías periódicas.

V-Ñ3. Normativas NFPA aplicables a infraestructura digital y centros de datos: Aunque las normas NFPA no regulan directamente la privacidad de datos, sí son críticas para la protección física de la infraestructura tecnológica donde se almacenan datos clínicos:

- **NFPA 70 (National Electrical Code):** Regula instalaciones eléctricas seguras en centros de datos y salas de servidores, reduciendo el riesgo de fallos eléctricos que puedan comprometer la disponibilidad de información médica [59].

- **NFPA 75:** Establece estándares de protección contra incendios para equipos electrónicos y centros de procesamiento de datos, fundamentales para la continuidad de servicios clínicos digitales [60].

- **NFPA 99 (Health Care Facilities Code):** Es especialmente relevante en entornos hospitalarios, debido a que, regula sistemas eléctricos y tecnológicos que soportan equipos médicos y sistemas de información en salud [61].

Estas normas son clave cuando se utilizan servidores locales (on-premise) o infraestructuras híbridas que alojan plataformas de diagnóstico asistido por inteligencia artificial.

V-Ñ4. Aplicación al entrenamiento de algoritmos de IA clínica: En proyectos de clasificación de imágenes médicas mediante aprendizaje profundo, como la detección de retinopatía diabética, el cumplimiento normativo implica:

- Uso de datos anonimizados o seudonimizados durante el entrenamiento del modelo.

- Implementación de controles de acceso y registros de auditoría sobre datasets clínicos.

- Almacenamiento seguro en infraestructura certificada bajo ISO/IEC 27001.

- Respaldo y recuperación ante desastres conforme a NFPA 75 y ISO 22301.

- Cumplimiento de la LOPDP ecuatoriana respecto al consentimiento informado y derechos del titular de los datos.

V-O. Regulación de normas y protección de datos clínicos aplicado en la Plataforma Retina Vision AI

La plataforma Retina Vision AI, orientada al diagnóstico asistido de retinopatía diabética mediante inteligencia artificial, se desarrolla bajo un enfoque legal y ético que prioriza la protección de datos clínicos, el respeto a la privacidad de los pacientes y el uso responsable de tecnologías digitales avanzadas.

Desde el marco legal ecuatoriano, el tratamiento de imágenes médicas y datos asociados se rige por la Ley Orgánica de Protección de Datos Personales (LOPDP), la cual clasifica la información de salud como dato sensible, exigiendo consentimiento informado, finalidad específica y la implementación de medidas técnicas que garanticen la confidencialidad, integridad y disponibilidad de la información. Asimismo, las disposiciones de la ARCSA establecen que los sistemas digitales utilizados con fines diagnósticos deben contar con mecanismos de seguridad informática, control de accesos y trazabilidad del uso de la información clínica [62].

En el ámbito internacional, RetinaVision AI adopta los lineamientos de los estándares ISO/IEC 27001 e ISO 27799, los cuales permiten estructurar un sistema de gestión de seguridad de la información enfocado en datos de salud, especialmente en entornos de computación en la nube. Estos estándares respaldan la implementación de cifrado de datos, autenticación multifactor, segmentación de bases de datos y auditorías de seguridad, reduciendo el riesgo de accesos no autorizados o pérdidas de información [57].

Desde una perspectiva ética, la plataforma se alinea con los principios de beneficencia, no maleficencia, justicia y autonomía, a través de normativas internacionales de seguridad y privacidad, sintetizadas en la Tabla VII, garantizando que los algoritmos de aprendizaje profundo sean entrenados exclusivamente con datos anonimizados o seudonimizados, evitando cualquier forma de discriminación algorítmica o uso indebido de la información. La adopción del enfoque privacy by design, respaldado por la ISO/IEC 27701, asegura que la protección de la privacidad sea integrada desde la etapa de diseño del sistema y no como un elemento

posterior [63].

Tabla VII
NORMATIVAS INTERNACIONALES APLICABLES A LA SEGURIDAD, PRIVACIDAD Y GESTIÓN DE DATOS CLÍNICOS EN RETINA VISION AI. FUENTE: AUTOR PROPIO.

Normativas de Seguridad y Privacidad en Sistemas de IA Médica	
ISO/IEC 27001	**Entidad:** International Organization for Standardization. **Tipo:** Estándar internacional. **Ámbito de aplicación:** Seguridad de la información. **Relación con datos clínicos y digitales:** Establece un Sistema de Gestión de Seguridad de la Información (SGSI), que protege la confidencialidad, integridad y disponibilidad de la información, incluyendo los datos clínicos digitales y los datos de salud almacenados en servidores locales o en la nube. **Aplicación directa en Retina Vision AI:** Control de accesos, cifrado de datasets de retinografías, gestión de incidentes de seguridad y respaldo seguro de la información clínica.
ISO/IEC 27002	**Entidad:** International Organization for Standardization. **Tipo:** Estándar internacional. **Ámbito de aplicación:** Controles de seguridad para la información clínica. **Relación con datos clínicos y digitales:** Define controles técnicos y organizativos que protegen la información personal y sensible dentro de sistemas informáticos. **Aplicación directa en Retina Vision AI:** Implementa políticas de acceso, mecanismos de autenticación y monitorea el uso de los datos clínicos por parte de usuarios autorizados.
ISO 27799	**Entidad:** International Organization for Standardization. **Tipo:** Estándar internacional. **Ámbito de aplicación:** Seguridad de la información en salud. **Relación con datos clínicos y digitales:** Proporciona requisitos de seguridad específicos para sistemas de información sanitaria y el manejo de imágenes médicas. **Aplicación directa en Retina Vision AI:** Protección de imágenes de fondo de ojo y de los metadatos clínicos utilizados durante el entrenamiento, validación y despliegue de los modelos de inteligencia artificial.
ISO/IEC 27701	**Entidad:** International Organization for Standardization. **Tipo:** Estándar internacional. **Ámbito de aplicación:** Gestión de privacidad de información clínica. **Relación con datos clínicos y digitales:** Extiende la norma ISO/IEC 27001 y ISO/IEC 27002 para establecer un Sistema de Gestión de Información de Privacidad. **Aplicación directa en Retina Vision AI:** Implementa los principios de privacy by design y privacy by default en la plataforma web, garantizando la protección de la información del paciente desde el diseño del sistema.

Adicionalmente, las normas NFPA 70, 75 y 99 complementan el marco ético y legal al garantizar la seguridad física y la continuidad operativa de la infraestructura tecnológica que soporta Retina Vision AI, evitando interrupciones que puedan afectar la disponibilidad del servicio o la integridad de los datos clínicos [64].

En conjunto, este marco legal y ético fortalece la confianza en Retina Vision AI como una solución tecnológica responsable, segura y alineada con las regulaciones nacionales e internacionales vigentes, promoviendo un uso ético de la inteligencia artificial en el ámbito de la salud.

VI. Marco Metodológico

El presente estudio adopta un enfoque cuantitativo con un diseño experimental comparativo, con el objetivo de evaluar de manera sistemática, controlada y reproducible el desempeño de distintas arquitecturas de aprendizaje profundo adaptadas para la clasificación multietapa de la Retinopatía Diabética (RD) mediante imágenes de fondo de ojo.

La estrategia metodológica se basa en la comparación directa de modelos bajo un entorno experimental homogéneo, en el cual se mantienen constantes el conjunto de datos, el esquema de partición, los criterios de entrenamiento y las métricas de evaluación. Este control permite aislar el impacto estructural de cada arquitectura y atribuir las diferencias de rendimiento exclusivamente a sus mecanismos internos de representación y procesamiento de características visuales.

El análisis se organiza en torno a dos categorías principales de modelos. Por un lado, se presenta un modelo CNN personalizado basado en EfficientNet-B0, el cual se modificó incorporándole capas convolucionales adicionales y un Modulo de External Attention, con la finalidad de que optimice su capacidad para prender representaciones especificas presentes en las retinografías. Por otro lado, se seleccionaron cuatro arquitecturas preentrenadas de vanguardia, reconocidas por ser clasificadores de imágenes eficientes: ResNet-50, DenseNet-121, YOLOv8-cls y Vision Transformer (ViT-B/16).

Dentro del grupo de redes neuronales convolucionales (CNN) preentrenadas se seleccionaron ResNet-50 y DenseNet-121, arquitecturas conocidas por presentar estrategias estructurales complementarias que proporcionan el análisis de distintos mecanismos de extracción jerárquica de características:

Dentro del grupo de redes neuronales convolucionales (CNN) preentrenadas se seleccionan ResNet-50 y DenseNet-121. Estas arquitecturas presentan estrategias estructurales complementarias que permiten analizar distintos mecanismos de extracción jerárquica de características:

- **ResNet-50:** Incorpora conexiones residuales que facilitan el entrenamiento de redes profundas al mitigar la degradación del gradiente, favoreciendo la estabilidad en la convergencia.

- **DenseNet-121:** Establece conexiones densas entre capas consecutivas, promoviendo la reutilización de características y mejorando el flujo de información a lo largo de la red.

La inclusión de estas arquitecturas permite examinar diferentes estrategias convolucionales para la identificación de patrones asociados a microaneurismas, hemorragias y exudados presentes en retinografías.

Se incorpora adicionalmente YOLOv8-cls, variante de clasificación perteneciente a la familia YOLOv8. Aunque su concepción original se vincula a la detección de objetos, su versión de clasificación integra optimizaciones estructurales orientadas a eficiencia en inferencia y reducción de parámetros. Su evaluación dentro del estudio permite analizar su comportamiento en tareas de clasificación médica y valorar su aplicabilidad en escenarios donde el equilibrio entre precisión diagnóstica y costo computacional constituye un factor determinante.

Asimismo, se incluye Vision Transformer (ViT-B/16), arquitectura basada en mecanismos de autoatención multicabezal. A diferencia de las CNN, que extraen características mediante convoluciones locales progresivas, ViT segmenta la imagen en parches de tamaño fijo y modela relaciones globales desde las primeras capas del modelo. Este enfoque permite capturar dependencias espaciales de largo alcance, lo cual resulta relevante en imágenes médicas donde la distribución de lesiones puede presentar patrones dispersos y heterogéneos.

La integración del modelo EfficientNet-B0 personalizado junto con las arquitecturas preentrenadas (ResNet-50, DenseNet-121, YOLOv8-cls y ViT-B/16) configura un marco comparativo amplio y técnicamente fundamentado. Esta estructura metodológica permite evaluar diferencias en capacidad de generalización, estabilidad de entrenamiento, eficiencia computacional y precisión diagnóstica, en coherencia directa con los objetivos planteados en el estudio.

VI-A. Diseño experimental: Variables y control

La investigación se estructura bajo un diseño experimental con control riguroso de variables, en el cual las arquitecturas de aprendizaje profundo constituyen la variable independiente principal, mientras que el desempeño cuantitativo obtenido en la tarea de clasificación multiclase representa la variable dependiente. La estandarización de las condiciones de entrenamiento y evaluación garantiza validez interna y permite atribuir las diferencias observadas exclusivamente a la configuración arquitectónica implementada.

VI-A1. Variables Independientes: Las variables independientes corresponden a los elementos estructurales y configuracionales asociados a los modelos evaluados:

- **Tipo de arquitectura:** Se evalúan dos categorías de modelos. Por un lado, un modelo CNN personalizado basado en EfficientNet-B0, modificado mediante la incorporación de capas convolucionales adicionales y un módulo de External Attention. Por otro lado, modelos preentrenados de referencia que incluyen: redes neuronales convolucionales (ResNet-50 y DenseNet-121), YOLOv8-cls en modalidad de clasificación y Vision Transformer (ViT-B/16).

- **Estrategia de transferencia de aprendizaje:** El modelo EfficientNet-B0 personalizado se entrena sin transferencia de aprendizaje, aprendiendo directamente las representaciones del dominio retiniano, mientras que los modelos preentrenados (ResNet-50, DenseNet-121, YOLOv8-cls y ViT-B/16) utilizan inicialización con pesos pre-entrenados en ImageNet y aplicación de ajuste fino fine-tuning en las capas superiores del modelo.

- **Configuración del clasificador final (head):** Adaptación de capas densas finales, implementación de función de activación Softmax para cinco clases diagnósticas y mecanismos adicionales cuando la arquitectura lo requiere.

VI-A2. Variables Dependientes: Las variables dependientes corresponden a las métricas de desempeño evaluadas exclusivamente sobre el conjunto de prueba, garantizando independencia respecto al proceso de entrenamiento. Estas métricas permiten medir la capacidad de generalización y discriminación multiclase de cada modelo:

- Exactitud global (Accuracy).

- Precisión por clase (Precision).

- Sensibilidad (Recall).

- Especificidad.

- Puntuación F1.

- Área bajo la curva ROC multiclase (AUC-ROC bajo el esquema One-vs-Rest).

- Matriz de confusión normalizada.

De manera complementaria, se analizan las curvas de pérdida correspondientes a entrenamiento y validación con el propósito de evaluar estabilidad, convergencia y posibles indicios de sobreajuste.

VI-A3. Variables Controladas: Con el fin de asegurar consistencia metodológica y garantizar que las diferencias observadas sean atribuibles únicamente a la arquitectura evaluada, se mantuvieron constantes las siguientes condiciones experimentales:

- Mismo conjunto de datos consolidado y misma versión preprocesada.
- Misma partición estratificada: 70 % entrenamiento, 15 % validación y 15 % prueba.
- Tamaño de entrada uniforme para todas las arquitecturas.
- Esquema de normalización y aumento de datos equivalente cuando corresponde.
- Mismo optimizador, misma tasa de aprendizaje inicial y misma estrategia de actualización de parámetros.
- Tamaño de lote constante.
- Número máximo de épocas definidas bajo el mismo criterio y aplicación de detención temprana.
- Misma semilla aleatoria para garantizar reproducibilidad.
- Entorno de ejecución homogéneo en términos de librerías, versiones y configuración de hardware.

VI-B. Origen, recopilación y unificación del dataset mediante Roboflow

El conjunto de datos empleado en la presente investigación fue construido a partir de la integración sistemática de múltiples bases de datos públicas especializadas en retinografías para la detección de Retinopatía Diabética (RD). La consolidación se realizó mediante la plataforma Roboflow, lo que permitió centralizar, estandarizar y versionar la información bajo una única estructura organizada denominada RETINOPATÍA DIABÉTICA V2, tal como se ilustra en la Figura 22.

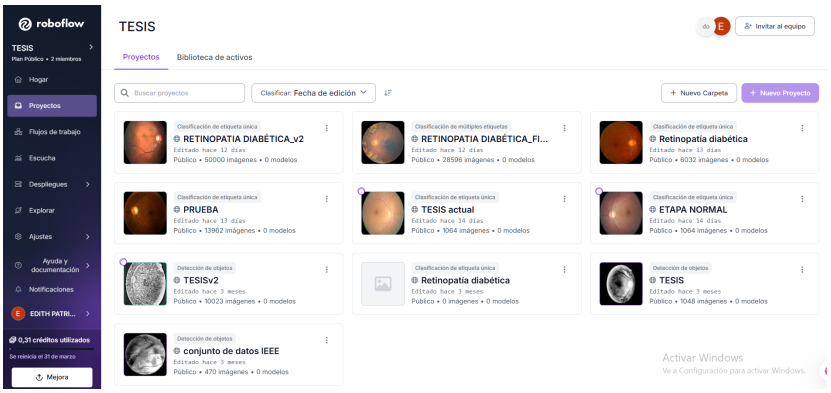


Figura 22. Interfaz de Roboflow utilizada para la gestión y organización de proyectos del dataset de retinopatía diabética. Fuente: Autor Propio.

El dataset final está conformado por un total de 50.000 imágenes de fondo de ojo, distribuidas equitativamente en cinco categorías diagnósticas correspondientes a los niveles de severidad definidos bajo el estándar ETDRS.

Las principales fuentes utilizadas para la construcción del dataset incluyen repositorios clínicos ampliamente reconocidos:

- **EyePACS:** Conjunto proveniente de programas de tamizaje clínico desarrollados en Estados Unidos, compuesto por retinografías adquiridas en campañas reales de detección oftalmológica.
- **APTOS:** Dataset recopilado mediante colaboraciones con instituciones clínicas de la región Asia-Pacífico, principalmente en India, incluyendo imágenes obtenidas en hospitales urbanos y rurales.
- **Messidor:** Proyecto clínico desarrollado en Francia, compuesto por imágenes adquiridas con cámaras oftalmológicas digitales en entornos hospitalarios especializados.

La integración de múltiples fuentes permitió incorporar variabilidad real asociada a diferencias en dispositivos de captura, resolución, iluminación, calidad de imagen y características demográficas, lo cual fortalece la capacidad de generalización de los modelos evaluados.

Roboflow fue utilizado como herramienta central dentro del pipeline de preparación de datos, permitiendo:

- Estandarizar formatos de imagen (extensión, codificación y dimensiones base).
- Unificar el esquema de etiquetado bajo cinco clases diagnósticas.
- Organizar automáticamente la estructura jerárquica compatible con frameworks de entrenamiento supervisado.
- Generar una versión documentada del dataset antes de su exportación.

Posteriormente, el dataset versionado fue exportado y almacenado en Google Drive, permitiendo su acceso directo desde el entorno de ejecución en Google Colab. Este procedimiento garantizó que todas las arquitecturas fueran entrenadas bajo exactamente la misma configuración de datos, asegurando reproducibilidad experimental y consistencia metodológica.

VI-B1. Criterios de selección: Para asegurar coherencia clínica y calidad técnica del conjunto consolidado, se aplicaron los siguientes criterios de selección y depuración:

- Inclusión exclusiva de retinografías a color con contenido clínicamente interpretable.
- Etiquetado compatible con la clasificación multiclase de cinco niveles (0-4).
- Se eliminaron imágenes borrosas, defectuosas, repetidas y aquellas que no corresponden a retinas o que tengan un formato invalidado.
- Conservación de variabilidad en iluminación y calidad con fines de robustez del modelo.

VI-B2. Estructura de clases diagnósticas: El dataset consolidado mantiene una distribución balanceada entre las cinco categorías ETDRS, con 10.000 imágenes por clase, lo que evita sesgos derivados de desbalance en

la prevalencia clínica de la enfermedad, tal como se muestra en la siguiente Tabla VIII.

Tabla VIII
PARTICIÓN GLOBAL DEL DATASET. FUENTE: AUTOR PROPIO.

Subconjunto	Número de imágenes
Entrenamiento	35.000
Validación	7.500
Prueba (Test)	7.500
Total	**50.000**

Esta estructuración permite abordar la tarea como un problema de clasificación multietapa, garantizando equilibrio estadístico y estabilidad en el proceso de entrenamiento.

VI-B3. Control de calidad, deduplicación y prevención de data leakage: Dado que el dataset fue consolidado a partir de múltiples fuentes públicas, se implementaron procedimientos técnicos para asegurar integridad y evitar sesgos experimentales:

- **Verificación de integridad:** Identificación y eliminación de archivos corruptos o no legibles.
- **Consistencia de etiquetas:** Validación de que todas las clases pertenezcan al conjunto $\{0, 1, 2, 3, 4\}$.
- **Deduplicación:** Detección y eliminación de imágenes duplicadas o casi duplicadas antes de la partición.
- **Prevención de fuga de información:** La división estratificada del dataset (70 % entrenamiento, 15 % validación y 15 % prueba) se realizó posteriormente al proceso de limpieza y consolidación, garantizando subconjuntos mutuamente excluyentes.

Este control metodológico asegura validez interna, evita contaminación entre subconjuntos y fortalece la confiabilidad de los resultados obtenidos durante la evaluación comparativa de arquitecturas.

VI-C. Preprocesamiento de imágenes

El preprocesamiento de imágenes constituye una etapa crítica dentro del pipeline experimental, debido a su influencia directa en la estabilidad del entrenamiento, la velocidad de convergencia y la calidad de las representaciones aprendidas por los modelos de aprendizaje profundo. Aunque las arquitecturas convolucionales y los modelos basados en transformadores poseen la capacidad de extraer características directamente desde datos crudos, la estandarización previa reduce variabilidad irrelevante asociada a condiciones de adquisición heterogéneas.

El pipeline implementado se basa en técnicas ampliamente utilizadas en la literatura para retinografías, particularmente en el enfoque descrito en el notebook de Kaggle “APTOS Eye Preprocessing in Diabetic Retinopathy”. El procedimiento fue adaptado y automatizado para garantizar reproducibilidad y aplicación uniforme sobre la totalidad del dataset consolidado, tal como se ilustra en la Figura 23.

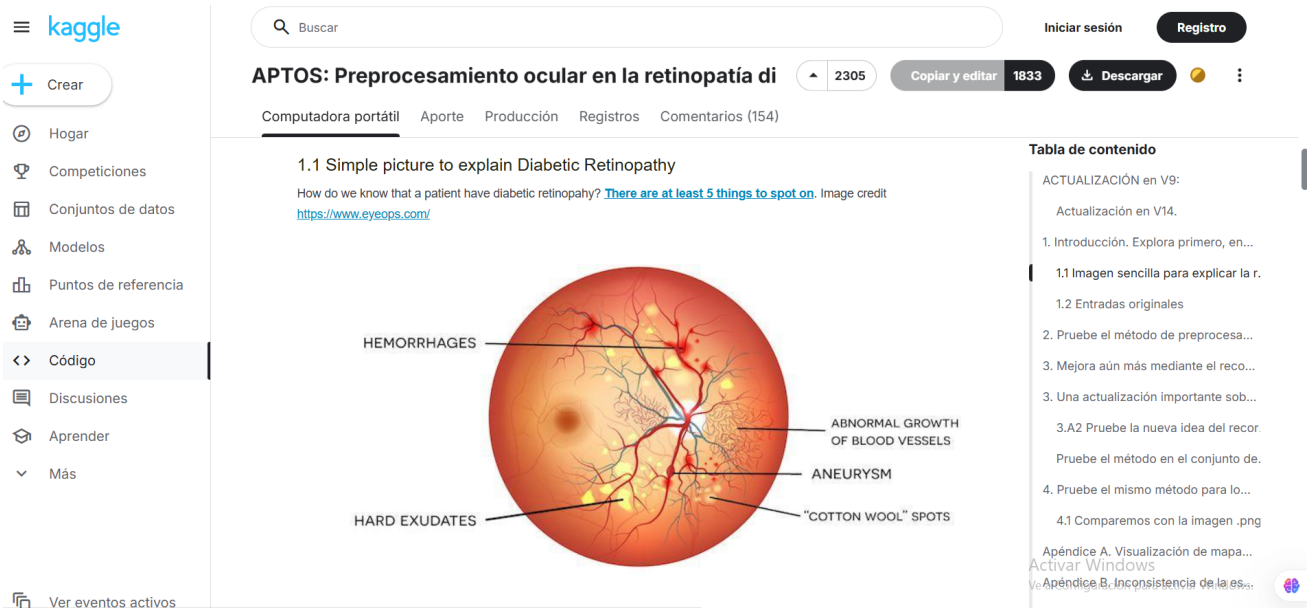


Figura 23. Interfaz del entorno Kaggle utilizada para la exploración y preprocesamiento del dataset APTOS en el estudio de retinopatía diabética. Fuente: Autor Propio.

VI-C1. Pipeline de preprocesamiento automatizado: El preprocesamiento fue implementado mediante un pipeline automatizado utilizando bibliotecas OpenCV y torchvision.transforms en PyTorch. Todas las operaciones fueron integradas dentro del flujo de carga de datos mediante DataLoaders, asegurando que cada imagen fuera transformada bajo los mismos parámetros en cada ejecución experimental. El pipeline se ejecuta de manera secuencial y determinística, evitando variaciones entre subconjuntos o clases.

VI-D. Técnicas de Preprocesamiento

VI-D1. Conversión a escala de grises para detección de región activa: Como etapa auxiliar para el recorte del campo retiniano, se empleó la conversión temporal a escala de grises mediante:
`cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)`

Esta conversión permitió generar una máscara binaria basada en intensidad: `mask = gray_img >tol`

La máscara identifica píxeles pertenecientes a la región retiniana activa, excluyendo zonas oscuras periféricas sin contenido clínico relevante.

VI-D2. Recorte automático del campo retiniano: Siguiendo el enfoque del notebook de referencia, se aplicó un umbral binario: `cv2.threshold(image, 1, 255, cv2.THRESH_BINARY)`

Posteriormente, mediante `cv2.findContours` se detectaron contornos y se calculó un `boundingRect` para recortar la región efectiva de la retina.

Este procedimiento no se utilizó para segmentación de lesiones, sino exclusivamente para eliminar fondo negro y reducir ruido periférico, concentrando el análisis en el disco retiniano.

VI-D3. Aplicación de máscara circular (Región de Interés): Se implementó la función `circle_crop()`, la cual:

- Calcula el centro geométrico de la imagen.
- Determina el radio mínimo posible dentro del encuadre.
- Genera una máscara circular mediante `cv2.circle`.
- Aplica la operación `cv2.bitwise_and` para conservar únicamente la región retiniana central.

Este procedimiento reduce artefactos periféricos y homogeniza el encuadre entre imágenes provenientes de diferentes dispositivos.

VI-D4. Realce de contraste mediante Unsharp Masking: Se aplicó la técnica de realce descrita en el notebook de Kaggle mediante:
`cv2.addWeighted(image, 4, cv2.GaussianBlur(image,(0,0),sigmaX), -4, 128)`

El procedimiento combina la imagen original con una versión suavizada por filtro Gaussiano, incrementando contraste local y resaltando estructuras como vasos sanguíneos, microaneurismas y exudados.

Esta técnica forma parte de la función `load_ben_color()` adaptada al presente estudio.

VI-D5. Estandarización geométrica: Tras el recorte y aplicación de la máscara circular, las imágenes fueron redimensionadas preservando la relación de aspecto original. Cuando fue necesario, se aplicó padding controlado para mantener proporciones sin distorsión geométrica.

La dimensión final establecida fue de 224×224 píxeles, tamaño que garantiza compatibilidad con las arquitecturas EfficientNet-B0, ResNet-50, DenseNet-121, YOLOv8-cls y Vision Transformer (ViT-B/16).

VI-D6. Redimensionamiento a tamaño estándar: El redimensionamiento se realizó mediante interpolación bilineal, equilibrando calidad visual y eficiencia computacional.

Aunque las imágenes originales poseen resoluciones superiores, múltiples estudios han demostrado que 224×224 píxeles es suficiente para capturar patrones discriminativos relevantes en clasificación de retinopatía diabética.

VI-D7. Normalización de color e intensidad: Para mejorar uniformidad inter-dataset, se aplicó CLAHE (Contrast Limited Adaptive Histogram Equalization) sobre el canal de luminancia en el espacio de color LAB.

Este procedimiento:

- Incrementa contraste local.
- Reduce variabilidad de iluminación.
- Evita saturación mediante limitación adaptativa.

Posteriormente, se aplicó normalización específica según el modelo:

Para el modelo EfficientNet-B0 personalizado: Escalamiento a rango [0,1] y normalización usando media y desviación estándar calculadas sobre el conjunto de entrenamiento.

Para los modelos con transferencia de aprendizaje (ResNet-50, DenseNet-121, YOLOv8-cls y ViT-B/16):

Se emplearon las estadísticas estándar de ImageNet, utilizando como media $[0{,}485,\ 0{,}456,\ 0{,}406]$ y como desviación estándar $[0{,}229,\ 0{,}224,\ 0{,}225]$.

VI-D8. Aplicación uniforme al dataset: El pipeline fue aplicado automáticamente sobre todas las clases (0–4) y subconjuntos (train, validation, test), replicando exactamente la misma estructura de carpetas. Cada imagen fue procesada bajo parámetros idénticos, garantizando consistencia metodológica y reproducibilidad experimental.

VI-E. División del conjunto de datos

El dataset consolidado de 50.000 imágenes fue dividido en tres subconjuntos mutuamente excluyentes mediante partición estratificada:

- **Entrenamiento 70 %:** 35.000 imágenes.
- **Validación 15 %:** 7.500 imágenes.
- **Prueba 15 %:** 7.500 imágenes.

La estratificación aseguró proporción equivalente de las cinco clases en cada subconjunto. La división se realizó una única vez al inicio del estudio y se mantuvo constante para todas las arquitecturas evaluadas, garantizando comparación justa y evitando fuga de información.

VI-F. Configuración experimental y entrenamiento

El proceso de entrenamiento se diseñó para mantener consistencia entre todos los modelos evaluados. El dataset de 50,000 imágenes se distribuyó en tres conjuntos: 70 % para entrenar los modelos, 15 % para validar su desempeño durante el entrenamiento, y 15 % para la evaluación final. Todas las arquitecturas recibieron imágenes del mismo tamaño y utilizaron parámetros de entrenamiento idénticos: 100 épocas completas, procesamiento en lotes de 32 imágenes, y el algoritmo de optimización Adam. Durante cada época, el modelo aprendió de los datos de entrenamiento mediante un proceso iterativo de predicción, cálculo de error y ajuste de parámetros. Al finalizar cada época, se evaluó el modelo con datos de validación, seleccionando la versión con mejor rendimiento para las pruebas finales.

VI-F1. Parámetros de entrenamiento: La configuración de entrenamiento se definió de forma homogénea para asegurar comparabilidad:

- **Número de clases:** 5 (grados 0 a 4).
- **Tamaño de entrada para entrenamiento:** 224×224 píxeles (RGB), compatible con modelos preentrenados.
- **Batch size:** 32.
- **Número de épocas:** 100.
- **Optimizador:** Adam, $\text{lr} = 1 \times 10^{-4}$.

- **Función de pérdida:** CrossEntropyLoss.
- **Regularización:** Dropout en heads y/o módulos adicionales según arquitectura.
- **Reproducibilidad:** Fijación de semillas (torch/numpy/random) y configuración determinista cuando aplique.

VI-F2. Procedimiento de entrenamiento: En cada época de entrenamiento, se iteró sobre todos los batches del conjunto de entrenamiento. Para cada batch:

1. Las imágenes y etiquetas fueron cargadas y movidas a la GPU.
2. Se realizó un forward pass computando las predicciones del modelo.
3. Se calculó la pérdida comparando las predicciones con las etiquetas verdaderas.
4. Se realizó backward propagation para calcular los gradientes.
5. Se actualizaron los pesos usando el optimizador.
6. Se acumuló la pérdida total de la época.

Al finalizar cada época de entrenamiento, se evaluó el modelo en el conjunto de validación para monitorear el progreso y detectar overfitting. Si el validation accuracy mejoraba, se guardaba una copia de los pesos del modelo como best model.

VI-G. Procedimientos Experimentales

El procedimiento experimental se desarrolló de manera estructurada y reproducible, siguiendo las siguientes etapas:

1. Obtención y exploración del conjunto de datos.
2. Aplicación del pipeline de preprocesamiento y división estratificada.
3. Implementación del modelo EfficientNet-B0 personalizado, con incorporación de capas convolucionales adicionales y módulo de *External Attention*.
4. Adaptación de modelos preentrenados (ResNet-50, DenseNet-121, YOLOv8-cls y ViT-B/16) mediante transferencia de aprendizaje.
5. Entrenamiento de todas las arquitecturas bajo condiciones experimentales homogéneas.
6. Evaluación final sobre el conjunto de prueba independiente.
7. Comparación sistemática del desempeño entre el modelo personalizado y los modelos preentrenados.

VI-H. Flujo metodológico del estudio

Con el objetivo de garantizar coherencia y reproducibilidad experimental, el desarrollo del estudio se estructuró en una secuencia organizada de etapas que abarcan desde la preparación del conjunto de datos hasta la evaluación final de los modelos.

El proceso inició con la obtención y exploración del dataset, seguido de la implementación del pipeline de preprocesamiento automatizado. Posteriormente, se realizó la división estratificada en conjuntos de entrenamiento, validación y prueba. Una vez definidos los subconjuntos, se procedió al entrenamiento de los modelos bajo condiciones experimentales homogéneas y finalmente a la evaluación sobre el conjunto de prueba independiente.

VI-I. Descripción metodológica por arquitectura

Las cinco arquitecturas evaluadas fueron implementadas bajo un esquema experimental homogéneo en cuanto a tamaño de entrada, optimización y criterios de evaluación. Las diferencias entre modelos se centran exclusivamente en su estructura interna de extracción de características, tipo de conectividad, mecanismo de atención y estrategia de aprendizaje.

A nivel general:

- **Modelo personalizado:** EfficientNet-B0, modificado mediante la incorporación de capas convolucionales adicionales y un módulo de External Attention, fue entrenado sin transferencia de aprendizaje, aprendiendo directamente las representaciones del dominio retiniano.
- **Modelos preentrenados:** ResNet-50, DenseNet-121, YOLOv8-cls y ViT-B/16 utilizaron pesos preentrenados en ImageNet con ajuste fino fine-tuning.
- Se incorporó un módulo de External Attention en las arquitecturas basadas en backbone profundo (EfficientNet-B0 personalizado, ResNet-50 y DenseNet-121), así como en ViT-B/16 antes de la clasificación final.
- ViT-B/16 es el único modelo basado en Transformer con autoatención global nativa, complementada con el módulo de External Attention añadido.
- YOLOv8-cls emplea una arquitectura originalmente diseñada para detección, adaptada a clasificación global.

VI-I1. Justificación de la personalización de EfficientNet-B0: Fue seleccionado como arquitectura base para el modelo personalizado debido a su escalamiento compuesto, que equilibra profundidad, ancho y resolución, resultando adecuado para imágenes médicas con estructuras anatómicas de diferentes escalas. La personalización implementada busca adaptar esta arquitectura genérica a las particularidades del dominio retiniano mediante las siguientes modificaciones:

- **Arquitectura híbrida:** Incorporación de 10 capas convolucionales adicionales después del backbone, con Batch Normalization y activación ReLU, siguiendo la progresión dimensional: 640→512→448→384→320 →256. Este bloque de refinamiento permite aprender representaciones de alto nivel específicas para características sutiles de la retinopatía diabética (microaneurismas, hemorragias y exudados).
- **Mecanismo de atención:** Adición de un módulo de External Attention con 8 cabezas y dimensión 32, diseñado para ponderar regiones relevantes como el área macular, vasos sanguíneos y lesiones antes de la

clasificación final.

- **Estrategia de aprendizaje:** Entrenamiento sin transferencia de aprendizaje, aprendiendo directamente las representaciones visuales a partir de las 50,000 retinografías del dataset, a diferencia de los modelos preentrenados evaluados.

- **Head de clasificación:** Capa Fully Connected (256→512) con función Softmax para clasificación en cinco clases diagnósticas, manteniendo uniformidad estructural con los demás modelos para garantizar comparabilidad.

Esta personalización transforma el modelo base en una herramienta especializada para detección de retinopatía diabética, incorporando elementos de interpretabilidad, modularidad y trazabilidad esenciales en aplicaciones médicas.

VI-I2. Resumen estructural individual: En esta sección se sintetizan las características estructurales principales de cada modelo evaluado, destacando su tipo de backbone, estrategia de conectividad y mecanismo de atención. Este resumen permite identificar de manera concisa las diferencias fundamentales entre modelos, facilitando la comparación técnica bajo un mismo protocolo experimental.

EfficientNet-B0, ResNet-50 y DenseNet-121:

Se implementaron tres arquitecturas preentrenadas como modelos base: EfficientNet-B0, ResNet-50 y DenseNet-121, utilizadas como backbones para la extracción de características profundas a partir de las imágenes de entrada. Aunque cada arquitectura presenta diferencias estructurales internas escalado compuesto en EfficientNet-B0, conexiones residuales en ResNet-50 y conectividad densa en DenseNet-121, todas comparten el mismo esquema de procesamiento posterior definido en el sistema.

Después de la extracción inicial de características mediante el backbone seleccionado, se incorporó un bloque de refinamiento compuesto por capas convolucionales adicionales con Batch Normalization y función de activación ReLU. Posteriormente, se aplicó una capa de Global Average Pooling para la reducción espacial de las características.

A continuación, se integró un módulo de External Attention (8 cabezas, dimensión 32) con el objetivo de mejorar la representación discriminativa antes de la etapa final de clasificación.

Finalmente, el modelo culmina con una capa Fully Connected y una función Softmax para la clasificación en 5 clases. La Figura 24 muestra el esquema unificado que resume el flujo completo de procesamiento aplicado a los tres modelos.

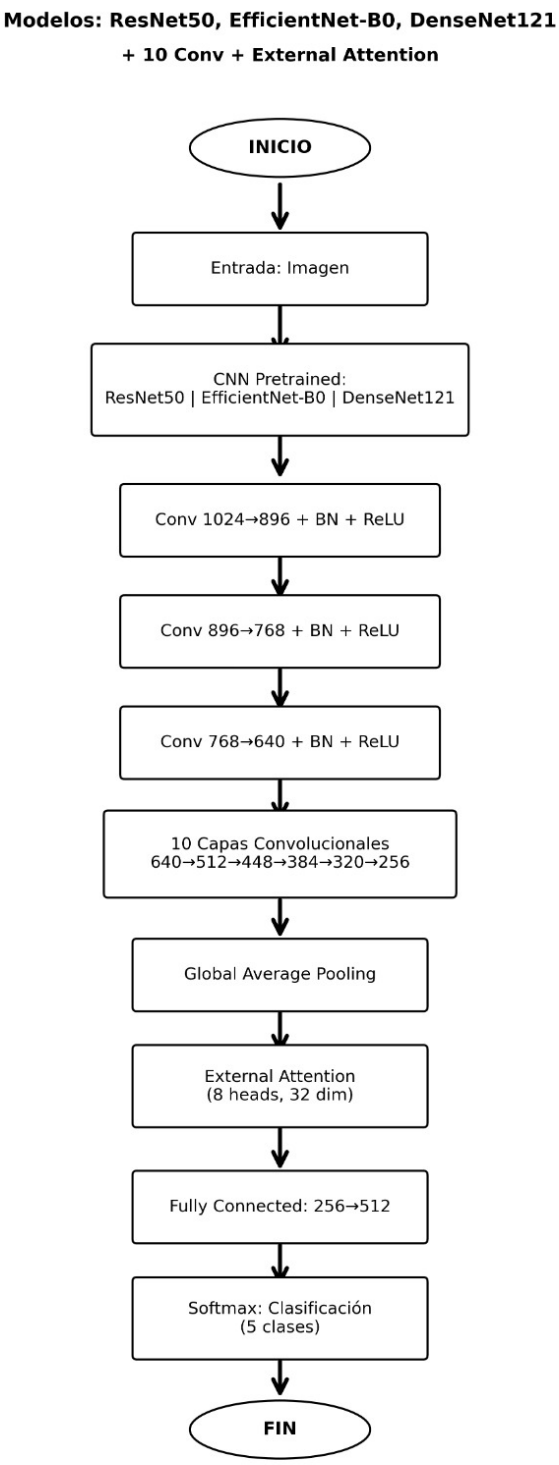


Figura 24. Esquema general de la arquitectura implementada para los modelos ResNet-50, EfficientNet-B0 y DenseNet-121, incluyendo el bloque de refinamiento convolucional y el módulo de External Attention previo a la clasificación. Fuente: Autor Propio.

YOLOv8-cls: Modelo basado en el backbone CSPDarknet de YOLOv8, adaptado a modalidad de clasificación directa de imagen. La Figura 25 muestra el flujo de procesamiento implementado. La arquitectura inicia con la

extracción de características mediante el backbone CSPDarknet y bloques C2f, seguida por un módulo Spatial Pyramid Pooling Fast (SPPF) para la integración de información contextual. Posteriormente, se aplica Global Average Pooling para la reducción espacial de características y una cabeza de clasificación Fully Connected que proyecta la representación final hacia cinco clases mediante una función Softmax.

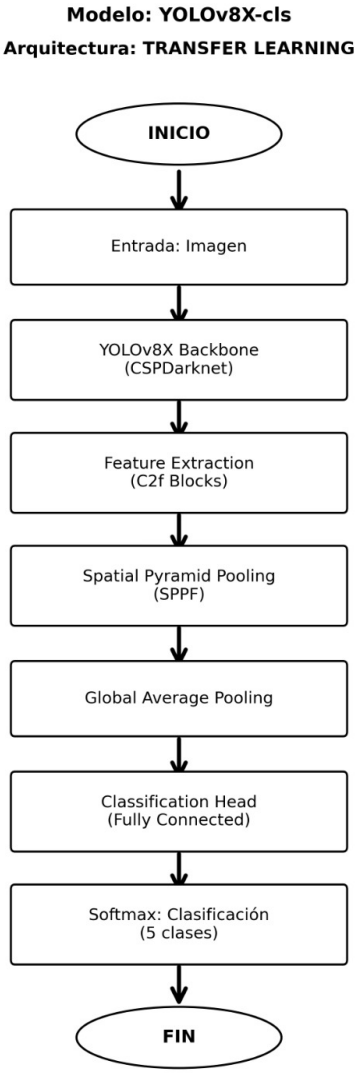


Figura 25. Esquema funcional del modelo YOLOv8-cls en modalidad de clasificación global de imagen. Fuente: Autor Propio.

ViT-B/16: Modelo basado en arquitectura Transformer que procesa la imagen mediante partición en parches 16×16 y codificación mediante autoatención multi-cabeza. La Figura 26 presenta el flujo completo implementado. Tras la etapa de Vision Transformer B/16 preentrenado y el paso por el Transformer Encoder (Self-Attention), se incorpora un bloque de refinamiento compuesto por una capa convolucional (768→640) con Batch Normalization y activación GELU. Posteriormente, se añaden diez capas convolucionales adicionales para la adaptación de características, seguidas de Global Average Pooling. Antes de la clasificación

final, se integra un módulo de External Attention (8 cabezas, dimensión 32) para reforzar la representación discriminativa. Finalmente, una capa Fully Connected proyecta la salida hacia cinco clases mediante función Softmax.

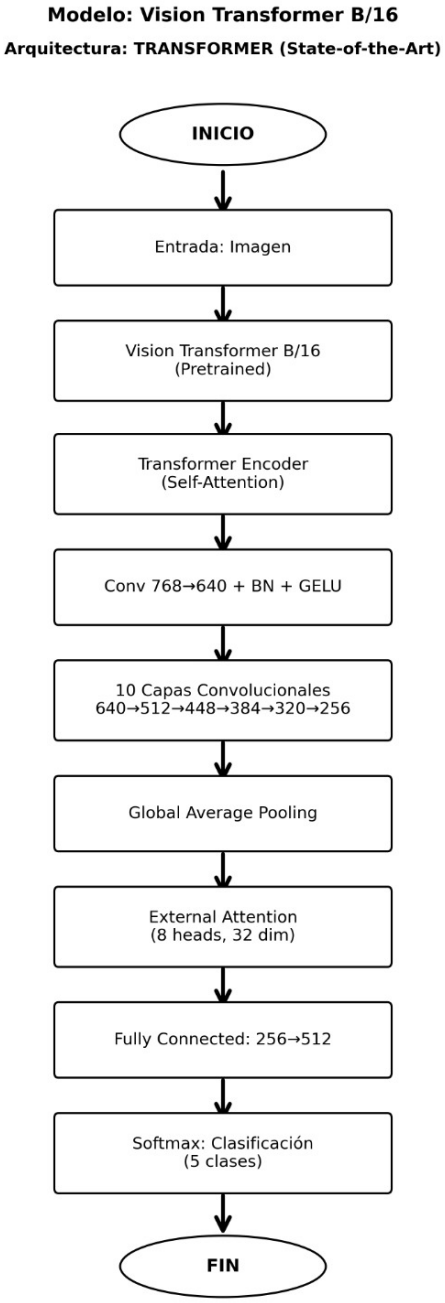


Figura 26. Esquema funcional del modelo ViT-B/16, incluyendo etapa Transformer, bloque de refinamiento convolucional y módulo de External Attention previo a la clasificación. Fuente: Autor Propio.

VI-J. Flujo de preprocesamiento y entrenamiento

La Figura 27 muestra la secuencia aplicada para preparar los datos y entrenar los modelos. El flujo inicia con la selección de 50,000 imágenes y continúa con un preprocesamiento estandarizado que incluye giro, redimensionamiento a 224×224 y normalización utilizando las estadísticas (mean y std) de ImageNet. Luego, se verifica que el dataset se encuentre limpio y balanceado como control de calidad previo a la partición estratificada en entrenamiento, validación y prueba (70/15/15). Finalmente, se entrena el conjunto de cinco arquitecturas consideradas bajo parámetros homogéneos (100 épocas, batch 32) y se evalúa el desempeño en el test set mediante Accuracy, Precision, Recall, F1 y AUC.

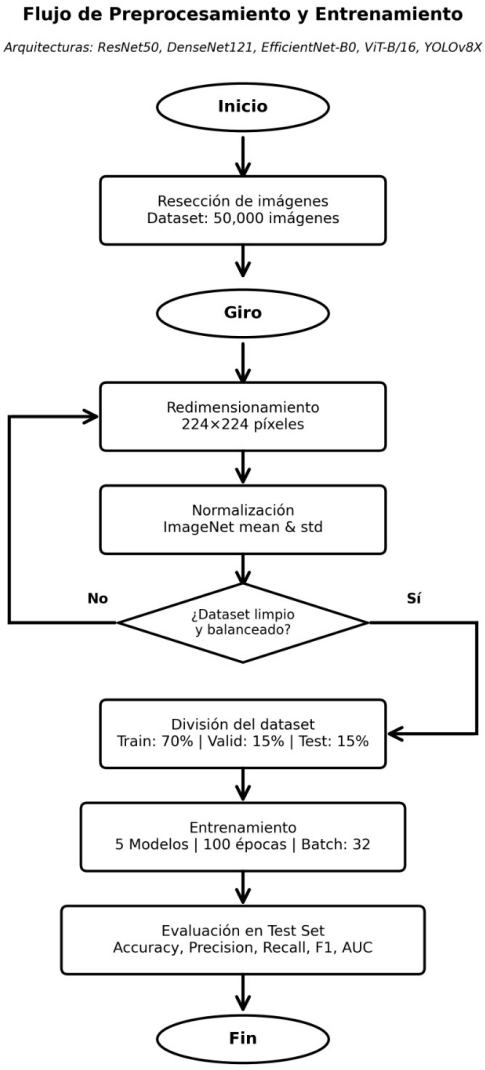


Figura 27. Flujo de preprocesamiento y entrenamiento aplicado para la comparación de arquitecturas en la clasificación multietapa de retinopatía diabética. Fuente: Autor Propio.

VI-J1. Diferencias estructurales entre modelos: Las cinco arquitecturas implementadas presentan enfoques sustancialmente diferentes en su mecanismo de extracción y procesamiento de características visuales. La Tabla IX sintetiza las principales diferencias arquitectónicas entre los modelos evaluados.

Tabla IX
COMPARACIÓN DE CARACTERÍSTICAS ESTRUCTURALES POR ARQUITECTURA. FUENTE: AUTOR PROPIO.

Característica	**EfficientNet-B0 (Personalizado)**	**ResNet-50**	**DenseNet-121**	**YOLOv8-cls**	**ViT-B/16**
Backbone	CNN escalada	Residual	Conectividad densa	CSPDarknet	Transformer
Pre-entrenamiento	No	ImageNet	ImageNet	ImageNet	ImageNet
Capas conv. adicionales	10 capas	10 capas	10 capas	No	10 capas
Mecanismo de atención	External Attention	External Attention	External Attention	No	Self-Attention + External Attention
Origen arquitectónico	Clasificación	Clasificación	Clasificación	Detección	Clasificación

El modelo EfficientNet-B0 personalizado constituye el único modelo entrenado sin transferencia de aprendizaje, aprendiendo exclusivamente a partir de las 50,000 retinografías del dataset. Su estructura base de escalamiento compuesto se complementa con diez capas convolucionales adicionales y un módulo de External Attention, diseñado para enfatizar regiones clínicamente relevantes.

ResNet-50 y DenseNet-121 representan dos enfoques clásicos de CNN profunda: el primero facilita la propagación del gradiente mediante conexiones residuales, mientras que el segundo maximiza la reutilización de características a través de conectividad densa. Ambos parten de pesos pre-entrenados en ImageNet y se complementan con el mismo bloque de refinamiento convolucional y External Attention añadido.

YOLOv8-cls constituye un caso particular al derivar de una arquitectura originalmente concebida para detección de objetos (CSPDarknet con fusión multi-escala), adaptada aquí para clasificación global. A diferencia del resto, no incorpora External Attention debido a las características inherentes de su diseño.

ViT-B/16 representa el paradigma basado en Transformer, procesando la imagen mediante parches y modelando dependencias globales a través de autoatención multi-cabeza. Sobre esta base se añadieron diez capas convolucionales y External Attention, combinando así mecanismos de atención global y local.

Esta diversidad estructural permite analizar comparativamente cinco paradigmas de aprendizaje profundo bajo condiciones experimentales idénticas, aislando el impacto de cada arquitectura en la tarea de clasificación multietapa de retinopatía diabética.

VI-K. Estrategia de evaluación y métricas de desempeño

La evaluación se realizó en dos fases:

VI-K1. Seguimiento durante entrenamiento: Durante cada época se registraron:

- Loss (entrenamiento y validación).
- Accuracy (entrenamiento y validación).

El mejor modelo fue seleccionado según la mayor exactitud en validación.

VI-K2. Evaluación final en conjunto de prueba: El conjunto de prueba independiente se utilizó exclusivamente para la evaluación final, calculando:

- Matriz de confusión multiclase.
- Precisión por clase.
- Sensibilidad (Recall) por clase.
- F1-score por clase.
- Especificidad por clase.
- AUC-ROC multiclase bajo esquema One-vs-Rest (OvR).

Las métricas fueron calculadas de manera homogénea para todos los modelos, garantizando una comparación objetiva del desempeño.

VI-L. Implementación de plataforma web Retina Vision IA

Se diseña, desarrolla y despliega el sitio web interactivo RetinaVisionAI, el cual integra de manera centralizada los modelos entrenados para la clasificación de retinopatía diabética bajo diferentes metodologías y arquitecturas. Esta fase constituye el puente entre la tecnología del backend y la experiencia del usuario, brindando una plataforma intuitiva, accesible y confiable donde los usuarios pueden cargar imágenes oftalmológicas y recibir un análisis automático basado en inteligencia artificial, tal como lo ilustra la Figura 28.

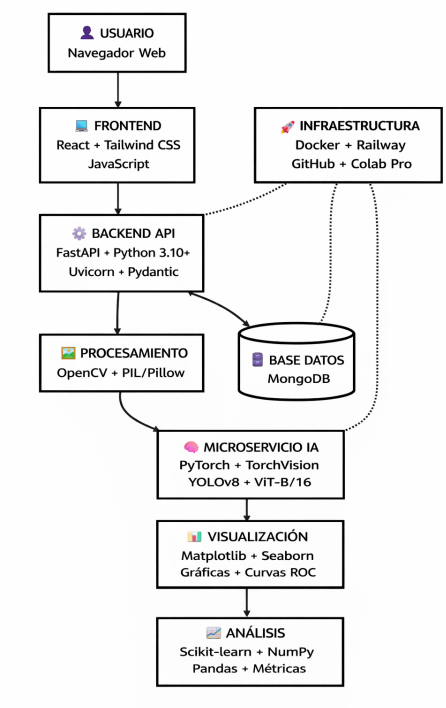


Figura 28. Ilustración de la arquitectura de la Plataforma Web RetinaVision AI para la detección automatizada de retinopatía diabética (RD). Fuente: Autor Propio.

VI-M. Componentes de la plataforma web

VI-M1. Backend (API REST): El backend del sistema es el cerebro de la aplicación, que actúa como el núcleo lógico responsable de comunicar el frontend y los modelos de inteligencia artificial y gestionar toda la lógica del sistema, está compuesto por una arquitectura de servicios REST, que emplea frameworks modernos como FatAPI o Flask, y que contribuye al desarrollo de aplicaciones robustas, escalables, seguras y de alto rendimiento.

Endpoints principales

- **POST /predict:** Es el endpoint con mayor relevancia sobre el análisis automatizado de la base de datos, dado que recibe la imagen enviada por el usuario, para después procesar la información empleando el modelo de inteligencia artificial y devolver el resultado del análisis, siendo la respuesta muestra el nivel de severidad de RD detectado en conjunto con la probabilidad asociadas a cada clase.

- **GET /models:** Permite consultar que modelos de inteligencia artificial están disponibles en el sistema, y su información básica como el nombre, versión y estado, facilitando la transparencia y control sobre los algoritmos utilizados.

- **GET /health:** Comprueba el estado operativo del servicio, lo que asegura que la API, los modelos cargados en la página y los recursos del sistema se encuentren disponibles y funcionen correctamente.

Funcionalidades del Backend

- El backend carga los modelos de inteligencia artificial, por lo que, al momento de iniciar el sistema, las solicitudes enviadas por los usuarios pasan a mantenerse disponible en la memoria, lo que conlleva a que durante cada consulta el modelo se encuentre preparado para usarse, esto promueve una respuesta más rápida, descartando la necesidad de aplicar recargas innecesarias en cada solicitud.

- El sistema prepara las imágenes que sube el usuario, mediante el ajuste de su tamaño, valores y su formato, con el propósito de que la información sea compatible con la arquitectura de los modelos.

- El sistema genera predicciones probabilísticas para cada etapa diagnóstica, gestiona los errores y valida los datos obtenidos, garantizando que el proceso sea integral y la estabilidad del servicio ante solicitudes incorrectas o incompletas por parte de los usuarios.

VI-M2. Frontend:

- El frontend de la plataforma se desarrolla utilizando tecnologías web estándar como HTML, CSS y JavaScript, integradas en una interfaz moderna, intuitiva y completamente responsiva. Este componente actúa como el punto de interacción directa entre el usuario y el sistema.

- El sistema incorpora un formulario de carga de imágenes con funcionalidad drag-and-drop, que facilita la selección de archivos y mejora la experiencia de uso, especialmente para usuarios sin conocimientos técnicos.

- La interfaz permite la visualización inmediata de la imagen cargada, así como la presentación clara de los resultados del análisis, incluyendo la interpretación del diagnóstico y gráficos de probabilidades por clase que facilitan la comprensión de los resultados obtenidos.

- El frontend incluye un disclaimer de uso académico, por lo que, informa al usuario que los resultados generados tienen fines investigativos y no sustituyen el criterio clínico profesional.

Consideraciones de implementación

- El sistema implementa mecanismos de validación que verifican el formato y tamaño de los archivos subidos, asegurando que únicamente imágenes compatibles sean procesadas por la plataforma.

- Se incorpora rate limiting a nivel de backend para prevenir el abuso del servicio y garantizar un acceso equitativo a los recursos, especialmente en escenarios con múltiples usuarios concurrentes.

- Por motivos de privacidad y ética, la plataforma no almacena permanentemente las imágenes subidas por los usuarios, garantizando la protección de datos sensibles y el cumplimiento de buenas prácticas en el manejo de información médica.

- El sistema incluye documentación de uso accesible y detallada, orientada a explicar el funcionamiento de la plataforma, el flujo de análisis y la correcta interpretación de los resultados, facilitando su utilización en contextos académicos y de investigación.

VI-M3. Técnicas y Herramientas: La implementación de este proyecto requirió el uso de un stack tecnológico robusto y bien establecido en el campo del Deep Learning. La selección de cada herramienta fue cuidadosamente considerada basándose en criterios de madurez tecnológica, soporte de la comunidad, documentación disponible, y adecuación para las necesidades específicas del proyecto.

VI-M4. Lenguaje de Programación: Python 3.10+ fue utilizado como lenguaje principal del proyecto debido a su amplia adopción en el campo de la inteligencia artificial y la ciencia de datos. Este lenguaje permitió integrar de manera eficiente los modelos de Deep Learning con el backend del sistema web desarrollado en FastAPI. Su sintaxis clara y legible facilitó la implementación, depuración y mantenimiento del código, así como la interoperabilidad con librerías especializadas para el procesamiento de imágenes, la evaluación de modelos y el despliegue de servicios de inferencia. Además, Python es el lenguaje dominante en entornos académicos y de investigación, lo que favorece la reproducibilidad del proyecto, al contar con el ecosistema más maduro de librerías optimizadas para computación numérica y científica, siendo el lenguaje para el cual la mayoría de los frameworks de Deep Learning se encuentran específicamente diseñados y optimizados.

VI-N. Frameworks de Deep Learning

VI-N1. PyTorch 2.0+: PyTorch fue empleado como framework principal para el desarrollo y entrenamiento de los modelos de Deep Learning utilizados en la clasificación de la retinopatía diabética. Su modelo de ejecución dinámica (eager execution) permitió depurar de forma intuitiva las arquitecturas implementadas y ajustar los hiperparámetros durante el entrenamiento. Asimismo, PyTorch ofrece soporte nativo para aceleración mediante GPU a través de CUDA, lo cual fue fundamental para reducir los tiempos de entrenamiento e inferencia del modelo. Esta herramienta proporciona un equilibrio adecuado entre flexibilidad y facilidad de uso, una API clara y pythónica para la implementación de arquitecturas personalizadas, y una amplia adopción en investigación académica, lo que facilitó la replicación y comparación de métodos reportados en la literatura científica.

VI-N2. TorchVision: Se utilizó como librería complementaria para la gestión de datasets y la aplicación de transformaciones sobre las imágenes, lo que permite la implementación de técnicas de preprocesamiento como redimensionamiento, normalización y aumento de datos (data augmentation), garantizando que las imágenes cumplan con los requisitos de entrada de las arquitecturas de red neuronal. Además, facilitó el uso de

modelos preentrenados ampliamente difundidos y su integración directa con los DataLoaders de PyTorch, proporcionando utilidades eficientes para la manipulación de imágenes y el manejo estructurado de conjuntos de datos.

VI-N3. Ultralytics YOLOv8: Para la implementación del modelo YOLOv8-CLASSIFY se empleó la librería oficial Ultralytics, la cual proporcionó una API de alto nivel para el entrenamiento, validación y evaluación del modelo. Esta herramienta permitió aplicar configuraciones predefinidas y técnicas automáticas de aumento de datos, optimizando el rendimiento del modelo en la tarea de clasificación de la severidad de la retinopatía diabética. Adicionalmente, su implementación optimizada y mantenida, junto con herramientas integradas de visualización y registro (logging), facilitó el monitoreo sistemático del proceso de entrenamiento y el análisis del comportamiento del modelo.

VI-N4. HuggingFace Transformers: La librería Transformers de HuggingFace fue utilizada para acceder al modelo Vision Transformer (ViT-B/16) preentrenado. Esta herramienta permitió incorporar arquitecturas basadas en Transformers, ampliamente utilizadas en investigaciones recientes, facilitando la aplicación de técnicas de transferencia de aprendizaje y la comparación experimental con modelos convolucionales tradicionales. Asimismo, proporcionó implementaciones estandarizadas, utilidades de preprocesamiento y una documentación exhaustiva que permitió integrar el modelo de manera confiable y reproducible dentro del pipeline experimental.

VI-Ñ. Procesamiento de imágenes

VI-Ñ1. OpenCV: Utilizada para realizar operaciones de procesamiento digital de imágenes previas a la inferencia del modelo. En particular, se aplicaron técnicas como CLAHE (Contrast Limited Adaptive Histogram Equalization) para mejorar el contraste de las imágenes de fondo de ojo, así como conversiones entre distintos espacios de color. Estas técnicas contribuyeron a resaltar estructuras relevantes para el diagnóstico automático, permitiendo además operaciones básicas de manipulación, lectura y escritura de imágenes, así como conversiones entre los espacios RGB, LAB y HSV dentro del pipeline de preprocesamiento.

VI-Ñ2. PIL/Pillow: La librería Pillow permitió la carga y validación inicial de las imágenes subidas por los usuarios a través del sistema web, lo que permite verificar la integridad de los archivos, realizar transformaciones básicas y asegurar la compatibilidad con los pipelines de preprocesamiento definidos en PyTorch, funcionando además como una interfaz eficiente entre las imágenes cargadas y los DataLoaders utilizados durante las fases de entrenamiento e inferencia.

VI-O. Análisis y evaluación

VI-O1. Scikit-learn: Utilizada para el cálculo de métricas de evaluación del desempeño de los modelos entrenados. Entre las métricas consideradas se incluyen accuracy, precisión, recall, F1-score, y el área bajo la Curva ROC (AUC-ROC), así como la generación de matrices de confusión. Estas métricas permitieron analizar el comportamiento del modelo en cada una de las clases clínicas de severidad de la retinopatía diabética, incorporando además herramientas de división de datasets con estratificación y utilidades de validación que fortalecieron el rigor del análisis experimental.

VI-O2. NumPy: Esencial para la manipulación de arreglos numéricos y la implementación de operaciones matemáticas esencial para la manipulación de arreglos numéricos y la implementación de operaciones matemáticas y estadísticas durante el análisis de resultados. Además, facilitó la conversión eficiente entre tensores de PyTorch y estructuras numéricas utilizadas en el análisis posterior, permitiendo la implementación de métricas personalizadas cuando fue requerido.

VI-O3. Pandas: Se empleó para organizar los resultados experimentales y las métricas de entrenamiento en estructuras tabulares. Permitió analizar la distribución de clases del dataset, almacenar historiales de entrenamiento, manipular metadatos asociados a las imágenes y exportar resultados en formato CSV para su posterior documentación y análisis.

VI-P. Visualización

VI-P1. Matplotlib: Se utilizó para generar visualizaciones fundamentales del proceso experimental, como curvas de entrenamiento correspondientes a la pérdida y la precisión, gráficos de distribución de clases y visualización de matrices de confusión. Estas representaciones gráficas facilitaron la interpretación del rendimiento del modelo y la identificación de posibles problemas de sobreajuste o subajuste.

VI-P2. Seaborn: Complementó a Matplotlib proporcionando visualizaciones estadísticas más claras y estéticamente adecuadas para entornos académicos, siendo útil para la representación de matrices de confusión mediante mapas de calor (heatmaps) mejorados y para el análisis comparativo entre los distintos modelos evaluados, integrándose directamente con estructuras de datos de Pandas.

VI-Q. Desarrollo web

El sistema web de RetinaVision AI fue desarrollado siguiendo una arquitectura cliente-servidor, separando claramente la interfaz de usuario del procesamiento lógico y del modelo de inteligencia artificial.

VI-Q1. Frontend: La interfaz de usuario fue desarrollada utilizando React 19, lo que permitió la creación de una aplicación de página única (Single Page Application, SPA) con navegación fluida mediante React Router. El diseño visual se implementó con Tailwind CSS, garantizando una interfaz moderna y responsiva, mientras que los componentes reutilizables se construyeron con Radix UI y shadcn/ui, cumpliendo estándares de accesibilidad. El frontend se encarga de la carga de imágenes, la validación inicial de los archivos y la visualización de los resultados diagnósticos generados por el sistema.

VI-Q2. Backend: El backend fue implementado con FastAPI, un framework web de alto rendimiento basado en Python. Este componente actúa como el núcleo lógico del sistema, recibiendo las imágenes desde el frontend, validando su formato mediante esquemas definidos y enviándolas al microservicio de inteligencia artificial para su procesamiento. La comunicación se realiza mediante una API REST utilizando formato JSON y peticiones HTTP POST, garantizando una integración eficiente y escalable, apoyada por el servidor ASGI Uvicorn para una ejecución de alto rendimiento.

VI-R. Infraestructura y despliegue

Durante la fase de desarrollo y experimentación se utilizaron Jupyter Notebooks para prototipado iterativo, y entornos de desarrollo integrados como Visual Studio Code o PyCharm para la implementación del código final. El control de versiones se realizó mediante Git, utilizando GitHub como repositorio remoto para facilitar la colaboración y el despliegue continuo.

El entrenamiento de los modelos se ejecutó principalmente en Google Colab Pro, aprovechando el acceso a GPUs de alto rendimiento, mientras que Google Drive se utilizó para el almacenamiento persistente de datasets y modelos entrenados. Para el despliegue del sistema se empleó Docker, permitiendo encapsular la aplicación y garantizar consistencia entre entornos, y la plataforma cloud Railway, que facilitó el despliegue automático y escalable del sistema web.

VI-S. Implementación de Hardware

Este proyecto fue desarrollado principalmente Google Colab Pro con acceso a GPUs Tesla T4 y A100, lo que proporcionó recursos computacionales suficientes para entrenar los tres modelos en tiempos razonables (entre 4-8 horas por modelo dependiendo de la configuración). Para la ejecución eficiente del proyecto se establecieron requisitos mínimos y óptimos de hardware.

- **GPU:** NVIDIA con al menos 8GB VRAM (GTX 1080 Ti, RTX 2070 o superior).
- **RAM:** 16GB sistema .
- **Almacenamiento:** 100GB disponibles (para dataset y modelos).
- **CPU:** Procesador multi-core moderno (Intel i7/i9 o AMD Ryzen 7/9).

Configuración óptima

- **GPU:** NVIDIA RTX 3090, A100 o similar (24GB+ VRAM).
- **RAM:** 32GB+ sistema .
- **Almacenamiento:** SSD NVMe de 500GB+.
- **CPU:** Procesador de alta gama con 8+ cores.

VI-T. Flujo de inferencia

En RetinaVisionAI, el flujo de inferencia se diseño como un proceso estructurado secuencialmente, donde cada imagen pasa primero por varias etapas antes de que el modelo emita una predicción,lo que garantizó precisión, seguridad y consistencia en los resultados. Dichas etapas se organizan en las siguientes etapas:

VI-T1. Carga de la Imagen: Ocurre en el frontend desarrollado en React, cuando el usuario selecciona una imagen desde su dispositivo, esta se envía al backend mediante una petición HTTP POST utilizando la API de FastAPI. Durante esta etapa, se emplea la librería Pillow para abrir y verificar que el archivo realmente contenga una imagen, lo que garantizó que al verificar el archivo, este corresponda a una imagen valida, evitando que el sistema intente procesar archivos no compatibles o corruptos, un problema que puede causar errores de ejecución o predicciones no válidas.

VI-T2. Validación del Formato: Cada arquitectura empleada en el proyecto contiene requisitos específicos en cuanto al tamaño, número de canales y las estructura de entradas, por lo que, después de la recepción correcta del archivo, se realiza una validación adicional a la estructura del archivo. Esto incluye:

- La verificación de que la extensión del archivo sea de un formato compatible (JPG o PNG), con las transformaciones definidas en TorchVision y OpenCV, antes de ingresar al pipleline de transformación de adaptación al modelo.
- La comprobación de su estructura interna, que integra la lectura de bytes, donde se busca que la contrucción de la matriz de pixeles sea validada, garantizando que la imagen sea interpretable y descartando la presencia de corrupción en la calidad de los pixeles.

- La revisión de las dimensiones mínimas adecuadas para asegurar su procesamiento posterior.

VI-T3. Estandarización de la Imagen de entrada: Se transforma la imagen cruda, usando herramiento como OpenCV y las transformaciones de TorchVision:

- Se aplica la conversión de los espacios de color.
- Se ajusta el contraste mediante técnicas como ClAHE, normalización de valores de pixel.
- Redimensionamiento al tamaño de entrada esperado en todos los modelos.
- Conversión a tensor.

La estandarización de la imagen, es aplicada en los modelos de inteligencia artificial entrenados con PyTorch y HuggingFce, debido a que, asegura que las imágenes nuevas se asemejen a las escalas numéricas de las imágenes usadas durante el entrenamiento. Las imágenes originales presentaron características específicas: mismos rangos de valores, mismo formato, misma distribución de colores, por lo que, los modelos aprendieron a reconocer estos patrones bajo esas condiciones, en consecuencia, si los modelos reciben nuevas imágenes con otro rango de color, escala de valores, tamaño de resolución o formato de archivo, van a recibir datos fuera de lo que se espera, lo que disminuye la precisión, y confiabilidad de los modelos.

VI-T4. Inferencia: En esta etapa, la imagen ya transformada en una representación numérica adecuada se envía al modelo previamente entrenado en modo evaluación, lo que garantiza un comportamiento estable durante la predicción. La inferencia se realiza mediante una propagación hacia adelante, generando como salida valores (logits) para cada nivel de severidad de retinopatía diabética. Estos valores se convierten mediante la función softmax en probabilidades normalizadas, permitiendo obtener una distribución completa en lugar de solo una etiqueta. Identificando casos con baja confianza, aportando mayor claridad sobre el grado de certeza del sistema. Cuando se dispone de GPU, la estandarización de los datos se acelera mediante cálculo paralelo, reduciendo el tiempo de respuesta.

VI-T5. Salida del Modelo y Representación Probabilística: La salida final del sistema no es únicamente una etiqueta categórica. El backend construye una respuesta estructurada en formato JSON que contiene:

- La clase con mayor probabilidad.
- El conjunto completo de probabilidades asociadas a cada categoría.

VI-U. Consentimiento informado y consideraciones éticas

RetinaVisionAI, incorpora un consentimiento informado, antes de permitir cualquier interacción con el sistema, lo que impulsa que el usuario lea la información y condiciones previas de uso, en consecuencia la plataforma permanece inactiva hasta que el usuario otorgue su consentimiento explícito, confirmando que comprende el alcance predictivo, los beneficios, las limitaciones y el uso de la herramienta.

El consentimiento informado incluye tres condiciones previo a la carga, indicando:

- El sitio web está diseñado para aplicarse exclusivamente con fines académicos y de investigación, esto debido a que, los modelos de inteligencia artificial fueron entrenados en Google Colab Pro, por lo que, no han pasado por procesos de validación clínica ni por procedimientos regulares que se exigen en los

sistemas de diagnóstico médico formal, en consecuencia, informar al usuario sobre su alcance ayuda a prevenir el uso inadecuado de los datos fuera del escenario para el cual fueron generados.

- La plataforma se compromete a proteger los datos de los usuarios, garantizando la privacidad y confidencialidad de la información, al procesar únicamente las imágenes del fondo del ojo para generar la predicción solicitada, resultados que no se almacena de manera indefinida, y tampoco se reutilizan con fines distintos a los aceptados por el usuario.

- Los resultados obtenidos de la plataforma no sustituyen el diagnóstico médico profesional, si bien estos algoritmos detectan patrones estadísticos en imágenes, y determinan la probabilidad asociada a las etapas de severidad de la RD, una de sus principales limitaciones es que no pueden integrar información clínica completa y tampoco puede estimar un juicio clínico global, debido a que, las predicciones automáticas deben ser interpretadas por profesionales de la salud.

VII. Resultados

VII-A. Pantalla inicial de términos y condiciones de uso

La Figura 29 presenta la ventana emergente de términos y condiciones de uso desplegada al inicio de la interacción con la plataforma web desarrollada. Esta interfaz constituye el primer elemento visible para el usuario antes de acceder a las funcionalidades de análisis, estableciendo las condiciones de uso, privacidad y limitaciones de responsabilidad asociadas al procesamiento automatizado de imágenes retinográficas dentro del sitio web.

En esta sección se especifica que la plataforma posee una finalidad académica y de investigación, orientada a evaluar el desempeño de modelos de aprendizaje profundo en la clasificación multietapa de la retinopatía diabética. Asimismo, se indica que las imágenes cargadas son procesadas únicamente durante la sesión activa, sin almacenamiento permanente, y que los resultados emitidos por el sistema corresponden a estimaciones algorítmicas que no sustituyen el criterio clínico profesional. Finalmente, la continuación hacia el módulo de análisis requiere la aceptación explícita de estas condiciones por parte del usuario.

Figura 29. Ventana inicial de términos y condiciones de uso de la plataforma web. Fuente: Autor Propio

VII-B. Resultados mostrados por el sitio web para el modelo personalizado

VII-B1. Caso sin retinopatía - Etapa 0: La Figura 30 muestra el resultado desplegado por la plataforma web para un caso analizado mediante el modelo personalizado y clasificado como *sin retinopatía*. En esta interfaz se integran la imagen original cargada por el usuario, el mapa de calor *Grad-CAM*, la distribución de probabilidades por clase y el diagnóstico individual emitido por la arquitectura *EfficientNet-B0 + 10 capas convolucionales + External Attention*.

Para este caso, el sitio web reporta una probabilidad de 91.1 % para la clase *Sin Retinopatía*, mientras que las probabilidades asociadas a las demás clases permanecen considerablemente bajas, destacándose únicamente un valor menor en la categoría *RD Leve*. El panel de diagnóstico individual, además de confirmar la predicción principal, incorpora información técnica relevante sobre el preprocesamiento aplicado, como el tamaño de entrada de 224 $\times$ 224 píxeles, el uso del espacio de color RGB y la normalización basada en *ImageNet*.

De manera complementaria, el mapa de calor *Grad-CAM* presentado en el sitio web permite identificar las regiones de la imagen que tuvieron mayor influencia en la decisión del modelo, aportando interpretabilidad visual al resultado. En conjunto, esta vista evidencia que la plataforma no solo proporciona una clasificación automática, sino también recursos gráficos y técnicos que facilitan la comprensión del análisis realizado por el modelo personalizado dentro del entorno web.

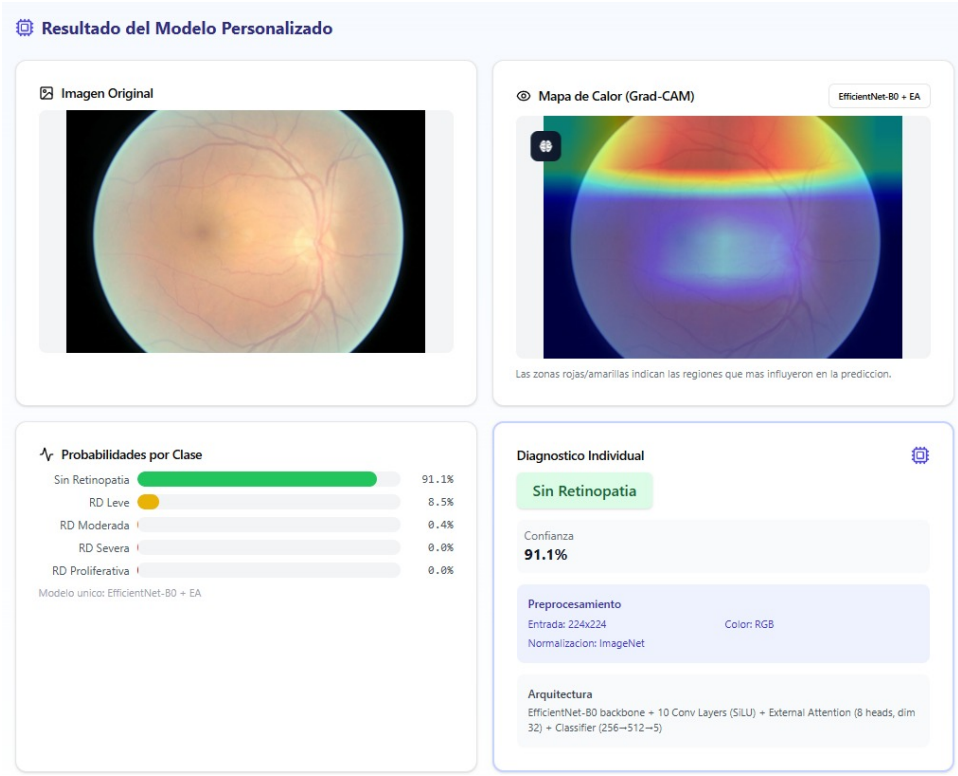


Figura 30. Resultado mostrado por la plataforma web para un caso clasificado como sin retinopatía, incluyendo imagen original, mapa de calor Grad-CAM, probabilidades por clase y diagnóstico individual. Fuente: Autor Propio

VII-B2. Caso con retinopatía - Etapa 2: La Figura 31 presenta el resultado mostrado por la plataforma web para una imagen clasificada como *retinopatía diabética moderada* mediante el modelo personalizado. En esta vista se integran la imagen original, el mapa de calor *Grad-CAM*, la distribución de probabilidades por clase y el diagnóstico individual generado por la arquitectura *EfficientNet-B0 + 10 capas convolucionales + External Attention*.

En el caso observado, la plataforma asigna la mayor probabilidad a la clase *RD Moderada*, con una confianza de 63.7 %. Asimismo, la segunda probabilidad más alta corresponde a la categoría *RD Leve*, con 33.5 %, mientras que las clases *Sin Retinopatía*, *RD Severa* y *RD Proliferativa* presentan valores considerablemente menores. Este comportamiento indica que el modelo implementado en el sitio web logra reconocer la etapa predominante de la enfermedad, aunque conserva cierta cercanía con la clase inmediatamente anterior, lo cual resulta coherente en escenarios donde existen similitudes visuales entre etapas contiguas de la retinopatía

diabética.

El panel de diagnóstico individual mostrado en la interfaz también resume información técnica del procesamiento realizado, incluyendo el tamaño de entrada de 224×224 píxeles, el uso del espacio de color RGB y la normalización basada en *ImageNet*. A su vez, el mapa de calor *Grad-CAM* resalta las regiones con mayor influencia en la decisión del modelo, fortaleciendo la interpretabilidad del resultado y la transparencia del proceso de clasificación presentado por la plataforma web.

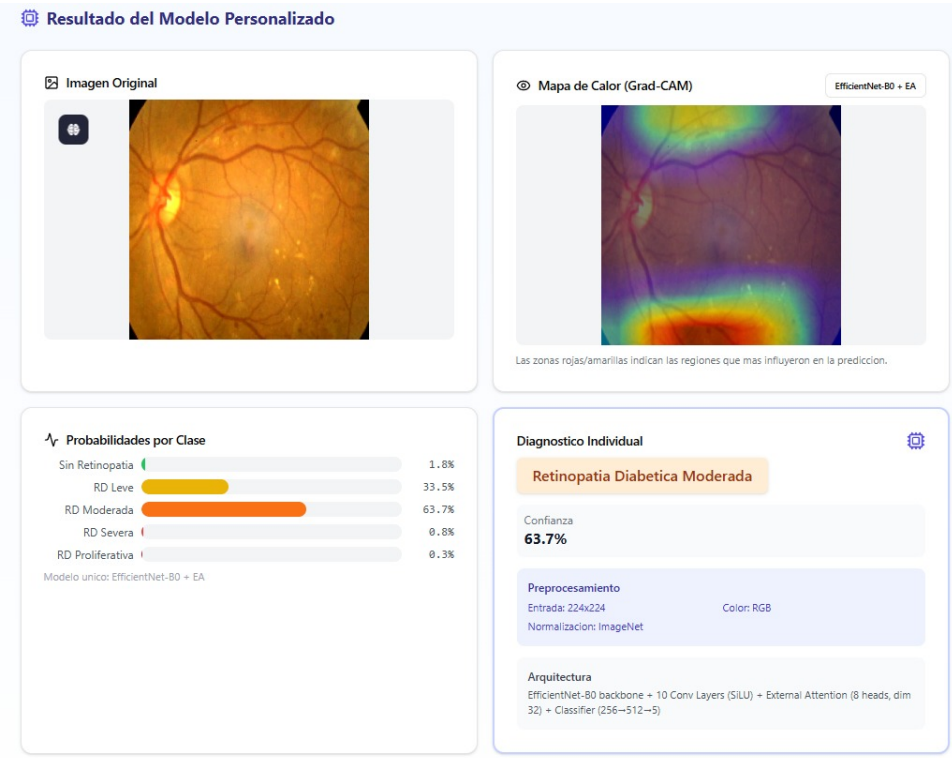


Figura 31. Resultado mostrado por la plataforma web para un caso clasificado como retinopatía diabética moderada, incluyendo imagen original, mapa de calor Grad-CAM, probabilidades por clase y diagnóstico individual. Fuente: Autor Propio

VII-C. Resultados de análisis y comparación entre modelos en la plataforma web

VII-C1. Caso sin retinopatía - Etapa 0: La Figura 33 presenta la comparación de confianza mostrada por la plataforma web entre el modelo principal y los demás modelos del ensamble para una imagen analizada dentro del sistema. En esta interfaz se observa que el modelo *EfficientNet-B0 + External Attention* asigna una alta probabilidad a la clase *Sin Retinopatía*, con una confianza de 91.1 %, mientras que los demás modelos exhiben distintos niveles de coincidencia y discrepancia respecto a dicha predicción. Esta visualización permite examinar el grado de consenso entre arquitecturas, así como la confianza relativa del resto de modelos evaluados.

Por otra parte, la Figura 32 muestra el panel de resultado general del análisis generado por la plataforma web, integrando la imagen original, el mapa de calor *Grad-CAM*, las probabilidades promedio por clase y el diagnóstico final emitido por consenso. En esta captura, el sitio web reporta la categoría *Retinopatía Diabética Leve*, con una confianza global de 77.0 % y un acuerdo de 2 de 5 modelos. Adicionalmente, el mapa de calor permite identificar las regiones de mayor influencia en la predicción final, facilitando la interpretación visual del resultado general entregado por el sistema.

En conjunto, ambas figuras evidencian que la plataforma web no se limita a mostrar la salida individual de un único modelo, sino que incorpora un esquema comparativo entre arquitecturas, permitiendo visualizar el nivel de acuerdo entre ellas y reforzando la comprensión del resultado final emitido al usuario.

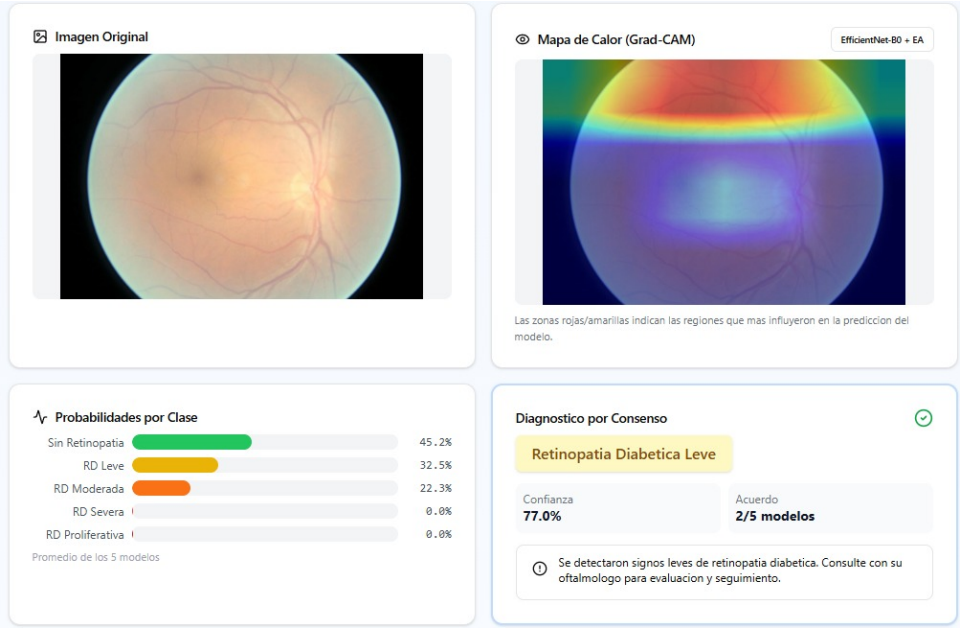


Figura 32. Resultado general del análisis por consenso mostrado por la plataforma web, incluyendo imagen original, mapa de calor Grad-CAM, probabilidades promedio por clase y diagnóstico final. Fuente: Autor Propio

Figura 33. Comparación de confianza mostrada por la plataforma web entre el modelo principal y el resto de modelos del ensamble para un caso clasificado como sin retinopatía. Fuente: Autor Propio

VII-C2. Caso con retinopatía - Etapa 2:

La Figura 35 presenta el resultado mostrado por la plataforma web para una imagen correspondiente a un caso de *retinopatía diabética moderada*, analizada mediante el modelo personalizado. En esta interfaz se visualizan la imagen original, el mapa de calor *Grad-CAM*, las probabilidades por clase y el diagnóstico individual generado por la arquitectura *EfficientNet-B0 + 10 capas convolucionales + External Attention*. Para este caso, la plataforma reporta la mayor probabilidad en la clase *RD Moderada*, con una confianza de 63.7 %, seguida de la categoría *RD Leve*, con 33.5 %. En contraste, las clases *Sin Retinopatía*, *RD Severa* y *RD Proliferativa* presentan valores significativamente menores.

Este comportamiento sugiere que el sistema identifica la presencia de alteraciones retinianas compatibles con una severidad intermedia, aunque conserva proximidad con la clase leve, lo que puede explicarse por la similitud visual existente entre etapas contiguas de la enfermedad. Asimismo, el mapa de calor *Grad-CAM* integrado en el sitio web permite localizar las regiones de la retinografía que tuvieron mayor influencia en la decisión del modelo, aportando interpretabilidad visual al proceso de clasificación.

Por su parte, la Figura 34 muestra la comparación de confianza presentada por la plataforma web entre el modelo principal y los demás modelos del ensamble para la misma imagen. En esta vista, el modelo personalizado mantiene su predicción en la categoría *Retinopatía Diabética Moderada* con 63.7 %, mientras que los demás modelos presentan discrepancias relevantes: *DenseNet121 + EA* clasifica el caso como *Severa*, *ResNet50 + EA* como *Leve*, *ViT-B/16 + EA* como *Sin Retinopatía* y *YOLOv8x-cls* nuevamente como *Leve*. La ausencia de coincidencias directas con el modelo principal (0/4 modelos) evidencia la complejidad del caso y la dificultad que representa la etapa moderada para las distintas arquitecturas evaluadas, especialmente cuando se presentan rasgos compartidos con clases adyacentes.

En este sentido, las capturas del sitio web permiten observar no solo la predicción individual del modelo principal, sino también la variabilidad existente entre arquitecturas al enfrentarse a casos clínicamente más ambiguos, lo cual aporta valor analítico a la interpretación de los resultados desplegados por la plataforma.

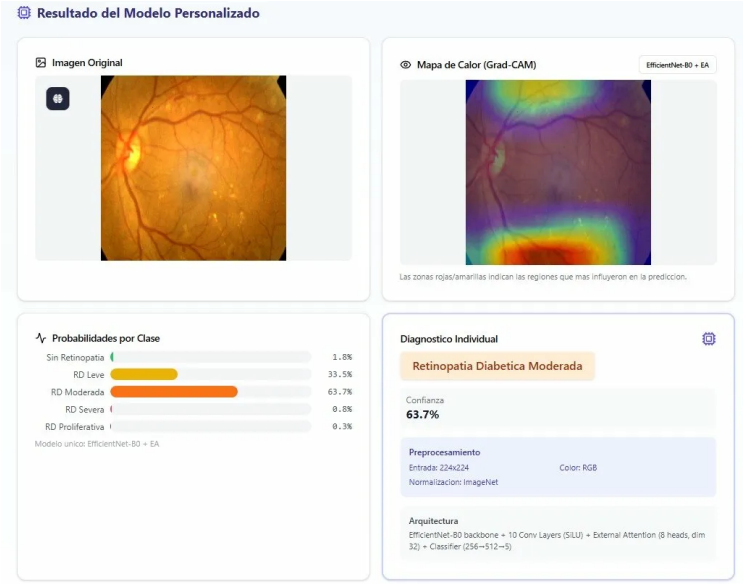


Figura 34. Comparación de confianza mostrada por la plataforma web entre el modelo principal y los demás modelos del ensamble para un caso correspondiente a la etapa 2 de la retinopatía diabética. Fuente: Autor Propio

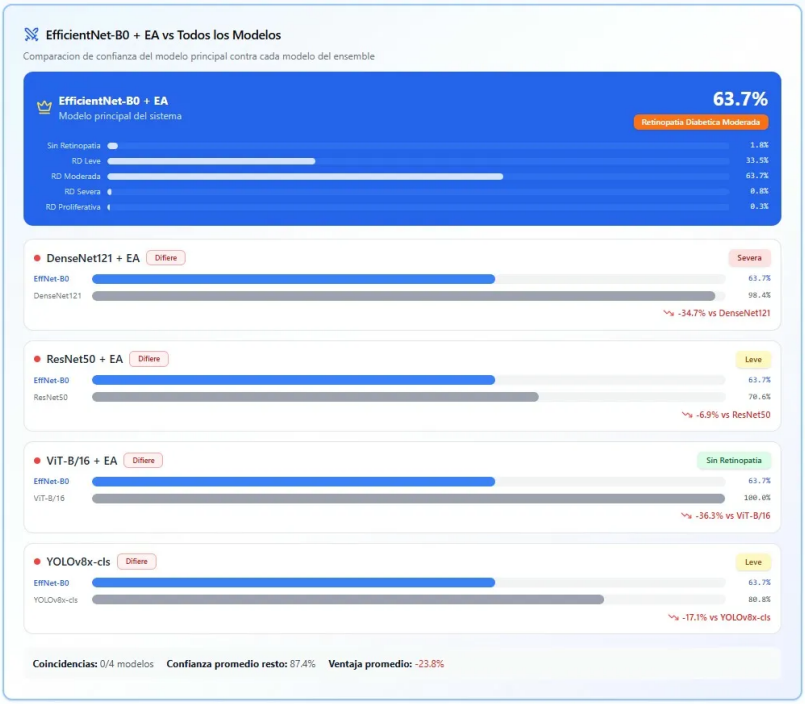


Figura 35. Resultado mostrado por la plataforma web para un caso clasificado como retinopatía diabética moderada. Fuente: Autor Propio

VII-D. Rendimiento durante el entrenamiento

El análisis del rendimiento durante el entrenamiento se organizó según la naturaleza arquitectónica de los modelos evaluados. En primer lugar, se presentan los resultados de las redes convolucionales profundas (CNN), posteriormente la arquitectura YOLOv8 en modalidad de clasificación, y finalmente el modelo basado en autoatención Vision Transformer (ViT-B/16). Esta organización permite contrastar la evolución estructural desde enfoques convolucionales clásicos hasta modelos basados en atención global.

En el caso del modelo personalizado basado en EfficientNet-B0, el entrenamiento se realizó mediante transferencia de aprendizaje con congelamiento parcial de capas, conservando los pesos preentrenados en ImageNet y ajustando las capas finales al problema de clasificación multietapa de retinopatía diabética. Durante las primeras épocas se observó una convergencia rápida, caracterizada por una disminución sostenida de la pérdida y un incremento progresivo de la precisión de validación. Sin embargo, hacia las últimas épocas se evidenció una brecha creciente entre entrenamiento y validación del orden de 16–18 %, lo que indica un sobreajuste moderado. A pesar de ello, el modelo alcanzó una precisión máxima de validación de 82.25 %, manteniendo un comportamiento estable sin divergencias abruptas.

Por su parte, ResNet-50 mostró una convergencia eficiente desde etapas tempranas, beneficiándose de su arquitectura residual, que facilita la propagación del gradiente en redes profundas. La precisión máxima de validación alcanzó 78.36 %. No obstante, también presentó una brecha entrenamiento–validación más pronunciada, del orden de 19–22 % en la fase final, lo que evidencia una tendencia mayor al sobreajuste en etapas avanzadas del entrenamiento. Este comportamiento es coherente con su mayor capacidad paramétrica y el ajuste fino prolongado.

En el caso de DenseNet-121, el entrenamiento fue progresivo y estable. Su conectividad densa favoreció la reutilización de características y un flujo eficiente de gradientes, lo que se reflejó en una dinámica de aprendizaje regular. El modelo alcanzó una precisión máxima de validación de 77.49 %; sin embargo, hacia el cierre del entrenamiento el *gap* train–val se incrementó hasta valores cercanos a 16 %, indicando un sobreajuste

moderado.

Posteriormente, se evaluó la arquitectura YOLOv8 en modalidad de clasificación. El modelo mostró un desempeño global elevado en el conjunto de prueba (Precision=0.8201, Recall=0.8188, F1=0.8192, Specificity=0.9547 y AUC macro=0.9620). Estos resultados indican una alta capacidad discriminativa global. Sin embargo, el análisis por clase evidenció mayor dificultad en la clase intermedia (Grado 2), donde el rendimiento fue inferior en comparación con clases extremas, reflejando la complejidad clínica de diferenciar estadios moderados.

Finalmente, el modelo Vision Transformer (ViT-B/16) presentó una convergencia inicial más lenta en comparación con las CNN, comportamiento esperable debido a su mecanismo de autoatención global. Una vez estabilizado el aprendizaje, el modelo alcanzó una precisión máxima de validación de 76.85 %. No obstante, hacia la fase final se observó una brecha train–val del orden de 18–21 %, lo que sugiere sobreajuste. Su desempeño debe interpretarse en conjunto con la matriz de confusión y las métricas por clase para determinar su aporte diferencial frente a arquitecturas convolucionales.

VII-E. Desempeño individual por modelo

Esta sección resume el comportamiento de cada arquitectura en dos niveles: (I) dinámica de aprendizaje durante *train/val* (convergencia, estabilidad y sobreajuste) y (II) rendimiento final en clasificación multiclase (matriz de confusión, métricas por clase y AUC-ROC OvR). Para mantener consistencia entre modelos, en cada subapartado se incluyen seis visualizaciones: historial de entrenamiento, matriz de confusión, métricas por clase, curvas/infos ROC, AUC por clase y análisis de *gap* (train–val).

VII-E1. EfficientNet-B0 (modelo personalizado) + External Attention: El modelo personalizado basado en EfficientNet-B0 presentó una convergencia rápida y estable en las primeras épocas. En la Figura 36 se observa la evolución de *loss* y *accuracy* para entrenamiento y validación, donde la disminución sostenida de la pérdida se acompaña de un incremento progresivo de la exactitud en validación. Hacia la fase final, el aprendizaje en entrenamiento continuó mejorando mientras la validación tendió a estabilizarse, reflejando una tendencia a sobreajuste moderado; el mejor desempeño de validación fue de 82.25 % de *val accuracy*.

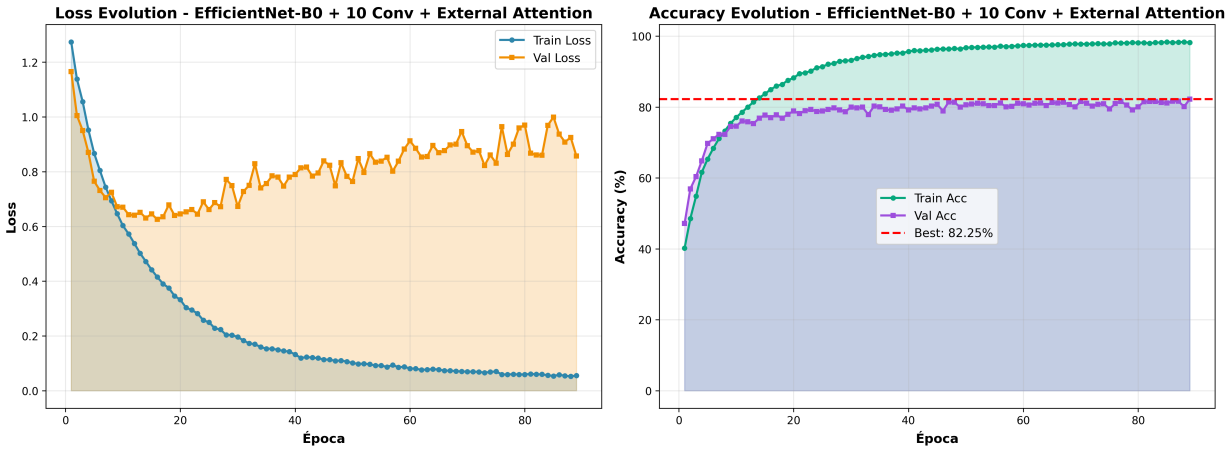


Figura 36. EfficientNet-B0: evolución de *loss* y *accuracy* en entrenamiento y validación. Fuente: Autor Propio.

La Figura 37 muestra la matriz de confusión en el conjunto de prueba. Se aprecia un alto número de aciertos en las clases extremas (grados 0 y 4), mientras que los principales errores se concentran entre clases contiguas, especialmente alrededor de la clase intermedia (Grado 2). Este comportamiento es consistente con la dificultad

clínica de separar estadios cercanos cuando las diferencias son sutiles.

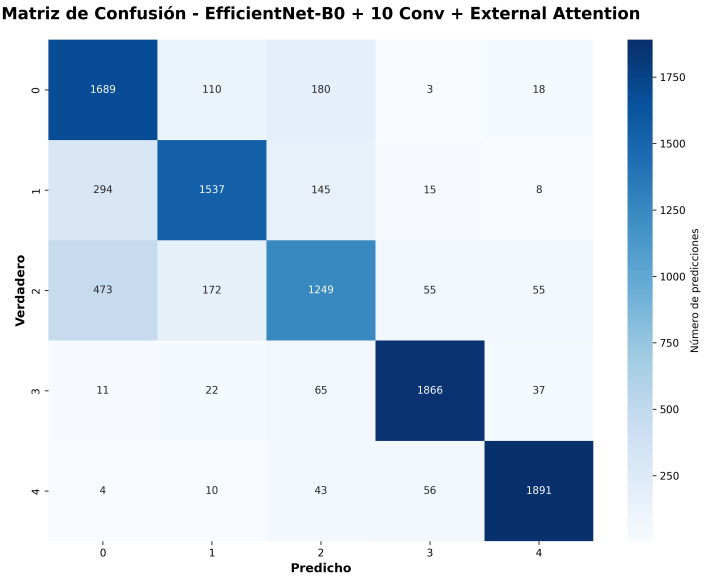


Figura 37. EfficientNet-B0: matriz de confusión multiclase (test). Fuente: Autor Propio.

En la Figura 38 se reportan las métricas por clase (Precision, Recall, F1 y Specificity). En términos generales, las clases avanzadas presentan valores elevados, mientras que la clase 2 tiende a mostrar un rendimiento relativamente menor, lo que confirma que el desafío principal del modelo se encuentra en la discriminación de estadios intermedios frente a grados vecinos.

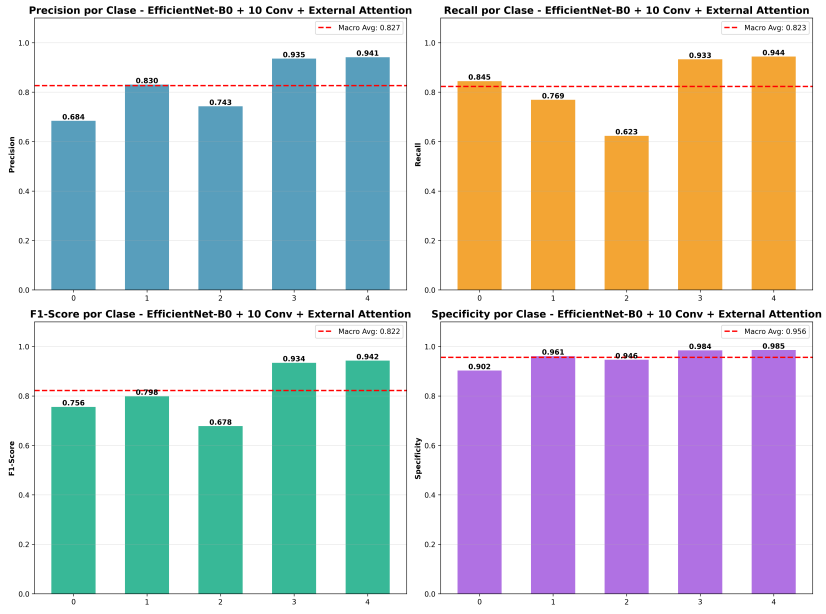


Figura 38. EfficientNet-B0: Precision/Recall/F1/Specificity por clase (test). Fuente: Autor Propio.

La Figura 39 presenta la información ROC en esquema one-vs-rest (OvR) para cada clase, permitiendo analizar la capacidad de separación del modelo bajo distintos umbrales. Complementariamente, la Figura 40 resume

el AUC por clase y el promedio macro, evidenciando alta capacidad discriminativa global con AUC macro de 0.954 y valores altos en todas las clases, particularmente en los grados más avanzados.

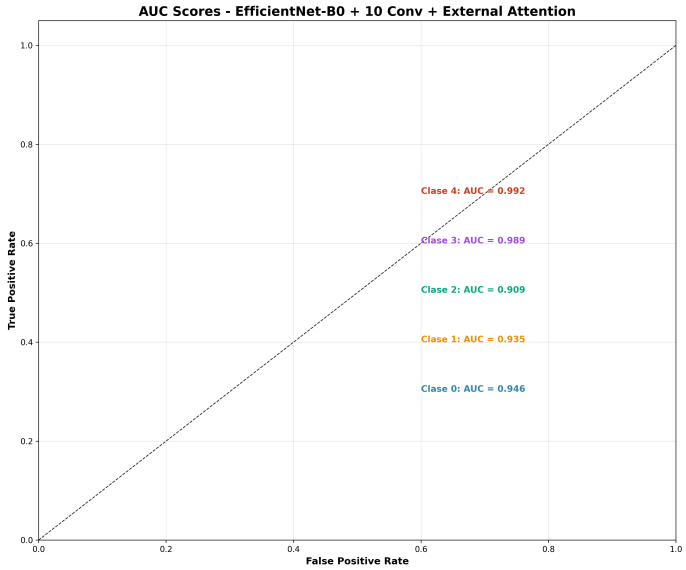


Figura 39. EfficientNet-B0 (personalizado): información ROC OvR por clase (test). Fuente: Autor Propio.

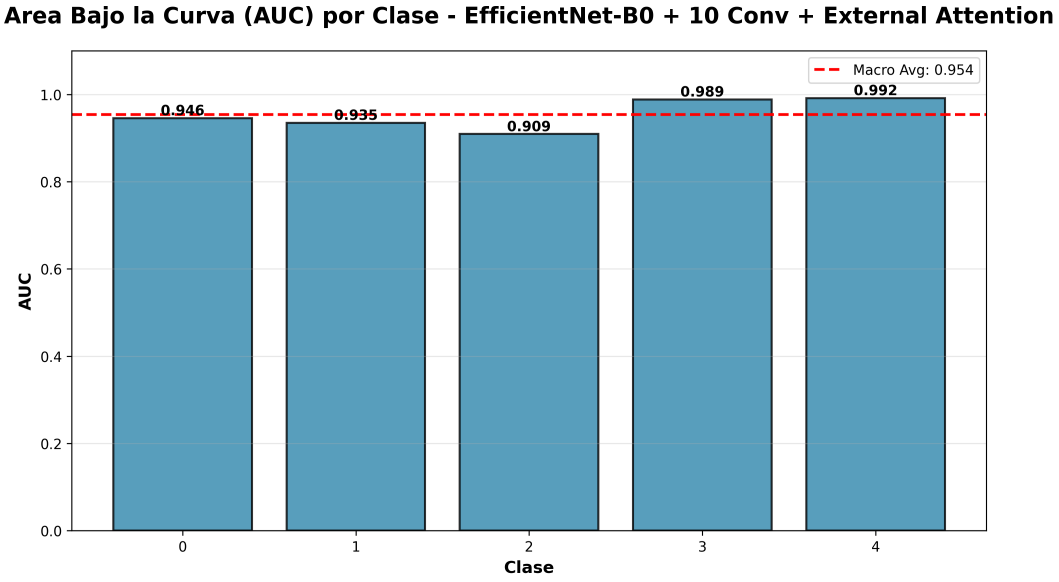


Figura 40. EfficientNet-B0 (personalizado): AUC por clase y promedio macro (test). Fuente: Autor Propio.

Finalmente, la Figura 41 muestra el análisis del *gap* entre entrenamiento y validación a lo largo de las épocas. Se observa un incremento progresivo hacia el final del entrenamiento, con valores aproximados de 16–18 %, lo que indica sobreajuste moderado. A pesar de ello, el modelo mantuvo estabilidad en validación y un rendimiento competitivo, lo que lo posiciona como la propuesta personalizada de referencia para la comparación con arquitecturas preentrenadas.

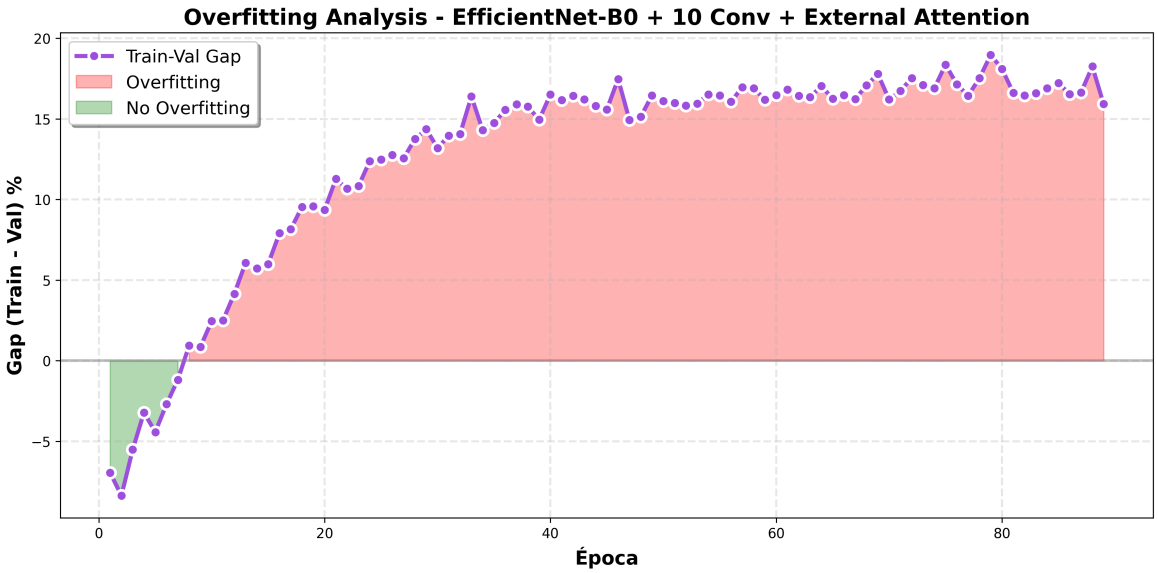


Figura 41. EfficientNet-B0: análisis del *gap* (% train − % val) por época. Fuente: Autor Propio.

VII-E2. ResNet-50 + External Attention: Presentó una convergencia eficiente desde las primeras épocas, favorecida por su estructura residual. El desempeño en validación se mantuvo alto con fluctuaciones leves, alcanzando un máximo de 78.36 % de *val accuracy*. La diferencia train–val se incrementó en la fase final y se mantuvo aproximadamente en el rango 19–22 %, lo que evidencia una tendencia marcada al sobreajuste sin observarse un colapso abrupto de la generalización.

La Figura 42 muestra el historial de entrenamiento y validación del modelo, donde se evidencia una convergencia eficiente desde las primeras épocas, favorecida por su estructura residual. El mejor desempeño en validación alcanzó 78.36 % de *val accuracy*.

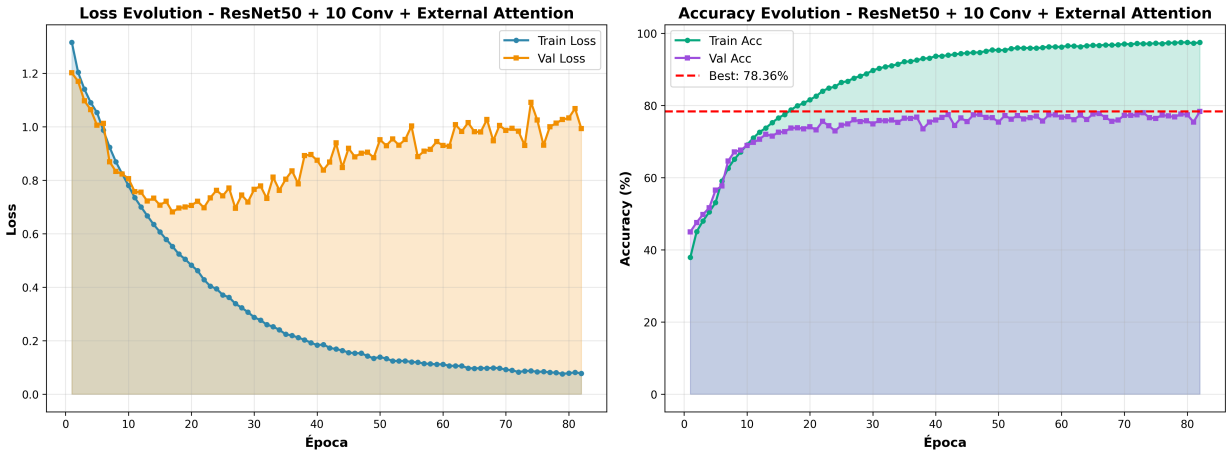


Figura 42. ResNet-50: evolución de *loss* y *accuracy* en entrenamiento y validación. Fuente: Autor Propio.

La Figura 43 presenta la matriz de confusión multiclase en el conjunto de prueba. Se observa que los aciertos se concentran en las clases extremas, mientras que los errores se presentan principalmente entre clases contiguas, especialmente alrededor del grado intermedio (Grado 2), lo que refleja la dificultad de separar estadios cercanos.

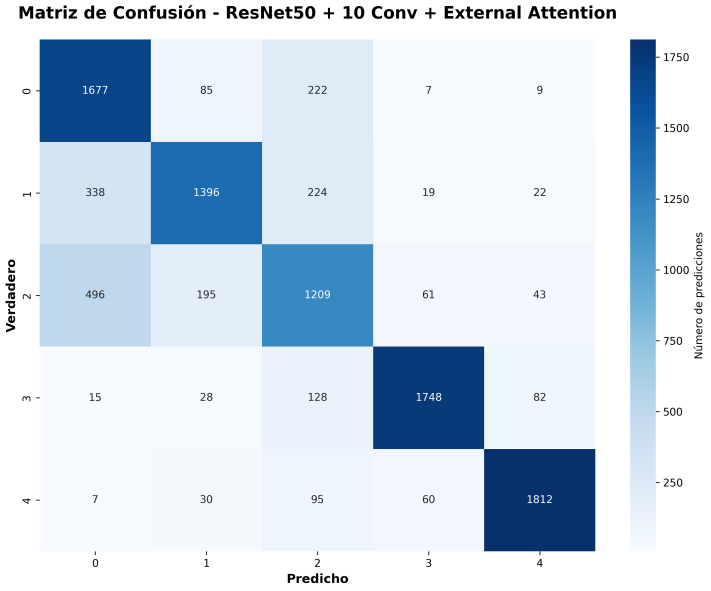


Figura 43. ResNet-50: matriz de confusión multiclase (test). Fuente: Autor Propio.

En la Figura 44 se reportan las métricas por clase (Precision, Recall, F1 y Specificity) en test. El comportamiento confirma un rendimiento más alto en clases avanzadas, mientras que la clase 2 tiende a registrar los valores más bajos, coherente con la complejidad clínica de los estadios intermedios.

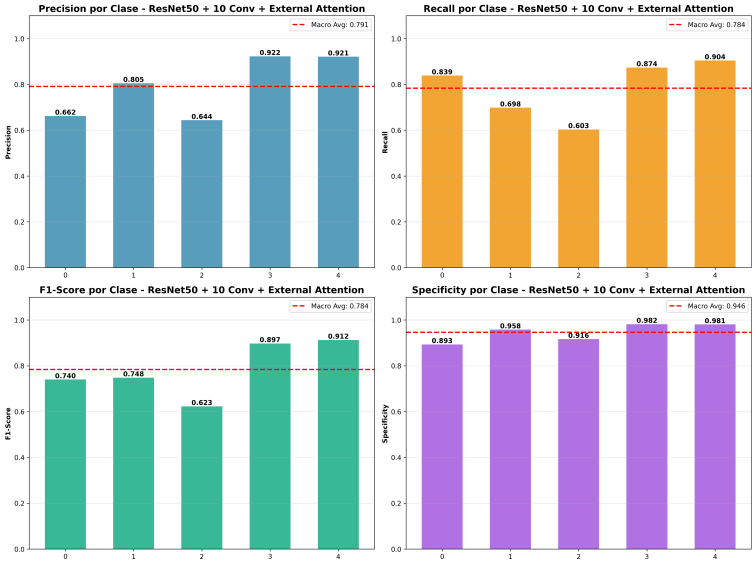


Figura 44. ResNet-50: Precision/Recall/F1/Specificity por clase (test). Fuente: Autor Propio.

La Figura 45 muestra la información ROC OvR por clase, permitiendo analizar la capacidad del modelo para distinguir cada categoría frente al resto bajo distintos umbrales de decisión.

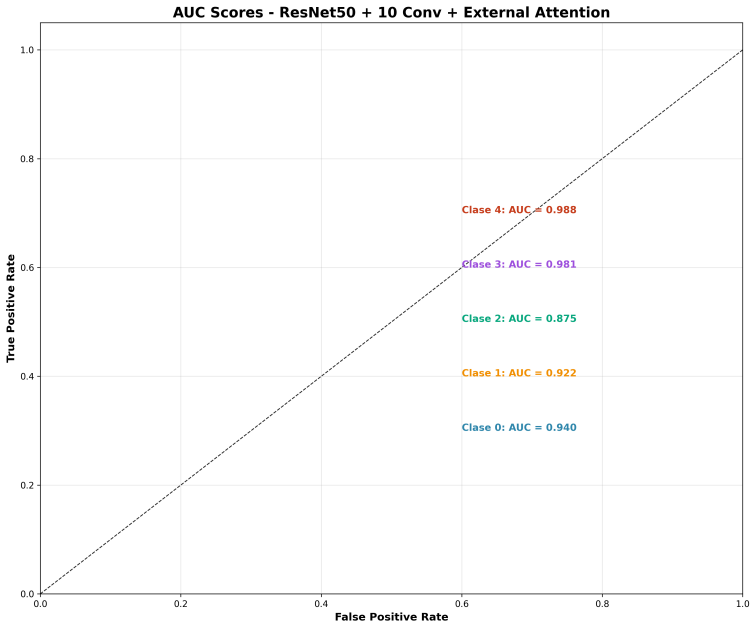


Figura 45. ResNet-50: información ROC OvR por clase (test). Fuente: Autor Propio.

La Figura 46 presenta el AUC por clase y el promedio macro, evidenciando una capacidad discriminativa global elevada con AUC macro de 0.941.

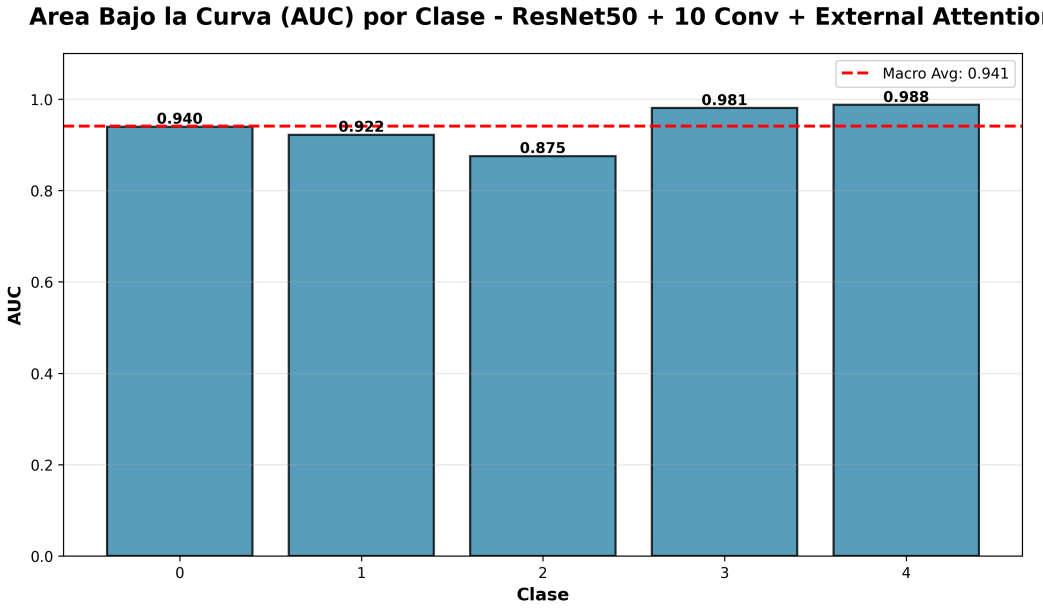


Figura 46. ResNet-50: AUC por clase y promedio macro (test). Fuente: Autor Propio.

Finalmente, la Figura 47 muestra el análisis del *gap* entre entrenamiento y validación a lo largo de las épocas. Se observa un incremento hacia la fase final, alcanzando aproximadamente 19–22 %, lo que indica sobreajuste moderado a marcado. A pesar de ello, la validación se mantuvo estable sin degradación abrupta, por lo que la generalización no colapsó durante el entrenamiento.

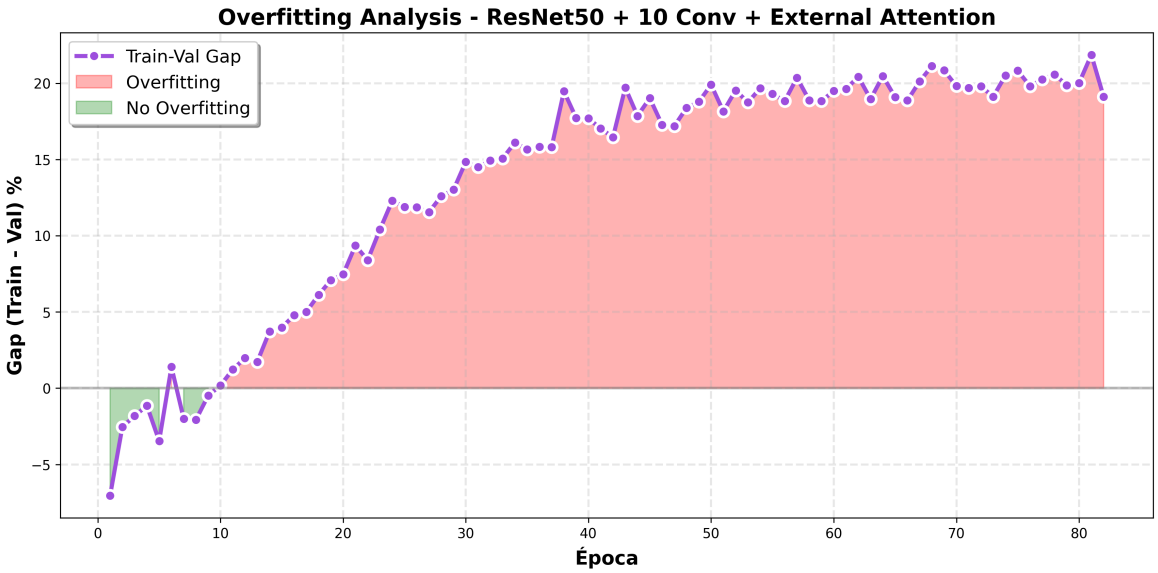


Figura 47. ResNet-50: análisis del *gap* (% train – % val) por época. Fuente: Autor Propio.

VII-E3. DenseNet-121 + External Attention: Presentó una dinámica de entrenamiento progresiva y regular entre las CNN evaluadas. La conectividad densa favoreció la reutilización de características y un flujo eficiente de gradientes, contribuyendo a una generalización consistente. Su mejor *val accuracy* fue 77.49 %. No obstante, hacia la fase final se observó un incremento del *gap* train–val hasta valores cercanos a 16 %, lo que indica sobreajuste moderado sin evidenciar una degradación abrupta del rendimiento en validación.

La Figura 48 presenta el historial de entrenamiento y validación de DenseNet-121, mostrando la evolución de *loss* y *accuracy*. Se aprecia una convergencia progresiva, con disminución sostenida de la pérdida y aumento gradual de la exactitud. Este comportamiento permite evaluar la estabilidad del aprendizaje y la diferencia entre curvas de entrenamiento y validación a lo largo de las épocas.

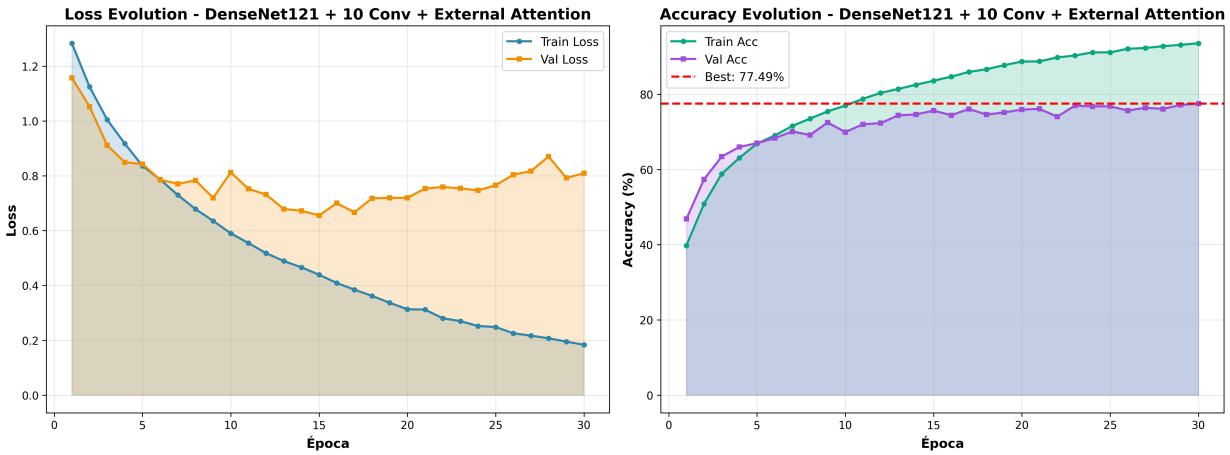


Figura 48. DenseNet-121: evolución de *loss* y *accuracy* en entrenamiento y validación. Fuente: Autor Propio.

La Figura 49 muestra la matriz de confusión multiclase en el conjunto de prueba. En ella se identifican los aciertos sobre la diagonal principal y los errores de clasificación fuera de ella. Se observa que las confusiones se concentran principalmente entre clases contiguas, especialmente alrededor del Grado 2, lo cual es coherente con

la dificultad de separar estadios intermedios que presentan características clínicas similares a grados vecinos.

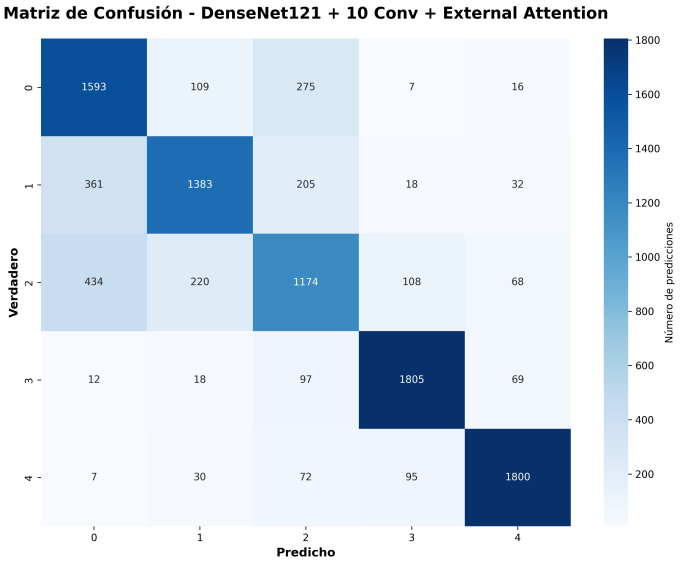


Figura 49. DenseNet-121: matriz de confusión multiclase (test). Fuente: Autor Propio.

La Figura 50 resume las métricas por clase (Precision, Recall, F1 y Specificity) en test. Esta visualización permite comparar el desempeño del modelo por categoría y detectar cuáles clases son mejor reconocidas. En general, las clases extremas tienden a presentar valores más altos, mientras que la clase intermedia registra métricas relativamente menores, indicando un reto mayor en la discriminación de grados moderados.

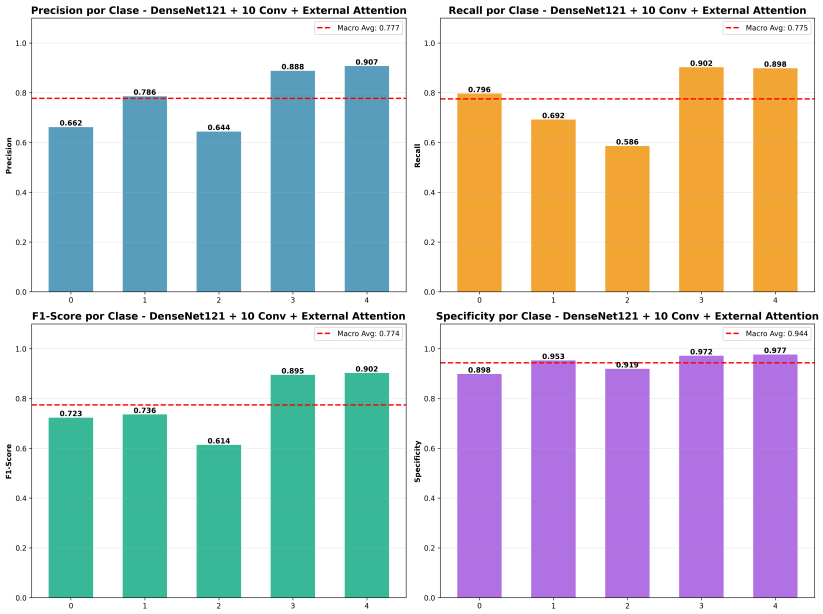


Figura 50. DenseNet-121: Precision/Recall/F1/Specificity por clase (test). Fuente: Autor Propio.

La Figura 51 presenta la información ROC bajo el esquema one-vs-rest (OvR) para cada clase. Esta gráfica permite analizar la capacidad del modelo para separar cada categoría frente al resto bajo distintos umbrales

de decisión, ofreciendo una visión del compromiso entre sensibilidad y especificidad por clase.

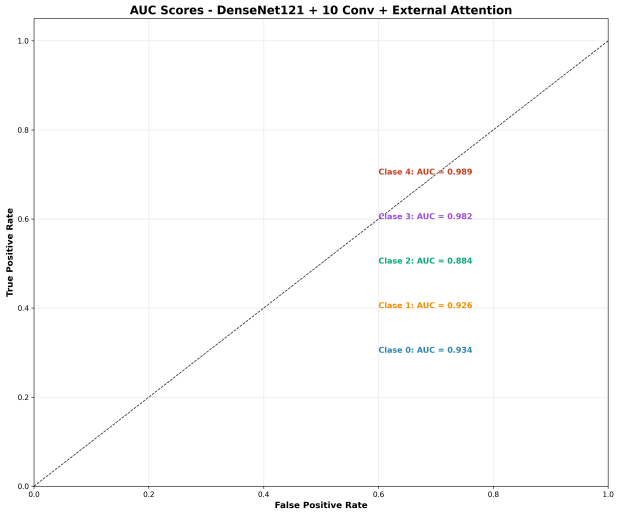


Figura 51. DenseNet-121: información ROC OvR por clase (test). Fuente: Autor Propio.

La Figura 52 muestra el AUC por clase y el promedio macro, cuantificando la capacidad discriminativa del modelo. Se observa un desempeño global elevado, con AUC macro de 0.943, destacando valores mayores en los grados avanzados y una reducción relativa en la clase 2, consistente con las confusiones observadas en la matriz de confusión y las métricas por clase.

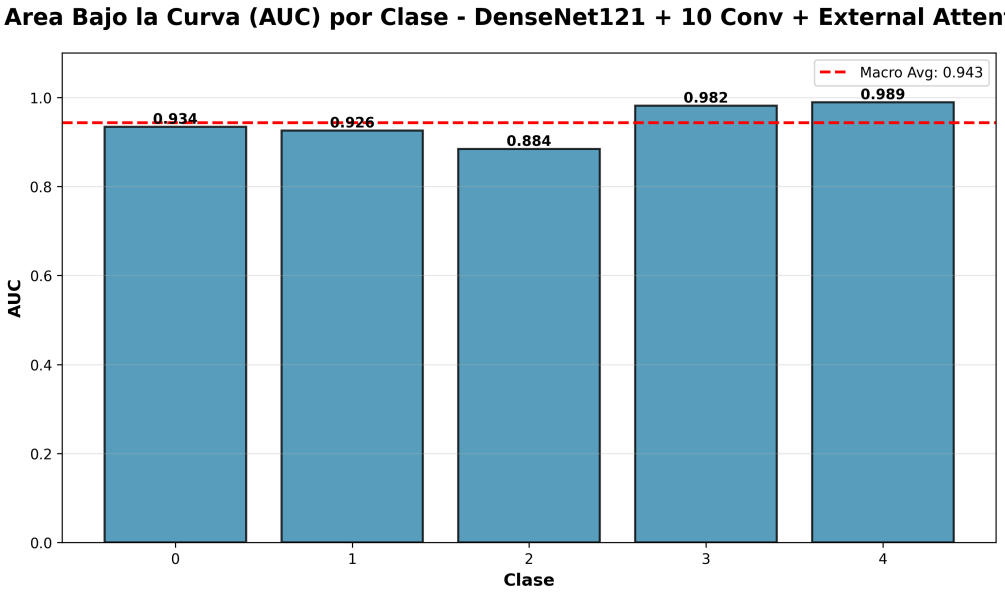


Figura 52. DenseNet-121: AUC por clase y promedio macro (test). Fuente: Autor Propio.

Finalmente, la Figura 53 presenta el análisis del *gap* entre entrenamiento y validación a lo largo de las épocas. Se evidencia un incremento progresivo hacia el final del entrenamiento, alcanzando valores cercanos a 16 %, lo que indica sobreajuste moderado. Esta visualización permite interpretar la generalización del modelo y complementar la lectura de las curvas del historial de entrenamiento.

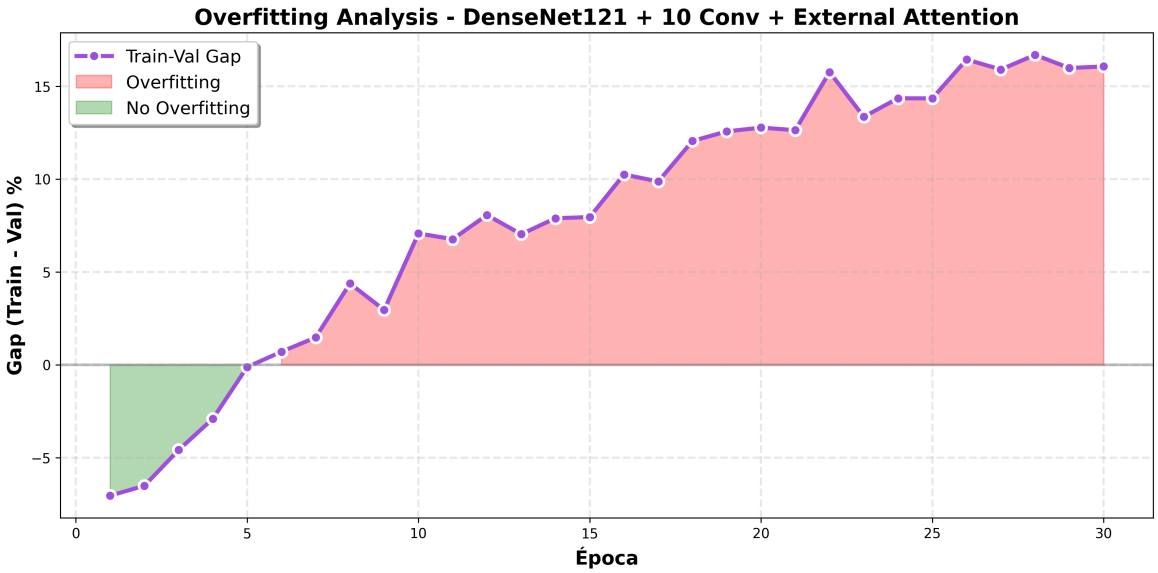


Figura 53. DenseNet-121: análisis del *gap* (% train – % val) por época. Fuente: Autor Propio.

VII-E4. YOLOv8-cls: En modalidad de clasificación YOLOv8-cls alcanzó métricas macro elevadas en el conjunto de prueba, con Precision=0.8201, Recall=0.8188, F1=0.8192, Specificity=0.9547 y AUC macro=0.9620. El análisis por clase mostró un desempeño superior en las clases 3 y 4 (F1 $\approx$ 0.92–0.93), mientras que la clase 2 presentó el mayor nivel de dificultad (F1 $\approx$ 0.69), coherente con la complejidad de distinguir estadios intermedios por patrones sutiles y distribuidos.

La Figura 54 corresponde a la matriz de confusión multiclase en el conjunto de prueba. En ella, los aciertos se observan en la diagonal principal y los errores fuera de esta. Se evidencia que las confusiones se concentran principalmente entre clases contiguas, especialmente alrededor de la clase 2, lo cual es consistente con la dificultad de separar estadios intermedios que comparten rasgos clínicos similares.

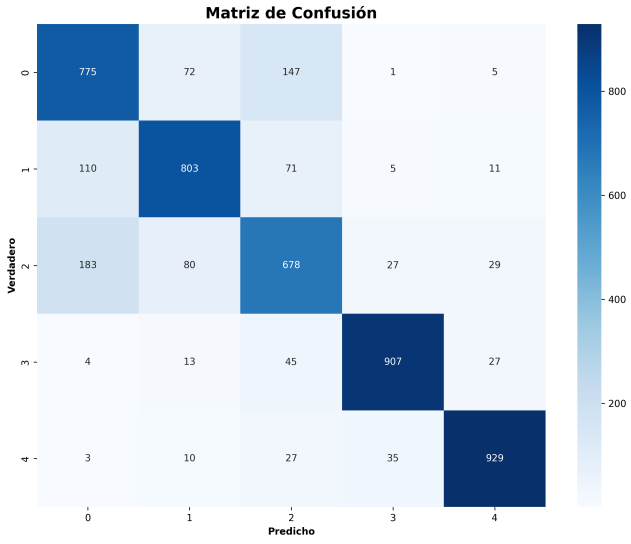


Figura 54. YOLOv8-cls: métricas macro globales (test). Fuente: Autor Propio.

La Figura 55 presenta las métricas por clase (Precision, Recall y F1) para el conjunto de prueba. Esta gráfica

permite comparar el rendimiento del modelo en cada grado de severidad, mostrando que las clases 3 y 4 alcanzan valores más altos, mientras que la clase 2 tiende a registrar los valores más bajos, reflejando mayor complejidad en su discriminación.

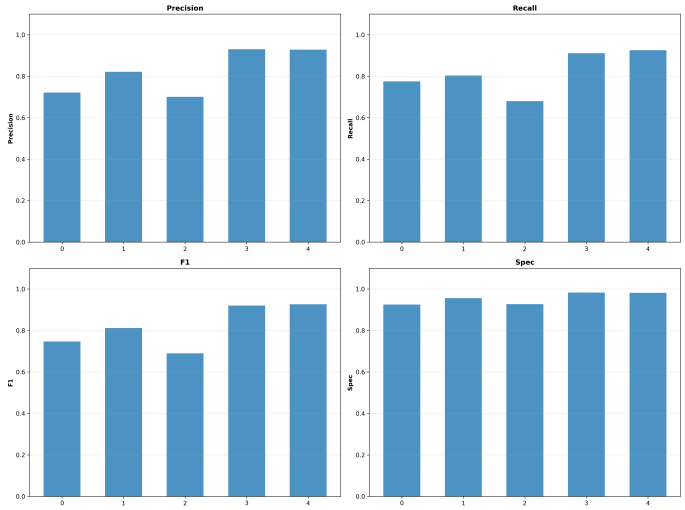


Figura 55. YOLOv8-cls: Precision/Recall/F1 por clase (test). Fuente: Autor Propio.

La Figura 56 muestra las curvas ROC en esquema one-vs-rest (OvR) para cada clase. Estas curvas describen la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos bajo distintos umbrales, permitiendo evaluar la capacidad de separación del modelo para cada categoría frente al resto.

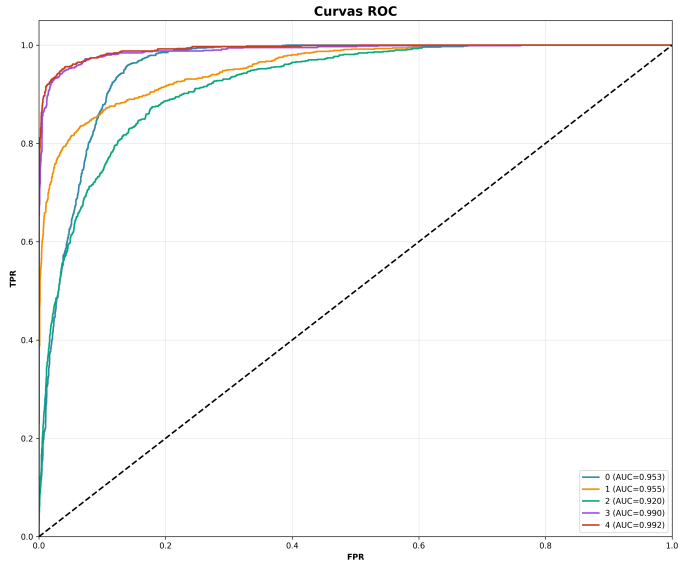


Figura 56. YOLOv8-cls: AUC por clase (test). Fuente: Autor Propio.

La Figura 57 resume el AUC por clase en test, cuantificando la capacidad discriminativa obtenida a partir de las curvas ROC. Se observan valores elevados en todas las clases, con una reducción relativa en la clase 2, lo cual coincide con las confusiones observadas en la matriz de confusión y las métricas por clase.

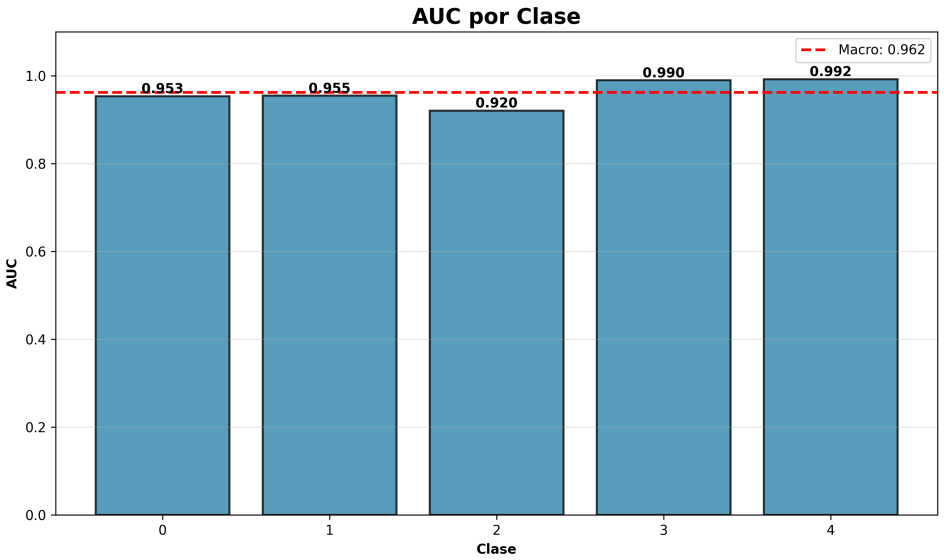


Figura 57. YOLOv8-cls: curvas ROC OvR por clase (test). Fuente: Autor Propio.

La Figura 58 presenta un formato comparativo que integra Precision, Recall, F1 y Specificity por clase, facilitando una lectura conjunta del desempeño en cada grado. Esta visualización refuerza que las clases avanzadas mantienen métricas superiores y que el reto principal se ubica en el grado intermedio.

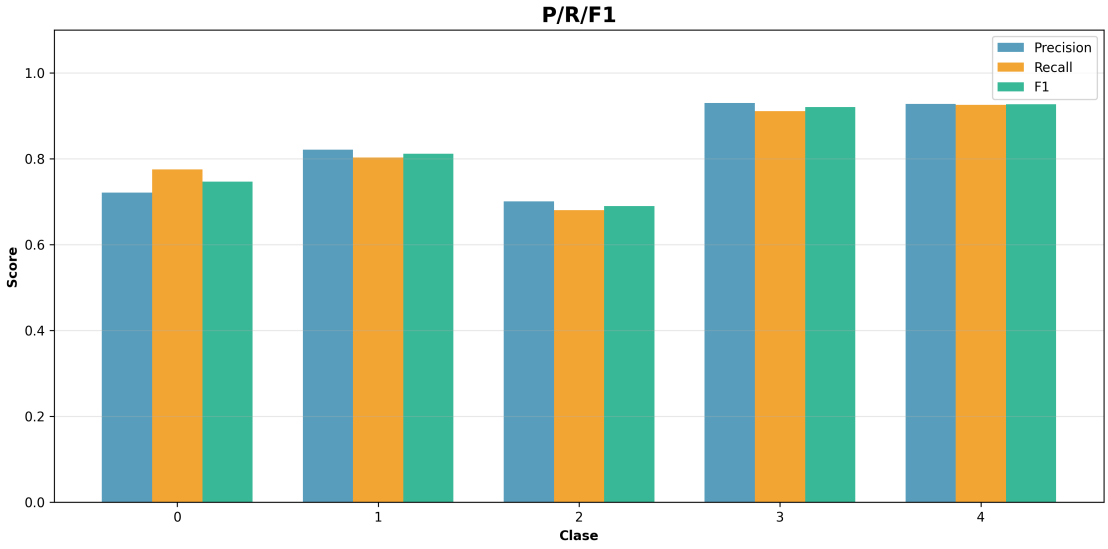


Figura 58. YOLOv8-cls: métricas por clase (Precision/Recall/F1/Specificity) en formato comparativo (test). Fuente: Autor Propio.

Finalmente, la Figura 59 muestra un resumen de métricas macro globales en el conjunto de prueba, proporcionando una visión consolidada del desempeño general del modelo (Precision macro, Recall macro, F1 macro, Specificity macro y AUC macro). Este resumen respalda la alta capacidad discriminativa global observada para YOLOv8-cls.

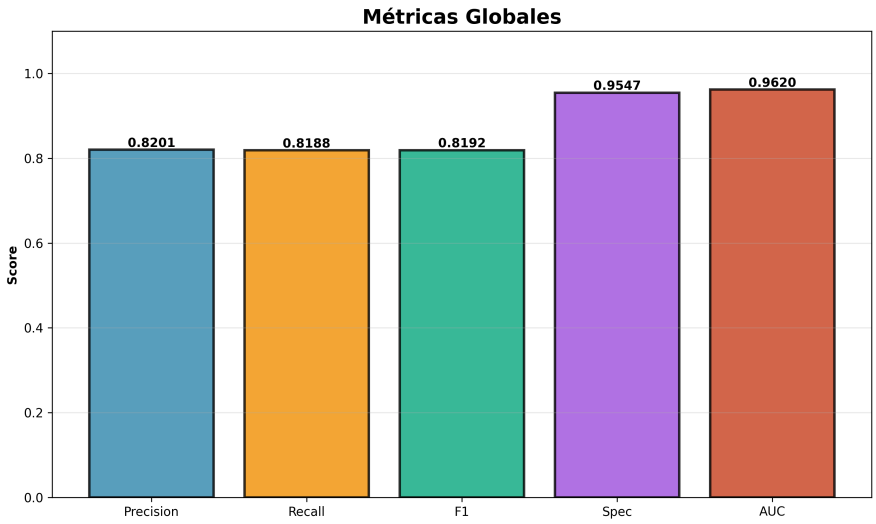


Figura 59. YOLOv8-cls: matriz de confusión multiclase (test). Fuente: Autor Propio.

VII-E5. Vision Transformer (ViT-B/16) + External Attention: Incorpora autoatención multi-cabeza para modelar dependencias globales entre parches, lo que resulta útil cuando la evidencia clínica está distribuida. En términos de dinámica de entrenamiento, este tipo de arquitectura suele requerir más estabilidad en la fase inicial; una vez alcanzado el régimen de convergencia, tiende a consolidar patrones globales relevantes. En este trabajo, se reporta su comportamiento con las mismas seis salidas estandarizadas para asegurar comparabilidad directa.

La Figura 60 muestra el historial de entrenamiento y validación del modelo, con la evolución de *loss* y *accuracy*. Se observa una convergencia progresiva, con estabilización posterior del desempeño en validación. El mejor valor alcanzado fue 76.85 % de *val accuracy*, lo que permite evaluar su comportamiento frente a arquitecturas convolucionales bajo el mismo escenario experimental.

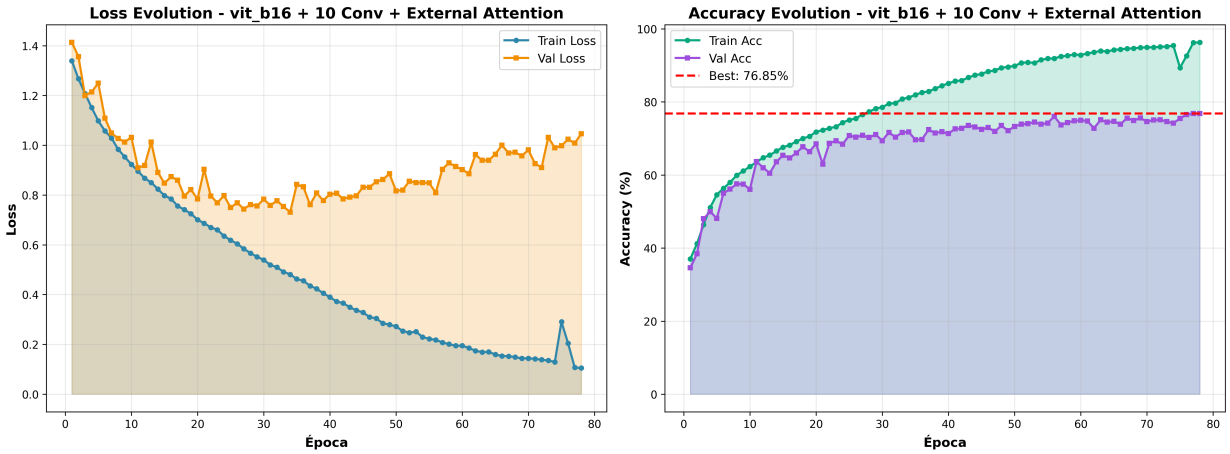


Figura 60. ViT-B/16: evolución de *loss* y *accuracy* en entrenamiento y validación. Fuente: Autor Propio.

La Figura 61 presenta la matriz de confusión multiclase en el conjunto de prueba. En ella se identifican los aciertos sobre la diagonal principal y las confusiones fuera de ella, las cuales se concentran principalmente entre clases contiguas, especialmente alrededor de la clase 2. Este patrón es consistente con la dificultad de

diferenciar estadios intermedios donde los signos clínicos pueden solaparse con grados vecinos.

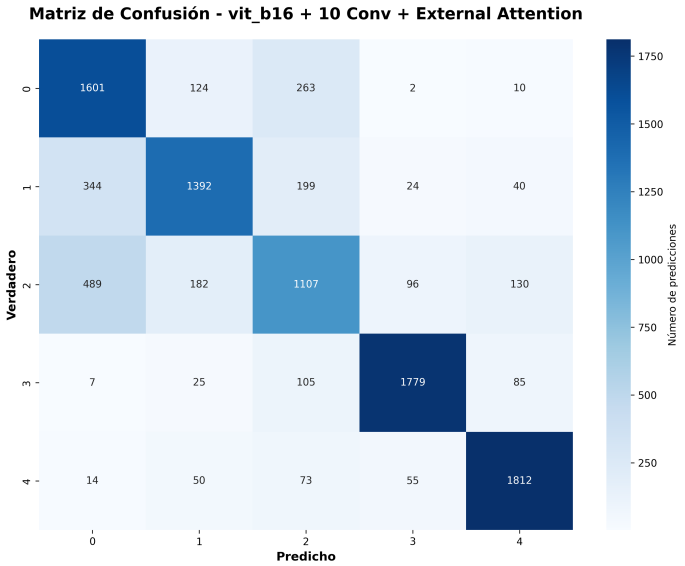


Figura 61. ViT-B/16: matriz de confusión multiclase (test). Fuente: Autor Propio.

La Figura 62 resume las métricas por clase (Precision, Recall, F1 y Specificity) en test. Esta visualización permite comparar el rendimiento por categoría y evidencia un desempeño más alto en clases avanzadas, mientras que la clase intermedia tiende a registrar métricas relativamente menores, manteniendo la misma tendencia observada en otras arquitecturas evaluadas.

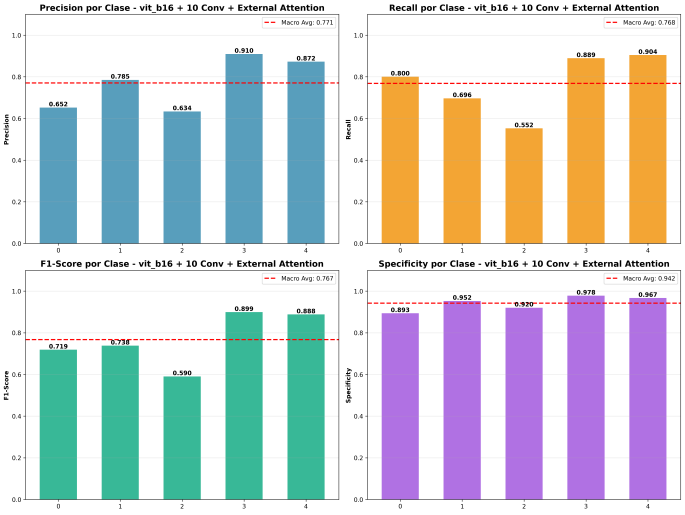


Figura 62. ViT-B/16: Precision/Recall/F1/Specificity por clase (test). Fuente: Autor Propio.

La Figura 63 muestra la información ROC en esquema one-vs-rest (OvR) por clase, permitiendo analizar la separabilidad del modelo bajo distintos umbrales de decisión. Complementariamente, la Figura 64 presenta el AUC por clase y el promedio macro, cuantificando la capacidad discriminativa global del modelo con AUC macro de 0.938, con valores elevados en las clases avanzadas y una reducción relativa en la clase 2.

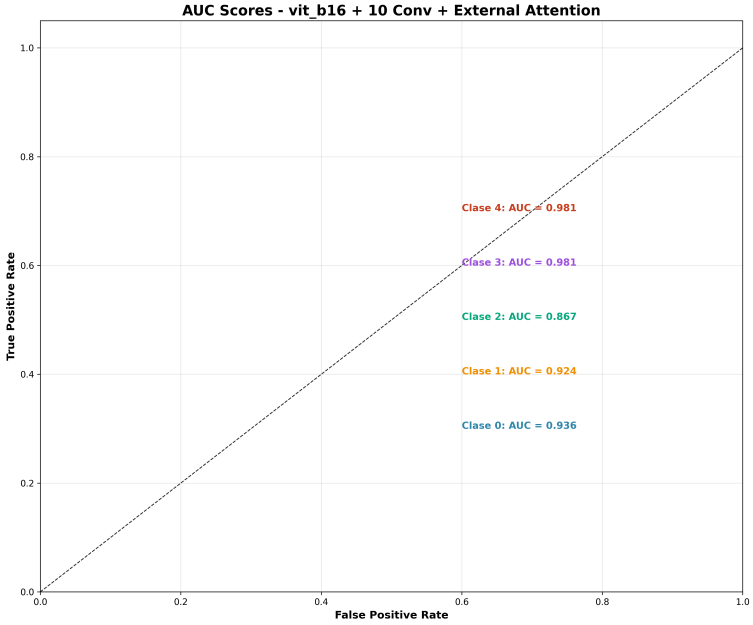


Figura 63. ViT-B/16: información ROC OvR por clase (test). Fuente: Autor Propio.

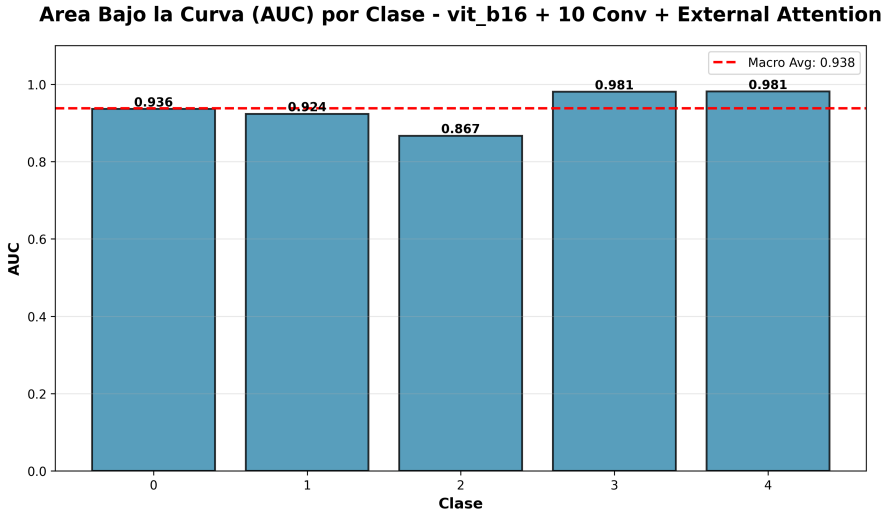


Figura 64. ViT-B/16: AUC por clase y promedio macro (test). Fuente: Autor Propio.

Finalmente, la Figura 65 presenta el análisis del *gap* entre entrenamiento y validación a lo largo de las épocas. Se observa un incremento hacia la fase final con oscilaciones, estabilizándose aproximadamente en el rango 18–21 %, lo que indica sobreajuste. Este resultado sugiere que, si bien ViT-B/16 puede capturar relaciones globales relevantes, su generalización depende de un control cuidadoso del ajuste fino, lo cual se analiza en el apartado comparativo posterior.

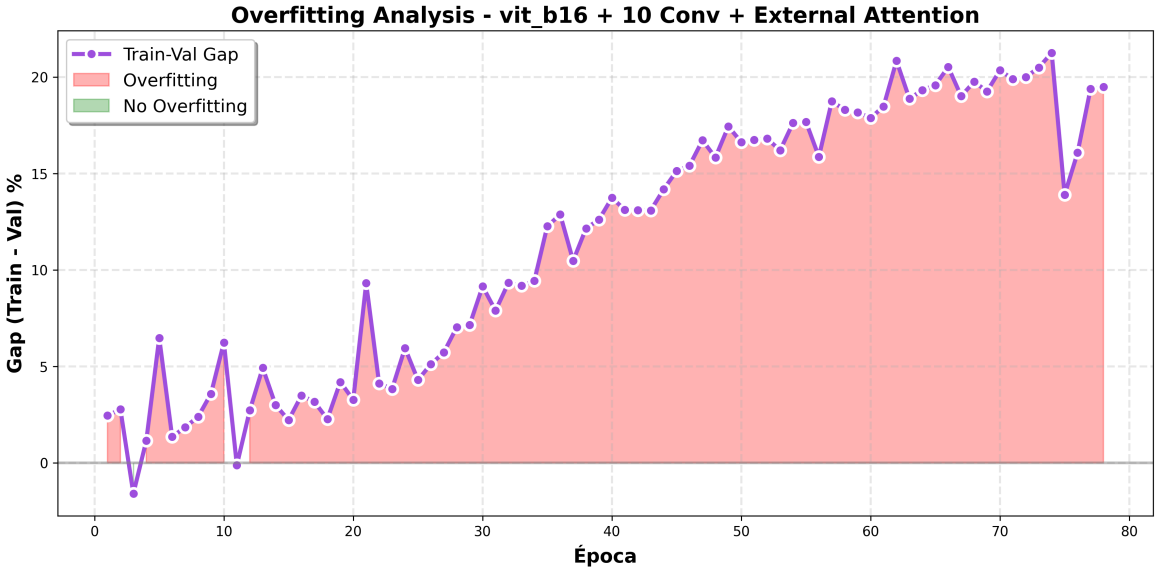


Figura 65. ViT-B/16: análisis del *gap* (% train – % val) por época. Fuente: Autor Propio.

VII-F. Comparación global entre arquitecturas

Para comparar arquitecturas de familias distintas (CNN con transferencia de aprendizaje, YOLOv8-cls y Transformer), se integraron métricas globales y evidencia visual. La comparación se apoya en dos criterios:

1. Rendimiento absoluto en el conjunto de prueba, mediante métricas macro (Precision, Recall, F1, Specificity y AUC macro).

2. Estabilidad/generalización, mediante *val accuracy* y el *gap* train–val cuando se dispone del historial de entrenamiento.

VII-F1. Comparación cuantitativa (métricas globales): La Tabla X resume el desempeño global de los modelos evaluados. En particular, el modelo personalizado (EfficientNet-B0 + External Attention) se contrasta con CNN preentrenadas (ResNet-50 y DenseNet-121), con el Transformer (ViT-B/16) y con YOLOv8-cls en modalidad de clasificación.

Tabla X
COMPARACIÓN DE MÉTRICAS GLOBALES POR ARQUITECTURA (TEST MACRO Y ESTABILIDAD EN VALIDACIÓN). FUENTE: AUTOR PROPIO.

Modelo	Val Acc (%)	Gap (%)	P macro	R macro	F1 macro	Spec macro	AUC macro
EfficientNet-B0 (personalizado) + EA	82.25	16–18	0.827	0.823	0.822	0.956	0.954
ResNet-50 + EA	78.36	19–22	0.791	0.784	0.784	0.946	0.941
DenseNet-121 + EA	77.49	∼16	0.777	0.775	0.774	0.944	0.943
ViT-B/16 + EA	76.85	18–21	0.771	0.768	0.767	0.942	0.938
YOLOv8-cls	–	–	0.8201	0.8188	0.8192	0.9547	0.9620

A partir de la Tabla X, YOLOv8-cls obtuvo el mayor AUC macro (0.9620), mientras que el modelo personalizado basado en EfficientNet-B0 alcanzó el mayor F1 macro entre las arquitecturas con entrenamiento supervisado estándar y presentó una alta especificidad macro (0.956). En general, la menor discriminación se observa en la clase intermedia (Grado 2) en todos los modelos, lo cual es coherente con la naturaleza

multietapa de la retinopatía diabética y el solapamiento visual entre grados contiguos.

VII-F2. Ranking y selección del modelo más eficiente: Para definir el modelo más eficiente se consideró el equilibrio entre rendimiento global (F1 macro y AUC macro) y comportamiento de generalización (menor *gap* y estabilidad en validación), priorizando además una alta especificidad para reducir falsos positivos en la práctica clínica. Con base en estos criterios, se obtiene la siguiente lectura comparativa:

- En desempeño global, YOLOv8-cls presenta el mayor AUC macro (0.9620) y un F1 macro competitivo (0.8192), destacando su capacidad discriminativa global.
- El modelo personalizado EfficientNet-B0 + EA alcanza el mejor balance de métricas macro entre las redes evaluadas (F1 macro=0.822, AUC macro=0.954) y mantiene una estabilidad razonable en validación, aun con sobreajuste moderado (gap 16–18 %).
- ResNet-50 + EA logra un rendimiento competitivo en AUC macro (0.941), pero exhibe el *gap* más alto (19–22 %), lo que sugiere una tendencia mayor a memorizar el conjunto de entrenamiento.
- DenseNet-121 + EA mantiene un desempeño global consistente (AUC macro=0.943) y un comportamiento de entrenamiento regular, con *gap* cercano a 16 %.
- ViT-B/16 + EA muestra desempeño global ligeramente inferior (AUC macro=0.938), y un *gap* elevado (18–21 %), lo que sugiere necesidad de control adicional del ajuste fino para mejorar generalización.

En conjunto, considerando rendimiento macro y estabilidad, el modelo más eficiente dentro del objetivo del estudio (comparación del modelo personalizado frente a modelos preentrenados) es EfficientNet-B0 (personalizado) + External Attention, ya que ofrece el mejor equilibrio entre F1/AUC macro y una estabilidad de entrenamiento comparable a las alternativas, sin requerir una arquitectura más pesada.

VII-G. Análisis de generalización

La generalización se evaluó mediante (I) el *gap* train–val y (II) la inspección visual de las curvas de entrenamiento (*loss/accuracy*). En las CNN y ViT se observó un patrón común: descenso sostenido de *loss* en entrenamiento y estabilización de la validación hacia etapas finales, acompañado de incremento del *gap*.

- DenseNet-121 presentó un *gap* cercano a 16 % al final del entrenamiento, indicando sobreajuste moderado con una trayectoria de aprendizaje regular Figura 53.
- EfficientNet-B0 (modelo personalizado) mostró un *gap* del orden de 16–18 %, manteniendo estabilidad en validación y métricas globales altas en test Figura 41.
- ResNet-50 evidenció el mayor *gap* (19–22 %), lo que sugiere una tendencia más marcada al sobreajuste durante el ajuste fino prolongado Figura 47.
- ViT-B/16 presentó un *gap* alto (18–21 %) con oscilaciones, consistente con mayor sensibilidad del entrenamiento en arquitecturas basadas en atención bajo el mismo conjunto de datos Figura 65.
- Para YOLOv8-cls se prioriza la lectura de desempeño en test (matriz de confusión, métricas por clase y AUC), destacando su robustez global y la mayor dificultad en la clase 2 Figuras 55 y 54.

VII-H. Discusión técnica de hallazgos

Los resultados muestran que el desempeño depende de cómo cada arquitectura representa patrones locales y relaciones globales en retina, así como del grado de estabilidad durante el ajuste fino:

- EfficientNet-B0 (modelo personalizado): Destaca por su equilibrio entre desempeño macro (F1=0.822, AUC=0.954) y estabilidad aceptable, reforzando su rol como propuesta central del estudio frente a modelos preentrenados.

- ResNet-50: Mantiene capacidad discriminativa elevada, pero su mayor *gap* sugiere menor generalización relativa bajo el mismo protocolo de entrenamiento.

- DenseNet-121: Muestra consistencia global y comportamiento regular, siendo una alternativa robusta aunque con métricas macro ligeramente inferiores al modelo personalizado.

- YOLOv8-cls: Logra la mayor AUC macro y alta especificidad, resaltando su capacidad separadora global; la principal debilidad se concentra en el Grado 2, donde se intensifican confusiones con clases vecinas.

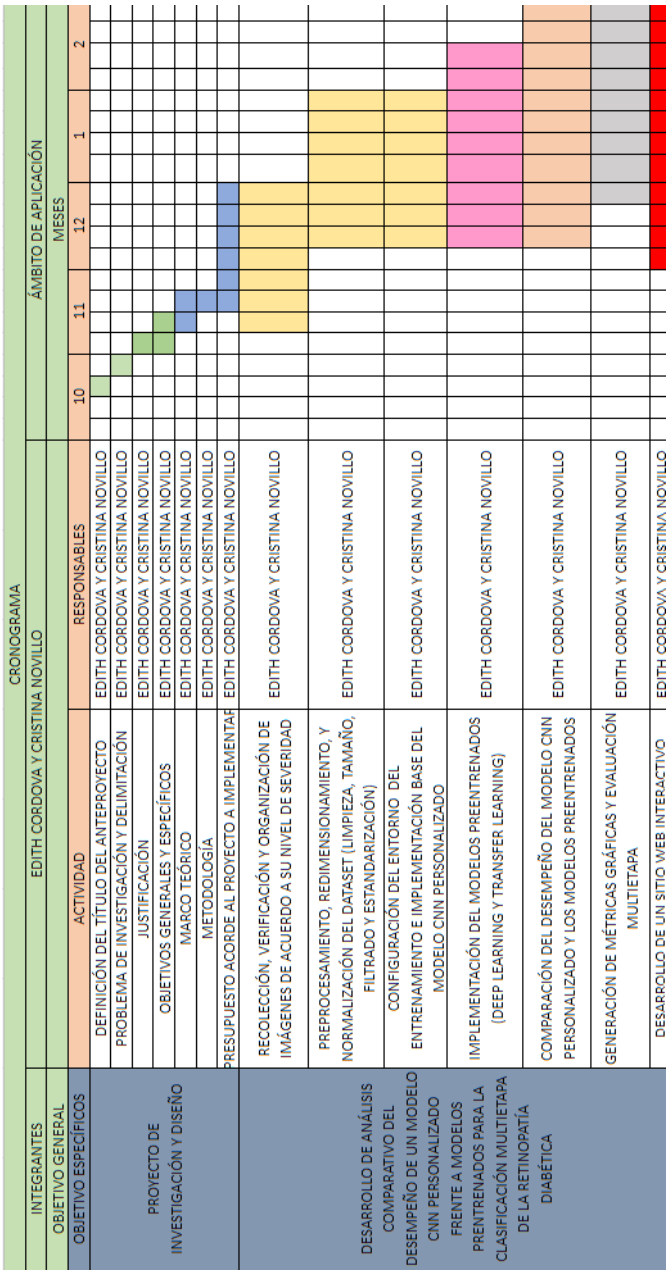
- ViT-B/16: Aporta modelado global por autoatención, pero su rendimiento y *gap* sugieren que requiere estrategias adicionales (regularización o ajustes de entrenamiento) para mejorar generalización bajo el mismo escenario.

En síntesis, los hallazgos describen tres perfiles: (I) mejor balance global y propuesta objetivo del estudio: EfficientNet-B0 (personalizado), (II) alta discriminación global: YOLOv8-cls, y (III) alternativas convolucionales robustas: ResNet-50 y DenseNet-121, con diferentes compromisos entre capacidad y generalización; ViT-B/16 complementa el análisis al introducir atención global como mecanismo principal.

VIII. CRONOGRAMA

A continuación se muestra el cronograma de trabajo en la siguiente tabla XI.

Tabla XI
CRONOGRAMA



IX. PRESUPUESTO

Tabla XII
PRESUPUESTO

Sección	**Componente**	**Cantidad**	**Valor unitario (USD)**	**Valor total (USD)**
Recursos computacionales	Servidor en la nube (GPU básica)	3 Meses	40.00	120.00
	Google Colab Pro / GPU	4 Meses	20.00	80.00
	Inteligencia Artificial de Nubes	3 Meses	69.00	207.00
Almacenamiento	Google Drive Adicional (100–200 GB)	3 Meses	12.00	36.00
Gestión del Dataset	Roboflow (Plan Básico Temporal)	3 Meses	25.00	75.00
Datos para entrenamiento	Dataset público de Kaggle (EyePACS, APTOS, Medisor)	3	0.00	0.00
Desarrollo del sistema	Desarrollo Frontend	1	30.00	30.00
	Desarrollo Backend + API	1	40.00	40.00
Administrativo	Impresiones	4 Meses	15.00	60.00
Despliegue web	Hosting web	1	60.00	60.00
	Dominio web	1	70.00	70.00
Software básico	Frameworks open source (PyTorch, OpenCV, etc.)	1	200.00	200.00
TOTAL				**978.00**

X. Conclusiones

El conjunto de datos es robusto, debido que la integración de múltiples repositorios y su unificación mediante Robloflow, logró equilibrar la misma cantidad de imágenes entre las etapas tempranas y avanzadas de la Retinopatía Diabética, lo que fortalece la representatividad clínica del modelo; no obstante, dado que las imágenes provienen de diferentes fuentes, se aplicaron técnicas de curación, preprocesamiento, redimensionamiento y normalización para identificar diferencias en iluminación, descartar imágenes borrosas, y repetidas, ajustar del tamaño, para posteriormente verificar las etiquetas de entrada, lo que garantiza un entrenamiento estable y que los resultados obtenidos sean reflejo solamente de la capacidad discriminativa propia de los modelos y no de las inconsistencias en las entradas del dataset.

El desarrollo del modelo personalizado se basó en EfficientNet-B0, por su equilibrio óptimo entre rendimiento, número de parámetros (5,3 millones) y eficiencia computacional (0,39 mil millones de FLOPs), gracias a su estrategia compound scaling y su aplicación en tareas de clasificación de imágenes médicas, proporcionó una base robusta en la que incorporando capas convolucionales adicionales y un módulo de *External Attention* se reforzó la selección de características relevantes en un problema clínico donde las diferencias entre etapas contiguas pueden ser sutiles. Este modelo alcanzó 82.25 % de *val accuracy* y, en el conjunto de prueba, obtuvo métricas macro de Precision=0.827, Recall=0.823, F1=0.822, Specificity=0.956 y AUC macro=0.954.

La comparación con modelos preentrenados evidenció que ResNet-50 (F1 macro=0.784, AUC macro=0.941) y DenseNet-121 (F1 macro=0.774, AUC macro=0.943) presentaron resultados competitivos, mientras que ViT-B/16 obtuvo un desempeño ligeramente inferior (F1 macro=0.767, AUC macro=0.938). En paralelo, YOLOv8-cls alcanzó métricas macro elevadas (Precision=0.8201, Recall=0.8188, F1=0.8192, Specificity=0.9547 y AUC macro=0.9620), destacando su alta capacidad discriminativa global. No obstante, al considerar el propósito central del estudio, EfficientNet-B0 personalizado se posiciona como la alternativa más conveniente por su equilibrio entre F1/AUC macro y su comportamiento general consistente frente al resto de CNN evaluadas. En términos de generalización, el análisis del *gap* train–val mostró sobreajuste moderado en todos los modelos con ajuste fino, con valores aproximados de 16–18 % en EfficientNet-B0, $\sim$16 % en DenseNet-121, 18–21 % en ViT-B/16 y 19–22 % en ResNet-50. Estos resultados confirman que, en imágenes médicas, la selección del modelo no debe basarse únicamente en la exactitud máxima, sino en el equilibrio entre desempeño global, estabilidad del entrenamiento y capacidad de mantener resultados consistentes al evaluar datos no vistos.

A pesar que durante el despliegue se evidenció limitaciones con respecto a los altos costos de suscripción y el cobro por imagen procesada, lo que restringió la ejecución prolongada de pruebas y la expansión del entorno digital, la implementación del algoritmo a la plataforma interactiva RetinaVisionAI mediante la infraestructura Railway, facilitó la conexión eficiente entre el frontend y el Backend, lo que permitió gestionar los modelos y garantizar su despliegue escalable y automatizado en la nube donde por medio de un proceso estructurado de inferencia multiclase, promovió la carga de retinografías en conjunto con la visualización de la predicción y sus probabilidades asociadas.

XI. Recomendaciones

Se recomienda fortalecer la robustez del sistema mediante la incorporación de conjuntos de datos externos y heterogéneos, considerando principalmente imágenes adquiridas en diferentes condiciones (equipos, resoluciones, iluminación y protocolos de captura), lo que permitirá estimar con mayor precisión la capacidad de generalización del modelo para la clasificación multietapa de retinopatía diabética, así como identificar posibles sesgos que provienen de los repositorios originales y de la variabilidad inherente a las retinografías.

Durante el desarrollo se usó un conjunto de datos consolidado a partir de la unión de diferentes repositorios públicos, y el aumento de datos se realizó de forma manual. Por ello, se recomienda implementar en trabajos futuros un pipeline automatizado de curación y aumento de datos que integre rotaciones controladas, cambios de brillo/contraste, recortes y variaciones de escala que permita estandarizar el proceso, garantizar trazabilidad y facilitar la replicabilidad del experimento. Asimismo, se sugiere documentar de forma explícita la procedencia de cada subconjunto y mantener un registro de versiones del dataset resultante para garantizar consistencia entre entrenamientos.

Adicionalmente, se recomienda incorporar estrategias de control del sobreajuste que no pudieron aplicarse de manera completa durante el proceso, como *early stopping* y los ajustes finos de hiperparámetros (tasa de aprendizaje, tamaño de lote, regularización y programación del aprendizaje). Estas técnicas podrían reducir la brecha observada entre el subconjunto train y val, mejorando la estabilidad durante el entrenamiento y aumentando la robustez del modelo ante datos no vistos, especialmente en la clase intermedia (Grado 2), donde se concentran la mayoría de confusiones.

De manera complementaria, se recomienda que en futuras versiones de la plataforma se use una infraestructura más económica que Railway, como Hugging Face Spaces o Streamlit Community Cloiud, que ofrecen planes gratuitos o de bajo costo orientados a prototipos y proyectos académicos. Este tipo de infraestructuras facilita el despliegue, la administración y el mantenimiento continuo de la plataforma, haciéndolas adecuadas para pruebas, demostraciones y ajustes de los modelos.

Referencias

[1] J. B. de la Puente, «La Retinopatía Diabética como predictor de morbimortalidad cardiovascular en personas con diabetes tipo 2,»

[2] A. Grzybowski, P. Brona, T. Krzywicki, M. Gaca-Wysocka, A. Berlińska y A. Świech, «Variability of Grading DR Screening Images among Non-Trained Retina Specialists,» *Journal of Clinical Medicine*, vol. 11, n.º 11, 2022, ISSN: 2077-0383. DOI: 10.3390/jcm11113125. dirección: https://www.mdpi.com/2077-0383/11/11/3125.

[3] A. Brant et al., «Performance of a Deep Learning Diabetic Retinopathy Algorithm in India,» *JAMA Network Open*, vol. 8, n.º 3, e250984-e250984, mar. de 2025, ISSN: 2574-3805. DOI: 10.1001/jamanetworkopen.2025.0984. eprint: https://jamanetwork.com/journals/jamanetworkopen/articlepdf/2831702/brant_2025_oi_250073_1744295100.15536.pdf. dirección: https://doi.org/10.1001/jamanetworkopen.2025.0984.

[4] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj y D. J. Inman, «1D convolutional neural networks and applications: A survey,» *Mechanical Systems and Signal Processing*, vol. 151, pág. 107 398, 2021, ISSN: 0888-3270. DOI: https://doi.org/10.1016/j.ymssp.2020.107398. dirección: https://www.sciencedirect.com/science/article/pii/S0888327020307846.

[5] N. E. Institute, *Retinopatía diabética — National Eye Institute*. dirección: https://www.nei.nih.gov/espanol/aprenda-sobre-la-salud-ocular/enfermedades-y-afecciones-de-los-ojos/retinopatia-diabetica.

[6] G. Tenorio y V. Ramírez-Sánchez, *Retinopatía diabética; conceptos actuales*, jul. de 2010. dirección: https://www.elsevier.es/en-revista-revista-medica-del-hospital-general-325-articulo-retinopatia-diabetica-conceptos-actuales-X0185106310902843.

[7] V. O. S. Catherine et al., *FACTORES DE RIESGO QUE INCIDEN EN RETINOPATÍA DIABÉTICA NO PROLIFERATIVA 2018*. dirección: http://scielo.senescyt.gob.ec/scielo.php?script=sci_arttext&pid=S2528-79072019000300058.

[8] M. M. Alfaro y M. G. Vargas, «Enfoque Integral en la Retinopatía Diabética: Desde la Prevención hasta el Tratamiento,» *Enfoque*, 2025.

[9] S. Mehta, *Retinopatía diabética*, mayo de 2024. dirección: https://www.msdmanuals.com/es/professional/trastornos-oft%C3%A1lmicos/enfermedades-retinianas/retinopat%C3%ADa-diab%C3%A9tica#Conceptos-clave_v7825060_es.

[10] R. De Jesús Palencia Vizcarra Rodolfo Palencia Díaz, *La era de la inteligencia artificial en la práctica médica – Medicina Interna de México*. dirección: https://medicinainterna.org.mx/article/la-era-de-la-inteligencia-artificial-en-la-practica-medica/.

[11] R. Yamashita, M. Nishio, R. K. Gian DO y K. Togashi, «Convolutional neural networks: an overview and application in radiology,» *Insights into Imaging*, vol. 9, n.º 4, págs. 611-629, jun. de 2018. DOI: 10.1007/s13244-018-0639-9. dirección: https://doi.org/10.1007/s13244-018-0639-9.

[12] H. Hashemian et al., «Application of Artificial Intelligence in Ophthalmology: An Updated Comprehensive Review,» *Journal of Ophthalmic and Vision Research*, vol. 19, n.º 3, págs. 354-367, sep. de 2024. DOI: 10.18502/jovr.v19i3.15893. dirección: https://pmc.ncbi.nlm.nih.gov/articles/PMC11444002/.

[13] C. Miller, T. Portlock, D. M. Nyaga y J. M. O'Sullivan, «A review of model evaluation metrics for machine learning in genetics and genomics,» *Frontiers in Bioinformatics*, vol. 4, pág. 1 457 619, sep. de 2024. DOI: 10.3389/fbinf.2024.1457619. dirección: https://doi.org/10.3389/fbinf.2024.1457619.

[14] A. Bonnet, *Accuracy vs. Precision vs. Recall in Machine Learning: What is the Difference?* Sep. de 2025. dirección: https://encord.com/blog/classification-metrics-accuracy-precision-recall/.

[15] *Clasificación: Exactitud, recuperación, precisión y métricas relacionadas*. dirección: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall?hl=es-419#:~:text=La%20puntuaci%C3%B3n%20F1%20es%20la,la%20precisi%C3%B3n%20y%20la%20recuperaci%C3%B3n.&text=Esta%20m%C3%A9trica%20equilibra%20la%20importancia,datos%20con%20desequilibrio%20de%20clases..

[16] F. C. Oettl et al., «A practical guide to the implementation of AI in orthopaedic research, Part 6: How to evaluate the performance of AI research?» *Journal of Experimental Orthopaedics*, vol. 11, n.º 3, e12039, mayo de 2024. DOI: 10.1002/jeo2.12039. dirección: https://doi.org/10.1002/jeo2.12039.

[17] S. Swaminathan y B. R. Tantri, «Confusion Matrix-Based Performance Evaluation Metrics,» *African Journal of Biomedical Research*, vol. 27, págs. 4023-4031, nov. de 2024. DOI: 10.53555/AJBR.v27i4S.4345.

[18] J. Li, «Area under the ROC Curve has the most consistent evaluation for binary classification,» *PLOS ONE*, vol. 19, dic. de 2024. DOI: 10.1371/journal.pone.0316019.

[19] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[20] G. Ola, J. Tyler y J. Nelson, «Introduction to Supervised Learning,» mar. de 2025.

[21] Y. LeCun, Y. Bengio y G. Hinton, «Deep learning,» *Nature*, vol. 521, n.º 7553, págs. 436-444, mayo de 2015. DOI: 10.1038/nature14539. dirección: https://doi.org/10.1038/nature14539.

[22] A. Solano, K. N. Dietrich, M. Martínez-Sober, R. Barranquero-Cardeñosa, J. Vila-Tomás y P. Hernández-Cámara, «Deep Learning Architectures for Diagnosis of Diabetic Retinopathy,» *Applied Sciences*, vol. 13, n.º 7, 2023, ISSN: 2076-3417. DOI: 10.3390/app13074445. dirección: https://www.mdpi.com/2076-3417/13/7/4445.

[23] B. M. Communications, *Detection of Diabetic Retinopathy Using Deep Learning Analysis - Retina Today*. dirección: https://retinatoday.com/articles/2021-sept/detection-of-diabetic-retinopathy-using-deep-learning-analysis.

[24] J. Holdsworth y M. Scapicchio, *Aprendizaje profundo*, jul. de 2025. dirección: https://www.ibm.com/mx-es/think/topics/deep-learning.

[25] M. Vakalopoulou, S. Christodoulidis, N. Burgos, O. Colliot y V. Lepetit, *Deep Learning: Basics and Convolutional Neural Networks (CNNs)*. ene. de 2023, págs. 77-115. DOI: 10.1007/978-1-0716-3195-9\{_}3. dirección: https://www.ncbi.nlm.nih.gov/books/NBK597497/?.

[26] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci y M. Parmar, «A review of convolutional neural networks in computer vision,» *Artificial Intelligence Review*, vol. 57, n.º 4, mar. de 2024. DOI: 10.1007/s10462-024-10721-6. dirección: https://doi.org/10.1007/s10462-024-10721-6.

[27] O. İrsoy y E. Alpaydın, «Dropout regularization in hierarchical mixture of experts,» *Neurocomputing*, vol. 419, págs. 148-156, 2021, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.08.052. dirección: https://www.sciencedirect.com/science/article/pii/S0925231220313321.

[28] L. Yang, J. Zhang, J. Shenouda, D. Papailiopoulos, K. Lee y R. D. Nowak, *PathProx: A Proximal Gradient Algorithm for Weight Decay Regularized Deep Neural Networks*, oct. de 2022. dirección: https://arxiv.org/abs/2210.03069?.

[29] H. Peng, Y. Yu y S. Yu, «Re-Thinking the Effectiveness of Batch Normalization and Beyond,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, n.º 1, págs. 465-478, sep. de 2023. DOI: 10.1109/tpami.2023.3319005. dirección: https://doi.org/10.1109/tpami.2023.3319005.

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

[31] Y. Saadna, S. Mezzoudj y M. Khelifa, «Efficient Transformer Architectures for Diabetic Retinopathy Classification from Fundus Images: DR-MobileViT, DR-EfficientFormer, and DR-SwinTiny,» *Informatica*, vol. 49, n.º 29, jul. de 2025. DOI: 10.31449/inf.v49i29.8695. dirección: https://www.informatica.si/index.php/informatica/article/view/8695?.

[32] S. J. Pan y Q. Yang, «A Survey on Transfer Learning,» *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, n.º 10, págs. 1345-1359, oct. de 2009. DOI: 10.1109/tkde.2009.191. dirección: https://doi.org/10.1109/tkde.2009.191.

[33] J. Yosinski, J. Clune, Y. Bengio y H. Lipson, «How transferable are features in deep neural networks?» *arXiv (Cornell University)*, vol. 27, págs. 3320-3328, nov. de 2014. DOI: 10.48550/arxiv.1411.1792. dirección: http://arxiv.org/abs/1411.1792.

[34] J.-H. Lee, D. Yoon, B. Ji, K. Kim y S. Hwang, *Rethinking Evaluation Protocols of Visual Representations Learned via Self-supervised Learning*, abr. de 2023. dirección: https://arxiv.org/abs/2304.03456?utm_source=chatgpt.com.

[35] A. Kumar, A. Raghunathan, R. Jones, T. Ma y P. Liang, *Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution*, feb. de 2022. dirección: https://arxiv.org/abs/2202.10054?.

[36] Z. Han, C. Gao, J. Liu, J. Zhang y S. Q. Zhang, «Parameter-efficient fine-tuning for large models: A comprehensive survey,» *arXiv preprint arXiv:2403.14608*, 2024.

[37] V. Gulshan et al., «Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs,» *JAMA*, vol. 316, n.º 22, págs. 2402-2410, dic. de 2016, ISSN: 0098-7484. DOI: 10.1001/jama.2016.17216. eprint: https://jamanetwork.com/journals/jama/articlepdf/2588763/joi160132.pdf. dirección: https://doi.org/10.1001/jama.2016.17216.

[38] I. D. Mienye, T. G. Swart, G. Obaido, M. Jordan y P. Ilono, «Deep Convolutional Neural Networks: A Comprehensive Review,» *Preprints*, 2024. DOI: 10.20944/preprints202408.1288.v1. dirección: https://doi.org/10.20944/preprints202408.1288.v1.

[39] I. Kandel y M. Castelli, «Transfer Learning with Convolutional Neural Networks for Diabetic Retinopathy Image Classification. A Review,» *Applied Sciences*, vol. 10, n.º 6, pág. 2021, mar. de 2020. DOI: 10.3390/app10062021. dirección: https://doi.org/10.3390/app10062021.

[40] X. Long et al., «EfficientNetB0-Based End-to-End Diagnostic System for Diabetic Retinopathy Grading and Macular Edema Detection,» *Diabetes Metabolic Syndrome and Obesity*, vol. Volume 18, págs. 1311-1321, abr. de 2025. DOI: 10.2147/dmso.s506494. dirección: https://pmc.ncbi.nlm.nih.gov/articles/PMC12042962/#abstract2.

[41] M. Feng, Y. Cai y S. Yan, «Enhanced ResNet50 for Diabetic Retinopathy Classification: External Attention and Modified Residual Branch,» *Mathematics*, vol. 13, n.º 10, 2025, ISSN: 2227-7390. DOI: 10.3390/math13101557. dirección: https://www.mdpi.com/2227-7390/13/10/1557.

[42] R. R. Oraá, C. P. M. Zamarro, M. García, J. O. Pérez, M. I. L. Gálvez y R. H. Sánchez, *Detección automática de patología en imágenes de fondo de ojo utilizando técnicas de deep learning*, 2020. dirección: https://dialnet.unirioja.es/servlet/articulo?codigo=8208056.

[43] C. Moura, P. Cortez, D. Assis, P. Motta y B. Silva, «YOLOv8 Deep Learning Model for Diabetic Retinopathy Fundus Image Segmentation and Disease Classification,» ene. de 2024, págs. 1-7. DOI: 10.21528/CBIC2023-159.

[44] N. Rizzieri, L. Dall'Asta y M. Ozoliņš, «Diabetic Retinopathy Features Segmentation without Coding Experience with Computer Vision Models YOLOv8 and YOLOv9,» *Vision*, vol. 8, n.º 3, 2024, ISSN: 2411-5150. DOI: 10.3390/vision8030048. dirección: https://www.mdpi.com/2411-5150/8/3/48.

[45] S. Liu, F. Shao, W. Chu, J. Dai y H. Zhang, «An Improved YOLOv8-Based Lightweight Attention Mechanism for Cross-Scale Feature Fusion,» *Remote Sensing*, vol. 17, n.º 6, 2025, ISSN: 2072-4292. DOI: 10.3390/rs17061044. dirección: https://www.mdpi.com/2072-4292/17/6/1044.

[46] J. de La Torre, «Transformadores: Fundamentos teóricos y aplicaciones,» *arXiv preprint arXiv:2302.09327*, 2023.

[47] *Vision Transformer: procesamiento de imágenes*. dirección: https://www.innovatiana.com/es/post/vision-transformer.

[48] O. Russakovsky et al., «ImageNet Large Scale Visual Recognition Challenge,» *International Journal of Computer Vision*, vol. 115, n.º 3, págs. 211-252, abr. de 2015. DOI: 10.1007/s11263-015-0816-y. dirección: https://doi.org/10.1007/s11263-015-0816-y.

[49] F. M. Dejene, T. G. Debelee, F. Schwenker, Y. M. Ayano y D. W. Feyisa, «Diabetic retinopathy screening using machine learning: a systematic review,» *BMC Biomedical Engineering*, vol. 7, n.º 1, pág. 12, sep. de 2025. DOI: 10.1186/s42490-025-00098-0. dirección: https://doi.org/10.1186/s42490-025-00098-0.

[50] J. Deng, W. Dong, R. Socher, L.-J. Li, N. K. Li y N. L. Fei-Fei, «ImageNet: A large-scale hierarchical image database,» *2009 IEEE Conference on Computer Vision and Pattern Recognition*, págs. 248-255, jun. de 2009. DOI: 10.1109/cvpr.2009.5206848. dirección: https://doi.org/10.1109/cvpr.2009.5206848.

[51] A. Krizhevsky, I. Sutskever y G. E. Hinton, «ImageNet classification with deep convolutional neural networks,» *Communications of the ACM*, vol. 60, n.º 6, págs. 84-90, mayo de 2017. DOI: 10.1145/3065386. dirección: https://doi.org/10.1145/3065386.

[52] O. A. M. Enríquez, «El derecho de protección de datos personales en los sistemas de inteligencia artificial,» *REVISTA IUS*, vol. 15, n.º 48, jun. de 2021. DOI: 10.35487/rius.v15i48.2021.743. dirección: https://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S1870-21472021000200179.

[53] P. Alvear, V. González, L. Ajila y H. Segarra, «Aplicación de la Ley Orgánica de Protección de Datos Personales en el sistema empresarial privado ecuatoriano,» *Revista Lex*, vol. 7, págs. 567-582, jul. de 2024. DOI: 10.33996/revistalex.v7i25.201.

[54] *TRATAMIENTO DE DATOS PERSONALES – Instituto Nacional de Estadística y Censos*. dirección: https://www.ecuadorencifras.gob.ec/institucional/tratamiento-de-datos-personales/.

[55] *ARCSA-DE-2023-016-AKRG Expídese la Normativa técnica sustitutiva para el control y funcionamiento del Sistema Nacional de Tecnovigilancia (SNTV) — Ecuador - Guía Oficial de Trámites y Servicios*. dirección: https://www.gob.ec/regulaciones/arcsa-2023-016-akrg-expidese-normativa-tecnica-sustitutiva-control-funcionamiento-sistema-nacional-tecnovigilancia-sntv.

[56] F. Kitsios, E. Chatzidimitriou y M. Kamariotou, «The ISO/IEC 27001 Information Security Management Standard: How to Extract Value from Data in the IT Sector,» *Sustainability*, vol. 15, n.º 7, 2023, ISSN: 2071-1050. DOI: 10.3390/su15075828. dirección: https://www.mdpi.com/2071-1050/15/7/5828.

[57] C. S. Lazo y B. Correa, *Estándares de Ciberseguridad Aplicables a los Sistemas Informáticos Sanitarios para Proteger los Datos Personales*, 2024. dirección: https://dialnet.unirioja.es/servlet/articulo?codigo=9262999.

[58] J. A. V. Cordero, *Las normas ISO/IEC como mecanismos de responsabilidad proactiva en el Reglamento General de Protección de Datos*, 2021. dirección: https://dialnet.unirioja.es/servlet/articulo?codigo=8222608.

[59] E. S. F. I. (ESFI), *The National Electrical Code (NEC) - Electrical Safety Foundation International*, jun. de 2023. dirección: https://www.esfi.org/workplace-safety/industry-codes-regulations/the-national-electrical-code-nec/.

[60] *NORMAS NFPA PARA DATA CENTER: NFPA 75 y 76*, mayo de 2020. dirección: https://eduardovillafuerteblog.wordpress.com/2017/12/18/normas-nfpa-para-data-center-nfpa-75-y-76/.

[61] F. Argüello, *NFPA 99 Código de Instalaciones de Atención Médica - Infoteknico*, dic. de 2025. dirección: https://www.infoteknico.com/nfpa-99/.

[62] A. d. l. C. Cardet Sarduy y J. d. P. Cárdenas Cobo, «Evolución normativa de la protección de datos personales en Ecuador: un análisis histórico,» *Informática y Sistemas: Revista de Tecnologías de la Informática y las Comunicaciones*, vol. 8, n.º 2, págs. 86-101, 2024. DOI: 10.33936/isrtic.v8.i2.6978. dirección: https://revistas.utm.edu.ec/index.php/Informaticaysistemas/article/download/6978/9149/34982.

[63] E. Koulierakis, «Certification as guidance for data protection by design,» *International Review of Law Computers Technology*, vol. 38, n.º 2, págs. 245-263, oct. de 2023. DOI: 10.1080/13600869.2023.2269498. dirección: https://doi.org/10.1080/13600869.2023.2269498.

[64] M. T. Sarker, G. Ramasamy, M. A. Qwaid, M. S. Hossen y M. G. Sadeque, «AI-driven smart grid optimization for hospital energy systems integrating renewable generation, predictive maintenance, and resilient infrastructure,» *Scientific Reports*, vol. 15, n.º 1, pág. 44 787, dic. de 2025. DOI: 10.1038/s41598-025-28907-5. dirección: https://doi.org/10.1038/s41598-025-28907-5.

XII. ANEXOS

XII-A. Capturas del sitio web RetinaVision AI

La presente sección reúne las capturas de pantalla correspondientes a la interfaz inicial del sitio web *RetinaVision AI*, desarrollado como parte del trabajo de titulación. Estas evidencias permiten visualizar la estructura general de la plataforma, su enfoque informativo sobre la retinopatía diabética y la organización de los apartados orientados a la explicación de la enfermedad y sus etapas clínicas.

—

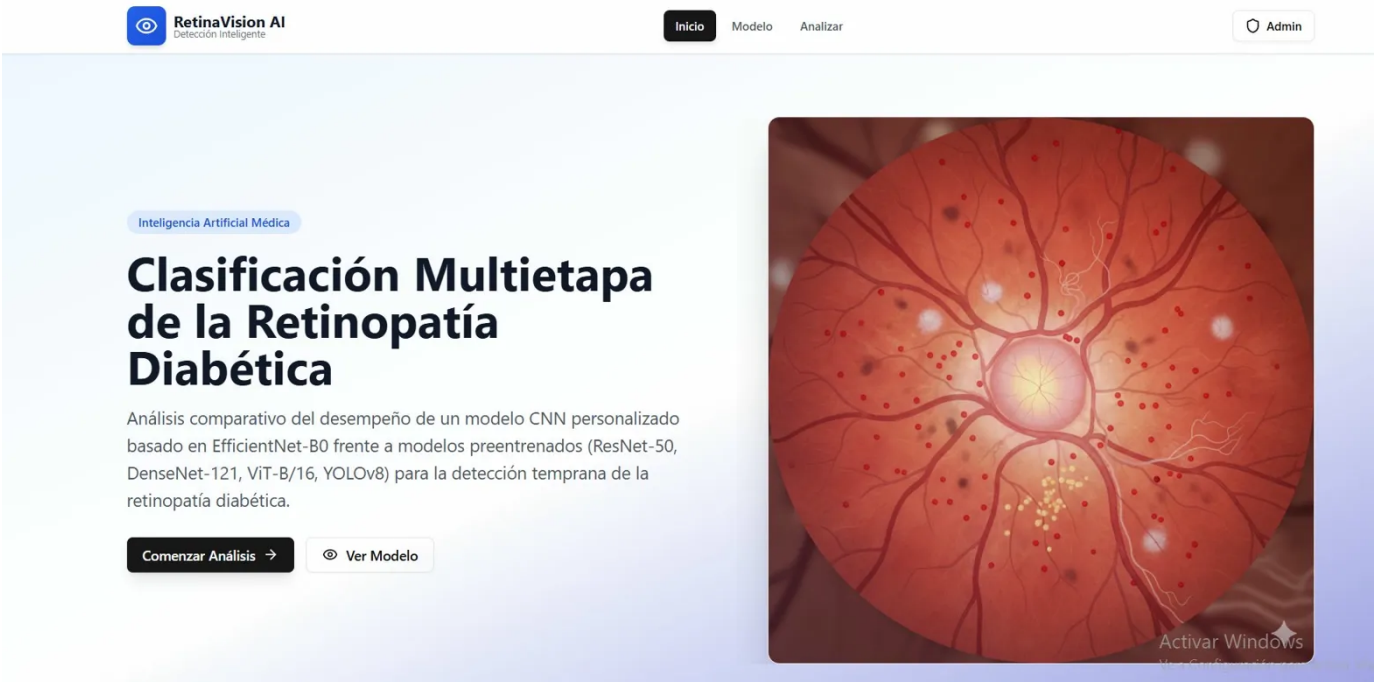


Figura 66. Pantalla de inicio del sitio web RetinaVision AI. Fuente: Autor Propio.

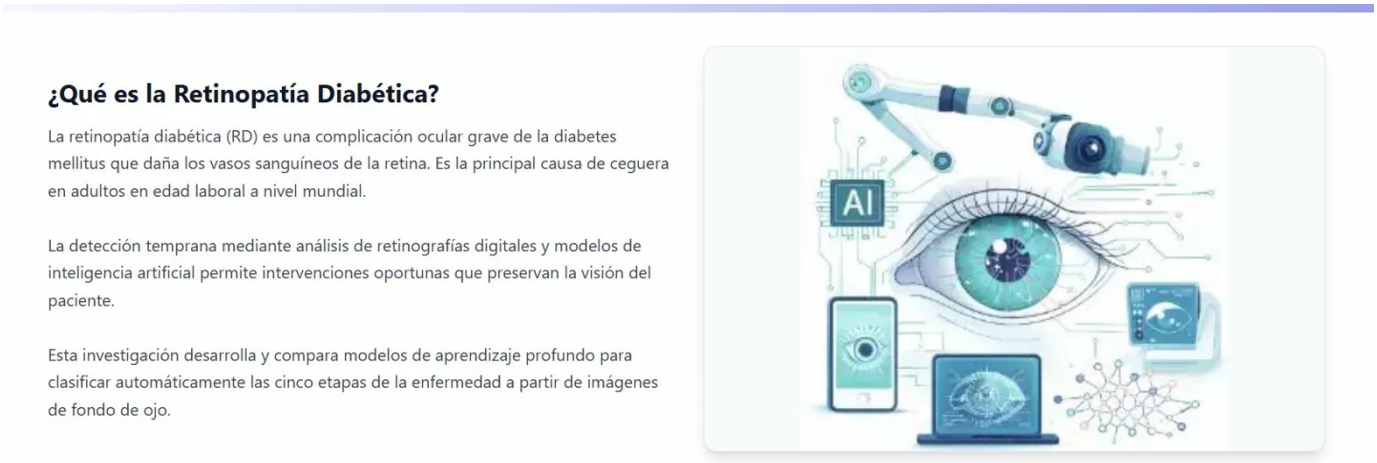


Figura 67. Sección informativa: definición de la retinopatía diabética dentro del sitio web. Fuente: Autor Propio.

Figura 68. Sección descriptiva de los grados generales de la retinopatía diabética. Fuente: Autor Propio.

Figura 69. Sección de clasificación multietapa de la retinopatía diabética, desde la etapa 0 hasta la etapa 4. Fuente: Autor Propio.

Figura 70. Vista de la etapa 0: retina normal sin presencia de retinopatía diabética. Fuente: Autor Propio.

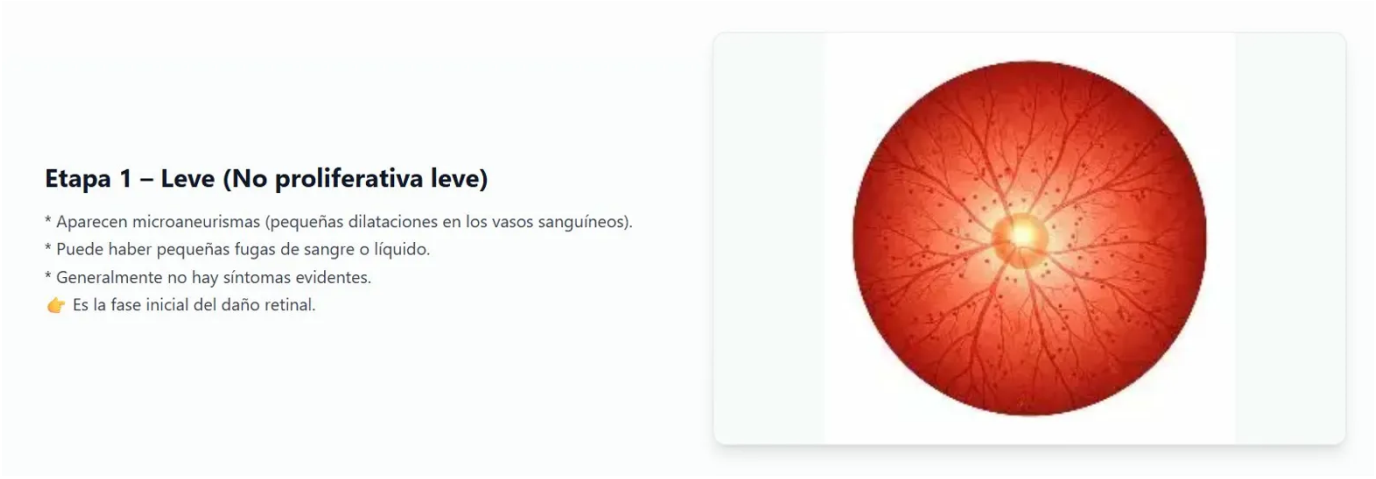


Figura 71. Vista de la etapa 1: retinopatía diabética no proliferativa leve. Fuente: Autor Propio.

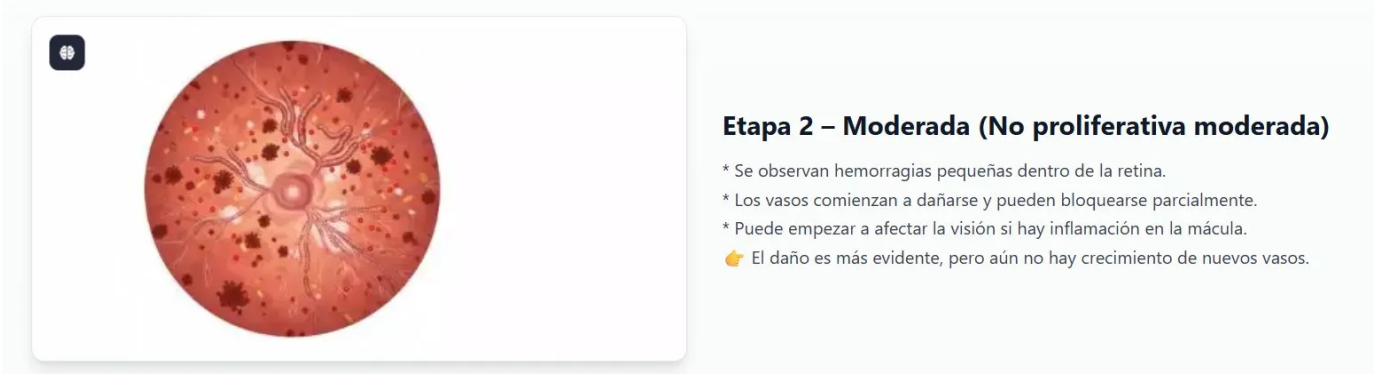


Figura 72. Vista de la etapa 2: retinopatía diabética no proliferativa moderada. Fuente: Autor Propio.

Figura 73. Vista de la etapa 3: retinopatía diabética no proliferativa severa. Fuente: Autor Propio.

Figura 74. Vista de la etapa 4: retinopatía diabética proliferativa. Fuente: Autor Propio.

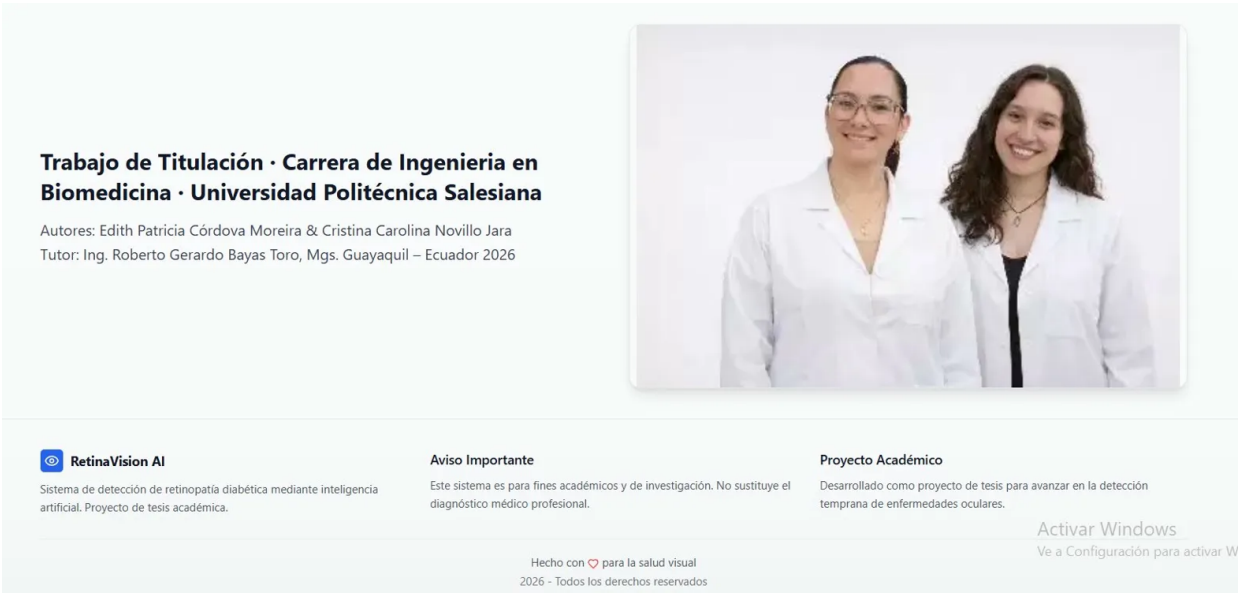


Figura 75. Sección final del sitio web con información académica del proyecto de titulación y autores. Fuente: Autor Propio.

XII-B. Capturas del módulo de modelos y análisis comparativo del sistema

La presente subsección muestra las capturas correspondientes al módulo de visualización, comparación y análisis de modelos implementado en la plataforma *RetinaVision AI*. En esta sección se presentan tanto la descripción general de los cinco modelos evaluados, como las métricas globales, curvas de entrenamiento, matrices de confusión, análisis por clase y la explicación individual de cada arquitectura utilizada en el estudio.

Figura 76. Pantalla principal de la sección de modelos del sistema RetinaVision AI, donde se introducen las cinco arquitecturas evaluadas en el estudio. Fuente: Autor Propio.

Figura 77. Sección de características generales del sistema, mostrando el enfoque ensemble, la clasificación multiclase, el tamaño de entrada de las imágenes y el uso de atención externa. Fuente: Autor Propio.

Comparativa de Rendimiento

Metricas de evaluacion de cada modelo sobre el conjunto de validacion (50,000 imagenes).

Modelo	Accuracy	F1	AUC	Precision	Recall	Epocas
DenseNet121 + EA	**77.49%**	77.39%	94.31%	77.73%	77.49%	30
EfficientNet-B0 + EA Mejor	**82.25%**	82.16%	95.41%	82.66%	82.26%	89
ResNet50 + EA	**78.36%**	78.41%	94.11%	79.08%	78.36%	82
ViT-B/16 + EA	**76.85%**	76.68%	93.77%	77.06%	76.85%	78
YOLOv8x-cls	**81.89%**	81.92%	96.20%	82.01%	81.88%	-

Figura 78. Tabla comparativa de rendimiento de los cinco modelos evaluados, incluyendo accuracy, F1, AUC, precision, recall y número de épocas. Fuente: Autor Propio.

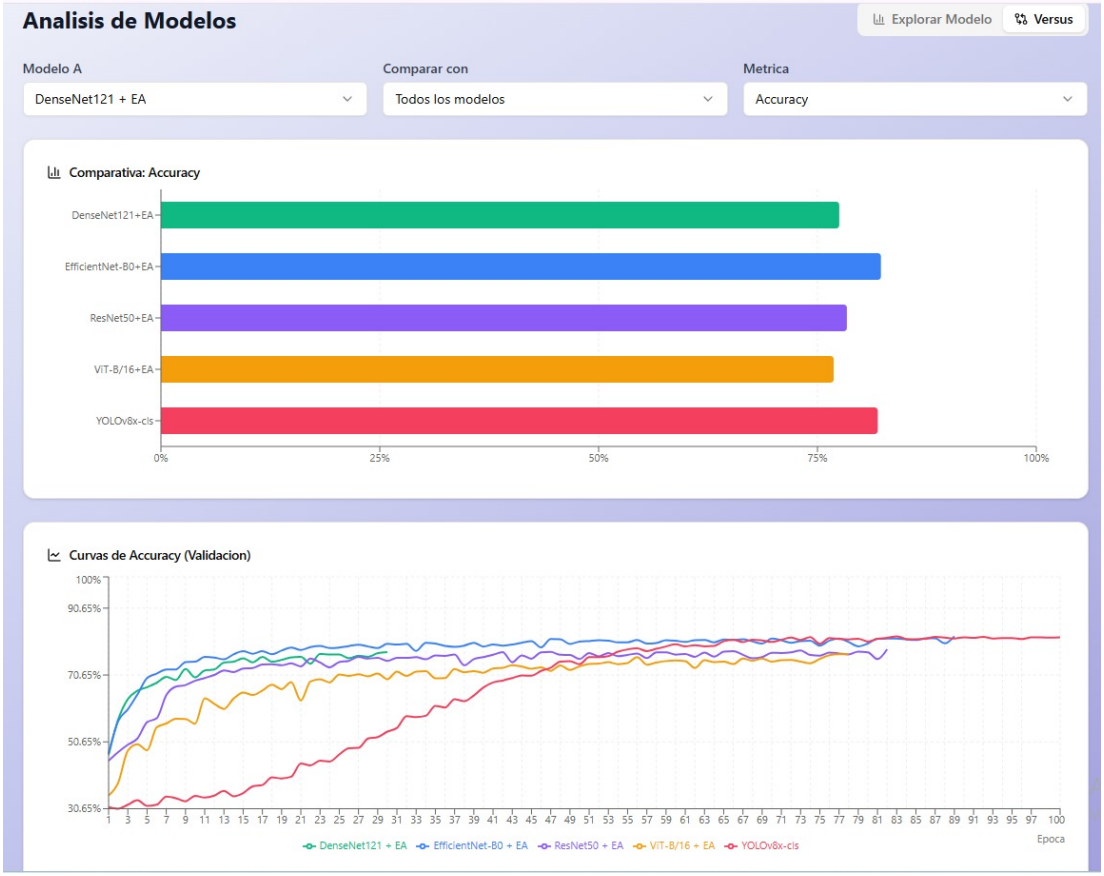


Figura 79. Vista del modo de comparación entre modelos, con gráfico de barras de accuracy y curvas de validación para las arquitecturas analizadas. Fuente: Autor Propio.

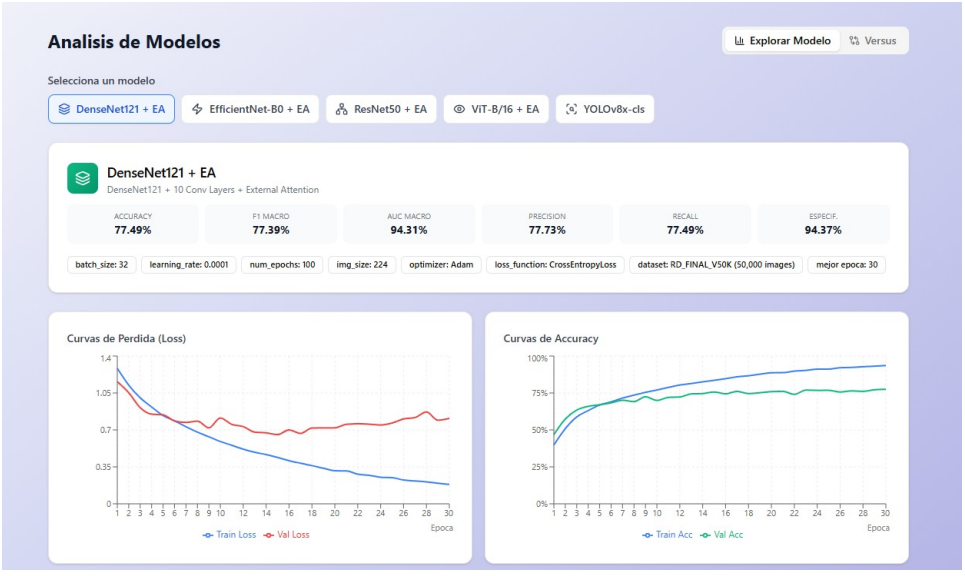
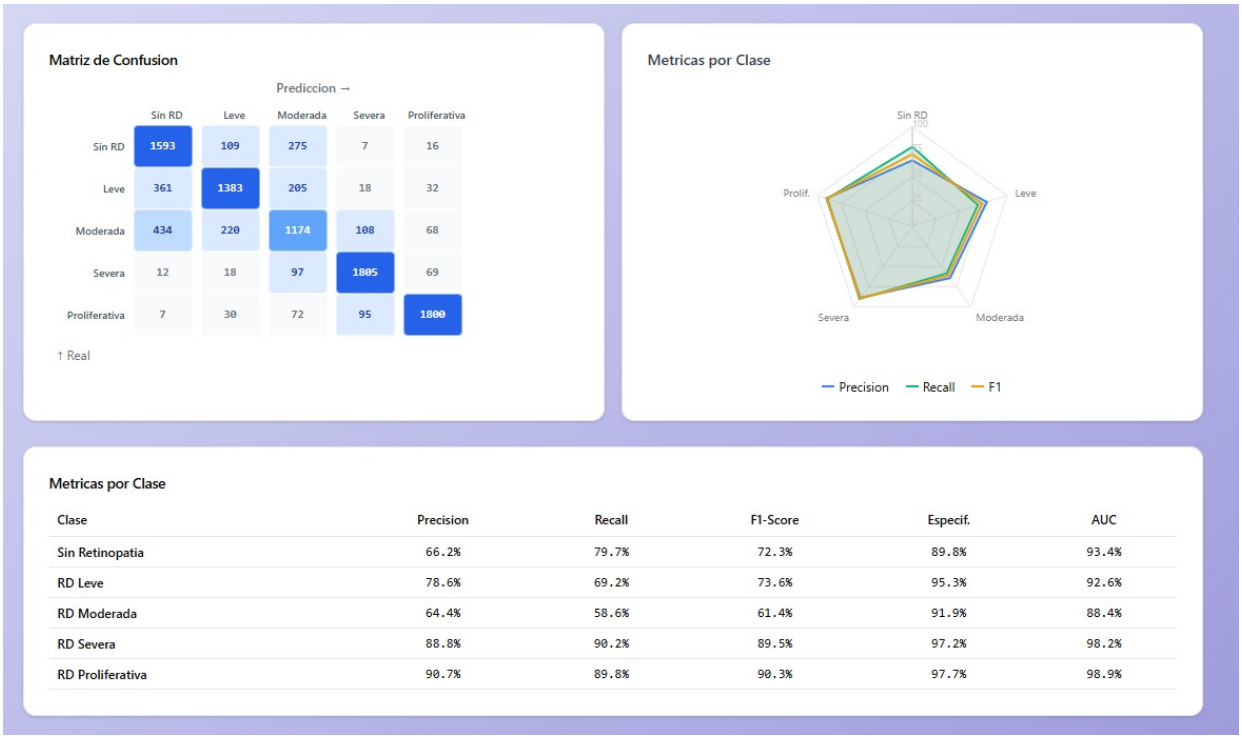


Figura 80. Vista detallada del modelo DenseNet121 + External Attention, incluyendo métricas globales, hiperparámetros y curvas de pérdida y exactitud. Fuente: Autor Propio.



Metricas por Clase

Clase	Precision	Recall	F1-Score	Especif.	AUC
Sin Retinopatia	66.2%	79.7%	72.3%	89.8%	93.4%
RD Leve	78.6%	69.2%	73.6%	95.3%	92.6%
RD Moderada	64.4%	58.6%	61.4%	91.9%	88.4%
RD Severa	88.8%	90.2%	89.5%	97.2%	98.2%
RD Proliferativa	90.7%	89.8%	90.3%	97.7%	98.9%

Figura 81. Matriz de confusión y métricas por clase del modelo DenseNet121 + External Attention, utilizadas para el análisis detallado del comportamiento multiclase. Fuente: Autor Propio.

Figura 82. Sección de estrategia de consenso del sistema y descripción del modelo EfficientNet-B0 + 10 Conv + External Attention, identificado como la arquitectura de mejor desempeño del estudio. Fuente: Autor Propio.

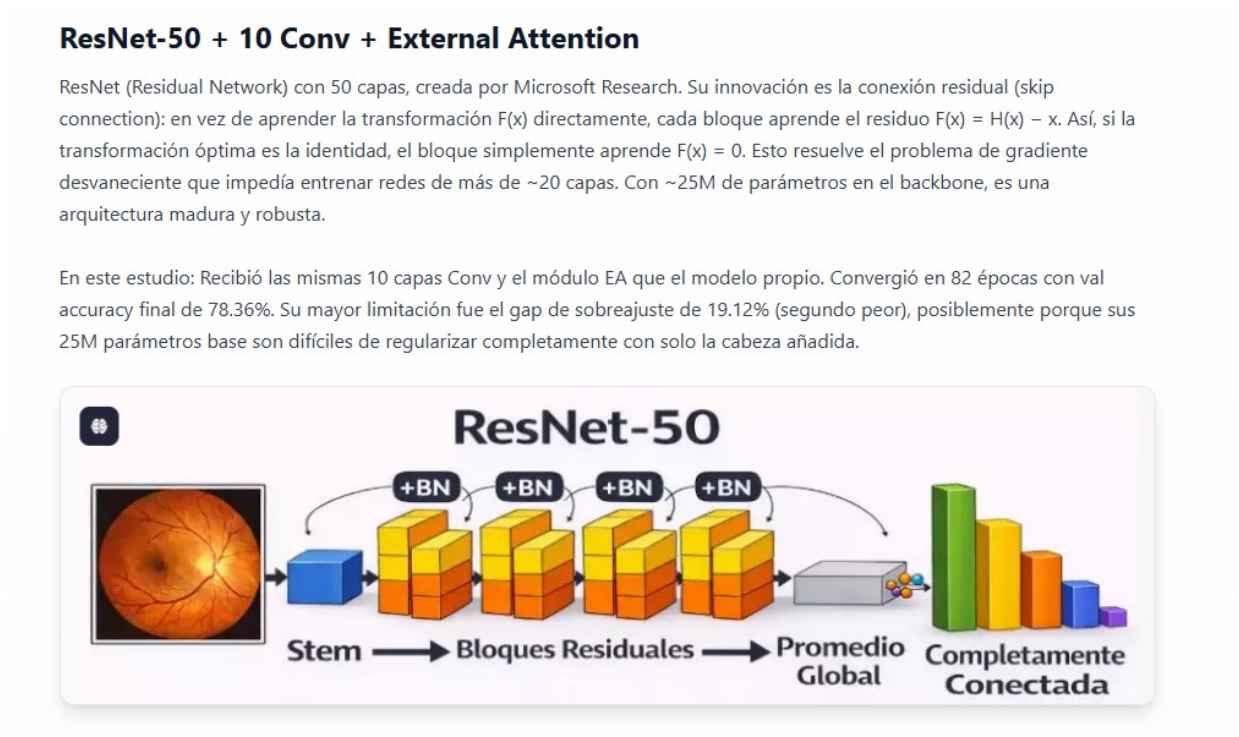


Figura 83. Descripción e ilustración del modelo ResNet-50 + 10 Conv + External Attention dentro de la plataforma web. Fuente: Autor Propio.

Figura 84. Descripción e ilustración del modelo DenseNet-121 + 10 Conv + External Attention, resaltando su convergencia rápida durante el entrenamiento. Fuente: Autor Propio.

YOLOv8X-cls — Ultralytics

La variante de clasificación de imagen de la familia YOLO (You Only Look Once), desarrollada por Ultralytics. Su backbone es CSP (Cross Stage Partial): divide el canal de features en dos partes, aplica bloques residuales solo en una parte y las concatena al final. Esto elimina cómputo redundante y reduce parámetros (~3.2M — el más ligero del estudio). A diferencia de los otros modelos, YOLOv8X-cls no utilizó las 10 capas Conv adicionales ni el módulo EA, siendo evaluado en su configuración nativa Ultralytics.

Dato destacado: Obtuvo el AUC macro más alto del estudio (0.9620), superando incluso al EfficientNet-B0 personalizado. Sin embargo, sus métricas son sobre el conjunto de test (no val), lo que limita la comparación directa. No se reportaron curvas de entrenamiento por época en modo clasificación Ultralytics.

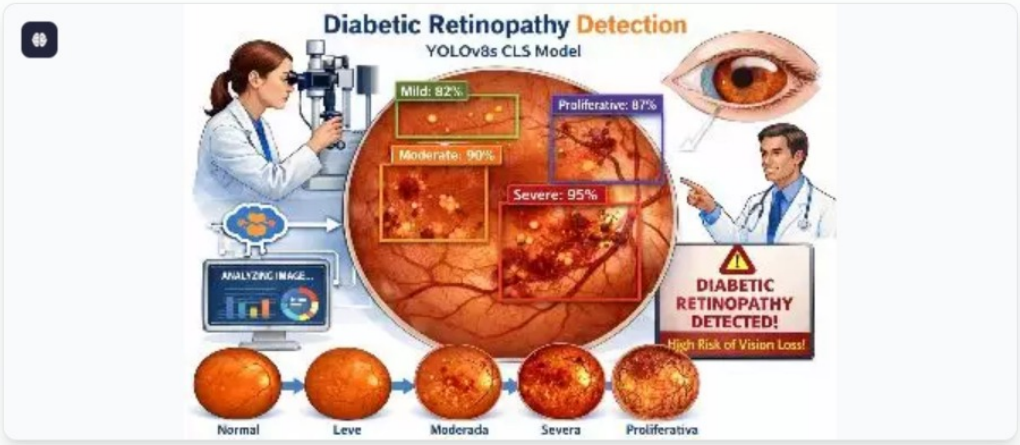


Figura 85. Descripción e ilustración del modelo YOLOv8x-cls, mostrando su aplicación para clasificación de retinopatía diabética en la plataforma. Fuente: Autor Propio.

ViT-B/16 + 10 Conv + External Attention

El primer modelo que aplica directamente la arquitectura Transformer (de NLP) a imágenes, sin convoluciones en el backbone. La imagen 224×224 se divide en 196 parches de 16×16 píxeles, cada uno se convierte en un embedding (token), y se procesan con 12 bloques de Multi-Head Self-Attention (MHSA). El token [CLS] especial agrega el contexto global para clasificación.

En este estudio: Mayor gap de sobreajuste (19.48%). ViT-B/16 fue diseñado para datasets gigantes (JFT-300M). Con 50K imágenes, su falta de inducción inductiva local (los transformers no asumen localidad espacial como las CNNs) lo penaliza. También presentó un spike brusco en épocas 75–76 (caída y recuperación de accuracy) documentado en el historial de entrenamiento.

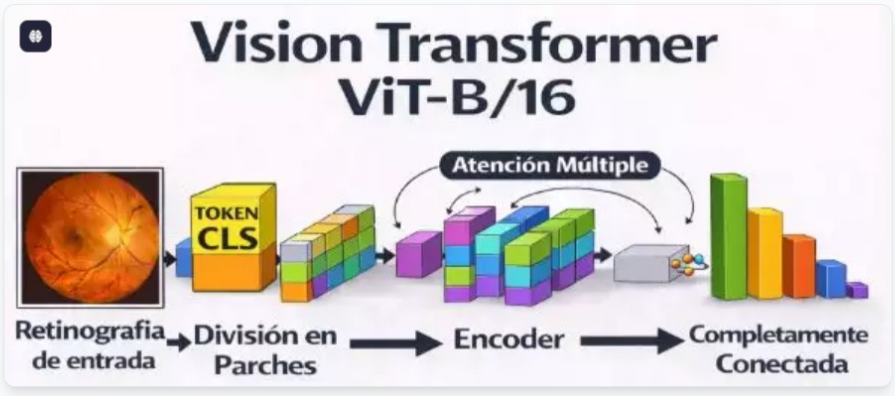


Figura 86. Descripción e ilustración del modelo ViT-B/16 + 10 Conv + External Attention, destacando su arquitectura basada en transformadores. Fuente: Autor Propio.

XII-C. Capturas del módulo de análisis y resultado de predicción

La presente subsección muestra las capturas correspondientes al módulo de análisis de imágenes retinográficas implementado en la plataforma *RetinaVision AI*. En esta sección se visualiza el proceso de carga de la imagen, la ejecución del análisis automático y la presentación de resultados generados por el modelo personalizado, incluyendo la predicción diagnóstica, el mapa de calor tipo Grad-CAM y las probabilidades asociadas a cada clase.

Figura 87. Interfaz del módulo de análisis de la plataforma RetinaVision AI, donde se realiza la carga de la imagen de fondo de ojo y se ejecuta el proceso de inferencia automática. Fuente: Autor Propio.

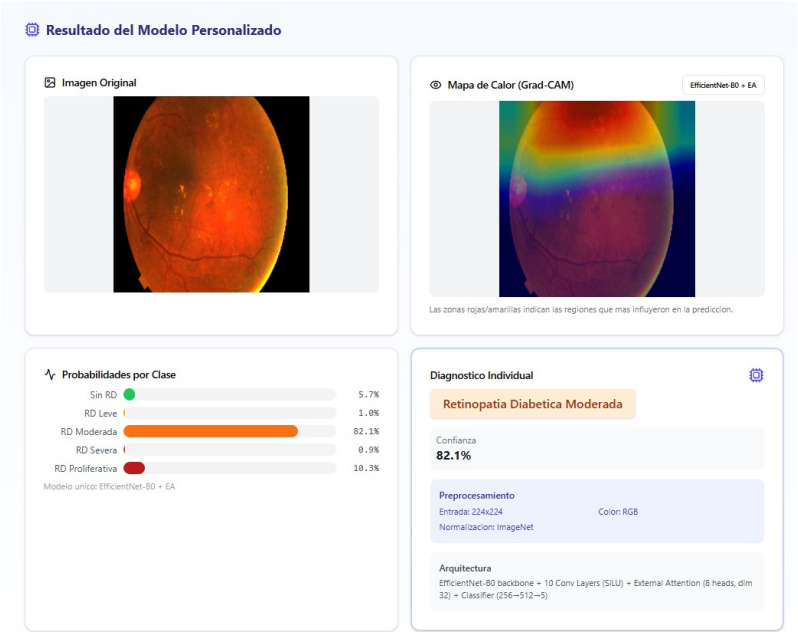


Figura 88. Resultado del análisis generado por el modelo personalizado, mostrando la imagen original, el mapa de calor Grad-CAM, las probabilidades por clase y el diagnóstico individual de la retinopatía diabética. Fuente: Autor Propio.

XII-D. Capturas del análisis general por consenso del sistema

La presente subsección reúne las capturas correspondientes al análisis general de la plataforma *RetinaVision AI*, en el cual se integra la salida de los modelos del ensamble para generar una predicción consensuada. Estas evidencias muestran la carga de la imagen retinográfica, la comparación de confianza entre modelos, la visualización del mapa de calor y la emisión del diagnóstico final por consenso.

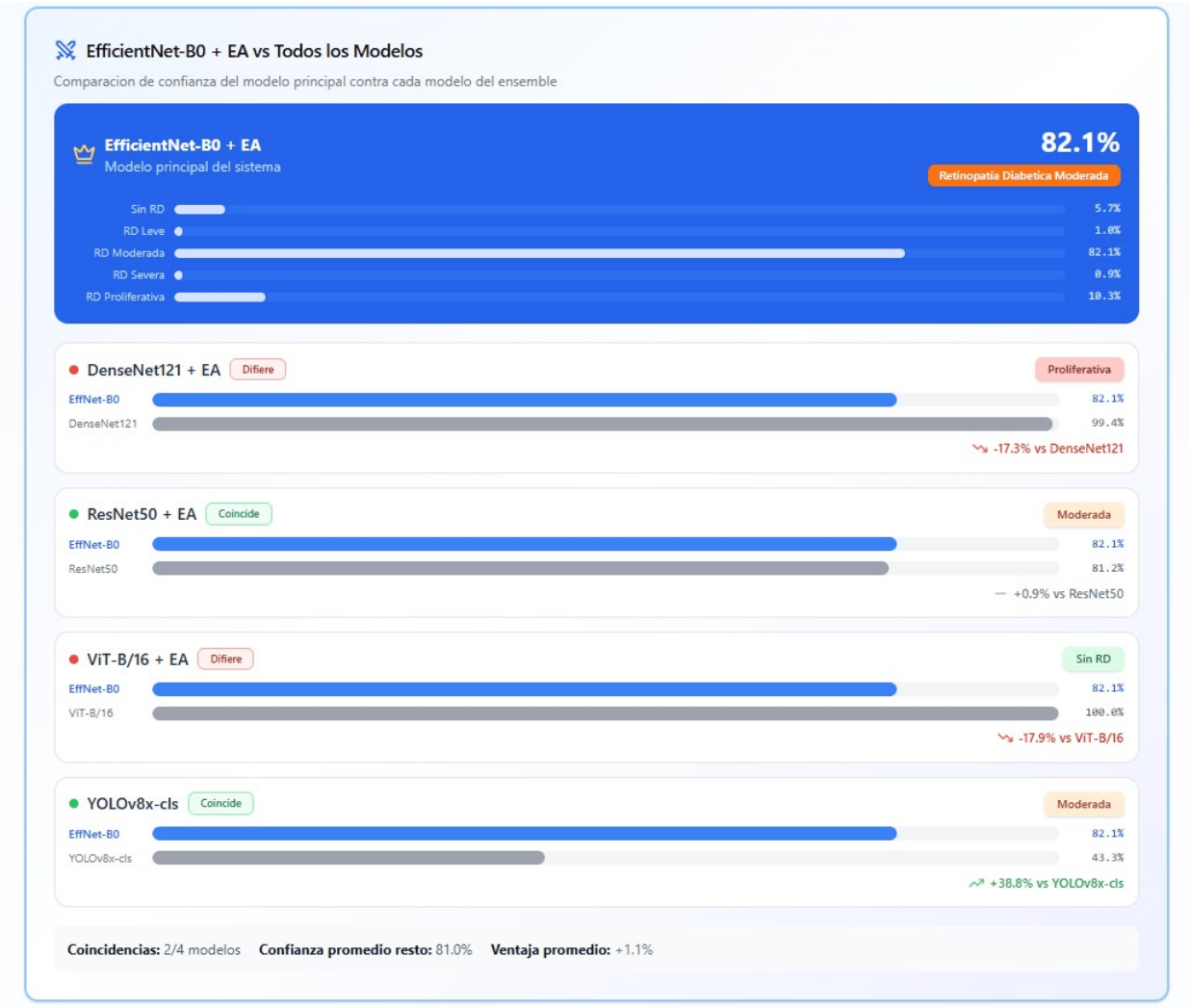


Figura 89. Comparación de confianza entre el modelo principal y los demás modelos del ensamble, mostrando coincidencias, diferencias y el porcentaje de acuerdo en la clasificación obtenida. Fuente: Autor Propio.

XII-E. Código de preprocesamiento del conjunto de datos

El Código 1 muestra el procedimiento empleado para montar Google Drive, descomprimir el conjunto de datos y aplicar el preprocesamiento a las imágenes retinográficas.

```
from google.colab import drive
import os
import shutil

# Try to unmount if it's already mounted, to ensure a clean state
try:
    drive.flush_and_unmount()
except ValueError:
    pass # Drive was not mounted, continue

# Ensure the mount point is empty before mounting
mount_point = '/content/drive'
if os.path.exists(mount_point) and os.listdir(mount_point):
    print(f Warning: Mount point '{mount_point}' is not empty. Attempting to clear its contents. )
    for filename in os.listdir(mount_point):
        filepath = os.path.join(mount_point, filename)
        try:
            if os.path.isfile(filepath) or os.path.islink(filepath):
                os.remove(filepath)
            elif os.path.isdir(filepath):
                shutil.rmtree(filepath)
        except Exception as e:
            print(f Error clearing {filepath}: {e} )
    print(f Contents of '{mount_point}' cleared. )

drive.mount('/content/drive', force_remount=True)

import zipfile

# Rutas
zip_path =  /content/drive/MyDrive/RD_FINAL_V50K/DATASET.zip
extract_path =  /content/drive/MyDrive/RD_FINAL_V50K/DATASET_descomprimido

# Crear carpeta de destino
os.makedirs(extract_path, exist_ok=True)

# Descomprimir
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print(     Listo! Dataset en: , extract_path)import zipfile
import os

# Rutas
zip_path =  /content/drive/MyDrive/RD_FINAL_V50K/DATASET.zip
extract_path =  /content/drive/MyDrive/RD_FINAL_V50K/DATASET_descomprimido

# Crear carpeta de destino
os.makedirs(extract_path, exist_ok=True)

# Descomprimir
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print(     Listo! Dataset en: , extract_path)import zipfile
import os

zip_path =  /content/drive/MyDrive/RD_FINAL_V50K/DATASET.zip
extract_path =  /content/DATASET

os.makedirs(extract_path, exist_ok=True)

with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print(     Listo en local! )
print(os.listdir(extract_path))import os
```

```python
import cv2
import numpy as np
from pathlib import Path

# Rutas
input_base =  /content/DATASET
output_base =  /content/drive/MyDrive/RD_FINAL_V50K/DATASET_PREPROCESADO

IMG_SIZE = 512

def crop_image_from_gray(img, tol=7):
       Crop exacto del notebook de Kaggle - versi n color corregida
    if img.ndim == 2:
        mask = img > tol
        return img[np.ix_(mask.any(1), mask.any(0))]
    elif img.ndim == 3:
        gray_img = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
        mask = gray_img > tol
        check_shape = img[:,:,0][np.ix_(mask.any(1), mask.any(0))].shape[0]
        if check_shape == 0:  # imagen muy oscura, retorna original
            return img
        else:
            img1 = img[:,:,0][np.ix_(mask.any(1), mask.any(0))]
            img2 = img[:,:,1][np.ix_(mask.any(1), mask.any(0))]
            img3 = img[:,:,2][np.ix_(mask.any(1), mask.any(0))]
            img = np.stack([img1, img2, img3], axis=-1)
        return img

def load_ben_color(path, sigmaX=10):
       Funci n exacta del notebook - crop + Ben Graham en color
    image = cv2.imread(path)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = crop_image_from_gray(image)
    image = cv2.resize(image, (IMG_SIZE, IMG_SIZE))
    image = cv2.addWeighted(image, 4, cv2.GaussianBlur(image, (0,0), sigmaX), -4, 128)
    return image

# Procesar splits
splits = [ train ,  test ,  valid ]
total, errores = 0, 0

for split in splits:
    split_path = Path(input_base) / split
    if not split_path.exists():
        continue

    for img_path in split_path.rglob( * ):
        if img_path.suffix.lower() in [ .jpg ,  .jpeg ,  .png ]:
            rel_path = img_path.relative_to(input_base)
            out_file = Path(output_base) / rel_path
            out_file.parent.mkdir(parents=True, exist_ok=True)

            try:
                img = load_ben_color(str(img_path))
                # Guardar en BGR para cv2
                img_bgr = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
                cv2.imwrite(str(out_file), img_bgr)
                total += 1
            except Exception as e:
                errores += 1
                print(f        Error en {img_path.name}: {e} )

    print(f     {split} procesado! )

print(f \  n     Listo! {total} im genes procesadas, {errores} errores. )
print(f        Guardado en: {output_base} )
```

Listing 1. Código de preprocesamiento del conjunto de datos. Fuente: Autor Propio.

XII-F. Código de entrenamiento del modelo personalizado

```

from google.colab import drive
import os
import shutil


try:
    from google.colab import drive
    drive.mount('/content/drive')
    print('    Google Drive montado\n')
except:
    print('       No es Google Colab o ya est  montado\n')

import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset
from torchvision import transforms, models
from PIL import Image
import os
import numpy as np
from tqdm import tqdm
import random
import json
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
from pathlib import Path
from sklearn.metrics import (confusion_matrix, classification_report,
                             precision_recall_fscore_support, roc_curve, auc,
                             roc_auc_score)

# ==================== CONFIGURACI N ====================
SEED = 42
torch.manual_seed(SEED)
torch.cuda.manual_seed(SEED)
np.random.seed(SEED)
random.seed(SEED)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False

DATASET_PATH =  /content/drive/MyDrive/RD_FINAL_V50K/DATASET_PREPROCESADO
OUTPUT_PATH =  /content/drive/MyDrive/RD_FINAL_V50K/efficientnetb0_external_attention

BATCH_SIZE = 32
LEARNING_RATE = 0.0001
NUM_EPOCHS = 100
IMG_SIZE = 224
NUM_WORKERS = 2

# ==================== EXTERNAL ATTENTION MODULE ====================
class ExternalAttention(nn.Module):
       External Attention Module
    def __init__(self, dim, num_heads=8, dim_head=64, dropout=0.):
        super().__init__()
        inner_dim = dim_head * num_heads
        self.num_heads = num_heads
        self.scale = dim_head ** -0.5

        self.to_qkv = nn.Linear(dim, inner_dim * 3, bias=False)
        self.to_out = nn.Sequential(
            nn.Linear(inner_dim, dim),
            nn.Dropout(dropout)
        )

        self.mem_k = nn.Parameter(torch.randn(num_heads, dim_head, dim_head))
        self.mem_v = nn.Parameter(torch.randn(num_heads, dim_head, dim_head))

    def forward(self, x):
        b, n, _, h = *x.shape, self.num_heads
        qkv = self.to_qkv(x).chunk(3, dim=-1)
```

```
        q, k, v = map(lambda t: t.reshape(b, n, h, -1).transpose(1, 2), qkv)

        attn = torch.einsum('bhnd,hdk->bhnk', q, self.mem_k) * self.scale
        attn = attn.softmax(dim=-1)

        out = torch.einsum('bhnk,hkd->bhnd', attn, self.mem_v)
        out = out.transpose(1, 2).reshape(b, n, -1)

        return self.to_out(out)

# ==================== MODELO EFFICIENTNET-B0 + 10 CAPAS CONV ====================
class EfficientNetB0WithExternalAttention(nn.Module):
    def __init__(self, num_classes, pretrained=True):
        super().__init__()

        efficientnet = models.efficientnet_b0(pretrained=pretrained)
        self.features = efficientnet.features

        self.extra_conv1 = nn.Sequential(
            nn.Conv2d(1280, 640, kernel_size=3, padding=1),
            nn.BatchNorm2d(640),
            nn.SiLU(inplace=True),
            nn.Dropout2d(0.3)
        )

        self.extra_conv2 = nn.Sequential(
            nn.Conv2d(640, 448, kernel_size=3, padding=1),
            nn.BatchNorm2d(448),
            nn.SiLU(inplace=True),
            nn.Dropout2d(0.3)
        )

        self.extra_conv3 = nn.Sequential(
            nn.Conv2d(448, 320, kernel_size=3, padding=1),
            nn.BatchNorm2d(320),
            nn.SiLU(inplace=True),
            nn.Dropout2d(0.3)
        )

        self.extra_conv4 = nn.Sequential(
            nn.Conv2d(320, 256, kernel_size=3, padding=1),
            nn.BatchNorm2d(256),
            nn.SiLU(inplace=True),
            nn.Dropout2d(0.25)
        )

        self.extra_conv5 = nn.Sequential(
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.BatchNorm2d(256),
            nn.SiLU(inplace=True),
            nn.Dropout2d(0.25)
        )

        self.extra_conv6 = nn.Sequential(
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.BatchNorm2d(256),
            nn.SiLU(inplace=True),
            nn.Dropout2d(0.25)
        )

        self.extra_conv7 = nn.Sequential(
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.BatchNorm2d(256),
            nn.SiLU(inplace=True),
            nn.Dropout2d(0.2)
        )

        self.extra_conv8 = nn.Sequential(
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.BatchNorm2d(256),
            nn.SiLU(inplace=True),
            nn.Dropout2d(0.2)
        )
```

```
        self.extra_conv9 = nn.Sequential(
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.BatchNorm2d(256),
            nn.SiLU(inplace=True),
            nn.Dropout2d(0.2)
        )

        self.extra_conv10 = nn.Sequential(
            nn.Conv2d(256, 256, kernel_size=3, padding=1),
            nn.BatchNorm2d(256),
            nn.SiLU(inplace=True)
        )

        self.gap = nn.AdaptiveAvgPool2d((1, 1))

        self.external_attention = ExternalAttention(
            dim=256, num_heads=8, dim_head=32, dropout=0.1
        )

        self.classifier = nn.Sequential(
            nn.Linear(256, 512),
            nn.SiLU(inplace=True),
            nn.Dropout(0.5),
            nn.Linear(512, num_classes)
        )

    def forward(self, x):
        x = self.features(x)
        x = self.extra_conv1(x)
        x = self.extra_conv2(x)
        x = self.extra_conv3(x)
        x = self.extra_conv4(x)
        x = self.extra_conv5(x)
        x = self.extra_conv6(x)
        x = self.extra_conv7(x)
        x = self.extra_conv8(x)
        x = self.extra_conv9(x)
        x = self.extra_conv10(x)

        x = self.gap(x).flatten(1)
        x = x.unsqueeze(1)
        x = self.external_attention(x)
        x = x.squeeze(1)
        x = self.classifier(x)

        return x

# ==================== DATASET ====================
class CustomImageDataset(Dataset):
    def __init__(self, root_dir, transform=None):
        self.root_dir = root_dir
        self.transform = transform
        self.images = []
        self.labels = []

        self.classes = sorted(os.listdir(root_dir))
        self.class_to_idx = {cls: idx for idx, cls in enumerate(self.classes)}

        for class_name in self.classes:
            class_dir = os.path.join(root_dir, class_name)
            if os.path.isdir(class_dir):
                for img_name in os.listdir(class_dir):
                    if img_name.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp')):
                        self.images.append(os.path.join(class_dir, img_name))
                        self.labels.append(self.class_to_idx[class_name])

    def __len__(self):
        return len(self.images)

    def __getitem__(self, idx):
        image = Image.open(self.images[idx]).convert('RGB')
        label = self.labels[idx]
```

```
        if self.transform:
            image = self.transform(image)
        return image, label

# ==================== M TRICAS COMPLETAS ====================
def calculate_metrics(y_true, y_pred, y_pred_proba, num_classes):
    cm = confusion_matrix(y_true, y_pred)
    precision, recall, f1, support = precision_recall_fscore_support(
        y_true, y_pred, average=None, zero_division=0
    )
    precision_macro, recall_macro, f1_macro, _ = precision_recall_fscore_support(
        y_true, y_pred, average='macro', zero_division=0
    )

    specificity = []
    for i in range(num_classes):
        tn = cm.sum() - (cm[i, :].sum() + cm[:, i].sum() - cm[i, i])
        fp = cm[:, i].sum() - cm[i, i]
        spec = tn / (tn + fp) if (tn + fp) > 0 else 0
        specificity.append(spec)

    try:
        auc_scores = []
        for i in range(num_classes):
            y_true_binary = (y_true == i).astype(int)
            y_score = y_pred_proba[:, i]
            if len(np.unique(y_true_binary)) > 1:
                auc_score = roc_auc_score(y_true_binary, y_score)
                auc_scores.append(auc_score)
            else:
                auc_scores.append(0.0)
        auc_macro = np.mean(auc_scores)
    except:
        auc_scores = [0.0] * num_classes
        auc_macro = 0.0

    metrics = {
        'confusion_matrix': cm.tolist(),
        'precision_per_class': precision.tolist(),
        'recall_per_class': recall.tolist(),
        'f1_per_class': f1.tolist(),
        'specificity_per_class': specificity,
        'auc_per_class': auc_scores,
        'support_per_class': support.tolist(),
        'precision_macro': float(precision_macro),
        'recall_macro': float(recall_macro),
        'f1_macro': float(f1_macro),
        'specificity_macro': float(np.mean(specificity)),
        'auc_macro': float(auc_macro)
    }

    return metrics

# ==================== ENTRENAMIENTO ====================
def train_model(model, train_loader, val_loader, criterion, optimizer, num_epochs, device, save_path,
    num_classes):
    best_acc = 0.0
    history = {
        'train_loss': [], 'train_acc': [],
        'val_loss': [], 'val_acc': [],
        'val_metrics': []
    }

    for epoch in range(num_epochs):
        print(f'\n{ = *70}')
        print(f' poca  {epoch+1}/{num_epochs}')
        print(f'{ = *70}')

        model.train()
        train_loss, train_correct, train_total = 0.0, 0, 0

        pbar = tqdm(train_loader, desc='           Entrenando', ncols=100)
        for inputs, labels in pbar:
```

```
            inputs, labels = inputs.to(device), labels.to(device)

            optimizer.zero_grad()
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            train_loss += loss.item() * inputs.size(0)
            _, predicted = torch.max(outputs, 1)
            train_total += labels.size(0)
            train_correct += (predicted == labels).sum().item()

        train_loss = train_loss / train_total
        train_acc = 100 * train_correct / train_total

        model.eval()
        val_loss, val_correct, val_total = 0.0, 0, 0
        all_preds, all_labels, all_probs = [], [], []

        with torch.no_grad():
            for inputs, labels in tqdm(val_loader, desc='       Validando', ncols=100):
                inputs, labels = inputs.to(device), labels.to(device)
                outputs = model(inputs)
                loss = criterion(outputs, labels)

                val_loss += loss.item() * inputs.size(0)
                probs = torch.softmax(outputs, dim=1)
                _, predicted = torch.max(outputs, 1)
                val_total += labels.size(0)
                val_correct += (predicted == labels).sum().item()

                all_preds.extend(predicted.cpu().numpy())
                all_labels.extend(labels.cpu().numpy())
                all_probs.extend(probs.cpu().numpy())

        val_loss = val_loss / val_total
        val_acc = 100 * val_correct / val_total

        val_metrics = calculate_metrics(np.array(all_labels), np.array(all_preds),
                                        np.array(all_probs), num_classes)

        history['train_loss'].append(train_loss)
        history['train_acc'].append(train_acc)
        history['val_loss'].append(val_loss)
        history['val_acc'].append(val_acc)
        history['val_metrics'].append(val_metrics)

        if val_acc > best_acc:
            best_acc = val_acc
            torch.save({
                'epoch': epoch,
                'model_state_dict': model.state_dict(),
                'optimizer_state_dict': optimizer.state_dict(),
                'best_acc': best_acc,
                'history': history,
                'num_classes': num_classes,
                'class_names': train_loader.dataset.classes,
                'best_metrics': val_metrics
            }, os.path.join(save_path, 'best_model.pth'))

    return history

# ==================== MAIN ====================
def main():
    os.makedirs(OUTPUT_PATH, exist_ok=True)
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

    train_transform = transforms.Compose([
        transforms.Resize((IMG_SIZE, IMG_SIZE)),
        transforms.RandomHorizontalFlip(),
        transforms.RandomRotation(15),
        transforms.ColorJitter(0.2, 0.2, 0.2),
```

```
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])

    val_transform = transforms.Compose([
        transforms.Resize((IMG_SIZE, IMG_SIZE)),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ])

    train_dataset = CustomImageDataset(os.path.join(DATASET_PATH, 'train'), train_transform)
    val_dataset = CustomImageDataset(os.path.join(DATASET_PATH, 'valid'), val_transform)

    train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True,
    num_workers=NUM_WORKERS, pin_memory=True)
    val_loader = DataLoader(val_dataset, batch_size=BATCH_SIZE, shuffle=False,
    num_workers=NUM_WORKERS, pin_memory=True)

    num_classes = len(train_dataset.classes)
    model = EfficientNetB0WithExternalAttention(num_classes=num_classes).to(device)

    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=LEARNING_RATE)

    history = train_model(model, train_loader, val_loader, criterion, optimizer, NUM_EPOCHS, device,
    OUTPUT_PATH, num_classes)

if __name__ == '__main__':
    main()
```

Listing 2. Código de entrenamiento del modelo personalizado EfficientNet-B0 + 10 capas convolucionales + External Attention. Fuente: Autor Propio.

XII-G. Código para generación del archivo JSON de resultados

```
import torch
import json
import numpy as np
from datetime import datetime
import os

MODEL_PATH =  /content/drive/MyDrive/RD_FINAL_V50K/efficientnetb0_external_attention/best_model.pth
OUTPUT_DIR =  /content/drive/MyDrive/RD_FINAL_V50K/efficientnetb0_external_attention
MODEL_NAME =  EfficientNet-B0 + 10 Conv + External Attention

def generate_json(checkpoint_path, output_path, model_name):
    checkpoint = torch.load(checkpoint_path, map_location='cpu', weights_only=False)
    history = checkpoint.get('history', {})
    best_acc = checkpoint.get('best_acc', 0)
    num_classes = checkpoint.get('num_classes', 0)
    class_names = checkpoint.get('class_names', [])
    epoch = checkpoint.get('epoch', 0)
    best_metrics = checkpoint.get('best_metrics', {})

    convergence_epoch = int(np.argmax(history['val_acc']) + 1) if history.get('val_acc') else None
    overfitting_gap = float(history['train_acc'][-1] - history['val_acc'][-1]) if
    history.get('train_acc') and history.get('val_acc') else None

    training_results = {
         metadata : {
             timestamp : datetime.now().strftime( %Y-%m-%d %H:%M:%S ),
             model_name : model_name,
             dataset :  RD_FINAL_V50K (50,000 images) ,
             seed : 42,
             framework :  PyTorch
        },
         configuration : {
             hyperparameters : {
                 batch_size : 32,
                 learning_rate : 0.0001,
                 num_epochs : 100,
                 img_size : 224,
                 optimizer :  Adam ,
```

```
                loss_function :  CrossEntropyLoss
            }
        },
        dataset_info : {
            num_classes : num_classes,
            class_names : class_names,
            total_images : 50000,
            images_per_class : 10000
        },
        training_history : {
            train_loss : history.get('train_loss', []),
            train_accuracy : history.get('train_acc', []),
            val_loss : history.get('val_loss', []),
            val_accuracy : history.get('val_acc', [])
        },
        final_results : {
            best_epoch : epoch + 1,
            best_val_accuracy : float(best_acc),
            final_train_loss : float(history['train_loss'][-1]) if history.get('train_loss') else
    None,
            final_val_loss : float(history['val_loss'][-1]) if history.get('val_loss') else None,
            final_train_accuracy : float(history['train_acc'][-1]) if history.get('train_acc') else
    None,
            final_val_accuracy : float(history['val_acc'][-1]) if history.get('val_acc') else None
        },
        best_metrics : best_metrics,
        performance_metrics : {
            best_val_acc : float(best_acc),
            convergence_epoch : convergence_epoch,
            overfitting_gap_percent : overfitting_gap
        }
    }

    json_path = os.path.join(output_path, 'complete_training_results.json')
    with open(json_path, 'w', encoding='utf-8') as f:
        json.dump(training_results, f, indent=4, ensure_ascii=False)

    print(f     JSON guardado: {json_path} )
    return training_results

if __name__ == '__main__':
    generate_json(MODEL_PATH, OUTPUT_DIR, MODEL_NAME)
```

Listing 3. Código para generar el archivo JSON con los resultados del modelo personalizado. Fuente: Autor Propio.

XII-H. Código para generación de visualizaciones del modelo personalizado

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import os

def plot_training_history(history, output_path, model_name):
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))
    epochs = range(1, len(history['train_loss']) + 1)

    ax1.plot(epochs, history['train_loss'], 'o-', label='Train Loss', linewidth=2, markersize=4,
    color='#2E86AB')
    ax1.plot(epochs, history['val_loss'], 's-', label='Val Loss', linewidth=2, markersize=4,
    color='#F18F01')
    ax1.set_xlabel(' poca ', fontsize=12, fontweight='bold')
    ax1.set_ylabel('Loss', fontsize=12, fontweight='bold')
    ax1.set_title(f'Loss Evolution - {model_name}', fontsize=14, fontweight='bold')
    ax1.legend(fontsize=11)
    ax1.grid(alpha=0.3)

    ax2.plot(epochs, history['train_acc'], 'o-', label='Train Acc', linewidth=2, markersize=4,
    color='#06A77D')
    ax2.plot(epochs, history['val_acc'], 's-', label='Val Acc', linewidth=2, markersize=4,
    color='#9D4EDD')
    best_acc = max(history['val_acc'])
    ax2.axhline(y=best_acc, color='red', linestyle='--', linewidth=2, label=f'Best: {best_acc:.2f}%')
```

```
    ax2.set_xlabel(' poca ', fontsize=12, fontweight='bold')
    ax2.set_ylabel('Accuracy (%)', fontsize=12, fontweight='bold')
    ax2.set_title(f'Accuracy Evolution - {model_name}', fontsize=14, fontweight='bold')
    ax2.legend(fontsize=11)
    ax2.grid(alpha=0.3)

    plt.tight_layout()
    plt.savefig(output_path, dpi=300, bbox_inches='tight')
    plt.close()

def plot_confusion_matrix(cm, class_names, output_path, model_name):
    plt.figure(figsize=(10, 8))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=class_names, yticklabels=class_names,
                cbar_kws={'label': 'N mero de predicciones'})
    plt.title(f'Matriz de Confusi n - {model_name}', fontsize=16, fontweight='bold', pad=20)
    plt.ylabel('Verdadero', fontsize=12, fontweight='bold')
    plt.xlabel('Predicho', fontsize=12, fontweight='bold')
    plt.tight_layout()
    plt.savefig(output_path, dpi=300, bbox_inches='tight')
    plt.close()

def plot_metrics_per_class(metrics, class_names, output_path, model_name):
    fig, axes = plt.subplots(2, 2, figsize=(16, 12))
    x = np.arange(len(class_names))
    width = 0.6

    metrics_list = [
        ('precision_per_class', 'Precision', axes[0, 0]),
        ('recall_per_class', 'Recall', axes[0, 1]),
        ('f1_per_class', 'F1-Score', axes[1, 0]),
        ('specificity_per_class', 'Specificity', axes[1, 1])
    ]

    for metric_key, metric_name, ax in metrics_list:
        bars = ax.bar(x, metrics[metric_key], width, alpha=0.8)
        ax.set_ylabel(metric_name, fontweight='bold')
        ax.set_title(f'{metric_name} por Clase - {model_name}', fontsize=14, fontweight='bold')
        ax.set_xticks(x)
        ax.set_xticklabels(class_names)
        ax.set_ylim(0, 1.1)
        ax.grid(axis='y', alpha=0.3)

        for bar in bars:
            height = bar.get_height()
            ax.text(bar.get_x() + bar.get_width()/2., height,
                    f'{height:.3f}', ha='center', va='bottom', fontweight='bold')

    plt.tight_layout()
    plt.savefig(output_path, dpi=300, bbox_inches='tight')
    plt.close()

def plot_overfitting_analysis(history, output_path, model_name):
    fig, ax = plt.subplots(figsize=(12, 6))
    epochs = range(1, len(history['train_acc']) + 1)
    gap = [t - v for t, v in zip(history['train_acc'], history['val_acc'])]

    ax.plot(epochs, gap, 'o-', linewidth=3, markersize=8, color='#9D4EDD',
            markeredgecolor='white', markeredgewidth=2, label='Train-Val Gap')
    ax.axhline(y=0, color='gray', linestyle='-', linewidth=2, alpha=0.5)
    ax.fill_between(epochs, 0, gap, where=[g > 0 for g in gap],
                    alpha=0.3, color='red', label='Overfitting')
    ax.fill_between(epochs, 0, gap, where=[g <= 0 for g in gap],
                    alpha=0.3, color='green', label='No Overfitting')

    ax.set_xlabel(' poca ', fontsize=14, fontweight='bold')
    ax.set_ylabel('Gap (Train - Val) %', fontsize=14, fontweight='bold')
    ax.set_title(f'Overfitting Analysis - {model_name}', fontsize=16, fontweight='bold')
    ax.legend(fontsize=13, frameon=True, shadow=True)
    ax.grid(alpha=0.3, linestyle='--', linewidth=1.5)

    plt.tight_layout()
    plt.savefig(output_path, dpi=300, bbox_inches='tight')
```

```
    plt.close()
```

Listing 4. Código para generar las visualizaciones del modelo personalizado. Fuente: Autor Propio.

XII-I. Código fuente de la plataforma web

A continuación se presentan los fragmentos más representativos del código fuente desarrollado para la plataforma web de detección de retinopatía diabética. El sistema fue implementado con FastAPI en el backend y React.js en el frontend, utilizando un ensemble de cinco modelos de aprendizaje profundo para la clasificación de imágenes de fondo de ojo.

XII-J. Módulo de External Attention

```
class ExternalAttention(nn.Module):
    def __init__(self, dim, num_heads=8, dim_head=64, dropout=0.):
        super().__init__()
        inner_dim = dim_head * num_heads
        self.num_heads = num_heads
        self.scale = dim_head ** -0.5

        self.to_qkv = nn.Linear(dim, inner_dim * 3, bias=False)
        self.to_out = nn.Sequential(
            nn.Linear(inner_dim, dim),
            nn.Dropout(dropout),
        )

        self.mem_k = nn.Parameter(torch.randn(num_heads, dim_head, dim_head))
        self.mem_v = nn.Parameter(torch.randn(num_heads, dim_head, dim_head))

    def forward(self, x):
        b, n, _, h = *x.shape, self.num_heads
        qkv = self.to_qkv(x).chunk(3, dim=-1)
        q, k, v = map(lambda t: t.reshape(b, n, h, -1).transpose(1, 2), qkv)

        attn = torch.einsum('bhnd,hdk->bhnk', q, self.mem_k) * self.scale
        attn = attn.softmax(dim=-1)

        out = torch.einsum('bhnk,hkd->bhnd', attn, self.mem_v)
        out = out.transpose(1, 2).reshape(b, n, -1)

        return self.to_out(out)
```

Listing 5. Módulo de External Attention. Fuente: Autor Propio.

XII-K. Arquitectura EfficientNet-B0 + External Attention

```
class EfficientNetB0WithExternalAttention(nn.Module):
    def __init__(self, num_classes=NUM_CLASSES, pretrained=False):
        super().__init__()
        efficientnet = models.efficientnet_b0(weights=None)
        self.features = efficientnet.features  # output: 1280 channels

        # 10 capas convolucionales con activacion SiLU
        # Canales: 1280->640->448->320->256->...->256
        self.extra_conv1 = nn.Sequential(
            nn.Conv2d(1280, 640, 3, padding=1),
            nn.BatchNorm2d(640), nn.SiLU(inplace=True), nn.Dropout2d(0.3))
        self.extra_conv2 = nn.Sequential(
            nn.Conv2d(640, 448, 3, padding=1),
            nn.BatchNorm2d(448), nn.SiLU(inplace=True), nn.Dropout2d(0.3))
        # ... (extra_conv3 a extra_conv10)
        self.extra_conv10 = nn.Sequential(
            nn.Conv2d(256, 256, 3, padding=1),
            nn.BatchNorm2d(256), nn.SiLU(inplace=True))

        self.gap = nn.AdaptiveAvgPool2d((1, 1))
        self.external_attention = ExternalAttention(
            dim=256, num_heads=8, dim_head=32, dropout=0.1)
        self.classifier = nn.Sequential(
            nn.Linear(256, 512), nn.SiLU(inplace=True),
```

```
            nn.Dropout(0.5), nn.Linear(512, num_classes))

    def forward(self, x):
        x = self.features(x)
        for i in range(1, 11):
            x = getattr(self, f'extra_conv{i}')(x)
        x = self.gap(x).flatten(1)
        x = x.unsqueeze(1)
        x = self.external_attention(x)
        x = x.squeeze(1)
        return self.classifier(x)
```

Listing 6. Arquitectura EfficientNet-B0 + External Attention. Fuente: Autor Propio.

XII-L. Preprocesamiento de imágenes y predicción PyTorch

```
from torchvision import transforms
from PIL import Image

# Normalizacion ImageNet
_IMAGENET_MEAN = [0.485, 0.456, 0.406]
_IMAGENET_STD = [0.229, 0.224, 0.225]

_inference_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=_IMAGENET_MEAN, std=_IMAGENET_STD),
])

def preprocess_image(image_bytes: bytes) -> torch.Tensor:
    img = Image.open(io.BytesIO(image_bytes)).convert('RGB')
    return _inference_transform(img).unsqueeze(0)  # [1, 3, 224, 224]

def _predict_pytorch(self, model, input_tensor):
    with torch.no_grad():
        logits = model(input_tensor.to(self.device))
        probs = F.softmax(logits, dim=1).squeeze(0)
        confidence, class_idx = probs.max(0)

    idx = class_idx.item()
    return {
        'prediction': CLASS_LABELS[idx],
        'confidence': round(confidence.item(), 4),
        'severity': SEVERITY_LEVELS[idx],
        'probabilities': [round(p.item(), 4) for p in probs],
    }

def _predict_yolo(self, model, pil_image):
    results = model.predict(pil_image, imgsz=224, verbose=False)
    probs = results[0].probs
    idx = probs.top1
    confidence = probs.top1conf.item()
    all_probs = probs.data.tolist()
    return {
        'prediction': CLASS_LABELS[idx],
        'confidence': round(confidence, 4),
        'severity': SEVERITY_LEVELS[idx],
        'probabilities': [round(p, 4) for p in all_probs],
    }
```

Listing 7. Preprocesamiento de imágenes y predicción PyTorch. Fuente: Autor Propio.

XII-M. Carga de modelos y servicio de inferencia (ModelService)

```
class ModelService:
    def __init__(self):
        self.models: dict = {}
        self.loaded = False
        self.loading = False
        self.models_loaded_count = 0
        self.device = torch.device('cpu')
```

```
    def load_all_models(self, models_dir: str):
           Carga modelos en un thread de fondo.
        self.loading = True
        thread = threading.Thread(
            target=self._load_models_sync,
            args=(models_dir,), daemon=True)
        thread.start()

    def _load_models_sync(self, models_dir: str):
        model_configs = [
            {'name': 'DenseNet121 + EA',
             'class': DenseNet121WithExternalAttention,
             'path': os.path.join(models_dir, 'densenet121_ea', 'best_model.pth'),
             'checkpoint_key': 'model_state_dict'},
            {'name': 'EfficientNet-B0 + EA',
             'class': EfficientNetB0WithExternalAttention,
             'path': os.path.join(models_dir, 'efficientnet_b0_ea', 'best_model.pth'),
             'checkpoint_key': 'model_state_dict'},
            {'name': 'ResNet50 + EA',
             'class': ResNet50WithExternalAttention,
             'path': os.path.join(models_dir, 'resnet50_ea', 'best_model.pth'),
             'checkpoint_key': 'model_state_dict'},
            {'name': 'ViT-B/16 + EA',
             'class': ViTB16WithExternalAttention,
             'path': os.path.join(models_dir, 'vit_b16_ea', 'best_model.pth'),
             'checkpoint_key': 'model_state_dict'},
            {'name': 'YOLOv8x-cls',
             'path': os.path.join(models_dir, 'yolov8x_cls', 'best.pt'),
             'checkpoint_key': 'yolo'},
        ]

        for cfg in model_configs:
            if cfg['checkpoint_key'] == 'yolo':
                self._load_yolo(cfg)
            else:
                self._load_pytorch(cfg)
            self.models_loaded_count = len(self.models)
            gc.collect()

        self.loaded = len(self.models) > 0
        self.loading = False

    def _load_pytorch(self, cfg: dict):
        model = cfg['class'](num_classes=5, pretrained=False)
        checkpoint = torch.load(
            cfg['path'], map_location=self.device, weights_only=False)
        model.load_state_dict(checkpoint[cfg['checkpoint_key']])
        del checkpoint
        gc.collect()
        model.to(self.device)
        model.eval()
        self.models[cfg['name']] = {'model': model, 'type': 'pytorch'}

    def _load_yolo(self, cfg: dict):
        from ultralytics import YOLO
        model = YOLO(cfg['path'])
        self.models[cfg['name']] = {'model': model, 'type': 'yolo'}

# Singleton
model_service = ModelService()
```

Listing 8. Carga de modelos y servicio de inferencia (ModelService). Fuente: Autor Propio.

XII-N. Predicción con ensemble de 5 modelos

```
def predict_all(self, image_bytes: bytes) -> list[dict]:
       Ejecuta los 5 modelos sobre la misma imagen.
    results = []
    input_tensor = preprocess_image(image_bytes)
    pil_image = Image.open(io.BytesIO(image_bytes)).convert('RGB')

```

```
    for name, info in self.models.items():
        try:
            if info['type'] == 'pytorch':
                result = self._predict_pytorch(info['model'], input_tensor)
            else:
                result = self._predict_yolo(info['model'], pil_image)
            result['model_name'] = name
            results.append(result)
        except Exception:
            results.append({
                'model_name': name,
                'prediction': 'Error',
                'confidence': 0.0,
                'severity': 'none',
                'probabilities': [0.0] * NUM_CLASSES,
            })
    return results
```

Listing 9. Predicción con ensemble de 5 modelos. Fuente: Autor Propio.

XII-Ñ. Detección de imagen no retinal basada en entropía

```
def check_retinal_validity(self, results: list[dict]) -> dict:

    Determina si la imagen es una retinograf a  v lida.
    Criterios: max_confidence < 0.45 y avg_entropy > 1.3 => no retinal.

    valid = [r for r in results if r.get('prediction') != 'Error']
    if not valid:
        return {'is_retinal': False, 'max_confidence': 0.0, 'avg_entropy': 0.0}

    max_confidence = max(r['confidence'] for r in valid)

    entropies = []
    for r in valid:
        probs = r.get('probabilities', [])
        if probs:
            entropy = -sum(p * math.log(p + 1e-10) for p in probs)
            entropies.append(entropy)

    avg_entropy = sum(entropies) / len(entropies) if entropies else 0.0
    is_retinal = not (max_confidence < 0.45 and avg_entropy > 1.3)

    return {
        'is_retinal': is_retinal,
        'max_confidence': round(max_confidence, 4),
        'avg_entropy': round(avg_entropy, 4),
    }
```

Listing 10. Detección de imagen no retinal basada en entropía. Fuente: Autor Propio.

XII-O. Generación de Grad-CAM

```
class GradCAM:
       Grad-CAM para EfficientNet-B0+EA, target layer: extra_conv10.

    def __init__(self, model, target_layer_name='extra_conv10'):
        self.model = model
        self.gradients = None
        self.activations = None
        self._hooks = []
        target_layer = getattr(model, target_layer_name)
        self._hooks.append(
            target_layer.register_forward_hook(self._save_activation))
        self._hooks.append(
            target_layer.register_full_backward_hook(self._save_gradient))

    def _save_activation(self, module, input, output):
        self.activations = output.detach()

    def _save_gradient(self, module, grad_input, grad_output):
```

```
        self.gradients = grad_output[0].detach()

    def generate(self, input_tensor, target_class=None):
        self.model.eval()
        input_tensor.requires_grad_(True)
        output = self.model(input_tensor)

        if target_class is None:
            target_class = output.argmax(dim=1).item()

        self.model.zero_grad()
        target_score = output[0, target_class]
        target_score.backward()

        weights = self.gradients.mean(dim=[2, 3], keepdim=True)
        cam = (weights * self.activations).sum(dim=1, keepdim=True)
        cam = F.relu(cam)

        cam = cam.squeeze().cpu().numpy()
        if cam.max() > 0:
            cam = (cam - cam.min()) / (cam.max() - cam.min())
        return cam, target_class

def generate_heatmap_overlay(model, input_tensor, original_image_bytes, alpha=0.5):
    gradcam = GradCAM(model, target_layer_name='extra_conv10')
    try:
        heatmap, predicted_class = gradcam.generate(input_tensor)
    finally:
        gradcam.remove_hooks()

    original = Image.open(io.BytesIO(original_image_bytes)).convert('RGB')
    orig_w, orig_h = original.size

    heatmap_resized = np.array(
        Image.fromarray((heatmap * 255).astype(np.uint8)).resize(
            (orig_w, orig_h), Image.BILINEAR)) / 255.0

    colormap = matplotlib.colormaps['jet']
    heatmap_colored = (colormap(heatmap_resized)[:, :, :3] * 255).astype(np.uint8)
    original_np = np.array(original)
    overlay = (original_np * (1 - alpha) + heatmap_colored * alpha).astype(np.uint8)

    overlay_image = Image.fromarray(overlay)
    buffer = io.BytesIO()
    overlay_image.save(buffer, format='JPEG', quality=90)
    return base64.b64encode(buffer.getvalue()).decode('utf-8'), predicted_class
```

Listing 11. Generación de Grad-CAM. Fuente: Autor Propio.

XII-P. Endpoint REST de predicción con ensemble

```
from fastapi import APIRouter, UploadFile, File, HTTPException

router = APIRouter(prefix= /predict , tags=[ AI Prediction ])

@router.post( / , response_model=MultiModelPredictionResponse)
async def predict_retinopathy(image: UploadFile = File(...)):
       Endpoint de prediccion con 5 modelos de IA.

    if not image.content_type or not image.content_type.startswith('image/'):
        raise HTTPException(status_code=400, detail= El archivo debe ser una imagen )

    if model_service.loading:
        raise HTTPException(
            status_code=503,
            detail=f Modelos cargando ({model_service.models_loaded_count}/5)
        )

    if not model_service.loaded:
        raise HTTPException(status_code=503, detail= Los modelos de IA no pudieron cargarse. )

    contents = await image.read()
```

```
    raw_results = model_service.predict_all(contents)
    results = [SingleModelResult(**r) for r in raw_results]

    validity = model_service.check_retinal_validity(raw_results)
    consensus = _compute_consensus(results)

    gradcam_overlay = None
    if validity['is_retinal']:
        gradcam_result = model_service.predict_with_gradcam(contents)
        if gradcam_result:
            gradcam_overlay = gradcam_result['overlay_base64']

    return MultiModelPredictionResponse(
        results=results,
        consensus=consensus,
        image_filename=image.filename or  imagen.jpg ,
        is_retinal=validity['is_retinal'],
        gradcam_overlay=gradcam_overlay,
    )
```

Listing 12. Endpoint REST de predicción con ensemble. Fuente: Autor Propio.

XII-Q. Endpoint REST de modelo personalizado

```
@router.post( /custom-model , response_model=CustomModelResponse)
async def predict_custom_model(image: UploadFile = File(...)):
       Prediccion usando solo EfficientNet-B0+EA.

    if not image.content_type or not image.content_type.startswith('image/'):
        raise HTTPException(status_code=400, detail= El archivo debe ser una imagen )

    contents = await image.read()
    result = model_service.predict_single_efficientnet(contents)

    import math
    probs = result.get('probabilities', [])
    confidence = result.get('confidence', 0.0)
    entropy = -sum(p * math.log(p + 1e-10) for p in probs)
    is_retinal = not (confidence < 0.45 and entropy > 1.3)

    gradcam_overlay = None
    if is_retinal:
        gradcam_result = model_service.predict_with_gradcam(contents)
        if gradcam_result:
            gradcam_overlay = gradcam_result['overlay_base64']

    return CustomModelResponse(
        model_name=result['model_name'],
        architecture=result['architecture'],
        prediction=result['prediction'],
        confidence=result['confidence'],
        severity=result['severity'],
        probabilities=result['probabilities'],
        preprocessing=result['preprocessing'],
        gradcam_overlay=gradcam_overlay,
        is_retinal=is_retinal,
        image_filename=image.filename or  imagen.jpg ,
    )
```

Listing 13. Endpoint REST de modelo personalizado. Fuente: Autor Propio.

XII-R. Consenso por votación de modelos

```
from collections import Counter

def _compute_consensus(results: list[SingleModelResult]) -> ConsensusResult:
    valid = [r for r in results if r.prediction != 'Error']
    total = len(results)

    if not valid:
```

```
        return ConsensusResult(
            prediction='Error', severity='none', confidence=0.0,
            agreement_count=0, total_models=total,
            recommendation='No se pudo realizar el analisis.')

    votes = Counter(r.severity for r in valid)
    winner_severity, agreement_count = votes.most_common(1)[0]

    winners = [r for r in valid if r.severity == winner_severity]
    avg_confidence = sum(r.confidence for r in winners) / len(winners)

    winner_idx = SEVERITY_LEVELS.index(winner_severity)
    prediction = CLASS_LABELS[winner_idx]

    return ConsensusResult(
        prediction=prediction,
        severity=winner_severity,
        confidence=round(avg_confidence, 4),
        agreement_count=agreement_count,
        total_models=total,
        recommendation=_get_recommendation(winner_severity),
    )

def _get_recommendation(severity: str) -> str:
    recommendations = {
        'none': 'No se detectaron signos de retinopatia diabetica. Se recomienda control anual de
    rutina.',
        'mild': 'Signos leves detectados. Consulte con su oftalmologo.',
        'moderate': 'Signos moderados. Consulta oftalmologica a la brevedad.',
        'severe': 'Signos severos. Atencion oftalmologica urgente.',
        'proliferative': 'RD proliferativa. Atencion inmediata requerida.',
    }
    return recommendations.get(severity, 'Consulte con un especialista.')
```

Listing 14. Consenso por votación de modelos. Fuente: Autor Propio.

XII-S. Generación de reporte PDF profesional

```
from fpdf import FPDF
import matplotlib.pyplot as plt
import matplotlib
matplotlib.use('Agg')

CLASS_LABELS = ['Sin Retinopatia', 'RD Leve', 'RD Moderada',
                'RD Severa', 'RD Proliferativa']

class RetinopathyReport(FPDF):
    def header(self):
        self.set_font('Helvetica', 'B', 10)
        self.cell(0, 8, 'Reporte de Analisis - Retinopatia Diabetica', align='C')
        self.ln(10)

    def section_title(self, title):
        self.set_font('Helvetica', 'B', 11)
        self.set_fill_color(41, 128, 185)
        self.set_text_color(255, 255, 255)
        self.cell(0, 8, f' {title}', fill=True)
        self.ln(10)
        self.set_text_color(0, 0, 0)

def generate_report(image_bytes, filename, results, consensus, is_retinal, gradcam_overlay_b64):
    pdf = RetinopathyReport('P', 'mm', 'A4')
    pdf.auto_page_break = False

    # Pagina 1: Informacion + imagenes + diagnostico
    pdf.add_page()
    pdf.section_title('Informacion del Analisis')
    # ... (datos del paciente, fecha, imagen original, Grad-CAM)

    # Pagina 2: Graficas
    pdf.add_page()
    pdf.section_title('Resultados por Modelo')
```

```
    # Generar graficas con matplotlib (donuts, barras, votacion)

    # Pagina 3: Tabla + escala + info tecnica
    pdf.add_page()
    pdf.section_title('Tabla de Resultados')
    # Tabla con prediccion de cada modelo

    return bytes(pdf.output())
```

Listing 15. Generación de reporte PDF profesional. Fuente: Autor Propio.

XII-T. Configuración del servidor FastAPI

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from fastapi.staticfiles import StaticFiles
from fastapi.responses import FileResponse
from pathlib import Path

app = FastAPI(
    title= Deteccion de Retinopatia Diabetica ,
    version= 2.0.0 ,
    description= API para deteccion de retinopatia diabetica con IA
)

app.add_middleware(
    CORSMiddleware,
    allow_origins=settings.ALLOWED_ORIGINS,
    allow_credentials=True,
    allow_methods=[ * ],
    allow_headers=[ * ],
)

@app.on_event( startup )
async def startup_event():
    await connect_to_mongo()
    from app.services.model_service import model_service
    models_dir = os.environ.get( MODELS_DIR ,  /app/models_weights )
    model_service.load_all_models(models_dir)

@app.on_event( shutdown )
async def shutdown_event():
    await close_mongo_connection()

PUBLIC_DIR = Path( /app/public )
if PUBLIC_DIR.exists():
    app.mount( /static , StaticFiles(
        directory=str(PUBLIC_DIR /  static )), name= static )

app.include_router(auth.router, prefix= /api )
app.include_router(pages.router, prefix= /api )
app.include_router(prediction.router, prefix= /api )
app.include_router(models.router, prefix= /api )

@app.get( /health )
async def health():
    return { status :  healthy }

@app.get( /{full_path:path} )
async def serve_spa(full_path: str):
    if full_path.startswith( api/ ):
        raise HTTPException(status_code=404)
    index_file = PUBLIC_DIR /  index.html
    if index_file.exists():
        return FileResponse(index_file, media_type= text/html )
```

Listing 16. Configuración del servidor FastAPI. Fuente: Autor Propio.

XII-U. Conexión asíncrona a MongoDB

```python
from motor.motor_asyncio import AsyncIOMotorClient
from app.core.config import settings

class Database:
    client: AsyncIOMotorClient = None
    db = None

db_instance = Database()

async def connect_to_mongo():
       Conectar a MongoDB de forma asincrona.
    db_instance.client = AsyncIOMotorClient(settings.MONGODB_URI)
    db_instance.db = db_instance.client[settings.MONGODB_DB_NAME]

async def close_mongo_connection():
       Cerrar conexion a MongoDB.
    db_instance.client.close()

def get_database():
       Obtener instancia de la base de datos.
    return db_instance.db
```

Listing 17. Conexión asíncrona a MongoDB. Fuente: Autor Propio.

XII-V. Cliente API del frontend (React.js)

```javascript
const API_BASE_URL = (
    window.location.hostname !== 'localhost'
) ? `${window.location.origin}/api`
  : process.env.REACT_APP_API_BASE_URL || `${window.location.origin}/api`;

const fetchWithAuth = async (url, options = {}) => {
    const token = localStorage.getItem('authToken');
    const headers = { 'Content-Type': 'application/json', ...options.headers };
    if (token) headers['Authorization'] = `Bearer ${token}`;

    const response = await fetch(url, { ...options, headers });
    if (!response.ok) {
        const error = await response.json().catch(() => ({}));
        throw new Error(error.message || 'Error en la peticion');
    }
    return response.json();
};

// Prediccion con ensemble de 5 modelos
export const predictionAPI = {
    analyze: async (imageFile) => {
        const formData = new FormData();
        formData.append('image', imageFile);
        const response = await fetch(
            `${window.location.origin}/api/predict/`,
            { method: 'POST', body: formData });
        if (!response.ok) {
            const err = await response.json().catch(() => ({}));
            throw new Error(err.detail || 'Error en el analisis');
        }
        return response.json();
    },

    // Prediccion con modelo personalizado
    analyzeCustomModel: async (imageFile) => {
        const formData = new FormData();
        formData.append('image', imageFile);
        const response = await fetch(
            `${window.location.origin}/api/predict/custom-model`,
            { method: 'POST', body: formData });
        if (!response.ok) {
            const err = await response.json().catch(() => ({}));
            throw new Error(err.detail || 'Error con modelo personalizado');
        }
        return response.json();
    },
```

```
47
48      // Descarga de reporte PDF
49      downloadReport: async (imageFile) => {
50          const formData = new FormData();
51          formData.append('image', imageFile);
52          const response = await fetch(
53              `${window.location.origin}/api/predict/report`,
54              { method: 'POST', body: formData });
55          if (!response.ok) {
56              const err = await response.json().catch(() => ({}));
57              throw new Error(err.detail || 'Error al generar reporte');
58          }
59          return response.blob();
60      }
61  };
```

Listing 18. Cliente API del frontend (React.js). Fuente: Autor Propio.