Maestría en

Ciencia de Datos y Máquinas de Aprendizaje con mención en Inteligencia Artificial

Trabajo previo a la obtención de título de Magíster en

Ciencia de Datos y Máquinas de Aprendizaje

AUTOR/ES:

Cristhian Nahim Cobos Morales

Alisson Monserratte López Mejía

Kevin Orlando Jiménez Saraguro

Diana Carolina Ortiz Pillajo

TUTOR/ES:

Karla Mora

Alejandro Cortés

TEMA:

Desarrollo de un sistema de visión por computadora e inteligencia artificial para la detección de sigatoka negra, fusarium raza 4 tropical (Foc R4T), cordana, pestalotiopsis y moko bacteriano en banano ecuatoriano.

Quito – Ecuador

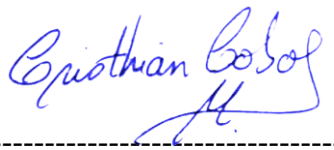
diciembre - 2025

REINVENTEMOS EL FUTURO

Certificación de autoría

Nosotros, **Cristhian Nahim Cobos Morales, Alisson Monserratte López Mejía, Kevin Orlando Jiménez Saraguro, Diana Carolina Ortiz Pillajo,** declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido presentado anteriormente para ningún grado o calificación profesional y que se ha consultado la bibliografía detallada.

Cedemos nuestros derechos de propiedad intelectual a la Universidad Internacional del Ecuador (UIDE), para que sea publicado y divulgado en internet, según lo establecido en la Ley de Propiedad Intelectual, su reglamento y demás disposiciones legales.



Firma
Cristhian Nahim Cobos Morales

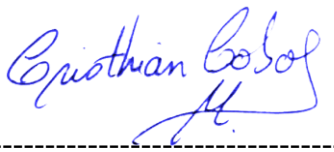
Firma
Alisson Monserratte López Mejía

Firma
Kevin Orlando Jiménez Saraguro

Firma
Diana Carolina Ortiz Pillajo

Autorización de Derechos de Propiedad Intelectual

Nosotros, Diana Carolina Ortiz Pillajo, Cristhian Nahim Cobos Morales, Kevin Orlando Jiménez Saraguro, Alisson Monserratte López Mejía , en calidad de autores del trabajo de investigación titulado Desarrollo de un sistema de detección de sigatoka negra, fusarium R4T y moko bacteriano en banano ecuatoriano mediante visión por computadora e inteligencia artificial, autorizamos a la Universidad Internacional del Ecuador (UIDE) para hacer uso de todos los contenidos que nos pertenecen o de parte de los que contiene esta obra, con fines estrictamente académicos o de investigación. Los derechos que como autores nos corresponden, lo establecido en los artículos 5, 6, 8, 19 y demás pertinentes de la Ley de Propiedad Intelectual y su Reglamento en Ecuador. D. M. Quito, diciembre 2025



Firma
Cristhian Nahim Cobos Morales

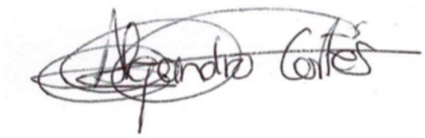
Firma
Alisson Monserratte López Mejía

Firma
Kevin Orlando Jiménez Saraguro

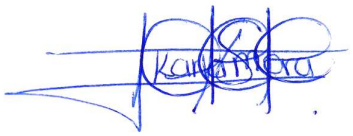
Firma
Diana Carolina Ortiz Pillajo

Aprobación de dirección y coordinación del programa

Nosotros, **Alejandro Cortés el Director EIG y Karla Mora Coordinadora UIDE,** declaramos que: **(Diana Carolina Ortiz Pillajo, Cristhian Nahim Cobos Morales, Kevin Orlando Jiménez Saraguro, Alisson Monserratte López Mejía)** son los autores exclusivos de la presente investigación y que ésta es original, auténtica y personal de ellos.



Alejandro Cortés
Director/a de la
Maestría en Ciencia de datos y máquinas de aprendizaje con mención en Inteligencia Artificial



Karla Mora
Coordinador/a de la
Maestría en Maestría en Ciencia de datos y máquinas de aprendizaje con mención en Inteligencia Artificial

Dedicatoria

A mi familia, por su amor y apoyo incondicional; por ser testigos de mi esfuerzo a lo largo de esta etapa y por nunca dejar de creer en mí.

Kevin Jimenez

A Isabella y Julieta como recordatorio de que el cielo es el límite.

Diana Ortiz

A mi familia, por ser mi guía e inspiración constante y acompañarme con amor siempre.

Alisson López

A mis padres, hermanos y sobrinos, que me acompañan en cada paso que doy.

Cristhian Cobos

Agradecimientos

Agradezco a Dios por acompañarme a cumplir una meta más; a mi familia, Edwin, Nelly y Leslie, por su amor y apoyo constante; a Kevin, porque todo es más bonito a tu lado; y a mis docentes, por su guía y conocimientos brindados.

Alisson López

Agradezco a Dios y a la Virgen por guiar cada uno de mis pasos. A mis padres y hermana, por su amor incondicional, su apoyo constante y por creer en mí en todo momento. A los docentes, por las grandes enseñanzas que nos dejaron., de manera muy especial, a mi amor Alisson, por estar siempre a mi lado, cuidándome, apoyándome y acompañándome en cada etapa de mi vida.

Kevin Jimenez

A mis padres Martha y Marco, a mis hermanos David, Gaby, Andrés, Diana, Iván y Lennin. a mis sobrinos Eitan, Sae, Gabriel, Emilio y Emma, a mi buen amigo Steven, ustedes son mi soporte y motivación en cada paso que doy.

Cristhian Cobos

Agradezco a Dios por su guía y por permitirme culminar este proceso. A mi esposo, por su amor, paciencia y apoyo constante a lo largo de este camino. A mis padres, maestros y tutores, por su orientación y enseñanzas. De manera especial, al Ingeniero Pablo Zamora, por su valiosa colaboración en la obtención de imágenes que hicieron posible el desarrollo de este proyecto.

Diana Ortiz

Resumen

En Ecuador el cultivo de banano representa un pilar económico y social de alta importancia, sin embargo, su sostenibilidad depende directamente de la supervivencia de las plantas para sobrevivir a enfermedades fitosanitarias que reducen en gran porcentaje su productividad lo que genera pérdidas económicas significativas. El presente trabajo propone un desarrollo de un sistema web basado en visión por computador y aprendizaje profundo para la detección automatizada de enfermedades en las hojas del banano a partir de imágenes. El aplicativo emplea técnicas de deep learning con arquitecturas como YOLOv8 y EfficientNet utilizando imágenes de hojas de banano sanas y con sintomatología asociada a enfermedades como Sigatoka negra, fusarium R4T y moko bacteriano, cordana, pestalotiopsis. El desarrollo del proyecto sigue un enfoque cualitativo y experimental, validando los modelos con métricas importantes como matriz de confusión, exactitud, precisión, recall y F1-score, este último cuando sea el caso.Estos modelos se integran en una arquitectura modular compuesta por back-end de inferencia implementado en Python y una interfaz web para la interacción del usuario, esta demo fue desplegada en un entorno local y expuesto mediante túneles seguros como ngrok, permitiendo la demostración funcional del sistema. Los resultados obtenidos evidencian un buen desempeño en la clasificación de enfermedades evaluadas, dejando en evidencia el potencial como herramienta de apoyo para evitar el diagnóstico tardío.

Palabras clave:

Visión por computador, aprendizaje profundo, enfermedades del banano, YOLOv8, EfficientNet, diagnóstico automatizado.

Abstract

In Ecuador, banana cultivation is an economic and social pillar of great importance. However, its sustainability depends directly on the survival of plants against phytosanitary diseases that greatly reduce their productivity, leading to significant economic losses. This paper proposes the development of a web-based system using computer vision and deep learning for the automated detection of diseases in banana leaves from images.The application uses deep learning techniques with architectures such as YOLOv8 and EfficientNet, using images of healthy banana leaves and leaves with symptoms associated with diseases such as black sigatoka, Fusarium R4T and bacterial moko, cordana, and pestalotiopsis. The project's development follows a qualitative and experimental approach, validating the models with important metrics such as confusion matrix, accuracy, precision, recall, F1-score, the latter when applicable.These models are integrated into a modular architecture composed of an inference backend implemented in Python and a web interface for user interaction. This demo was deployed in a local environment and exposed through secure tunnels such as ngrok, allowing for a functional demonstration of the system. The results obtained show good performance in the classification of the diseases evaluated, highlighting the potential of this tool as a support for avoiding late diagnosis.

Keywords: *Computer vision, deep learning, banana diseases, YOLOv8, EfficientNet, automated diagnosis.*

Tabla de contenido

Lista de tablas

Lista de figuras

Capítulo 1

1. Introducción

1.1.Definición del proyecto

El proyecto consiste en desarrollar un sistema para la detección automatizada de enfermedades en el banano mediante técnicas de visión por computadora y modelos de aprendizaje profundo. El sistema se entrena con imágenes de hojas y plantas, provenientes de repositorios públicos y adquiridas directamente en cultivos de banano. El enfoque considera una preparación del dataset mediante depuración y aumentación, el etiquetado manual de las clases, la delimitación de regiones de interés y la segmentación de imágenes. Posteriormente, se comparan dos modelos de detección: el primero un CNN basado en arquitecturas disponibles en timm y el segundo un modelo YOLOv8, validados mediante métricas estándar. Estos modelos se integran en una arquitectura modular compuesta por un back-end y una interfaz de usuario, los cuales se implementan en un prototipo desplegado en un entorno local expuesto mediante túneles seguros con ngrok, como demostración funcional del proyecto.

1.2. Justificación e importancia del trabajo de investigación

El banano es un pilar económico y social alrededor del mundo debido a su gran aporte en la seguridad alimentaria. La cadena de producción proviene principalmente de países de Centroamérica, Sudamérica y Filipinas representando el 90% de las exportaciones de este producto a nivel mundial (Food and Agriculture Organization of the United Nations [FAO], s. f.).

Estos cultivos son los responsables de generar millones de empleos directos e indirectos en países tropicales o del neotrópico (Jones, 2000, como se citó en Churchill, 2011), este cultivo

adquiere alta relevancia, ya que representa el 60% del banano importado en el 2022 desde Sudamérica (Munhoz et al., 2024). Para Ecuador es uno de los principales productos de exportación. Los cultivos de las musáceas representan alrededor de 345.765 hectáreas de cultivo (Agencia de Regulación y Control Fito y Zoosanitario, 2025, p.1). Durante el 2024 se reportaron cerca de USD 1.94 mil millones (Corporación Financiera Nacional [CFN], 2024) representando un producto de importancia no solo agrícola, sino también productiva. Según datos del MAGAP (2023, p. 1) la actividad bananera contribuye alrededor de 25.4% al VAB agropecuario; Sin embargo, la sostenibilidad del cultivo se ve comprometida por las presencias de fitopatologías que son de especial atención, ya que enfermedades como; sigatoka negra, el *Fusarium oxysporum* f. El Sp cubense raza 4 tropical (Foc R4T) y el moko bacteriano representan una amenaza directa para los cultivos (Churchill, 2011; Pinzón-Núñez et al., 2024), la sigatoka negra *Pseudocercospora fijiensis***,** es reconocida como una de las principales amenazas, ya que reduce la capacidad fotosintética de la planta y con ello el rendimiento de la cosecha (Noar et al., 2022).

Por otro lado, el moko es causada por *Ralstonia solanacearum* raza 2 es una de las plagas más destructivas contra la producción, ya que pueden representar hasta el 100% de la pérdida y la sintomatología principal es la afectación al sistema vascular empezando por el amarillamiento de las hojas, este se encuentra entre los responsables más relevantes de la marchitez en esta especie de cultivo (Grajales-Amorocho et al., 2024). Por otro lado, *Fusarium oxysporum f. sp*. cubense Raza 4 Tropical (Foc TR4)**,** representa una amenaza latente para los cultivos, aunque actualmente no hay evidencia de que esté presente en el país ecuatoriano se ha confirmado su presencia en 21 países, el contagio empieza por el suelo y su manejo resulta difícil y costoso (Agencia de Regulación y Control Fito y Zoosanitario, 2024).

En la actualidad el diagnóstico de estas enfermedades se realiza mediante la inspección personal y manual con personal capacitado, y una confirmación posterior mediante pruebas moleculares, lo que representa un proceso demorado y costoso en especial frente a grandes hectáreas de cultivo (Blandón et al., 2024). Además de estar sujetos a sesgos. Las soluciones basadas en IA permiten diagnósticos más rápidos y consistentes, apoyando la agricultura de precisión y la sostenibilidad de la industria (Jiménez et al., 2025; Romero-García et al., 2025, pp. 68-70).

1.3.Alcance

Detección automatizada mediante visión computarizada de tres enfermedades infecciosas de banano: sigatoka negra, el *Fusarium oxysporum* f. El Sp cubense raza 4 tropical (Foc TR4) y moko bacteriana, mediante técnicas de deep learning para clasificación y detección de imágenes con la comparación de dos modelos YOLOv8 y EfficientNet. La metodología parte de la construcción de dos datasets a partir de imágenes públicas y capturas obtenidas en campo, las cuales fueron sometidas a procesos de etiquetado, transformación y organización. Posteriormente, se realiza el entrenamiento de modelos y la validación cuantitativa con métricas estándar. El producto final consiste en un despliegue del prototipo en entorno local expuesto mediante túneles seguros con ngrok a modo de demostración, por lo que los resultados se orientan a la validación experimental y académica del enfoque propuesto.

1.4.Objetivos

1.4.1. Objetivo general.

Desarrollar un sistema de visión por computadora e inteligencia artificial para la detección de sigatoka negra, fusarium raza 4 tropical (Foc R4T), cordana, pestalotiopsis y moko bacteriano en banano ecuatoriano.

1.4.2. Objetivos específicos.

Construir un dataset de imágenes con sintomatología de sigatoka negra, fusarium raza 4 tropical (Foc R4T) y moko bacteriano, así como imágenes de plantas sanas, a partir de fuentes públicas y capturas de campo, mediante procesos de etiquetado, limpieza y transformación.

Implementar dos arquitecturas de deep learning, YOLOv8 y EfficientNet, para la detección de las tres enfermedades a partir del conjunto de datos anteriormente generados.

Comprobar el desempeño de las arquitecturas mediante métricas de evaluación.

Desplegar un prototipo funcional en entorno local expuesto mediante túneles seguros con ngrok como demostración del sistema desarrollado.

Capítulo 2

2. Revisión de literatura

2.1.Estado del arte

Durante la última década, la visión por computadora y, en particular, el aprendizaje profundo, han representado una vía práctica para el diagnóstico de fitopatologías a partir de imágenes, impulsadas por la disponibilidad de bases de datos abiertas y el rendimiento de las redes neuronales convolucionales (CNN). Un hito ampliamente citado es el uso de PlantVillage, repositorio que habilitó la investigación masiva de clasificación de enfermedades de hojas mediante aprendizaje profundo (Mohanty et al., 2016).

Sin embargo, la evidencia acumulada señala que los modelos entrenados y evaluados en condiciones controladas pueden degradar su desempeño en campo por cambios de dominio (iluminación, fondo, variabilidad de cámaras, oclusiones y ruido). Esto se documenta tanto en análisis experimentales sobre limitaciones del reconocimiento en condiciones reales (Barbedo, 2018) como en estudios específicos sobre sesgos del dataset PlantVillage, donde variables no relacionadas con el síntoma pueden contribuir al desempeño, como fondos o patrones engañosos (Noyan, 2022). En conjunto, este tipo de evidencia desplaza el enfoque desde “alta exactitud en laboratorio” hacia “robustez y generalización en condiciones reales”, que constituye uno de los retos centrales del área.

A nivel metodológico, el campo ha evolucionado desde la clasificación de hojas completas hacia tareas más específicas y operativas: (i) detección y (ii) segmentación de lesiones para localizar síntomas, (iii) estimación de severidad para aproximarse a escalas fitopatológicas, y (iv) integración con georreferenciación y plataformas móviles o UAV (drones) para monitoreo

y agricultura de precisión. Específicamente en el cultivo de banano, esta progresión se observa en trabajos que exploran detección temprana (p. ej., con imágenes hiperespectrales para sigatoka negra) y en propuestas que buscan despliegue en dispositivos móviles para asistencia al agricultor (Ugarte Fajardo et al., 2020; Sanga et al., 2020).

En paralelo, se ha incrementado la disponibilidad de conjuntos de datos específicos para el banano; por ejemplo, se han publicado datasets para clasificación de enfermedades foliares, como BananaLSD con sigatoka, cordana y pestalotiopsis; así como datasets más recientes con mayor variedad, orientados a escenarios de captura menos controlados (Arman et al., 2023; Mduma & Elinisa, 2025). Este avance es especialmente relevante porque habilitan evaluaciones más cercanas al uso real y reduce la dependencia exclusiva de PlantVillage.

2.2.Marco teórico

2.2.1. Cultivo de banano

El banano es uno de los cultivos de mayor relevancia económica y alimentaria a nivel mundial, ocupando el cuarto lugar entre los productos alimenticios básicos en términos de volumen de producción. En Ecuador es muy importante para la parte agrícola, además, es un sector estratégico (Cervantes-Álava et al., 2023). Según Cervantes-Álava et al. (2023, p. 2), el banano aporta directamente el 2 % del Producto Interno Bruto (PIB) en el ámbito comercial y social, y la industria bananera genera aproximadamente 1 millón de familias y beneficia a 2,5 millones de personas, lo que representa el 6% de la población total del país.

La industria bananera genera aproximadamente 1 millón de empleos directos e indirectos, lo que representa el 6 % de la población total del país, incluyendo productores, trabajadores de plantaciones, transportistas, personal de empacadoras, y trabajadores de servicios auxiliares.

Además, el sector concentra cerca del 45 % de las exportaciones agrícolas totales y genera ingresos superiores a los 3.000 millones de dólares anuales, lo que convierte a Ecuador en el mayor exportador mundial de banano, con una participación del 28-30 % del mercado global.

2.2.2. Enfermedades del banano

El cultivo de banano es altamente susceptible a diversas enfermedades causadas por patógenos fúngicos, bacterianos y virales, las cuales representan una de las principales limitantes para su productividad y sostenibilidad. De acuerdo con Arunkumar y Suthin Raj (2021), estas enfermedades no solo afectan el rendimiento y la calidad del fruto, sino que también incrementan los costos de producción debido a la necesidad de implementar medidas de control constantes y, en muchos casos, intensivas. El control químico de enfermedades como la sigatoka negra puede incrementar los costos de producción en más del 30% (Cervantes-Álava et al., 2023, p. 2).

Entre las enfermedades más relevantes se encuentran la marchitez por Fusarium y la sigatoka negra, consideradas las más devastadoras a nivel mundial (Arunkumar & Suthin Raj, 2021). Estas patologías se caracterizan por su rápida diseminación, persistencia en el suelo y dificultad de erradicación una vez establecidas. Asimismo, enfermedades bacterianas como la pudrición blanda y virales como el Banana bunchy top virus generan pérdidas significativas, especialmente en sistemas de producción con bajo manejo fitosanitario (Arunkumar & Suthin Raj, 2021).

El manejo de estas enfermedades requiere un enfoque integral que combine prácticas culturales, uso de material vegetal sano, variedades resistentes y control químico racional (Arunkumar & Suthin Raj, 2021). El Manejo Integrado de Enfermedades (MID) se presenta como la estrategia más efectiva, ya que permite reducir la dependencia de agroquímicos,

minimizar el impacto ambiental y mantener la viabilidad económica del cultivo. Este enfoque se fundamenta en la prevención, el monitoreo constante y la aplicación oportuna de medidas de control basadas en el conocimiento del patógeno y su interacción con el ambiente y el hospedero (Arunkumar & Suthin Raj, 2021).

Tabla 1

Principales enfermedades foliares y sistémicas del banano

Enfermedad	**Tipo**	**Órgano afectado**	**Síntomas representativos**
Cordana	Fúngica	Hoja	Manchas ovaladas a alargadas, de color marrón claro con bordes oscuros, generalmente rodeadas por halos amarillos
Pestalotiopsis	*Pestalotiopsis spp*	Hoja	Manchas irregulares de color marrón oscuro con bordes definidos, necrosis progresiva del tejido foliar
Sigatoka (negra/amarilla)	*Pseudocercospora fijiensis / Mycosphaerella musicola*	Hoja	Estrías cloróticas iniciales que evolucionan a manchas necróticas oscuras; reducción del área fotosintética
Fusarium TR4	*Fusarium oxysporum f. sp.* cubense raza Tropical 4	Hoja	Amarillamiento progresivo, marchitez, necrosis vascular interna, colapso de la planta
Moko negro	*Ralstonia solanacearum*	Sistema vascular	Marchitez, exudado bacteriano, pudrición interna del pseudo tallo y frutos
Pudrición bacteriana	Bacteriana	Sistema vascular y fruto	Pudrición blanda, mal olor, colapso de tejidos
Banana bunchy top	Viral		Hojas erectas y estrechas, crecimiento atrofiado, reducción del racimo

Nota. Adaptado de "Evaluación de fungicidas utilizados en el manejo de sigatoka negra en el cultivo de banano", por A. Cervantes-Álava et al. (2023).

2.2.3. Redes neuronales convolucionales

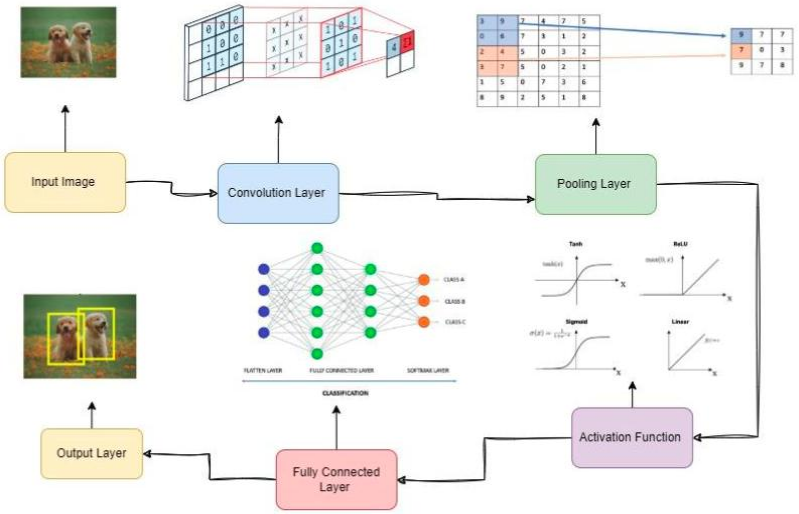
Las Redes Neuronales Convolucionales (CNN) se fundamentan en principios matemáticos y estructurales que explican su alta capacidad para el análisis de datos visuales. Estas arquitecturas aprovechan la estructura espacial de las imágenes mediante operaciones de convolución, permitiendo capturar patrones locales y relaciones jerárquicas de manera eficiente. A diferencia de modelos tradicionales, las CNN reducen la complejidad del aprendizaje gracias al uso de conectividad local y pesos compartidos, lo que mejora su generalización y desempeño en tareas de visión por computador (Taye, 2023).

Las capas convolucionales actúan como extractores automáticos de características, donde los primeros niveles aprenden patrones simples como bordes y contornos, mientras que las capas más profundas representan estructuras de mayor nivel semántico (Taye, 2023). Esta organización jerárquica permite que las CNN sean robustas frente a variaciones espaciales, ruido y transformaciones menores en las imágenes de entrada. Además, el uso de funciones de activación no lineales incrementa la capacidad del modelo para aproximar funciones complejas (Taye, 2023).

Las CNN pueden interpretarse como sistemas que combinan principios de invariancia, estabilidad y composicionalidad, lo cual explica su éxito en problemas de clasificación y reconocimiento visual (Taye, 2023). Estas propiedades hacen que las CNN sean especialmente adecuadas para aplicaciones donde la información espacial es relevante, como el análisis de imágenes médicas o agrícolas, incluyendo la detección automática de enfermedades en hojas de plantas.

Figura 1

Arquitectura de CNN



Nota. La arquitectura muestra el flujo de procesamiento desde la imagen de entrada, pasando por la capa convolucional, la capa de pooling, hasta la capa totalmente conectada y la función de activación. Adaptada de Taye (2023).

2.3.El cultivo de banano y la problemática fitosanitaria

El banano (Musa spp.) es un cultivo de amplia distribución en regiones tropicales y subtropicales y constituye un eje económico para múltiples países productores. En sistemas orientados a exportación predomina el subgrupo Cavendish, apreciado por rendimiento y características comerciales, aunque con susceptibilidades relevantes ante enfermedades foliares y vasculares (Churchill, 2011; Ploetz, 2015).

Desde la perspectiva fitosanitaria, este trabajo se enfoca en tres problemas frecuentes y de alto impacto: Sigatoka negra, marchitez por *fusarium oxysporum* f. sp. *cubense* (con énfasis en TR4) y Moko del banano asociado a *Ralstonia solanacearum*. Estas enfermedades reducen rendimiento y calidad comercial, incrementan costos de control y monitoreo, y en escenarios de

alta presión de patógeno pueden comprometer la viabilidad de plantaciones completas (Churchill, 2011; Ploetz, 2015).

2.3.1. Sigatoka negra

La sigatoka negra (black leaf streak disease) es causada por *Pseudocercospora fijiensis* anteriormente clasificada como *Mycosphaerella fijiensis* y se reconoce como una de las enfermedades foliares más importantes del banano a nivel mundial (Churchill, 2011). El patógeno coloniza el tejido foliar y produce estrías cloróticas que progresan a lesiones necróticas, como se aprecia en la Figura 2. Con la pérdida de área fotosintética funcional reduce la productividad y puede forzar cosechas anticipadas.

En el control, además de prácticas culturales, se han empleado programas intensivos de fungicidas. Para estandarizar evaluación y reducir variabilidad entre observadores, se proponen escalas diagramáticas de severidad, aunque el proceso sigue siendo esencialmente visual y dependiente del evaluador (Pinzón-Núñez et al., 2024).

Figura 2

Síntomas en hoja producidos por sigatoka negra en banano (Musa spp)

Nota. Reproducida de Linero-Ramos et al. (2024), Figura 7B, *Drones, 8*(9), 503.

2.3.2. Marchitez por Fusarium y relevancia de TR4

La marchitez por *Fusarium* en banano es causada por *Fusarium oxysporum* f. sp. *cubense* (Foc), patógeno de suelo que invade raíces y xilema Figura 3. La variante conocida como tropical race 4 (TR4) es particularmente crítica por su capacidad de afectar Cavendish y por su persistencia en suelo, lo que limita estrategias curativas y desplaza la gestión hacia medidas preventivas, bioseguridad y detección temprana (Ploetz, 2015).

En años recientes, la literatura ha enfatizado la expansión y gestión de TR4, destacando la necesidad de vigilancia, contención, y prácticas integradas de manejo, especialmente en regiones productoras de alto peso exportador (Munhoz et al., 2024; Ploetz, 2015).

Figura 3

Síntomas en hoja producidos por Fusarium TR4 en banano (Musa spp.)

Nota. Fotografía de dominio público obtenida de Flickr (Nelson, 2011)

2.3.3. Cordana

También llamada mancha foliar diamante, es causada por un complejo fúngico especialmente por *Neocordana musae*, los signos están asociados con son lesiones necróticas con anillos concéntricos y halo amarillo brillante Figura 4, esta fitopatología se asocia a otras similares y pueden caer en infecciones mixtas (Fu et al., 2021)

Figura 4

Síntomas en hoja producidos por Cordana en banano (musa spp.)

Nota. Adaptado de P9140533, por (Nelson, 2011), Flickr

2.3.4. Pestalotiopsis

Este nombre recibe las enfermedades causadas por hongos de la familia *pestalotiopsis*, entre los signos que presenta la hoja enferma son manchas, tizones y necrosis Figura 5, por lo que la presencia de estos puede causar un impacto directo en la sanidad vegetal (Cui et al., 2024).

Figura 5

Síntomas en hoja producidos por Pestalotiopsis en banano (musa spp.)

Nota. Tomado de BananaLSD (Arman et al., 2023)

2.3.5. Moko del banano

El Moko del banano es una marchitez bacteriana asociada a *Ralstonia solanacearum* en varios documentos técnicos se reporta en términos de “razas” o “patotipos” según el enfoque institucional. La diseminación puede ocurrir por suelo, agua, herramientas, vectores y material de propagación, y el manejo se centra en erradicación temprana, desinfección rigurosa y uso de material certificado, dada la severidad del impacto y la velocidad de propagación en finca (Agencia de Regulación y Control Fito y Zoosanitario, 2024).

Figura 6

Síntomas en hoja producidos por moko en banano (Musa spp.)

Nota. Adaptada de la Figura 4 de Blomme et al. (2017), *Frontiers in Plant Science*. Artículo de acceso abierto bajo licencia Creative Commons Attribution (CC BY).

2.3.6. Diagnóstico visual y motivación para automatización

En los tres casos descritos, la inspección visual es el punto de partida operacional en campo. No obstante, presenta limitaciones: alta dependencia de experiencia, costo de cobertura en grandes superficies, variabilidad por condiciones de luz y heterogeneidad del entorno, y dificultad para detectar estadios tempranos. Estas restricciones justifican el interés por sistemas de apoyo basados en imágenes y análisis automático, especialmente cuando las capturas pueden provenir de teléfonos inteligentes, cámaras convencionales o plataformas UAV (Barbedo, 2018; Upadhyay et al., 2025).

2.4. Fundamentos de visión por computador y aprendizaje profundo

La visión por computador busca dotar a sistemas automáticos de la capacidad de interpretar información visual para resolver tareas como clasificación, detección de objetos y segmentación. Con el auge del aprendizaje profundo, estas tareas han mejorado sustancialmente al aprender representaciones directamente desde datos, evitando depender exclusivamente de descriptores diseñados manualmente (Goodfellow et al., 2016).

2.4.1.1. Imágenes digitales y preprocesamiento

Una imagen RGB puede representarse como una matriz de píxeles con tres canales. Antes del entrenamiento, se aplican transformaciones típicas: redimensionamiento, normalización y aumento de datos rotaciones, cambios de brillo, recortes, etc. En sanidad vegetal, el aumento de datos contribuye a robustez ante cambios habituales en campo (iluminación, ángulo, fondo), aunque no reemplaza la necesidad de datos representativos del dominio real (Barbedo, 2018).

2.4.2. Redes neuronales convolucionales

Las CNN aplican filtros convolucionales para capturar patrones locales (bordes, texturas), y en capas profundas aprenden estructuras más abstractas asociadas a la clase. Su entrenamiento ajusta parámetros para minimizar una función de pérdida, usualmente entropía cruzada, mediante variantes de descenso de gradiente sobre imágenes etiquetadas. En fitopatología, esto se traduce en aprender texturas, clorosis, necrosis y patrones de lesión en lámina foliar (Mohanty et al., 2016).

2.4.3. Transfer learning y modelos preentrenados

Dado que los datasets agrícolas suelen ser más pequeños que repositorios masivos, se emplea transfer learning: se parte de modelos preentrenados (comúnmente sobre ImageNet) y se ajustan a la nueva tarea. Este enfoque es dominante en el diagnóstico de enfermedades por hoja

por su eficiencia y desempeño con datos moderados (Mohanty et al., 2016; Russakovsky et al., 2015).

2.4.4. Métricas de evaluación e interpretabilidad

La evaluación utiliza métricas como accuracy, precisión, recall, F1 y matrices de confusión, poniendo atención a desempeño por clase relevante por el costo asimétrico de falsos negativos en enfermedad. Además, la interpretabilidad ha ganado importancia: Grad-CAM permite visualizar regiones que contribuyen a la decisión del modelo, facilitando validación técnica y comunicación con expertos agronómicos (Selvaraju et al., 2020).

2.5.Detección de enfermedades de plantas mediante deep learning

El uso de deep learning para enfermedades de plantas se ha expandido rápidamente, apoyado en revisiones sistemáticas que reportan mejoras sostenidas en desempeño y proliferación de arquitecturas eficientes. Estas revisiones también resaltan retos persistentes: sesgo de datasets, desbalance de clases, robustez en campo y necesidad de interpretabilidad (Liu & Wang, 2021; Upadhyay et al., 2025).

En particular, el caso PlantVillage consolidó la línea de clasificación con CNN, mostrando niveles altos de desempeño en pruebas internas (Mohanty et al., 2016). No obstante, trabajos posteriores evidencian que parte del desempeño puede explicarse por correlaciones espurias (p. ej., fondos, condiciones controladas), reforzando el argumento de que la evaluación debe incluir escenarios realistas y protocolos de generalización (Barbedo, 2018; Noyan, 2022).

2.6.Arquitecturas YOLOv8 Y EfficientNet en segmentación y etiquetado foliar

El deep learning se emplea dentro del diagnóstico de fitopatologías para; la segmentación foliar y el etiquetado, entre las familias más utilizadas destacan YOLO que usan un problema de

regresión directa sobre la imagen completa y permite procesar la información en tiempo real (Redmon et al., 2016) y las redes convolucionales eficientes como EfficientNet que optimiza los factores precisión con costo computacional (Tan & Le, 2019).

2.6.1. YOLOv8

Arquitectura desarrollada y mantenida por Ultralytics, incorpora un backbone optimizado, variantes orientadas a la detección como a segmentación y clasificación, adopta un enfoque simplificando los anchos con lo que se mejora la localización, mientras mantiene velocidades de inferencia para sistemas en tiempo real (Ultralytics, 2023).

2.6.2. EfficientNet

EfficienteNet está diseñada a partir de una búsqueda de arquitectura y un sistema que ajusta ancho, profundidad y resolución en red mediante un solo coeficiente de escala, por lo que comprende una familia de redes convulsiones (Tan & Le, 2019)

Capítulo 3

3. Desarrollo

3.1.Metodología del desarrollo del proyecto

El desarrollo del presente proyecto se fundamenta dentro de un enfoque metodológico mixto que combina elementos cuantitativos y cualitativos, orientado a la construcción de un sistema de clasificación de enfermedades en hojas de banano mediante técnicas de aprendizaje profundo. La metodología adoptada sigue un modelo iterativo e incremental, característico del desarrollo de sistema de inteligencia artificial, donde cada fase permite refinamientos sucesivos basados en los resultados obtenidos.

El proceso metodológico se estructura en seis fases principales interconectadas: (1) adquisición y preparación del conjunto de datos, (2) diseño de la arquitectura del sistema, (3) implementación de los modelos de clasificación, (4) entrenamiento y optimización de hiperparámetros, (5) evaluación y validación del desempeño, y (6) desarrollo e integración del prototipo funcional. Cada fase incorpora ciclos de retroalimentación que permiten ajustes basados en métricas de rendimiento y observaciones cualitativas.

La validación cuantitativa del sistema se realiza mediante métricas estándar de clasificación multiclase, incluyendo exactitud (*accuracy*), precisión (*precision*), exhaustividad (recall) y puntuación F1 (F1-score). Adicionalmente, se emplean matrices de confusión para el análisis detallado del comportamiento del clasificador por clase, permitiendo identificar patrones de confusión entre enfermedades con sintomatología visual similar.

El enfoque experimental contempla la comparación sistemática de dos familias de arquitecturas de redes neuronales profundas: redes convolucionales tradicionales basadas en

EfficientNet , y arquitecturas de detección de objetos adaptadas para clasificación mediante YOLOv8. Esta comparación permite evaluar no solo el rendimiento e n términos de exactitud, sino también aspectos críticos para el despliegue como velocidad de inferencia, consumo de memoria y tamaño del modelo.

3.2.Arquitectura general del sistema

El sistema desarrollado implementa una arquitectura de software modular basada en el patrón cliente-servidor, diseñada para facilitar escalabilidad, el mantenimiento y la evolución independiente de sus componentes. Esta arquitectura separa claramente las responsabilidades entre la interfaz del usuario, la lógica del negocio, los servicios de inferencia y la capa de persistencia de datos.

La arquitectura del sistema se compone de cuatro capas fundamentales que operan de manera coordinada para proporcionar el servicio de clasificación de enfermedades.

Capa de presentación (*Front-end*): Implementada como una aplicación web estática utilizando tecnologías HTML5, CSS3 y JavaScript vanilla. Esta capa proporciona la interfaz de usuario para la carga de imágenes, visualización de resultados de clasificación y consulta del historial de predicciones. El diseño responsivo garantiza accesibilidad desde dispositivos móviles, considerando que los usuarios finales podrían utilizar smartphones directamente en campo.

Capa de servicios (API REST): Desarrollada con el *framework* FastAPI de Python, esta capa expone *endpoints* RESTful que gestionan las solicitudes del cliente. FastAPI fue seleccionado por su alto rendimiento, soporte nativo para operaciones asíncronas, generación automática de documentación OpenAPI y validación de tipos mediante Pydantic. Los principales

endpoints incluyen: /predict para clasificación de imágenes, \history para consulta de predicciones anteriores, \feedback para retroalimentación del usuario, y \stats para estadísticas agregadas.

Capa de inferencia (Motor de clasificación): Constituye el núcleo del sistema donde residen los modelos de aprendizaje profundo entrenados. Esta capa encapsula toda la lógica de preprocesamiento de imágenes, ejecución del modelo y posprocesamiento de resultados. Se implemente como un módulo independiente que puede cargar diferentes arquitecturas de modelo (EfficientNet, YOLOv8) según la configuración, facilitando la experimentación y actualización de modelos sin modificar otras capas del sistema.

Capa de persistencia (Base de datos): Implementada con PostgreSQL, esta capa almacena el historial de predicciones, retroalimentación de usuarios y metadatos del sistema. El esquema de base de datos está diseñado para soportar análisis posteriores del comportamiento del modelo en producción y facilitar procesos de mejora continua mediante el etiquetado de nuevas muestras.

3.3. Flujo de datos del sistema

El flujo de procesamiento de una solicitud de clasificación sigue una secuencia bien definida que garantiza la trazabilidad y reproducibilidad de los resultados:

Paso 1 – Captura de imagen: El usuario carga una imagen de hoja de banano a través de la interfaz web. El frontend valida el formato del archivo (JPEG, PNG) y el tamaño máximo permitido antes de enviar la solicitud al servidor.

Paso 2 – Recepción y validación: El endpoint /predict de la API recibe la imagen codificada en formato multipart/form-data. Se realizan validaciones adicionales incluyendo verificación de integridad del archivo y detección del tipo MIME real.

Paso 3 – Preprocesamiento: La imagen se decodifica y transforma según los requerimientos del modelo, redimensionando a 224x224 píxeles, normalización de valores de pixeles al rengo [0,1], y aplicación de las trasformaciones específicas del modelo preentrenado (normalización ImageNet para EfficientNet).

Paso 4 – Inferencia: El tensor preprocesado se pasa al modelo de clasificación. El motor de inferencia ejecuta el forward pass y obtiene las probabilidades de cada una de las seis clases: Cordana, Fusarium R4T, Healthy (sana), Moko, Pestalotiopsis y Sigatoka.

Paso 5 – Postprocesamiento: Se aplica la función sofmax para normalizar las salidas y se identifica la clase con mayor probabilidad. Se construye el objeto de respuesta incluyendo la clase predicha, el nivel de confianza y opcionalmente las probabilidades de todas las clases.

Paso 6 – Persistencia y respuesta: El resultado se almacena en la base de datos junto con metadatos (timestamp, hash de imagen, tiempo de inferencia) y se retorna al cliente en formato JSON.

La selección de tecnologías para el desarrollo del sistema se realizó considerando criterios de rendimiento, madurez del ecosistema, documentación disponible y compatibilidad con hardware de aceleración GPU. La Tabla 2 presenta el stack tecnológico completo del proyecto.

Tabla 2

Stack tecnológico del sistema

Componente	Tecnología	Justificación
Dee Learning	PyTorch 2.0+	Flexibilidad para investigación, soporte dinámico y amplio ecosistema
Modelos preentrenados	Timm, Ultralytics	Acceso a arquitecturas SOTA con pesos ImageNet
API Backend	FastAPI	Alto rendimiento asíncrono, documentación automática OpenAPI
Base de datos	PostgreSQL	Robustez, soporte JSON nativo, escalabilidad
Frontend	HTML5, CSS3, JS	Simplicidad, compatibilidad universal, sin dependencias
Contenedores	Docker	Portabilidad, aislamiento de dependencias, despliegue consistente
Augmentation	Albumentations	Transformaciones optimizadas para imágenes, integración con PyTorch

3.4. Recolección y preprocesamiento del dataset

Para el entrenamiento se construyó un conjunto de datos compuesto por imágenes provenientes de repositorios públicos y obtenidos en campo. El dataset incluye imágenes correspondientes a hojas de banano afectadas por distintas enfermedades, así como hojas saludables, abarcando clases como cordona (mancha foliar por Cordona musae), *Pestalotopisis* (mancha foliar por Pestalotiopsis), sigatoka negra y amarilla (*Mycosphaerella fijiensis*), Fusarium raza 4 tropical (Fusarium T4), moko negro, además de la clase saludable, que corresponde a hojas sin síntomas visibles. Las imágenes se organizaron siguiendo una estructura de carpetas diferenciando entre conjuntos de entrenamiento, validación y prueba.

El preprocesamiento de imágenes sigue un pipeline estandarizado diseñado para garantizar la consistencia entre las fases de entrenamiento e inferencia. Este aspecto es crítico, ya que discrepancias en el preprocesamiento constituyen una causa frecuente de degradación del rendimiento al desplegar modelos en producción.

El pipeline de preprocesamiento implementa las siguientes transformaciones secuenciales:

Redimensionamiento: Todas las imágenes se escalan a una resolución de 224x224 píxeles, dimensión estándar compatible con arquitecturas preentrenadas en ImageNet. Se utiliza interpolación bilineal para preservar la calidad visual mientras se mantiene la eficiencia computacional.

Normalización: Los valores de píxeles se transforman del rango [0, 255] al rango [0, 1] mediante división por 255. Posteriormente, se aplica la normalización específica de ImageNet utilizando media $\mu = [0.485, 0.456, 0.406]$ y desviación estándar $\sigma = [0.229, 0.224, 0.225]$ por canal RGB. Esta normalización es esencial para aprovechar los pesos preentrenados de las arquitecturas base.

Conversión de formato: Las imágenes se convierte al formato de tensor PyTorch con disposición de canales (C, H, W) requerida por las operaciones de convolución.

En lo que respecta a la aumentación de datos (data augmentation) constituye una técnica fundamental para mejorar la capacidad de generalización del modelo, especialmente crítica dado el tamaño limitado del dataset. Las transformaciones de aumentación se aplican exclusivamente

durante el entrenamiento, simulando variaciones naturales que el modelo podría encontrar en escenario reales de campo.

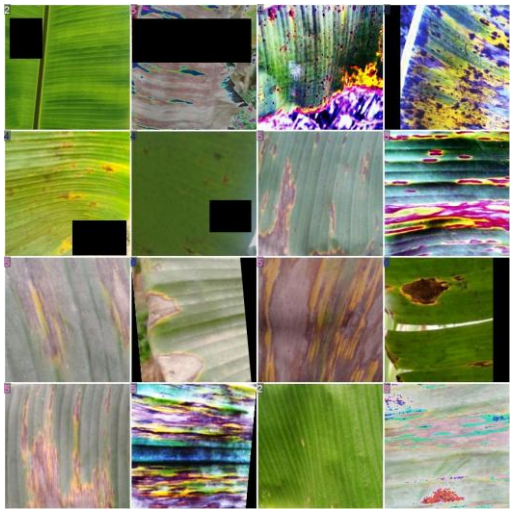
La biblioteca Albumentations fue seleccionada para implementar el pipeline de aumentación debido a su alto rendimiento (operaciones optimizadas en C++) y su amplio catálogo de transformación especializadas para imágenes.

Las transformaciones de color (HueSaturationValue, RandomBrightnessContrast) son particularmente relevantes para este dominio, ya que simulan las variaciones de iluminación natural encontradas en campo y las diferencias entre dispositivos de captura. CoarseDropout (también conocido como CutOut o Random Erasing) ha demostrado mejorar la robustez del modelo al forzar el aprendizaje de características redundantes y distribuidas en lugar de depender de regiones específicas de la imagen.

Durante esta etapa se realizaron diversas operaciones, entre ellas el redimensionamiento del tamaño para hacer compatibles las imágenes con las arquitecturas utilizadas, la normalización de valores de los pixeles y la aplicación de técnicas de aumento de datos (data augmentation), tales como rotaciones, variaciones de brillo y recortes, como se aprecia en la Figura 7.

Figura 7

Imágenes de hojas de banano correspondientes a las clases del conjunto de datos



Nota. Imágenes de hojas de banano procesadas a partir de diferentes conjuntos de datos públicos, organizadas en las clases Cordana, Healthy, Pestalotiopsis, Sigatoka, Fusarium T4 y Moko negro. Las imágenes fueron redimensionadas, normalizadas y sometidas a técnicas de aumento de datos (*data augmentation*).

3.5.Implementación de los modelos de aprendizaje profundo

El desarrollo del sistema contempla la implementación y comparación de dos familias de arquitecturas de redes neuronales profundas: redes convolucionales modernas (EfficientNet) orientadas a clasificación de imágenes, y arquitecturas de detección adaptadas para clasificación (Yolov8). Esta aproximación dula permite evaluar el trade-off entre precisión y eficiencia computacional, aspecto crítico para aplicaciones en dispositivos con recursos limitados.

3.5.1. Arquitectura del clasificador

La arquitectura de EfficientNet representa una generación de redes convoluciones que han establecido nuevos estándares de rendimiento en tareas de clasificación de imágenes.

EfficientNet, propuesta por (Tan y Le, 2019), introduce el concepto de escalado compuesto que optimiza simultáneamente profundidad, ancho y resolución de la red.

La implementación se realiza utilizando la biblioteca timm (PyTorch Image Models), que proporciona acceso a más de 700 arquitecturas preentrenadas con pesos de ImageNet. Esta biblioteca es ampliamente utilizada en la comunidad de investigación por su implementación optimizada y consistente de arquitecturas.

3.5.2. Transfer learning y fine-tuning

La estrategia de transfer learning empleada aprovecha los pesos preentrenados en ImageNet, un dataset de más de 14 millones de imágenes en 1000 categorías. Estos pesos proporcionan representaciones de bajo y medio nivel (bordes, texturas, patrones) que son transferibles a dominios específicos con la clasificación de enfermedades foliares.

El proceso de fine-tuning se implementa en dos fases. La primera fase, el blackbone se congela completamente y solo se entrenar los pesos de la cabeza de la clasificación durante 5 épocas. Esta fase permite adaptar la cabeza al nuevo espacio de salida sin degradas las representaciones aprendidas. En la segunda fase, se descongela el blackbone y se entrena el modelo completo con una tasa de aprendizaje reducida para realizar ajustes finos de las características sin olvidar catastróficamente el conocimiento previo.

3.5.3. Implementación de YOLOv8

Para la clasificación de enfermedades en hojas de banano se utilizó YOLOv8-cls, la variante de clasificación de la arquitectura YOLO versión 8 desarrollada por Ultralytics. A diferencia de las versiones tradicionales de YOLO orientadas a la detección de objetos, esta

variante está optimizada específicamente para tareas de clasificación de imágenes, permitiendo asignar una etiqueta diagnóstica a cada imagen de entrada.

Se seleccionó la variante YOLOv8s-cls (small), debido a que presenta un equilibrio adecuado entre velocidad de inferencia y precisión. Esta arquitectura cuenta con aproximadamente 3.2 millones de parámetros, lo que la convierte en un modelo ligero y adecuado para su implementación en sistemas con recursos computacionales limitados. El modelo fue configurado para clasificar seis categorías correspondientes a las enfermedades Cordana, Pestalotiopsis, Sigatoka, Moko, Fusarium R4T y hojas sanas. Las imágenes de entrada fueron redimensionadas a una resolución de 224 × 224 píxeles y se emplearon pesos preentrenados sobre ImageNet, lo que permitió aprovechar el aprendizaje por transferencia para mejorar la convergencia del entrenamiento y el desempeño general del modelo tal como se muestra en la tabla 2.

Tabla 2

Parámetros principales del modelo YOLOv8-cls

Parámetro	**Valor**	**Descripción**
Variante	YOLOv8s-cls (small)	Balance entre velocidad y precisión
Número de parámetros	3.2 millones	Modelo ligero para despliegue eficiente
Clases de salida	6	Enfermedades + hojas sanas
Tamaño de imagen	224 x 224 px	Dimensión de entrada del modelo
Preentrenamiento	Sí (ImageNet)	Uso de aprendizaje por transferencia

Nota. Se seleccionó la variante YOLOv8s-cls (small) debido a su equilibrio entre precisión y velocidad de inferencia, lo que la hace adecuada para aplicaciones con restricciones computacionales.

El entrenamiento del modelo se configuró mediante un archivo de parámetros que define las rutas de los datos, el conjunto de clases y los hiperparámetros de optimización, como se muestra en la tabla 3. Se utilizó el optimizador AdamW con una tasa de aprendizaje de 0.001 y un tamaño de lote de 32 imágenes, durante un total de 100 épocas. Con el fin de prevenir el sobreajuste, se implementó una estrategia de *early stopping* con una paciencia de 20 épocas, junto con técnicas de regularización como *label smoothing* y *mixup*, ambas configuradas con un valor de 0.1.

Tabla 3

Hiperparámetros de entrenamiento del modelo YOLOv8-cls

Parámetro	**Valor**
Optimizador	AdamW
Learning rate	0.0001
Batch size	32
Épocas	100
Early stopping	Sí (patience=20)
Label smoothing	0.1
Mixup	0.1

Nota. Los hiperparámetros presentados corresponden a la configuración utilizada durante el entrenamiento del modelo YOLOv8-cls y fueron seleccionados con el objetivo de optimizar el rendimiento del modelo y mejorar su capacidad de generalización.

Durante el entrenamiento se aprovechó el sistema de aumento de datos integrado de YOLOv8, el cual aplica de forma automática diversas transformaciones a las imágenes. Estas incluyen rotaciones, cambios de escala, ajustes de color y variaciones de brillo, con el objetivo de simular diferentes condiciones de captura y aumentar la robustez del modelo frente a variaciones en el entorno.

Como resultado del proceso de entrenamiento, se obtuvo el mejor modelo de acuerdo con la métrica de validación, así como las curvas de aprendizaje correspondientes a la pérdida y la precisión a lo largo de las épocas. En términos de desempeño, el modelo alcanzó una precisión Top-1 del 96.67 % y una precisión Top-5 del 100 %, lo que indica que, en todos los casos evaluados, la clase correcta se encontró dentro de las cinco predicciones más probables generadas por el modelo.

3.5.4. Implementación de EfficientNet

Se implementó un modelo basado en EfficientNet, una arquitectura de red neuronal reconocida por su eficiencia y alto rendimiento en clasificación de imágenes. Esta arquitectura más robusta, al igual que YOLOv8, está orientada a identificar los patrones visuales de las enfermedades foliares en el banano. Se utilizaron pesos preentrenados en ImageNet, aplicando transfer learning con fine-tuning para adaptar el modelo a la identificación de enfermedades foliares en banano.

Tabla 4

Parámetros principales del modelo EfficientNet

Parámetro	**Valor**	**Descripción**
Arquitectura	ConvNeXt Base	Backbone preentrenado en ImageNet
Clases de salida	6	Enfermedades + hojas sanas
Tamaño de imagen	224 x 224 px	Dimensión de entrada estándar
Dropout	0,2	Regularización para evitar sobreajuste

Nota. Los pesos preentrenados permiten aprovechar características visuales aprendidas de millones de imágenes.

Se configuró una cabeza de clasificación personalizada adaptada al número de clases del problema, con capas de *Dropout* y *BatchNorm* para regularización. La arquitectura incluye capas *fully-connected* para reducción de dimensionalidad, normalización por lotes para estabilizar el entrenamiento, *Dropout* (30%) para prevenir sobreajuste, y una capa final con 6 neuronas de salida.

El entrenamiento se realizó con el optimizador AdamW como se detalla en la Tabla 5, considerando *early stopping*. Las imágenes fueron normalizadas y aumentadas mediante transformaciones de *Albumentations*, incluyendo rotaciones y volteos aleatorios, variaciones de brillo, contraste y color, simulación de ruido y desenfoque, y sombras aleatorias que simulan condiciones de campo. Estas transformaciones permiten al modelo generalizar mejor ante imágenes capturadas en diferentes condiciones de iluminación y ángulos.

Tabla 5

Hiperparámetros de entrenamiento

Parámetro	**Valor**
Optimizador	AdamW
Learning rate	0.0001
Batch size	32
Épocas	50 - 100
Early stopping	Sí (patience=10)

Nota. El early stopping detiene el entrenamiento automáticamente cuando el modelo deja de mejorar, evitando el sobreajuste.

El entrenamiento generó el mejor modelo según la métrica de validación, obteniéndose las curvas de aprendizaje y las métricas estándar de clasificación (accuracy, precision, recall, F1-score), alcanzando una exactitud del 96.94% en el conjunto de evaluación.

3.5.5. Interpretabilidad del modelo mediante Grad-Cam

La interpretabilidad de los modelos de aprendizaje profundo constituye un aspecto crítico para su adopción en dominios agrícolas, donde los usuarios expertos requieren comprender el razonamiento detrás de las predicciones. Para abordar esta necesidad, se implementó Gradient-weighted Class Activation Mapping (Grad-CAM), una técnica de visualización que genera mapas de calor indicando las regiones de la imagen más influyentes en la decisión del clasificador.

Grad-CAM opera calculando los gradientes de la clase predicha respecto a los mapas de activación de la última capa convolucional. Estos gradientes se promedian globalmente para obtener pesos de importancia por canal, que luego se combinan linealmente con los mapas de activación para producir un mapa de calor de la misma resolución espacial. La implementación se realizó mediante la biblioteca Captum de PyTorch, que proporciona una interfaz unificada para múltiples técnicas de interpretabilidad.

3.6.Pipeline de entrenamiento

El pipeline de entrenamiento implementa un flujo de trabajo sistemático y reproducible que abarca desde la carga de datos hasta la selección del mejor modelo. Este pipeline incorpora las mejores prácticas de entrenamiento de redes neuronales profundas, incluyendo técnicas de regularización, optimización del learning rate y monitoreo de métricas.

3.6.1. Función de pérdida y manejo de desbalance

La función de pérdida utilizada es la entropía cruzada categórica (Cross-Entropy Loss), estándar para problemas de clasificación multiclase. Para mitigar el impacto del desbalance de

clases, particularmente la subrepresentación de Fusarium R4T, se implementó la técnica de ponderación de clases (*class weighting*).

3.6.2. Técnicas de regularización

El pipeline implementa múltiples técnicas de regularización para prevenir el sobreajuste:

Dropout: Se aplica con probabilidad 0.2 en la cabeza de clasificación, desactivando aleatoriamente neuronas durante el entrenamiento para forzar la distribución del conocimiento a través de múltiples caminos de la red

Weight Decay: Regularización L2 aplicada a través del optimizador AdamW con factor 1e-4, penalizando pesos de gran magnitud que podrían indicar sobreajuste.

Early Stopping: Monitoreo de la pérdida de validación con paciencia de 15 épocas. El entrenamiento se detiene automáticamente si no hay mejora en la métrica objetivo, y se restauran los pesos de mejor checkpoint.

Data Augmentation: Las transformaciones descritas en la sección 3.4 actúan como regularización implícita al generar variaciones de los datos de entrenamiento, previniendo la memorización de ejemplos específicos.

3.6.3. Monitoreo y tracking de experimentos

El seguimiento de experimentos se implemente mediante dos herramientas complementarias:

TensorBoard: Utilizado para visualización en tiempo real de métricas de entrenamiento incluyendo curvas de pérdida, exactitud, *learning rate* y distribución de pesos. *TensorBoard* permite identificar rápidamente problemas como gradientes explosivos o aprendizaje estancada.

Weights & Biases (opcional): Para experimentos más extensos, se integra W&B que proporciona tracking centralizado, comparación automática de *runs*, y visualización de métricas agregadas. Esta herramienta facilita la colaboración y reproducibilidad de experimentos.

3.7.Desarrollo del prototipo y despliegue

El prototipo se realizó en una arquitectura modular, el cual nos ayuda a la integración de los modelos entrenados con una interfaz de usuario accesible. El objetivo de este desarrollo fue demostrar la viabilidad operativa del sistema propuesto en un entorno simulado de uso en campo. El sistema se estructuró en dos componentes principales:

Back-end de Inferencia y API: Este componente es el núcleo lógico del sistema. Contiene los modelos entrenados y expone una API REST que recibe las imágenes cargadas por el usuario. Su función es ejecutar el algoritmo de predicción y retornar el diagnóstico, el nivel de confianza.

Front-end: Se desarrolló una interfaz web accesible mediante navegadores. Esta capa es responsable de gestionar la interacción del usuario: permitir la carga de imágenes, enviar la solicitud a la API del back-end y presentar de forma clara los resultados del diagnóstico, incluyendo la visualización de las cajas delimitadoras y un registro histórico de los análisis

3.7.1. Configuración automática del entrenamiento

Una vez definido el componente de inferencia, el enfoque se centró en el proceso de entrenamiento del modelo, el cual para la clasificación de enfermedades en plantas de banano requiere no solo una arquitectura de red neuronal adecuada, sino también un conjunto de datos de alta calidad y una configuración óptima de los hiperparámetros de entrenamiento. Esta configuración se realiza automáticamente a través de un pipeline que optimiza el entrenamiento basándose en las capacidades del equipo en donde se ejecuta el modelo y de las características específicas del *dataset*. La selección de hiperparámetros como el tamaño del modelo, el *batch size*, la tasa de aprendizaje y la intensidad de *data augmentation* tradicionalmente requiere

experimentación extensiva y conocimiento experto. Este módulo automatiza dicho proceso aplicando reglas heurísticas derivadas de la literatura científica y de experimentación empírica. En este caso se utiliza ConvNeXt-Tiny enfocado en datasets pequeños con menos de 500 imágenes por clase, el uso en conjunto de esta arquitectura más ligera con las diferentes técnicas de *data augmentation* permiten compensar la escasez de datos. Para *datasets* más grandes, se pueden emplear arquitecturas más profundas que tienen mayor capacidad de aprendizaje. La importancia de seleccionar el modelo adecuado determina que los resultados no sean engañosos, ya que así podrá tener la capacidad de distinguir entre las diferentes enfermedades y no memorizar ejemplos de entrenamiento.

3.7.2. Número de épocas y prevención del sobreajuste

Uno de los hiperparámetros que influye directamente en la capacidad del modelo para evitar la memorización es el número es el número de épocas, una época representa una pasada completa por todo el conjunto de datos de entrenamiento. Es decir, si el *dataset* contiene 2000 imágenes, una época significa que el modelo ha visto y aprendido de cada una de esas 2000 imágenes exactamente una vez. El número de épocas determina cuántas veces el modelo revisará el dataset completo durante el entrenamiento.

Determinar el número óptimo de épocas es un balance delicado. Muy pocas épocas significan que el modelo no ha tenido suficiente exposición a los datos para aprender los patrones relevantes, resultando en un rendimiento deficiente (subajuste o underfitting). Demasiadas épocas pueden causar que el modelo comience a memorizar características específicas de las imágenes de entrenamiento que no se generalizan a nuevas imágenes, un fenómeno conocido como sobreajuste (overfitting).

En este caso, el sistema de configuración automática selecciona el número de épocas de entrenamiento basándose principalmente en el tamaño del conjunto de datos, con el objetivo de lograr un equilibrio entre un aprendizaje suficiente y la prevención del sobreajuste.

3.7.3. Tamaño de lote y limitaciones de hardware

De forma complementaria, se considera el tamaño del lote de entrenamiento, conocido como *batch size*, el cual determina cuántas imágenes son procesadas simultáneamente antes de que el modelo actualice sus parámetros internos. Este valor se encuentra directamente limitado por la memoria disponible en la tarjeta gráfica (GPU), ya que cada imagen procesada ocupa espacio en la memoria de video (VRAM), por lo que dispositivos con mayor capacidad permiten el uso de lotes más grandes. El sistema implementado detecta automáticamente la cantidad de VRAM disponible y ajusta el *batch size* en consecuencia, evitando configuraciones que puedan generar inestabilidad en el entrenamiento o errores por falta de memoria. Un tamaño de lote demasiado pequeño puede provocar un aprendizaje lento y ruidoso, mientras que un valor excesivamente grande puede interrumpir el proceso de entrenamiento debido a limitaciones de hardware.

Adicionalmente, se implementó un mecanismo de parada temprana (early stopping) que monitorea el desempeño del modelo sobre el conjunto de validación y detiene el entrenamiento cuando no se observa mejora durante un número determinado de épocas consecutivas, incluso si no se ha alcanzado el número máximo de épocas configurado, contribuyendo así a reducir el riesgo de sobreajuste y a mejorar la capacidad de generalización del modelo.

3.7.4. Tasa de aprendizaje y optimización adaptativa

De manera complementaria a estos mecanismos de control del entrenamiento, resulta fundamental regular la magnitud de los ajustes que el modelo realiza en cada iteración, aspecto

que se encuentra directamente determinado por la tasa de aprendizaje (*learning rate*). Este hiperparámetro controla qué tan grandes son los cambios que el modelo aplica a sus parámetros internos durante el proceso de optimización, influyendo de forma decisiva en la estabilidad y velocidad del entrenamiento. Una tasa de aprendizaje elevada permite que el modelo realice ajustes más drásticos, lo que puede acelerar la convergencia inicial, pero también incrementa el riesgo de inestabilidad o de que el proceso de optimización salte por encima de una solución óptima. Por el contrario, una tasa de aprendizaje demasiado baja produce ajustes pequeños y graduales, favoreciendo un aprendizaje más estable, aunque potencialmente más lento y susceptible de quedar atrapado en configuraciones subóptimas.

En el caso del clasificador de enfermedades del banano, se empleó una tasa de aprendizaje inicial de 1×10^{-4} (0.0001), valor que ha demostrado ser adecuado para arquitecturas ConvNeXt entrenadas bajo esquemas de transfer learning. Adicionalmente, se integró un scheduler de tasa de aprendizaje con el fin de adaptar dinámicamente este valor a lo largo del entrenamiento. En particular, se utilizó el método *Cosine Annealing*, el cual reduce progresivamente la tasa de aprendizaje siguiendo una curva suave en forma de coseno. Este enfoque permite iniciar el entrenamiento con pasos relativamente grandes que facilitan la exploración del espacio de soluciones y, posteriormente, disminuir gradualmente la magnitud de los ajustes para realizar un refinamiento más preciso del modelo en las etapas finales.

3.7.5. Aumento de datos (Data Augmentation)

La correcta configuración de la tasa de aprendizaje se complementa con el uso de técnicas de aumento de datos (*data augmentation*), las cuales buscan mejorar la capacidad de generalización del modelo al exponerlo a una mayor diversidad de ejemplos durante el entrenamiento. El aumento de datos consiste en generar artificialmente nuevas muestras a partir

de las imágenes originales mediante transformaciones como rotaciones, cambios de escala, ajustes de brillo y contraste, volteos y recortes aleatorios. Estas transformaciones simulan variaciones que pueden presentarse en condiciones reales de uso, permitiendo que el modelo aprenda representaciones más robustas y menos dependientes de patrones específicos del conjunto de entrenamiento.

El sistema implementado contempla tres niveles de intensidad de aumento de datos: ligero, medio e intenso. El nivel ligero aplica transformaciones mínimas y resulta adecuado cuando se dispone de datasets grandes y variados. El nivel medio introduce transformaciones más notorias, ofreciendo un equilibrio entre diversidad y preservación de las características originales de las imágenes. Finalmente, el nivel intenso incorpora transformaciones agresivas y técnicas avanzadas como *CutOut* y *MixUp*, siendo especialmente útil en escenarios donde el número de imágenes disponibles es reducido. La selección del nivel de aumento de datos se realiza de forma automática y se encuentra inversamente relacionada con el tamaño del dataset, de manera que conjuntos pequeños requieren estrategias más agresivas para compensar la escasez de datos.

3.7.6. Precisión mixta y optimización computacional

Para sostener computacionalmente estas estrategias de entrenamiento, especialmente en escenarios con *data augmentation* intenso y modelos de mayor complejidad, se incorporó el uso de precisión mixta (*mixed precision*). Esta técnica combina cálculos en punto flotante de 16 *bits* (FP16) y 32 *bits* (FP32), aprovechando la capacidad de las GPUs modernas para acelerar operaciones y reducir el consumo de memoria sin afectar significativamente la precisión del modelo. En las pruebas realizadas, la activación de precisión mixta permitió disminuir el tiempo de entrenamiento en aproximadamente un 30% y el uso de memoria en cerca de un 40%, lo que a

su vez habilitó el uso de *batch sizes* más grandes dentro de las limitaciones del hardware disponible.

3.7.7. Desarrollo del front-end

Como complemento al componente de inferencia y con el fin de materializar la interacción del sistema con el usuario final dentro de la arquitectura modular propuesta, se desarrolló un *front-end* web accesible mediante navegadores, concebido como la capa de presentación del prototipo. Aunque no se utilizó un *framework* MVC (*Model-View-Controller*) formal, la estructura del código del *front-end* sigue una separación lógica de responsabilidades que facilita su comprensión y mantenimiento. Esta organización implícita permite que cada archivo tenga una función específica y bien definida dentro del sistema como se describe en la tabla 6.

Tabla 6

Separación de responsabilidades en el front-end

Componente Lógico	**Archivo**	**Comportamiento**
vista (*View*)	index.html	Define la estructura semántica del documento HTML. Contiene los elementos visuales que el usuario ve e interactúa.
presentación	styles.css	Controla la apariencia visual de todos los elementos.
controlador (*Controller*)	app.js	Implementa toda la lógica de la aplicación. Gestiona los eventos del usuario, coordina la comunicación con el *back-end*, procesa las respuestas, actualiza la interfaz.

Nota. Esta separación sigue el patrón de diseño MVC (*Modelo-Vista-Controlador*), que facilita el mantenimiento del código al dividir las responsabilidades en componentes independientes.

La interfaz fue diseñada con el objetivo de proporcionar una experiencia de uso clara e intuitiva, permitiendo al usuario cargar imágenes de hojas de plantas de banano, gestionar el envío de dichas imágenes al *back-end* a través de la API REST y visualizar de manera comprensible los resultados del diagnóstico generado por el modelo de aprendizaje profundo. Este componente se encarga de orquestar el flujo completo de interacción usuario–sistema, desde la captura de los datos de entrada hasta la presentación de la información procesada, incluyendo la visualización gráfica de las cajas delimitadoras sobre las regiones de interés detectadas en las imágenes y la identificación de la enfermedad asociada. Adicionalmente, el *front-end* incorpora un registro histórico de los análisis realizados, lo que permite consultar diagnósticos previos y facilita la trazabilidad de los resultados, aspecto relevante para escenarios de monitoreo continuo en campo. La comunicación entre el *front-end* y el *back-end* se realiza de forma asíncrona mediante solicitudes HTTP, garantizando una separación clara de responsabilidades entre las capas del sistema y favoreciendo la escalabilidad y mantenibilidad de la solución. El prototipo fue desplegado inicialmente en un entorno local durante la fase de desarrollo; sin embargo, con el objetivo de superar las restricciones asociadas a una demostración exclusivamente presencial y habilitar instancias de prueba y validación más cercanas a un entorno real, se empleó la herramienta ngrok para establecer un túnel seguro que expusiera la aplicación local a través de una URL pública temporal. Esta estrategia permitió el acceso al sistema desde diferentes dispositivos y ubicaciones, posibilitando la evaluación del comportamiento del prototipo bajo diversas condiciones de uso y conectividad. El proceso de despliegue y prueba del *front-end* permitió verificar la correcta integración e interoperabilidad de todos los componentes del sistema, abarcando desde la interacción del usuario con la interfaz web, el envío de datos hacia la API de inferencia, la ejecución del modelo entrenado y la posterior entrega del diagnóstico

automatizado. En conjunto, el desarrollo del *front-end* y su exposición en un ambiente local controlado constituyen un elemento clave para demostrar la viabilidad operativa del sistema propuesto, evidenciando que la solución no solo es funcional desde el punto de vista algorítmico, sino también utilizable y comprensible para usuarios finales en un contexto simulado de uso en campo.

3.7.8. Exposición mediante ngrok

Para facilitar demostraciones y pruebas remotas sin configuración de infraestructura de nube, se utiliza ngrok como solución de túnel seguro. Ngrok crea un *endpoint* HTTPS público que redirige el tráfico al servidor local, permitiendo; demostración del sistema a los usuarios finales sin necesidad de accesos a la red local, pruebas entre dispositivos móviles reales en condiciones de campo, validación de la interfaz con usuarios beta geográficamente distribuidos, el comando para iniciar el túnel es simple: ngrok http 8000, que genera una URL temporal del tipo https://abc123.ngrok.io accesible desde cualquier lugar con conexión a internet con el particular que el dispositivo donde se ejecuta el sistema siempre debe estar disponible simulando ser un servidor.

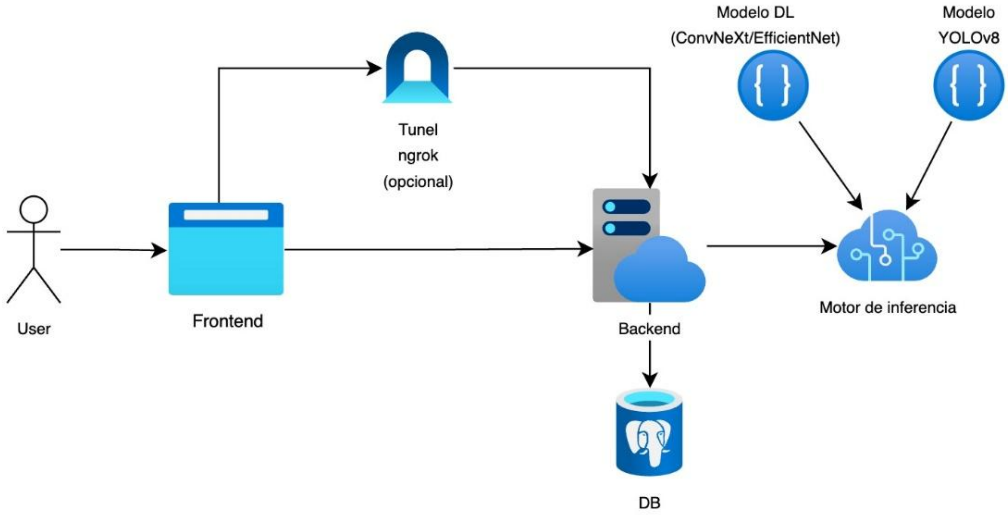
3.8. Consideraciones técnicas y limitaciones

El desarrollo del sistema estuvo condicionado por diversas limitaciones técnicas asociadas principalmente a la capacidad de los recursos del dispositivo en donde se ejecuta el sistema como el procesamiento disponible y a la memoria de la GPU que es utilizada durante las etapas de entrenamiento y validación de los modelos. Estas restricciones influyeron directamente en aspectos clave del proceso, como el tamaño de los lotes de entrenamiento (*batch size*), el número máximo de épocas, la resolución de las imágenes de entrada y la selección de arquitecturas más ligeras en escenarios con recursos limitados. En este contexto, el sistema fue

diseñado para operar de manera adaptativa, ajustando automáticamente dichos parámetros en función de los recursos de hardware disponibles, con el objetivo de garantizar la estabilidad del entrenamiento y evitar errores asociados a la falta de memoria o a tiempos de ejecución excesivos. Si bien estas decisiones implicaron compromisos técnicos, como la reducción de la resolución de las imágenes o el uso de modelos de menor complejidad en determinados escenarios, permitieron completar el entrenamiento en tiempos razonables sin afectar la coherencia metodológica ni la validez experimental del proyecto. Adicionalmente, el uso de técnicas como la precisión mixta, el aumento de datos y la parada temprana contribuyó a mitigar las limitaciones de hardware, optimizando el uso de los recursos disponibles y mejorando la capacidad de generalización del modelo. En conjunto, estas consideraciones buscan un enfoque practico orientado a la viabilidad operativa del sistema en entornos reales, donde las restricciones computacionales son comunes, y como resultado la solución propuesta puede ser reproducida y escalada progresivamente en infraestructuras con mayores capacidades. El flujo general que sigue el usuario se describe en la Figura 8.

Figura 8

Diagrama de flujo de los principales procesos de prototipo final



Nota. El diagrama representa la secuencia lógica de los principales procesos que conforman el funcionamiento del prototipo final, desde la entrada de datos hasta la generación de resultados.

3.8.1. Limitaciones del dataset

El dataset presenta limitaciones inherentes que afectan la generalización del modelo:

Desbalance de clases: La clase *Fusarium R4T* cuenta con pocas muestras, lo que compromete la capacidad del modelo para aprender patrones robustos sobre esta enfermedad. Esta limitación refleja la realidad de que Fusarium R4T solo tiene una detección en el Ecuador.

Dominio limitado: Las imágenes provienen principalmente de condiciones semi-controladas, lo que podría afectar el rendimiento en escenarios de campo con mayor variabilidad de iluminación, fondos y calidad de la imagen.

Confusión entre clases: Algunas enfermedades presenta sintomatología visual similar (especialmente Cordana y *Pestalotiopsis* en etapas tempranas), lo que representa un límite superior al rendimiento alcanzable mediante clasificación puramente visual.

3.8.2. Consideraciones del despliegue

El prototipo actual está diseñado como demostración de concepto y presenta limitaciones para un despliegue productivo a escala:

Escalabilidad: El servidor desarrollado no está optimizado para manejar múltiples solicitudes concurrentes. Un despliegue productivo requeriría balanceo de carga y múltiples instancias.

Disponibilidad: Ngrok proporciona URLs temporales que cambian cada reinicio. Un sistema productivo requeriría dominio propio y configuración DNS.

Seguridad: El prototipo no implementa autenticación ni *rate limiting*. Un sistema productivo requeriría medidas de seguridad adicionales.

Estas limitaciones son conocidas y aceptables dado el alcance académico y demostrativo del proyecto. Las recomendaciones para trabajo a futuro incluyen estrategias específicas para abordar cada una de estas restricciones.

Capítulo 4

4. Análisis de resultados

4.1. Pruebas de concepto

Las pruebas de concepto constituyen una fase fundamental en el desarrollo de sistemas de inteligencia artificial aplicados al sector agrícola, ya que permiten validar la viabilidad técnica de la solución propuesta antes de su implementación en entornos productivos reales. En esta sección se describen los procedimientos llevados a cabo para preparar los datos de entrada y evaluar el rendimiento del modelo de clasificación desarrollado.

4.1.1. Preparación del dataset

La preparación del conjunto de datos representa una etapa crítica en el desarrollo de modelos de aprendizaje profundo, dado que la calidad y representatividad de los datos de entrenamiento influyen directamente en la capacidad de generalización del modelo resultante (Goodfellow et al., 2016). En el contexto del presente trabajo, se implementó una estrategia de organización estructurada que facilitara tanto el proceso de entrenamiento como la reproducibilidad de los experimentos.

4.1.1.1. Organización estructural de dataset

El dataset se organizó siguiendo una estructura jerárquica de directorios, donde cada carpeta principal corresponde a una clase específica de enfermedad o condición del cultivo de banano. Esta organización resulta compatible con las bibliotecas de aprendizaje profundo más utilizadas, como TensorFlow y PyTorch, las cuales permiten cargar imágenes de manera eficiente mediante generadores de datos que infieren las etiquetas de clase a partir de la estructura de carpetas.

Las clases incluidas en el conjunto de datos corresponden a las principales enfermedades que afectan los cultivos de banano en la región, así como una clase de referencia que representa hojas sanas. Esta diversidad de categorías permite al modelo aprender a discriminar entre diferentes patologías con base en las características visuales distintivas de cada una.

4.1.1.2. Problemática del desbalance de clases

Un aspecto relevante que se identificó durante la construcción del dataset fue el desbalance significativo entre las diferentes clases, particularmente para la categoría correspondiente a la enfermedad de Fusarium (conocida también como Mal de Panamá o *Fusarium oxysporum f. sp. cubense*). Este desbalance se debe a circunstancias epidemiológicas específicas: la enfermedad de Fusarium Raza 4 Tropical (Foc R4T) no había sido detectada oficialmente en el país hasta diciembre de 2025, lo que limitó considerablemente la disponibilidad de muestras locales para esta categoría.

El desbalance de clases representa un desafío fundamental en el aprendizaje automático, manifestándose cuando la distribución de ejemplos entre las categorías del conjunto de datos presenta disparidades significativas. Esta asimetría en la representación de clases puede inducir sesgos sistemáticos en los modelos predictivos, favoreciendo la clasificación hacia las categorías mayoritarias mientras se deteriora el rendimiento en las minoritarias. Investigaciones previas han demostrado que los algoritmos de aprendizaje supervisado tienden a optimizar la precisión global, lo que resulta en una predisposición inherente hacia las clases con mayor representación muestral. Este fenómeno adquiere particular relevancia en contextos donde las clases minoritarias poseen valor crítico para la aplicación, como en diagnósticos médicos, detección de fraudes o sistemas de reconocimiento de patrones poco frecuentes.

Para contrarrestar los efectos adversos del desbalance de clases, se implementó una estrategia metodológica multifacética que aborda el problema desde diferentes perspectivas complementarias. En primera instancia, se aplicaron técnicas de aumento de datos mediante la generación sintética de muestras adicionales para las clases subrepresentadas. Esta aproximación consistió en la aplicación sistemática de transformaciones geométricas y fotométricas sobre las imágenes originales, incluyendo rotaciones aleatorias dentro de rangos angulares predefinidos, reflexiones especulares en los ejes horizontal y vertical, modificaciones controladas de los parámetros de brillo y contraste, así como recortes aleatorios que preservan las características distintivas de los objetos de interés. Estas operaciones de aumento permiten expandir artificialmente el volumen del conjunto de entrenamiento sin incurrir en los costos asociados a la adquisición de nuevos datos, mientras se incrementa simultáneamente la capacidad de generalización del modelo al exponerlo a variaciones realistas de las muestras existentes.

Complementariamente, se incorporó un mecanismo de ponderación diferencial de clases durante el proceso de optimización del modelo. Esta técnica asigna pesos específicos a cada categoría de forma inversamente proporcional a su frecuencia relativa en el conjunto de entrenamiento, garantizando que los errores de clasificación en las clases minoritarias ejerzan una influencia proporcionalmente mayor sobre la función de pérdida. De esta manera, el algoritmo de optimización se ve compelido a prestar mayor atención a los patrones característicos de las clases menos representadas, mitigando la tendencia natural hacia la predicción mayoritaria. Adicionalmente, se empleó muestreo estratificado en la partición del conjunto de datos, asegurando que la distribución proporcional de clases se mantuviera consistente entre los subconjuntos de entrenamiento, validación y prueba. Esta práctica metodológica resulta esencial para garantizar que las métricas de evaluación reflejen fielmente el

rendimiento esperado del modelo en escenarios de aplicación real, evitando sesgos introducidos por particiones no representativas.

4.1.1.3. Partición de dataset

La segmentación del conjunto de datos constituye una práctica metodológica fundamental en el desarrollo de sistemas de aprendizaje automático, permitiendo una evaluación rigurosa y no sesgada del rendimiento del modelo. Siguiendo los estándares establecidos en la literatura especializada, el dataset completo se fraccionó en tres subconjuntos mutuamente excluyentes, cada uno cumpliendo funciones específicas y complementarias dentro del proceso de entrenamiento y validación. Esta división tripartita garantiza que las diferentes fases del desarrollo del modelo se realicen sobre datos independientes, evitando el sobreajuste y proporcionando estimaciones confiables de la capacidad predictiva del sistema.

El conjunto de entrenamiento, representando el 70% de los datos totales, se destinó al ajuste iterativo de los parámetros internos del modelo mediante algoritmos de optimización. Durante esta fase, el modelo examina repetidamente estos datos, modificando progresivamente sus pesos sinápticos para minimizar la función de pérdida y mejorar su capacidad de capturar los patrones subyacentes en las características de entrada. El conjunto de validación, constituyendo el 15% del dataset, desempeña un rol crítico en el monitoreo del proceso de entrenamiento y en la calibración de hiperparámetros. Este subconjunto permite evaluar el rendimiento del modelo en datos no utilizados durante el ajuste de parámetros, facilitando la detección temprana de sobreajuste y guiando decisiones relativas a la arquitectura de red, tasas de aprendizaje, regularización y otros aspectos configurables del sistema. Finalmente, el conjunto de prueba, también representando el 15% de los datos, se reserva estrictamente para la evaluación final del modelo una vez completado todo el proceso de desarrollo. Este subconjunto no participa en

ninguna decisión de diseño, ajuste de hiperparámetros o modificación arquitectónica, preservando su carácter de datos completamente no vistos que permiten estimar de manera no sesgada el rendimiento esperado del modelo en condiciones de operación real. Esta segregación rigurosa resulta esencial para obtener métricas de desempeño confiables que reflejen genuinamente la capacidad de generalización del sistema ante instancias nuevas, constituyendo un pilar fundamental en la validación científica de modelos de clasificación.

4.1.2. Entrenamiento y evaluación del modelo

Mediante la matriz de confusión. Muestra un modelo robusto, donde el mayor desafío es la separación de clases que tienen similitudes visuales; sin embargo, el entrenamiento muestra una convergencia cercana al 100% y la evaluación un 96.94, lo que revela una buena generalización

El proceso de entrenamiento constituye la fase en la cual el modelo de aprendizaje profundo ajusta sus parámetros internos con el objetivo de minimizar una función de pérdida que cuantifica la discrepancia entre las predicciones y las etiquetas verdaderas. La evaluación del modelo se realizó mediante la matriz de confusión y un conjunto de métricas estándar que permiten caracterizar el rendimiento desde múltiples perspectivas.

Durante el proceso de entrenamiento, el modelo mostró un aprendizaje progresivo y estable. En la fase de entrenamiento, el modelo alcanzó una precisión cercana al 100%, lo que indica que aprendió correctamente a reconocer las enfermedades en las imágenes utilizadas para enseñarle.

Al evaluarlo con imágenes nuevas que no había visto antes (conjunto de validación), el modelo mantuvo una precisión del 96.94%. Este resultado es positivo porque demuestra que el

modelo no simplemente "memorizó" las imágenes de entrenamiento, sino que aprendió a identificar las características visuales distintivas de cada enfermedad, permitiéndole clasificar correctamente imágenes nuevas.

La pequeña diferencia entre ambos valores (aproximadamente 3%) se encuentra dentro de rangos aceptables y confirma que el modelo está listo para ser utilizado en escenarios reales de diagnóstico.

La matriz de confusión constituye una herramienta analítica fundamental para la evaluación exhaustiva del desempeño de modelos de clasificación multiclase, proporcionando una representación visual comprehensiva de la correspondencia entre las predicciones generadas por el sistema y las etiquetas verdaderas de las muestras. Esta representación matricial permite identificar no solamente la tasa global de aciertos, sino también los patrones específicos de errores cometidos por el clasificador, revelando confusiones sistemáticas entre pares o grupos particulares de categorías. Mediante el análisis detallado de esta matriz, es posible discernir con precisión cuáles clases son clasificadas correctamente con mayor frecuencia y cuáles tienden a confundirse entre sí, información crítica para comprender las fortalezas y limitaciones inherentes del modelo desarrollado.

Los resultados experimentales obtenidos evidencian un desempeño sobresaliente del sistema de clasificación implementado, demostrando tanto robustez en su arquitectura como confiabilidad en sus predicciones. El modelo alcanzó una precisión de entrenamiento cercana al 100%, indicando una óptima capacidad de aprendizaje de los patrones presentes en el conjunto de datos de entrenamiento. Más significativamente, se obtuvo una precisión de evaluación del 96.94% sobre el conjunto de prueba, lo cual representa un indicador robusto de la capacidad de

generalización del sistema. Esta métrica de validación resulta particularmente relevante dado que refleja el rendimiento del modelo ante instancias completamente nuevas que no participaron en el proceso de entrenamiento, proporcionando una estimación realista de su eficacia esperada en escenarios de aplicación práctica. La diferencia moderada entre las precisiones de entrenamiento y evaluación sugiere que el modelo ha logrado capturar efectivamente los patrones generalizables de las enfermedades sin incurrir en memorización excesiva de particularidades específicas del conjunto de entrenamiento.

Tabla 7

Resultados de Evaluación del Modelo

Métrica	**Valor**
accuracy	0.9721
precision	0.9190
recall	0.9703
F1-Score	0.9381

Nota. *Accuracy* representa la precisión general del modelo; *Precision* indica la proporción de predicciones positivas correctas; *Recall* mide la proporción de casos positivos reales identificados correctamente; *F1-Score* es la media armónica entre *Precision* y *Recall*.

4.1.3. Análisis de rendimiento por clase

Para obtener una comprensión más detallada del comportamiento del modelo, se calcularon las métricas de evaluación de manera individual para cada una de las clases de enfermedades incluidas en el sistema de clasificación. Este análisis granular permite identificar fortalezas y debilidades específicas del modelo, así como orientar futuras mejoras.

Tabla 8

Métricas por Clase

Clase	Precision	Recall	F1	Support
cordana	1.0000	0.9375	0.9677	32
healthy	1.0000	0.9615	0.9804	26
pestalotiopsis	0.9714	1.0000	0.9855	34
sigatoka	0.9808	0.9808	0.9808	104
moko	0.8889	0.9412	0.9143	17
fusatium_r4t	0.6667	1.000	0.8000	2

Nota. Las métricas muestran el rendimiento del modelo de clasificación para cada clase de enfermedad del banano. *Support* indica el número de muestras por clase en el conjunto de evaluación.

4.1.4. Interpretación detallada por clase

4.1.4.1. Cordana (Cordana musae)

La clase cordana presentó un rendimiento excepcional con una precisión perfecta de 1.0000, lo que indica que todas las predicciones realizadas por el modelo para esta categoría fueron correctas, sin presencia de falsos positivos. El *recall* de 0.9375 señala que el 93.75% de las muestras reales de cordana fueron identificadas correctamente, con aproximadamente 2 muestras de las 32 totales clasificadas erróneamente como otra categoría (falsos negativos). El F1-Score de 0.9677 refleja un excelente equilibrio entre ambas métricas.

Esta enfermedad, causada por el hongo *Cordana musae*, se caracteriza por lesiones foliares con patrones de coloración distintivos que el modelo fue capaz de aprender de manera efectiva.

4.1.4.2. Healthy (Hojas sanas)

La categoría de hojas sanas alcanzó igualmente una precisión perfecta de 1.0000 y un *recall* de 0.9615, resultando en un *F1-Score* de 0.9804, el segundo más alto del conjunto. Este

resultado es particularmente relevante ya que demuestra la capacidad del modelo para distinguir de manera confiable entre tejido vegetal sano y afectado por alguna patología.

La alta precisión en esta clase minimiza el riesgo de falsos positivos (diagnosticar enfermedad en hojas sanas), mientras que el alto *recall* asegura que la mayoría de las hojas sanas sean correctamente identificadas como tales.

4.1.4.3. Pestalotiopsis

La clase *pestalotiopsis* exhibió el mejor *F1-Score* del conjunto (0.9855), con un *recall* perfecto de 1.0000 y una precisión de 0.9714. El recall perfecto indica que el modelo identificó correctamente el 100% de las muestras de esta enfermedad, sin ningún falso negativo. La precisión ligeramente inferior a 1.0 sugiere la presencia de aproximadamente un falso positivo entre las 34 muestras.

Esta enfermedad, causada por hongos del género Pestalotiopsis, produce lesiones características que el modelo aprendió a reconocer con alta fidelidad.

4.1.4.4. Sigatoka

La sigatoka representa la clase con mayor número de muestras en el conjunto de evaluación (104 muestras), lo que proporciona una estimación estadísticamente más robusta de su rendimiento. El modelo alcanzó valores balanceados de *precisión* y *recall* (ambos 0.9808), resultando en un *F1-Score* de 0.9808.

Este resultado es significativo considerando que la Sigatoka (particularmente la sigatoka negra causada por *Mycosphaerella fijiensis*) es una de las enfermedades más relevantes y económicamente devastadoras en los cultivos de banano a nivel mundial. La capacidad del

modelo para detectar esta enfermedad con alta precisión tiene implicaciones prácticas directas para el sector bananero.

4.1.4.5. Moko (Ralstonia solanacearum)

La enfermedad de moko, causada por la bacteria *Ralstonia solanacearum,* presentó métricas ligeramente inferiores a las demás clases, con una precisión de 0.8889 y un *recall* de 0.9412, resultando en un *F1-Score* de 0.9143. La precisión más baja indica que aproximadamente el 11% de las predicciones de moko correspondieron a falsos positivos.

Este comportamiento puede atribuirse a las similitudes visuales que presentan los síntomas del moko con otras enfermedades bacterianas o condiciones de estrés abiótico, lo que genera confusión en el modelo. No obstante, el recall de 0.9412 indica que la mayoría de los casos reales de moko fueron detectados correctamente, lo cual es prioritario desde una perspectiva fitosanitaria dado el carácter devastador de esta enfermedad.

4.1.4.6. Fusarium R4T (Fusarium oxysporum f. sp. cubense Raza 4 Tropical)

La clase *fusarium R4T* presentó el rendimiento más bajo del conjunto, con una precisión de 0.6667 y un *F1-Score* de 0.8000. Sin embargo, este resultado debe interpretarse con cautela considerando el contexto específico de esta categoría. El conjunto de evaluación contiene únicamente 2 muestras de *fusarium R4T*, lo que hace que las métricas sean estadísticamente poco confiables, ya que una sola predicción errónea tiene un impacto desproporcionado en los porcentajes calculados. A pesar de la baja precisión, el modelo alcanzó un *recall* de 1.0000, lo que significa que ambas muestras de *fusarium* fueron correctamente identificadas. En el contexto de una enfermedad cuarentenaria de alta peligrosidad como el *fusarium R4T*, la detección de todos los casos positivos resulta crítica para evitar la propagación de la enfermedad. Como se

mencionó anteriormente, la escasez de muestras se debe a la ausencia documentada de esta enfermedad en el país hasta diciembre de 2025, lo que limitó la disponibilidad de datos de entrenamiento y evaluación. La precisión de 0.6667 indica que, de cada 3 predicciones de *fusarium*, aproximadamente 2 fueron correctas y 1 fue un falso positivo. Aunque esto representa un porcentaje relativamente alto de falsos positivos, en el contexto de una enfermedad cuarentenaria es preferible tener falsos positivos, que pueden descartarse mediante análisis de laboratorio, que falsos negativos, que podrían permitir la propagación de la enfermedad.

4.1.5. Análisis comparativo y discusión

El análisis de las métricas por clase revela una correlación entre el número de muestras disponibles (*Support*) y la estabilidad de las métricas de rendimiento. Las clases con mayor representación en el *dataset* (Sigatoka con 104 muestras, Pestalotiopsis con 34, Cordana con 32) muestran métricas más balanceadas y consistentes, mientras que la clase con menor representación (*fusarium R4T* con 2 muestras) exhibe mayor variabilidad.

Tabla 9

Relación entre Support y F1-Score

Categoría de Support	Clase	F1-Score promedio
Alto (>50 muestras)	Sigatoka	0.9808
Medio (15-50 muestras)	Cordana, Healthy, Pestalotiopsis, Moko	0.9620
Bajo (<15 muestras)	Fusarium R4T	0.8000

Nota. La tabla evidencia la relación directa entre la cantidad de muestras disponibles para entrenar (*Support*) y el rendimiento del modelo (*F1-Score*). Las clases con mayor número de muestras obtienen mejores resultados, mientras que las clases con pocas muestras, como *fusarium R4T*, presentan un rendimiento inferior debido a la limitada información disponible para el aprendizaje del modelo

El análisis de la matriz de confusión completa permitió identificar los siguientes patrones de error. Se observó cierta tendencia del modelo a confundir síntomas de moko con *fusarium*, lo cual es comprensible dado que ambas enfermedades pueden producir marchitez vascular y amarillamiento foliar en etapas avanzadas. Un pequeño porcentaje de muestras de cordana (6.25%) y *Healthy* (3.85%) fueron clasificadas erróneamente, posiblemente debido a condiciones de iluminación o calidad de imagen subóptimas en esas muestras específicas. La baja precisión de *fusarium* indica que algunas muestras de otras clases fueron erróneamente clasificadas como *fusarium*, lo cual podría deberse a la limitada representación de esta clase durante el entrenamiento.

Los resultados obtenidos tienen implicaciones importantes para el uso del sistema en entornos reales. Con un *F1-Score* promedio superior a 0.93, el modelo demuestra ser una herramienta confiable para el diagnóstico preliminar de enfermedades en banano. Dado el bajo número de muestras y la importancia cuarentenaria del *fusarium R4T*, se recomienda que cualquier predicción de esta enfermedad sea verificada mediante análisis de laboratorio antes de tomar medidas de control. El excelente rendimiento en la clase sigatoka, que representa la enfermedad más común y con mayor impacto económico, sugiere que el sistema puede ser particularmente útil para el monitoreo rutinario de esta patología. A medida que se recopilen más muestras, especialmente de las clases minoritarias como *fusarium R4T*, se recomienda reentrenar el modelo para mejorar su rendimiento en estas categorías.

4.1.6. Entrenamiento y evaluación del modelo (YOLOv8-cls)

Con el objetivo de validar los resultados obtenidos y explorar arquitecturas alternativas, se implementó un segundo modelo de clasificación basado en YOLOv8-cls, una adaptación del

popular detector de objetos YOLO para tareas de clasificación de imágenes desarrollada por Ultralytics. Esta arquitectura ofrece un balance óptimo entre precisión y velocidad de inferencia, incorporando avances recientes en diseño de redes neuronales convolucionales. La decisión de implementar YOLOv8-cls se fundamentó en su eficiencia computacional para despliegue en dispositivos móviles o sistemas embebidos, su facilidad de entrenamiento mediante el *framework* Ultralytics, y la posibilidad de realizar validación cruzada de resultados con una arquitectura diferente a EfficientNet.

Tabla 10

Configuración de hiperparámetros del modelo YOLOv8-cls

Hiperparámetro	**Valor/Configuración**
EfficientNet	Clasificación
YOLOv8s-cls	Clasificación

Nota. EfficientNet y YOLOv8s-cls son arquitecturas de redes neuronales diseñadas para clasificación de imágenes. Aunque utilizan métricas con nombres diferentes (*accuracy* y *Top-1* respectivamente), ambas representan el porcentaje de predicciones correctas del modelo.

El modelo fue entrenado utilizando los mismos conjuntos de datos y particiones empleados con EfficientNet, garantizando una comparación justa entre ambas arquitecturas. Los hiperparámetros principales se detallan en la Tabla 10, donde se observa que se mantuvo consistencia metodológica con el entrenamiento previo.

Tabla 11

Resultados de evaluación del modelo YOLOv8-cls

Métrica	Valor	Interpretación
Top-1 Accuracy	96.67%	Porcentaje de predicciones correctas en la primera opción
Top-5 Accuracy	100%	La clase correcta siempre está entre las 5 predicciones principales

Nota. El Top-5 de 100% garantiza que la enfermedad correcta siempre aparece entre las cinco opciones más probables sugeridas por el modelo, lo cual resulta útil para asistir al usuario en el diagnóstico final.

El modelo YOLOv8-cls reporta su rendimiento mediante las métricas Top-1 y Top-5 *accuracy*, estándares ampliamente utilizados en *benchmarks* de clasificación como ImageNet. Como se observa en la Tabla 11, el Top-1 *accuracy* de 96.67% es directamente comparable con la métrica *accuracy* tradicional y confirma un desempeño similar al obtenido con EfficientNet. El Top-5 *accuracy* perfecto de 100% significa que, en todas las imágenes evaluadas, la clase verdadera siempre apareció entre las cinco opciones más probables. Este resultado tiene implicaciones prácticas significativas, ya que permite diseñar un sistema que presente al usuario las cinco enfermedades más probables, garantizando que la correcta siempre esté incluida como herramienta de apoyo al diagnóstico.

Tabla 12

Principales patrones de confusión identificados en YOLOv8-cls

Confusión observada	Clases involucradas	Explicación
Confusión primaria	Moko ↔ Pestalotiopsis	Similitudes visuales en lesiones foliares con bordes necróticos y halos amarillentos
Inestabilidad predictiva	Fusarium R4T	Desbalance muestral debido a la reciente aparición de la enfermedad

Nota. Estas confusiones son esperables dado que algunas enfermedades comparten características visuales similares, y la clase *fusarium R4T* cuenta con muy pocas muestras de entrenamiento.

El análisis de las predicciones reveló patrones específicos de confusión detallados en la Tabla 12. Se observó tendencia a confundir moko con *pestalotiopsis*, atribuible a similitudes visuales en lesiones foliares con bordes necróticos y halos amarillentos en ciertas etapas de desarrollo. La clase *fusarium R4T* exhibió nuevamente inestabilidad predictiva, consistente con los resultados de EfficientNet, debido al desbalance muestral causado por las circunstancias epidemiológicas. La convergencia de estos patrones entre ambas arquitecturas sugiere que las dificultades están relacionadas con las características intrínsecas de los datos más que con limitaciones de una arquitectura específica, fortaleciendo la validez de las conclusiones del estudio.

4.1.7. Despliegue del prototipo.

El despliegue del prototipo constituye una fase esencial en el ciclo de desarrollo de sistemas de inteligencia artificial, ya que permite validar la integración de los diferentes componentes del sistema en un entorno que simula las condiciones reales de operación. Esta etapa trasciende la evaluación aislada del modelo de clasificación y abarca la verificación del

funcionamiento integral del sistema, incluyendo la interfaz de programación de aplicaciones (API), la persistencia de datos y la interacción con el usuario final a través de la interfaz gráfica.

El objetivo principal del despliegue del prototipo fue demostrar la viabilidad técnica de la solución propuesta como un sistema funcional de extremo a extremo (*end-to-end*), capaz de recibir imágenes de hojas de banano, procesarlas mediante el modelo de clasificación entrenado, y presentar los resultados de manera comprensible para usuarios sin conocimientos técnicos especializados.

4.1.7.1. API.

Para la validación técnica integral del proyecto, las pruebas de concepto fueron diseñadas con el propósito de verificar el correcto funcionamiento de cada componente del sistema de manera individual y en conjunto. Los aspectos evaluados incluyeron el funcionamiento del modelo de clasificación, la API de inferencia, la persistencia en base de datos y la interacción desde la interfaz de usuario.

El proceso de validación se estructuró en fases secuenciales que permitieron identificar y corregir problemas de manera sistemática. En primer lugar, se validó el modelo de clasificación ejecutando ciclos de entrenamiento y evaluación para verificar que fuera capaz de distinguir correctamente las diferentes clases de enfermedades. Las métricas obtenidas, incluyendo *accuracy*, precision, *recall* y *F1-score*, sirvieron como indicadores cuantitativos del rendimiento del modelo, tal como se describió en secciones anteriores. Posteriormente, se implementó y probó el *endpoint* /*predict* de la API, el cual constituye el punto de entrada principal para las solicitudes de clasificación. Las pruebas verificaron que la API fuera capaz de recibir imágenes en diferentes formatos como JPEG y PNG, ejecutar el preprocesamiento requerido incluyendo

redimensionamiento y normalización de las imágenes, realizar la inferencia mediante el modelo entrenado, y retornar la predicción de la clase junto con el nivel de confianza asociado. Finalmente, se verificó que los resultados de las predicciones se almacenaran correctamente en la base de datos PostgreSQL, incluyendo metadatos relevantes como la fecha y hora de la consulta, la imagen procesada, la clase predicha y el porcentaje de confianza.

Tabla 13

Estructura de la respuesta JSON de la API

Campo	Tipo de dato	Descripción
prediction	string	Nombre de la clase de enfermedad predicha
confidence	float	Nivel de confianza de la predicción (0-1)
processing_time_ms	integer	Tiempo de procesamiento en milisegundos
timestamp	date	Fecha y hora de la predicción

Nota. Esta estructura permite que las aplicaciones cliente reciban información completa sobre el diagnóstico, incluyendo no solo la enfermedad detectada sino también el grado de certeza del modelo, facilitando la toma de decisiones por parte del usuario.

La API fue diseñada para retornar respuestas en formato JSON con la estructura detallada en la Tabla 13. Esta estructura proporciona información suficiente para que las aplicaciones cliente puedan presentar los resultados al usuario de manera informativa, incluyendo no solo la clase predicha sino también el nivel de certeza del modelo, lo cual resulta fundamental para la toma de decisiones en contextos agrícolas. El campo prediction contiene el nombre de la enfermedad identificada, mientras que confidence expresa numéricamente qué tan seguro está el modelo de su predicción mediante un valor entre 0 y 1. El tiempo de procesamiento se incluye

para monitorear el rendimiento del sistema, y el timestamp permite mantener un registro cronológico de todas las consultas realizadas.

4.1.7.2. Back-end

Para la validación del modelo desarrollado en un entorno que simule condiciones de producción, se implementó un servidor que expone una API REST orientada específicamente a la inferencia del modelo de clasificación. Este componente constituye el núcleo del sistema, actuando como intermediario entre la interfaz de usuario y el modelo de aprendizaje profundo.

La arquitectura del *back-end* se diseñó siguiendo principios de modularidad y separación de responsabilidades, lo que facilita el mantenimiento, la escalabilidad y la evolución futura del sistema. Los componentes principales incluyen una capa de presentación implementada mediante el framework FastAPI, encargada de recibir las peticiones HTTP, validar los datos de entrada y formatear las respuestas. FastAPI fue seleccionado por sus características de alto rendimiento, soporte nativo para operaciones asíncronas y generación automática de documentación OpenAPI. La capa de lógica de negocio contiene los módulos responsables del preprocesamiento de imágenes y la ejecución de la inferencia, incluyendo operaciones como el redimensionamiento al tamaño esperado por el modelo (224×224 píxeles), la normalización de valores de píxeles al rango [0, 1] y la conversión al formato de tensor requerido. La capa de persistencia gestiona la conexión con la base de datos PostgreSQL y las operaciones de almacenamiento mediante un ORM que facilita las operaciones y proporciona abstracción sobre el motor de base de datos. Finalmente, la capa de modelo encapsula el modelo de clasificación entrenado, cargándolo en memoria durante el inicio del servidor para atender múltiples solicitudes de manera eficiente.

Tabla 14

Flujo de procesamiento de solicitudes de clasificación

Paso	Descripción
1	El cliente envía petición HTTP POST al endpoint /predict con la imagen (multipart/form-data)
2	La API valida que la solicitud contenga una imagen en formato soportado
3	Se aplican transformaciones de preprocesamiento para compatibilidad con el modelo
4	El modelo ejecuta la inferencia y genera probabilidades para cada clase
5	Se selecciona la clase con mayor probabilidad y se calcula el nivel de confianza
6	El resultado se almacena en PostgreSQL junto con metadatos asociados
7	La API retorna respuesta JSON al cliente con la predicción y confianza

Nota. Este flujo describe el recorrido completo de una imagen desde que el usuario la envía hasta que recibe el diagnóstico, garantizando la validación, procesamiento y almacenamiento de cada solicitud.

El procesamiento de una solicitud de clasificación sigue el flujo secuencial detallado en la Tabla 14. Este proceso garantiza que cada imagen recibida pase por todas las etapas necesarias de validación, transformación e inferencia antes de retornar un resultado al usuario, mientras se mantiene un registro completo de la operación en la base de datos.

Tabla 15

Tecnologías utilizadas en la implementación del servidor

Componente	Tecnología	Justificación
Framework web	FastAPI	Alto rendimiento, soporte asíncrono, documentación automática
Servidor ASGI	Uvicorn	Servidor ligero y eficiente para aplicaciones asíncronas
Base de datos	PostgreSQL	Robustez, soporte ACID, escalabilidad
ORM	SQLAlchemy	Abstracción de base de datos, gestión de migraciones

ML Framework	TensorFlow/Keras	Compatibilidad con el modelo entrenado

Nota. La selección de tecnologías priorizó el rendimiento, la escalabilidad y la facilidad de mantenimiento, utilizando herramientas de código abierto ampliamente adoptadas en la industria.

La selección de tecnologías para la implementación del servidor, detallada en la Tabla 15, se basó en criterios de rendimiento, compatibilidad y facilidad de desarrollo. Cada componente fue elegido considerando tanto las necesidades actuales del sistema como su capacidad de escalabilidad futura.

Durante la fase de validación y pruebas con usuarios externos, se utilizó Ngrok como solución para exponer temporalmente el servidor local a través de Internet. Ngrok establece un túnel seguro que conecta el servidor de desarrollo local con una URL pública accesible desde cualquier dispositivo con conexión a Internet. Esta aproximación ofreció ventajas significativas durante las pruebas, incluyendo rapidez de configuración que permite exponer el servidor en segundos sin infraestructura de red compleja, seguridad mediante cifrado TLS que garantiza la confidencialidad de los datos transmitidos, flexibilidad para facilitar pruebas con usuarios reales sin despliegue en producción, y capacidades de monitoreo a través de una interfaz web para inspeccionar solicitudes entrantes. Es importante señalar que Ngrok se utilizó exclusivamente durante la fase de validación del prototipo. Para un despliegue en producción, se recomienda utilizar infraestructura de nube dedicada como AWS, Google Cloud o Azure, con configuraciones apropiadas de seguridad, balanceo de carga y alta disponibilidad.

4.1.7.3. Front-end

La interfaz de usuario constituye el punto de contacto entre el sistema de clasificación y los usuarios finales, por lo que su diseño debe priorizar la usabilidad, la claridad en la

presentación de resultados y la accesibilidad para usuarios con diferentes niveles de experiencia técnica. El *front-end* desarrollado permite a los usuarios interactuar con el sistema de manera intuitiva, desde la carga de imágenes hasta la visualización de resultados y el acceso al historial de consultas.

Tabla 16

Funcionalidades principales del front-end

Funcionalidad	Descripción	Características
Carga de imágenes	Interfaz para subir imágenes de hojas de banano	Arrastrar y soltar, selector de archivos, validación de formatos JPEG y PNG
Visualización de predicciones	Presentación de resultados de clasificación	Nombre de enfermedad, porcentaje de confianza, indicador visual de color, recomendaciones preliminares
Historial de consultas	Registro de predicciones anteriores	Seguimiento temporal, comparación de múltiples muestras
Retroalimentación visual	Indicadores de estado del procesamiento	Spinners, barras de progreso durante la inferencia

Nota. Estas funcionalidades fueron diseñadas para ofrecer una experiencia de uso intuitiva, permitiendo que usuarios sin conocimientos técnicos puedan realizar diagnósticos de manera sencilla.

El *front-end* del prototipo incluye las funcionalidades principales detalladas en la Tabla 16. Los usuarios pueden cargar imágenes mediante un componente de arrastrar y soltar o un selector de archivos tradicional, con validación automática del tipo de archivo antes de enviar la solicitud al servidor. Una vez procesada la imagen, el sistema presenta el resultado de la clasificación de manera clara, incluyendo el nombre de la enfermedad detectada o la indicación de hoja sana, el porcentaje de confianza del modelo, una representación visual mediante indicadores de color para interpretación rápida, e información adicional sobre la enfermedad con

recomendaciones preliminares. El sistema mantiene un registro de consultas realizadas que permite revisar predicciones anteriores, facilitando el seguimiento de la evolución de cultivos a lo largo del tiempo. Durante el procesamiento, se proporcionan indicadores visuales que informan al usuario sobre el estado de la solicitud, mejorando la experiencia de uso.

Tabla 17

Flujo de validación end-to-end del sistema

Etapa	**Componente responsable**	**Acción realizada**
carga de imagen	Front-end	Usuario selecciona o arrastra imagen de hoja de banano
envió al servidor	Front-end	Transmisión mediante petición HTTP POST
procesamiento	Back-end	Preprocesamiento de imagen y ejecución del modelo
recepción de resultados	Front-end	Recibe respuesta JSON con predicción y confianza
presentación	Front-end	Muestra resultados de manera visual e intuitiva
actualización del historial	Front-end + Back-end	Registra consulta para referencia futura

Nota. Este flujo describe la interacción completa entre el usuario y el sistema, validando que todos los componentes funcionan correctamente de forma integrada.

La validación del *front-end* se realizó mediante pruebas de flujo completo que verificaron la correcta integración de todos los componentes del sistema. El flujo validado, ilustrado en la Figura 9 y descrito en la Tabla 17, comprende desde la carga inicial de la imagen por parte del usuario hasta el registro final de la consulta en el historial, pasando por todas las etapas intermedias de comunicación entre *front-end* y *back-end*.

Tabla 18

Principios de diseño de la interfaz de usuario

Principio	**Implementación**
Simplicidad	Presentación únicamente de elementos necesarios, evitando sobrecarga de información
Retroalimentación inmediata	Confirmación visual de acciones del usuario (selección de imagen, inicio de procesamiento)
Manejo de errores	Mensajes claros y constructivos indicando cómo resolver problemas

Nota. Estos principios garantizan que la interfaz sea accesible para usuarios con diferentes niveles de experiencia técnica, priorizando la facilidad de uso en entornos de campo.

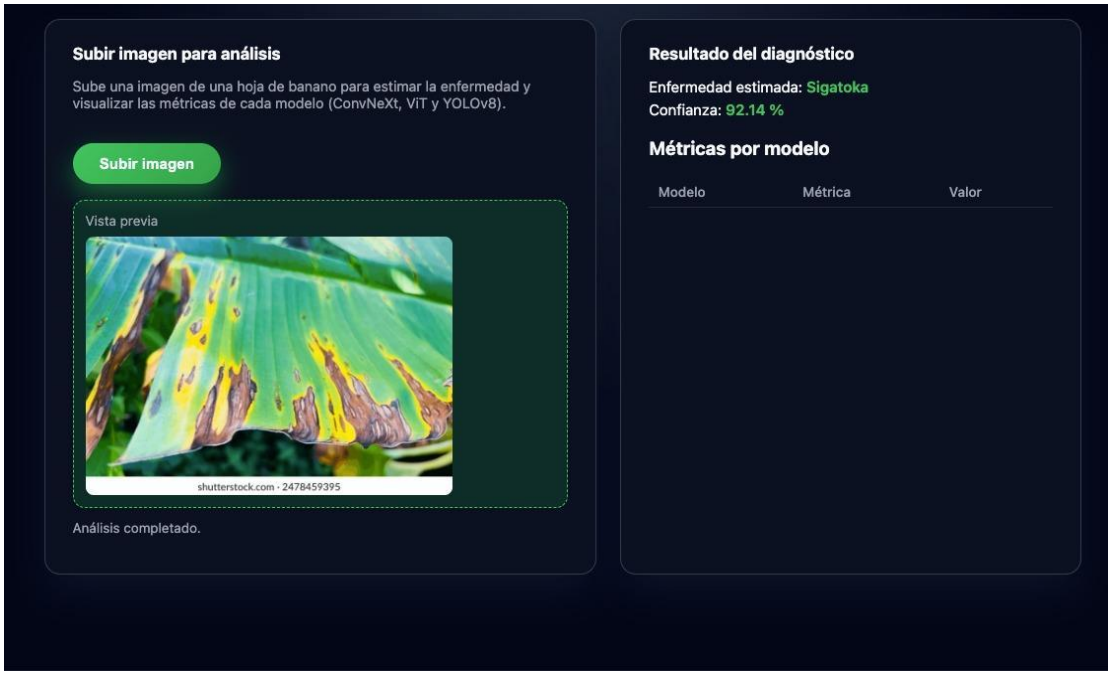
El diseño de la interfaz se fundamentó en los principios detallados en la Tabla 18. La simplicidad evita sobrecargar al usuario con opciones innecesarias, mientras que la retroalimentación inmediata proporciona confirmación visual de cada acción realizada. El manejo de errores presenta mensajes claros como "Formato de imagen no soportado. Por favor, utilice archivos JPEG o PNG", orientando al usuario hacia la solución.

Las pruebas de validación se ejecutaron con el prototipo funcionando de manera local, conectando el *front-end* con el back-end a través de la URL proporcionada por Ngrok. Esta configuración permitió simular un escenario de uso real donde el usuario accede al sistema desde un dispositivo diferente al servidor, validando la correcta comunicación entre componentes a través de la red. Finalmente, se verificó el flujo completo del sistema desde la carga de la imagen en el *front-end*, como se ilustra en la Figura 9, la obtención del resultado y la visualización del historial, ejecutando de esta manera el prototipo de manera local. Los resultados confirmaron que el sistema es capaz de procesar solicitudes de clasificación de manera exitosa, desde la

interacción inicial del usuario hasta la presentación final de los resultados, validando la viabilidad técnica de la solución propuesta.

Figura 9

Interfaz de Usuario prototipo



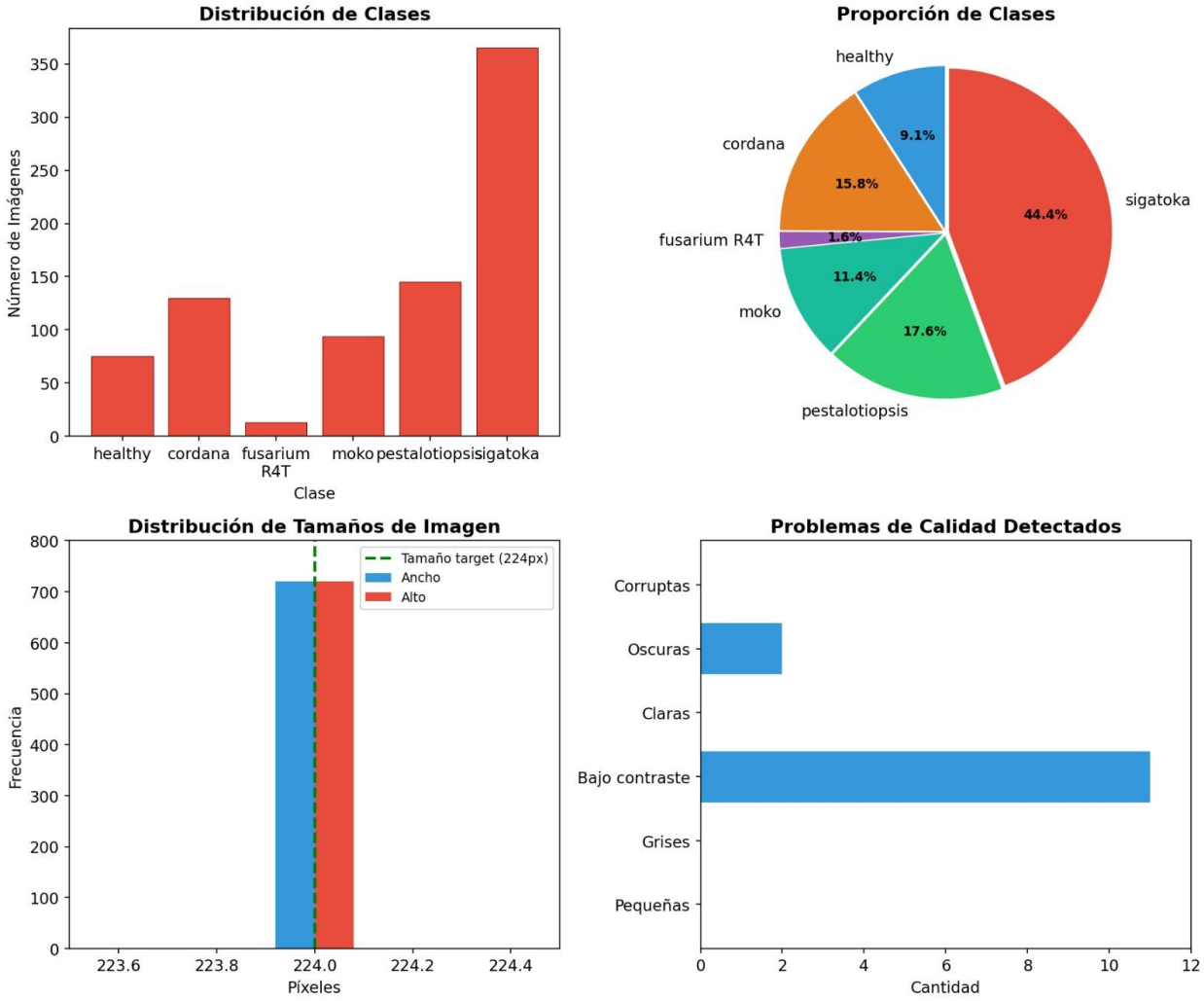
Nota. Interfaz de usuario del prototipo funcional desarrollado mediante una API y una UI para la visualización de resultados del modelo.

4.2.Análisis de Resultados

Se emplearon 1025 imágenes distribuidas en 6 clases como se puede ver en la figura 10, con un tamaño de 224x224, el *batch size* de 64 y una tasa de aprendizaje de 0.001, lo que se puede visualizar en la Tabla 19. El resultado es un aprendizaje en un inicio rápido, pero con la suficiente estabilidad para no dañar la generalización, lo que establece la estabilidad en la convergencia.

Figura 10

Características del conjunto de datos empleado



Nota. La figura muestra la composición del conjunto de datos empleado, incluyendo la cantidad de imágenes por clase utilizadas en las etapas de entrenamiento, validación y prueba.

4.2.1. Desempeño global del modelo

El modelo EfficientNet obtuvo las métricas de *accuracy* 96.94% y F1 macro 96.67%, con menos probabilidad de acierto en sigatoka y cordana debido a su origen epidemiológico y la

coincidencia visual, mientras que el modelo de YOLOv8-cls alcanzó el 96.67% para el top 1 y el 100% para el top -5 y, las confusiones aquí se concentraron en moko y *Pestalotiopsis*, como se expresa en la Tabla 8.

Tabla 20

Comparación de los dos modelos EfficientNet y YOLOv8s-cls

Modelo	**Tipo**	**Métrica principal**
efficientNet	Clasificación	Accuracy 96.94
YOLOv8s-cls	Clasificación	Top1 96.67% y Top5 100%

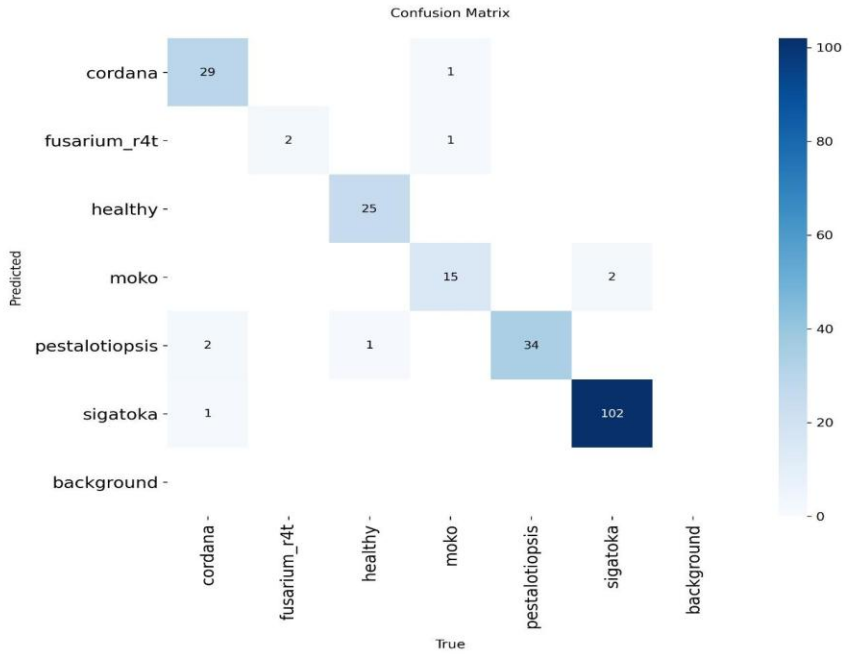
Nota. EfficientNet reporta accuracy global; YOLOv8s-cls reporta Top1 (primera predicción correcta) y Top5 (correcta entre las cinco primeras predicciones).

4.2.2. Matriz de confusión

La matriz de confusión figura 11 normalizada deja en evidencia la alta capacidad en discriminación entre clases, los valores reflejados son aproximadamente 0.90 en la mayoría de categorías. Las clases sigatoka y *Pestalotiopsis* muestran los mayores aciertos, mientras que la enfermedad de moko registra el menor desempeño relativo cercano a 0.88, presentando confusiones principalmente con la clase sigatoka, esto debido a las similitudes que existen entre las lesiones foliares. Asimismo, se observan errores puntuales entre Cordana y Pestalotiopsis.

Figura 11

Matriz de confusión del modelo



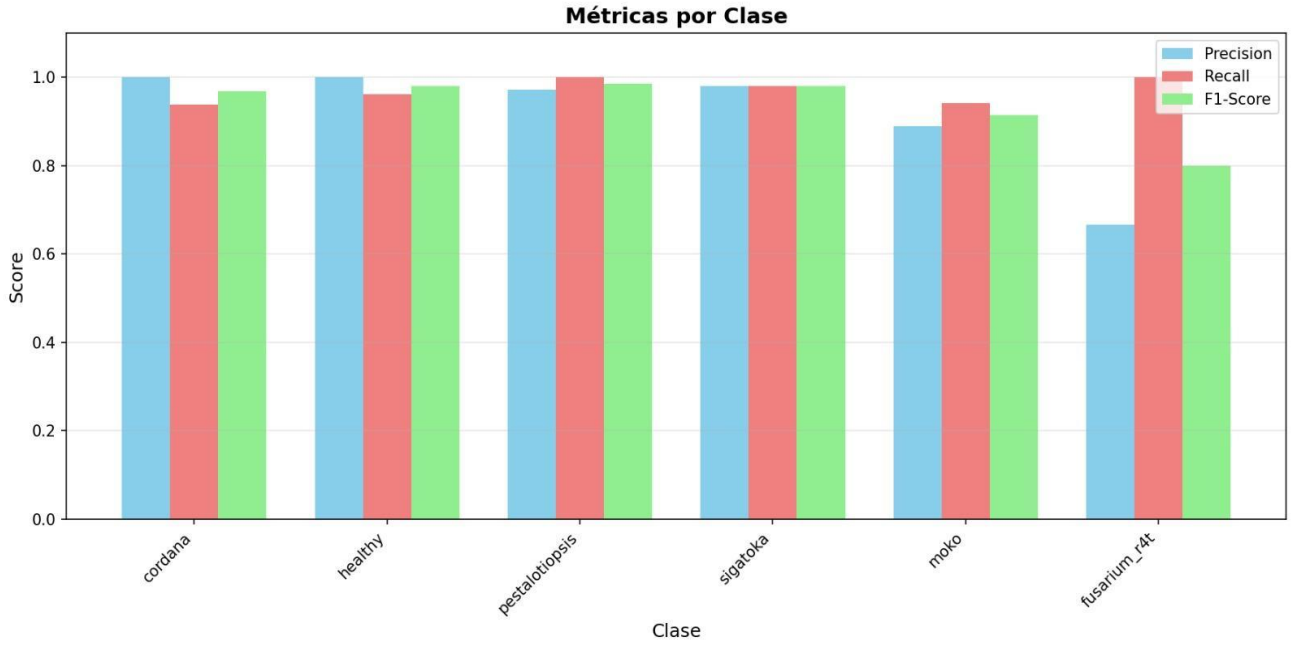
Nota. Matriz de confusión del modelo de clasificación, que refleja el desempeño del modelo al comparar las clases reales frente a las clases predichas durante la fase de evaluación.

4.2.3. Resultados por clase

La figura 12 presenta los resultados del modelo de clasificación por clase, expresados mediante las métricas de *precision*, *recall, F1-scrore.* En la figura se observa las clases en su mayoría alcanzan valores altos y consistentes en las tres métricas, lo que indica un desempeño estable del modelo para estas categorías.

Figura 12

Métricas del modelo por clase



Nota. Métricas de *precision, recall y F1-score* calculadas por clase para el modelo de clasificación, considerando las diferentes enfermedades foliares del banano.

4.2.4. Resultados del prototipo

4.2.4.1. API

El sistema incorpora una capa de servicio encargada de mediar la comunicación entre el front-end y el back-end de inferencia, donde se encuentra desplegado el modelo previamente entrenado. Esta capa expone endpoints que permiten el envío de imágenes para su procesamiento, gestionando tanto la transferencia de datos como la ejecución del modelo de inferencia de manera eficiente. A pesar del costo computacional asociado a la evaluación del modelo y a la carga de archivos de imagen, los tiempos de respuesta se mantienen dentro de

rangos óptimos, lo que garantiza una experiencia de usuario fluida y adecuada para entornos de uso en tiempo real.

La respuesta generada por el servicio se entrega en formato JSON, siguiendo una estructura estandarizada que incluye: La clase predicha correspondiente a la enfermedad detectada (por ejemplo, sigatoka negra o Sana), El nivel de confianza asociado a la predicción, expresado mediante un *confidence score*, las probabilidades estimadas para las demás clases posibles, lo que permite un análisis más completo del resultado de la inferencia. Adicionalmente, esta capa implementa mecanismos robustos de validación y manejo de errores, permitiendo identificar y responder adecuadamente ante entradas inválidas por parte del usuario, tales como imágenes en formatos no soportados, archivos que exceden los límites de tamaño establecidos o solicitudes mal formadas. De esta manera, se asegura la estabilidad del sistema y se mejora la confiabilidad del servicio expuesto.

4.2.4.2. Base de datos

Se incorporó una base de datos relacional como mecanismo de persistencia, garantizando la integridad, consistencia y disponibilidad de la información generada por el sistema. Esta capa de almacenamiento permite registrar de manera estructurada los resultados derivados del análisis de las imágenes procesadas, facilitando la gestión y consulta de un historial de enfermedades detectadas por cada usuario. Como resultado, se mejora la trazabilidad de los datos y se optimiza la experiencia del usuario al proporcionar acceso a información histórica relevante.

Adicionalmente, con el objetivo de asegurar la escalabilidad y evolución de la plataforma, se habilitaron nuevos *endpoints* a través de una API REST para el registro de *feedback*. Estos *endpoints* permiten capturar información complementaria relacionada con los

diagnósticos obtenidos, lo que proporciona un mayor contexto sobre cada enfermedad detectada. Esta funcionalidad sienta las bases para futuras iteraciones del sistema, como el refinamiento de los modelos de análisis, la validación de resultados y la mejora en los procesos de tratamiento y seguimiento de las enfermedades identificadas.

Capítulo 5

5. Conclusiones y recomendaciones

5.1. Conclusiones

Los resultados obtenidos evidencian que el modelo de aprendizaje profundo implementado presenta un alto nivel de desempeño global, alcanzando una exactitud (*accuracy*) del 96,43%, así como valores elevados de *precision, recall y F1-score*. Estos indicadores demuestran que el sistema es capaz de identificar correctamente las enfermedades del banano con un bajo margen de error.

El valor del *recall* global (97,06%) indica que el sistema tiene una alta capacidad para detectar la mayoría de los casos positivos, reduciendo la probabilidad de que una planta enferma sea clasificada como sana. Esto es especialmente relevante en el contexto agrícola, donde los falsos negativos pueden tener consecuencias económicas significativas.

El análisis por clase muestra que el modelo presenta un desempeño sobresaliente en las clases sigatoka y pestalotiopsis, con valores de *F1-score* superiores al 97%, lo que evidencia una correcta diferenciación visual de estas enfermedades. Sin embargo, la clase cordana presenta una precisión inferior en comparación con las demás, lo que sugiere una mayor confusión con otras clases, posiblemente debido a similitudes visuales en los síntomas o a una menor cantidad de muestras disponibles.

Se observa que la clase sigatoka cuenta con un mayor número de muestras (*support*), lo cual se refleja en un desempeño más estable del modelo. Esto confirma que la cantidad y diversidad de datos influyen directamente en la capacidad de generalización del sistema.

5.2. Recomendaciones

Se recomienda incrementar la cantidad de imágenes, especialmente para clases con menor número de muestras como fusarium, cordana y healthy, incorporando variaciones en iluminación, ángulos y estados de la enfermedad, con el fin de mejorar la precisión del modelo en escenarios reales.

Se recomienda validar el sistema en entornos reales de cultivo, utilizando imágenes capturadas directamente por agricultores o técnicos agrícolas, lo que permitiría evaluar el desempeño del modelo frente a condiciones no controladas.

Se recomienda agregar información de condiciones meteorológicas y de suelo al momento de obtener las imágenes. Esto aportaría significativamente en la clasificación de enfermedades que compartan indicadores entre enfermedades.

Referencias bibliográficas

Abade, A. S., Ferreira, P. A., & Vidal, F. B. (2020). Plant diseases recognition on images using convolutional neural networks: A systematic review. *arXiv preprint arXiv:2009.04365*. https://doi.org/10.48550/arXiv.2009.04365

Agencia de Regulación y Control Fito y Zoosanitario. (2024). Guía técnica de Moko. https://www.agrocalidad.gob.ec/wp-content/uploads/2024/10/Guia_Ralstonia.pdf

Agencia de Regulación y Control Fito y Zoosanitario. (2025). *Boletín epidemiológico Foc R4T: Resultados de la vigilancia fitosanitaria del período enero–marzo, 2025*. Comunidad Andina. https://www.comunidadandina.org/documents/temas/dg-com/sanidad-vegetal/Foc_R4T_Ecuador_1_Trimestre.pdf

Agencia de Regulación y Control Fito y Zoosanitario. (2024). *Boletín informativo: Estrategias para prevenir el ingreso de Fusarium oxysporum f.sp. raza 4 tropical (Foc R4T) (Período enero–marzo 2024)*. https://www.agrocalidad.gob.ec/wp-content/uploads/2024/07/BOLETIN-Foc-R4T-Ene–Mar-2024.pdf

Arman, S. E., Bhuiyan, M. A. B., Abdullah, H. M., Islam, S., Chowdhury, T. T., & Hossain, M. A. (2023). BananaLSD: A banana leaf images dataset for classification of banana leaf diseases using machine learning. *Data in Brief*, *50*, 109608

Arunkumar, R., & Suthin Raj, T. (2021). *Major diseases of banana and its management*. ResearchGate. https://www.researchgate.net/publication/352056982_Major_Diseases_of_Banana_and_its_Management

Blandón, J. C., Gómez, D. J., Sierra, J. M., & Hincapié, D. J. (2024). First report of white root rot caused by *Rosellinia necatrix* on blackberry (*Rubus glaucus*) in Colombia. *Plant Pathology*, *73*(3), 856. https://doi.org/10.1111/ppa.13863

Blomme, G., Dita, M., Jacobsen, K. S., Pérez Vicente, L., Molina, A., Ocimati, W., Poussier, S., & Prior, P. (2017). Bacterial diseases of bananas and enset: Current state of knowledge and integrated approaches toward sustainable management. *Frontiers in Plant Science*, *8*, 1290. https://doi.org/10.3389/fpls.2017.01290

Barbedo, J. G. A. (2018). Factors influencing the use of deep learning for plant disease recognition. *Biosystems Engineering*, *172*, 84–91. https://doi.org/10.1016/j.biosystemseng.2018.05.013

Cervantes-Álava, A., Sánchez-Urdaneta, A., Colmenares, C., & Quevedo-Guerrero, J. (2023). Evaluación de fungicidas utilizados en el manejo de sigatoka negra en el cultivo de banano. *Revista de la Facultad de Agronomía de la Universidad del Zulia*, *40*(2), e234016. https://doi.org/10.47280/RevFacAgron(LUZ).v40.n2.06

Cedeño Campoverde, K. J. (2024). *Sistema de teledetección utilizando drones para el monitoreo de banano* [Tesis de grado no publicada]. Universidad Técnica Estatal de Quevedo.

Corporación Financiera Nacional. (2024, October). *Ficha sectorial banano*. http://cfn.fin.ec/wp-content/uploads/2024/10/Ficha-Sectorial-Banano.pdf

Cui, X., Hao, Z., Chen, M., Song, S., Zhang, J., Li, Y., Li, J., Liu, Y., & Luo, L. (2024). Identification and pathogenicity of pestalotioid species on Alpinia oxyphylla in Hainan Province, China. *Journal of Fungi*, *10*(6), 371. https://doi.org/10.3390/jof10060371

Churchill, A. C. L. (2011). *Mycosphaerella fijiensis*, the black leaf streak pathogen of banana: Progress towards understanding pathogen biology and detection, disease development, and the challenges of control. *Molecular Plant Pathology*, *12*(4), 307–328. https://doi.org/10.1111/j.1364-3703.2010.00672.x

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. *In Proceedings of the International Conference on Learning Representations*. https://openreview.net/forum?id=YicbFdNTTy

Fu, Y. P., Qi, Y. X., Xie, Y. X., Peng, J., Zeng, F. Y., & Zhang, X. (2021). Identification and biological characteristics of the pathogen causing Cordana leaf spot of banana in Hainan Province. *Journal of Tropical Biology*, *12*(1), 61-68. https://doi.org/10.15886/j.cnki.rdswxb.2021.01.009

Food and Agriculture Organization of the United Nations. (s. f.). *Bananas*. https://www.fao.org/markets-and-trade/commodities-overview/bananas-tropical-fruits/bananas/5/en

Grajales-Amorocho, M., Acosta-Minoli, C., Muñoz-Pizza, D. M., Manrique-Arias, O., & Munoz-Loaiza, A. (2024). Analysis of Moko disease propagation on plantain (*Musa* AAB Simmonds) through a model based on system dynamics. *European Journal of Plant Pathology*, *168*, 37–50. https://doi.org/10.1007/s10658-023-02764-2

Géron, A. (2022). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (3rd ed.). O'Reilly Media.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

Liu, J., & Wang, X. (2021). Plant diseases and pests detection based on deep learning: A review. *Plant Methods*, *17*, 22. https://doi.org/10.1186/s13007-021-00722-9

Jiménez, N., Orellana, S., Mazon-Olivo, B., Rivas-Asanza, W., & Ramírez-Morales, I. (2025). Detection of leaf diseases in banana crops using deep learning techniques. *AI*, *6*(3), 61. https://doi.org/10.3390/ai6030061

Linero-Ramos, R., Parra-Rodríguez, C., Espinosa-Valdez, A., Gómez-Rojas, J., & Gongora, M. (2024). Assessment of dataset scalability for classification of Black Sigatoka in banana crops using UAV-based multispectral images and deep learning techniques. *Drones*, *8*(9), 503. https://doi.org/10.3390/drones8090503

Ministerio de Agricultura y Ganadería. (2023). *Boletín situacional banano 2023*. Sistema de Información Pública Agropecuaria (SIPA). https://sipa.agricultura.gob.ec/boletines/situacionales/2023/boletin_situacional_banano_2023.pdf

Mduma, N., & Elinisa, C. (2025). Banana leaves imagery dataset. *Scientific Data*, *12*, 477. https://doi.org/10.1038/s41597-025-04456-4

Munhoz, T., Vargas, J., Teixeira, L., Staver, C., & Dita, M. (2024). *Fusarium* tropical race 4 in Latin America and the Caribbean: Status and global research advances towards disease management. *Frontiers in Plant Science*, *15*, 1397617. https://doi.org/10.3389/fpls.2024.1397617

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Frontiers in Plant Science*, *7*, 1419. https://doi.org/10.3389/fpls.2016.01419

Noar, R. D., Thomas, E., & Daub, M. E. (2022). Genetic characteristics and metabolic interactions between *Pseudocercospora fijiensis* and banana: Progress toward controlling black sigatoka. *Plants*, *11*(7), 948. https://doi.org/10.3390/plants11070948

Noyan, M. A. (2022). Uncovering bias in the PlantVillage dataset. *arXiv*. https://doi.org/10.48550/arXiv.2206.04374

Pinzón-Núñez, A. M., Feria-Gómez, D. F., Pérez-Ochoa, G. M., Ramírez-Gil, J. G., & Sanjuán, T. (2024). New standard area diagram set for assessing black sigatoka in bananas. *European Journal of Plant Pathology*, *170*, 535–548. https://doi.org/10.1007/s10658-024-02917-x

Romero-García, C. V., Saraguro-Reyes, C. M., Mazon-Olivo, B. E., & Morocho-Román, R. F. (2025). Agricultura de precisión en la producción de banano: Revisión sistemática. *Ingenium et Potentia. Revista Electrónica Multidisciplinaria de Ciencias Básicas, Ingeniería y Arquitectura*, *7*(12), 50-80. https://doi.org/10.35381/i.p.v7i12.4450

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2020). Grad-CAM: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, *128*(2), 336–359. https://doi.org/10.1007/s11263-019-01228-7

Sanga, S., Mero, V., & Machuve, D. (2020). Mobile-based deep learning models for banana diseases detection. *arXiv*. https://doi.org/10.48550/arXiv.2004.03718

Nelson, S. (2011, May 2). *Banana Cordana leaf spot 1* [Photograph]. Flickr. https://www.flickr.com/photos/scotnelson/5680832197

Taye, M. M. (2023). Theoretical understanding of convolutional neural networks: Concepts, architectures, applications, and future directions. *Computation*, *11*(3), 52. https://doi.org/10.3390/computation11030052

Ploetz, R. C. (2015). Management of Fusarium wilt of banana: A review with special reference to tropical race 4. *Crop Protection*, *73*, 7-15. https://doi.org/10.1016/j.cropro.2015.01.007

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779-788). https://doi.org/10.1109/CVPR.2016.91

Paladines Larco, F. A. (2025). *Visión por computadora y aprendizaje profundo para conteo y salud de plantas de banano* [Tesis de grado no publicada]. Escuela Superior Politécnica del Litoral.

Thiagarajan, J. D., Kulkarni, S. V., Jadhav, S. A., Waghe, A. A., Raja, S. P., Rajagopal, S., Poddar, H., & Subramaniam, S. (2024). Analysis of banana plant health using machine learning techniques. *Scientific Reports*, *14*(1), 15041. https://doi.org/10.1038/s41598-024-63930-y

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning* (pp. 6105-6114). PMLR. http://proceedings.mlr.press/v97/tan19a.html

Ultralytics. (2023). *Ultralytics YOLO documentation*. https://docs.ultralytics.com

Ugarte Fajardo, J., Bayona Andrade, O., Criollo Bonilla, R., Cevallos-Cevallos, J., Mariduena-Zavala, M., & Vicente Villardón, J. L. (2020). Early detection of black Sigatoka in banana leaves using hyperspectral images. *Applications in Plant Sciences*, *8*(8), e11383.

Upadhyay, A., Chandel, N. S., Singh, K. P., Chakraborty, S. K., Nandede, B. M., Kumar, M., & Elbeltagi, A. (2025). Deep learning and computer vision in plant disease detection: A comprehensive review of techniques, models, and trends in precision agriculture. *Artificial Intelligence Review*, *58*(3), 1–64. https://doi.org/10.1007/s10462-024-11100-x

Apéndice A. Repositorio de código del proyecto

El código fuente, modelos entrenados y recursos técnicos desarrollados en este trabajo de titulación se encuentran disponibles públicamente para fines de reproducibilidad, validación académica y transparencia científica.

Repositorio de GitHub: https://github.com/kevinjimenez/banana-disease-classifier

Ubicación de imágenes/dataset dentro del repositorio: dataset/test y dataset/train.

Apéndice B. Base de datos con imágenes propias

Imágenes moko:

https://drive.google.com/drive/folders/1K0cs3Ni50f2cXKOcRdLtCu31ZszNWjst?usp=drive_link