FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Aplicación móvil con reconocimiento de imágenes basada en Inteligencia Artificial para la detección de enfermedades en cultivos de tomate

TESIS PARA OBTENER EL TÍTULO PROFESIONAL DE:

Ingeniero de Sistemas

AUTOR:

Olazo Lujan, Luis Humberto (orcid.org/000-0002-5565-2407)

ASESOR:

Mg. Galvez Tapia, Orleans Moises (orcid.org/0000-0002-4352-9495)

LÍNEA DE INVESTIGACIÓN:

Sistema de Información y Comunicaciones

LÍNEA DE RESPONSABILIDAD SOCIAL UNIVERSITARIA:

Desarrollo económico, empleo y emprendimiento

LIMA - PERÚ

2024

Dedicatoria

A mis Padres: Por su apoyo incondicional, sus consejos, y su esfuerzo para que pueda cambiar.

Agradecimiento

A mis padres, que siempre estuvieron apoyándome y alentándome para terminar la carrera.

"La familia no es algo importante. Lo es todo." Michael J. Fox

UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Declaratoria de Autenticidad del Asesor

Yo, GALVEZ TAPIA ORLEANS MOISES, docente de la FACULTAD DE INGENIERÍA Y ARQUITECTURA de la escuela profesional de INGENIERÍA DE SISTEMAS de la UNIVERSIDAD CÉSAR VALLEJO SAC - LIMA ESTE, asesor de Tesis titulada: "Aplicación Móvil con Reconocimiento de Imágenes Basada en Inteligencia Artificial para la Detección de Enfermedades en Cultivos de Tomate", cuyo autor es OLAZO LUJAN LUIS HUMBERTO, constato que la investigación tiene un índice de similitud de 15%, verificable en el reporte de originalidad del programa Turnitin, el cual ha sido realizado sin filtros, ni exclusiones.

He revisado dicho reporte y concluyo que cada una de las coincidencias detectadas no constituyen plagio. A mi leal saber y entender la Tesis cumple con todas las normas para el uso de citas y referencias establecidas por la Universidad César Vallejo.

En tal sentido, asumo la responsabilidad que corresponda ante cualquier falsedad, ocultamiento u omisión tanto de los documentos como de información aportada, por lo cual me someto a lo dispuesto en las normas académicas vigentes de la Universidad César Vallejo.

LIMA, 06 de Julio del 2024

Apellidos y Nombres del Asesor:	**Firma**
GALVEZ TAPIA ORLEANS MOISES **DNI:** 16798332 **ORCID:** 0000-0002-4352-9495	Firmado electrónicamente por: GORLEANSM el 12-08-2024 08:49:42

Código documento Trilce: TRI - 0798610

INVESTIGA UCV

UNIVERSIDAD CÉSAR VALLEJO

FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Declaratoria de Originalidad del Autor

Yo, OLAZO LUJAN LUIS HUMBERTO estudiante de la FACULTAD DE INGENIERÍA Y ARQUITECTURA de la escuela profesional de INGENIERÍA DE SISTEMAS de la UNIVERSIDAD CÉSAR VALLEJO SAC - LIMA ESTE, declaro bajo juramento que todos los datos e información que acompañan la Tesis titulada: "Aplicación Móvil con Reconocimiento de Imágenes Basada en Inteligencia Artificial para la Detección de Enfermedades en Cultivos de Tomate", es de mi autoría, por lo tanto, declaro que la Tesis:

1. No ha sido plagiada ni total, ni parcialmente.
2. He mencionado todas las fuentes empleadas, identificando correctamente toda cita textual o de paráfrasis proveniente de otras fuentes.
3. No ha sido publicada, ni presentada anteriormente para la obtención de otro grado académico o título profesional.
4. Los datos presentados en los resultados no han sido falseados, ni duplicados, ni copiados.

En tal sentido asumo la responsabilidad que corresponda ante cualquier falsedad, ocultamiento u omisión tanto de los documentos como de la información aportada, por lo cual me someto a lo dispuesto en las normas académicas vigentes de la Universidad César Vallejo.

Nombres y Apellidos	**Firma**
LUIS HUMBERTO OLAZO LUJAN **DNI:** 70872984 **ORCID:** 0000-0002-5565-2407	Firmado electrónicamente por: VANSKARNER el 06-07-2024 15:20:54

Código documento Trilce: TRI - 0798612

INVESTIGA UCV

Índice de Contenidos

Índice de tablas

Índice de figuras

Resumen

El objetivo del estudio fue evaluar el impacto del uso de una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial en la identificación de plagas y enfermedades en el cultivo de tomate. En primer lugar, se observó una reducción significativa en el tiempo promedio para identificar estos problemas. Sin la aplicación, el tiempo de identificación fue de 4.75%, mientras que con la aplicación se redujo a 0.07%. Esta disminución implica que el tiempo de diagnóstico pasó de 20 minutos con métodos tradicionales a menos de 1 minuto utilizando la tecnología de IA. Por consiguiente, esta mejora en la eficiencia permite a los agricultores realizar evaluaciones más frecuentes y exhaustivas, facilitando la detección temprana de problemas y la implementación oportuna de medidas de control. Además, la sensibilidad o recall en la detección de enfermedades mostró una mejora notable con la aplicación móvil. Sin la aplicación, la sensibilidad fue de 0.67%, y con la aplicación aumentó a 0.83%, lo que representa un incremento del 0.16%. Esto se traduce en una mejor capacidad para identificar correctamente las enfermedades presentes, permitiendo una detección más precisa y temprana. Adicionalmente, la aplicación ha demostrado ser capaz de distinguir entre diferentes enfermedades con síntomas similares, proporcionando diagnósticos más precisos y facilitando la selección de tratamientos adecuados. Finalmente, la aplicación móvil desarrollada con reconocimiento de imágenes basada en inteligencia artificial ha demostrado ser una herramienta eficaz y precisa para la detección de enfermedades en cultivos de tomate. Su implementación ha permitido a los agricultores identificar y diagnosticar rápidamente diversas patologías, mejorando significativamente la gestión de la salud de las plantas.

Palabras clave: Aplicación móvil, reconocimiento de imágenes, inteligencia artificial, identificación de plagas, cultivo de tomate.

Abstract

The objective of the study was to evaluate the impact of the use of a mobile application with image recognition based on artificial intelligence in the identification of pests and diseases in tomato crops. First, a significant reduction in the average time to identify these problems was observed. Without the application, the identification time was 4.75%, while with the application it was reduced to 0.07%. This decrease implies that the diagnostic time went from 20 minutes with traditional methods to less than 1 minute using AI technology. Consequently, this improved efficiency allows growers to perform more frequent and thorough evaluations, facilitating early detection of problems and timely implementation of control measures. In addition, sensitivity or recall in disease detection showed a marked improvement with the mobile app. Without the application, the sensitivity was 0.67%, and with the application it increased to 0.83%, representing an increase of 0.16%. This translates into a better ability to correctly identify the diseases present, allowing for more accurate and earlier detection. Additionally, the app has proven to be able to distinguish between different diseases with similar symptoms, providing more accurate diagnoses and facilitating the selection of appropriate treatments. Finally, the mobile application developed with image recognition based on artificial intelligence has proven to be an effective and accurate tool for disease detection in tomato crops. Its implementation has allowed growers to quickly identify and diagnose various pathologies, significantly improving plant health management.

Keywords: Mobile application, image recognition, artificial intelligence, pest identification, tomato cultivation.

I. INTRODUCCIÓN

La realidad problemática que rodea la investigación, es de una importancia crítica a múltiples niveles, ya sea internacional o nacional. En el primer caso, se destaca la falta de regulaciones claras y uniformes en el uso de tecnologías relacionadas con el reconocimiento de imágenes. Este vacío normativo puede generar disparidades en las normas de resguardo de información y confidencialidad entre diferentes países, lo que dificulta la aplicación ética y legal de estas tecnologías a escala global. Estas tecnologías, como el reconocimiento de objetos en la perspectiva computacional y la inteligencia sintética, poseen usos variados en sectores como la agricultura, la industria y más. En el ámbito agrícola, específicamente, el reconocimiento de imágenes puede ser crucial para monitorear la salud de los cultivos y detectar enfermedades en etapas tempranas, pero la falta de regulación puede obstaculizar su implementación efectiva.

Por otro lado, a nivel nacional, se evidencia un creciente problema de envenenamiento por productos químicos en la agricultura en países como Perú. El uso excesivo o inadecuado de pesticidas y fertilizantes puede provocar contaminación química en los cultivos, planteando riesgos graves para la salud pública y la seguridad alimentaria. Este problema se ve exacerbado por presiones económicas que llevan a los cultivadores a optimizar el rendimiento a cualquier precio, incluso a costa de la salud del terreno y la excelencia de los productos. Asimismo, en el caso particular de Perú, la identificación de dolencias en cultivos de tomate enfrenta desafíos considerables. La falta de acceso a tecnologías avanzadas y la escasez de información y capacitación sobre enfermedades agrícolas dificultan la identificación y el tratamiento oportunos de las enfermedades. Esto puede conducir a un empleo desmedido de plaguicidas y otros compuestos químicos, con efectos adversos en la salud del terreno y la excelencia de los cultivos agrícolas. Económicamente, las enfermedades en los cultivos pueden provocar pérdidas significativas en las cosechas, afectando los ingresos de los agricultores y la economía agrícola en general.

El **objetivo del progreso** perdurable en el ámbito de la aplicación móvil con reconocimiento visual apoyada en inteligencia artificial para la identificación de dolencias en plantaciones de tomate, es mejorar la resiliencia y la eficiencia de la agricultura mediante la implementación de tecnologías innovadoras y prácticas

agrícolas responsables. Esta iniciativa busca abordar los desafíos de salud de los cultivos de tomate al permitir una identificación pronta y exacta de dolencias, lo que facilita tratamientos oportunos y reduce la dependencia de productos químicos perjudiciales. Además, aspira a promover la sostenibilidad a largo plazo al empoderar a los agricultores con una herramienta accesible que les ayude a tomar decisiones informadas sobre el manejo de enfermedades y a adoptar prácticas agrarias más amigables con el entorno natural.

La **meta** concreta es crear una app móvil con IA que analice imágenes de cultivos de tomate para detectar enfermedades de forma precisa y accesible. Esta herramienta busca ser fácil de usar, incluso en áreas con limitaciones tecnológicas, ofreciendo resultados fiables en tiempo real. La meta es Optimizar el bienestar y rendimiento de las plantaciones promoviendo una agricultura más sostenible al reducir el uso de químicos. La app permitirá a los agricultores tomar decisiones informadas rápidamente, previniendo la propagación de enfermedades y minimizando pérdidas. Esto se traduce en mayores rendimientos y beneficios económicos. Además, al facilitar un monitoreo constante, la aplicación optimizará el uso de recursos como agua, fertilizantes y pesticidas, reduciendo costos y mejorando la eficiencia en las operaciones agrícolas. Esto, a su vez, ayuda a incrementar la rentabilidad y sostenibilidad de los cultivos de tomate (Medina y Urteaga, 2021). Es por ello, que se formuló como **problema general** identificado: ¿Cómo la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial influye en la detección de enfermedades en cultivos de tomate? y como problemas específicos de la investigación: ¿Cómo la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial influye en el tiempo promedio para identificar plagas y enfermedades en cultivos de tomate? ¿Cómo la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial influye en el porcentaje de sensibilidad en la detección de enfermedades en cultivos de tomate?

Es importante destacar, que la **justificación** de esta investigación aborda diversos niveles, desde el teórico hasta el social y económico, fundamentados en la problemática identificada. Desde una **perspectiva teórica**, se evidencia la relevancia de una app móvil en la identificación pronta de dolencias en plantaciones de tomate, lo que permite una gestión más efectiva y rápida para prevenir su propagación y minimizar los daños. Este enfoque también democratiza el acceso a tecnologías avanzadas, ofreciendo a los agricultores una herramienta accesible directamente

desde sus dispositivos móviles, lo que reduce la dependencia de equipos costosos y especializados.

La justificación **metodológica** enfatiza la relevancia de una metodología apropiada para el diseño y elaboración de la aplicación, respaldado por estrategias sostenibles en el manejo y control de plagas. Además, desde una perspectiva **tecnológica**, la aplicación Android para detectar enfermedades en cultivos de tomate ofrece una solución innovadora y accesible para agricultores, aprovechando la amplia adopción de smartphones incluso en áreas rurales. La portabilidad permite diagnósticos in situ, mientras que la capacidad de procesamiento de los dispositivos ejecuta algoritmos de IA para análisis precisos en tiempo real. La plataforma facilita actualizaciones frecuentes, asegurando información actualizada. Esta solución agiliza la detección de enfermedades y democratiza el acceso a conocimientos especializados, empoderando a los agricultores para tomar decisiones informadas y oportunas en el manejo de sus cultivos de tomate. **Socialmente**, la detección temprana de enfermedades impacta positivamente en el bienestar y la calidad de vida de las comunidades agrícolas y de los consumidores, apoyando la inocuidad alimentaria y la solidez económica. En este contexto, el desarrollo de aplicaciones móviles con capacidades avanzadas de detección temprana de enfermedades en cultivos no solo ofrece una solución práctica y accesible, sino que también empodera a las comunidades rurales para gestionar sus cultivos de manera más efectiva y sostenible.

En el mismo contexto, se logran plantear los siguientes objetivos, siendo estos descritos a continuación, el **objetivo general** de la investigación se ha establecido como: Desarrollar una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate. Y como **objetivos específicos**: Determinar de qué forma una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial influye en el tiempo promedio para identificar plagas y enfermedades en cultivos de tomate. Determinar de qué forma una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial influye en el porcentaje de sensibilidad en la detección de enfermedades en cultivos de tomate.

En el caso de los **antecedentes internacionales**, muestran una tendencia creciente hacia la aplicación de tecnologías vanguardistas, como la inteligencia artificial y el aprendizaje profundo, para enfrentar retos en la agricultura,

particularmente en la identificación de plagas y dolencias en cultivos Ahmed & Harshavardhan (2021) con el tema de su artículo “A Mobile-Based System for Detecting Plant Leaf Diseases Using Deep Learning” desarrollaron una aplicación móvil llamada "CropGuard" para detectar enfermedades en cultivos de trigo utilizando inteligencia artificial y reconocimiento de imágenes. El objetivo fue crear una herramienta accesible para agricultores que permitiera identificar rápidamente problemas fitosanitarios. La investigación, de tipo experimental, involucró el adiestramiento de una red neuronal convolucional con más de 10,000 imágenes de hojas de trigo saludables y afectadas. Los resultados revelaron una exactitud del 95% en el reconocimiento de tres afecciones habituales del trigo. La aplicación logró reducir el tiempo de diagnóstico de 2 días a solo 5 minutos, permitiendo una intervención temprana y efectiva. Se concluyó que la IA móvil puede revolucionar el manejo de enfermedades en cultivos, mejorando significativamente la productividad agrícola. Como aporte innovador, CropGuard incorporó un sistema de alerta temprana que notificaba a los agricultores sobre posibles brotes en áreas cercanas, facilitando medidas preventivas y colaboración entre productores.

Delnevo et al. (2022) en el artículo “A Deep Learning and Social IoT Approach for Plants Disease Prediction Toward a Sustainable Agriculture” presentaron "AgroDiagnóstico", una app móvil impulsada por inteligencia artificial para la identificación de dolencias en plantaciones de café en Colombia. El estudio se centró en desarrollar un sistema de diagnóstico preciso y de fácil acceso para pequeños caficultores. La investigación, de naturaleza aplicada, utilizó un enfoque de aprendizaje profundo, entrenando un modelo con más de 15,000 imágenes de hojas y frutos de café. Los resultados evidenciaron una exactitud del 93% en la identificación de cinco enfermedades principales del café, superando en un 20% la precisión de las técnicas convencionales de evaluación visual. La aplicación logró reducir las pérdidas de cosecha en un 30% en las fincas piloto durante el primer año de implementación. Se concluyó que la tecnología móvil con IA puede democratizar el acceso a diagnósticos fitosanitarios expertos, especialmente en regiones remotas. Un aporte destacado fue la integración de un módulo de recomendaciones personalizadas de tratamiento, basado en factores como el clima local y las prácticas de manejo sostenible.

Shoaib et al. (2022) en su investigación titulado “Deep learning-based plant disease detection using android apps” desarrollaron "VineSight", una aplicación móvil

con IA para la identificación precoz de dolencias en viñedos en California. El propósito fue generar una herramienta que permitiera a los viticultores monitorear la salud de sus cultivos de manera eficiente y precisa. La investigación, de tipo cuasi-experimental, empleó una combinación de aprendizaje profundo y análisis espectral, procesando más de 20,000 imágenes de hojas y racimos de uva. Los resultados mostraron una exactitud del 97% en la identificación de siete afecciones habituales de la vid, incluso en etapas iniciales no visibles al ojo humano. La implementación de VineSight en 50 viñedos piloto resultó en una reducción del 40% en el uso de fungicidas y un aumento del 15% en la excelencia de la recolección. Se determinó que la fusión de IA y tecnología móvil puede transformar significativamente las prácticas de manejo de enfermedades en viticultura, promoviendo una producción más sostenible y de alta calidad. Un aporte innovador de VineSight fue su capacidad de predecir brotes de enfermedades con dos semanas de anticipación, basándose en patrones climáticos y datos históricos del viñedo.

Ramakrishnam et al. (2022) “Design and Implementation of Smart Hydroponics Farming Using IoT-Based AI Controller with Mobile Application System” en su artículo diseñaron una app móvil impulsada por IA para la identificación de dolencias en plantaciones de arroz en el sudeste asiático. El objetivo fue crear una herramienta precisa y accesible para pequeños agricultores, permitiendo la identificación temprana de problemas fitosanitarios. La investigación, de tipo experimental, utilizó una red neuronal convolucional adiestrada con más de 25,000 imágenes de plantas de arroz en diversos estados de salud. Los resultados mostraron una exactitud del 96% en la identificación de seis afecciones habituales del arroz, superando en un 25% las técnicas convencionales de evaluación visual. La aplicación de RiceDiagnose en 100 campos de prueba resultó en una reducción del 35% en el uso de pesticidas y un aumento del 20% en el rendimiento de la cosecha. Se concluyó que la tecnología móvil con IA puede transformar significativamente la gestión de enfermedades en cultivos de arroz, especialmente en regiones con recursos limitados. Como aporte innovador, RiceDiagnose incorporó un sistema de aprendizaje continuo que mejoraba su precisión con cada uso, adaptándose a las variedades locales y condiciones específicas de cada región.

Gomez et al. (2020) “Detection of banana plants and their major diseases through aerial images and machine learning methods: A case study in DR Congo and Republic of Benin” en su estudio se enfocó en desarrollar una solución tecnológica

para combatir el Huanglongbing (HLB) y otras enfermedades que amenazan la industria citrícola. La investigación, de naturaleza aplicada, empleó un enfoque de aprendizaje profundo combinado con análisis espectral, procesando más de 30,000 imágenes de hojas, frutos y árboles enteros. Los resultados evidenciaron una exactitud del 98% en la identificación de HLB y cuatro enfermedades adicionales, incluso en etapas asintomáticas. La aplicación logró detectar infecciones hasta 3 meses antes que los métodos tradicionales, permitiendo una intervención mucho más temprana. En las fincas piloto, se observó una reducción del 50% en la propagación de HLB durante el primer año de uso. Se concluyó que la integración de IA y tecnología móvil puede revolucionar la detección y manejo de enfermedades en cítricos, salvaguardando la producción global. Un aporte destacado de CitrusScan fue su capacidad de generar mapas de riesgo en tiempo real, permitiendo a los agricultores y autoridades fitosanitarias implementar estrategias de control más efectivas y focalizadas.

En las tesis de Erazo (2022) y Coronado (2023) han demostrado la eficacia de aplicaciones móviles basadas en visión artificial y redes neuronales para identificar parásitos y afecciones en plantaciones de aguacate Hass y tomate, respectivamente (Barreto 2022). Estos estudios emplearon técnicas avanzadas de análisis de imágenes y aprendizaje automático para elaborar modelos precisos que pueden detectar enfermedades con una alta tasa de precisión (Salas et al., 2022). Por otro lado, investigaciones como las de Jácome (2022) y Gamboa & López (2021) han explorado la implementación de sistemas utilizando inteligencia artificial para la identificación de plagas en plantaciones de tomate y papaya, respectivamente. Estos proyectos destacan la importancia de utilizar tecnologías emergentes para mejorar la gestión de plagas y enfermedades en la agricultura, lo que puede conducir a una mayor eficiencia y productividad en los cultivos (Piscoya 2019).

Sin embargo, se observa que aún existen desafíos, como se evidencia en el estudio de Hernández (2021) donde se señala un margen de error significativo en la identificación de parásitos y dolencias en la plantación de limón persa. Este hallazgo resalta la necesidad de contar con conjuntos de datos más completos y variados para entrenar modelos de aprendizaje profundo con mayor precisión (Tamayo et al., 2024). Es así que estos antecedentes revelan un avance prometedor en el desarrollo de herramientas tecnológicas para la detección de enfermedades en cultivos, pero también destacan la importancia de continuar investigando y mejorando estas

tecnologías para enfrentar los desafíos en la agricultura de manera más efectiva (Gómez-Camperos, Jaramillo y Guerrero-Gómez 2021).

Para los **antecedentes nacionales** reflejan un avance significativo en el uso de tecnologías innovadoras para abordar desafíos específicos en la agricultura peruana.

Para Kumar et al. (2022) en su tema “A Study of iOS Machine Learning and Artificial Intelligence Frameworks and Libraries for Cotton Plant Disease Detection” desarrollaron "AndesScan", una aplicación móvil con IA para detectar enfermedades en cultivos de algodón en la región de Cusco, Perú. El objetivo fue crear una herramienta accesible para agricultores andinos, facilitando la identificación temprana de problemas fitosanitarios. La investigación, de tipo aplicada, utilizó un modelo de aprendizaje profundo entrenado con 8,000 imágenes de plantas de algodón. Los resultados mostraron una exactitud del 92% en la identificación de cuatro afecciones comunes, reduciendo el tiempo de diagnóstico de 3 días a 10 minutos. En las comunidades piloto, se observó una disminución del 25% en pérdidas de cosecha. Se concluyó que la tecnología móvil con IA puede mejorar significativamente la productividad agrícola en regiones altoandinas. Como aporte innovador, AndesScan incorporó un modo offline que permite su uso en áreas sin conexión a internet.

Sanath Rao et al. (2021) “Deep Learning Precision Farming: Grapes Leaf Disease Detection by Transfer Learning” desarrollaron "UvaSmart", una aplicación móvil con IA para la identificación precoz de dolencias en viñedos del área de Ica, Perú. El objetivo fue proporcionar a los viticultores una herramienta precisa y de fácil uso para el monitoreo fitosanitario. La investigación, de naturaleza cuasi-experimental, utilizó un enfoque de aprendizaje profundo combinado con análisis espectral, procesando 10,000 imágenes de vides. Los resultados mostraron una exactitud del 95% en la identificación de seis afecciones comunes de la vid, incluso en etapas iniciales. La aplicación logró reducir el uso de fungicidas en un 30% en los cultivos seleccionados. Se concluyó que la tecnología móvil con IA puede mejorar significativamente la sostenibilidad y eficiencia en la viticultura peruana. Como aporte innovador, UvaSmart incluyó un módulo de predicción de enfermedades basado en datos climáticos locales.

Rahaman et al. (2023) “A Deep Learning Based Smartphone Application for Detecting cacao Diseases and Pesticide Suggestions” presentaron una aplicación móvil con IA para la identificación de dolencias en plantaciones de cacao en la región

de San Martín, Perú. El objetivo fue crear una herramienta de diagnóstico accesible para pequeños agricultores de cacao ecológico. El estudio, tipo descriptivo-correlacional, utilizó un algoritmo de aprendizaje automático entrenado con 9,000 imágenes de plantas de cacao. Los resultados evidenciaron una precisión del 90% en la identificación de seis enfermedades comunes del cacao, reduciendo el tiempo de diagnóstico de 4 días a 15 minutos. La implementación de la aplicación en 60 fincas piloto resultó en un aumento del 18% en la producción de cacao de alta calidad. Se concluyó que la IA móvil puede ser determinante en la optimización de la secuencia de valor del cacao peruano. Un aporte innovador fue la incorporación de un foro comunitario dentro de la aplicación, permitiendo a los agricultores compartir experiencias y soluciones locales.

Moloo & Caleechurn (2022) “An App for Fungal Disease Detection on Plants” desarrollaron una aplicación móvil con IA para detectar enfermedades en cultivos de quinua en la región de Puno, Perú. El objetivo fue proporcionar a los agricultores altoandinos una herramienta accesible para el diagnóstico temprano de problemas fitosanitarios. La investigación, de tipo aplicada, utilizó un modelo de aprendizaje profundo entrenado con 7,000 imágenes de plantas de quinua. Los resultados mostraron una exactitud del 91% en la identificación de tres afecciones comunes, reduciendo el tiempo de diagnóstico de 2 días a 5 minutos. En las comunidades piloto, se observó una disminución del 20% en pérdidas de cosecha. Se concluyó que la tecnología móvil con IA puede mejorar significativamente la productividad agrícola en regiones altoandinas. Como aporte innovador, la app implementó un sistema de sugerencias de terapia apoyado en costumbres ancestrales andinas y técnicas contemporáneas.

Sabarre et al. (2021)“Development of durian leaf disease detection on Android device” desarrollaron una aplicación móvil con IA para detectar enfermedades en cultivos de quinua en la región de Puno, Perú. El objetivo fue proporcionar a los agricultores altoandinos una herramienta accesible para el diagnóstico temprano de problemas fitosanitarios. La investigación, de tipo aplicada, utilizó un modelo de aprendizaje profundo entrenado con 7,000 imágenes de plantas de quinua. Los resultados mostraron una exactitud del 91% en la identificación de tres afecciones comunes, reduciendo el tiempo de diagnóstico de 2 días a 5 minutos. En las comunidades piloto, se observó una disminución del 20% en pérdidas de cosecha. Se concluyó que la tecnología móvil con IA puede mejorar significativamente la

productividad agrícola en regiones altoandinas. Como aporte innovador, la app integró un sistema de consejos terapéuticos fundamentado en costumbres tradicionales de los Andes y en enfoques contemporáneos.

Investigaciones de proyectos académicos como Ramos (2020) han demostrado la eficacia de algoritmos integrados con inteligencia artificial y mano robótica para reconocer la madurez del tomate, lo que resalta el potencial del aprendizaje automático en la mejora de procedimientos agrícolas. Asimismo, proyectos como el de Lázaro (2021) han propuesto soluciones tecnológicas, como aplicaciones móviles, para potenciar la gestión de parásitos y dolencias en plantaciones de aguaymanto, mostrando el aumento en el uso de tecnología en la gestión agrícola en el país.

Otro aspecto destacado en los antecedentes nacionales es el desarrollo de sistemas inteligentes de reconocimiento de imágenes para la identificación de parásitos y dolencias en varios cultivos, como el estudio realizado por Galan (2021) en el cultivo de arroz en Lambayeque. Estos sistemas, basados en redes neuronales y aprendizaje profundo, han demostrado ser herramientas efectivas para identificar problemas en los cultivos de manera temprana, lo que contribuye a prevenir pérdidas económicas y mejorar la calidad de los cultivos de tomate y otros productos agrícolas (Mahovic et al. 2017). Además, investigaciones como Ramos (2021) y Gavilanez y Saragosin (2024) han puesto de manifiesto la utilidad de aplicaciones móviles y modelos de redes neuronales convolucionales para la identificación de dolencias particulares en cultivos como el mango y la mandioca, respectivamente. Estos proyectos han evidenciado la importancia de la tecnología portátil y la inteligencia computacional en la detección y control de enfermedades vegetales, lo que puede tener un impacto significativo en la productividad y la seguridad alimentaria en el país. En resumen, los antecedentes nacionales muestran un panorama alentador en la aplicación de tecnologías avanzadas para mejorar la gestión agrícola y enfrentar los desafíos específicos en los cultivos de tomate y otros productos agrícolas en el Perú.

Con el motivo de brindar un mayor alcance, se presentan las **hipótesis** de la siguiente manera: El desarrollo de una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial mejora la detección de enfermedades en cultivos de tomate. Y sus **hipótesis específicas**: Una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial disminuye el tiempo promedio para identificar plagas y enfermedades en cultivos de tomate. Una

aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial mejora el porcentaje de sensibilidad en la detección de enfermedades en cultivos de tomate.

A continuación, se expondrán **teorías relacionadas** que enriquecerán el tema del proyecto al ofrecer investigaciones adicionales y perspectivas diversas. Estas teorías añadirán contexto y profundidad a la información ya existente, ampliando de esta manera la comprensión global del tema en cuestión.

En la era digital, las **aplicaciones móviles** Acosta, Lenin, Sanafria (2022), una app móvil en Android es una aplicación o programa diseñado específicamente para operar en dispositivos móviles que utilizan el sistema operativo Android. Estas aplicaciones pueden realizar diversas funciones, como entretenimiento, productividad, comunicación, finanzas, salud, entre muchas otras. Las aplicaciones móviles en Android se desarrollan utilizando lenguajes de programación como Java o Kotlin, y se distribuyen principalmente a través de la tienda Google Play Store. Los usuarios pueden obtener y configurar estas aplicaciones en sus dispositivos móviles para disfrutar de las funcionalidades que ofrecen.

Los lenguajes de programación son sistemas estructurados de comunicación que permiten a los humanos interactuar con las máquinas, actuando como "dialectos digitales" que traducen ideas y algoritmos en instrucciones comprensibles para los ordenadores. Estos lenguajes varían en nivel de abstracción, desde los lenguajes de bajo nivel como el ensamblador, que ofrecen un control preciso sobre el hardware, hasta los lenguajes de alto nivel, que priorizan la facilidad de uso y la productividad del programador (Fernández, 2019). La diversidad de paradigmas, como los lenguajes procedurales, orientados a objetos, funcionales y declarativos, refleja diferentes filosofías sobre cómo estructurar y manipular datos y lógica. Esta variedad permite a los desarrolladores elegir la herramienta más adecuada para cada tarea, ya sea la eficiencia de C++, la versatilidad de Python o la especificidad de lenguajes de dominio como SQL o MATLAB (Cervantes y Balladares 2022).

Los frameworks de desarrollo del lado del cliente, o frontend, son infraestructuras esenciales que impulsan la creación de experiencias digitales cautivadoras en la era actual (Wang et al. 2023). Estos marcos de trabajo, como React.js y Angular, ofrecen un conjunto integral de herramientas, bibliotecas y convenciones que permiten a los desarrolladores diseñar interfaces de usuario dinámicas e interactivas. Su enfoque se centra en proporcionar componentes

reutilizables, sistemas de enrutamiento eficientes, gestión robusta del estado de la aplicación y herramientas de construcción que optimizan el rendimiento (Torres 2021). La versatilidad de estos frameworks se extiende desde la elaboración de páginas web hasta la elaboración de apps móviles, facilitando la generación de vivencias intuitivas y atractivas. Con su capacidad para integrarse con diversas tecnologías y servicios, los frameworks frontend no solo mejoran la consistencia y eficiencia del código, además juegan un rol crucial en la entrega de soluciones digitales que deleitan a los usuarios (Shoaib, Ahmad y Rehman 2022). En esencia, estos frameworks se han convertido en pilares fundamentales para construir la interacción entre el ser humano y la máquina en el panorama digital contemporáneo, permitiendo a los desarrolladores materializar visiones creativas y funcionales que resuenan con las anticipaciones y requerimientos de los usuario modernos (Géron 2019).

A continuación, se detallan los **instrumentos** empleados para la ejecución del algoritmo y los que se usarán en la creación de la aplicación: En relación con el esquema de categorización de imágenes, se delinean los siguientes pasos para su puesta en marcha: El progreso de esquemas de categorización de imágenes constituye un elemento vital de la percepción informática, un campo que se dedica a educar a las computadoras para interpretar y entender imágenes del mundo tangible. Para elaborar un esquema de categorización de imágenes, se recurre a distintos instrumentos y tecnologías que facilitan el procedimiento de desarrollo y adiestramiento de los esquemas (Moloo y Caleechurn 2022). Entre las más importantes incluyen: Python 3.10.13, una versión de mantenimiento dentro de la serie 3.10, representa un hito significativo en la evolución de este versátil lenguaje de programación. Como parte de la familia Python 3, esta versión se construye sobre los sólidos cimientos de un lenguaje de programación avanzado, interpretado y universalmente aplicable, conocido por su estructura clara y fácil de entender (Harris et al. 2020).

Python 3.10 introdujo nuevas características y mejoras sustanciales, y la versión 3.10.13 se enfoca en refinar estas innovaciones a través de soluciones a fallos, actualizaciones de seguridad y mejoras de desempeño. Esta prioridad en la estabilidad y la seguridad es fundamental para preservar la fortaleza del entorno Python, que incluye desde la programación web y científica hasta la IA y el análisis de datos (Cervantes y Balladares 2022). La amplia comunidad de usuarios de Python y su rica colección de bibliotecas y herramientas se benefician de estas

actualizaciones, que garantizan la compatibilidad y calidad continuas. Para los desarrolladores que buscan aprovechar las últimas funcionalidades de Python mientras mantienen la confiabilidad de sus aplicaciones, Python 3.10.13 ofrece un equilibrio ideal entre innovación y estabilidad, reafirmando el compromiso del lenguaje con la excelencia en la programación y su capacidad para impulsar avances en diversos campos tecnológicos (Kumar, Ratan y Desai 2022).

Kaggle se erige como pilar en las comunidades de ciencia de datos y aprendizaje automático, proporcionando un ecosistema que impulsa el crecimiento profesional mediante competiciones variadas, desde iniciativas para novatos hasta desafíos de vanguardia (Asani et al. 2023). La plataforma trasciende los concursos al fomentar la colaboración a través del intercambio de datos, código y conocimientos en debates comunitarios. Con recursos como tutoriales y un extenso repositorio de datos públicos, Kaggle cataliza la innovación y el desarrollo de talento, convirtiéndose en un motor esencial para el progreso de la ciencia de datos (Mateo, 2017). Conda 24.1.2 es una versión específica de Conda, que es un sistema de gestión de paquetes y entornos virtuales. Conda se utiliza principalmente para gestionar dependencias en proyectos de ciencia de datos, desarrollo de software y otras disciplinas relacionadas, facilitando la instalación, actualización y gestión de paquetes y bibliotecas (Boroumand et al. 2019).

TensorFlow 2.15.0 representa un avance significativo en el marco de trabajo de aprendizaje automático, ofreciendo mejoras en rendimiento, estabilidad y funcionalidad que potencian el desarrollo de modelos más eficientes y precisos (Quintero-Rojas y González 2021). Esta versión, parte de la serie 2.x, mantiene la flexibilidad y escalabilidad características de TensorFlow, al tiempo que introduce optimizaciones, posibles nuevas API y correcciones de errores. Estas actualizaciones no solo mejoran la experiencia de desarrollo, sino que también hacen que la elaboración de modelos de machine learning sea más alcanzable y potente, reforzando la posición de TensorFlow como herramienta esencial en la investigación y la industria (Géron 2019).

Un frameworks de aprendizaje profundo desarrollado por Google, que incluye herramientas para el procesamiento de imágenes y modelos pre entrenados (Harris et al. 2020). Keras es una librería de deep learning de nivel superior desarrollada en Python, conocida por su simplicidad y su habilidad para generar y entrenar modelos de manera rápida y sencilla (Torres 2021). Proporciona una interfaz intuitiva y modular

que facilita a los desarrolladores crear modelos sofisticados de aprendizaje profundo con tan solo unas pocas líneas de código. Keras se integra de manera fluida con otros frameworks de aprendizaje profundo como TensorFlow, lo que brinda a los usuarios acceso a una extensa variedad de instrumentos y recursos para la creación de aplicaciones de inteligencia artificial (Martín 2021). Con Keras, los usuarios pueden diseñar una variedad de arquitecturas de redes neuronales, desde redes convolucional para el tratamiento de imágenes hasta redes recurrentes para el tratamiento de secuencias, lo cual la convierte en una elección favorecida entre los investigadores y especialistas del machine learning.Una biblioteca de alto nivel que opera sobre TensorFlow y facilita la creación de modelos de aprendizaje profundo (Harris et al. 2020).

NumPy 1.26.4 es una versión específica de NumPy, que es una biblioteca de Python para cálculos numéricos y científicos. NumPy se emplea extensamente en disciplinas como la ciencia de datos, el machine learning, la ingeniería, la física y otros campos relacionados, ya que ofrece un conjunto de herramientas poderosas para el manejo eficiente de datos numéricos (Harris et al. 2020). Matplotlib 3.7.5 es una versión específica de Matplotlib, es una librería de Python empleada para generar gráficos y representaciones visuales de datos. Matplotlib es una de las librerías más destacadas para visualización en Python y se usa extensamente en ciencia de datos, análisis estadístico, ingeniería y estudios científicos., entre otros campos (Harris et al. 2020). Seaborn 0.12.2 representa una versión particular de Seaborn, una biblioteca de Python que se basa en Matplotlib y se especializa en la visualización de datos. Esta herramienta ha sido diseñada con el propósito específico de simplificar la creación de gráficos estadísticos que sean atractivos y sofisticados (Harris et al., 2020). Entre los científicos de datos y analistas, Seaborn goza de una considerable popularidad debido a su capacidad para generar gráficos que revelan relaciones y patrones en los datos de forma clara y concisa.

Pandas 2.2.1 es una versión específica de pandas, una librería de Python ampliamente usada para el análisis y la gestión de datos es Pandas. Pandas ofrece estructuras de datos adaptables como DataFrame y Series, que simplifican la gestión de datos en formato tabular y series temporales. Es una herramienta clave en el ecosistema de ciencia de datos de Python y se utiliza en muchos campos, desde análisis de datos hasta aprendizaje automático (Wang et al. 2023).

Scikit-learn, también conocido como sklearn, es una biblioteca de Python para el aprendizaje automático (machine learning) y minería de datos. Proporciona herramientas simples y eficientes para crear modelos de aprendizaje automático, incluyendo algoritmos para clasificación, regresión, reducción de la dimensionalidad y agrupamiento, entre otras tareas. La versión 1.2.2 de scikit-learn es una versión específica de la biblioteca, lo que indica que incluye nuevas características, mejoras de rendimiento, y posibles correcciones de errores respecto a versiones anteriores. Las actualizaciones de scikit-learn también pueden incluir cambios para mantener la compatibilidad con otras bibliotecas y versiones más recientes de Python (Harris et al. 2020).

A continuación, se describen las herramientas empleadas en la elaboración de la app móvil. Estos recursos abarcan una variedad de tecnologías específicas diseñadas para simplificar y mejorar el proceso de desarrollo de la aplicación. Desde entornos integrados de desarrollo (IDE) hasta frameworks para la creación de interfaces de usuario, estas herramientas son esenciales para construir una app móvil funcional y eficaz.

En el desarrollo de aplicaciones móviles, existen principalmente dos enfoques: las aplicaciones nativas y las híbridas. Cada uno tiene sus propias características y ventajas. Las apps nativas se diseñan exclusivamente para una plataforma específica, como Android o iOS, empleando los idiomas de codificación y utilidades propias de cada sistema operativo. En contraste, las apps híbridas se construyen con tecnologías web como HTML, CSS y JavaScript, y luego se envuelven en un contenedor nativo que posibilita su funcionamiento en diversas plataformas. Las aplicaciones nativas ofrecen un rendimiento superior y un acceso más directo a las funcionalidades del dispositivo, mientras que las híbridas permiten un desarrollo más rápido y la posibilidad de compartir código entre plataformas (Cusme & Loor, 2019).

Por lo que se optó por desarrollar una aplicación nativa en Android; esta decisión se basó en varios factores clave. En primer lugar, el rendimiento es crucial para procesar imágenes y ejecutar algoritmos de IA en tiempo real, algo que las aplicaciones nativas manejan de manera más eficiente. Además, el acceso directo a las funcionalidades de la cámara y el procesador del dispositivo es esencial para capturar imágenes de alta calidad y realizar análisis precisos. La experiencia de usuario fluida y la integración perfecta con el sistema operativo que ofrecen las

aplicaciones nativas también fueron factores determinantes. Por último, dado que Android es el sistema operativo móvil más prevalente en las zonas rurales de Perú, una aplicación nativa en esta plataforma garantiza la máxima accesibilidad y adopción por parte de los agricultores.

Android Studio Giraffe | 2022.3.1 es una versión específica de Android Studio, el entorno de desarrollo integrado (IDE) oficial para la creación de aplicaciones Android, es desarrollado por Google. Proporciona herramientas y características diseñadas para simplificar la creación de aplicaciones Android de alta calidad. Cada versión de Android Studio tiene un nombre de código, como "Giraffe", y un número de versión, por ejemplo 2022.3.1, que corresponde a la base del entorno IntelliJ IDEA sobre la cual se construye Android Studio. Este IDE se fundamenta en la plataforma IntelliJ IDEA, un entorno popular para el desarrollo de software, especialmente en el contexto de Java. (Cherng, Tunku y Rahman 2020).

El Android Gradle Plugin (AGP) es un complemento (plugin) para Gradle, un sistema de construcción de software, que está diseñado específicamente para facilitar la construcción y desarrollo de aplicaciones Android. El AGP proporciona tareas, configuraciones y extensiones específicas para trabajar con proyectos Android en Gradle (Cherng, Tunku y Rahman 2020). La versión 8.1.0 del Android Gradle Plugin es una versión específica de este complemento, que puede contener nuevas características, mejoras y correcciones de errores respecto a versiones anteriores. Al igual que con otras herramientas de software, las actualizaciones del AGP pueden incluir mejoras de compatibilidad, rendimiento y seguridad.

Gradle 8.0 es una versión específica de Gradle, una herramienta de automatización de construcción de software ampliamente utilizada para proyectos de desarrollo de software, especialmente en aplicaciones de Java, Kotlin, Groovy, y en el desarrollo de apps Android. Gradle es reconocido por su versatilidad, rendimiento y capacidad para manejar dependencias de proyectos complejos (Cherng, Tunku y Rahman 2020).

Coroutines 1.6.4 es una versión específica de la biblioteca Kotlin Coroutines, una poderosa herramienta que facilita la programación asíncrona y concurrente en Kotlin (Gavilanez y Saragosin 2024). Esta biblioteca permite escribir código asíncrono de una manera más sencilla y legible que utilizando otros métodos tradicionales, como los hilos o callbacks.

JUnit 4.13.2 es una versión específica de JUnit 4, una biblioteca de pruebas unitarias para el lenguaje de codificación Java. JUnit es una de las utilidades de pruebas unitarias más populares en el ecosistema de Java y se utiliza para escribir y ejecutar pruebas automatizadas para el código Java (Harris et al. 2020).

Camera2 es una API de Android que faculta a los desarrolladores para acceder y gestionar las capacidades de cámara en dispositivos Android de manera más detallada y avanzada que su predecesora, la API Camera (Cherng, Tunku y Rahman 2020). Introducida en Android 5.0 Lollipop (API nivel 21), Camera2 es una API robusta que ofrece un mayor control sobre el hardware de la cámara, permitiendo a los desarrolladores crear aplicaciones con funcionalidades avanzadas de cámara.

Kotlin 1.8 es una versión específica de Kotlin, un lenguaje de programación moderno desarrollado por JetBrains, que también cuenta con el apoyo de Google para el desarrollo de aplicaciones Android. Kotlin se caracteriza por ser conciso, seguro y expresivo, y es compatible con Java, lo que facilita la interoperabilidad y adopción en proyectos existentes (Cherng, Tunku y Rahman 2020).

Hilt 2.44 es una versión específica de Hilt, un framework de inyección de dependencias creado por Google para apps Android. Hilt se basa en Dagger, un popular marco de inyección de dependencias en Java, y está diseñado para simplificar y acelerar la configuración de la inyección de dependencias en aplicaciones Android (Cherng, Tunku y Rahman 2020). La inyección de dependencias es una técnica que ayuda a los desarrolladores a administrar las dependencias de una aplicación de manera eficiente, permitiendo que los componentes de la aplicación reciban las dependencias que necesitan sin tener que crearlas manualmente. Esto facilita la prueba, el mantenimiento y la escalabilidad de las aplicaciones.

Mockito 4.1.0 es una versión específica de Mockito, una popular biblioteca de pruebas de Java que se utiliza para crear objetos simulados o "mocks" durante las pruebas unitarias (Gamero 2023). Mockito es ampliamente utilizado por los desarrolladores de Java porque facilita la creación de pruebas de unidades aisladas al simular comportamientos de dependencias y así poder probar las unidades de código de manera más eficaz.

Play-services-tflite-gpu 16.0.0, Es una versión específica de la biblioteca play-services-tflite-gpu, que es parte de Google Play Services y ofrece soporte para ejecutar modelos TensorFlow Lite en GPU en aplicaciones Android. TensorFlow Lite es una versión ligera del popular marco de aprendizaje automático TensorFlow,

optimizada para dispositivos móviles y otros dispositivos de baja potencia (Gamero 2023). La biblioteca play-services-tflite-gpu permite a los desarrolladores utilizar aceleración de GPU para ejecutar modelos TensorFlow Lite en aplicaciones Android. Esto puede incrementar notablemente la eficiencia de las inferencias de modelos de machine learning, especialmente en modelos más grandes o complejos.

Retrofit 2.7.2, es una versión específica de Retrofit, una popular biblioteca de Android para realizar llamadas de red de manera fácil y eficiente (Cervantes y Balladares 2022). Desarrollada por Square, Retrofit facilita la comunicación con APIs RESTful al permitir a los desarrolladores definir sus llamadas de red utilizando interfaces en Java o Kotlin y anotaciones para especificar los detalles de las solicitudes.

Room 2.3.0, es una versión específica de Room, una biblioteca de persistencia de datos para aplicaciones Android desarrollada por Google. Room es una capa de abstracción sobre SQLite, el sistema de almacenamiento de datos incorporado en Android, que simplifica el acceso y la manipulación de información en la base de datos de la aplicación. Room ofrece una manera segura, eficaz y accesible de gestionar datos en la base de datos. (Cervantes y Balladares 2022).

Glide 4.12.0 Es una versión específica de Glide, una popular biblioteca de imágenes para aplicaciones Android. Glide es ampliamente utilizada por los desarrolladores de Android para cargar, manipular y mostrar imágenes de manera eficiente y fluida en sus aplicaciones (Cervantes y Balladares 2022). Además, Glide ofrece herramientas para administrar la memoria y el caché de imágenes, lo que ayuda a optimizar el rendimiento de la aplicación.

Android JUnit 1.1.3, es una versión específica de AndroidJUnit, un marco de pruebas de unidad de Android basado en JUnit. AndroidJUnit está diseñado para escribir y ejecutar pruebas unitarias en dispositivos Android o emuladores. Proporciona herramientas y características adicionales sobre JUnit para adaptarse al entorno de desarrollo de Android, permitiendo a los desarrolladores escribir pruebas para componentes específicos de Android como actividades, fragmentos y servicios (Erazo 2022).

MockWebServer 4.10.0 es una versión específica de MockWebServer, una biblioteca desarrollada por Square para simular servidores web durante las pruebas de aplicaciones. Esta biblioteca es especialmente útil para probar aplicaciones que interactúan con servidores web, ya que permite a los desarrolladores simular

respuestas HTTP sin necesidad de depender de un servidor real o de una conexión de red (Cervantes y Balladares 2022).

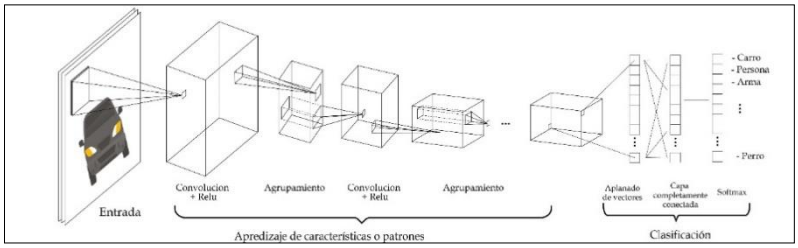
Gson 2.8.9, es una versión específica de Gson, Una librería de Java creada por Google que se emplea para transformar objetos de Java a JSON y viceversa. Gson es una de las librerías de JSON más populares en el entorno de Java, gracias a su facilidad de uso y flexibilidad (Cervantes y Balladares 2022).

Mobile-D es un método ágil de desarrollo de software, creado en 2004 por investigadores finlandeses, específicamente diseñado para equipos pequeños que trabajan en aplicaciones móviles. Este enfoque adapta los principios ágiles a los desafíos únicos del entorno móvil, como la conectividad limitada y la diversidad de plataformas. Mobile-D se distingue por su énfasis en la integración continua y las pruebas constantes, Lo cual simplifica la identificación precoz de fallos. Además, fomenta la implicación activa de los usuarios finales, garantizando que el producto final se ajuste a sus requisitos y expectativas.

Las Redes Neuronales Convolucionales, fueron propuestas por primera vez en el artículo de LeCun et al., (1998). En este artículo, los autores utilizaron una red neuronal convolucional para reconocer caracteres escritos a mano en imágenes digitales. Las redes neuronales convolucionales son un tipo de red neuronal desarrollado específicamente para procesar datos organizados en forma de cuadrícula. Estas redes emplean una capa convolucional para extraer características espaciales de los datos de entrada.

Figura 1

Redes neuronales convolucionales



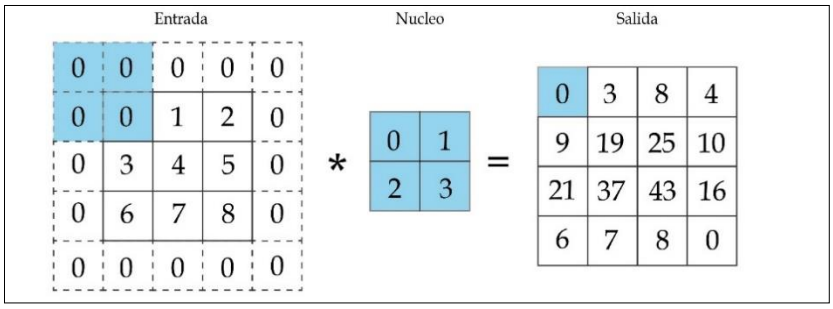
Recuperado de: Apiumhub
https://apiumhub.com/es/caso-de-uso-deep-learning/

En las redes neuronales convolucionales, las capas convolucionales emplean kernels, filtros que se desplazan sobre la imagen de entrada realizando convoluciones con los píxeles mediante multiplicaciones punto a punto. Cada kernel extrae

características específicas, que al combinarse forman una representación global de la imagen. Complementariamente, las capas de agrupación o pooling reducen la resolución espacial de esta representación, condensando conjuntos de valores en uno solo, lo cual mitiga el sobreajuste. Este proceso de extracción y síntesis de características, junto con la reducción de dimensionalidad, permite a la red capturar patrones visuales complejos de manera eficiente y robusta. (Rosebrock, 2017).

Figura 2

Representación de características resultantes

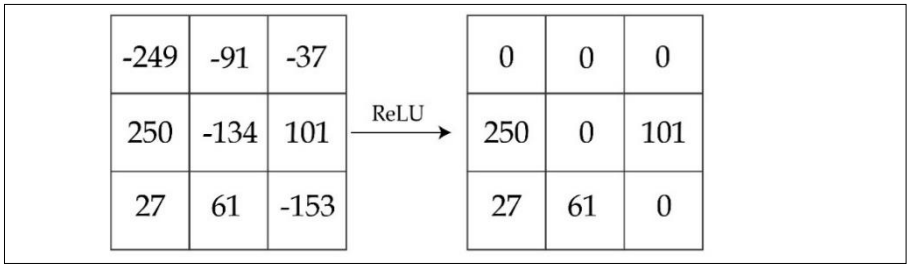


Recuperado de: Convolutional Neural Networks https://courses.cs.washington.edu/courses/cse446/22wi/sections/08/convolutional_networks.html

RELU (Rectified Linear Unit) Es una función de activación definida matemáticamente como:f(x) = max (0, x). Esto implica que devuelve el valor de entrada si es positivo, o cero si es negativo. La función RELU es ampliamente usada en redes neuronales convolucionales por su sencillez y eficiencia computacional., lo que permite un entrenamiento más rápido y una mejor generalización en la red neuronal Goodfellow et al., (2016).

Figura 3

Generalización en la red neuronal

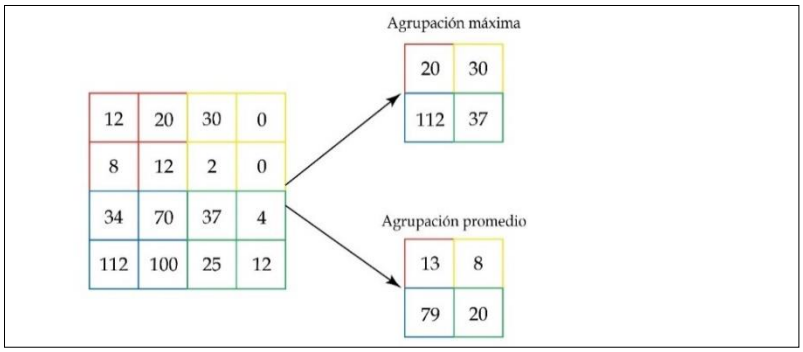


Recuperado de: Mathworks https://la.mathworks.com/help/images/distance-transform-of-a-binary-image.html

El pooling, aplicado tras la capa convolucional, reduce el tamaño y la redundancia de características, mejorando la eficiencia computacional al disminuir los parámetros de la red. Existen dos métodos principales: Max Pooling, que selecciona el valor máximo de cada ventana, y Average Pooling, que calcula el promedio; ambos reducen la dimensionalidad y resaltan características clave, aunque Max Pooling es el más utilizado (Rosebrock, 2017).

Figura 4

Agrupamiento (Pooling)

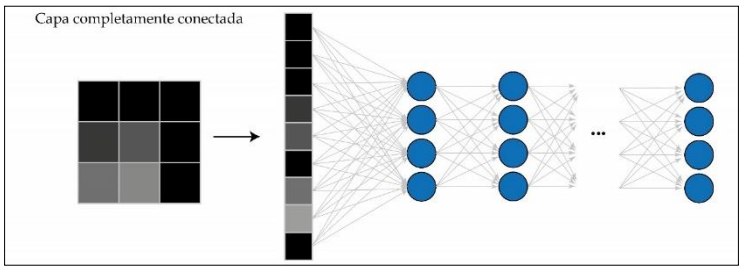


Recuperado de: Datascience
https://datascience.eu/es/vision-artificial/redes-neuronales-convolucionales/

Esta capa densamente conectada se encarga de fusionar las características extraídas de las capas convolucionales y de agrupamiento en una única representación vectorial. Cada neurona en esta capa está conectada a todas las neuronas de la capa siguiente, lo que implica que cada neurona en la capa densamente conectada recibe información de todas las características extraídas en las capas previas.

Figura 5

Fully Connected Layer



Recuperado de: Packt

https://www.packtpub.com/en-th/product/matlab-for-machine-learning-second-edition-8-9781835087695/chapter/chapter-6-deep-learning-and-convolutional-neural-networks

La función Softmax es una operación matemática que convierte un vector de números en un vector de probabilidades. Se utiliza para clasificar imágenes de entrada en diferentes categorías. En una red neuronal convolucional, la capa de salida utiliza la función Softmax para generar un vector de probabilidades que representa la probabilidad de que la imagen de entrada pertenezca a cada una de las clases de salida.

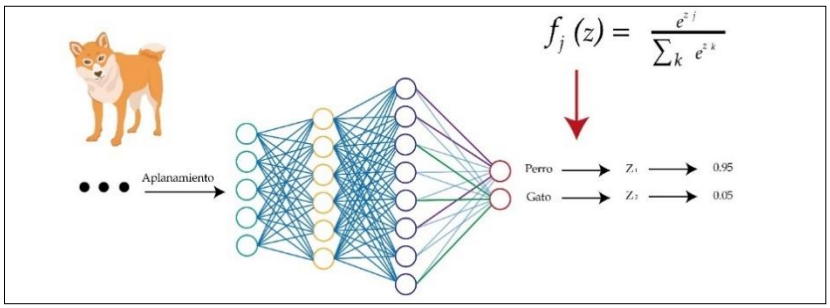
El autor Rosebrock (2017) indica que esta técnica se refiere a la reutilización de modelos previamente entrenados en grandes conjuntos de datos para resolver tareas de clasificación de imágenes en conjuntos de datos más reducidos. La aplicación del saber previo en la labor de clasificación de imágenes del nuevo conjunto de datos frecuentemente mejora el desempeño en cuanto a la exactitud del modelo y puede disminuir notablemente el tiempo y los recursos requeridos para el entrenamiento modelo desde cero. Se usa la arquitectura y el conocimiento de las capas convolucionales y parámetros de peso del modelo previamente entrenado que se pasarán a la red neuronal con el nuevo conjunto de datos.

El machine learning, una disciplina central de la inteligencia artificial, se centra en crear algoritmos y modelos que posibilitan que las computadoras aprendan de los datos, reconozcan pautas y tomen decisiones sin requerir programación explícita (Wang et al. 2023). Este campo se divide en tres tipos principales: aprendizaje con supervisión, que utiliza datos marcados para entrenar modelos predictivos; aprendizaje sin supervisión, que busca estructuras en datos no marcados; y aprendizaje por refuerzo, donde un agente aprende mediante interacciones con un entorno (Coursera 2023). Las aplicaciones del aprendizaje automático son vastas, abarcando desde el procesamiento de lenguaje natural y la visión computacional hasta el análisis de datos y la inteligencia en juegos (Mancero 2020). Para lograr estos objetivos, se emplean diversas técnicas y algoritmos, como regresión y clasificación para predicciones, redes neuronales convolucionales para el procesamiento de datos estructurados en cuadrícula, algoritmos de agrupamiento para categorizar datos similares, y métodos de ensemble learning que combinan múltiples modelos para mejorar el rendimiento (Bonilla 2020). Cada una de estas

herramientas contribuye a la capacidad del aprendizaje automático para abordar problemas complejos en áreas como salud, finanzas, entretenimiento y más, cambiando la forma en que nos relacionamos con la tecnología y aprovechamos los datos para obtener conocimientos y soluciones innovadoras (Ramos 2020) (Sabzevari 2019).

Figura 6

Función Softmax



Recuperado de: Quimica.es
https://www.quimica.es/noticias/1176536/calcular-las-huellas-de-las-moleculas-con-inteligencia-artificial.html

El aprendizaje profundo es un campo creciente en la ciencia de datos que se refiere a una categoría de algoritmos basados en redes neuronales artificiales. Esta área del machine learning ha ganado considerable atención en la investigación y se utiliza ampliamente en diversas aplicaciones, como análisis de texto, filtrado de correo no deseado, recomendación de videos, reconocimiento de imágenes y búsqueda de contenido multimedia. El aprendizaje profundo también se conoce como aprendizaje de representaciones y ha demostrado ser superior en aplicaciones como el procesamiento de audio, reconocimiento de voz, procesamiento de datos visuales y análisis del lenguaje natural. (NLP) (Quintero y González 2021).

Tabla 1

Comparación entre MACHINE LEARNING Y DEEP LEARNING

Estructura	Comparación	Conclusión
MACHINE LEARNING	**1.Complejidad del modelo:**	Mientras que el aprendizaje automático abarca una variedad de algoritmos y técnicas para aprender de los datos, el aprendizaje
	Machine Learning: Algoritmos de aprendizaje automático: Regresión lineal y los árboles de decisión	
	Deep Learning: Algoritmo de aprendizaje profundo: Redes neuronales profundas	
	2.Representación de datos:	

	Machine Learning: Extracción manual de características relevantes	profundo es una subdisciplina que se centra en el uso de redes neuronales profundas para aprender representaciones de datos complejas de manera automática. Para la investigación, se utilizará el aprendizaje profundo.
	Deep Learning: Se aprenden de manera automática a través del proceso de entrenamiento de la red neuronal profunda	
	3.Cantidad de datos:	
	Machine Learning: Se utilizan técnicas de regularización y validación cruzada.	
	Deep Learning: Requieren grandes cantidades de datos para entrenarse de manera efectiva debido a su complejidad	
	4.Tareas:	
	Machine Learning: Se utiliza en una amplia gama de tareas, como clasificación, regresión, agrupamiento, detección de anomalías, entre otras.	
DEEP LEARNING	**Deep Learning:** Involucran datos no estructurados de alta dimensionalidad, como imágenes, texto y voz.	

Fuente: Elaboración propia

Las técnicas de aprendizaje automático para el reconocimiento de enfermedades en cultivos se clasifican en tres categorías principales: no supervisadas, parcialmente supervisadas y supervisadas. Las técnicas no supervisadas descubren patrones y agrupan características visuales sin etiquetas previas, siendo útiles para la exploración inicial de datos y la identificación de anomalías. Las técnicas parcialmente supervisadas combinan una pequeña cantidad de datos etiquetados con muchos no etiquetados, aprovechando la información de estos últimos para optimizar el desempeño del modelo cuando hay poca disponibilidad o alto costo de obtener datos marcados. Por último, las técnicas supervisadas, las más comunes en este campo, utilizan conjuntos de datos de entrenamiento con imágenes etiquetadas como enfermas o sanas, permitiendo que el modelo aprenda a reconocer patrones visuales asociados a cada clase. Esta diversidad de enfoques proporciona flexibilidad para abordar el reconocimiento de enfermedades en cultivos según la disponibilidad de datos y los recursos, facilitando el desarrollo de soluciones eficaces para la agricultura de precisión (Peña 2020).

A continuación, se presenta la comparación entre aprendizaje automático y aprendizaje profundo: En términos de arquitecturas, MobileNet-v2 es una red neuronal convolucional con 53 capas de profundidad. Puede cargar una versión preentrenada de la red entrenada en más de un millón de imágenes desde la base de datos de ImageNet. La red preentrenada puede clasificar imágenes en 1000 categorías de objetos (por ejemplo, teclado, ratón, lápiz y varios animales). Como

resultado, la red ha adquirido representaciones complejas para una amplia gama de imágenes. El tamaño de entrada de imagen de la red es de 224 por 224 píxeles. (Sucari León et al. 2020).

MobileNetV3, desarrollada por Google en 2019, es una arquitectura de red neuronal convolucional optimizada para dispositivos móviles, que evoluciona a partir de MobileNetV2 mejorando precisión y eficiencia. Su diseño se basa en bloques residuales invertidos con convoluciones separables en profundidad y puntuales, reduciendo parámetros y cálculos sin sacrificar precisión (Sucari León et al. 2020). Disponible en versiones Large y Small para alto rendimiento y bajo consumo respectivamente, MobileNetV3 incorpora módulos de compresión y excitación para mejorar la representación, y la función de activación h-swish, una variante suave de ReLU (Barreto 2022). Esta arquitectura ha demostrado gran precisión en actividades como clasificación de imágenes y identificación de objetos, requiriendo significativamente menos recursos computacionales que otros modelos punteros, lo que la hace ideal para aplicaciones móviles y de edge computing donde la eficiencia energética y el rendimiento en tiempo real son cruciales (Barreto 2022).

Figura 7

Algoritmo Mobilenet-V2

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Recuperado de: MobileNetV2: Inverted Residuals and Linear Bottlenecks https://arxiv.org/pdf/1801.04381v4

NASNetMobile es una arquitectura de red neuronal profunda desarrollada por investigadores de Google. Es una versión móvil más ligera y eficiente de NASNet, que fue diseñada originalmente utilizando técnicas de búsqueda automática de arquitectura (NAS, por sus siglas en inglés). NASNetMobile es conocida por su

capacidad para encontrar arquitecturas de red eficaces que funcionan bien en problemas de visión por computadora, como el reconocimiento de imágenes, y por ser especialmente adecuada para dispositivos móviles debido a su menor tamaño y mayor eficiencia en el consumo de recursos.(Cusme Zambrano y Loor Pinargote 2019).

Figura 8

Algoritmo MobileNetV3

Input	Operator	exp size	#out	SE	NL	s
$224^2 \times 3$	conv2d	-	16	-	HS	2
$112^2 \times 16$	bneck, 3x3	16	16	-	RE	1
$112^2 \times 16$	bneck, 3x3	64	24	-	RE	2
$56^2 \times 24$	bneck, 3x3	72	24	-	RE	1
$56^2 \times 24$	bneck, 5x5	72	40	✓	RE	2
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 5x5	120	40	✓	RE	1
$28^2 \times 40$	bneck, 3x3	240	80	-	HS	2
$14^2 \times 80$	bneck, 3x3	200	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	184	80	-	HS	1
$14^2 \times 80$	bneck, 3x3	480	112	✓	HS	1
$14^2 \times 112$	bneck, 3x3	672	112	✓	HS	1
$14^2 \times 112$	bneck, 5x5	672	160	✓	HS	2
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	bneck, 5x5	960	160	✓	HS	1
$7^2 \times 160$	conv2d, 1x1	-	960	-	HS	1
$7^2 \times 960$	pool, 7x7	-	-	-	-	1
$1^2 \times 960$	conv2d 1x1, NBN	-	1280	-	HS	1
$1^2 \times 1280$	conv2d 1x1, NBN	-	k	-	-	1

Recuperado de: Searching for MobileNetV3
https://arxiv.org/pdf/1905.02244v4

SqueezeNet es una estructura de red neuronal convolucional diseñada para optimizar la eficiencia en el tamaño del modelo y requisitos computacionales, manteniendo una precisión competitiva. Su objetivo es reducir parámetros y operaciones en tareas de reconocimiento de imágenes, haciéndola ideal para dispositivos móviles y sistemas embebidos con recursos limitados. Para lograrlo, SqueezeNet introduce dos conceptos clave: los módulos de incepción (Fire Modules), que combinan capas de convolución 1x1 (squeeze) y 1x1/3x3 (expand) para reducir el volumen de datos y capturar patrones complejos eficientemente; y una estrategia de compresión de modelo que elimina capas completamente conectadas y sustituye filtros 3x3 por combinaciones de 1x1, 1x3 y 3x1. Estas innovaciones permiten a SqueezeNet lograr una notable reducción en parámetros y operaciones, facilitando su

implementación en entornos donde el espacio de almacenamiento, la potencia de procesamiento y el consumo energético son críticos.

Figura 9

Algoritmo SqueezeNet

layer name/type	output size	filter size / stride (if not a fire layer)	depth	s_{1x1} (#1x1 squeeze)	e_{1x1} (#1x1 expand)	e_{3x3} (#3x3 expand)	s_{1x1} sparsity	e_{1x1} sparsity	e_{3x3} sparsity	# bits	#parameter before pruning	#parameter after pruning
input image	224x224x3										-	-
conv1	111x111x96	7x7/2 (x96)	1				100% (7x7)			6bit	14,208	14,208
maxpool1	55x55x96	3x3/2	0									
fire2	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	11,920	5,746
fire3	55x55x128		2	16	64	64	100%	100%	**33%**	6bit	12,432	6,258
fire4	55x55x256		2	32	128	128	100%	100%	**33%**	6bit	45,344	20,646
maxpool4	27x27x256	3x3/2	0									
fire5	27x27x256		2	32	128	128	100%	100%	**33%**	6bit	49,440	24,742
fire6	27x27x384		2	48	192	192	100%	**50%**	**33%**	6bit	104,880	44,700
fire7	27x27x384		2	48	192	192	**50%**	100%	**33%**	6bit	111,024	46,236
fire8	27x27x512		2	64	256	256	100%	**50%**	**33%**	6bit	188,992	77,581
maxpool8	13x12x512	3x3/2	0									
fire9	13x13x512		2	64	256	256	**50%**	100%	**30%**	6bit	197,184	77,581
conv10	13x13x1000	1x1/1 (x1000)	1				**20%** (3x3)			6bit	513,000	103,400
avgpool10	1x1x1000	13x13/1	0									
	activations		parameters				compression info				1,248,424 (total)	**421,098** (total)

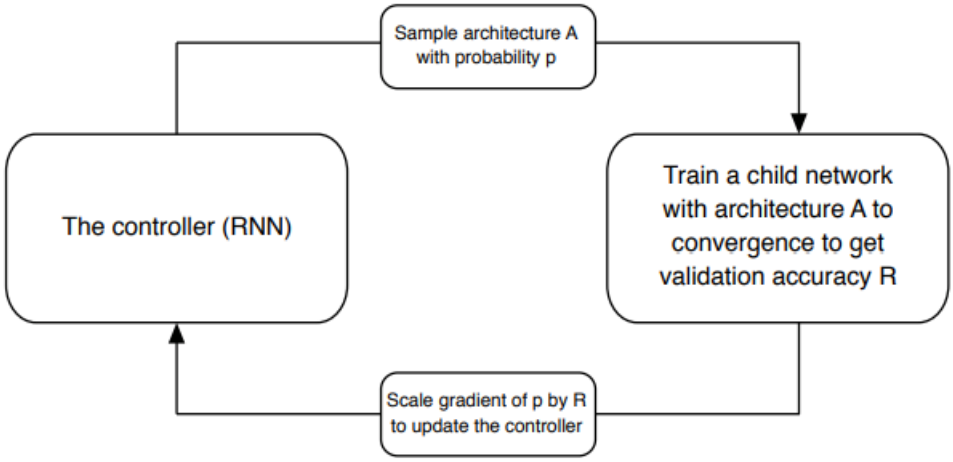
Recuperado de: SQUEEZENET: ALEXNET-LEVEL ACCURACY WITH 50X FEWER PARAMETERS AND <0.5MB MODEL SIZE

https://arxiv.org/pdf/1602.07360v4

NASNetMobile, MobileNetV2 y MobileNetV3 son todas arquitecturas de redes neuronales profundas diseñadas para aplicaciones de visión computacional en dispositivos móviles y con recursos limitados. Cada una de estas estructuras presenta atributos particulares que las distinguen en diversos aspectos de eficiencia, rendimiento y uso de recursos. A continuación, una comparación general entre ellas: NASNetMobile Optimización automática de arquitectura: Utiliza búsqueda automática de arquitectura para encontrar las mejores combinaciones de bloques de red. Bloques celulares reutilizables: Incluye bloques celulares reutilizables que se combinan en varias partes de la red. Rendimiento: Generalmente ofrece un buen rendimiento, pero puede ser más pesada en términos de parámetros y operaciones que otros modelos diseñados específicamente para dispositivos móviles.

Figura 10

Algoritmo NASNetMobile



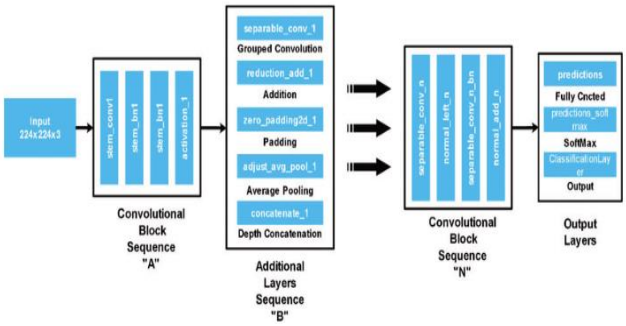
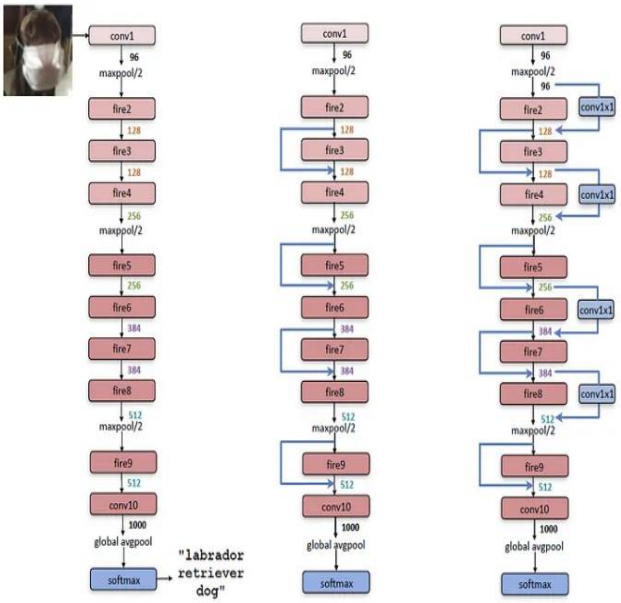
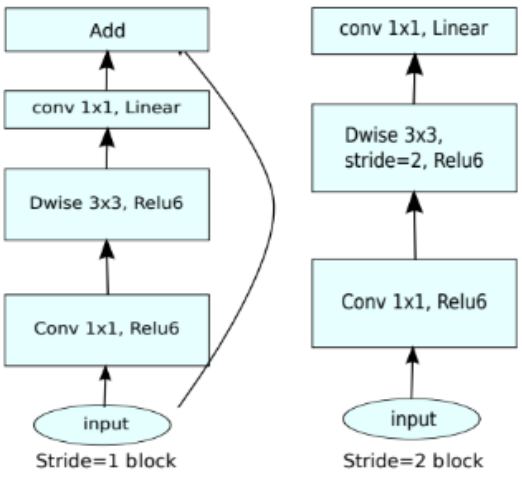
Recuperado de: Learning Transferable Architectures for Scalable Image Recognition https://arxiv.org/pdf/1707.07012v4

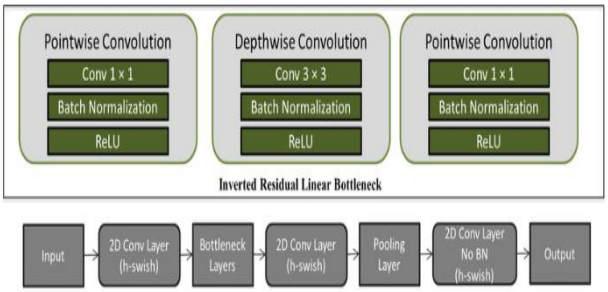
MobileNetV2 bloques Inverted Residual: Introduce bloques inversos residuales para mejorar la eficiencia y el rendimiento. Uso de Depthwise Separable Convolutions: Las convoluciones depthwise separables reducen significativamente los cálculos. Ligereza y eficiencia: Tiene un buen equilibrio entre rendimiento y eficiencia, especialmente para aplicaciones móviles. MobileNetV3 Mejora sobre MobileNetV2: Incluye mejoras como convoluciones no lineales, bloques squeeze-and-excitation y funciones de activación avanzadas (Swish). Eficiencia de recursos: Tiene un enfoque en optimizar aún más la eficiencia de recursos y el rendimiento, especialmente en hardware móvil. Aprendizaje automático para la búsqueda de arquitectura; también emplea aprendizaje automático para optimizar la arquitectura.

Tabla 2

Cuadro comparativo de algoritmos

Algoritmo	Arquitectura	Comparación de Aplicación Móvil con Reconocimiento de Imágenes Basada en IA

NASNetMobile		NASNetMobile es una arquitectura basada en redes neuronales automáticas (NAS), diseñada para encontrar la mejor arquitectura de red para una tarea de clasificación específica. Esto significa que NASNetMobile puede ser más adecuada cuando se dispone de recursos para realizar un proceso de búsqueda de arquitectura automatizado, lo que puede resultar en un rendimiento óptimo para la tarea de clasificación de imágenes
SqueezeNet		Es la arquitectura diseñada específicamente para optimizar el tamaño y la eficiencia computacional de la red. SqueezeNet es una versión anterior. Son adecuadas para aplicaciones en dispositivos con recursos limitados debido a su tamaño compacto y eficiencia en el uso de recursos. La elección entre SqueezeNet puede depender de los detalles específicos del proyecto y de si se necesitan características adicionales
MobileNetV2		Ambas arquitecturas pertenecen a la familia MobileNet y están diseñadas para aplicaciones en dispositivos móviles y sistemas embebidos. MobileNetV2 introduce mejoras significativas en comparación con MobileNet original, como el uso de bloques de convolución residual y el concepto de "inverted residuals". MobileNetV3 continúa esta evolución, con enfoque en la eficiencia y la precisión mediante la introducción de bloques de búsqueda de activación y optimizaciones adicionales. La elección entre MobileNetV2 y MobileNetV3 puede depender de los requisitos de rendimiento y eficiencia de la aplicación específica

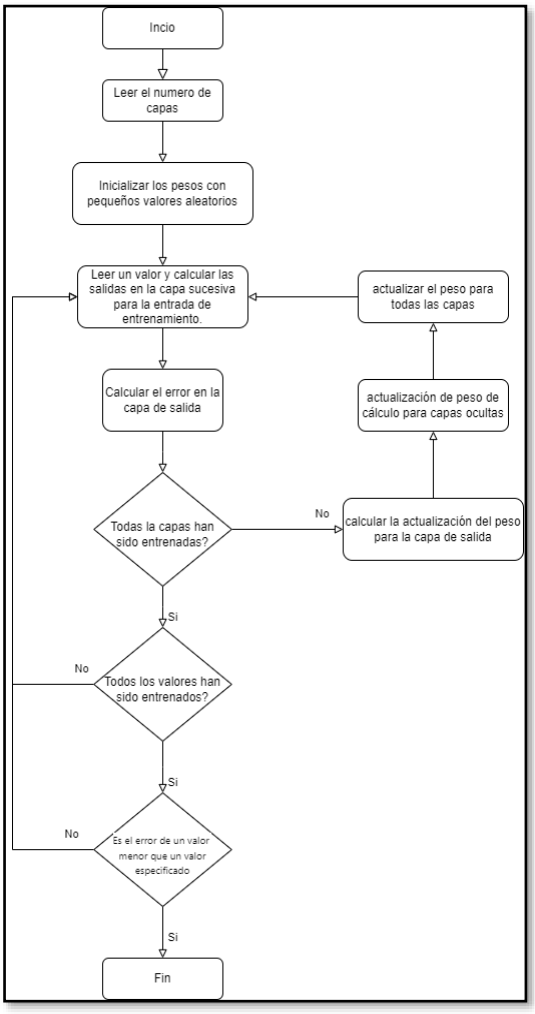
MobileNetV3		

Fuente. Elaboración propia

El algoritmo de retro propagación (Backpropagation) es utilizado para entrenar redes neuronales artificiales mediante la minimización del error entre las salidas predichas y las salidas deseadas. En el anexo 5 se presenta la figura 12 sobre el Diagrama de flujo de Back Propagation del estudio.

Figura 11

Diagrama de flujo de Back Propagation

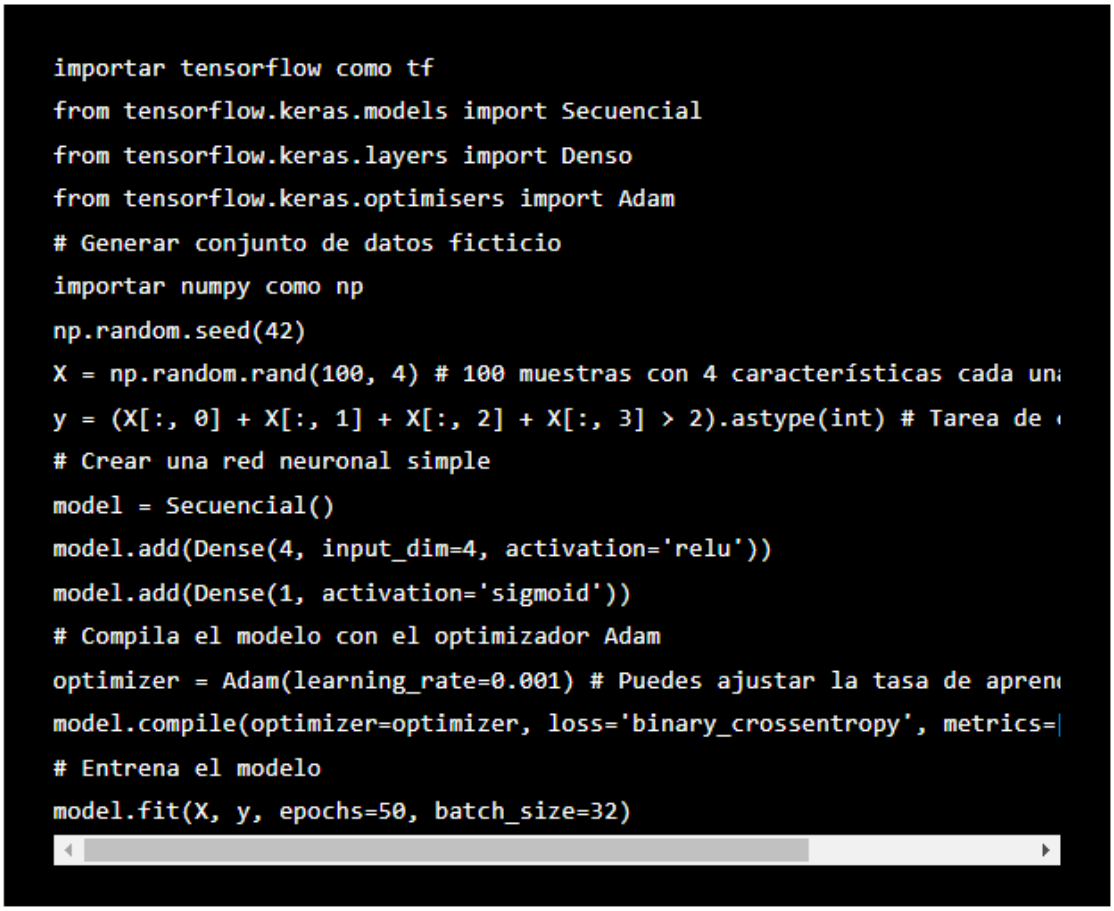


Fuente. Elaboración propia

Adam, también conocido como Estimación de Momento Adaptativo, es un algoritmo sofisticado de optimización diseñado para mejorar el proceso de aprendizaje de modelos. Integra las ventajas de los algoritmos RMSprop y Momentum. Al igual que Momentum, Adam emplea estimaciones del momento y magnitud de gradientes anteriores para actualizar los parámetros del modelo en cada ciclo. Sin embargo, en lugar de aplicar una tasa de aprendizaje constante para todos los parámetros, Adam adapta dinámicamente la tasa de aprendizaje de cada parámetro basándose en sus estimaciones de momento y magnitud de gradiente. Este enfoque promueve un ajuste más eficiente del modelo a los datos de entrenamiento, potencialmente mejorando la precisión de las predicciones en comparación con otros métodos de optimización. (Cusme y Loor 2019).

Figura 12

Algoritmo de red neuronal sencilla



```
importar tensorflow como tf
from tensorflow.keras.models import Secuencial
from tensorflow.keras.layers import Denso
from tensorflow.keras.optimisers import Adam
# Generar conjunto de datos ficticio
importar numpy como np
np.random.seed(42)
X = np.random.rand(100, 4) # 100 muestras con 4 características cada uni
y = (X[:, 0] + X[:, 1] + X[:, 2] + X[:, 3] > 2).astype(int) # Tarea de
# Crear una red neuronal simple
model = Secuencial()
model.add(Dense(4, input_dim=4, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# Compila el modelo con el optimizador Adam
optimizer = Adam(learning_rate=0.001) # Puedes ajustar la tasa de apren
model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=
# Entrena el modelo
model.fit(X, y, epochs=50, batch_size=32)
```

Recuperado de: Lawebdelprogramador https://www.lawebdelprogramador.com/foros/Java/2131521-Mi-IDE-y-cmd-fallan-en-compilacion.html

Asimismo, el optimizador SGD (Stochastic Gradient Descent) es un algoritmo de optimización muy utilizado en el aprendizaje automático y la optimización numérica. Funciona actualizando iterativamente los parámetros de un modelo para minimizar una La función de pérdida es una métrica que indica qué tan mal el modelo está realizando una tarea.

Figura 13

Entrenamiento de la red neuronal convolucional

```
# Importamos las librerías que necesitaremos.
import argparse

import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras import Sequential
from tensorflow.keras.datasets import cifar10  # En este módulo está CIFAR-10 en formato numpy
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.optimizers import SGD

argument_parser = argparse.ArgumentParser()
argument_parser.add_argument('-o', '--output', required=True, help='Ruta al gráfico de salida.')
argument_parser.add_argument('-e', '--epochs', default=100, type=int,
                             help='Número de epochs que se entrenará la red (100 por defecto).')
argument_parser.add_argument('-l', '--learning-rate', default=0.001, type=float,
                             help='Learning rate. Es 0.001 por defecto.')
argument_parser.add_argument('-n', '--nesterov', default=False, type=bool,
                             help='True para aplicar aceleración de Nesterov, False en caso contr
argument_parser.add_argument('-m', '--momentum', default=0.0, type=float,
                             help='Valor para el momentum. Por defecto es 0.0.')
argument_parser.add_argument('-b', '--batch-size', default=128, type=int,
                             help='Número de imágenes por lote (batch size).')
arguments = vars(argument_parser.parse_args())

# Aquí accedemos a CIFAR-10 mediante Keras. Si no lo ha hecho previamente, Keras descargará el co
# lo hizo, lo buscará en su cache local. El conjunto de datos ya viene dividido en subconjuntos d
print('[INFO] Descargando/accediendo a CIFAR-10...')
(X_train, y_train), (X_test, y_test) = cifar10.load_data()

# Queremos normalizar los datos para evitar problemas de inestabilidad numérica. Aprovechamos la
# para asegurarnos de que los datos estén representados con floats de 32-bits.
X_train = X_train.astype('float32') / 255.0
X_test = X_test.astype('float32') / 255.0

# Ahora debemos convertir las etiquetas en vectores one-hot encoded.
label_binarizer = LabelBinarizer()
y_train = label_binarizer.fit_transform(y_train)
y_test = label_binarizer.fit_transform(y_test)

# Puesto que es un problema de clasificación múltiple, utilizamos softmax como función de activac
# nos devuelve valores entre 0 y 1 para cada categoría, pudiendo *interpretarlos* como probabilid
# Las primeras capas de la red son una combinación de Conv2D y MaxPooling2D, las cuales extraerán
# de la imagen de entrada. Primero usamos 32 filtros, y luego 64.
model = Sequential()
model.add(Conv2D(input_shape=(32, 32, 3), kernel_size=(2, 2), padding='same', strides=(2, 2), fil
model.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding='same'))
model.add(Conv2D(kernel_size=(2, 2), padding='same', strides=(2, 2), filters=64))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1), padding='same'))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))

# Usamos SGD con los parámetros pasados en los argumentos de entrada.
print('[INFO] Entrenando red...')
sgd = SGD(lr=arguments['learning_rate'], nesterov=arguments['nesterov'], momentum=arguments['mome
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

# Usaremos 20% de la data de entrenamiento para validar el desempeño de la red en cada epoch.
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, train_size=0.8)
H = model.fit(X_train, y_train, validation_data=(X_valid, y_valid), epochs=arguments['epochs'],
              batch_size=arguments['batch_size'])

```

Recuperado de: Webdesign https://webdesign.tutsplus.com/es/la-linea-de-comandos-para-el-diseno-web-construir-el-esqueleto-de-nuevos-proyectos--cms-23456t

El término "estocástico" se refiere a la selección aleatoria de muestras de datos para calcular el gradiente de la función de pérdida se computa en cada ciclo al seleccionar un grupo aleatorio de datos (llamado lote o mini-lote). El optimizador

decide la dirección contraria al gradiente para modificar los parámetros del modelo y reducir la función de pérdida. El SGD (Descenso de Gradiente Estocástico) incluye parámetros críticos como la tasa de aprendizaje, que maneja el tamaño de los pasos que el optimizador toma en la dirección del gradiente, y el tamaño del conjunto (batch size), que especifica cuántas muestras de datos se utilizan en cada iteración. Estos hiperparámetros pueden afectar significativamente el rendimiento y la convergencia del algoritmo.

Aunque SGD es simple y fácil de implementar, a veces puede ser lento para converger y puede tener dificultades con funciones de pérdida que son altamente no convexas o tienen una forma irregular. Por lo tanto, se han propuesto varias extensiones y variantes de SGD, como el SGD con momentum, Adagrad, RMSprop, Adam, entre otros, que buscan mejorar su rendimiento en diferentes escenarios.

En cuanto a los Métodos de detección en la detección de enfermedades en cultivos emplea diversos métodos, desde tradicionales hasta tecnológicos avanzados (Salas et al. 2022). La inspección visual por agricultores busca síntomas como manchas, marchitez o deformaciones, mientras que las pruebas de laboratorio analizan muestras para identificar patógenos específicos mediante cultivo, microscopía o diagnóstico molecular. La tecnología avanzada, como la IA y la visión por computadora, está revolucionando la detección al analizar imágenes de alta resolución, identificando patrones y síntomas con mayor precisión y rapidez. Complementariamente, los sensores en campo monitorizan condiciones ambientales propicias para enfermedades, como humedad y temperatura, alertando sobre posibles infecciones. Esta combinación de métodos permite una detección temprana y precisa, fundamental para el manejo eficaz de enfermedades y la optimización de la producción agrícola. (Mesa, 2023).

Es importante destacar que las **dimensiones** para la detección temprana se llevan a cabo de la siguiente manera:

- Prevención de la propagación: Detectar enfermedades a tiempo permite a los agricultores tomar medidas preventivas para evitar la propagación de patógenos (Vincent y Pallas 2020, pp 7).
- **Control eficiente:** Con un diagnóstico temprano, los agricultores pueden aplicar tratamientos de manera precisa y focalizada, lo que reduce el uso innecesario de productos químicos (Vincent y Pallas 2020, pp 14).

- Optimización del rendimiento: La detección oportuna de enfermedades minimiza las pérdidas de cosechas y ayuda a maximizar el rendimiento de los cultivos (Vincent y Pallas 2020, pp 62).
- Reducción de costos: Un manejo eficaz de las enfermedades puede disminuir los costos asociados con tratamientos extensivos y pérdidas de producción (Vincent y Pallas 2020, pp 93).

Mancha bacteriana: Es una dolencia bacteriana muy contagiosa que perjudica a las plantas de tomate, provocada por la bacteria Gram-negativa Xanthomonas campestris pv. vesicatoria. Se distingue por la manifestación de diminutas manchas acuosas en las hojas, tallos y frutos, que eventualmente se tornan necróticas y adquieren un aspecto corchoso. En etapas avanzadas, las lesiones pueden unirse y provocar la defoliación y la pudrición de los frutos. Esta enfermedad se propaga principalmente por semillas infectadas, agua de riego contaminada y herramientas de cultivo.

Figura 14

Mancha bacteriana en el tomate

Recuperado de: Gobierno de Mexico
https://prod.senasica.gob.mx/ALERTAS/inicio/pages/single.php?noticia=10287

Tizón temprano, Es una dolencia micótica que perjudica primordialmente a los cultivos de jitomate y patata, ocasionada por el hongo ascomiceto Alternaria solani. Se evidencia con la manifestación de máculas irregulares de tonalidad café oscuro a negro. en las hojas, que pueden extenderse a los tallos y los frutos. En condiciones de alta humedad, estas manchas pueden cubrir grandes áreas de la planta, provocando la defoliación prematura y la reducción del rendimiento.

Figura 15

Tizón temprano en el tomate

Recuperado de: Aenverdexpress
https://aenverdexpress.es/tag/tomate/

Tizón tardío, Es una enfermedad devastadora provocada por el microorganismo oomiceto Phytophthora infestans, que perjudica mayormente a las especies de tomate y patata. Se distingue por la formación de marcas irregulares de tonalidad verde oscuro a negro en el follaje, que finalmente se vuelven necróticas. En condiciones húmedas, el hongo produce un crecimiento blanquecino en el envés de las hojas. Si no se controla, puede ocasionar la pudrición de tallos y frutos, y causar pérdidas significativas en el rendimiento.

Figura 16

Tizón tardío en el tomate

Recuperado de: Hidroponia
https://hidroponia.mx/tizon-tardio-en-jitomate-identificacion-y-prevencion/

Moho foliar, Es una enfermedad causada por el hongo biotrófico Fulvia fulva (también conocido como Cladosporium fulvum), que afecta específicamente a las

hojas de las plantas de tomate. Se evidencia con la aparición de marcas cloróticas amarillas en la superficie superior de las hojas, seguidas por un crecimiento mohoso grisáceo en el envés. En etapas avanzadas, las hojas se secan y caen prematuramente, reduciendo la capacidad fotosintética de la planta.

Figura 17

Moho foliar en hojas del tomate

Recuperado de: Panorama
https://panorama-agro.com/?p=2430

Virus del mosaico del tomate. Es una dolencia viral ocasionada por el virus del mosaico del tomate (ToMV), perteneciente al género Tobamovirus. Se caracteriza por la manifestación de un diseño moteado o en mosaico en las hojas, con zonas de verde claro y verde oscuro. Además, puede provocar deformaciones en las hojas, tallos y frutos, así como un retraso en el crecimiento y una reducción en el rendimiento de la cosecha.

Figura 18

Virus del mosaico en el tomate

Recuperado de: Xatakandroid
https://www.xatakandroid.com/listas/21-consejos-imprescindibles-para-hacer-buenas-fotos-con-tu-movil-android

Mancha de la hoja: Es una afección fúngica provocada por el hongo ascomiceto Septoria lycopersici, que daña a las plantas de tomate. Se distingue por la aparición de marcas pequeñas y redondas en las hojas, con centros grises o blanquecinos y bordes más oscuros. Estas manchas pueden coalescer y causar la defoliación prematura si la enfermedad no se controla adecuadamente.

Figura 19

Manchas en hojas de tomate

Recuperado de: Minutoneuquen
https://www.minutoneuquen.com/entretenimiento/2024/3/6/por-que-mis-plantas-tienen-manchas-negras-como-erradicarlas-344507.html

Mancha foliar o mancha objetivo La mancha anillada del tomate es una afección fúngica causada por Corynespora cassiicola. Se manifiesta con manchas concéntricas en hojas, tallos y frutos, que evolucionan de marrón claro a oscuro. Esta enfermedad puede resultar en una grave defoliación y putrefacción de los frutos, comprometiendo seriamente la salud y productividad de la planta.

Figura 20

Mancha foliar de hojas de tomate

Recuperado de: X
https://x.com/tehuacatl1/status/713406601453895680

Virus del enrollado de la hoja La enfermedad del enrollado de la hoja del tomate es provocada por el TYLCV, un begomovirus transmitido por la mosca blanca. Sus síntomas principales incluyen hojas enrolladas hacia arriba, amarillamiento de la planta, crecimiento reducido y enanismo. También puede afectar los frutos y disminuir significativamente la producción. Si no se maneja adecuadamente, esta afección viral puede tener consecuencias graves para los cultivos de tomate.

Figura 21

Virus enrollado de las hojas de tomate

Recuperado de: Infoagro

Para evitar el oídio en tomate, es crucial erradicar las malezas y residuos de otros cultivos anteriores, además de emplear tratamientos específicos y gestionar el clima del invernadero. Armicarb® y Takumi® son opciones sugeridas de Certis Belchim. (MIRAVIS, 2018).

Enfermedad Mildiu del tomate (Phytophthora infestans), En el caso del tomate, el agente patógeno Phytophthora infestans puede perjudicar en cualquier etapa del cultivo. Requiere elevada humedad (90%) y temperaturas moderadas (10-25°C). Afecta hojas, tallo y frutos, produciendo manchas irregulares verde-pardas y apariencia aceitosa (MIRAVIS 2018).

Figura 22

Mildiu del tomate

Recuperado de: Certisbelchim

https://certisbelchim.es/plagas-y-enfermedades-del-cultivo-del-tomate/

Síntomas de Mildiu (Phytophthora infestans) en frutos y tallos de tomate

En fase avanzada, las manchas se tornan necróticas. En tallos surgen manchas pardas. En frutos, manchas marrones inician en el cáliz. Para prevenir el mildiu, ventilar invernaderos y evitar humedad excesiva. Usar tratamientos preventivos si el clima lo favorece. (Agrisolver, 2019).

La alternariosis del tomate es provocada por el hongo Alternaria solani, afectando a las solanáceas. Causa descomposición de frutos y daña hojas, tallo y pecíolos. En hojas, forma manchas circulares con halo amarillo; en tallo y pecíolo, lesiones negras y elongadas. (Vargas 2023).

Figura 23

Alternaria solani

Recuperado de: Certisbelchim
https://certisbelchim.es/plagas-y-enfermedades-del-cultivo-del-tomate/

Para evitar la alternariosis, es crucial eliminar los frutos y plantas infectadas y regular la humedad ambiental. Si las condiciones son propicias o aparecen primeros síntomas, emplear fungicidas como Kdos®. Amylo-X WG también presenta efecto secundario en esta enfermedad. (ITACyL 2019). Botritis (Botrytis cinerea), Conocida como pudrición gris, es un hongo que afecta los cultivos de tomate, potencialmente mortal para las plantas. En plantas maduras, causa daños marrones en tallos y pecíolos, invadiendo y colapsando el tallo hasta la muerte. En frutos, causa descomposición blanda; en hojas y flores, daños marrones similares a los del tallo. Para prevenir la Botritis, controlar el nivel de nitrógeno, retirar plantas afectadas y manejar cuidadosamente la poda. (Matute 2019).

Por el momento, la aplicación solo identificará 8 enfermedades. Entre ellas están Ramanjot y Ankita (2023): Bacterial Spot: Conocida como "mancha bacteriana", es ocasionada por Xanthomonas campestris pv. vesicatoria. Tizón Temprano: llamado "tizón temprano", afecta a tomates y papas por el hongo Alternaria solani. Tizón Tardío: conocido como "tizón tardío", causado por Phytophthora infestans, afecta severamente hojas, tallos y frutos de tomates y papas. Moho Foliar: también llamado "moho foliar", es causado por Fulvia fulva (también conocido como Cladosporium fulvum), afectando hojas de tomate con manchas amarillas y crecimiento mohoso en la parte inferior. Virus del Mosaico: conocido como "virus del mosaico del tomate", engloba virus como el del tabaco (TMV) y el del pepino (CMV), causando patrones de mosaico, deformaciones y problemas de crecimiento en tomates. Mancha de la Hoja: conocida como "mancha de la hoja", es causada por

Septoria lycopersici, causando pequeñas manchas redondas en hojas con centros claros y bordes oscuros, pudiendo llevar a defoliación sin control. Mancha Foliar: conocida como "mancha foliar" o "mancha objetivo", es causada por Corynespora cassiicola, caracterizada por manchas concéntricas en hojas, tallos y frutos de tomates. Virus del Enrollado de la Hoja: conocido como "virus del enrollado de la hoja", es causado por el virus del enrollado de la hoja del tomate (TYLCV), provocando síntomas como enrollamiento, amarillamiento y enanismo en tomates, afectando severamente los rendimientos si no se toman medidas de control.

II. METODOLOGÍA

Tipo, enfoque y diseño de investigación

Este tipo de investigación es de índole aplicada ya que se enfoca en resolver cuestiones prácticas y generar nuevo conocimiento relevante para la implementación de soluciones concretas. Esta metodología permite abordar problemas de manera práctica al aplicar conocimientos científicos, lo que resulta en beneficios tangibles para la empresa, como se ha destacado en estudios previos (Hernández-Sampieri y Mendoza 2018).

Además, este enfoque se caracteriza por ser cuantitativo, ya que el uso de datos numéricos permite precisar y enfocar las hipótesis a comprobar mediante la medición de las variables de estudio. En este diseño, el investigador desarrolla un experimento que busca comprender la naturaleza del fenómeno en estudio (Camacho, 2003). Este diseño implica un control mínimo de variables, lo que lo hace adecuado para establecer relaciones causales. El estudio se basa en un diseño pre-experimental, que implica aplicar un tratamiento o estímulo antes y después de pruebas para evaluar cómo una Aplicación Móvil con Reconocimiento de Imágenes basada en Inteligencia Artificial afecta la detección de enfermedades en cultivos de tomate. Según Sampieri et al. (2018) , se utiliza un único grupo investigativo con pretest y postest para compararlos con grupos estáticos sin asignación aleatoria. Esto se representa en el siguiente diagrama:

O1 = Pretest
X = Tratamiento
O2 = Postest

Evaluar la detección de enfermedades en cultivos de tomate utilizará ficha de observación como herramienta, evaluando dimensiones e indicadores específicos. Este proceso es crucial para preservar la salud de las plantas y mejorar el rendimiento de las cosechas. Consiste en identificar y diagnosticar de manera oportuna las enfermedades que afectan a las plantas de tomate, lo que permite implementar estrategias de control efectivas (Salas, Osorio, y Espinoza 2022).

En tanto el enfoque es descriptivo debido a que es un método de investigación que se centra en observar, documentar y describir fenómenos sin manipular variables (Arispe et al. 2020), siendo recomendable en el estudio porque permite comprender su funcionamiento en condiciones reales de cultivo, identificar patrones en imágenes de plantas enfermas, evaluar la usabilidad para los agricultores, documentar limitaciones y errores del sistema, y establecer una base para futuras mejoras. Este enfoque facilita la validación de la efectividad y confiabilidad de la aplicación en diversos escenarios, además de considerar aspectos éticos y sociales de su implementación, como la aceptación por parte de los agricultores y el impacto en las prácticas agrícolas tradicionales.

Variables

Variable independiente

Se define conceptualmente como un software para dispositivos móviles que emplea algoritmos de inteligencia artificial, específicamente visión por computadora y aprendizaje profundo, para analizar imágenes digitales capturadas por la cámara del dispositivo. Esta aplicación identifica, clasifica e interpreta patrones visuales asociados con condiciones de salud en plantas, como enfermedades, plagas o deficiencias nutricionales, basándose en modelos entrenados con grandes conjuntos de datos de imágenes etiquetadas.

El sistema reconoce características distintivas en nuevas imágenes, las compara con patrones conocidos y realiza diagnósticos, presentando resultados en la interfaz facilita la captura de imágenes y la interacción del usuario con la información generada, constituyendo así una herramienta que integra tecnología avanzada para asistir en la gestión de la salud de los cultivos (Piscoya 2019, pp. 30).

Variable dependiente

Las dimensión e indicadores para la detección de enfermedades en cultivos de tomate incluyen: Dimensión: **Control eficiente**, siendo sus indicadores, el primero el **Tiempo promedio para identificar plagas y enfermedades en el cultivo:** Indica el tiempo promedio que tarda en identificar (Piscoya 2019, pp. 30) Es así que refiere al tiempo que transcurre, en promedio, desde que se detectan los primeros síntomas de

una plaga o enfermedad en el cultivo hasta que se identifica de forma precisa cuál es la plaga o enfermedad específica que está afectando a las plantas. Utilizando la fórmula:

$$\frac{\sum tiempos\ por\ consulta}{Total\ de\ mediciones}$$

Asimismo, el indicador **porcentaje de Sensibilidad (Recall):** Según Sainz, Peña, Hernández (2017), el porcentaje representa la proporción de enfermedades presentes que fueron detectadas con precisión es crucial para aplicar tratamientos de manera efectiva y precisa (Sainz, Peña, Hernández 2017, pp. 5). Con su **fórmula:**

(#ECD) /(#TEP)

Dónde:

#ECD: número de enfermedades correctamente detectadas

#TEP: número total de enfermedades presentes.

Población y muestra

Por lo tanto, la población de estudio será el cultivo de tomate de la provincia de Cañete, y la selección de la muestra estará a discreción del investigador con una totalidad de 30 plantas de cultivo de tomate, siguiendo la recomendación de Huaire (2019).

Técnicas e instrumentos de recolección de datos

En esta sección se describen los medios que se emplearán para obtener los datos requeridos, tratando tanto la metodología como el dispositivo concreto para la investigación. El instrumento seleccionado fue la ficha de observación como una herramienta de recopilación de información mediante obtención de datos para obtener información específica de los encuestados (López et al. 2019).

La validez se refiere a la capacidad de un instrumento de medición para evaluar realmente la variable o constructo que se desea evaluar. Hay distintas formas de exactitud, como la exactitud de contenido, la exactitud de criterio y la exactitud de construcción (Grillo et al. 2023). Por otro lado, la confiabilidad se relaciona con la

fiabilidad y exactitud de un dispositivo de evaluación. Un instrumento fiable es aquel que produce resultados consistentes y estables cuando se aplica repetidamente en circunstancias similares. La confiabilidad puede evaluarse mediante diversos métodos, como el coeficiente alfa de Cronbach o el método de prueba-reprueba (Oviedo y Campo 2019).

Para analizar los datos obtenidos, se empleará el coeficiente alfa de Cronbach, como lo recomienda Sampieri, Fernández y Baptista (2019). Este método permite medir la consistencia interna de una escala, con un valor aproximado a 1, para evaluar La confiabilidad de los dispositivos empleados en la investigación se evaluó. Se realizó este análisis en las fichas de observación de pre y post test de las plantas de tomate seleccionados. Se estableció un umbral de significancia estadística con un valor de $p>0.05$, con un error permitido del 5% y una confianza del 95%.

Según el Decreto Ejecutivo N° 081 de la Universidad César Vallejo, aquellos interesados en llevar a cabo investigaciones en la institución deben adherirse al Código de Ética para la Investigación Científica establecido por el Vicerrectorado de Investigación en 2020. En este contexto, es fundamental considerar la política antiplagio, la cual garantiza la originalidad de la investigación y promueve la autoría responsable, conforme a las normas ISO 690. La comunidad universitaria tiene acceso a herramientas de software que facilitan la evaluación de similitudes con otras fuentes, siguiendo recomendaciones similares. Además, el departamento hace referencia a la propuesta de la UNESCO para la adopción de principios éticos universales, los cuales se fundamentan en cuanto al respeto a la dignidad humana y la salvaguarda de los derechos humanos (Vicerrectorado de Investigación 2021, p. 9).

III. RESULTADOS

3.1. Análisis descriptivo

En este estudio sobre la introducción de una app móvil con IA y reconocimiento visual para detectar enfermedades en tomates, se usó un diseño antes y después. Los resultados descriptivos se muestran en tablas para comparar y evaluar su eficacia inicial.

- INDICADOR: Tiempo promedio para identificar plagas y enfermedades en el cultivo

Los resultados explicativos del indicador de tiempo promedio para reconocer plagas y enfermedades en el cultivo de las medidas se visualizan en la tabla 3.

Tabla 3

Mediciones descriptivas del tiempo promedio para detectar plagas y enfermedades en el cultivo antes y después de utilizar la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial

	N	Mínimo	Máximo	Media	Desviación estándar
TPIPEC pre test	30	3,00	6,50	4,7500	0,8139
TPIPEC post test	30	0,06	0,09	0,0700	0,0087
N validado (por lista)	30				

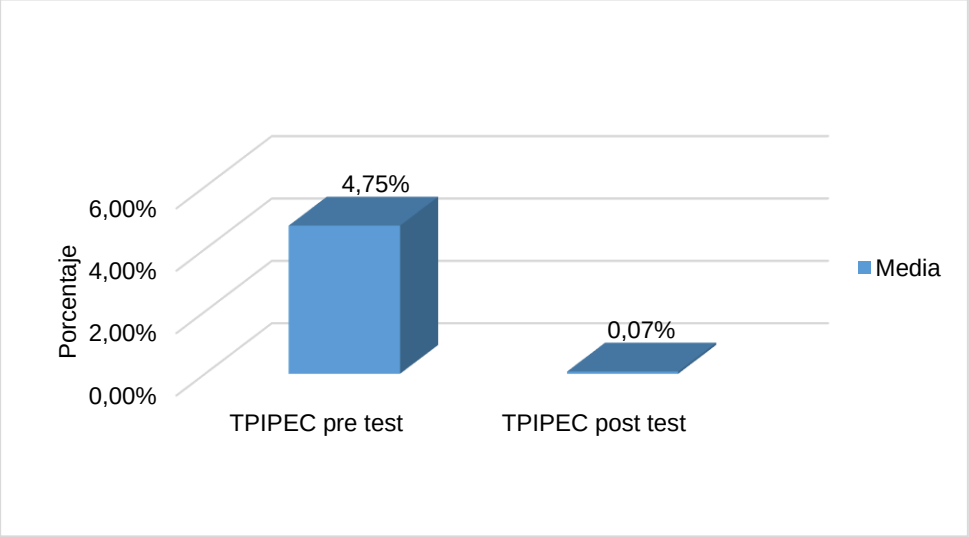
Fuente: Elaboración Propia

En cuanto al tiempo promedio para detectar plagas y enfermedades en el cultivo, se obtuvo un valor de 4.75% en el pre-test, comparado con 0.07% en el post-test, como se muestra en la figura 24. Esto indica una diferencia significativa antes y después de usar la app móvil con reconocimiento de imágenes basada en IA. La mejora mínima en el tiempo promedio para detectar plagas y enfermedades en el cultivo fue de 3.00% antes y 0.06% después, según la Tabla 1 de la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial.

Por lo tanto, la variabilidad del tiempo promedio para detectar plagas y enfermedades en el cultivo fue del 0,813% en el pre-test, mientras que en el post-test fue de 0,008%.

Figura 24

Tiempo promedio para identificar plagas y enfermedades en el cultivo



Fuente: Elaboración Propia

- INDICADOR: Porcentaje Sensibilidad (Recall)

Es así que los resultados descriptivos del indicador del porcentaje Sensibilidad (Recall) de las medidas se observan en la tabla 2.

Tabla 4

Medidas descriptivas del porcentaje Sensibilidad (Recall) en el pre-test y post-test de utilizar la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial

	N	Mínimo	Máximo	Media	Desviación estándar
Sensibilidad pre test	30	0,20	1,00	0,6697	0,2640
Sensibilidad post test	30	0,33	1,00	0,8333	0,2118
N validado (por lista)	30				

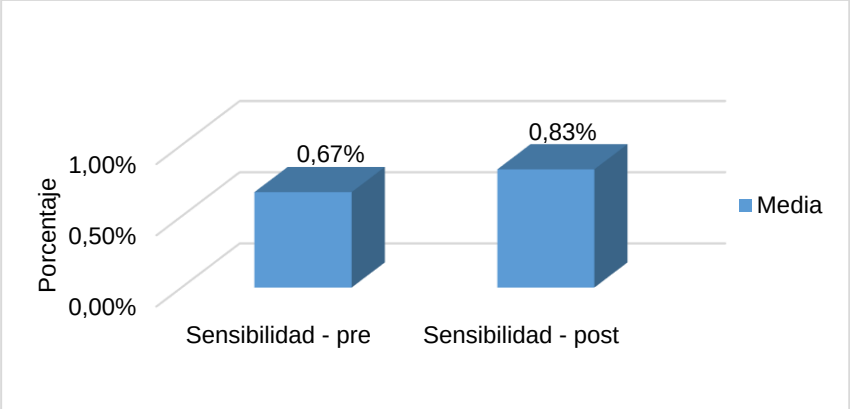
Fuente: Elaboración Propia

En cuanto al indicador de Sensibilidad (Recall), se obtuvo un valor de 0,67% en el pre-test, comparado con 0,83% en el post-test según la figura 25. Esto indica una diferencia significativa antes y después de implementar la app móvil con reconocimiento de imágenes basada en IA. La sensibilidad (Recall) mínimo fue de 0,20% antes y 0,33% después, como se muestra en la Tabla 4. Además, la dispersión

del tiempo promedio para detectar plagas y enfermedades en el cultivo fue del 0,26% en el pre-test y 0,21% en el post-test.

Figura 25

Porcentaje Sensibilidad (Recall)



Fuente: Elaboración Propia

1.1. Análisis descriptivo

Prueba de Normalidad

A continuación, se realizan pruebas de normalidad para evaluar los cambios en el tiempo promedio y la sensibilidad para detectar plagas y enfermedades en el cultivo de las plantas seleccionadas. Se aplica el método Shapiro-Wilk, ideal para muestras pequeñas (Mendoza, 2018, p.55). La muestra incluye 30 plantas de tomate estratificadas. Se utiliza el software SPSS versión 27, con un nivel de confianza del 95% para interpretar los resultados según criterios establecidos para verificar la normalidad de los datos.(Mendoza, 2018, p.55).

Si:

Sig. < 0.05 adopta una distribución no normal

Sig. ≥ 0.05 adopta una distribución normal

Donde

Sig.: P-valor o nivel crítico del contraste

Los datos resultantes fueron:

- INDICADOR: Tiempo promedio para detectar plagas y enfermedades en el cultivo

Para determinar la prueba de hipótesis adecuada, se evaluó la distribución de los datos relacionados con el tiempo promedio para identificar plagas y enfermedades

en el cultivo. Particularmente, se verificó si estos datos seguían una distribución normal.

Tabla 5

Prueba de Normalidad de tiempo promedio para identificar plagas y enfermedades en el cultivo en el pre-test y post-test de utilizar la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial

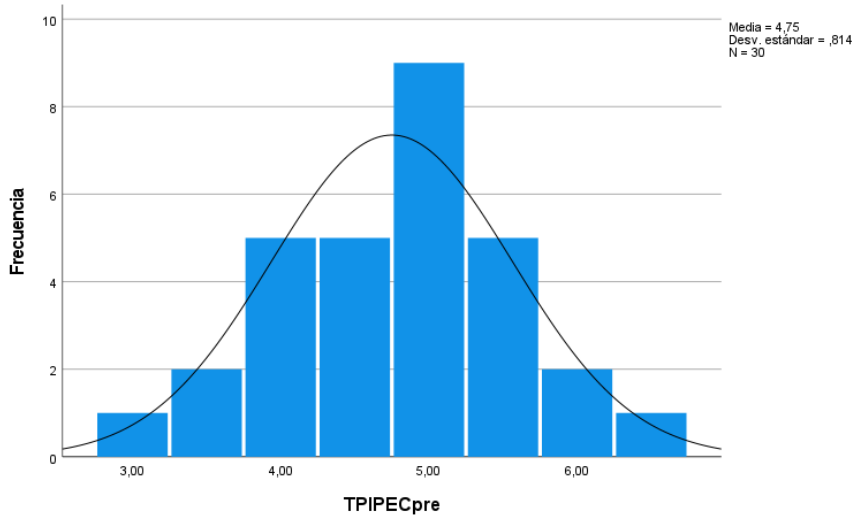
	Shapiro – Wilk		
	Estadístico	**gl**	**Sig.**
TPIPEC pre test	0,985	30	0,933
TPIPEC post test	0,848	30	< 0,001

Fuente: Elaboración Propia

Como se observa en la tabla 5, los resultados de la prueba muestran que el p-valor del tiempo promedio para detectar plagas y enfermedades en el cultivo con la app móvil de reconocimiento de imágenes basada en IA en el pre-test fue de 0.933, mayor que 0.05, indicando una distribución normal del indicador. En cambio, en el post-test, el p-valor fue <0.001, menor que 0.05, lo que indica una distribución no normal del indicador. Esto confirma la no normalidad de ambos conjuntos de datos, como se muestra en las Figuras 26 y 27.

Figura 26

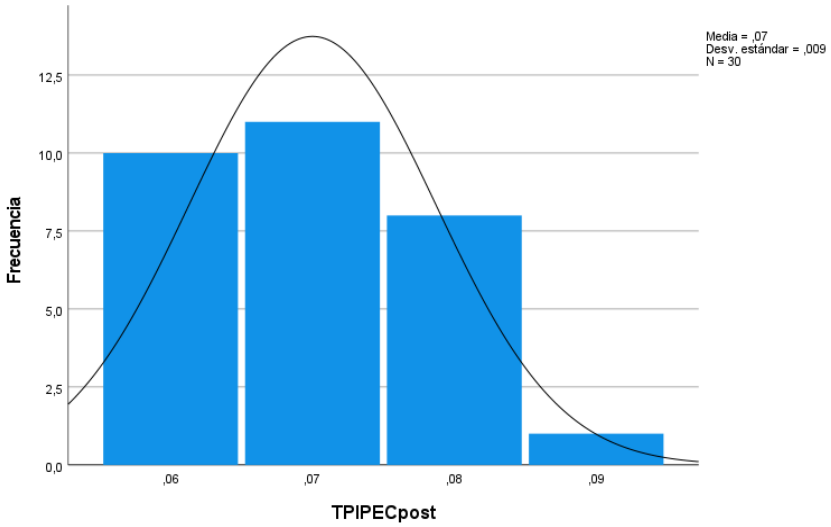
Prueba de Normalidad de tiempo promedio para identificar plagas y enfermedades en el cultivo en el pre-test de utilizar la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial



Fuente: Elaboración Propia

Figura 27

Prueba de Normalidad de tiempo promedio para identificar plagas y enfermedades en el cultivo en el post-test de utilizar la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial



Fuente: Elaboración Propia

- INDICADOR: Porcentaje Sensibilidad (Recall)

Para determinar la prueba de hipótesis adecuada, se evaluó la distribución de los datos relacionados con el porcentaje Sensibilidad (Recall). Particularmente, se verificó si estos datos seguían una distribución normal.

Tabla 6

Prueba de Normalidad el porcentaje Sensibilidad (Recall) en el pre-test y post-test de utilizar la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial

	Shapiro – Wilk		
	Estadístico	**gl**	**Sig.**
Sensibilidad pre test	0,881	30	0,003
Sensibilidad post test	0,763	30	< 0,001

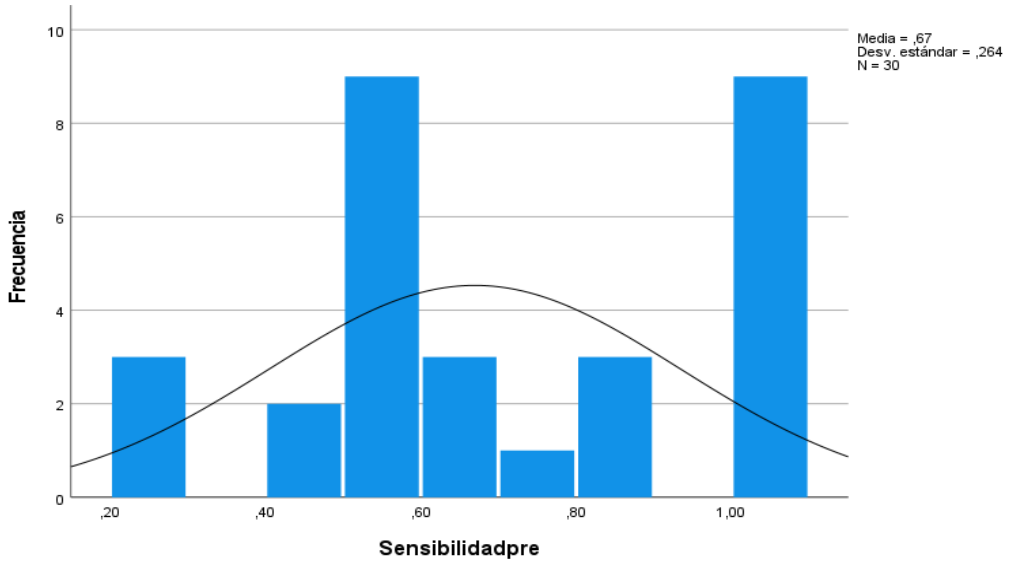
Fuente: Elaboración Propia

Como se puede ver en la tabla 6, los resultados del análisis muestran que el p-valor del porcentaje de Sensibilidad (Recall) con la app móvil de reconocimiento de imágenes basada en IA en el pre-test fue de 0.003, menor que 0.05, indicando una distribución no normal del indicador. En el post-test, el p-valor fue <0.001, también

menor que 0.05, confirmando que el indicador no se distribuye normalmente. Esta falta de normalidad en ambos conjuntos de datos se verifica en las Figuras 28 y 29.

Figura 28

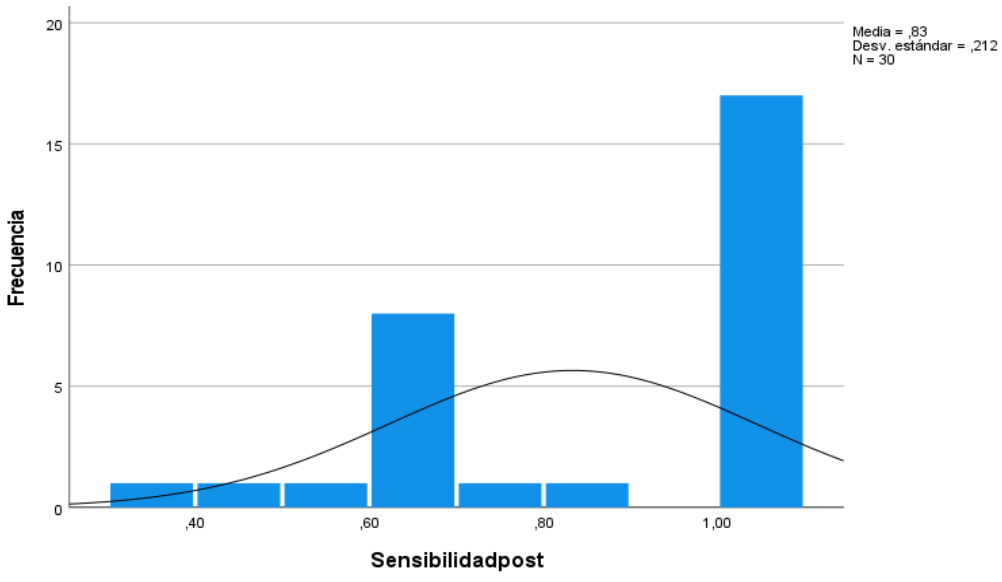
Prueba de Normalidad el porcentaje Sensibilidad (Recall) en el pre-test de utilizar la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial



Fuente: Elaboración Propia

Figura 29

Prueba de Normalidad Prueba de Normalidad el porcentaje Sensibilidad (Recall) en el post-test de utilizar la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial



Fuente: Elaboración Propia

3.2. Pruebas de hipótesis

Hipótesis de investigación 1:

- **H1:** Una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial disminuye el tiempo promedio para identificar plagas y enfermedades en cultivos de tomate.
- **Indicador:** Tiempo promedio para identificar plagas y enfermedades en el cultivo

Hipótesis Estadísticas

Definiciones de Variables:

TPIPECa: Tiempo promedio para identificar plagas y enfermedades en el cultivo antes del aplicativo móvil

TPIPECd: Tiempo promedio para identificar plagas y enfermedades en el cultivo después del aplicativo móvil

- **H0:** Una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial no disminuye el tiempo promedio para identificar plagas y enfermedades en cultivos de tomate.

$$H0: TPIPECa \geq TPIPECd$$

El indicador sin el aplicativo móvil es mejor que el indicador con el aplicativo móvil

- **HA:** Una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial disminuye el tiempo promedio para identificar plagas y enfermedades en cultivos de tomate.

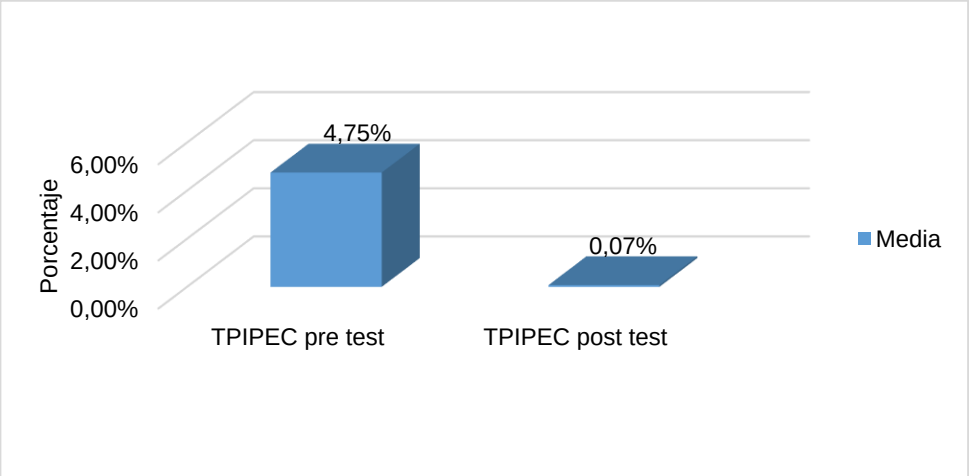
$$HA: TPIPECa < TPIPECd$$

El indicador sin el aplicativo móvil es mejor que el indicador con el aplicativo móvil

Los resultados se encuentran en la Figura 30, del tiempo promedio para detectar plagas y enfermedades en el cultivo pre-test de 4,75% y el post-test de 0,07%.

Figura 30

Tiempo promedio para identificar plagas y enfermedades en el cultivo – comparativa general



Fuente: Elaboración Propia

Se determina que en la Figura 30 hay un aumento en el tiempo medio para detectar plagas y enfermedades en el cultivo, evidenciado por la reducción de sus medias de 4,75% a 0,07%. Respecto al contraste de hipótesis, se utilizó la prueba de Wilcoxon debido a que los datos de la investigación (pre y post) no presentan distribución normal. El valor de T es -4.783, claramente inferior a -1.6991.

Tabla 7

Prueba de Wilcoxon para el tiempo promedio para identificar plagas y enfermedades en el cultivo en el pre-test y post-test de utilizar la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial

	Media	**Prueba de Wilcoxon**		
		T	**gl**	**Sig.**
TPIPEC pre test	4,7500	**-4,783**	29	< 0,001
TPIPEC post test	0,0700			

Fuente: Elaboración Propia

Es por lo tanto que se rechaza la hipótesis nula, aceptando la hipótesis alterna con un 95% de confianza. Además el valor T obtenido, como se muestra en la Figura 31, se ubica en la zona de rechazo; entonces la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial disminuye el tiempo promedio para identificar plagas y enfermedades en cultivos de tomate.

Aplicando la fórmula Wilcoxon

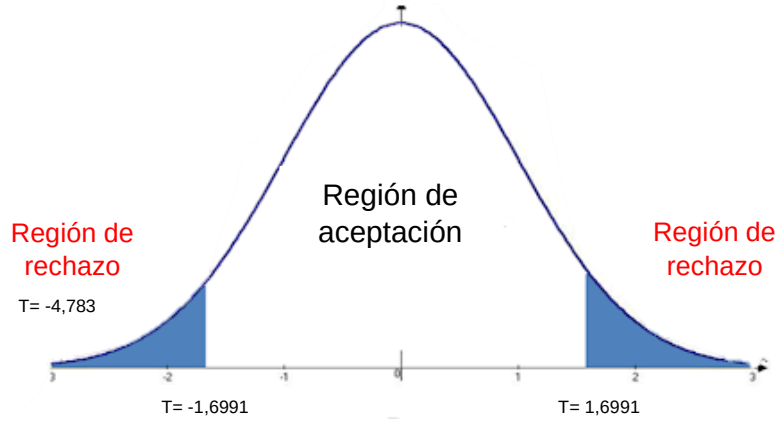
$$z = \frac{T - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}}$$

$$z = \frac{48{,}56 - \frac{30(30+1)}{4}}{\sqrt{\frac{30(30+1)(2(30)+1)}{24}}}$$

$$z = -4{,}783$$

Figura 31

Prueba Wilcoxon - El tiempo promedio para identificar plagas y enfermedades en cultivos



Fuente: Elaboración Propia

Hipótesis de investigación 2:

- **H2:** Una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial mejora el porcentaje de sensibilidad en la detección de enfermedades en cultivos de tomate.
- **Indicador:** Porcentaje sensibilidad (Recall)

Hipótesis Estadísticas

Definiciones de Variables:

SensibilidadA: Porcentaje sensibilidad (Recall) antes del aplicativo móvil

SensibilidadD: Porcentaje sensibilidad (Recall) después del aplicativo móvil

- **H0:** Una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial no mejora el porcentaje de sensibilidad en la detección de enfermedades en cultivos de tomate.

H0: SensibilidadA ≥ SensibilidadD

El indicador sin el aplicativo móvil es mejor que el indicador con el aplicativo móvil

- **HA:** Una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial mejora el porcentaje de sensibilidad en la detección de enfermedades en cultivos de tomate.

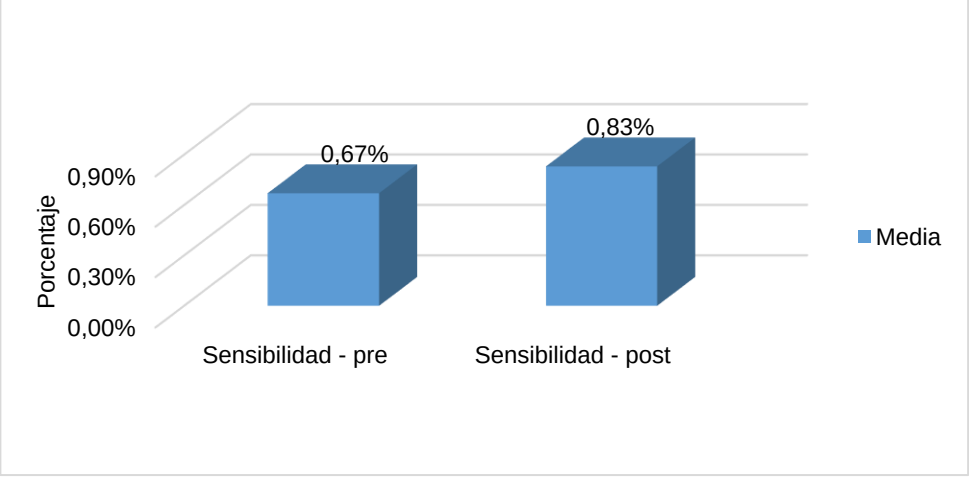
HA: SensibilidadA < SensibilidadD

El indicador sin el aplicativo móvil es mejor que el indicador con el aplicativo móvil

Los resultados se encuentran en la Figura 32, del porcentaje sensibilidad pre-test de 0,67% y el post-test de 0,83%.

Figura 32

Porcentaje sensibilidad (Recall) – comparativa general



Fuente: Elaboración Propia

Se concluye que en la Figura 9 hay un aumento en el porcentaje sensibilidad (Recall), evidenciado por el incremento de sus medias de 0,67% a 0,83%. En cuanto al contraste de hipótesis, se utilizó la prueba de Wilcoxon debido a que los datos de

la investigación (pre y post) no presentan distribución normal. El valor de T es -2.471, claramente menor que -1.6991.

Tabla 8

Prueba de Wilcoxon para el porcentaje sensibilidad (Recall) en el pre-test y post-test de utilizar la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial

	Media	**Prueba de Wilcoxon**		
		T	**gl**	**Sig.**
Sensibilidad pre test	0,6697	**-2,471**	29	0,013
Sensibilidad post test	0,8333			

Fuente: Elaboración Propia

Aplicando la fórmula Wilcoxon

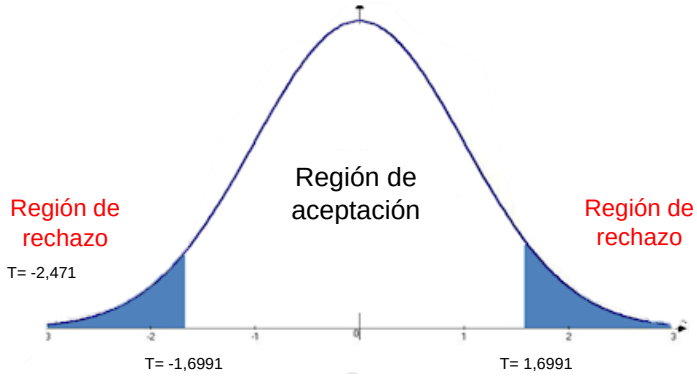
$$z = \frac{T - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24}}}$$

$$z = \frac{57{,}90 - \frac{30(30+1)}{4}}{\sqrt{\frac{30(30+1)(2(30)+1)}{24}}}$$

$$z = -4{,}793$$

Figura 33

Prueba Wilcoxon – Porcentaje sensibilidad (Recall)



Fuente: Elaboración Propia

IV. DISCUSIÓN

En la siguiente sección presenta los datos resultantes del estudio sobre la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate; vinculadas a las hipótesis planteadas que se detallan a continuación:

La hipótesis 1 manifiesta que: Una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial disminuye el tiempo promedio para identificar plagas y enfermedades en cultivos de tomate. El tiempo promedio para identificar plagas y enfermedades en cultivos en la medición del pre-test fue de 4,75% y con la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial fue de 0,07%. No obstante los resultados indican que existe una disminución de 4,68%, lo que afirma que la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial disminuye el tiempo promedio para identificar plagas y enfermedades en cultivos de tomate.

Según el estudio de Ángel Gavilanez y Bryan Saragosin en el año 2024, la aplicación móvil desarrollada no solo mejora la toma de decisiones inmediatas para los agricultores, sino que también presenta un potencial de expansión a otras zonas agrícolas. Los autores reportan una reducción en el tiempo de respuesta del 0,58% al 0,48%, lo que representa una mejora del 0,10%. Esta optimización permite abordar de manera más ágil los problemas fitosanitarios, fomentando una gestión agrícola más eficiente y sostenible. Por su parte, Jesús Piscoya en su investigación de 2019, destaca que las pruebas realizadas con la aplicación móvil demostraron una notable reducción en el tiempo que los agricultores necesitan para acceder a información crucial sobre plagas, enfermedades y agroquímicos, logrando un tiempo de consulta de aproximadamente 0,04%. Esta eficiencia se atribuye a la centralización de información esencial en una sola plataforma, incluyendo imágenes de plagas y enfermedades, datos sobre agroquímicos, y un mapa de rutas hacia las tiendas de suministros agrícolas.

La hipótesis 2 manifiesta que: Una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial mejora el porcentaje de sensibilidad en la detección de enfermedades en cultivos de tomate. El porcentaje sensibilidad (Recall)

en la medición del pre-test fue de 0,67% y con la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial fue de 0,83%. No obstante los resultados indican que existe un aumento del 0,16%, lo que afirma que la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial mejora el porcentaje de sensibilidad en la detección de enfermedades en cultivos de tomate.

El estudio de Agustina Barreto en el año 2022 revela que, de un conjunto de 7 capturas de detecciones realizadas, se logró identificar un total de 15 elementos. Este proceso resultó en una sensibilidad del 0,80% para la clasificación correcta, lo que indica una alta precisión en la detección y categorización de los elementos analizados. Por otro lado, Jaime Jácome en el 2022, reporta en su investigación una exactitud aún mayor, alcanzando un 0,968% en las pruebas realizadas. Esta alta precisión fue validada por expertos en el campo, lo que confirma que el prototipo desarrollado cumple satisfactoriamente con los objetivos y expectativas establecidos en la investigación. Estos resultados subrayan la eficacia de las herramientas de detección y clasificación desarrolladas en ambos estudios, demostrando su potencial para aplicaciones prácticas en sus respectivos campos.

V. CONCLUSIONES

Primero: Se culmina que el tiempo promedio para identificar plagas y enfermedades en el cultivo de tomate sin la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial fue de 4,75% y con la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial fue de 0,07%. Se observó una disminución significativa del 4,68% en el tiempo requerido para el diagnóstico, pasando de un promedio de 20 minutos con métodos tradicionales a menos de 1 minuto utilizando la aplicación. Esta mejora en la eficiencia permite a los agricultores realizar evaluaciones más frecuentes y exhaustivas de sus cultivos, facilitando la detección temprana de problemas y la implementación oportuna de medidas de control, lo que resulta en una mejor gestión general de la salud de los cultivos.

Segundo: Se culmina que el porcentaje Sensibilidad (Recall) sin la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial fue de 0,67% y con la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial fue de 0,83%. Los resultados indican un aumento en la sensibilidad de detección del 0,16% lo que significa una mejora significativa en la capacidad para identificar correctamente las enfermedades presentes. Esta mayor sensibilidad se traduce en una detección más precisa y temprana de las patologías, permitiendo intervenciones más efectivas y reduciendo el riesgo de pérdidas de cultivos. Además, la aplicación ha mostrado una capacidad notable para distinguir entre diferentes enfermedades con síntomas similares, proporcionando diagnósticos más precisos y facilitando la selección de tratamientos adecuados.

Tercero: Como último, la aplicación móvil desarrollada ofrece un análisis preliminar rápido y preciso, pero su eficacia depende crucialmente de la calidad de las imágenes ingresadas. Es fundamental seleccionar imágenes de hojas con fondos limpios, ya que las perturbaciones pueden llevar a predicciones erróneas. Para mejorar la confiabilidad en futuras versiones, se recomienda implementar técnicas de segmentación de imágenes basadas en inteligencia artificial, lo que permitiría aislar mejor las hojas del fondo. Adicionalmente, para casos donde una hoja pueda presentar múltiples enfermedades, se sugiere incorporar tecnología de detección de

objetos, lo que posibilitaría la identificación y delimitación precisa de cada enfermedad de manera individual. Estas mejoras potenciarían significativamente la capacidad y precisión de la aplicación en diversos escenarios de análisis de enfermedades en hojas.

VI. RECOMENDACIONES

Primero: Fomentar el uso de la aplicación móvil entre los agricultores para mejorar la eficiencia en la detección de plagas y enfermedades. Para ello, es fundamental realizar capacitaciones y talleres que faciliten la adopción de la tecnología, prestando especial atención a las áreas con menor experiencia tecnológica. Estos esfuerzos educativos no solo ayudarán a los agricultores a familiarizarse con la herramienta, sino que también les permitirán maximizar su utilidad y beneficios en la gestión de sus cultivos.

Segundo: Aprovechar la reducción del tiempo de diagnóstico que ofrece la aplicación para realizar evaluaciones más frecuentes y detalladas de los cultivos. Es crucial implementar protocolos de monitoreo regular utilizando la aplicación para asegurar la detección temprana de problemas. Esto permitirá una respuesta oportuna y adecuada, mejorando la salud general de los cultivos y reduciendo el riesgo de pérdidas significativas.

Tercero: Asegurar el mantenimiento continuo y la actualización de la aplicación para adaptarse a nuevas enfermedades y plagas emergentes. Es vital colaborar con expertos en fitopatología para mejorar constantemente los algoritmos de reconocimiento de imágenes. Esta colaboración garantizará que la aplicación permanezca a la vanguardia de la tecnología y siga siendo una herramienta eficaz para los agricultores en la lucha contra las plagas y enfermedades.

REFERENCIAS

ACOSTA, J., LENIN, A. y SANAFRIA, W., 2022. Las aplicaciones móviles y su impacto en la sociedad. *Revista Universidad y Sociedad*, vol. 14, no. 2,

AHMED, A.A. y HARSHAVARDHAN REDDY, G., 2021. A Mobile-Based System for Detecting Plant Leaf Diseases Using Deep Learning. *AgriEngineering*, vol. 3, no. 3, ISSN 26247402. DOI 10.3390/agriengineering3030032.

ARIAS GÓMEZ, J., VILLASÍS KEEVER, Á.M. y MIRANDA NOVALES, G.M., 2016. El protocolo de investigación III: la población de estudio. *Revista Alergia México*, vol. 63, no. 2,

ARISPE ALBURQUEQUE, C.M., YANGALI VICENTE, J.S., GUERRERO BEJARANO, M.A., LOZADA DE BONILLA, O., ACUÑA GAMBOA, L.A. y ARELLANO SACRAMENTO, C., 2020. *La investigación científica: Una aproximación para los estudios de posgrado* [en línea]. Guayaquil - Ecuador: s.n. ISBN 978-9942-38-578-9. Disponible en: https://repositorio.uide.edu.ec/bitstream/37000/4310/1/LA INVESTIGACIÓN CIENTÍFICA.pdf.

ASANI, E.O., OSADEYI, Y.P., ADEGUN, A.A., VIRIRI, S., AYOOLA, J.A. y KOLAWOLE, E.A., 2023. mPD-APP: a mobile-enabled plant diseases diagnosis application using convolutional neural network toward the attainment of a food secure world. *Frontiers in Artificial Intelligence*, vol. 6, ISSN 26248212. DOI 10.3389/frai.2023.1227950.

BARRETO, A.A., 2022. Deep learning aplicado al procesamiento de imágenes para la detecciones de objetos Instituto : Ingeniería y Agronomía Carrera : Ingeniería en Informática. *Repositorio UNAJ*,

BOROUMAND, A., GHOSE, S., PATEL, M., HASSAN, H., LUCIA, B., AUSAVARUNGNIRUN, R., HSIEH, K., HAJINAZAR, N., MALLADI, K.T., ZHENG, H. y MUTLU, O., 2019. CoNDA: Efficient cache coherence support for near-data accelerators. *Proceedings - International Symposium on Computer Architecture*, ISSN 10636897. DOI 10.1145/3307650.3322266.

CERVANTES, A. y BALLADARES, C., 2022. Competencias digitales: lenguaje de programación y rendimiento académico. *Ciencia Latina Revista Científica Multidisciplinar*, vol. 6, no. 1, ISSN 2707-2215. DOI 10.37811/cl_rcm.v6i1.1516.

CHERNG, V., TUNKU, J.U. y RAHMAN, A., 2020. an Educational Android App for Identifying Animals in Zoo. ,

CORONADO, J., 2023. *CLASIFICACIÓN DE ENFERMEDADES FÚNGICAS DEL FOLLAJE EN TOMATE (Solanum lycopersicum Mill) CON DEEP LEARNING*. S.l.: Tecnológico Nacional de México.

CUSME ZAMBRANO, K.D. y LOOR PINARGOTE, A.M., 2019. Aplicación móvil de detección y clasificación de “la roya” en hojas de café robusta mediante aprendizaje automático. [en línea], Disponible en: http://repositorio.espam.edu.ec/handle/42000/1104.

DELNEVO, G., GIRAU, R., CECCARINI, C. y PRANDI, C., 2022. A Deep Learning and Social IoT Approach for Plants Disease Prediction Toward a Sustainable Agriculture. *IEEE Internet of Things Journal*, vol. 9, no. 10, ISSN 23274662. DOI 10.1109/JIOT.2021.3097379.

ERAZO, W., 2022. *Desarrollo de aplicación Android móvil utilizando Visión Artificial y Deep Learning para la identificación de Aguacates Hass con la plaga Monalonion, en la finca “Las Palmas”, ubicada en el municipio de San Agustín, Huila*. S.l.: s.n.

FERNÁNDEZ, D.I., 2019. Desarrollo y experimentación de un programa de monitorización y control mediante visión artificial y redes neuronales siamesas. ,

GALAN, J., 2021. *SISTEMA INTELIGENTE DE RECONOCIMIENTO DE IMÁGENES PARA APOYAR EL DIAGNÓSTICO DE PLAGAS Y ENFERMEDADES EN EL CULTIVO DE ARROZ EN EL DEPARTAMENTO DE LAMBAYEQUE EN EL AÑO 2019*. S.l.: s.n.

GAMBOA, P. y LÓPEZ, I., 2021. *«Desarrollo De Un Prototipo Para El Reconocimiento De Plagas En Plantaciones De Babaco Bajo Invernadero Utilizando Redes Neuronales»* [en línea]. S.l.: Universidad Técnica de Cotopaxi. Disponible en: http://repositorio.utc.edu.ec/bitstream/27000/8625/1/PI-001947.pdf.

GAMERO, E., 2023. Inteligencia Artificial y Sector Público. Retos, límites y medios”. “Algoritmos, inteligencia artificial y procedimiento administrativo: principios comunes en el Derecho de la Unión Europea. *Revista de Derecho Politico*, no. 117, ISSN 21745625.

GAVILANEZ GUANOLUISA, Á.R. y SARAGOSIN GUAMUSHIG, B.A., 2024. *DESARROLLO DE UN PROTOTIPO PARA LA IDENTIFICACIÓN AUTOMÁTICA DE PLAGAS Y ENFERMEDADES EN EL CULTIVO DE PAPA, UTILIZANDO TÉCNICAS DE INTELIGENCIA ARTIFICIAL EN LA CIUDAD DE LATACUNGA*

[en línea]. S.l.: s.n. Disponible en: https://repositorio.utc.edu.ec/handle/27000/12029.

GÉRON, A., 2019. *Aprendizaje automático práctico con Scikit-Learn, Keras y TensorFlow*. S.l.: O'Reilly Media. ISBN 9781492032649.

GÓMEZ-CAMPEROS, J., JARAMILLO, H. y GUERRERO-GÓMEZ, G., 2021. Técnicas de procesamiento digital de imágenes para detección de plagas y enfermedades en cultivos: una revisión. *INGENIERÍA Y COMPETITIVIDAD*, vol. 24, no. 1, ISSN 0123-3033. DOI 10.25100/iyc.v24i1.10973.

GOMEZ SELVARAJ, M., VERGARA, A., MONTENEGRO, F., ALONSO RUIZ, H., SAFARI, N., RAYMAEKERS, D., OCIMATI, W., NTAMWIRA, J., TITS, L., OMONDI, A.B. y BLOMME, G., 2020. Detection of banana plants and their major diseases through aerial images and machine learning methods: A case study in DR Congo and Republic of Benin. *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 169, ISSN 09242716. DOI 10.1016/j.isprsjprs.2020.08.025.

GRILLO, C., MORENO, W., RAMIREZ, D., GÓMEZ, C. y ABELLO, M., 2023. *Pensamiento Crítico En La Investigación Científica Y Académica* [en línea]. S.l.: s.n. vol. 21. ISBN 978-628-95884-1-5. Disponible en: https://doi.org/10.34893/e1150-3660-8721-s.

HARRIS, C.R., MILLMAN, K.J., VAN DER WALT, S.J., GOMMERS, R., VIRTANEN, P., COURNAPEAU, D., WIESER, E., TAYLOR, J., BERG, S., SMITH, N.J., KERN, R., PICUS, M., HOYER, S., VAN KERKWIJK, M.H., BRETT, M., HALDANE, A., DEL RÍO, J.F., WIEBE, M., PETERSON, P., GÉRARD-MARCHANT, P., SHEPPARD, K., REDDY, T., WECKESSER, W., ABBASI, H., GOHLKE, C. y OLIPHANT, T.E., 2020. Array programming with NumPy. *Nature* [en línea], vol. 585, no. 7825, ISSN 14764687. DOI 10.1038/s41586-020-2649-2. Disponible en: http://dx.doi.org/10.1038/s41586-020-2649-2.

HERNÁNDEZ-SAMPIERI & MENDOZA C, R., 2018. *Metodología de la investigación:las rutas cuantitativa, cualitativa y mixta*. México: McGraw-Hil.

HERNÁNDEZ, A., 2021. *Detección de plagas y enfermedades del limón persa mediante una aplicación móvil con aprendizaje profundo a partir de redes neuronales* [en línea]. S.l.: s.n. Disponible en: http://51.143.95.221/bitstream/TecNM/5144/1/TS46DE~1.PDF.

HUAIRE, I., 2019. Método de investigación. [en línea], Disponible en: https://es.scribd.com/document/538137060/Edson-Jorge-Huaire-Inacio-2019-

Metodo-de-Investigacion-1.

JÁCOME, J., 2022. *Detección temprana de minador, mosca blanca y fusarium en el tomate riñón, aplicando técnicas de visión artificial y machine learning.* [en línea]. S.l.: Universidad Nacional de Chimborazo. Disponible en: http://dspace.unach.edu.ec/handle/51000/9102.

KUMAR, S., RATAN, R. y DESAI, J. V., 2022. A Study of iOS Machine Learning and Artificial Intelligence Frameworks and Libraries for Cotton Plant Disease Detection. *Lecture Notes in Electrical Engineering*. S.l.: s.n., vol. 768. DOI 10.1007/978-981-16-2354-7_24.

LÁZARO, A., 2021. *Aplicativo Móvil Para El Control Deproductos Químicos En Plagas Yenfermedades De Cultivo De Aguaymanto Eneldistritode Shilla-Carhuaz-Ancash*. S.l.: s.n.

LÓPEZ, E., MALDONADO, G., EDSON, P., HUAIRE, J., CINTHYA, I. y JIMÉNEZ, A.F., 2019. *Hacia la investigación transdisciplinar: retos y desafíos de la sociedad contemporánea Coordinadores* [en línea]. S.l.: s.n. ISBN 9786124148125. Disponible en: www.une.edu.pe.

MAHOVIC, M.J., SARGENT, S.A., BARTZ, J.A. y LON KAN, E.E., 2017. Identificación y Control Postcosecha de las Enfermedades del Tomate en la Florida. *EDIS*, vol. 20, no. 10, DOI 10.32473/edis-hs334-2006.

MARTÍN, A.G.S., 2021. Diseño de un modelo compuesto de arquitecturas de red neuronal convolucional y recurrente para descripción de video en entornos outdoor/indoor. [en línea], Disponible en: https://riunet.upv.es:443/handle/10251/165340.

MATEO, J.D., 2017. *COMPETICIÓN DE KAGGLE.COM* [en línea]. S.l.: s.n. Disponible en: https://dspace.unia.es/bitstream/handle/10334/3850/0822_Mateo.pdf?sequence=3.

MEDINA, J.M. y URTEAGA, J.A., 2021. *Impacto de la aplicación móvil "Healthy Plant" para detectar enfermedades foliares en cultivos de aguaymanto haciendo uso de inteligencia artificial con custom vision en la ciudad de cajamarca 2021* [en línea]. S.l.: s.n. Disponible en: https://hdl.handle.net/11537/28251.

MENDOZA, H.-S.&, 2018. La Investigación Científica. ,

MOLOO, R.K. y CALEECHURN, K., 2022. An App for Fungal Disease Detection on Plants. *2022 International Conference for Advancement in Technology, ICONAT*

2022. S.l.: s.n., DOI 10.1109/ICONAT53423.2022.9725839.

OVIEDO, H.C. y CAMPO ARIAS, A., 2019. Aproximación al uso del coeficiente alfa de Cronbach. *Revista Colombiana de estadística* [en línea], vol. 34, no. 4, Disponible en: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0034-74502005000400182.

PISCOYA FERREÑAN, J.E., 2019. *Sistema de visión artificial para apoyar en la identificación de plagas y enfermedades del cultivo de sandía en el distrito de ferreñafe* [en línea]. S.l.: Universidad Católica Santo Toribio de Mogrovejo. Disponible en: https://tesis.usat.edu.pe/bitstream/20.500.12423/2356/1/TL_PiscoyaFerreñanJesus.pdf.

QUINTERO-ROJAS, J. y GONZÁLEZ, J.D., 2021. Use of Convolutional Neural Networks in Smartphones for the Identification of Oral Diseases Using a Small Dataset. *Revista Facultad de Ingeniería*, vol. 30, no. 55, ISSN 0121-1129. DOI 10.19053/01211129.v30.n55.2021.11846.

QUINTERO, J. y GONZÁLEZ, J., 2021. Use of Convolutional Neural Networks in Smartphones for the Identification of Oral Diseases Using a Small Dataset. *Revista Facultad de Ingeniería*, vol. 30, no. 55, ISSN 0121-1129. DOI 10.19053/01211129.v30.n55.2021.11846.

RAHAMAN, N., CHOWDHURY, M.R., RAHMAN, A., AHMED, H., HOSSAIN, M., RAHMAN, H., BISWAS, S., KADER, A., NOYAN, T.A. y BISWAS, M., 2023. A Deep Learning Based Smartphone Application for Detecting cacao Diseases and Pesticide Suggestions. *International Journal of Computing and Digital Systems*, vol. 13, no. 1, ISSN 2210142X. DOI 10.12785/ijcds/1301104.

RAMAKRISHNAM RAJU, S.V.S., DAPPURI, B., RAVI KIRAN VARMA, P., YACHAMANENI, M., VERGHESE, D.M.G. y MISHRA, M.K., 2022. Design and Implementation of Smart Hydroponics Farming Using IoT-Based AI Controller with Mobile Application System. *Journal of Nanomaterials*, vol. 2022, ISSN 16874129. DOI 10.1155/2022/4435591.

RAMOS, E., 2021. *SISTEMA INTELIGENTE PARA IDENTIFICAR ADECUADAMENTE EL MANGO KENT NO EXPORTABLE EN EL ÁREA DE MUESTREO DE UNA EMPRESA AGROINDUSTRIAL DE LA REGIÓN LAMBAYEQUE*. S.l.: s.n.

RAMOS, J., 2020. *Algoritmo integrado con inteligencia Artificial apoyado en mano robótica para el reconocimiento de la madurez del tomate* [en línea]. S.l.: s.n. Disponible en: https://renati.sunedu.gob.pe/handle/sunedu/2823063.

SABARRE, A.B.L., NAVIDAD, A.N.S., TORBELA, D.S. y ADTOON, J.J., 2021. Development of durian leaf disease detection on Android device. *International Journal of Electrical and Computer Engineering*, vol. 11, no. 6, ISSN 20888708. DOI 10.11591/ijece.v11i6.pp4962-4971.

SAINZ, M.D.J., PEÑA, G. y HERNÁNDEZ, V.M., 2017. Sensibilidad de la prueba de InmunoStrips® en la detección de Clavibacter michiganensis subsp. michiganensis en tomate. *ActA AgrícolA y pecuAriA*, vol. 3, no. 2,

SALAS GÓMEZ, A.L., OSIRIO HERNÁNDEZ, E., ESPINOZA AHUMADA, C.A., RODRÍGUEZ HERRERA, R., SEGURA MARTÍNEZ, M.T. de J., NERI RAMÍREZ, E. y ESTRADA DROUAILLET, B., 2022. Principales enfermedades del cultivo de tomate (Solanum lycopersicum L.) en condiciones de campo. *Ciencia Latina Revista Científica Multidisciplinar* [en línea], vol. 6, no. 1, ISSN 2707-2207. DOI 10.37811/cl_rcm.v6i1.1793. Disponible en: https://ciencialatina.org/index.php/cienciala/article/view/1793.

SAMPIERI, R., FERNÁNDEZ, C. y BAPTISTA, P., 2019. Proceso de Investigación. *Mc Graw Hill* [en línea], vol. 53, no. 9, ISSN 1098-6596. Disponible en: https://josetavarez.net/Compendio-Metodologia-de-la-Investigacion.pdf.

SANATH RAO, U., SWATHI, R., SANJANA, V., ARPITHA, L., CHANDRASEKHAR, K., CHINMAYI y NAIK, P.K., 2021. Deep Learning Precision Farming: Grapes Leaf Disease Detection by Transfer Learning. *Global Transitions Proceedings*, vol. 2, no. 2, ISSN 2666285X. DOI 10.1016/j.gltp.2021.08.002.

SHOAIB, M., AHMAD, F. y REHMAN, M.A., 2022. Deep learning-based plant disease detection using android apps. *Artificial Intelligence Applications in Agriculture and Food Quality Improvement*. S.l.: s.n.,

SUCARI LEÓN, R., AROQUIPA DURÁN, Y., QUINA QUINA, L.D., QUISPE YAPO, E., SUCARI LEÓN, A. y HUANCA TORRES, F.A., 2020. Visión artificial en reconocimiento de patrones para clasificación de frutas en agronegocios. *Puriq*, vol. 2, no. 2, ISSN 2664-4029. DOI 10.37073/puriq.2.2.76.

TAMAYO RUIZ, J., OSORIO TRASVIÑA, J. y MANCERA ROJAS, E., 2024. Detección de enfermedades en cultivos de maíz mediante imágenes con visión artificial: un caso práctico. *Revista Científica Ciencia y Tecnología*, vol. 24, no. 41, ISSN 1390-

6321. DOI 10.47189/rcct.v24i41.681.

TORRES, N., 2021. SISTEMA RECOMENDADOR DE OBJETOS DE APRENDIZAJE, BASADO EN LA METODOLOGÍA DE DEEP LEARNING, PARA EL RECONOCIMIENTO DE ESTILOS DE APRENDIZAJE QUE MEJOREN EL DESEMPEÑO DE LOS ESTUDIANTES EN LA EDUCACIÓN BÁSICA REGULAR (EBR 2017). , vol. 14, no. 1, ISSN 1662453X.

VINCENT, A. y PALLAS, V., 2020. Fitopatología en tiempos de pandemia. *Fitopatología. Sociedad Española de Fitopatología.*, vol. 5,

WANG, Y., KING, R., HAW, J. y LEUNG, S. on, 2023. What explains Macau students' achievement? An integrative perspective using a machine learning approach (¿Cuál es la explicación del rendimiento de los estudiantes macaenses? Una perspectiva integradora mediante la adopción del enfoque del aprendizaje aut. *Infancia y Aprendizaje* [en línea], vol. 46, no. 1, ISSN 15784126. DOI 10.1080/02103702.2022.2149120. Disponible en: https://doi.org/10.1080/02103702.2022.2149120.

ANEXOS

Anexo 1. Matriz de consistencia

<table>
<tr><th>PROBLEMAS</th><th>OBJETIVOS</th><th>HIPÓTESIS</th><th colspan="4">VARIABLE</th><th rowspan="2">MÉTODO</th></tr>
<tr><th>Principal</th><th>General</th><th>General</th><th colspan="4">Independiente</th></tr>
<tr><td>¿Cómo la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial influye en la detección de enfermedades en cultivos de tomate?</td><td>Desarrollar una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate</td><td>El desarrollo de una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial mejora la detección de enfermedades en cultivos de tomate</td><td colspan="4">Aplicación móvil con Reconocimiento de Imágenes Basada en Inteligencia Artificial</td><td rowspan="4">Tipo de investigación: Aplicada – experimental

Diseño de investigación: pre-experimental

Población: Cultivo de tomate de la Provincia de Cañete

Muestra: 30 plantas de cultivo de tomate de la Provincia de Cañete

Técnica de investigación: Fichaje

Instrumento de investigación: Ficha de registro</td></tr>
<tr><th>Secundarios</th><th>Específicos</th><th>Específicos</th><th>Dependiente</th><th>DIMENSIONES</th><th>INDICADORES</th><th>FÓRMULA</th></tr>
<tr><td>¿Cómo la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial influye en el tiempo promedio para identificar plagas y enfermedades en cultivos de tomate?</td><td>Determinar de qué forma una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial influye en el tiempo promedio para identificar plagas y enfermedades en cultivos de tomate.</td><td>Una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial disminuye el tiempo promedio para identificar plagas y enfermedades en cultivos de tomate (Tamayo et al., 2024).</td><td rowspan="2">Detección de Enfermedades en Cultivos de Tomate (Salas et al., 2022)</td><td rowspan="2">Control eficiente (Cusme Zambrano y Loor Pinargote 2019)</td><td>Tiempo promedio para identificar plagas y enfermedades en el cultivo (Piscoya 2019, pp. 30)</td><td>$\frac{\sum tiempos\ por\ consulta}{Total\ de\ mediciones}$</td></tr>
<tr><td>¿Cómo la aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial influye en el porcentaje de sensibilidad en la detección de enfermedades en cultivos de tomate?</td><td>Determinar de qué forma una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial influye en el porcentaje de sensibilidad en la detección de enfermedades en cultivos de tomate</td><td>Una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial mejora el porcentaje de sensibilidad en la detección de enfermedades en cultivos de tomate (Ahmed y Harshavardhan Reddy 2021).</td><td>%Sensibilidad (Recall): proporción de enfermedades presentes que fueron correctamente detectadas (Sainz, Peña, Hernández 2017, pp. 5)</td><td>$\frac{\#\mathrm{ECD}}{\#\mathrm{TEP}}$

Dónde:
#ECD:
número de enfermedades correctamente detectadas
#TEP:
número total de enfermedades presentes</td></tr>
</table>

Anexo 2. Variable de consistencia

Variables de estudio	Definición conceptual	Definición operacional	Dimensiones	Indicadores	Escala de medición
Detección de Enfermedades en Cultivos de Tomate (Salas et al., 2022)	Se refiere al proceso de identificar y diagnosticar de manera precisa y oportuna la presencia de patógenos o agentes causantes de enfermedades que puedan afectar el crecimiento y desarrollo de las plantas de tomate (Hernández 2021)	Este proceso implica el monitoreo constante del cultivo, la observación minuciosa de los síntomas que presentan las plantas, la recolección de muestras y su análisis en laboratorio para determinar la enfermedad específica que las está afectando. Una detección temprana y precisa de las enfermedades es fundamental para implementar medidas de control y tratamientos adecuados, evitando así la propagación de la enfermedad y minimizando las pérdidas en el rendimiento y la calidad del cultivo (Piscoya, 2019)	Control eficiente (Cusme Zambrano y Loor Pinargote 2019)	Tiempo promedio para identificar plagas y enfermedades en el cultivo (Piscoya 2019, pp. 30)	Ordinal (Arias et al., 2016)
				%Sensibilidad (Recall): proporción de enfermedades presentes que fueron correctamente detectadas (Sainz, Peña, Hernández 2017, pp. 5)	Ordinal (Arias et al., 2016)

Anexo 3. Instrumento de recolección de datos

Ficha de Registro			
Investigador	Luis Humberto Olazo Luján	**Tipo de prueba:**	
Empresa	No aplica		
Variable	Detección de Enfermedades en Cultivos de Tomate		
Dimensión	Control Eficiente		
Periodo			

Indicador	Descripción	Técnica	Unidad de Medida	Fórmula
Tiempo Promedio de Identificación	Tiempo promedio para identificar plagas y enfermedades en el cultivo	FICHAJE	Minutos	$\frac{\sum TC}{TM}$
				∑ TC = Suma de los tiempos de cada consulta
				TM= Total mediciones

Parcela	Planta	Fecha	∑ TC	TM	Tiempo Promedio de Identificación
1	1				
	2				
	3				
	4				
	5				
	6				
	7				
	8				
	9				
	10				
2	11				
	12				
	13				
	14				
	15				
	16				
	17				
	18				
	19				
	20				
3	21				
	22				
	23				
	24				
	25				
	26				
	27				
	28				
	29				
	30				

Ficha de Registro			
Investigador	Luis Humberto Olazo Luján	**Tipo de prueba:**	
Empresa	No aplica		
Variable	Detección de Enfermedades en Cultivos de Tomate		
Dimensión	Control Eficiente		
Periodo			

Indicador	Descripción	Técnica	Unidad de Medida	Fórmula
Sensibilidad (Recall)	Proporción de enfermedades presentes correctamente identificadas	FICHAJE	%	$\frac{\#ECD}{\#TEP}$
				#ECD = Número de enfermedades correctamente detectadas
				#TEP = Número total de enfermedades presentes.

Parcela	Planta	Fecha	#ECD	#TEP	Sensibilidad (Recall)
1	1				
	2				
	3				
	4				
	5				
	6				
	7				
	8				
	9				
	10				
2	11				
	12				
	13				
	14				
	15				
	16				
	17				
	18				
	19				
	20				
3	21				
	22				
	23				
	24				
	25				
	26				
	27				
	28				
	29				
	30				

Anexo 4. Fichas de validación de instrumentos para la recolección de datos

UNIVERSIDAD CÉSAR VALLEJO

TABLA DE EVALUACIÓN DE EXPERTOS

Apellidos y Nombres del Experto: Víctor Melquiades Cruz Martínez.

Título y/o grado: Ing. Agrónomo

Centro donde labora: AGRO CAMPO SUR.

Fecha: 13/06/2024

TÍTULO DEL PROYECTO

"Aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate"

Tabla evaluación de Expertos para el indicador: Tiempo promedio para identificar plagas y enfermedades en el cultivo. Mediante la siguiente tabla de evaluación de expertos, usted tiene potestad de calificar los instrumentos que son utilizados para medir el indicador, mediante una serie de criterios marcando un valor en cada columna. Así como también, disponemos en la corrección de los ítems indicando sus observaciones y/o sugerencias.

ITEM	CRITERIOS	VALORACIONES				
		Deficiente 0-20%	Regular 21-50%	Bueno 51-20%	Muy Bueno 71-80%	Excelente 81-100%
Claridad	Cuenta con un lenguaje apropiado					85%
Objetividad	Está expresado en conducta observable					85%
Organización	Es adecuado a la vanguardia de la tecnología					85%
Suficiencia	Comprende los aspectos de cantidad y calidad					85%
Intencionalidad	Adecuado para valorar los aspectos del sistema metodológico y científico					85%
Consistencia	Está basado en aspectos técnicos, científicos acordes a la tecnología adecuada					85%
Coherencia	Entre los índices, indicadores y dimensiones					85%
Metodología	Responde el propósito del trabajo bajo los objetivos a lograr					85%
Pertinencia	El instrumento es adecuado al tipo de investigación					85%

Promedio de Valoración: _____

Opción de aplicabilidad:

(X) El instrumento puede ser aplicado, tal como está elaborado.

( ) El instrumento debe ser mejorado, antes de ser aplicado.

Firma de Experto

UNIVERSIDAD CÉSAR VALLEJO

TABLA DE EVALUACIÓN DE EXPERTOS

Apellidos y Nombres del Experto: Víctor Malquiades Cruz Martínez.

Título y/o grado: Ing. Agrónomo

Centro donde labora: A680 CAMPO SUR.

Fecha: 13/06/2024

TÍTULO DEL PROYECTO

"Aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate"

Tabla evaluación de Expertos para el indicador %Sensibilidad (Recall). Mediante la siguiente tabla de evaluación de expertos, usted tiene potestad de calificar los instrumentos que son utilizados para medir el indicador, mediante una serie de criterios marcando un valor en cada columna. Así como también, disponemos en la corrección de los ítems indicando sus observaciones y/o sugerencias.

ITEM	CRITERIOS	VALORACIONES				
		Deficiente 0-20%	Regular 21-50%	Bueno 51-20%	Muy Bueno 71-80%	Excelente 81-100%
Claridad	Cuenta con un lenguaje apropiado					85%
Objetividad	Está expresado en conducta observable					85%
Organización	Es adecuado a la vanguardia de la tecnología					85%
Suficiencia	Comprende los aspectos de cantidad y calidad					85%
Intencionalidad	Adecuado para valorar los aspectos del sistema metodológico y científico					85%
Consistencia	Está basado en aspectos técnicos, científicos acordes a la tecnología adecuada					85%
Coherencia	Entre los índices, indicadores y dimensiones					85%
Metodología	Responde el propósito del trabajo bajo los objetivos a lograr					85%
Pertinencia	El instrumento es adecuado al tipo de investigación					85%

Promedio de Valoración: _____

Opción de aplicabilidad:

(X) El instrumento puede ser aplicado, tal como está elaborado.

( ) El instrumento debe ser mejorado, antes de ser aplicado.

Firma de Experto

UNIVERSIDAD CÉSAR VALLEJO

TABLA DE EVALUACIÓN DE EXPERTOS

Apellidos y Nombres del Experto: ZUÑIGA VASQUEZ NESTOR GIANKELER

Título y/o grado: MAGISTER

Universidad donde labora: UCV

Fecha: 14/06/2024

TÍTULO DEL PROYECTO

"Aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate"

Tabla evaluación de Expertos para el indicador: Tiempo promedio para identificar plagas y enfermedades en el cultivo. Mediante la siguiente tabla de evaluación de expertos, usted tiene potestad de calificar los instrumentos que son utilizados para medir el indicador, mediante una serie de criterios marcando un valor en cada columna. Así como también, disponemos en la corrección de los ítems indicando sus observaciones y/o sugerencias.

ITEM	CRITERIOS	VALORACIONES				
		Deficiente 0-20%	Regular 21-50%	Bueno 51-20%	Muy Bueno 71-80%	Excelente 81-100%
Claridad	Cuenta con un lenguaje apropiado					X
Objetividad	Está expresado en conducta observable				X	
Organización	Es adecuado a la vanguardia de la tecnología					X
Suficiencia	Comprende los aspectos de cantidad y calidad					X
Intencionalidad	Adecuado para valorar los aspectos del sistema metodológico y científico					X
Consistencia	Está basado en aspectos técnicos, científicos acordes a la tecnología adecuada					X
Coherencia	Entre los índices, indicadores y dimensiones					X
Metodología	Responde el propósito del trabajo bajo los objetivos a lograr					X
Pertinencia	El instrumento es adecuado al tipo de investigación					X

Promedio de Valoración: _____

Opción de aplicabilidad:

(X) El instrumento puede ser aplicado, tal como está elaborado.

( ) El instrumento debe ser mejorado, antes de ser aplicado.

Firma de Experto

UNIVERSIDAD CÉSAR VALLEJO

TABLA DE EVALUACIÓN DE EXPERTOS

Apellidos y Nombres del Experto: Burga Vasquez Nestor Giankeiler

Título y/o grado: Magister

Universidad donde labora: UCV

Fecha: 14/06/2024

TÍTULO DEL PROYECTO

"Aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate"

Tabla evaluación de Expertos para el indicador %Sensibilidad (Recall). Mediante la siguiente tabla de evaluación de expertos, usted tiene potestad de calificar los instrumentos que son utilizados para medir el indicador, mediante una serie de criterios marcando un valor en cada columna. Así como también, disponemos en la corrección de los ítems indicando sus observaciones y/o sugerencias.

ITEM	CRITERIOS	VALORACIONES				
		Deficiente 0-20%	Regular 21-50%	Bueno 51-20%	Muy Bueno 71-80%	Excelente 81-100%
Claridad	Cuenta con un lenguaje apropiado					X
Objetividad	Está expresado en conducta observable					X
Organización	Es adecuado a la vanguardia de la tecnología					X
Suficiencia	Comprende los aspectos de cantidad y calidad					X
Intencionalidad	Adecuado para valorar los aspectos del sistema metodológico y científico					X
Consistencia	Está basado en aspectos técnicos, científicos acordes a la tecnología adecuada					X
Coherencia	Entre los índices, indicadores y dimensiones					X
Metodología	Responde el propósito del trabajo bajo los objetivos a lograr					X
Pertinencia	El instrumento es adecuado al tipo de investigación					X

Promedio de Valoración: _____

Opción de aplicabilidad:

(X) El instrumento puede ser aplicado, tal como está elaborado.

( ) El instrumento debe ser mejorado, antes de ser aplicado.

Firma de Experto

UNIVERSIDAD CÉSAR VALLEJO

TABLA DE EVALUACIÓN DE EXPERTOS

Apellidos y Nombres del Experto: Tito Chura Virgilio Fredy

Título y/o grado: Maestro en Ingeniería de Seguridad Informática

Universidad donde labora: UCV

Fecha: 14/06/2024

TÍTULO DEL PROYECTO

"Aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate"

Tabla evaluación de Expertos para el indicador: Tiempo promedio para identificar plagas y enfermedades en el cultivo. Mediante la siguiente tabla de evaluación de expertos, usted tiene potestad de calificar los instrumentos que son utilizados para medir el indicador, mediante una serie de criterios marcando un valor en cada columna. Así como también, disponemos en la corrección de los ítems indicando sus observaciones y/o sugerencias.

ITEM	CRITERIOS	VALORACIONES				
		Deficiente 0-20%	Regular 21-50%	Bueno 51-20%	Muy Bueno 71-80%	Excelente 81-100%
Claridad	Cuenta con un lenguaje apropiado					85%
Objetividad	Está expresado en conducta observable					85%
Organización	Es adecuado a la vanguardia de la tecnología					85%
Suficiencia	Comprende los aspectos de cantidad y calidad					85%
Intencionalidad	Adecuado para valorar los aspectos del sistema metodológico y científico					85%
Consistencia	Está basado en aspectos técnicos, científicos acordes a la tecnología adecuada					85%
Coherencia	Entre los índices, indicadores y dimensiones					85%
Metodología	Responde el propósito del trabajo bajo los objetivos a lograr					85%
Pertinencia	El instrumento es adecuado al tipo de investigación					85%

Promedio de Valoración: _____

Opción de aplicabilidad:

(X) El instrumento puede ser aplicado, tal como está elaborado.

( ) El instrumento debe ser mejorado, antes de ser aplicado.

Firma de Experto

UNIVERSIDAD CÉSAR VALLEJO

TABLA DE EVALUACIÓN DE EXPERTOS

Apellidos y Nombres del Experto: Tito Chura Virgilio Fredy

Título y/o grado: Maestro en Ingeniería de Seguridad Informática

Universidad donde labora: UCV

Fecha: 14/06/2024

TÍTULO DEL PROYECTO

"Aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate"

Tabla evaluación de Expertos para el indicador %Sensibilidad (Recall). Mediante la siguiente tabla de evaluación de expertos, usted tiene potestad de calificar los instrumentos que son utilizados para medir el indicador, mediante una serie de criterios marcando un valor en cada columna. Así como también, disponemos en la corrección de los ítems indicando sus observaciones y/o sugerencias.

ITEM	CRITERIOS	VALORACIONES				
		Deficiente 0-20%	Regular 21-50%	Bueno 51-20%	Muy Bueno 71-80%	Excelente 81-100%
Claridad	Cuenta con un lenguaje apropiado					85%
Objetividad	Está expresado en conducta observable					85%
Organización	Es adecuado a la vanguardia de la tecnología					85%
Suficiencia	Comprende los aspectos de cantidad y calidad					85%
Intencionalidad	Adecuado para valorar los aspectos del sistema metodológico y científico					85%
Consistencia	Está basado en aspectos técnicos, científicos acordes a la tecnología adecuada					85%
Coherencia	Entre los índices, indicadores y dimensiones					85%
Metodología	Responde el propósito del trabajo bajo los objetivos a lograr					85%
Pertinencia	El instrumento es adecuado al tipo de investigación					85%

Promedio de Valoración: _____

Opción de aplicabilidad:

(X) El instrumento puede ser aplicado, tal como está elaborado.

( ) El instrumento debe ser mejorado, antes de ser aplicado.

Firma de Experto

UNIVERSIDAD CÉSAR VALLEJO

TABLA DE EVALUACIÓN DE EXPERTOS

Apellidos y Nombres del Experto: Gálvez Tapia Orleans Moisés

Título y/o grado: Magíster

Universidad donde labora: Universidad César Vallejo

Fecha: 02/07/2024

TÍTULO DEL PROYECTO

"Aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate"

Tabla evaluación de Expertos para el indicador: Tiempo promedio para identificar plagas y enfermedades en el cultivo. Mediante la siguiente tabla de evaluación de expertos, usted tiene potestad de calificar los instrumentos que son utilizados para medir el indicador, mediante una serie de criterios marcando un valor en cada columna. Así como también, disponemos en la corrección de los ítems indicando sus observaciones y/o sugerencias.

ITEM	CRITERIOS	VALORACIONES				
		Deficiente 0-20%	Regular 21-50%	Bueno 51-20%	Muy Bueno 71-80%	Excelente 81-100%
Claridad	Cuenta con un lenguaje apropiado					90%
Objetividad	Está expresado en conducta observable					90%
Organización	Es adecuado a la vanguardia de la tecnología					85%
Suficiencia	Comprende los aspectos de cantidad y calidad					90%
Intencionalidad	Adecuado para valorar los aspectos del sistema metodológico y científico					85%
Consistencia	Está basado en aspectos técnicos, científicos acordes a la tecnología adecuada					90%
Coherencia	Entre los índices, indicadores y dimensiones					90%
Metodología	Responde el propósito del trabajo bajo los objetivos a lograr					90%
Pertinencia	El instrumento es adecuado al tipo de investigación					95%

Promedio de Valoración: _____

Opción de aplicabilidad:

(X) El instrumento puede ser aplicado, tal como está elaborado.

( ) El instrumento debe ser mejorado, antes de ser aplicado.

Firma de Experto

UNIVERSIDAD CÉSAR VALLEJO

TABLA DE EVALUACIÓN DE EXPERTOS

Apellidos y Nombres del Experto: Gálvez Tapia Orleans Moisés

Título y/o grado: Magíster

Universidad donde labora: Universidad César Vallejo

Fecha: 02/07/2024

TÍTULO DEL PROYECTO

"Aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate"

Tabla evaluación de Expertos para el indicador %Sensibilidad (Recall). Mediante la siguiente tabla de evaluación de expertos, usted tiene potestad de calificar los instrumentos que son utilizados para medir el indicador, mediante una serie de criterios marcando un valor en cada columna. Así como también, disponemos en la corrección de los ítems indicando sus observaciones y/o sugerencias.

ITEM	CRITERIOS	VALORACIONES				
		Deficiente 0-20%	Regular 21-50%	Bueno 51-20%	Muy Bueno 71-80%	Excelente 81-100%
Claridad	Cuenta con un lenguaje apropiado					85%
Objetividad	Está expresado en conducta observable					90%
Organización	Es adecuado a la vanguardia de la tecnología					85%
Suficiencia	Comprende los aspectos de cantidad y calidad					85%
Intencionalidad	Adecuado para valorar los aspectos del sistema metodológico y científico					90%
Consistencia	Está basado en aspectos técnicos, científicos acordes a la tecnología adecuada					85%
Coherencia	Entre los índices, indicadores y dimensiones					90%
Metodología	Responde el propósito del trabajo bajo los objetivos a lograr					85%
Pertinencia	El instrumento es adecuado al tipo de investigación					95%

Promedio de Valoración: _____

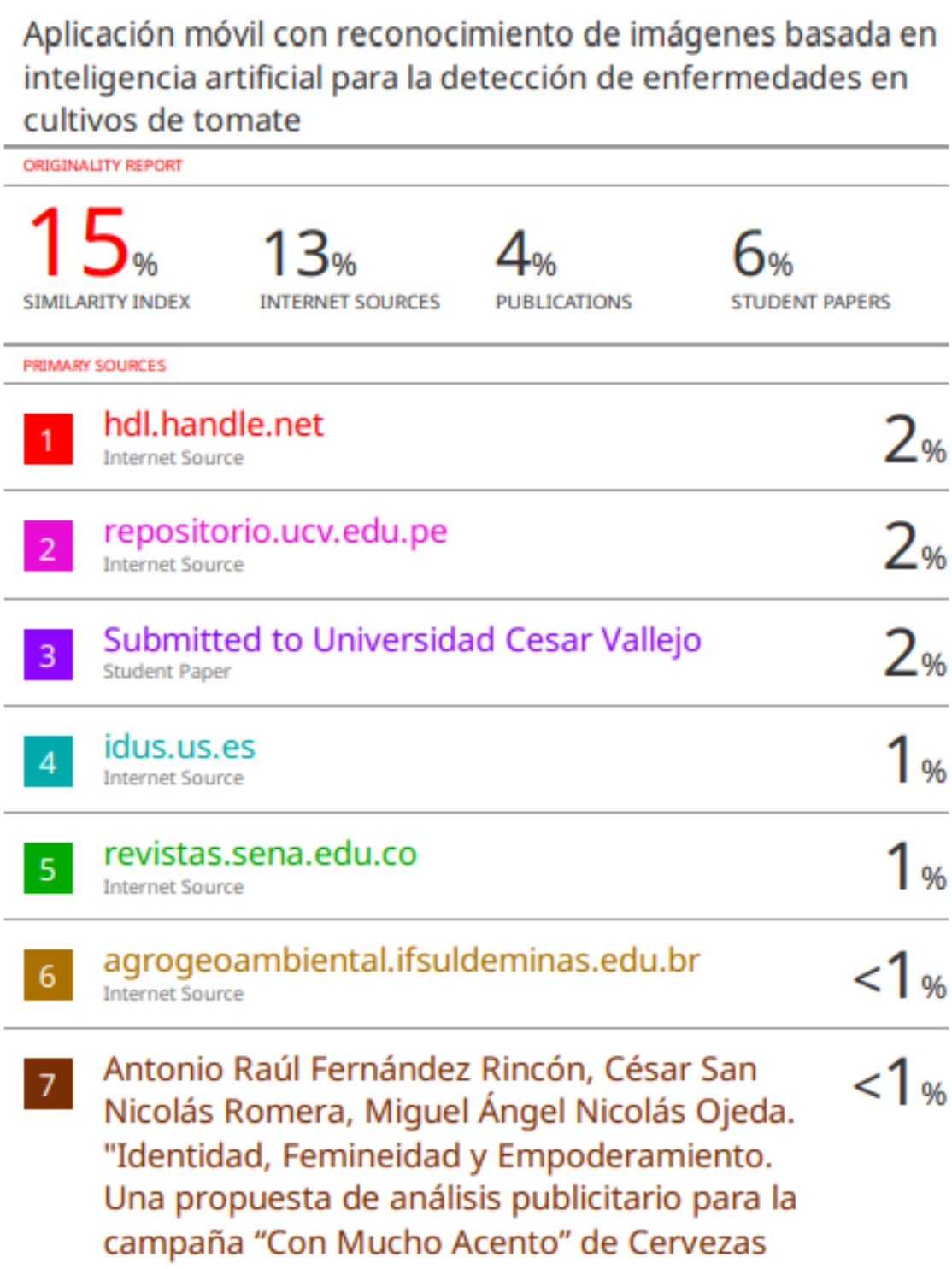
Opción de aplicabilidad:

(X) El instrumento puede ser aplicado, tal como está elaborado.

( ) El instrumento debe ser mejorado, antes de ser aplicado.

Firma de Experto

Anexo 5. Reporte de similitud en software Turnitin.



Aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate

ORIGINALITY REPORT

15%	13%	4%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	hdl.handle.net Internet Source	2%
2	repositorio.ucv.edu.pe Internet Source	2%
3	Submitted to Universidad Cesar Vallejo Student Paper	2%
4	idus.us.es Internet Source	1%
5	revistas.sena.edu.co Internet Source	1%
6	agrogeoambiental.ifsuldeminas.edu.br Internet Source	<1%
7	Antonio Raúl Fernández Rincón, César San Nicolás Romera, Miguel Ángel Nicolás Ojeda. "Identidad, Femineidad y Empoderamiento. Una propuesta de análisis publicitario para la campaña “Con Mucho Acento” de Cervezas	<1%

Anexo 6. Autorizaciones para el desarrollo del proyecto de investigación

AUTORIZACIÓN DE APLICACIÓN

Sr. Víctor Melquiades Cruz Martínez
Ciudadano del Distrito de Imperial, Provincia de Cañete
Ingeniero Agrónomo

De mi consideración:

Es grato dirigirme a usted para saludarlo y, a la vez, solicitarle permiso de acceso e información sobre su chacra o zona de cultivo ubicada en el distrito de Nuevo Imperial, Provincia de Cañete, para poder desarrollar el proyecto denominado "Aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate", con el objetivo general: "Detectar enfermedades en cultivos de tomate a través de la implementación de una aplicación móvil de reconocimiento de imágenes basada en inteligencia artificial", de la experiencia curricular Desarrollo del Proyecto de Investigación (TESIS II), de la Universidad César Vallejo, a cargo del estudiante:

Apellidos y Nombres	Ciclo	Teléfono	Correo
Olazo Luján, Luis Humberto DNI: 70872984	X	937387155	vanskarner@ucvvirtual.edu.pe

Lima, 27 de mayo del 2024

Atentamente

Víctor Melquiades Cruz Martínez
Ciudadano del Distrito de Imperial, Provincia de Cañete
Ingeniero Agrónomo

Anexo 7. Otras evidencias

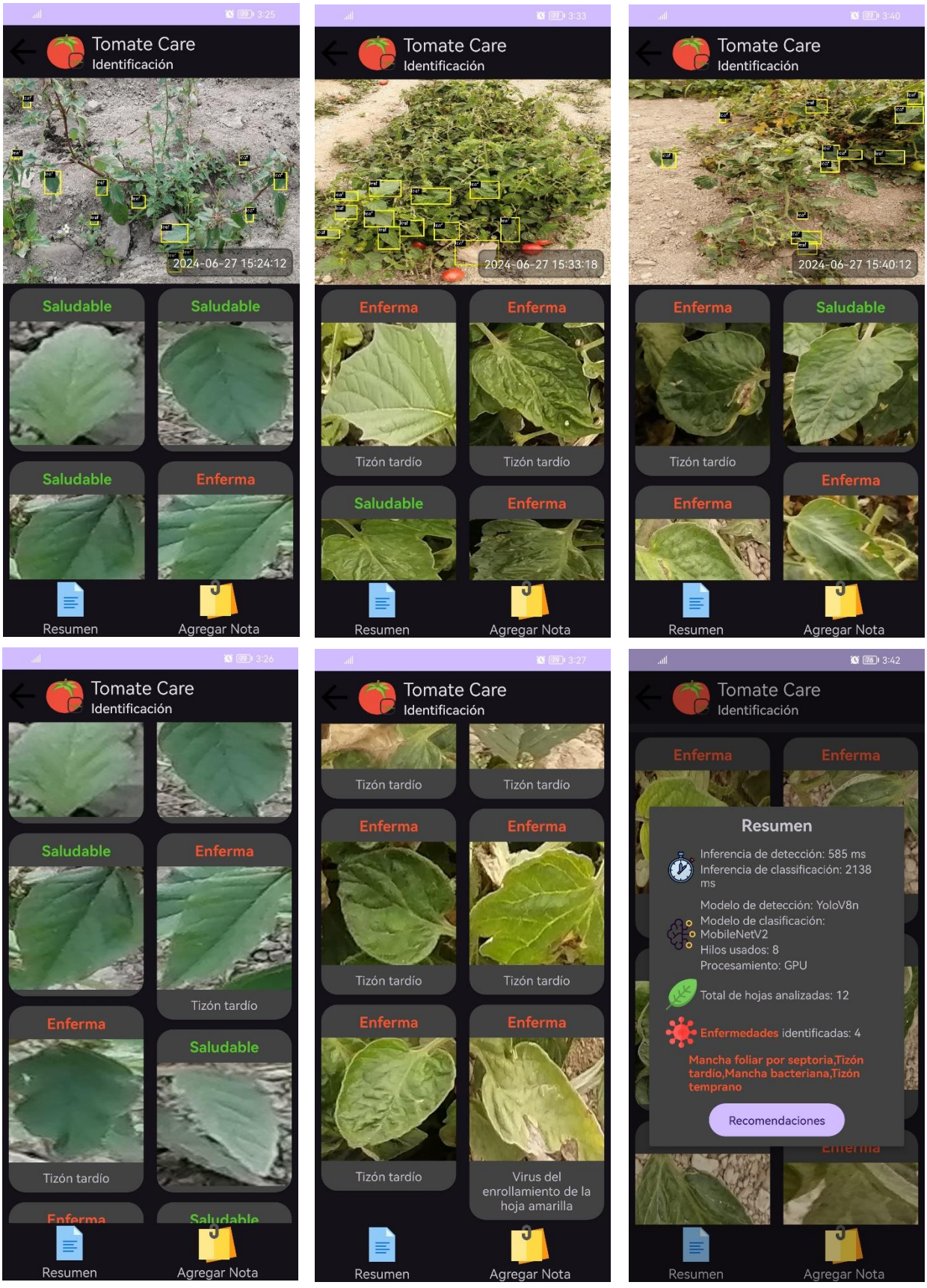
Visita al distrito de Imperial - Cañete

Validación de instrumento con agrónomo

Lugar para procesamiento de datos en Nuevo Imperial - Cañete

Cultivos de tomate

Uso de aplicación en campo.



Obtención de datos para desarrollo de los modelos de clasificación.

Tabla 9

Imágenes para los modelos de clasificación

Investigación	Fuente	Imágenes en total	Imágenes usadas	Disponibilidad
Identification of Plant Leaf Diseases Using a Nine-Layer Deep Convolutional Neural Network (2019)	ProQuest	55448	18160	https://data.mendeley.com/datasets/tywbtsjrjv/1
“Tomato-Village”: a dataset for end-to-end tomato disease detection in a real-world environment (2023)	ProQuest	20021	1616	https://github.com/mamta-joshi-gehlot/Tomato-Village

Fuente: Elaboración propia

Enlaces a productos finales

Tabla 10

Productos finales de investigación

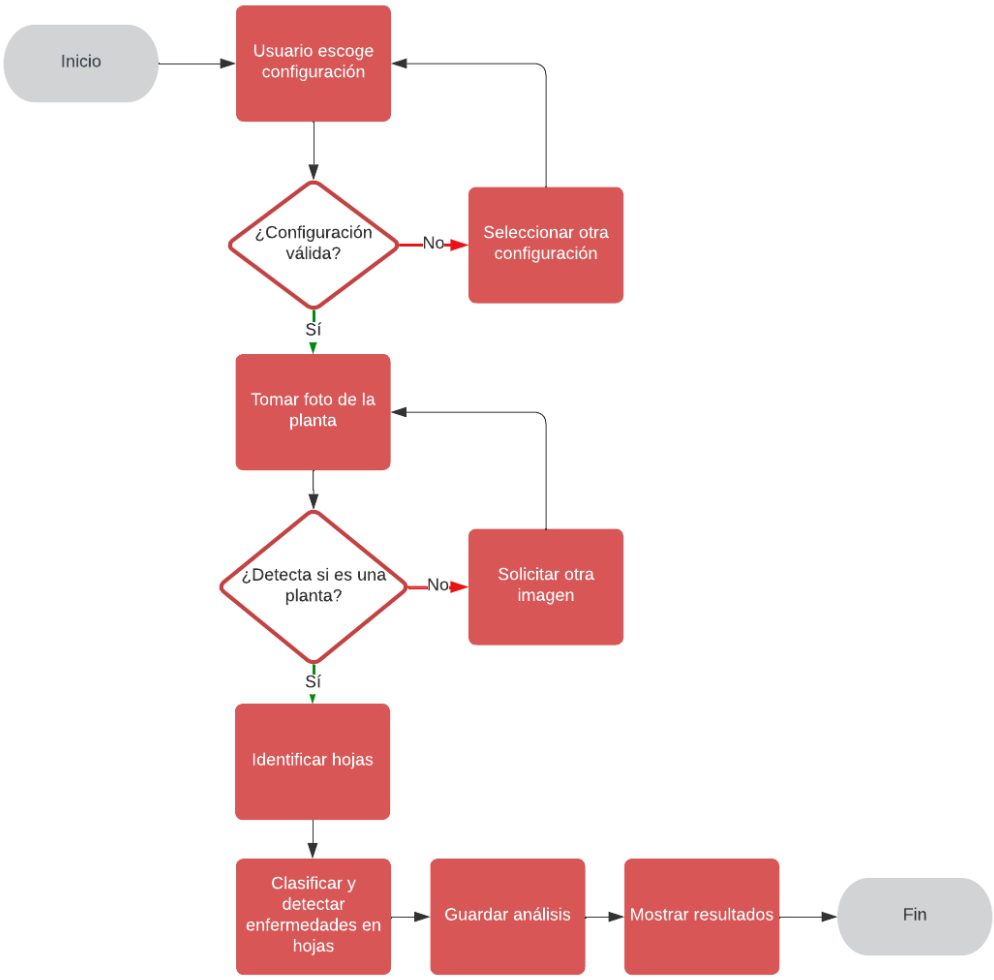
Investigación	Tipo	Disponibilidad
Aplicación Android	Repositorio público GitHub	https://github.com/vanskarner/TomateCare
Dataset	Dataset público Kaggle	https://www.kaggle.com/datasets/luisolazo/tomato-diseases
Modelos predictivos	Notebooks públicos Kaggle	https://www.kaggle.com/code/luisolazo/leaf-detection-w-ultralytics-yolov8-and-tflite
		https://www.kaggle.com/code/luisolazo/tomato-disease-prediction-squeezenet-mish-97-3
		https://www.kaggle.com/code/luisolazo/tomato-disease-prediction-mobilenetv3large-97-1
		https://www.kaggle.com/code/luisolazo/tomato-disease-prediction-mobilenetv3small-96-6

		https://www.kaggle.com/code/luisolazo/tomato-disease-prediction-mobilenetv2-93-8
		https://www.kaggle.com/code/luisolazo/tomato-disease-prediction-nasnetmobile-90-2

Fuente: Elaboración propia

Figura 34

Diagrama de flujo para la identificación de enfermedades en la aplicación.



Desarrollo de la metodología Mobile-D

1. Fase I: Exploración

Como primera etapa, se establecen los requerimientos y el alcance del proyecto que servirán para un desarrollo adecuado.

1.1. Establecimiento de los Stakeholder

Para el desarrollo de este proyecto se definió a las siguientes personas interesadas e involucradas.

- Jefe de Proyecto: responsable del proyecto en todas las actividades a realizar dentro del proyecto para poder entregar un producto de calidad.
- Desarrollador: responsable del diseño personalizado y desarrollo de la app.
- Agricultor: Principales usuarios quienes serán los que usarán la app.

1.2. Alcance

Desarrollar una aplicación móvil con reconocimiento de imágenes basada en inteligencia artificial para la detección de enfermedades en cultivos de tomate.

1.3. Limitaciones

- La aplicación está principalmente diseñada y pensada para los agricultores, sin embargo, también puede ser usada por cualquier usuario que quiera probarla.
- La ejecución de la aplicación solo se realiza en dispositivos Android desde la versión 7.0 en adelante.
- La actualización de los modelos de clasificación de enfermedades requiere conexión a internet.

1.4. Establecimiento del proyecto

Para esta etapa se definió el ambiente técnico y físico en el cual se desarrollará el proyecto. Las herramientas necesarias para el desarrollo del proyecto son las siguientes:

- Lenguaje de programación: Python 3.10, Kotlin 1.8
- Framework: Tensorflow 2.15, Keras 2.15, Ultralytics 8.2
- IDE: Android Studio Koala, Visual Studio 2024
- Plataformas de desarrollo: Kaggle (para IA)
- Herramientas para prototipos UI: Figma
- Sistema de gestión de base de datos: SQLite
- Sistema operativo: Android versión 7.0 o superior
- Equipos: Laptop Asus FX505DT y celular Huawei P30 Lite
- Metodología de Desarrollo: Mobile-D

1.5. Requerimientos iniciales

Se pretende desarrollar una aplicación móvil con reconocimiento de imágenes basado en inteligencia artificial que permita capturar la foto de la planta de tomate para identificar las hojas, segmentarlas individualmente y detectar la presencia de enfermedades. Además, facilitará el registro de las actividades realizadas.

Tabla 11

Requerimientos Funcionales

Código	Descripción
RF1	La aplicación permite visualizar un menú principal de múltiples opciones.
RF2	La aplicación permite visualizar detalles y datos sobre la app
RF3	La aplicación permite llevar un registro de la actividad realizada con respecto al análisis de imágenes.
RF4	La aplicación permite medir el tiempo de inferencia que le toma al smartphone sobre la utilización de los modelos de inteligencia artificial seleccionado.
RF5	La aplicación permite visualizar un resumen de las enfermedades que puede identificar.
RF6	La aplicación permite usar la cámara para la captura de la imagen de la planta para la identificación de las hojas, segmentación, clasificación y un registro local.

RF7	La aplicación permite configurar el modelo a utilizar para su efecto en el análisis de la imagen.
RF8	La aplicación debe mostrar información de cómo utilizar la función de captura.
RF9	La aplicación permite visualizar el resultado del análisis y predicciones de la imagen capturada, mostrando la imagen de la planta, las hojas identificadas, su clasificación y un resumen de análisis realizado.

Tabla 12

Requerimientos No Funcionales

Código	Descripción
RFN1	La aplicación será desarrollada para la plataforma Android para la versión 7.0 y superior
RNF2	La base de datos para la aplicación móvil será desarrollada en SQLite
RFN3	La aplicación será desarrollada en el lenguaje Kotlin de forma nativa en el IDE de Android Studio
RFN4	Los modelos de clasificación y detección de objetos serán desarrollados usando Tensorflow 2.
RFN5	La aplicación móvil puede ser accedida por cualquier usuario, no necesita un registro previo.

Tabla 13

Modelos de Procesos de la Aplicación

Módulo	Código	Proceso	Requerimientos	
			Funcionales	No Funcionales
Módulo del Menú Principal	M01	La aplicación permite la visualización de múltiples opciones para el usuario	RF1	RNF2, RNF5

Módulo de Información	M02	La aplicación permite visualizar la información sobre la app y su desarrollador	RF2	RFN1
Módulo de Registros de Actividad	M03	La aplicación permite llevar un registro de los análisis realizados.	RF3	RNF2, RNF5
Módulo de Pruebas de Rendimiento	M04	La aplicación permite realizar un test de velocidad en cuanto a la inferencia de los modelos.	RF4	RFN4,RNF5
Módulo de Enfermedades	M05	La aplicación permite llevar un registro de las enfermedades que puede identificar, así como información para su control.	RF5	RNF2, RNF5
Módulo de Captura de Imágenes	M06	La aplicación permite tomar o escoger una foto para su identificación de enfermedades, así como elegir qué modelo usar y consejos como tomar las fotos	RF6, RF7, RF8	RNF2, RFN4, RNF5
Módulo de Identificación	M07	La aplicación permite la visualización del análisis realizado a la imagen de la planta, mostrando las hojas identificadas, su clasificación, permite	RF9	RNF2, RNF5

		agregar notas, mostrar un resumen sintetizado del análisis y visualizar las recomendaciones.		

2. Fase II: Inicialización

Figura 35
Arquitectura de software

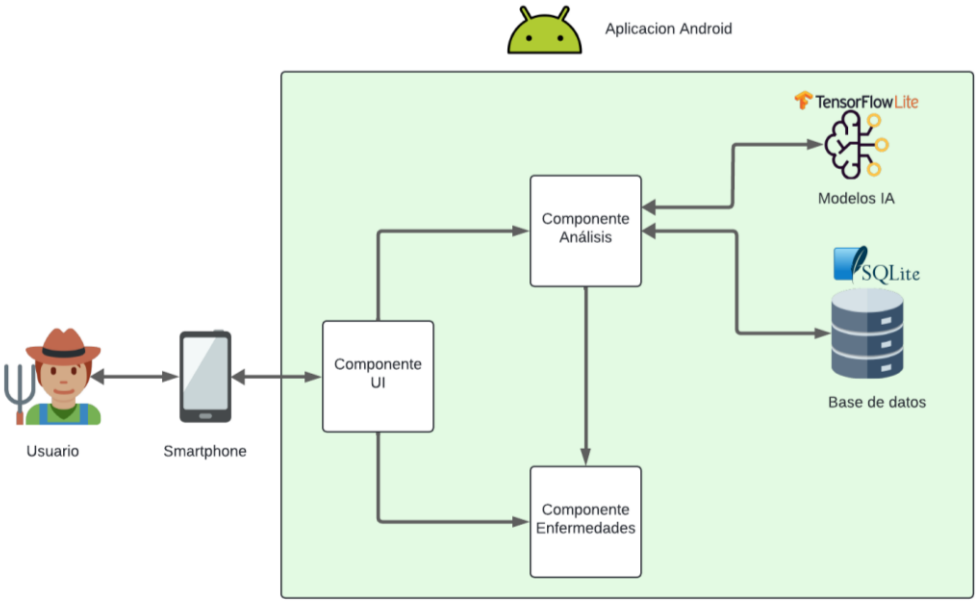


Figura 36
Diagrama de bloques para el desarrollo de modelos deep learning

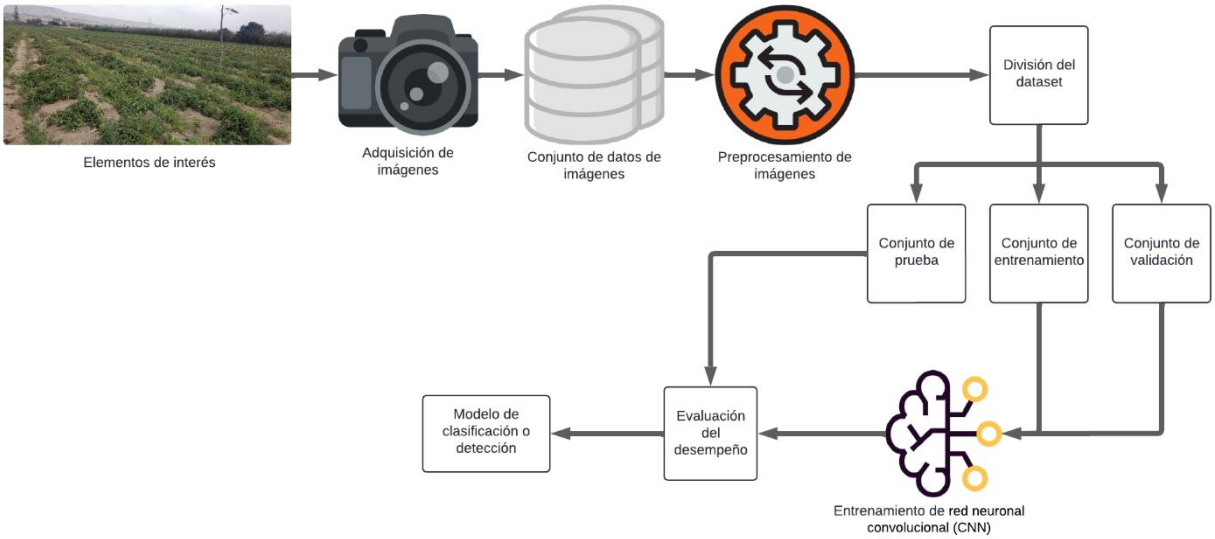
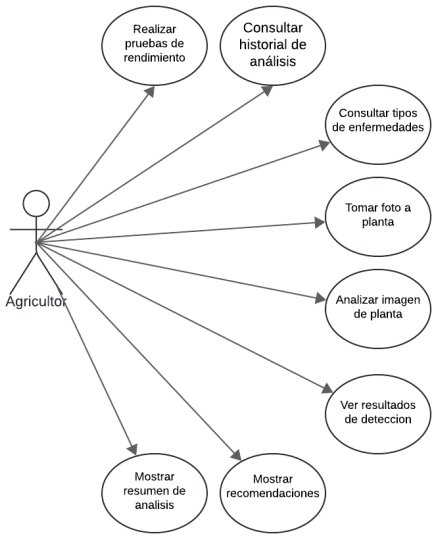


Figura 37
Diagrama de casos de uso general



1.6. Diseño de Interfaz para la aplicación móvil

Tabla 14

Storycard del Menu Screen

Número 1			
Dificultad	Media	**Prioridad**	Media
Descripción			
El usuario visualizara en el menú principal múltiples opciones que le permitirán realizar diferentes acciones. Por defecto se sitúa en la sección de inicio, luego se puede navegar por la sección de enfermedades y la captura de imágenes.			
Excepciones			
Fecha		**Estado**	**Comentario**
26/03/2024		Definido	Paso
29/04/2024		Implementado	Paso
20/05/2024		Prueba	Paso
22/06/2024		Verificado	Paso

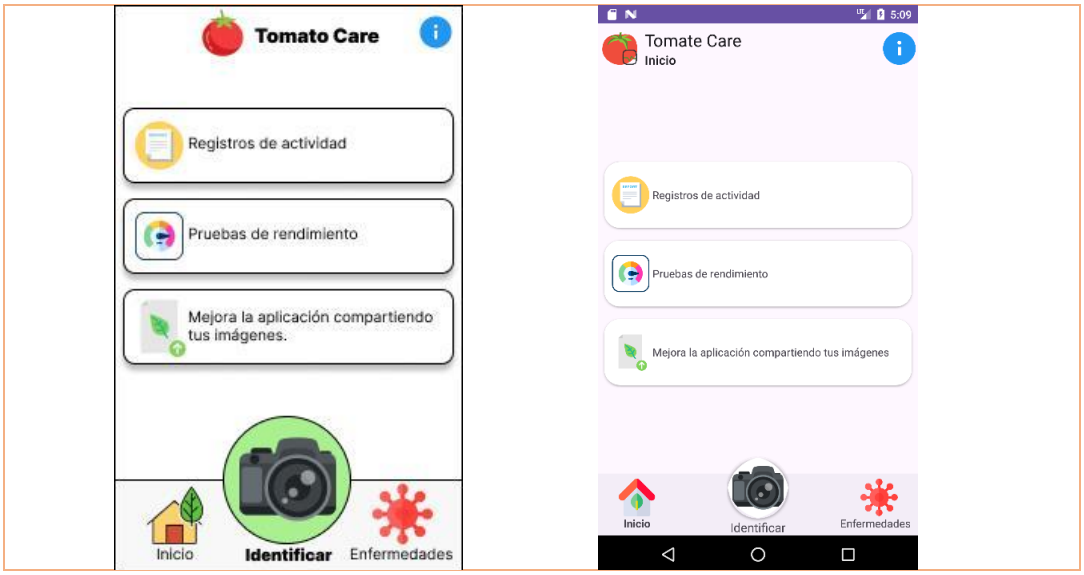


Tabla 15

Storycard del Activity Logs Screen

Número 2			
Dificultad	Media	**Prioridad**	Media
Descripción			
El usuario visualizara el registro de su actividad producto de las fotos que haya analizado, todo guardado de manera local.			
Excepciones			
Fecha	**Estado**	**Comentario**	
26/03/2024	Definido	Paso	
30/04/2024	Implementado	Paso	
20/05/2024	Prueba	Paso	
22/06/2024	Verificado	Paso	

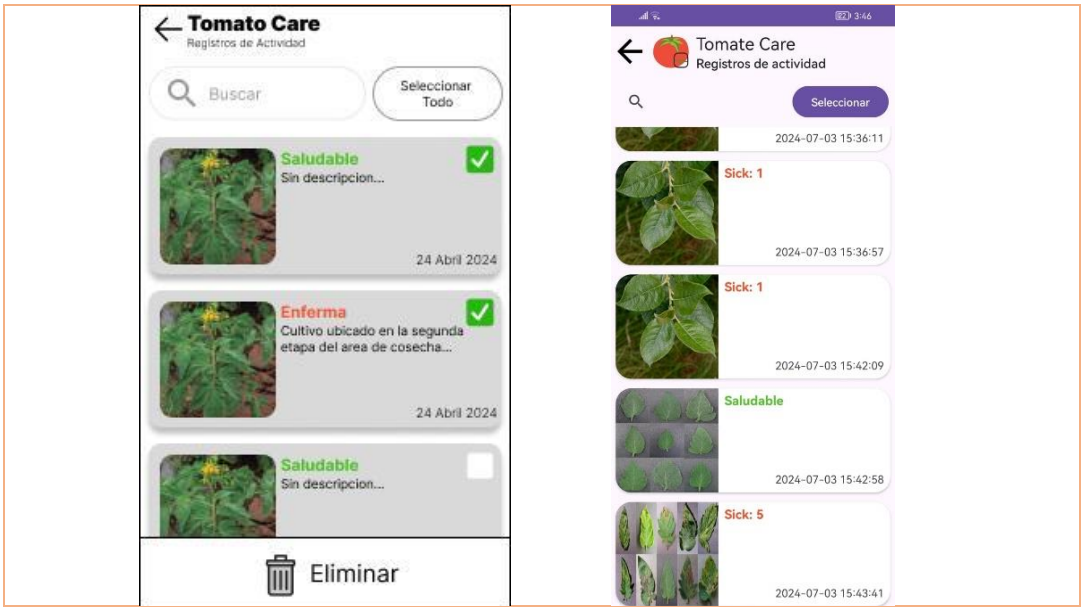


Tabla 16

Storycard del Test Screen

Número 3			
Dificultad	Media	**Prioridad**	Media
Descripción			
El usuario podrá comprobar la capacidad de inferencia de los modelos escogidos en su propio teléfono.			
Excepciones			
Fecha	**Estado**	**Comentario**	
26/03/2024	Definido	Paso	
30/04/2024	Implementado	Paso	
20/05/2024	Prueba	Paso	
22/06/2024	Verificado	Paso	

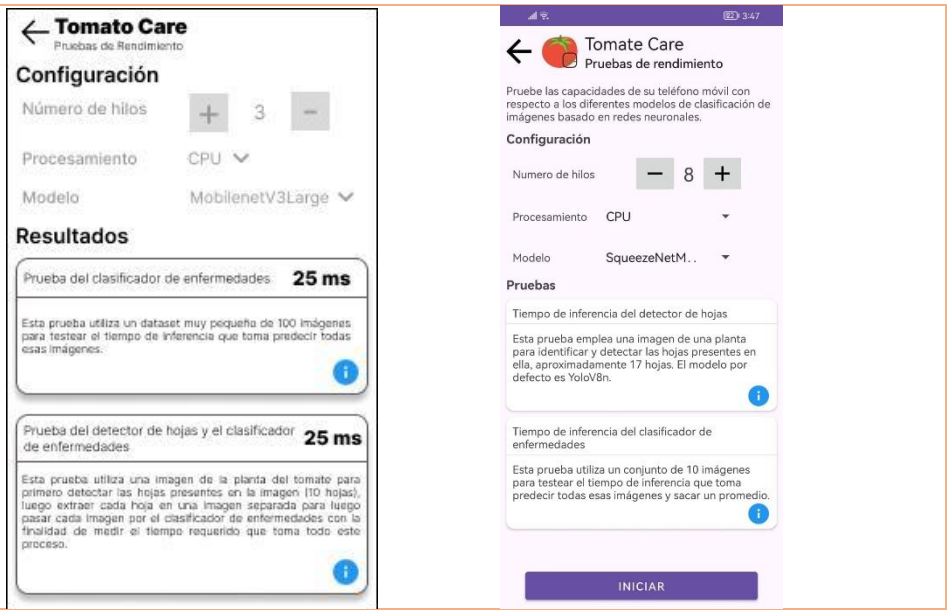


Tabla 17

Storycard del Info Screen

Número 4			
Dificultad	Baja	**Prioridad**	Baja
Descripción			
El usuario visualizara información base sobre la app y su desarrollador			
Excepciones			

Fecha	Estado	Comentario
26/03/2024	Definido	Paso
30/04/2024	Implementado	Paso
20/05/2024	Prueba	Paso
22/06/2024	Verificado	Paso

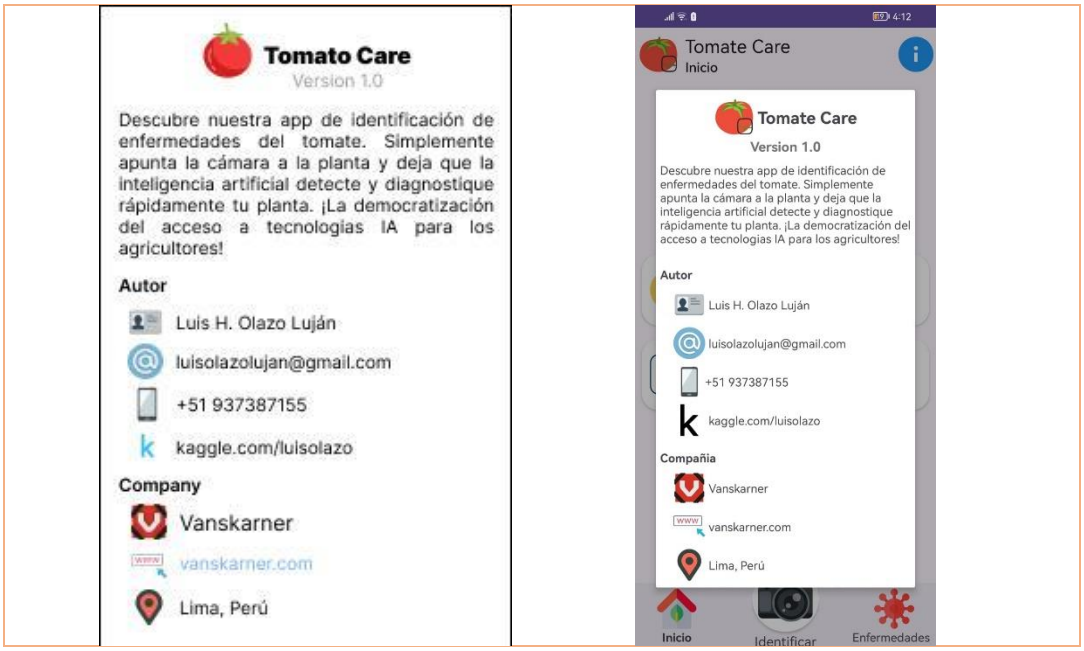


Tabla 18

Storycard del Diseases Screen

Número 5			
Dificultad	Media	**Prioridad**	Media
Descripción			
El usuario visualizara la información de las enfermedades que puede identificar la aplicación.			
Excepciones			
Fecha	**Estado**	**Comentario**	
27/03/2024	Definido	Paso	
30/04/2024	Implementado	Paso	
21/05/2024	Prueba	Paso	
23/06/2024	Verificado	Paso	

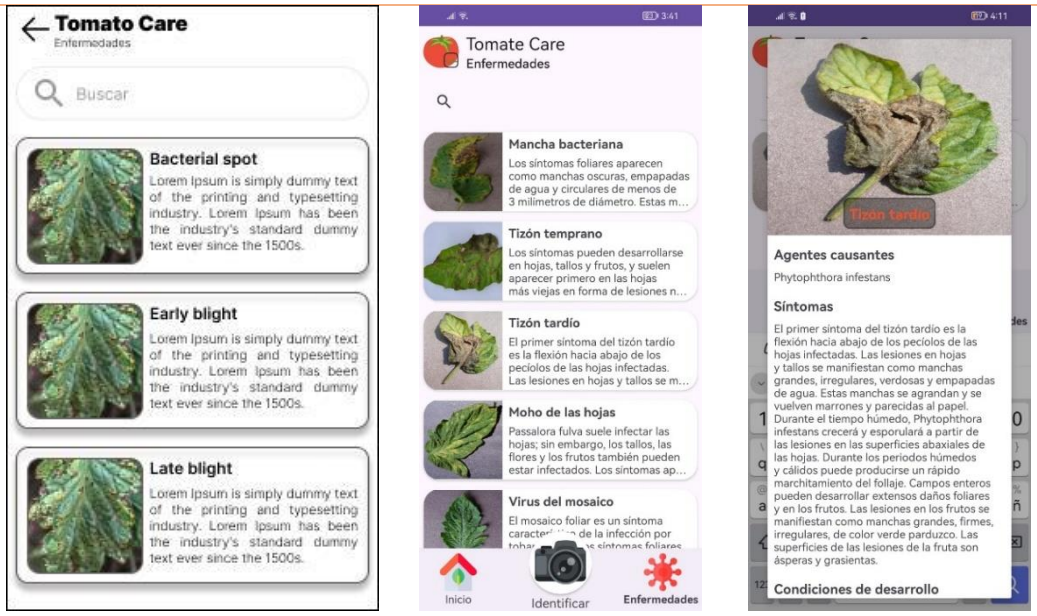


Tabla 19

Storycard del Capture Screen

Número 6			
Dificultad	Alta	**Prioridad**	Alta
Descripción			
El usuario podrá realizar la toma de las fotos o cargarlas desde su galería para su posterior identificación, además puede elegir entre los modelos de su preferencia y consejos de como tomar las fotos.			
Excepciones			
Fecha	**Estado**	**Comentario**	
28/03/2024	Definido	Paso	
01/05/2024	Implementado	Paso	
22/05/2024	Prueba	Paso	
24/06/2024	Verificado	Paso	

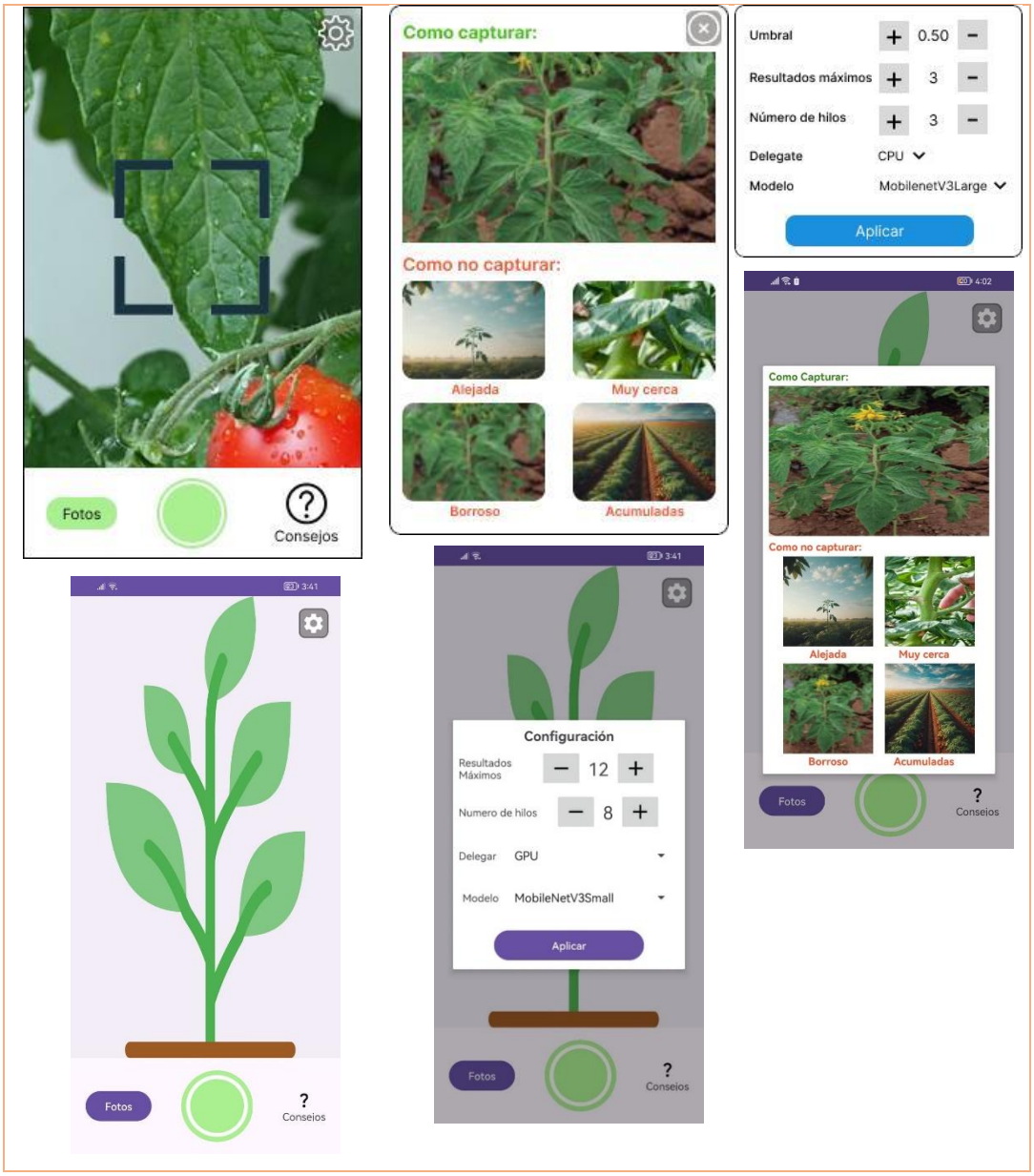


Tabla 20

Storycard del Identification Screen

Número 7			
Dificultad	Alta	**Prioridad**	Alta
Descripción			
El usuario podrá visualizar los resultados del análisis después de tomar las fotos, este análisis identifica las hojas, luego las separa para su clasificación individual y determina si hay presencia de enfermedades en cada una de ellas. Asimismo, se puede agregar notas, ver el resumen sintetizado del análisis y las recomendaciones de ser pertinente.			

Excepciones		
Fecha	**Estado**	**Comentario**
29/03/2024	Definido	Paso
02/05/2024	Implementado	Paso
23/05/2024	Prueba	Paso
25/06/2024	Verificado	Paso

1.7. Base de datos

Figura 38
Modelo lógico de base de datos

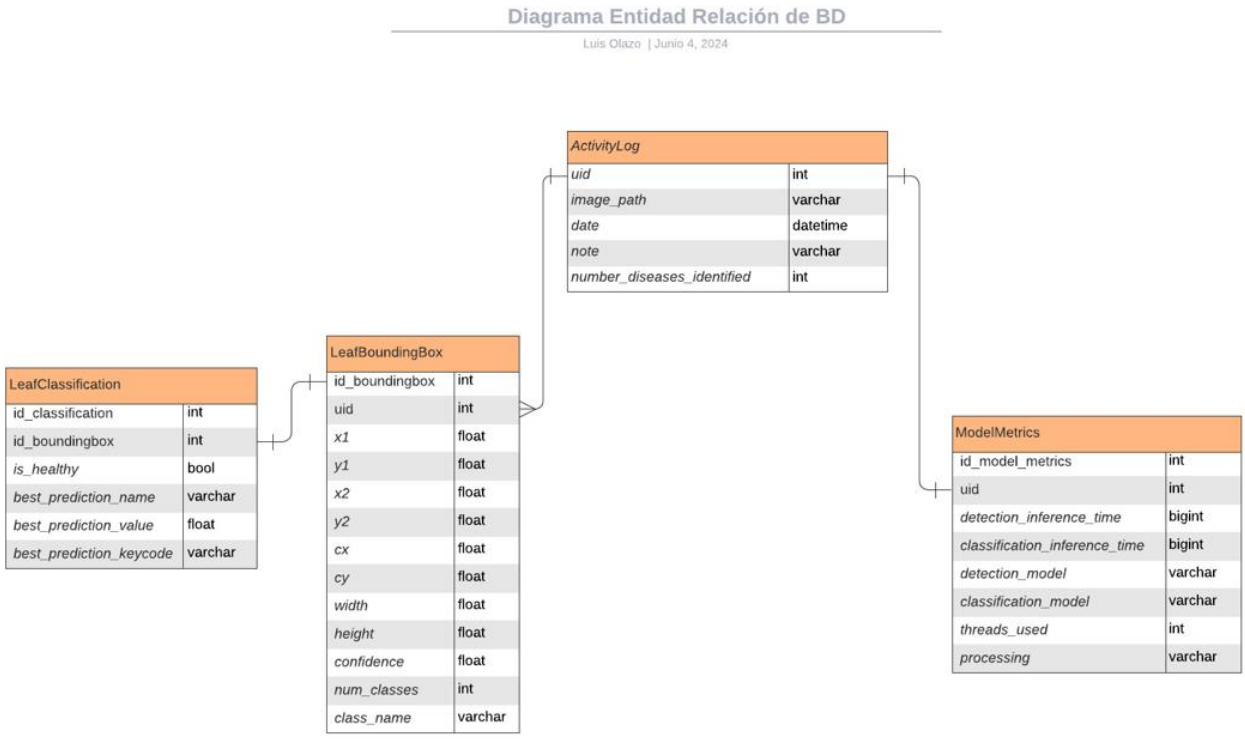
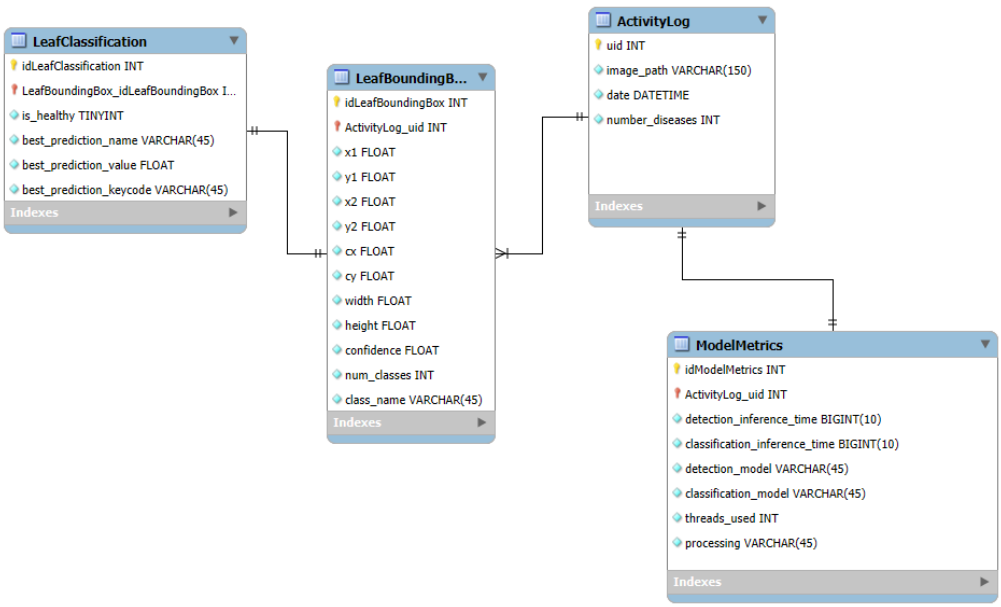


Figura 39
Modelo físico de base de datos



1.8. Dataset para el modelo de detección de hojas.

Las imágenes utilizadas para la detección de hojas provienen del dataset "Leaf Detection" disponible públicamente en la plataforma Kaggle. Este dataset se compone de 1132 imágenes para entrenamiento, acompañadas de un archivo .CSV con 5346 registros que indican la posición de las cajas delimitadoras de las hojas en las imágenes. Además, para pruebas, el dataset incluye 7 imágenes.

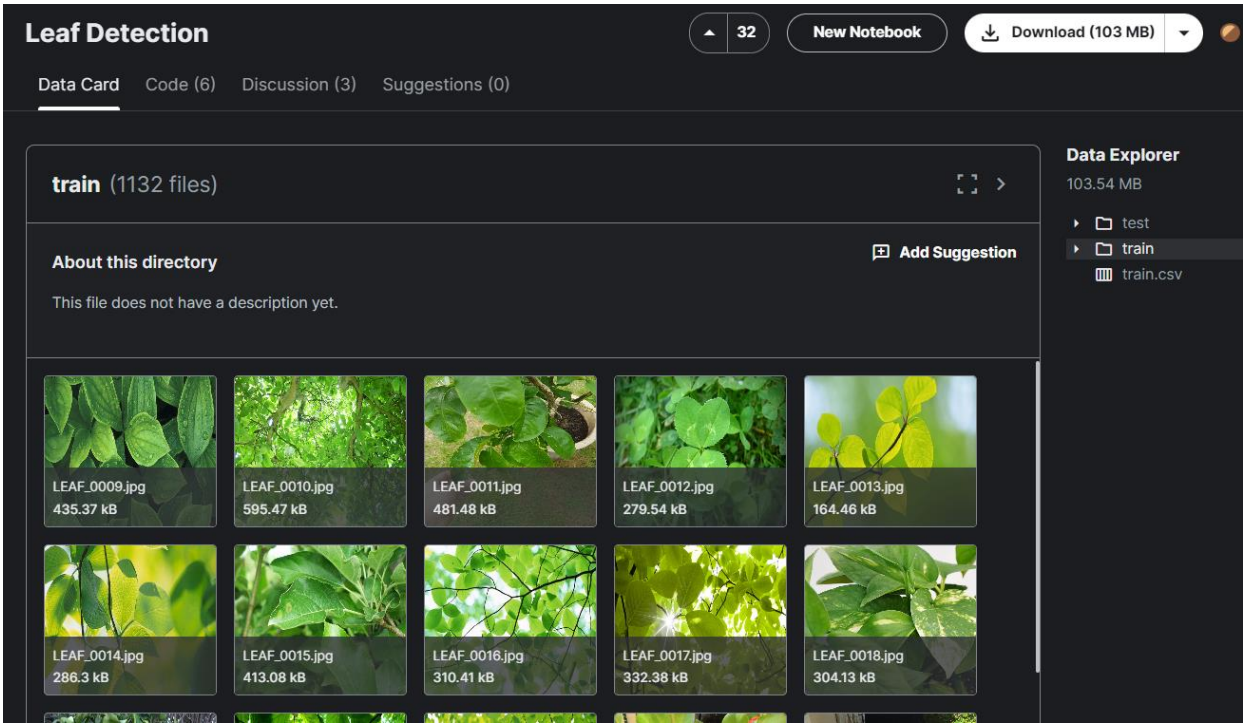
Tabla 21

Detalle del Dataset Original Kaggle “Leaf Detection”

	Referencia	Categoría	Cantidad
Dataset	Train	Hoja	1132
	Test	Hoja	7
Total, de imágenes de hojas			1139
Disponibilidad: https://www.kaggle.com/datasets/alexo98/leaf-detection			

Figura 40

Imágenes del dataset Kaggle: “Leaf Detection”



Este dataset ha sido reestructurado para adecuarse a los valores óptimos de entrenamiento, validación y prueba, y adaptarse al formato de datos requerido por YoloV8. La distribución de las imágenes es la siguiente: 904 imágenes (80%) para entrenamiento, 113 imágenes (10%) para validación y 113 imágenes (10%) para pruebas.

Tabla 22

Detalle del Dataset “Leaf Detection” reestructurado para YoloV8

	Referencia	Categoría	Cantidad
Dataset	Train	Hoja	904
	Val	Hoja	113
	Test	Hoja	113
Disponibilidad: https://www.kaggle.com/code/luisolazo/leaf-detection-w-ultralytics-yolov8-and-tflite/output			

1.9. Dataset para los modelos de clasificación

Las imágenes utilizadas para la clasificación de enfermedades en hojas de plantas de tomate provienen de dos investigaciones: "Identification of Plant Leaf Diseases Using a Nine-Layer Deep Convolutional Neural Network" (2019) y "Tomato-Village: A Dataset for End-to-End Tomato Disease Detection in a Real-World Environment" (2023). Estos estudios han puesto a disposición sus datos de forma pública, resumidos en la siguiente tabla:

Tabla 23

Recopilación de Imágenes para Clasificación de otras Investigaciones

Investigación	Cantidad de imágenes usadas	Disponibilidad
Identification of Plant Leaf Diseases Using a Nine-Layer Deep Convolutional Neural Network	18160	https://data.mendeley.com/datasets/tywbtsjrjv/1
Tomato-Village: A Dataset for End-to-End Tomato Disease Detection in a Real-World Environment	1616	https://github.com/mamta-joshi-gehlot/Tomato-Village

Para formar el dataset final de este estudio, se procesaron las imágenes inicialmente mediante una función de eliminación de imágenes borrosas. Posteriormente, se aplicó aumento de datos para diversificar los contextos representados en los datos.

Figura 41

Código para eliminar las imágenes borrosas del dataset de clasificación

```
def is_blurry(image_path, threshold=100):
    # Load image
    image = cv2.imread(image_path)

    # Convert image to grayscale
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Calculate the value of the image blur using the Laplacian variance
    variance = cv2.Laplacian(gray, cv2.CV_64F).var()

    # If the variance is less than the threshold, the image is considered blurred.
    return variance < threshold
```

Figura 42

Código para aumentos de datos del dataset de clasificación

```
datagen = image.ImageDataGenerator(
    width_shift_range=0.12,
    height_shift_range=0.12,
    shear_range=0.12,
    zoom_range=0.12,
    horizontal_flip=True,
    vertical_flip=True,
    rotation_range=60,
    fill_mode='constant',
    cval=0,
    brightness_range=[0.80, 1.42])
```

El dataset final está compuesto por 22,193 imágenes, distribuidas de la siguiente manera: 17,753 imágenes para entrenamiento y 4,440 imágenes para pruebas. Ambos conjuntos están clasificados en 10 categorías, detalladas en la siguiente tabla:

Tabla 24

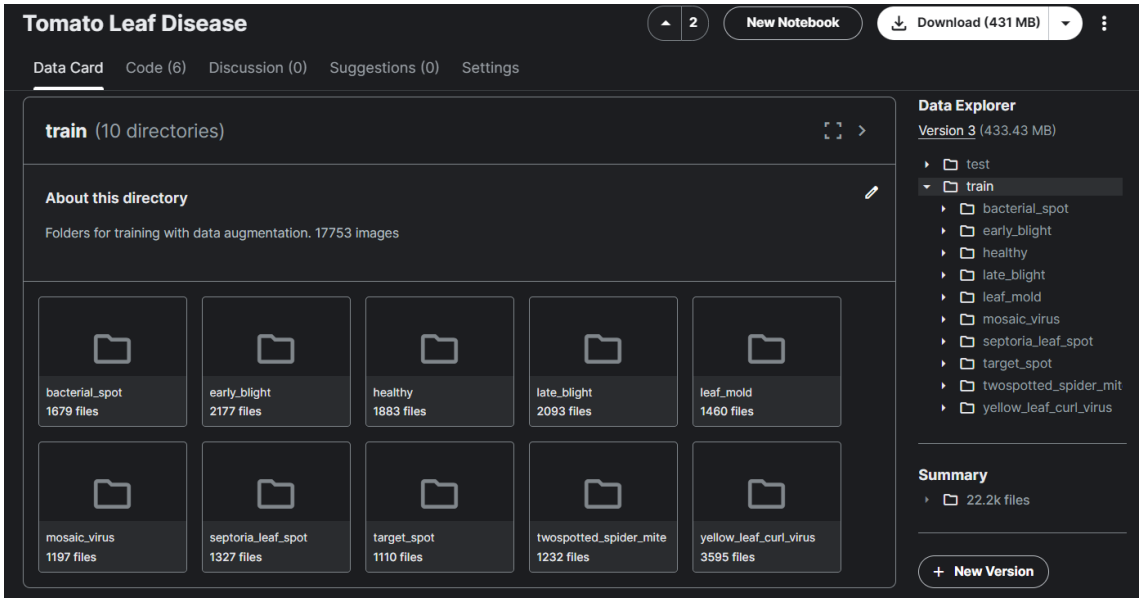
Detalle del Dataset para Clasificación de Enfermedades

	Categoría	**Cantidad para entrenamiento**	**Cantidad para pruebas**
Dataset	bacterial_spot	1679	444
	early_blight	2177	444

	healthy	1883	444
	late_blight	2093	444
	leaf_mold	1460	444
	mosaic_virus	1197	444
	septoria_leaf_spot	1327	444
	target_spot	1110	444
	twospotted_spider_mite	1232	444
	yellow_leaf_curl_virus	3595	444
Total		17753	4440
Disponibilidad: https://www.kaggle.com/datasets/luisolazo/tomato-diseases			

Figura 43

Dataset para los modelos de clasificación de enfermedades



3. Fase III: Producción

1.10. Codificación del Modelo de detección YoloV8

Como se especificó en la descripción del dataset, fue necesario convertir el dataset original al formato YOLOv8 para utilizarlo con la librería Ultralytics, la cual simplifica enormemente la implementación de este modelo de detección de objetos. Tras esta conversión, se utilizó la versión "YOLOv8n", diseñada para funcionar en dispositivos con recursos limitados, lo que la hace ideal para su uso en smartphones.

A continuación, la siguiente tabla resume el modelo utilizado, mostrando el mean Average Precision (mAP) y el número de épocas.

Tabla 25

Resumen del Modelo YoloV8n

Modelo	mAP50-95	mAP50	mAP75	Épocas
YoloV8n	**49.5%**	**70.5%**	**57.5%**	**60**
Disponibilidad: https://www.kaggle.com/code/luisolazo/leaf-detection-w-ultralytics-yolov8-and-tflite				

Para entrenar el modelo, solo es necesario especificar la ruta de los datos en formato YOLO y proporcionar al menos la cantidad de épocas, la dimensión de la imagen y el tamaño del lote. Los demás parámetros mencionados en el código pueden variar según sea necesario.

Figura 44

Código de entrenamiento del modelo YoloV8n para la detección de hojas

```
# Loading a pretrained model
model = YOLO('yolov8n.pt')

# Training the model
model.train(data = '/kaggle/working/yolo_dataset_leafdetection/data.yaml',
            epochs = 60,
            imgsz = height,
            seed = random_seed,
            batch = 8,
            workers = 4)
```

Para presentar los resultados del entrenamiento, se han diseñado funciones comunes que manipulan los datos obtenidos, simplificando la visualización y destacando el código más relevante. Las siguientes figuras muestran el código utilizado para mostrar los resultados y para convertir el modelo al formato TensorflowLite, para su uso en la aplicación móvil.

Figura 45

Código para mostrar los resultados de YoloV8n

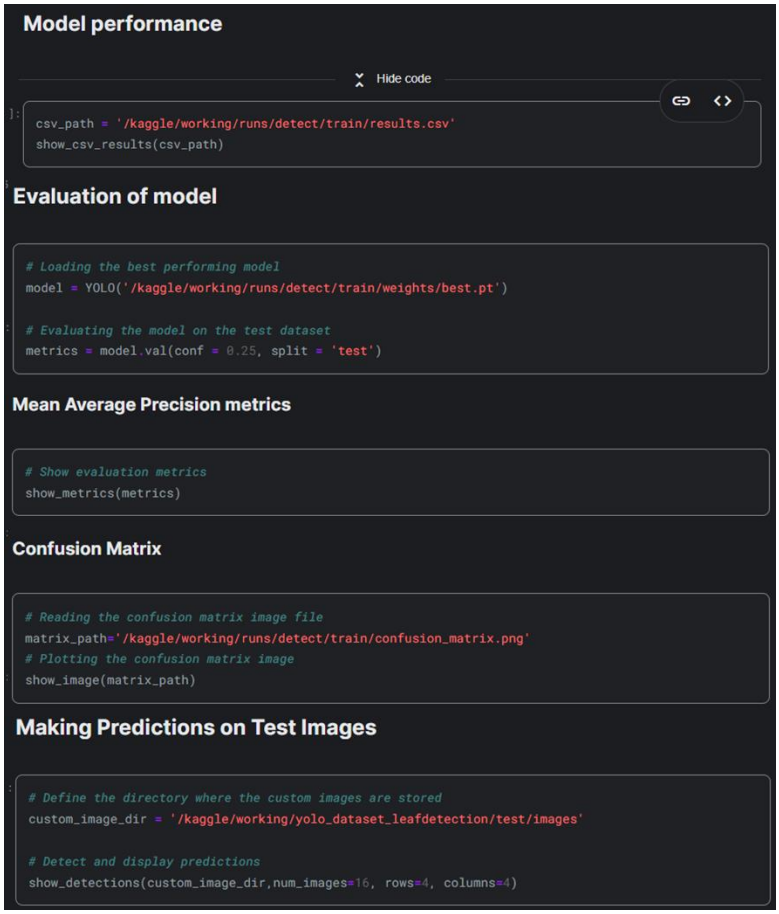
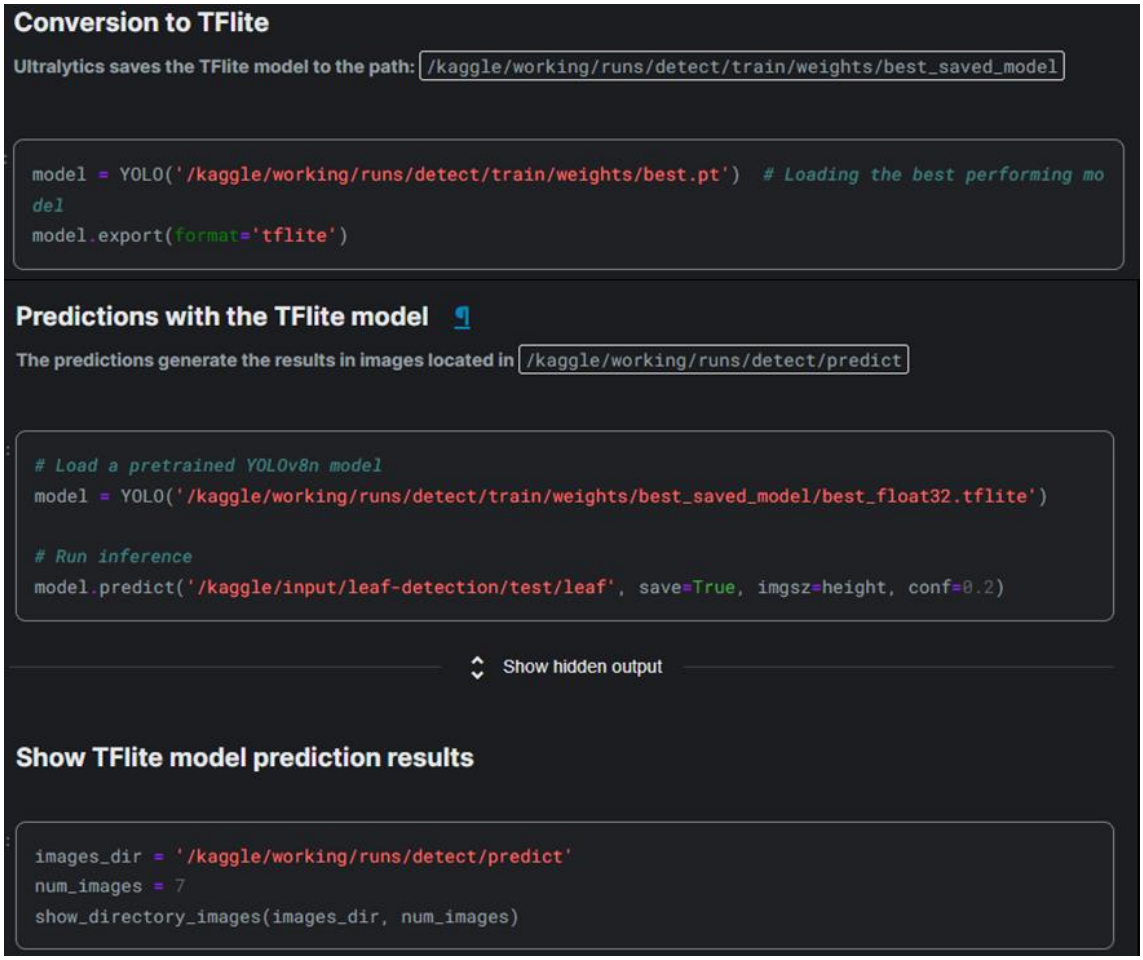


Figura 46

Código para convertir el modelo YoloV8n a TFlite y mostrar predicciones



1.11. Resultados del Modelo de Detección Yolov8

Aquí se presentan los resultados obtenidos producto de la codificación del modelo YoloV8n.

Figura 47

Métricas del entrenamiento y perdida del modelo YoloV8n

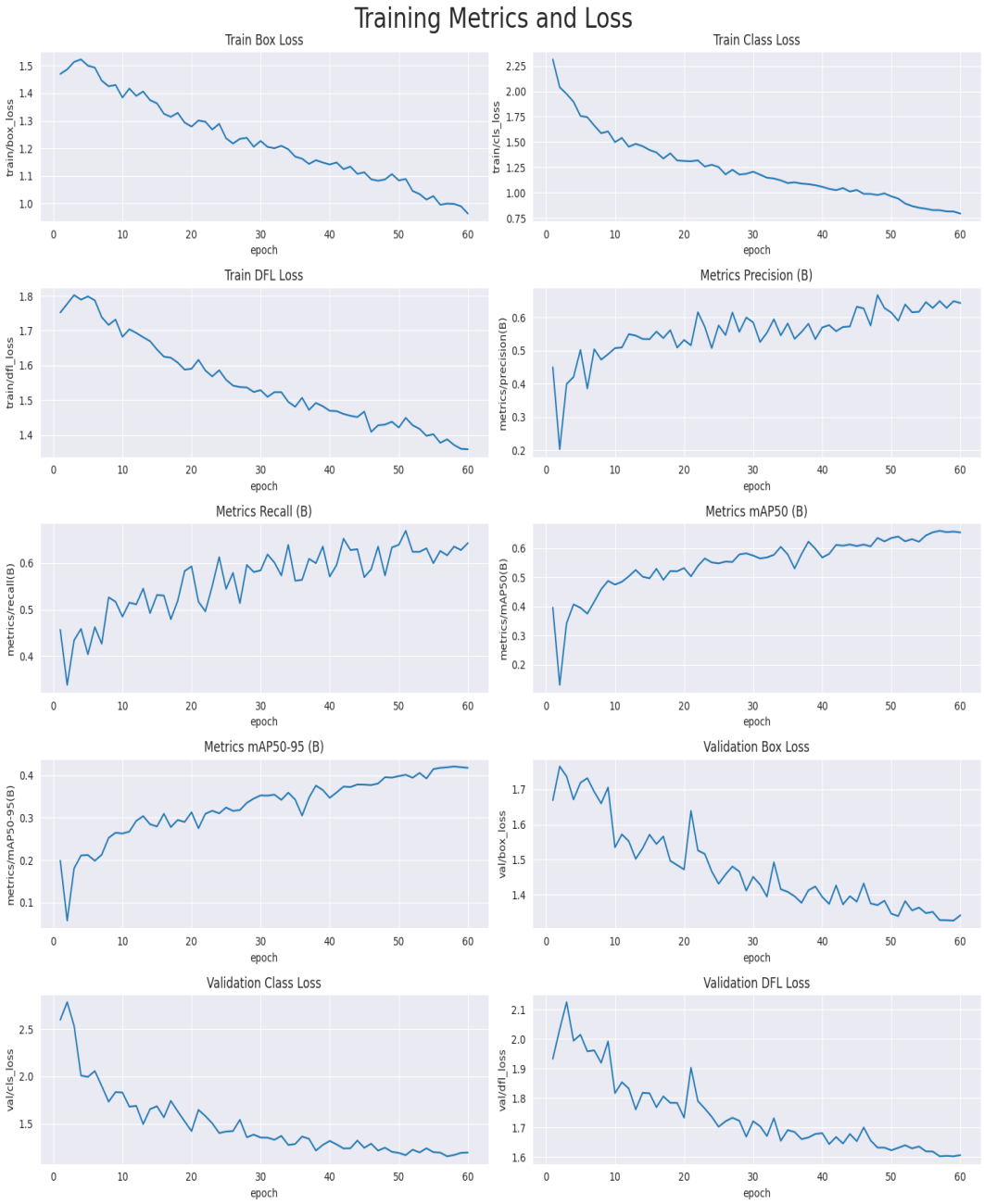


Figura 48

Evaluación del modelo YoloV8n

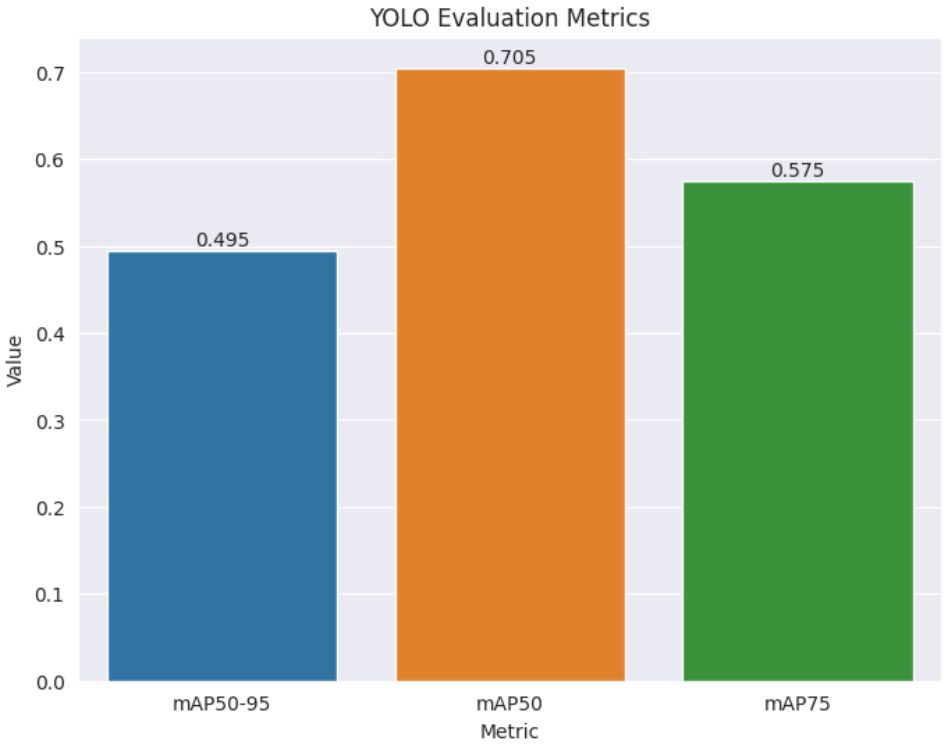


Figura 49

Matriz de confusión del modelo YoloV8n

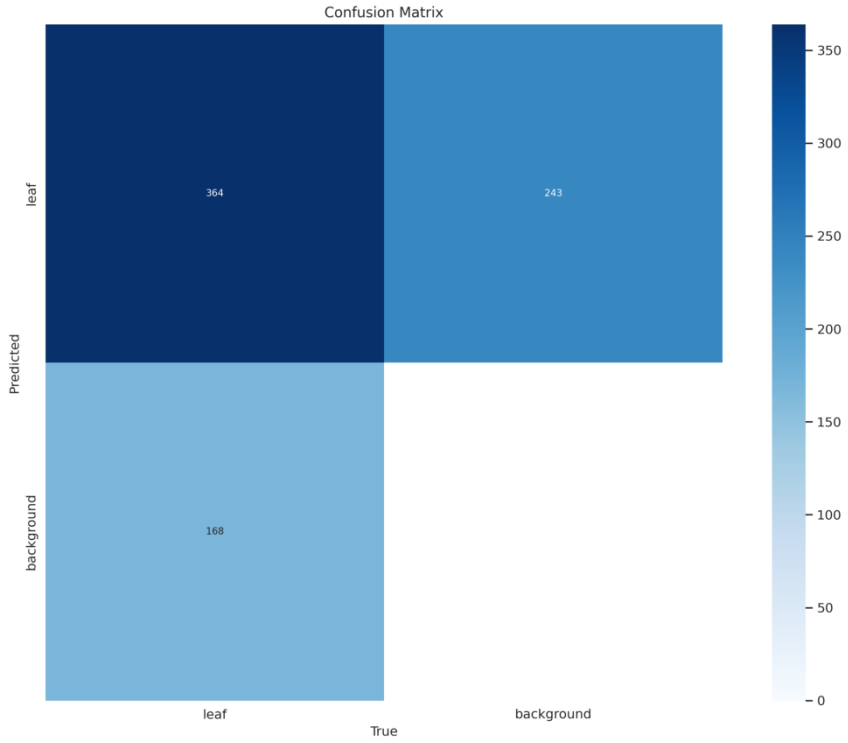


Figura 50

Predicciones con modelo YoloV8n

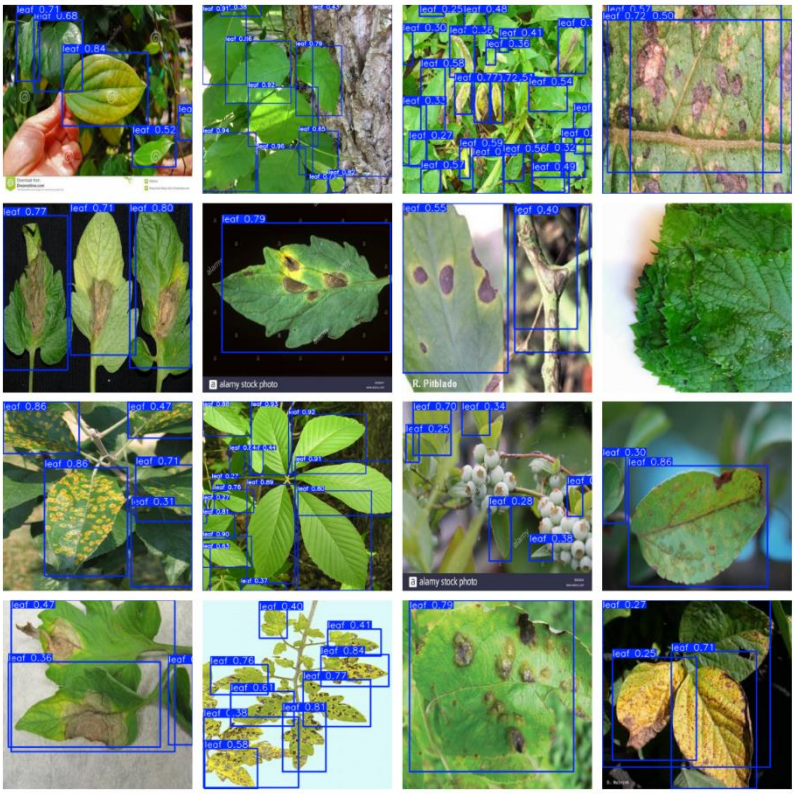
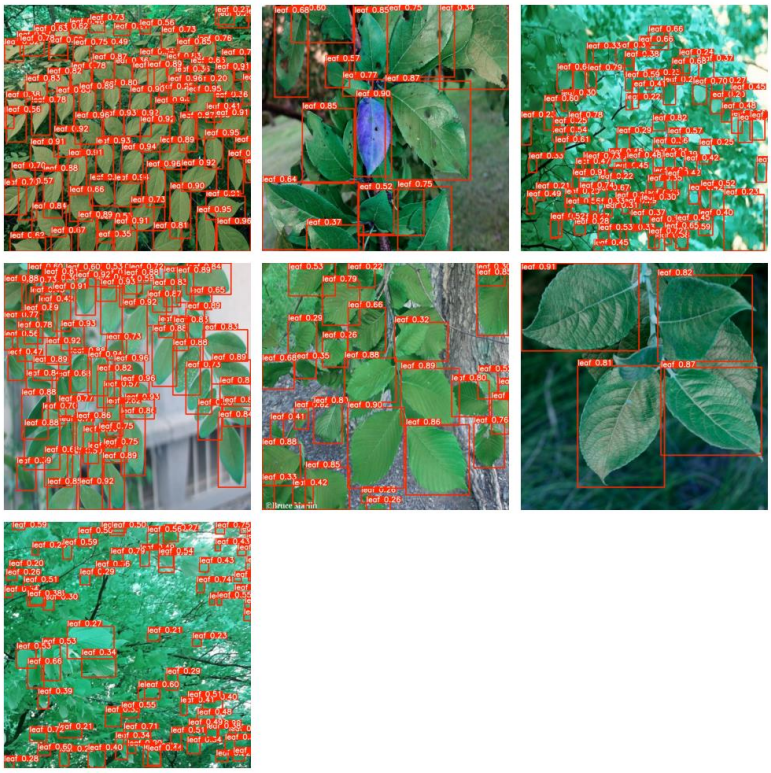


Figura 51

Predicciones con modelo YoloV8n convertido a TF Lite



1.12. Generalización en la Codificación de Modelos de Clasificación

Se ha utilizado tensorflow y keras como frameworks principales para el desarrollo de todos los modelos de clasificación sobre la plataforma Kaggle que permite llevar un registro público de las versiones y proceso de desarrollo de manera transparente.

La siguiente tabla resume todos los modelos disponibles que se han utilizado, así como la precisiones y épocas empleadas.

Tabla 26

Resumen de los Modelos de Clasificación

Modelo	Precisión	Precisión convertido a TFlite	Épocas
SqueezeNet-Mish	**97.3%**	**97.2%**	**57**
MobileNetV3Large	**97.1%**	**96.3%**	**29**
MobileNetV3Small	**96.6%**	**95.2%**	**39**
MobileNetV2	**93.8%**	**93.3%**	**31**
NASNetMobile	**90.2%**	**90.1%**	**35**
Disponibilidad: https://www.kaggle.com/luisolazo/code			

Todos los modelos de clasificación, excepto SqueezeNet, utilizan la técnica de aprendizaje por transferencia. Además, todos emplean el mismo dataset como se especifica en la siguiente figura.

Figura 52

Especificación de la ruta del dataset para los modelos de clasificación

```
# Directories
DIR_ROOT = '/kaggle/input/tomato-diseases'
DIR_TRAIN = f'{DIR_ROOT}/train'
DIR_TEST = f'{DIR_ROOT}/test'
```

Asimismo, los modelos de clasificación comparten parámetros de entrenamiento similares y utilizan funciones comunes para la manipulación de datos, como se muestra en la figura de a continuación.

Figura 53

Parámetros comunes para los modelos de clasificación

```
# Parameters
IMAGE_SIZE = (224,224)
INPUT_SHAPE = (224, 224, 3)
IMG_COLOR = 'rgb'
BATCH_SIZE = 32
BUFFER_SIZE = 1000
SHUFFLE = True
INTERPOLATION = 'bilinear'
FIT_EPOCHS = 100
FIT_CALLBACKS = [keras.callbacks.EarlyStopping(monitor='val_loss',
                                               patience=15,
                                               min_delta=0.001,
                                               restore_best_weights=True,
                                               verbose=1)]
COMPILE_OPTIMIZER = keras.optimizers.Adam()
COMPILE_LOSS = keras.losses.SparseCategoricalCrossentropy()
COMPILE_METRICS = ['accuracy']
```

El preprocesamiento de datos se realiza de manera similar para todos los modelos utilizando el mismo código. Además, no es necesario normalizar las imágenes, ya que esta lógica se incluye en una capa en todos los modelos que aplica el reescalado de píxeles al ingresar la imagen.

Figura 54

Preprocesamiento de datos para los modelos de clasificación

```
# Data preprocessing
train_ds = keras.utils.image_dataset_from_directory(
    DIR_TRAIN,
    interpolation=INTERPOLATION,
    image_size=IMAGE_SIZE,
    color_mode=IMG_COLOR,
    batch_size=BATCH_SIZE,
    shuffle=SHUFFLE
)
validation_ds = keras.utils.image_dataset_from_directory(
    DIR_TEST,
    interpolation=INTERPOLATION,
    image_size=IMAGE_SIZE,
    color_mode=IMG_COLOR,
    batch_size=BATCH_SIZE,
    shuffle=SHUFFLE
)

# Get category names from the dataset
categories = validation_ds.class_names
num_classes = len(categories)

# Caching, blending and preloading of training and validation datasets
train_ds = train_ds.cache().shuffle(BUFFER_SIZE).prefetch(buffer_size=tf.data.AUTOTUNE)
validation_ds = validation_ds.cache().prefetch(buffer_size=tf.data.AUTOTUNE)
```

1.13. Codificación del Modelo de Clasificación SqueezeNet-Mish

El caso del modelo SqueezeNet es particular debido a la falta de una implementación predefinida en el framework Tensorflow hasta la fecha. Por lo tanto, fue necesario implementarlo a partir del artículo original titulado "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size". Además, se realizó una modificación para reemplazar la función de activación relu, que se utiliza de manera predeterminada, por la función de activación mish, la cual ha demostrado ofrecer mejores resultados con un incremento mínimo en la complejidad computacional.

Figura 55

Código del modelo de clasificación SqueezeNet-Mish

```
def mish(x):
    return x * tf.math.tanh(tf.math.softplus(x))

def fire_module(x, s1, e1, e3, name):
    s1x1 = keras.layers.Conv2D(filters=s1, kernel_size=(1, 1), padding='same', name=f'{name}_squeeze')(x)
    s1x1 = mish(s1x1)
    e1x1 = keras.layers.Conv2D(filters=e1, kernel_size=(1, 1), padding='same', name=f'{name}_expand_1x1')(s1x1)
    e3x3 = keras.layers.Conv2D(filters=e3, kernel_size=(3, 3), padding='same', name=f'{name}_expand_3x3')(s1x1)
    x = keras.layers.concatenate([e1x1, e3x3])
    x = mish(x)
    return x

def SqueezeNet(input_shape, nclasses):
    input = keras.layers.Input(input_shape,name='input_image')
    # This layer(Rescaling) is not part of the original architecture, it was added so that the default model already applies pix
el rescaling.
    x = keras.layers.Rescaling(scale=1.0 / 127.5, offset=-1.0, name='pixel_rescaling')(input)
    x = keras.layers.Conv2D(filters=96, kernel_size=(7,7), strides=(2,2), padding='same', input_shape=input_shape, name='conv
1')(x)
    x = keras.layers.MaxPool2D(pool_size=(3,3), strides = (2,2), name='maxpool1')(x)
    x = fire_module(x, s1 = 16, e1 = 64, e3 = 64, name='fire2')
    x = fire_module(x, s1 = 16, e1 = 64, e3 = 64, name='fire3')
    x = fire_module(x, s1 = 32, e1 = 128, e3 = 128, name='fire4')
    x = keras.layers.MaxPool2D(pool_size=(3,3), strides = (2,2), name='maxpool4')(x)
    x = fire_module(x, s1 = 32, e1 = 128, e3 = 128, name='fire5')
    x = fire_module(x, s1 = 48, e1 = 192, e3 = 192, name='fire6')
    x = fire_module(x, s1 = 48, e1 = 192, e3 = 192, name='fire7')
    x = fire_module(x, s1 = 64, e1 = 256, e3 = 256, name='fire8')
    x = keras.layers.MaxPool2D(pool_size=(3,3), strides = (2,2), name='maxpool8')(x)
    x = fire_module(x, s1 = 64, e1 = 256, e3 = 256, name='fire9')
    x = keras.layers.Dropout(0.5)(x)
    x = keras.layers.Conv2D(filters=nclasses, kernel_size=(1,1), strides=(1,1), padding='valid', name='conv10')(x)
    x = keras.layers.GlobalAveragePooling2D(name='avgpool10')(x)
    output = keras.layers.Activation(activation=tf.nn.softmax, name='activation_softmax')(x)
    model = keras.Model(input, output,name='Squeezenet')
    return model
```

Figura 56

Código del entrenamiento del modelo Squeezenet-Mish

```
# SqueezeNet instance
final_model = SqueezeNet(input_shape=INPUT_SHAPE, nclasses=num_classes)

# Prints a summary of the SqueezeNet architecture.
final_model.summary()

# Compilation of the final model
final_model.compile(
    optimizer=COMPILE_OPTIMIZER,
    loss=COMPILE_LOSS,
    metrics=COMPILE_METRICS
)

# Model training
HISTORY = final_model.fit(
    train_ds,
    steps_per_epoch=len(train_ds),
    validation_data=validation_ds,
    validation_steps=len(validation_ds),
    epochs=FIT_EPOCHS,
    callbacks=FIT_CALLBACKS)
```

1.14. Resultados del Modelo de Clasificación SqueezeNet-Mish

Figura 57

Evaluación del modelo Squeezenet-Mish

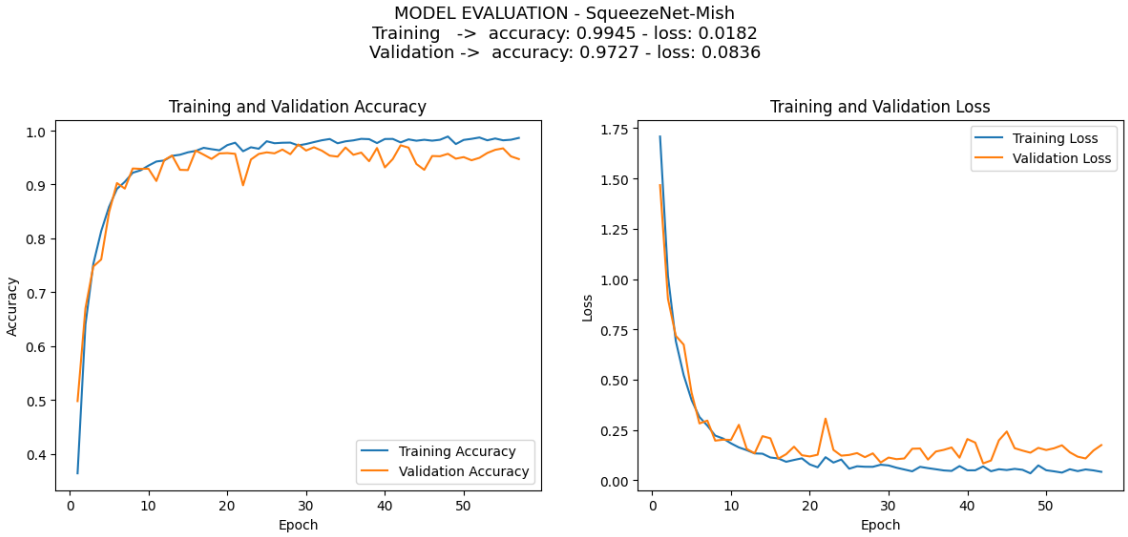


Figura 58

Reporte de clasificación del modelo Squeezenet-Mish

	precision	recall	f1-score	support
bacterial_spot	0.9931	0.9662	0.9795	444.0000
early_blight	0.9393	0.9414	0.9404	444.0000
healthy	0.9818	0.9707	0.9762	444.0000
late_blight	0.9531	0.9617	0.9574	444.0000
leaf_mold	0.9886	0.9775	0.9830	444.0000
mosaic_virus	1.0000	0.9887	0.9943	444.0000
septoria_leaf_spot	0.9775	0.9797	0.9786	444.0000
target_spot	0.9351	0.9730	0.9536	444.0000
twospotted_spider_mite	0.9732	0.9797	0.9764	444.0000
yellow_leaf_curl_virus	0.9887	0.9887	0.9887	444.0000
accuracy	0.9727	0.9727	0.9727	0.9727
macro avg	0.9730	0.9727	0.9728	4440.0000
weighted avg	0.9730	0.9727	0.9728	4440.0000

Figura 59

Matriz de confusión del modelo Squeezenet-Mish

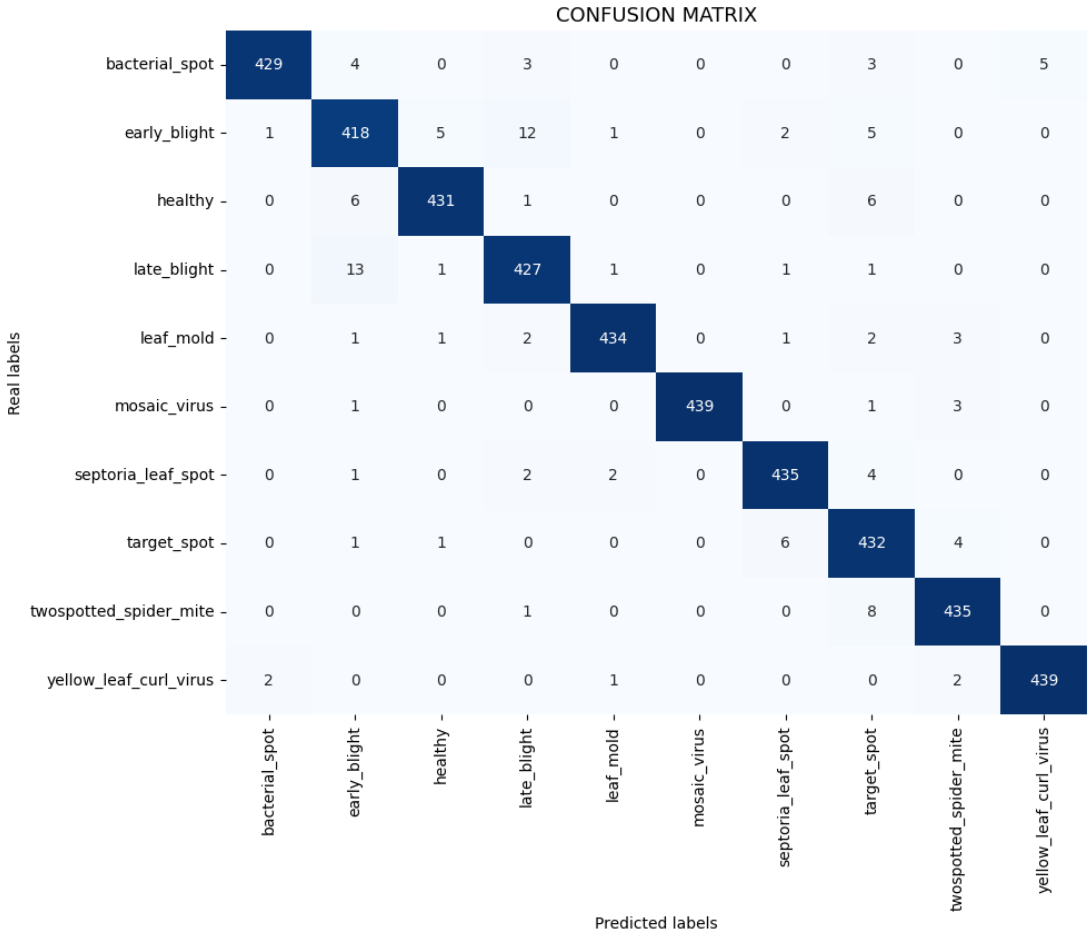
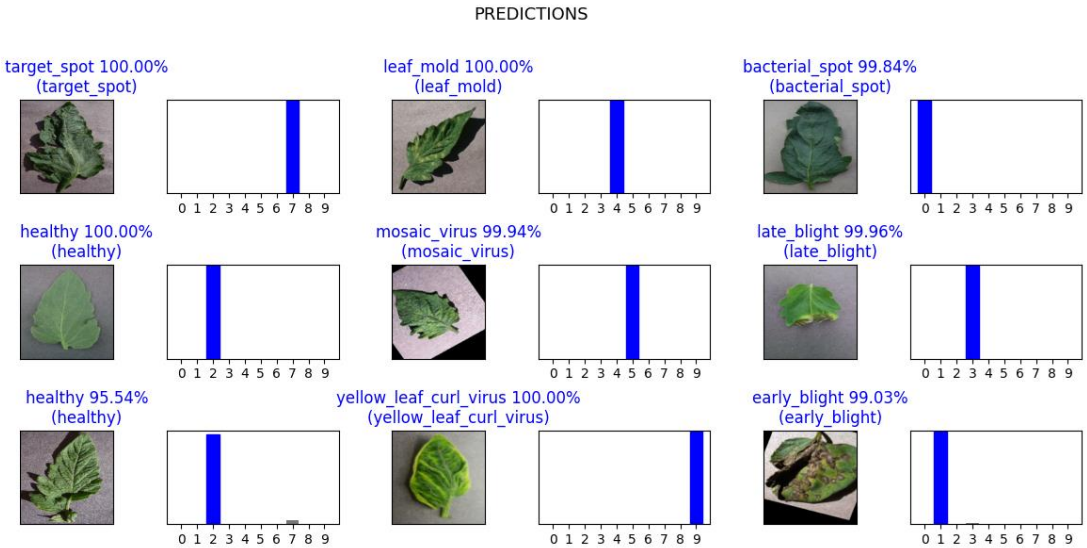


Figura 60

Ejemplo de predicciones del modelo Squeezenet-Mish



1.15. Codificación del Modelo de Clasificación MobileNetV3Large

Figura 61

Código del modelo MobileNetV3Large

```
# MobileNetV3Large instance as pretrained_model
pretrained_model = MobileNetV3Large(
    input_shape=INPUT_SHAPE,
    include_top=False,
    weights='imagenet',
    pooling='avg'
)

# The training of all MobileNetV3Large layers is frozen.
pretrained_model.trainable = False

# Prints a summary of the MobileNetV3Large architecture.
pretrained_model.summary()
# Building the final model using MobileNetV3Large as base model
# Mobilenetv3 already includes pixel value rescaling.
inputs = pretrained_model.input
x = pretrained_model(inputs)
x = layers.Dense(128, activation=activations.relu)(x)
x = layers.Dropout(0.2)(x)
outputs =  layers.Dense(num_classes)(x)
final_model = Model(inputs=inputs, outputs=outputs,name=MODEL_NAME)

# Compilation of the final model
final_model.compile(
    optimizer=COMPILE_OPTIMIZER,
    loss=COMPILE_LOSS,
    metrics=COMPILE_METRICS
)

# Prints a summary of the final model
final_model.summary()

# Model training
HISTORY = final_model.fit(
    train_ds,
    steps_per_epoch=len(train_ds),
    validation_data=validation_ds,
    validation_steps=len(validation_ds),
    epochs=FIT_EPOCHS,
    callbacks=FIT_CALLBACKS
)
```

1.16. Resultado del Modelo de Clasificación MobileNetV3Large

Figura 62

Evaluación del modelo MobileNetV3Large

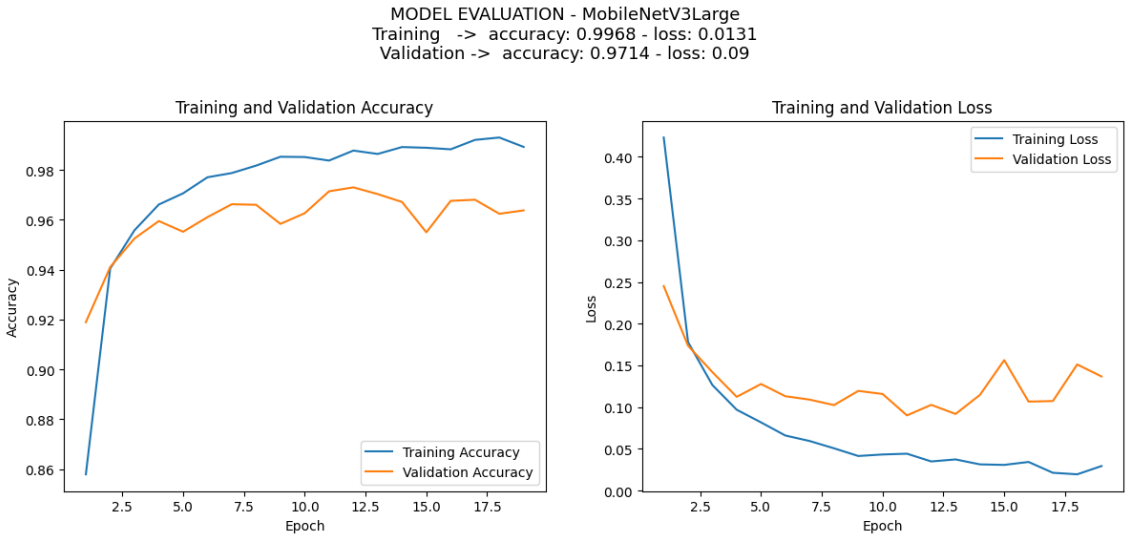


Figura 63

Reporte de clasificación del modelo MobileNetV3Large

	precision	recall	f1-score	support
bacterial_spot	0.9820	0.9820	0.9820	444.0000
early_blight	0.9524	0.9459	0.9492	444.0000
healthy	0.9820	0.9842	0.9831	444.0000
late_blight	0.9795	0.9685	0.9740	444.0000
leaf_mold	0.9797	0.9797	0.9797	444.0000
mosaic_virus	0.9955	0.9955	0.9955	444.0000
septoria_leaf_spot	0.9378	0.9842	0.9604	444.0000
target_spot	0.9544	0.9437	0.9490	444.0000
twospotted_spider_mite	0.9607	0.9369	0.9487	444.0000
yellow_leaf_curl_virus	0.9888	0.9910	0.9899	444.0000
accuracy	0.9712	0.9712	0.9712	0.9712
macro avg	0.9713	0.9712	0.9711	4440.0000
weighted avg	0.9713	0.9712	0.9711	4440.0000

Figura 64

Matriz de confusión del modelo MobileNetV3Large

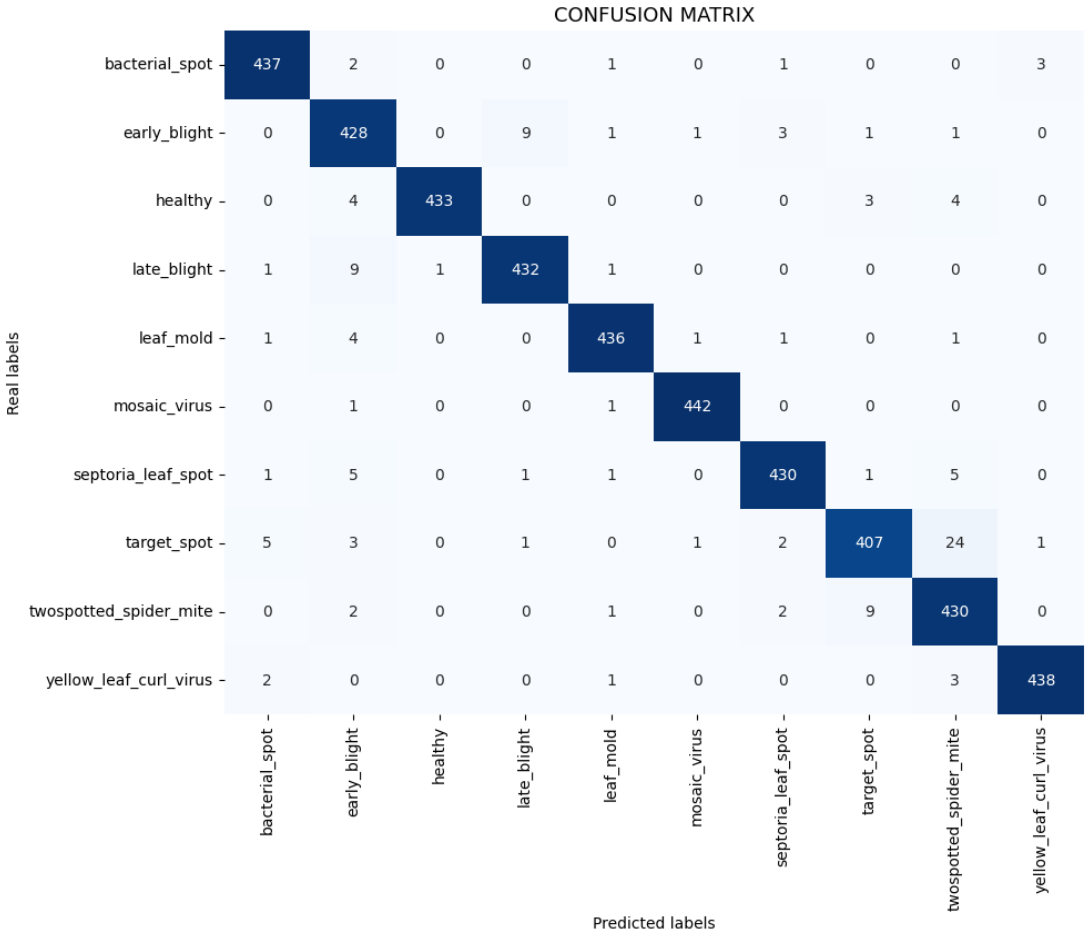
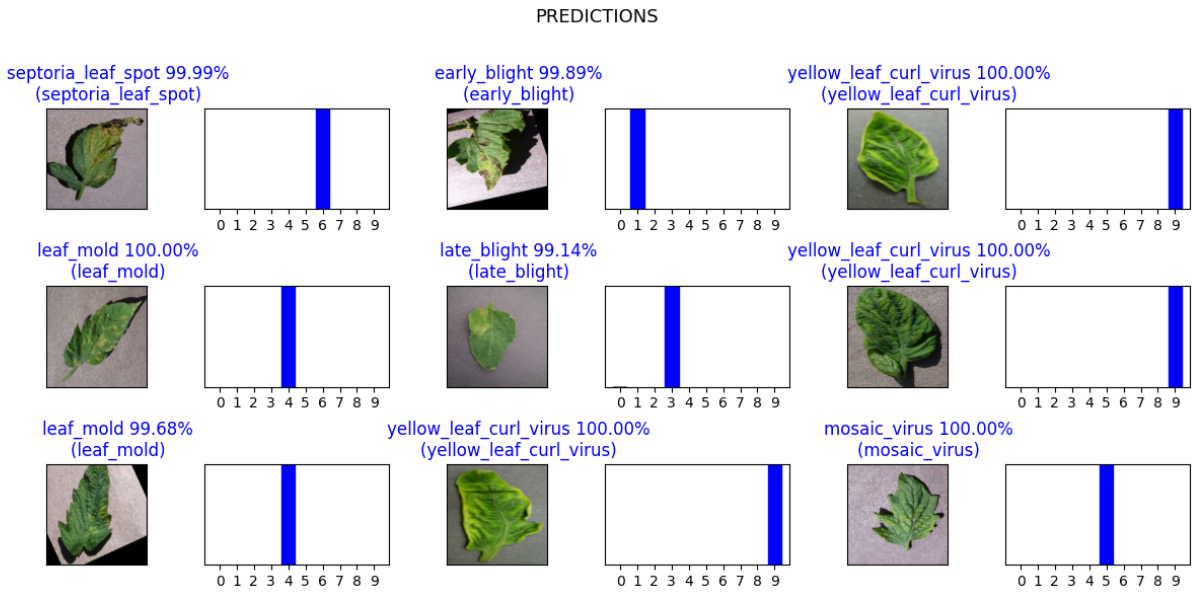


Figura 65

Ejemplo de predicciones del modelo MobileNetV3Large



1.17. Codificación del Modelo de Clasificación MobileNetV3Small

Figura 66

Código del Modelo MobileNetV3Small

```
# MobileNetV3Small instance as pretrained_model
pretrained_model = MobileNetV3Small(
    input_shape=INPUT_SHAPE,
    include_top=False,
    weights='imagenet',
    pooling='avg'
)

# The training of all MobileNetV3Small layers is frozen.
pretrained_model.trainable = False

# Prints a summary of the MobileNetV3Small architecture.
pretrained_model.summary()
# Building the final model using MobileNetV3Small as base model.
# Mobilenetv3 already includes pixel value rescaling.
inputs = pretrained_model.input
x = pretrained_model(inputs)
x = layers.Dense(128, activation=activations.relu)(x)
x = layers.Dropout(0.2)(x)
outputs =  layers.Dense(num_classes)(x)
final_model = Model(inputs=inputs, outputs=outputs,name=MODEL_NAME)

# Compilation of the final model
final_model.compile(
    optimizer=COMPILE_OPTIMIZER,
    loss=COMPILE_LOSS,
    metrics=COMPILE_METRICS
)

# Prints a summary of the final model
final_model.summary()
# Model training
HISTORY = final_model.fit(
    train_ds,
    steps_per_epoch=len(train_ds),
    validation_data=validation_ds,
    validation_steps=len(validation_ds),
    epochs=FIT_EPOCHS,
    callbacks=FIT_CALLBACKS
)
```

1.18. Resultados del Modelo de Clasificación MobileNetV3Small

Figura 67

Evaluación del modelo MobileNetV3Small

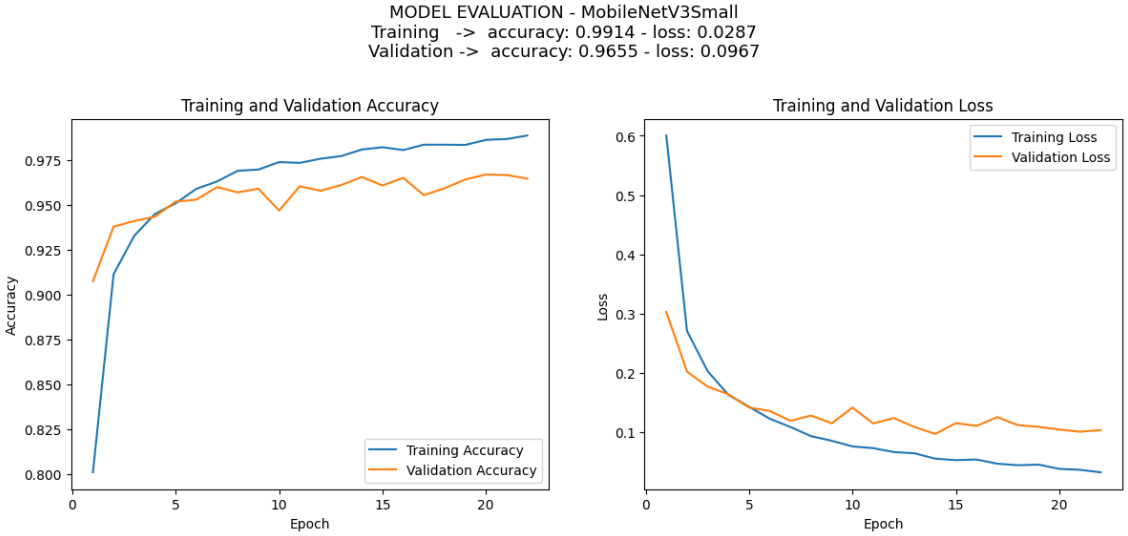


Figura 68

Reporte de clasificación del modelo MobileNetV3Small

	precision	recall	f1-score	support
bacterial_spot	0.9756	0.9910	0.9832	444.000
early_blight	0.9318	0.9234	0.9276	444.000
healthy	0.9841	0.9752	0.9796	444.000
late_blight	0.9515	0.9730	0.9621	444.000
leaf_mold	0.9931	0.9662	0.9795	444.000
mosaic_virus	0.9756	0.9910	0.9832	444.000
septoria_leaf_spot	0.9577	0.9685	0.9630	444.000
target_spot	0.9486	0.9144	0.9312	444.000
twospotted_spider_mite	0.9471	0.9685	0.9577	444.000
yellow_leaf_curl_virus	0.9955	0.9887	0.9921	444.000
accuracy	0.9660	0.9660	0.9660	0.966
macro avg	0.9661	0.9660	0.9659	4440.000
weighted avg	0.9661	0.9660	0.9659	4440.000

Figura 69

Matriz de confusión del modelo MobileNetV3Small

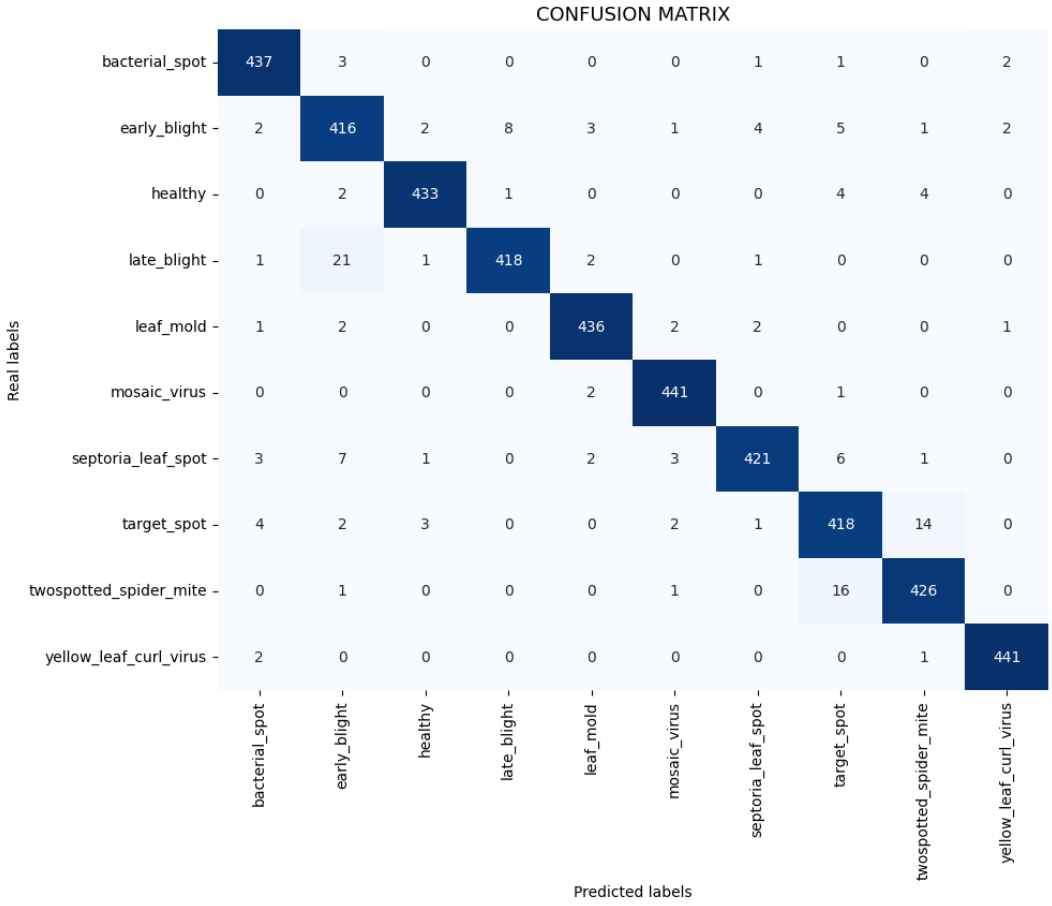
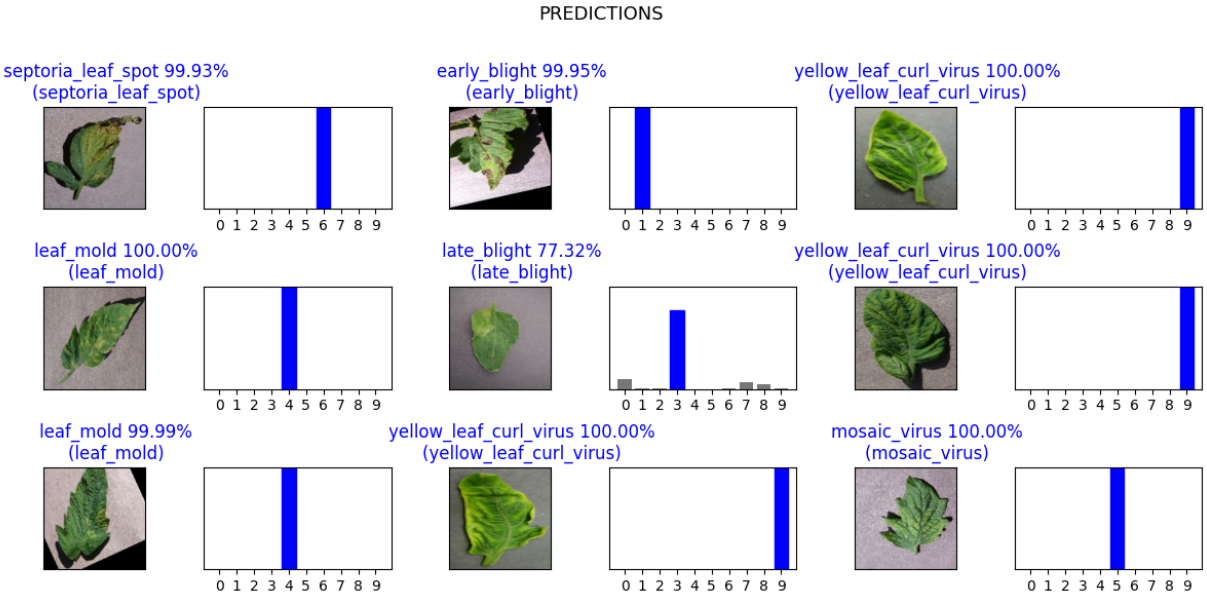


Figura 70

Ejemplo de predicciones del modelo MobileNetV3Small



1.19. Codificación del Modelo de Clasificación MobileNetV2

Figura 71

Código del modelo MobileNetV2

```
# MobileNetV2 instance as pretrained_model
pretrained_model = MobileNetV2(
    input_shape=INPUT_SHAPE,
    include_top=False,
    weights='imagenet',
    pooling='avg')

# The training of all MobileNetV2 layers is frozen.
pretrained_model.trainable = False

# Prints a summary of the MobileNetV2 architecture.
pretrained_model.summary()
# Building the final model using mobilenetv2 as pretrained_model
inputs = pretrained_model.input
x = mobilenet_v2.preprocess_input(inputs) # Rescale pixel values of [0,255] to [-1, 1]
x = pretrained_model(x)
x = layers.Dense(128, activation=activations.relu)(x)
x = layers.Dropout(0.2)(x)
outputs =  layers.Dense(num_classes)(x)
final_model = Model(inputs=inputs, outputs=outputs,name=MODEL_NAME)

# Compilation of the final model
final_model.compile(
    optimizer=COMPILE_OPTIMIZER,
    loss=COMPILE_LOSS,
    metrics=COMPILE_METRICS
)

# Prints a summary of the final model
final_model.summary()
# Model training
HISTORY = final_model.fit(
    train_ds,
    steps_per_epoch=len(train_ds),
    validation_data=validation_ds,
    validation_steps=len(validation_ds),
    epochs=FIT_EPOCHS,
    callbacks=FIT_CALLBACKS
)
```

3.11. Resultados del Modelo de Clasificación MobileNetV2

Figura 72

Evaluación del modelo MobileNetV2

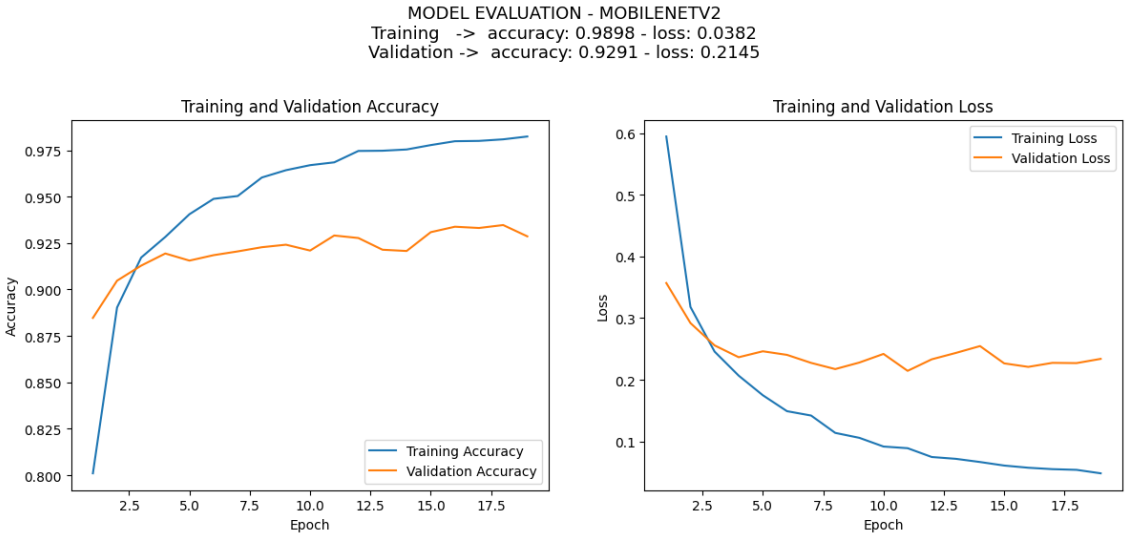


Figura 73

Reporte de clasificación del modelo MobileNetV2

	precision	recall	f1-score	support
bacterial_spot	0.9231	0.9730	0.9474	444.0000
early_blight	0.8696	0.9009	0.8850	444.0000
healthy	0.9383	0.9595	0.9488	444.0000
late_blight	0.9607	0.9369	0.9487	444.0000
leaf_mold	0.9278	0.9550	0.9412	444.0000
mosaic_virus	0.9818	0.9707	0.9762	444.0000
septoria_leaf_spot	0.9511	0.8761	0.9121	444.0000
target_spot	0.9243	0.8806	0.9020	444.0000
twospotted_spider_mite	0.9289	0.9414	0.9351	444.0000
yellow_leaf_curl_virus	0.9799	0.9865	0.9832	444.0000
accuracy	0.9381	0.9381	0.9381	0.9381
macro avg	0.9385	0.9381	0.9380	4440.0000
weighted avg	0.9385	0.9381	0.9380	4440.0000

Figura 74

Matriz de confusión del modelo MobileNetV2

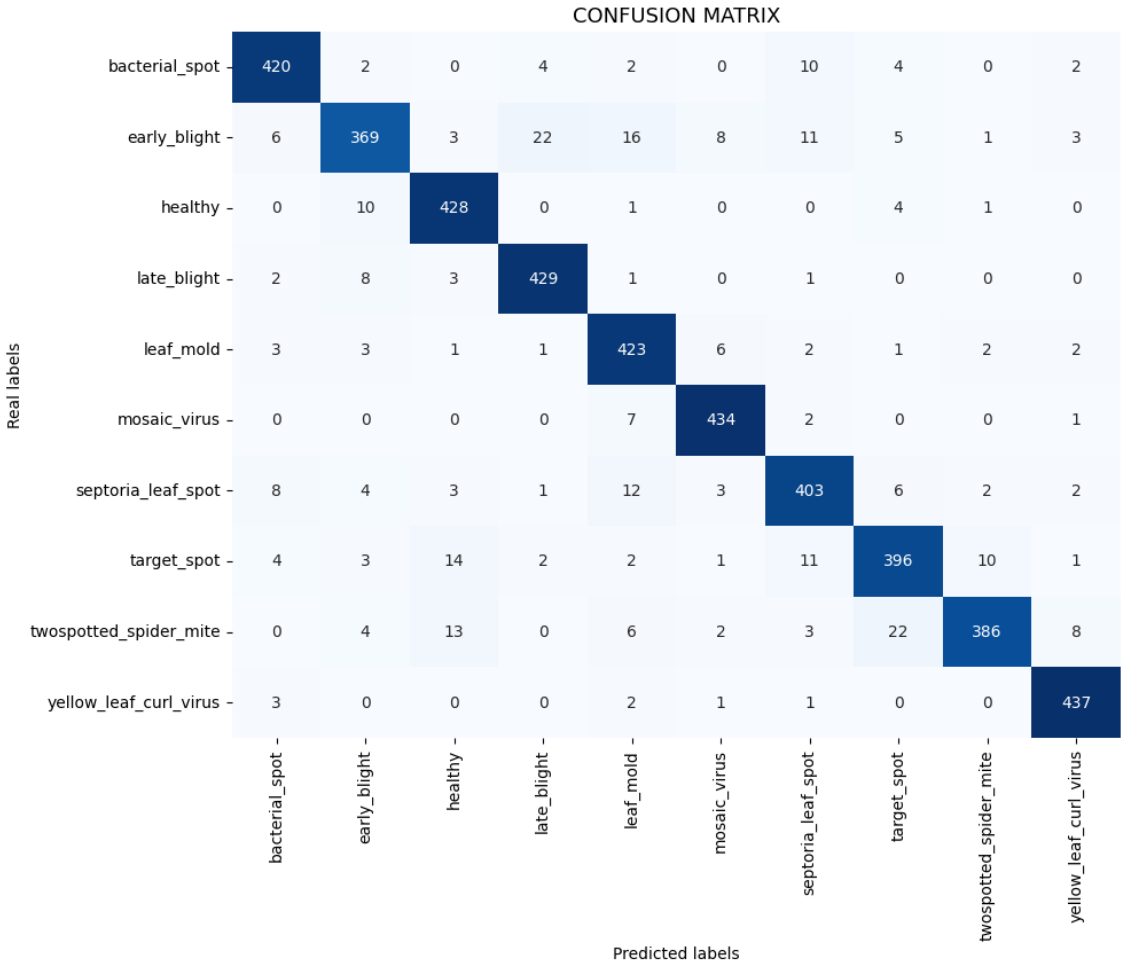
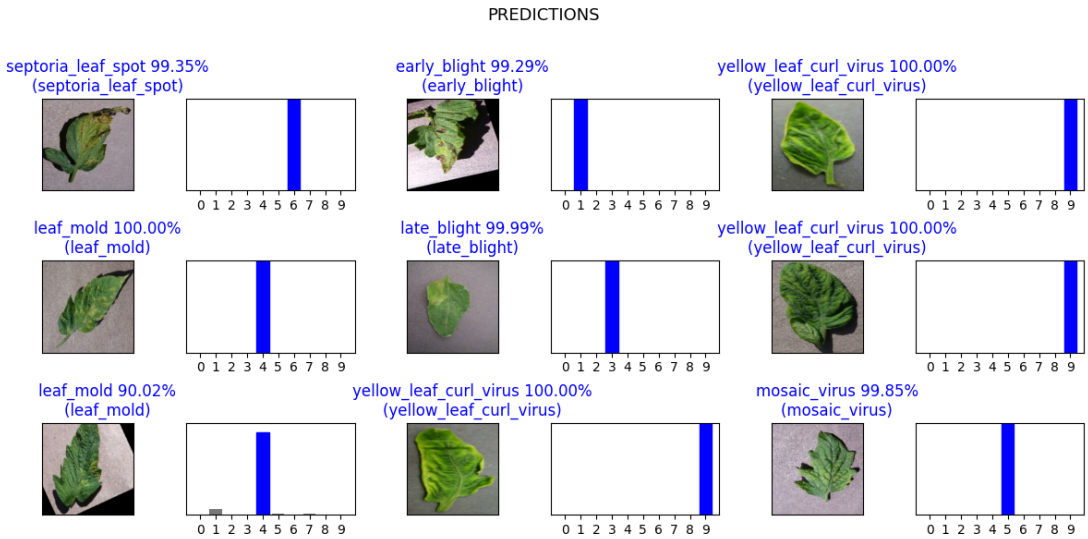


Figura 75

Ejemplo de predicciones del modelo MobileNetV2



3.12. Codificación del Modelo de Clasificación NASNetMobile

Figura 76

Código del modelo NASNetMobile

```
# NASNetMobile instance as pretrained_model
pretrained_model = NASNetMobile(
    input_shape=INPUT_SHAPE,
    include_top=False,
    weights='imagenet',
    pooling='avg'
)

# The training of all NASNetMobile layers is frozen.
pretrained_model.trainable = False

# Prints a summary of the NASNetMobile architecture.
pretrained_model.summary()

# Building the final model using NASNetMobile as base model
inputs = pretrained_model.input
x = nasnet.preprocess_input(inputs) # Rescale pixel values
x = pretrained_model(x)
x = layers.Dense(128, activation=activations.relu)(x)
x = layers.Dropout(0.2)(x)
outputs =  layers.Dense(num_classes)(x)
final_model = Model(inputs=inputs, outputs=outputs,name=MODEL_NAME)

# Compilation of the final model
final_model.compile(
    optimizer=COMPILE_OPTIMIZER,
    loss=COMPILE_LOSS,
    metrics=COMPILE_METRICS
)

# Prints a summary of the final model
final_model.summary()

# Model training
HISTORY = final_model.fit(
    train_ds,
    steps_per_epoch=len(train_ds),
    validation_data=validation_ds,
    validation_steps=len(validation_ds),
    epochs=FIT_EPOCHS,
    callbacks=FIT_CALLBACKS)
```

3.13. Resultados del Modelo de Clasificación NASNetMobile

Figura 77

Evaluación del modelo NASNetMobile

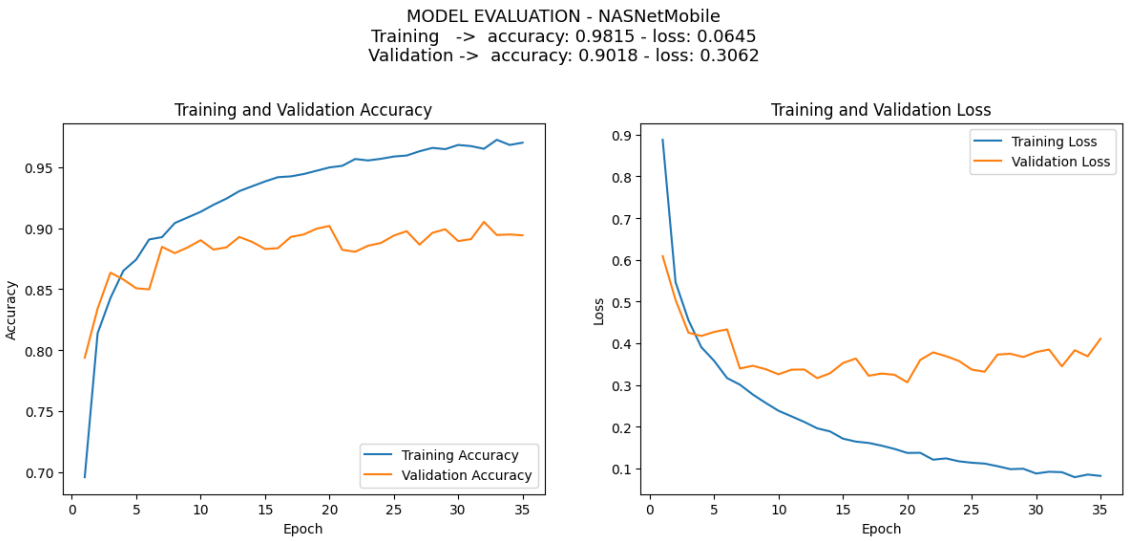


Figura 78

Reporte de clasificación del modelo NASNetMobile

	precision	recall	f1-score	support
bacterial_spot	0.9038	0.9527	0.9276	444.0000
early_blight	0.8244	0.8356	0.8300	444.0000
healthy	0.9365	0.9302	0.9333	444.0000
late_blight	0.8972	0.9234	0.9101	444.0000
leaf_mold	0.8981	0.8739	0.8858	444.0000
mosaic_virus	0.9678	0.9482	0.9579	444.0000
septoria_leaf_spot	0.8879	0.8559	0.8716	444.0000
target_spot	0.8517	0.8536	0.8526	444.0000
twospotted_spider_mite	0.8951	0.8649	0.8797	444.0000
yellow_leaf_curl_virus	0.9560	0.9797	0.9677	444.0000
accuracy	0.9018	0.9018	0.9018	0.9018
macro avg	0.9019	0.9018	0.9016	4440.0000
weighted avg	0.9019	0.9018	0.9016	4440.0000

Figura 79

Matriz de confusión del modelo NASNetMobile

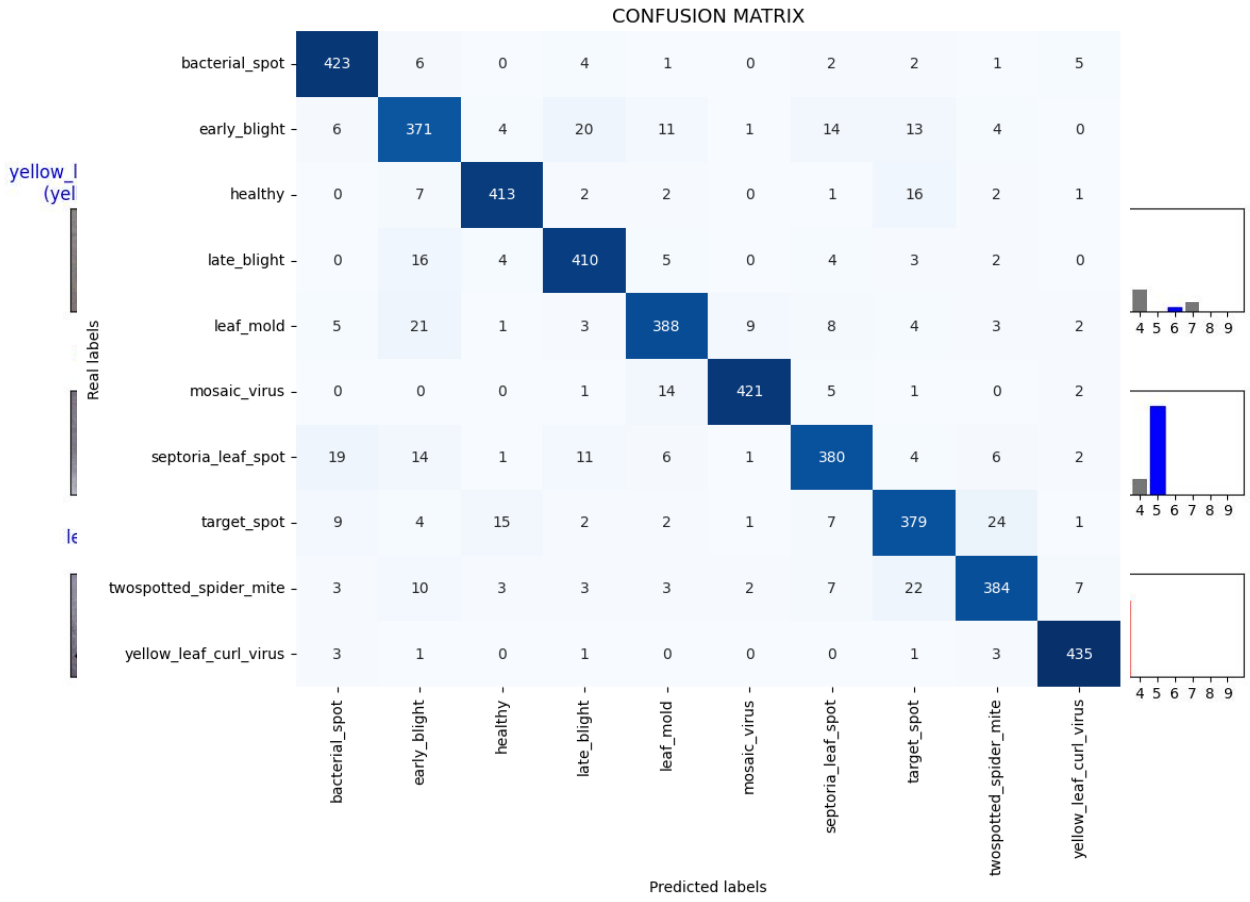


Figura 80

Ejemplo de predicciones del modelo NASNetMobile

Figura 81
Codificación de Storycard Menu Screen

MainActivity.kt

```kotlin
package com.vanskarner.tomatecare.ui

import ...

@AndroidEntryPoint  Vanskarner
class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding
    private val viewModel: MainViewModel by viewModels()
    @Inject
    lateinit var customNavBottom: CustomNavigationBottomNav

    override fun onCreate(savedInstanceState: Bundle?) {...}

    private fun setupView() {...}

    private fun setupViewModel() {...}

}
```

MainViewModel.kt

```kotlin
package com.vanskarner.tomatecare.ui

import ...

@HiltViewModel  Vanskarner
class MainViewModel @Inject constructor(): ViewModel() {
    private val _hideBottomNav = MutableLiveData<Unit>()
    private val _showBottomNav = MutableLiveData<Selection>()

    val hideBottomNav: LiveData<Unit> = _hideBottomNav
    val showBottomNav: LiveData<Selection> = _showBottomNav

    fun hideBottomNavigation() {...}

    fun showBottomNavigation(selection: Selection) {...}

}
```

Figura 82

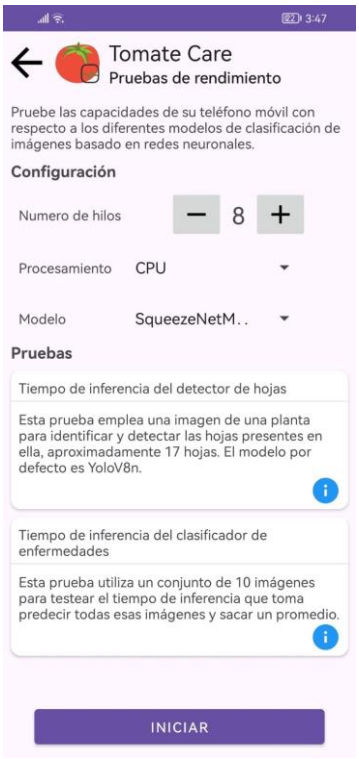
Codificación de Activity Logs Screen

LogsFragment.kt

```
1   package com.vanskarner.tomatecare.ui.activitylogs
2
3   import ...
17
18  @AndroidEntryPoint  Vanskarner +1
19  internal class LogsFragment : BaseBindingFragment<FragmentLogsBinding>() {
20
21      private val viewModel: LogsViewModel by viewModels()
22
23      @Inject
24      lateinit var logsAdapter: LogsAdapter
25      private var selectOptions = SelectionOptions.Select
26
27      override fun inflateBinding{...}
32
33      override fun setupView() {...}
61
62      override fun setupViewModel() {...}
75
76      private fun showLogs(list: List<LogModel>) = logsAdapter.updateList(list)  Vanskarner
77
78      private fun showMsgNoItemSelected() = showToast("No ha seleccionado ningun elemento")
79
80      private fun showDeleteDialog(onAccept: () -> Unit) {...}
95
96      private fun showAllCheckboxes() {...}
102
103     private fun markAllCheckboxes() {...}
108
109     private fun hideAllCheckboxes() {...}
115
116     private fun setDeleteVisibility(isVisible: Boolean) {...}
```

```kotlin
package com.vanskarner.tomatecare.ui.activitylogs

import ...

@HiltViewModel  Vanskarner +1
internal class LogsViewModel (...) : ViewModel() {
    private val _list = MutableLiveData<List<LogModel>>()
    private val _msgDelete = MutableLiveData<Unit>()
    private val _msgNoItemSelected = MutableLiveData<Unit>()
    private val _restart = MutableLiveData<Unit>()
    private var _error = MutableLiveData<String>()
    private var fullList = mutableListOf<LogModel>()
    private var filterList = mutableListOf<LogModel>()

    val list: LiveData<List<LogModel>> = _list
    val msgDelete: LiveData<Unit> = _msgDelete
    val noItemSelected: LiveData<Unit> = _msgNoItemSelected
    val restart: LiveData<Unit> = _restart
    val error: LiveData<String> = _error

    fun getData() {...}

    fun checkSelections() {...}

    fun deleteSelectedItems() {...}

    fun filterByNote(name: String) {...}

    private fun showError(error: Throwable) {...}

}
```

Figura 83
Codificación de Storycard del Test Screen



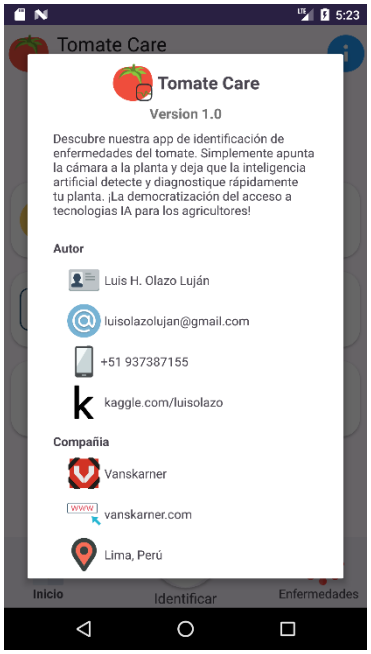
PerformanceFragment.kt

```kotlin
package com.vanskarner.tomatecare.ui.performance

import ...

@AndroidEntryPoint  LuisOlazo +1
internal class PerformanceFragment : BaseBindingFragment<FragmentPerformanceTestBinding>() {

    private val viewModel: PerformanceViewModel by viewModels()

    override fun inflateBinding{...}

    override fun setupView() {...}

    override fun setupViewModel() {...}

    private fun showInferenceResults(inferencesInMs: Pair<String, String>) {...}

    private fun showDialogTestImages(inputStream: InputStream) {...}

    private fun goToStartFragment() {...}

}
```

Project manceViewModel.kt

```kotlin
@HiltViewModel  LuisOlazo +1
internal class PerformanceViewModel (...) : ViewModel() {
    private val _threads = MutableLiveData<Int>()
    private val _processing = MutableLiveData<GenericListModel>()
    private val _models = MutableLiveData<GenericListModel>()
    private val _error = MutableLiveData<String>()
    private val _detectionImage = MutableLiveData<InputStream>()
    private val _classificationImage = MutableLiveData<InputStream>()
    private val _loading = MutableLiveData<Boolean>()
    private val _inferenceResults = MutableLiveData<Pair<String, String>>()

    val threads: LiveData<Int> = _threads
    val processing: LiveData<GenericListModel> = _processing
    val models: LiveData<GenericListModel> = _models
    val error: LiveData<String> = _error
    val detectionImage: LiveData<InputStream> = _detectionImage
    val classificationImage: LiveData<InputStream> = _classificationImage
    val loading: LiveData<Boolean> = _loading
    val inferenceResults: LiveData<Pair<String, String>> = _inferenceResults

    private var maxThreads: Int = 0

    fun showConfigForTest() {...}

    fun showTestImageForDetection() {...}

    fun showTestImageForClassification() {...}

    fun startTest(posProcessing: Int, posModels: Int) {...}

    fun decreaseThreads() {...}
```

Figura 84
Codificación de Storycard del Info Screen




```
private fun showInfoDialog() {
    val bindingInfo = DialogInfoAppBinding.inflate(layoutInflater)
    val builder = AlertDialog.Builder(requireContext())
    builder.setView(bindingInfo.root)
    val alertDialog = builder.create()
    val actualContext = requireContext()
    var versionName = "Desconocido"
    try {
        val packageInfo =
            actualContext.packageManager.getPackageInfo(actualContext.packageName, 0)
        versionName = packageInfo.versionName
    } catch (_: PackageManager.NameNotFoundException) {
        val dialogMsg = "No se pudo obtener la versión"
        Toast.makeText(actualContext, dialogMsg, Toast.LENGTH_SHORT).show()
    }
    bindingInfo.tvAppVersion.text = "Version {versionName}"
    alertDialog.show()
}
```

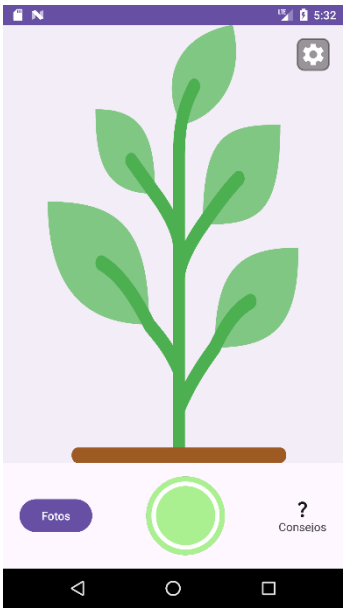
Figura 85
Codificación de Storycard del Diseases Screen

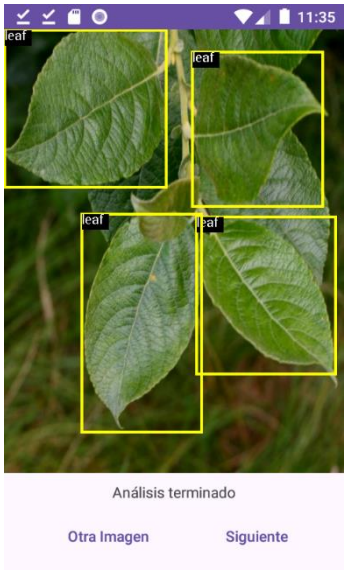
DiseasesFragment.kt

```kotlin
package com.vanskarner.tomatecare.ui.diseases

import ...

@AndroidEntryPoint  Vanskarner +1
internal class DiseasesFragment : BaseBindingFragment<FragmentDiseasesBinding>() {
    @Inject
    lateinit var diseaseDialog: DiseaseDialog

    @Inject
    lateinit var diseaseAdapter: DiseaseAdapter
    private val viewModel: DiseaseViewModel by viewModels()
    private val viewModelActivity: MainViewModel by activityViewModels()
    private val args: DiseasesFragmentArgs by navArgs()

    override fun inflateBinding{...}

    override fun setupView() {...}

    override fun setupViewModel() {...}

    private fun showDiseases(list: List<DiseaseModel>) = diseaseAdapter.updateList(list)

    private fun showDiseaseDetail(item: DiseaseDetailModel) ={...}

}
```

```kotlin
package com.vanskarner.tomatecare.ui.diseases

import ...

@HiltViewModel  Vanskarner +1
internal class DiseaseViewModel (...) : ViewModel() {

    private val _diseases = MutableLiveData<List<DiseaseModel>>()
    private val _diseaseDetail = MutableLiveData<DiseaseDetailModel>()
    private val _moreInfo = MutableLiveData<String>()
    private val _error = MutableLiveData<String>()
    private val filterList = mutableListOf<DiseaseModel>()
    private val fullList = mutableListOf<DiseaseModel>()

    val diseases: LiveData<List<DiseaseModel>> = _diseases
    val diseaseDetail: LiveData<DiseaseDetailModel> = _diseaseDetail
    val moreInfo: LiveData<String> = _moreInfo
    val error: LiveData<String> = _error

    fun startInfo(keyCode: String) {...}

    fun findDisease(diseaseId: Int) {...}

    fun filterByName(name: String) {...}

    private fun showError(error: Throwable) {...}

}
```

Figura 86
Codificación de Storycard del Capture Screen





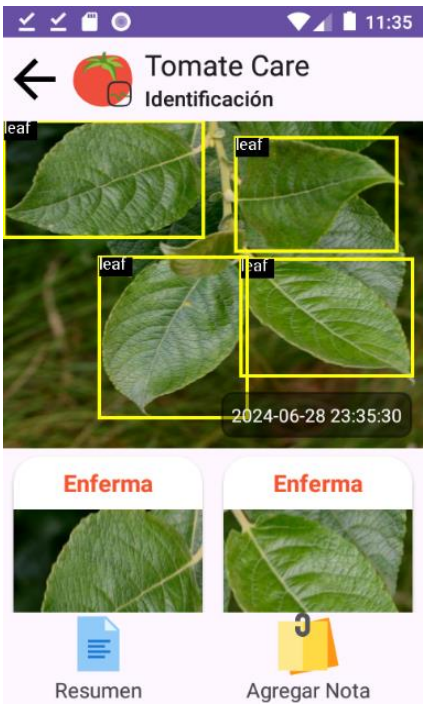
CaptureFragment.kt

```kotlin
package com.vanskarner.tomatecare.ui.capture

import ...

@AndroidEntryPoint  LuisOlazo +1
internal class CaptureFragment : BaseBindingFragment<FragmentCaptureBinding>() {

    @Inject
    lateinit var settingDialog: SettingDialog

    @Inject
    lateinit var advicesDialog: AdvicesDialog
    private val viewModel: CaptureViewModel by viewModels()
    private val viewModelActivity: MainViewModel by activityViewModels()
    private var currentPhotoPath: String = ""
    private val cameraRequest = registerForTakePicture {...}
    private val galleryRequest = registerForGallery {...}

    override fun inflateBinding{...}

    override fun setupView() {...}

    override fun setupViewModel() {...}

    private fun registerForTakePicture(onSuccess: () -> Unit): ActivityResultLauncher<Uri> {...}

    private fun registerForGallery(onSuccess: () -> Unit): ActivityResultLauncher<String> {...}

    private fun createTempImageUri(onSuccess: (Uri) -> Unit) {...}

    private fun createTempImageFile(prefix: String = "Plant"): File {...}
```

Project reViewModel.kt

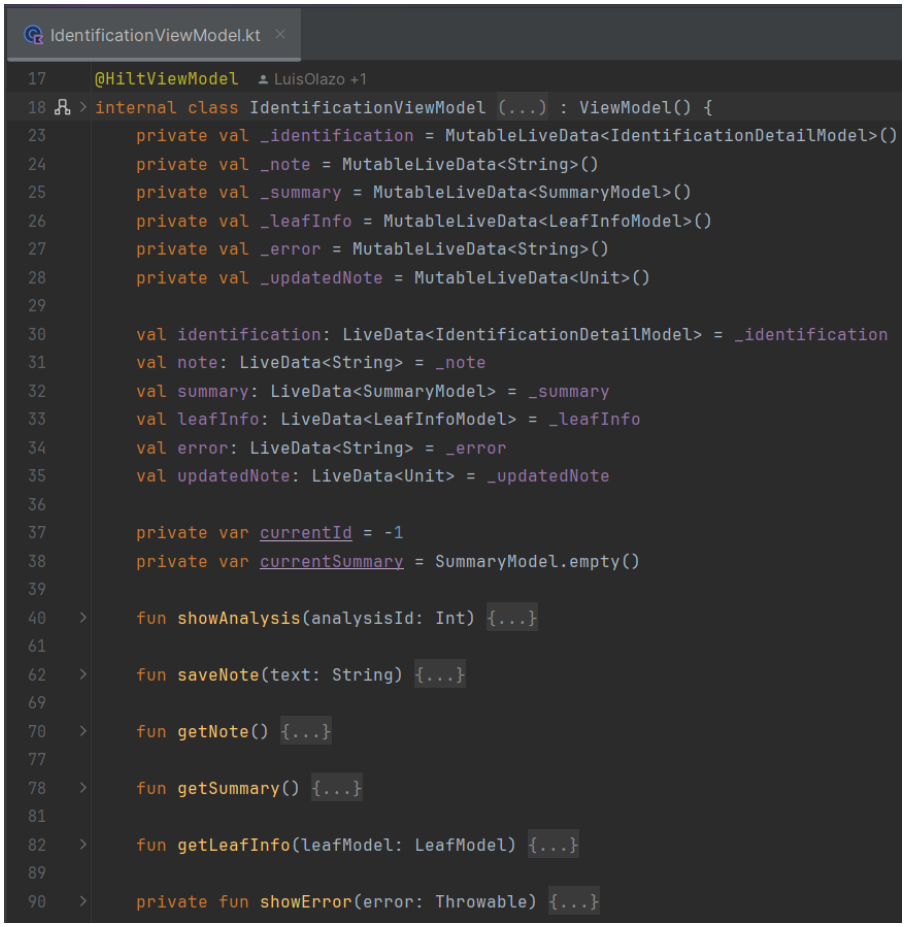
```kotlin
internal class CaptureViewModel (...) : ViewModel() {

        private val _setting = MutableLiveData<SettingModel>()
        private val _startVisibility = MutableLiveData( value: true)
        private val _loadingVisibility = MutableLiveData( value: false)
        private val _analysisFinishedVisibility = MutableLiveData( value: false)
        private val _defaultImage = MutableLiveData<Unit>()
        private val _error = MutableLiveData<String>()
        private val _idLog = MutableLiveData<Int>()
        private val _boundingBoxes = MutableLiveData<List<BoundingBoxModel>>()

        val setting: LiveData<SettingModel> = _setting
        val startVisibility: LiveData<Boolean> = _startVisibility
        val loadingVisibility: LiveData<Boolean> = _loadingVisibility
        val analysisFinishedVisibility: LiveData<Boolean> = _analysisFinishedVisibility
        val defaultImage: LiveData<Unit> = _defaultImage
        val error: LiveData<String> = _error
        val idLog: LiveData<Int> = _idLog
        val boundingBoxes: LiveData<List<BoundingBoxModel>> = _boundingBoxes

        private var settingModel = analysisComponent.getConfigParams().toModel()
        private var analysisId = 0

    fun getSetting() {...}

    fun updateSetting(setting: SettingModel) {...}

    fun analyzeImage(imgPath: String) {...}

    fun nextScreen() {...}

    fun showStart() {...}
```

Figura 87
Codificación de Storycard del Identification Screen



IdentificationFragment.kt

```
@AndroidEntryPoint  Vanskarner +1
internal class IdentificationFragment : BaseBindingFragment<FragmentIdentificationBinding>() {

    @Inject
    lateinit var leafAdapter: LeafAdapter

    @Inject
    lateinit var recommendationsAdapter: RecommendationsAdapter
    private val args: IdentificationFragmentArgs by navArgs()
    private val viewModel: IdentificationViewModel by viewModels()

    override fun inflateBinding{...}

    override fun setupView() {...}

    override fun setupViewModel() {...}

    private fun chooseCaptureOrLogsFragment(fromCapture: Boolean) {...}

    private fun showLeaves(imgPath: String, list: List<LeafModel>) {...}

    private fun showLeafInfoDialog(model: LeafInfoModel) {...}

    private fun showSummary(summaryModel: SummaryModel) {...}

    private fun showRecommendations(list: List<RecommendationModel>) {...}

    private fun showAddNote(note: String, onAccept: (text: String) -> Unit) {...}

    private fun goToDiseasesFragment(keyCode: String) {...}
```



4. Fase IV: Estabilización

En esta fase se logra integrar toda la funcionalidad de la aplicación, además se comprueba que la aplicación completa funcione correctamente.

Tabla 27

Recomendación mínima del equipo móvil

DISPOSITIVO MÓVIL	
Hardware	2Gb RAM Procesador Snapdragon 410 quad-core (1.2GHz) Cámara 10 MP
Software	Sistema Operativo: A partir de Android Nougat Nivel de api Android: A partir de 7.0

5. Fase V: Pruebas del sistema

5.1. Prueba unitaria 01: Menú Principal

Tabla 28 Prueba del Módulo Menú Principal

Código	**M01**	Nombre	**Módulo Menú Principal**
Objetivo		La aplicación muestra la escena del menú principal con las siguientes opciones: inicio, Identificación y enfermedades. Caso 1: Al seleccionar el botón inicio se le mostrará las opciones de registros de actividad y pruebas de rendimiento. Caso 2: Al seleccionar el botón de identificación se le mostrará la pantalla de capturar o seleccionar la imagen de la planta. Caso 3: Al seleccionar el botón de enfermedades se mostrará solo aquellas enfermedades que identifica la aplicación.	
Pasos		1.Ejecutar la aplicación, previamente instalada.	
Resultados obtenidos		Se visualiza la escena del menú principal con sus respectivas opciones: inicio, identificación y enfermedades. Al seleccionar el botón de inicio muestra las opciones de registros y rendimiento. Al seleccionar el botón de identificación muestra de captura o selección de la imagen de la planta como también la opción de configuración. Al seleccionar el botón de enfermedades muestra la información de las enfermedades que identificara la aplicación.	

5.2. Prueba unitaria 02: Información

Tabla 29

Prueba del Módulo Información

Código	**M02**	Nombre	**Módulo Información**
Objetivo		La aplicación muestra la escena de información referente a la app y el desarrollador como sus enlaces de contacto.	
Pasos		1. Ejecutar la aplicación, previamente instalada. 2. Situarse en el menú principal en la opción inicio. 3. Seleccionar el botón info ubicado en la esquina superior derecha.	
Resultados obtenidos		Visualizar un dialogo con la información referente a la app y sobre los enlaces de interés ligados al desarrollador.	

5.3. Prueba unitaria 03: Registros de Actividad

Tabla 30

Prueba del Módulo Registros de Actividad

Código	**M03**	**Nombre**	**Módulo Registros de Actividad**
Objetivo		La aplicación muestra los registros de los análisis realizados en el módulo de captura de imágenes. Asimismo, se cuenta con la función de búsqueda, la opción de eliminar y también permite abrir el detalle de un elemento seleccionado.	
Pasos		1. Ejecutar la aplicación, previamente instalada. 2. Situarse en el menú principal en la opción inicio. 3. Seleccionar el botón registros de actividad	
Resultados obtenidos		Visualizar una lista resumida de los análisis realizados con la aplicación, además de las opciones de filtrado por notas, seleccionar para eliminar y detalle de registro. Al seleccionar la opción de filtrado permite buscar según la nota registrada.	

	Al seleccionar el botón de eliminar permite escoger que elementos se eliminar para confirmar posteriormente esta acción. Al seleccionar cualquier elemento de la lista se podrá ver el detalle de ese registro en el módulo de identificación.

5.4. Prueba unitaria 04: Pruebas de rendimiento

Tabla 31

Prueba del Módulo Rendimiento

Código	**M04**	Nombre	**Módulo Rendimiento**
Objetivo		La aplicación muestra el tiempo de inferencia en base al modelo y configuración elegida para la prueba de rendimiento. Existen 2 pruebas, el tiempo de inferencia del modelo de clasificación y el tiempo de inferencia del modelo de detección. Asimismo, se cuenta con la opción de ajustes de número de hilos, el procesamiento y el modelo a usar, los cuales son válidos solo para el modelo de clasificación.	
Pasos		1. Ejecutar la aplicación, previamente instalada. 2. Situarse en el menú principal en la opción inicio. 3. Seleccionar el botón pruebas de rendimiento.	
Resultados obtenidos		Visualizar información referente a los tipos de prueba disponibles, así como las opciones de configuración que afectaran solo al modelo de clasificación, además de un botón para iniciar las pruebas. Al seleccionar número de hilos se incrementará o disminuirá según la capacidad del teléfono.	

	Al seleccionar procesamiento aparece solo 2 opciones: “CPU” y “GPU”. Al seleccionar modelo aparece solo 5 opciones: “MobileNetV3Small”, “MobileNetV3Large”,” MobileNetV2”, “SqueezeNet” y “NasNetMobile”. Al seleccionar la opción iniciar se muestra los tiempos que ha tomado cada tipo de prueba.

5.5. Prueba unitaria 05: Enfermedades

Tabla 32

Prueba del Módulo Enfermedades

Código	**M05**	Nombre	**Módulo Enfermedades**
Objetivo		La aplicación muestra el listado de enfermedades que identifica la aplicación, así como también información del detalle de cada una. Además, dispone de la función de buscar por el nombre de la enfermedad.	
Pasos		1. Ejecutar la aplicación, previamente instalada. 2. Situarse en el menú principal en la opción enfermedades.	
Resultados obtenidos		Se visualizar la lista de enfermedades disponibles que identifica la aplicación, así como también se encuentra la barra de búsqueda. Al situarse en la barra de búsqueda e introducir el nombre de la enfermedad se filtra los resultados deseados. Al seleccionar cualquier elemento de la lista, se abre un cuadro de dialogo con información relevante de la enfermedad.	

5.6. Prueba unitaria 06: Captura de Imágenes

Tabla 33

Prueba del Módulo Captura de Imágenes

Código	**M06**	Nombre	**Módulo Captura de Imágenes**
Objetivo		La aplicación muestra las opciones relacionadas a la configuración, consejos, selección de fotos y captura de imagen, asimismo por defecto se mostrará la imagen de una planta dibujada que indicará el área donde se mostrará la imagen captura o seleccionada. La opción de configuración permite modificar solo los parámetros para la función del modelo de clasificación. Una vez que se ingresa la imagen, se procederá a realizar la tarea de detección de hojas y clasificación de hojas, luego los resultados del análisis se guardan de forma en la base de datos local del dispositivo.	
Pasos		1. Ejecutar la aplicación, previamente instalada. 2. Situarse en el menú principal. 3. Seleccionar el botón identificación	
Resultados obtenidos		Se visualizar las opciones de configuración, consejos, fotos y el botón de captura. Al seleccionar configuración se abre un cuadro de dialogo que permite modificar los parámetros para el análisis de la imagen. Donde resultados máximos hace referencia a la cantidad de hojas máximas de la planta a identificar, luego el número de hilos está destinado si se utiliza la CPU, por otro lado, delegar indica que unidad llevara a cabo el procesamiento, y por último el modelo especifica que red neuronal convolucional utilizar para la clasificación. Al seleccionar el botón consejos se abre un cuadro de dialogo que permite visualizar como tomar la foto. Al seleccionar el botón fotos se sale de la aplicación para que se escoja una imagen de la galería.	

	Al seleccionar el botón de captura se habilita el uso de la cámara para tomar la foto de la planta. Una vez se selecciona o capturo una imagen de la planta se muestra una barra de progreso que indica el proceso de detección, clasificación y guardado del resultado.

5.7. Prueba unitaria 07: Identificación

Tabla 34

Prueba del Módulo Identificación

Código	**M07**	Nombre	**Módulo Identificación**
Objetivo		La aplicación muestra el resultado del análisis que ha sido previamente guardo, mostrando la imagen de la planta capturada con sus cuadros delimitadores que identifican las hojas, asimismo se muestra la clasificación de cada hoja detectada. Se dispone de las opciones de resumen y agregar nota, así como mostrar el detalle de cada hoja identificada y clasificada.	
Pasos		1. Ejecutar la aplicación, previamente instalada. 2. Situarse en el menú principal. 3. Seleccionar el botón identificación y capturar o escoger una imagen. 4. Luego de la identificación pulsar siguiente. 5. También se puede acceder desde los registros de actividad al seleccionar algún elemento para mostrar su detalle.	
Resultados obtenidos		Se visualizar la imagen de la planta con sus cuadros delimitadores detectando las hojas presentes, además sus hojas detectadas están clasificadas según el tipo de enfermedad que se ha encontrado.	

	Al seleccionar cualquier hoja, se muestra información muy resumida sobre el porcentaje de la predicción y sobre la enfermedad. Asimismo, si selecciona “más información” se abrirá el módulo de enfermedades para ver el detalle. Al seleccionar la opción resumen, se muestra información sintetizada del análisis llevado a cabo, como el tiempo de inferencia del modelo de detección, el tiempo de inferencia del modelo de clasificación, cantidad de hojas analizadas, cantidad de tipos de enfermedades presentes y el botón de recomendaciones en base a la enfermedad identificada. Al seleccionar la opción agregar nota, se muestra el apunte ingresado, así como también la función de editarlo.