Pontificia Universidad Javeriana de Cali

Facultad de Ingeniería y Ciencias

Carrera de Ingeniería de sistemas y computación

Proyecto de grado

Modelos de predicción del tiempo de vida útil y clasificación de madurez en aguacates Hass mediante Deep Learning

Sebastian Enriquez Estrada

Samuel Escobar Llanos

Director:

Cristian Alejandro Torres Valencia

Codirector:

Juan Carlos Martínez Arias

Santiago de Cali, 2026

Abstract

This project developed an intelligent system combining computer vision and deep learning to classify the ripeness stage of Hass avocados and predict their remaining shelf life in days. The initiative arose in response to the high levels of food waste generated during the distribution and commercialization of this fruit in the region. Two models were implemented: a classification model that identifies ripeness across four maturity stages, and a regression model that estimates the number of days remaining until complete deterioration, both trained on a publicly available dataset of Hass avocado images captured across different ripening states.

A functional interface was integrated to allow users to upload photographs and visualize both the predicted ripeness class and the estimated shelf life. The system enabled the analysis of avocado ripeness stages, promoting their consumption at the right time and thereby reducing the possibility of food waste, providing a practical and objective tool for distributors, retailers, and consumers to make informed decisions based on validated performance metrics. Results include two deep learning models with validated performance metrics, a comprehensive analysis of their effectiveness and accuracy, and a functional interface prototype integrating both prediction systems, thereby contributing to the application of artificial intelligence technologies in the Colombian agricultural sector.

Keywords: Deep learning, computer vision, ripeness classification, shelf life prediction, Hass avocado, image processing.

Resumen

Este proyecto desarrolló un sistema inteligente que combina visión por computador y aprendizaje profundo para clasificar la madurez del aguacate Hass y predecir su vida útil en días. La iniciativa surgió como respuesta al alto nivel de desperdicio de esta fruta en los procesos de distribución y comercialización en la región. El diseño contempló dos modelos, uno de clasificación en cuatro etapas de madurez y otro de regresión para estimar los días hasta el deterioro completo, ambos entrenados con una base de datos pública de imágenes de aguacates en distintos estados de maduración.

Se integró una interfaz que permite cargar fotografías y visualizar tanto la clase de madurez como la predicción de vida útil. El sistema permitió el análisis del estado de madurez de aguacates, promoviendo su consumo en tiempos adecuados y reduciendo así la posibilidad de desperdicio, brindando una herramienta práctica para distribuidores, comerciantes y consumidores mediante decisiones objetivas basadas en métricas validadas. Los resultados obtenidos incluyen dos modelos de aprendizaje profundo con métricas de desempeño validadas, un análisis de efectividad y precisión, y un prototipo funcional de interfaz que integra ambos sistemas, contribuyendo así a la aplicación de tecnologías de inteligencia artificial en el sector agrícola colombiano.

Palabras clave: Aprendizaje profundo, visión por computador, clasificación de madurez, predicción de vida útil, aguacate Hass, procesamiento de imágenes.

ÍNDICE GENERAL

INTRODUCCIÓN 11

1. DESCRIPCIÓN DEL PROBLEMA 12

1.1. Planteamiento del Problema 12

1.1.1. Formulación 13

1.1.2. Sistematización 13

1.2. Objetivos 13

1.2.1. Objetivo General 13

1.2.2. Objetivos Específicos 14

1.3. Alcance 14

2. MARCO DE REFERENCIA 15

2.1. Marco Teórico 15

2.1.1. Aguacate Hass 15

2.1.2. Visión por Computador y Preprocesamiento 16

2.1.3. Espacios de Color 19

2.1.4. Extracción de características 20

2.1.5. Aprendizaje de Máquina 21

2.1.6. Aprendizaje Profundo 25

2.1.7. Métricas de Evaluación 30

2.2. Trabajos Relacionados 35

3. DESARROLLO DEL PROYECTO 37

3.1. Preparación y preprocesamiento del conjunto de Datos 37

3.1.1. Identificación y selección del conjunto de datos 37

3.1.2. Análisis y depuración del conjunto de datos 38

3.1.3. Segmentación de los aguacates para los modelos clásicos de aprendizaje de máquina 39

3.1.4. Preprocesamiento de imágenes para los modelos de aprendizaje profundo 41

3.1.5. Extracción de características para los modelos clásicos de aprendizaje de máquina 42

3.2. Diseño e implementación de modelos de clasificación y regresión 46

3.2.1. Estrategia general de la implementación 46

3.2.2. Modelos clásicos de aprendizaje de máquina 47

3.2.3. Modelos de aprendizaje profundo 51

3.3. Evaluación del desempeño de los modelos 54

3.3.1. Evaluación de los modelos clásicos de aprendizaje automático 54

3.3.2. Evaluación de los modelos de aprendizaje profundo CNN 60
3.3.3. Resultados de la experimentación con Fine-Tuning 70
3.3.4. Elección de los modelos para clasificación y regresión 74
3.4. Desarrollo de la interfaz funcional 76
3.4.1. Diseño de la interfaz 76
3.4.2. Estructura del sistema 79
3.4.3. Implementación del frontend 79
3.4.4. Implementación del backend 80
3.4.5. Limitaciones del sistema 80
4. CONCLUSIONES Y TRABAJOS FUTUROS 82
4.1. Conclusiones 82
4.2. Trabajos Futuros 83
5. REFERENCIAS BIBLIOGRÁFICAS 85
6. ANEXOS 93

ÍNDICE DE FIGURAS

1. Estados de madurez del aguacate Hass. Obtenido de [10]. 16
2. Resultado de la segmentación aplicada sobre distintas imágenes, mostrando la imagen original (a) junto a las máscaras generadas por el modelo en diferentes configuraciones (b-k). Obtenido de [16]. 18
3. Representación visual de los espacios de color RGB (a), HSV (b) y CIELab (c). Obtenido de [20]. 20
4. Estructura general de una Red Neuronal Convolucional (CNN). Obtenido de [30]. 25
5. Arquitectura de MobileNetV3. Obtenido de [34]. 26
6. Arquitectura VGG16. Obtenido de [35]. 27
7. Arquitectura EfficientNet-B3. Obtenido de [39]. 28
8. Arquitectura ResNet-18. Obtenido de [43]. 29
9. Estructura de la matriz de confusión con sus cuatro cuadrantes: verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN). Obtenido de [54]. 33
10. Imagen representativa del conjunto de datos de la clase Verde. Obtenido de [61]. 38
11. Ejemplos representativos de las cuatro clases de madurez de una muestra seleccionada del conjunto de datos. Obtenido de [61]. 39
12. Muestra de aguacates de la clase Verde Sin Segmentación. Obtenido de [61]. 40
13. Muestra de aguacates de la clase Verde Con Segmentación. Obtenido de [61]. 40
14. Flujo del proceso de extracción de características. 42
15. Muestra de aguacate de la clase premaduro segmentado, junto a sus histogramas normalizados del canal Hue, con diferentes cantidades de bins. 45
16. Ejemplos de los histogramas normalizados sobre el canal Hue por cada clase. 45
17. Pipeline completo de los modelos de clasificación. 48
18. Pipeline completo de los modelos de regresión. 50
19. Cabeza y capa de salida para clasificación y regresión 52
20. Configuraciones de fine-tuning del backbone de ResNet-18 según el porcentaje de capas descongeladas. 53
21. Matriz de confusion MLP con 31 características 55
22. Matriz de confusion Random Forest con 76 características. 55
23. Gráfica de dispersión del SVR con 76 características. 58
24. Gráfica de Perdida y Accuracy por época de ResNet-18. En los dos bloques de arriba no se usó aumento de datos y en los dos bloques de abajo se hizo lo contrario. 62
25. Gráfica de Perdida y Accuracy por época de EfficientNet-B3. En los dos bloques de arriba no se usó aumento de datos y en los dos bloques de abajo se hizo lo contrario. 63
26. Gráfica de Perdida y Accuracy por época de VGG16. En los dos bloques de arriba no se usó aumento de datos y en los dos bloques de abajo se hizo lo contrario. 63
27. Gráfica de Perdida y Accuracy por época de MobileNetV3. En los dos bloques de arriba no se usó aumento de datos y en los dos bloques de abajo se hizo lo contrario. 64

28. Matriz de confusion ResNet-18 sin aumento de datos....65
29. Matriz de confusion EfficientNet-B3 con aumento de datos....65
30. Gráfica de Perdida MSE EfficientNet-B3 con Aumento de datos.....68
31. Gráfica de Perdida MSE ResNet18 sin Aumento de datos.....68
32. Gráfica de dispersión de días EfficientNet-B3 con Aumento de datos....69
33. Gráfica de dispersión de días ResNet18 sin Aumento de datos.....69
34. Gráficas de Perdida y Accuracy por época de EfficientNet-B3 con distintos porcentajes de descongelamiento del backbone. 71
35. Gráficas de Perdida y Accuracy por época de ResNet-18 con distintos porcentajes de descongelamiento del backbone. 72
36. Matriz de confusion EfficientNet-B3 con distintos porcentajes de descongelamiento del backbone. 73
37. Diagrama general de la arquitectura del sistema. 76
38. Diagrama de actividades de AvoCheck. Flujo de interacción entre el usuario y el sistema.....77
39. Pantalla de inicio de AvoCheck en versión móvil y escritorio. 78
40. Pantalla de resultados de AvoCheck en versión móvil y escritorio. 78

ÍNDICE DE TABLAS

1. Resumen de las 76 características extraídas por imagen para los modelos clásicos de aprendizaje de máquina. 43
2. Resultados de las métricas de clasificación obtenidos por los modelos clásicos usando 31 y 76 características. 54
3. Resultados de las métricas de regresión obtenidos por los modelos clásicos usando 31 y 76 características 57
4. Resultados de clasificación obtenidos por las arquitecturas CNN con y sin aumento de datos....60
5. Resultados de regresión obtenidos por las arquitecturas CNN con y sin aumento de datos. 67
6. Resultados de las métricas de clasificación con diferentes porcentajes de descongelamiento en la segunda etapa de fine-tuning en los cuatro modelos. 70

INTRODUCCIÓN

En la actualidad, el desperdicio de alimentos representa uno de los desafíos más urgentes para la seguridad alimentaria global. Aproximadamente 1/3 de los alimentos producidos para consumo humano se pierde o desperdicia cada año, lo que equivale a 1.3 billones de toneladas. Colombia, siendo uno de los principales productores de aguacate en América Latina, enfrenta esta problemática especialmente en la cadena de distribución del aguacate Hass, donde hasta el 30% de los frutos se desperdician durante el proceso de selección y empaque, y un 5% se pierden en los supermercados. Esta situación genera pérdidas económicas para distribuidores e insatisfacción en consumidores que adquieren productos sin cumplir expectativas de calidad o conservación. La raíz del problema radica en la falta de herramientas objetivas para evaluar el estado de madurez y estimar el tiempo de vida útil, ya que los métodos tradicionales de inspección visual son subjetivos.

Frente a esta problemática, el presente proyecto desarrolló un sistema basado en aprendizaje profundo y visión por computador para clasificar el estado de madurez del aguacate Hass y predecir el tiempo de días hasta su deterioro completo mediante análisis de características visuales. Se implementaron 2 modelos, uno de clasificación que identifica el estado de madurez en 4 categorías, y uno de regresión que estima los días restantes de vida útil. Ambos fueron entrenados con una base de datos pública de imágenes de aguacate Hass en diferentes estados de maduración, aplicando técnicas de preprocesamiento para garantizar una cantidad suficiente de datos consistentes que favorezcan el entrenamiento de los modelos. Adicionalmente, se desarrolló una interfaz funcional para cargar imágenes y obtener la clasificación de madurez y predicción de vida útil.

El sistema permitió el análisis del estado de madurez de aguacates Hass, promoviendo su consumo en tiempos adecuados y reduciendo así la posibilidad de desperdicio, proporcionando una herramienta objetiva y de fácil uso que podría permitir optimizar la gestión de inventarios, mejorar la planificación de ventas, y aumentar la satisfacción del consumidor. Los resultados obtenidos incluyen 2 modelos de aprendizaje profundo con métricas de desempeño validadas, un análisis de la efectividad y precisión de los modelos, y un prototipo funcional de interfaz que integra ambos sistemas de predicción. Así, el proyecto pudo contribuir al avance de soluciones tecnológicas aplicadas al sector agrícola colombiano, demostrando la viabilidad de usar técnicas de aprendizaje profundo para la estimación del estado de madurez y la vida útil del aguacate Hass.

1. DESCRIPCIÓN DEL PROBLEMA

1.1. Planteamiento del Problema

A nivel mundial, las pérdidas y el desperdicio de alimentos son uno de los desafíos más críticos para la seguridad alimentaria. Según un estudio realizado por Gustavsson et al. para la FAO, aproximadamente un tercio de los alimentos producidos para el consumo humano se pierde o se desperdicia globalmente, lo cual equivale aproximadamente a 1.3 billones de toneladas por año [1]. Además, en los países industrializados como Estados Unidos por ejemplo, el desperdicio de alimentos per cápita por parte de los consumidores alcanza entre los 95-115kg/año, mientras que, en la zona sur de África y el sur y sudeste de Asia, esta cifra es solamente de 6-11kg/año [1], indicando que el problema del desperdicio se concentra principalmente en países con mayor poder adquisitivo, donde los consumidores suelen carecer de conocimiento o prácticas adecuadas para aprovechar correctamente los productos frescos. Asimismo, una fruta como el aguacate, siendo una fruta climatérica, presenta desafíos particulares debido a su rápido proceso de maduración después de la cosecha, lo que genera dificultades en su manejo postcosecha y comercialización [2].

Hoy en día, Colombia se ha consolidado como uno de los principales productores de aguacate en América Latina, con una producción en constante crecimiento que busca posicionarse en muchos mercados internacionales [3]. Los supermercados de la ciudad de Cali manejan volúmenes altos de esta fruta, pero enfrentan un impacto económico negativo debido a la dificultad para evaluar correctamente el estado de madurez del producto y estimar su tiempo de vida útil. Esta situación no solo implica pérdidas económicas para los distribuidores, sino que también limita la satisfacción de los consumidores, quienes en muchos casos adquieren un producto que no cumple con sus expectativas en cuanto a calidad, sabor o tiempo de conservación. La problemática se agrava por el hecho de que hasta el 30% de los aguacates se desperdician durante el proceso de selección y empaque por métodos inadecuados para medir su madurez, y un 5% adicional se pierde en los supermercados [4], situación que se refleja claramente en el contexto de los supermercados locales donde la falta de herramientas precisas para determinar el punto óptimo de madurez genera pérdidas económicas tanto para los establecimientos comerciales como insatisfacción en los consumidores.

Dado que los métodos tradicionales de evaluación de madurez del aguacate se basan principalmente en la inspección visual y al tacto, estos son subjetivos y requieren de experiencia por parte de quienes lo aplican. Esto lleva a que diferentes evaluadores, tales como los productores durante la selección y empaque, los comerciantes en puntos de venta, y los consumidores finales al momento de la compra, obtengan

conclusiones distintas sobre el mismo fruto. Consecuentemente, tanto consumidores como comerciantes, se enfrentan a retos para identificar el momento óptimo de compra y consumo, lo que resulta en algunos casos en pérdidas de frutos debido a evaluaciones incorrectas, en el consumo de aguacates deteriorados, o en la compra de frutos que aún no han alcanzado su punto de madurez adecuado. Por eso, las tecnologías emergentes como el aprendizaje profundo o la visión por computador han demostrado un potencial significativo para la evaluación automática de calidad en productos agrícolas [5]. Varios estudios han reportado el uso exitoso de técnicas de machine learning para la clasificación de frutas basándose en características visuales [6]. No obstante, existe una pequeña brecha en el desarrollo de modelos específicos para aguacates Hass que no solo clasifique su estado de madurez, sino que también prediga el tiempo estimado en días del deterioro del aguacate.

1.1.1. Formulación

¿Cómo desarrollar un modelo basado en técnicas de aprendizaje profundo y visión por computador que identifique automáticamente el estado de madurez de aguacates tipo Hass y prediga el tiempo estimado en días de su deterioro completo mediante el análisis de características visuales?

1.1.2. Sistematización

Para abordar el problema, se plantean las siguientes preguntas especificas:

1.1.2.1. ¿Qué técnicas de preprocesamiento de datos son más adecuadas para optimizar la base de datos de imágenes de aguacates Hass y mejorar el entrenamiento de los modelos?

1.1.2.2. ¿Qué modelos de aprendizaje profundo son más efectivos para identificar el estado de madurez del aguacate Hass y predecir su tiempo de deterioro completo basándose en análisis de imágenes?

1.1.2.3. ¿Cómo se puede evaluar la efectividad y la precisión de los resultados obtenidos por los modelos?

1.1.2.4. ¿Cómo desarrollar una interfaz que facilite la interacción práctica con los modelos implementados?

1.2. Objetivos

1.2.1. Objetivo General

Desarrollar un sistema que clasifique el estado de madurez del aguacate tipo Hass y prediga el tiempo estimado en días hasta su deterioro completo, mediante la implementación de técnicas de aprendizaje profundo y visión por computador para el análisis de características visuales.

1.2.2. Objetivos Específicos

1.2.2.1. Preparar la base de datos de imágenes de aguacates Hass aplicando técnicas de preprocesamiento para garantizar una cantidad de datos consistentes y diversa que favorezcan el entrenamiento de los modelos de aprendizaje profundo.

1.2.2.2. Implementar modelos de aprendizaje profundo basados en técnicas de visión por computador que permitan la identificación del estado de madurez del aguacate Hass y la predicción del tiempo estimado en días hasta su deterioro completo.

1.2.2.3. Evaluar la efectividad y precisión de los modelos de aprendizaje profundo mediante métricas de desempeño para verificar su rendimiento.

1.2.2.4. Desarrollar una interfaz que permita el uso de manera práctica los modelos de identificación del estado de madurez y de predicción del tiempo de deterioro del aguacate Hass.

1.3. Alcance

El sistema se enfocará en la clasificación automática del aguacate Hass en cuatro etapas de madurez: verde, premaduro, maduro, y podrido. Esta clasificación se realizará a partir de imágenes tomadas con cámaras convencionales, las cuales constituirán la base de datos de entrenamiento y validación de los modelos. El sistema integrará dos modelos de aprendizaje profundo, un modelo de clasificación para identificar el estado de madurez del fruto y un modelo de regresión para estimar el tiempo de vida útil en días hasta el deterioro completo del aguacate. Para ello, se emplearán técnicas de visión por computador que extraen patrones relevantes de las imágenes, como color, textura y apariencia superficial, con el fin de garantizar una predicción lo más objetiva y consistente posible. Los resultados se mostrarán en una interfaz sencilla que permitirá cargar imágenes y visualizar la clasificación y predicción de manera práctica.

Es importante señalar que el prototipo estará limitado a la clasificación de madurez en las cuatro categorías mencionadas y a la predicción del tiempo de vida útil bajo condiciones controladas de captura de imágenes. Asimismo, el sistema solo utilizará características visuales específicas que sean determinantes para identificar el estado de madurez del fruto, sin incluir información de otro tipo de sensores o variables externas.

2. MARCO DE REFERENCIA

2.1. Marco Teórico

El uso de técnicas de visión por computador y aprendizaje profundo ha cobrado relevancia en la clasificación y control de calidad de frutas, al ofrecer soluciones más objetivas y consistentes que la inspección visual tradicional. Para ello, es necesario comprender conceptos clave como, primero, las características del aguacate Hass y los criterios de madurez hortícola. Posteriormente, se abordan los fundamentos de visión por computador y las técnicas de preprocesamiento de imágenes. Luego, se desarrollan conceptos de aprendizaje de máquina, incluyendo clasificación, regresión, redes neuronales convolucionales y aumento de datos. Finalmente, se describen las métricas de evaluación para cuantificar el desempeño de los modelos. Estos conceptos proporcionan el sustento teórico para desarrollar soluciones que contribuyan a reducir el desperdicio alimentario en la cadena de suministro del aguacate.

2.1.1. Aguacate Hass

2.1.1.1. Características del aguacate Hass

El aguacate Hass constituye aproximadamente el 80% de la producción mundial de aguacates. Su cáscara, de textura rugosa, cambia de verde a un tono púrpura oscuro a medida que madura, pesa entre 150 y 350 gramos y tiene una pulpa cremosa con 12-20% de aceite [7]. Se trata de una fruta climatérica, lo que significa que su maduración comienza tras la cosecha, gracias a la producción de etileno. A diferencia de otras frutas, el aguacate no madura en el árbol; su proceso de maduración empieza únicamente después de ser cosechado, y generalmente dura entre 7 y 11 días a temperatura ambiente [8].

2.1.1.2. Madurez Hortícola

La madurez hortícola se entiende como el estado de una fruta o verdura en el que sus características físicas y sensoriales cumplen con los estándares exigidos para el mercado y el consumo humano. En esta etapa, el producto presenta un aspecto visual, textura, aroma y sabor que lo hacen aceptable para su comercialización y atractivo para el consumidor final. Los atributos evaluados en la madurez hortícola suelen ser el color externo, el tamaño, forma y la apariencia superficial, donde se valoran características como brillo, uniformidad y ausencia de defectos visibles, entre otras características. La determinación de la madurez hortícola es fundamental porque influye directamente en la aceptación del producto, en su valor comercial y en su vida útil [9]. Un ejemplo visual de este proceso se puede observar en la Fig. 1, donde se ilustran los cuatro estados de madurez del aguacate Hass, diferenciados por su color y textura principalmente, desde el estado inmaduro, caracterizado por una piel verde brillante y firme, hasta el estado sobremaduro, donde la cáscara está oscura y deteriorada.

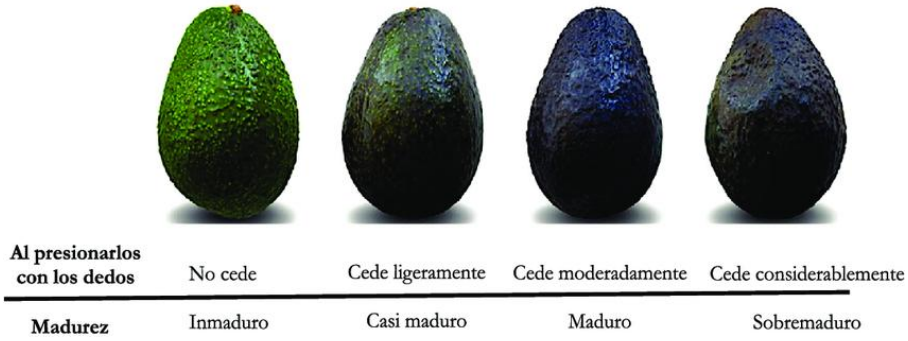


Fig.1. Estados de madurez del aguacate Hass. Obtenido de [10].

2.1.2. Visión por Computador y Preprocesamiento

2.1.2.1. Visión por Computador

La visión por computador es un campo interdisciplinario que permite a sistemas computacionales adquirir, procesar, analizar e interpretar información visual mediante procesamiento de imágenes digitales. Su objetivo como tal es permitir que las máquinas extraigan información significativa de imágenes y vídeos para realizar tareas como reconocimiento a objetos, detección de patrones, clasificación, medición de características visuales y seguimiento de movimiento. A diferencia del procesamiento de imágenes tradicional donde tanto la entrada como la salida son imágenes, en visión por computador la entrada es una imagen o vídeo y la salida puede ser una imagen mejorada, un análisis del contenido de la imagen, o incluso un comportamiento del sistema basado en ese análisis [11].

2.1.2.2. Preprocesamiento de Imágenes

El preprocesamiento de imágenes es fundamental en proyectos que utilizan imágenes capturadas por cámaras, ya que estas pueden estar sujetas a variaciones de iluminación, color y ruido. Este proceso incluye operaciones como la normalización de color, que ajusta valores de píxel para una representación homogénea; la ecualización de histograma, que mejora el contraste distribuyendo más uniformemente los niveles de gris; y la reducción de ruido, mediante filtros que eliminan imperfecciones sin afectar detalles clave. Estas transformaciones no solo mejoran la calidad visual, sino que facilitan la convergencia durante el entrenamiento y aumentan la robustez y precisión del modelo. El preprocesamiento estandariza las entradas y mitiga problemas como imágenes subexpuestas o sobreexpuestas, lo cual es esencial para mantener consistencia en las predicciones de un modelo [12].

2.1.2.2.1. Redimensionamiento

Normalmente los modelos de aprendizaje de máquina requieren que todas las imágenes de entrada tengan las mismas dimensiones, ya que procesan los datos como matrices de tamaño fijo. Dado que las imágenes de un conjunto de datos suelen tener resoluciones distintas, el redimensionamiento consiste en escalar cada imagen a un tamaño uniforme antes de ingresarla al modelo. Además de garantizar la compatibilidad con la arquitectura del modelo, porque cada arquitectura trabaja con un tamaño de imagen distinto, trabajar con imágenes de menor resolución reduce el número de parámetros que el modelo debe procesar, lo que disminuye el tiempo de entrenamiento sin sacrificar necesariamente la precisión [13].

2.1.2.2.2. Normalización

La normalización es el proceso de ajustar los valores de los píxeles de una imagen a un rango numérico estándar, entre 0 y 1, dividiendo cada valor por 255, que es el máximo valor posible en una imagen de 8 bits. Cuando los valores de entrada tienen rangos muy amplios o desiguales, el modelo puede tener dificultades para aprender de forma estable. Al normalizar, se garantiza que todos los píxeles contribuyan de manera equitativa al proceso de aprendizaje, lo que facilita la convergencia del modelo durante el entrenamiento [14].

2.1.2.3. Aumento de Datos

El aumento de datos consiste en aplicar transformaciones sobre imágenes existentes para generar versiones nuevas que mantengan la etiqueta original, con el fin de enriquecer la diversidad del conjunto de entrenamiento y evitar sobreajuste. Estas transformaciones incluyen rotaciones, volteos horizontales y verticales, escalados, cambios de brillo, recortes y ruido. Técnicas más avanzadas usan meta-aprendizaje o modelos basados en políticas óptimas. El objetivo es expandir efectivamente el espacio de entrenamiento sin recolectar nuevos datos. En contextos prácticos, la aplicación de aumento mejora la generalización y la robustez del modelo en condiciones reales, como diferentes fondos o iluminaciones [15].

2.1.2.4. Segmentación de Imágenes

La segmentación de imágenes es el proceso mediante el cual una imagen se divide en regiones, separando así el objeto de interés del fondo y de otros elementos no relevantes. El objetivo es que se trabaje únicamente sobre el objeto que importa, evitando que el entorno de la imagen afecte los resultados [12].

Cuando los métodos tradicionales de umbralización no son suficientes para lograr esta separación con precisión, se recurre a técnicas basadas en aprendizaje profundo. Una de las arquitecturas más destacadas para esta tarea es $U^2 - Net$**,** una red neuronal diseñada específicamente para identificar y delimitar el objeto principal dentro de una imagen. Su funcionamiento se basa en una estructura anidada en dos niveles: el primer

nivel analiza la imagen a diferentes escalas para capturar tanto detalles finos como el contexto general de la escena, mientras que el segundo nivel combina toda esa información para generar una máscara precisa que distingue el objeto del fondo. Gracias a este diseño, $U^2 - Net$ logra resultados precisos sin requerir un costo computacional excesivo [16]. La Fig. 2 ilustra ejemplos de segmentación generados por esta arquitectura, donde se puede apreciar cómo el modelo identifica y delimita correctamente el objeto principal en imágenes con distintos fondos y niveles de complejidad, produciendo máscaras binarias que separan con precisión el objeto de interés del resto de la imagen.

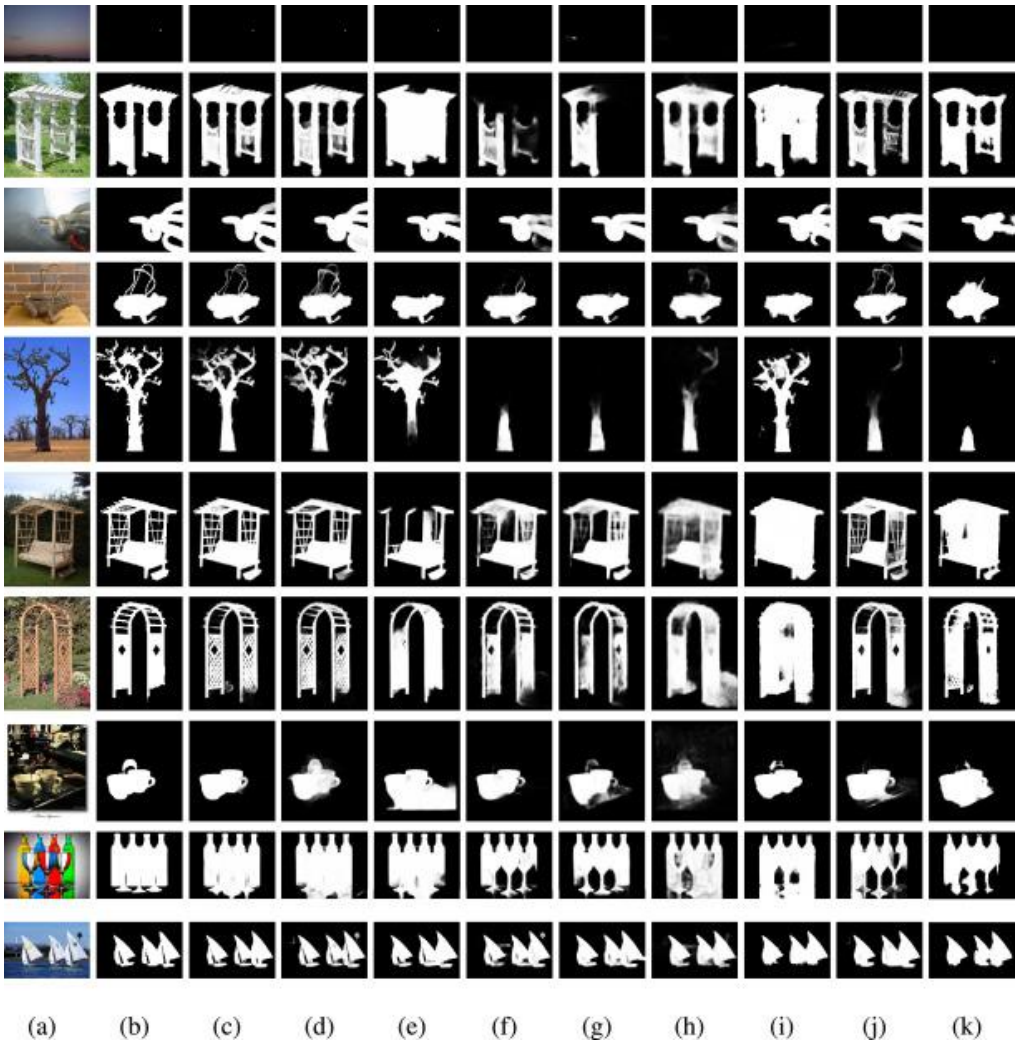


Fig.2. Resultado de la segmentación aplicada sobre distintas imágenes, mostrando la imagen original (a) junto a las máscaras generadas por el modelo en diferentes configuraciones (b-k). Obtenido de [16].

2.1.2.5. SMOTE

Synthetic Minority Over-sampling Technique, más conocido como SMOTE, es una técnica de preprocesamiento diseñada para tratar situaciones en las que una o varias clases tienen muchas menos muestras que las demás, problema conocido como desbalance de clases. Su propósito principal es que el modelo aprenda a reconocer correctamente la clase minoritaria, evitando que se sesgue hacia la clase que tiene más muestras. Para lograrlo, SMOTE genera ejemplos nuevos y sintéticos, tomando un ejemplo de la clase minoritaria, buscando otros ejemplos similares a él dentro de esa misma clase, y creando un nuevo ejemplo en un punto intermedio entre ellos. De esta forma, en lugar de repetir datos ya existentes, rellena los espacios vacíos entre ejemplos similares de la clase minoritaria, haciendo que esa clase quede mejor representada y que el modelo tenga más variedad de ejemplos para aprender de ella [17].

2.1.3. Espacios de Color

2.1.3.1. Espacio RGB

El espacio de color RGB representa cada píxel de una imagen como la combinación de tres canales: rojo (R), verde (G) y azul (B). Cada canal toma valores enteros entre 0 y 255, donde 0 indica ausencia total de ese color y 255 su máxima intensidad, dando a lugar a más de 16 millones de colores posibles. Es el espacio más común en dispositivos digitales como cámaras y pantallas, ya que es el formato nativo en que se capturan y almacenan la mayoría de las imágenes [18].

2.1.3.2. Espacio HSV

El espacio HSV organiza el color en tres componentes: Hue (matiz), Saturation (saturación) y Value (brillo). El matiz indica el tipo de color en el espectro, la saturación mide qué tan puro o intenso es ese color, y el brillo expresa cuánta luz refleja [19]. A diferencia del RGB, el HSV separa la información de color de la información de luminosidad, lo que lo hace más intuitivo y alineado con la percepción visual humana, especialmente útil cuando los cambios de iluminación pueden afectar la clasificación [18].

2.1.3.3. Espacio CIELab

El espacio CIELab representa el color mediante tres ejes: L*, que indica la luminosidad, a*, que va del verde al rojo, y b*, que va del azul al amarillo. Su principal característica es que fue diseñado para reflejar cómo el ojo humano percibe las diferencias de color, de manera que dos colores que numéricamente estén más alejados entre sí también se vean más distintos visualmente [19]. Esto lo convierte en un espacio especialmente útil para detectar cambios de color sutiles, como los que ocurren durante el proceso de maduración de una fruta, donde pequeñas variaciones en el tono pueden indicar cambios significativos en su estado.

A modo de comparación visual, la Fig. 3 muestra las diferencias estructurales entre los tres espacios de color. El cubo RGB (a), donde cada dimensión representa un canal de color primario; el cilindro HSV (b), que separa el matiz, la saturación y el brillo en componentes independientes; y la esfera CIELab (c), organizada en torno a los ejes de luminosidad y los rangos verde-rojo y azul-amarillo.

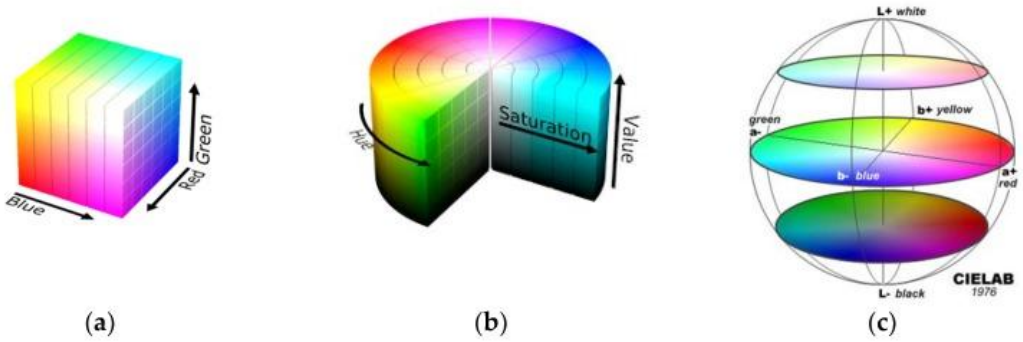


Fig. 3. Representación visual de los espacios de color RGB (a), HSV (b) y CIELab (c). Obtenido de [20].

2.1.4. Extracción de características

2.1.4.1. Características estadísticas de color

Las características estadísticas de color son valores numéricos que resumen la distribución del color en una imagen. Las más utilizadas son la media, que indica el valor promedio de intensidad de cada canal de color, y la desviación estándar, que mide qué tan dispersos están los valores de los píxeles respecto a esa media. Estas características se pueden calcular sobre diferentes espacios de color, permitiendo capturar desde el tono dominante hasta la variabilidad de brillo del objeto analizado [21]. En el contexto de clasificación de frutas, estas estadísticas son especialmente útiles porque el color cambia de forma progresiva y predecible durante el proceso de maduración.

2.1.4.2. Características de Textura (GLCM)

La Matriz de Co-ocurrencia de Niveles de Gris, conocida como GLCM, es un método estadístico para analizar la textura de una imagen. Consiste en construir una matriz que registra con qué frecuencia dos píxeles con ciertos niveles de gris aparecen juntos en una dirección y distancia determinadas dentro de la imagen [22]. A partir de esta matriz se calculan distintas métricas como el contraste, que mide las diferencias de intensidad entre píxeles vecinos; la homogeneidad, que indica qué tan uniformes son esas diferencias; la correlación, que evalúa la dependencia lineal entre píxeles adyacentes; el Angular Second Moment (ASM), que mide la uniformidad de la textura, donde valores altos indican que la superficie es homogénea y repetitiva; y la entropía, que cuantifica la complejidad o aleatoriedad de la textura [23]. Juntas, estas métricas permiten describir características visuales de la superficie del objeto que el color por sí solo no captura.

2.1.4.3. Características histogramas de color

El histograma de color es una representación que indica cuántos píxeles de una imagen tienen cada valor de intensidad dentro de un canal de color. Para construirlo, el rango de valores posibles se divide en intervalos llamados bins, y se cuenta cuántos píxeles caen dentro de cada uno. Esto permite ver de manera clara si una imagen está dominada por tonos oscuros, claros o intermedios en cada canal. A diferencia de la media o la desviación estándar, el histograma no resume el color en un solo número, sino que muestra toda su distribución [24].

2.1.5. Aprendizaje de Máquina

2.1.5.1. Modelo de regresión

La regresión es una técnica de aprendizaje automático cuyo propósito principal es predecir el valor numérico continuo de una variable de interés (variable respuesta) usando información de una o más variables explicativas (variables predictoras). Este método permite al algoritmo aprender la relación estadística entre estas variables a partir de datos de entrenamiento y usar ese conocimiento para hacer predicciones sobre nuevos datos. Estos modelos abordan dos tipos de problemas, primero, realizar predicciones del valor de la variable respuesta cuando se conocen los valores de las variables explicativas, y segundo, analizar cómo los cambios en las variables explicativas afectan la variable de interés. Por ejemplo, predecir el precio de una vivienda basándose en su tamaño y número de habitaciones, o estimar el consumo de combustible de un vehículo según su potencia, estas son aplicaciones típicas de estos modelos [25].

2.1.5.1.1. Support Vector Regression (SVR)

Support Vector Regression es una variante de las Máquinas de Vectores de Soporte (SVM) adaptada para resolver problemas de regresión. En lugar de buscar un hiperplano que separe clases, busca una función que se ajuste a los datos de entrenamiento con una desviación máxima permitida ϵ respecto a los valores reales [26]. Funciona como un modelo de transformación de vista que utiliza regiones locales de interés en una vista específica para predecir la información de movimiento o características correspondientes en una vista diferente, permitiendo así mapear datos de alta dimensión desde ángulos de visión no observados hacia un plano común [27]. El modelo normaliza las características a un ángulo de visión común antes de calcular la similitud mediante métricas como la distancia euclidiana, considerada una técnica clásica de aprendizaje automático que ofrece mayor robustez frente a variaciones de vista que otros modelos tradicionales [27].

2.1.5.1.2. Random Forest Regressor

Al igual que su contraparte de clasificación, construye un conjunto de árboles de decisión de forma independiente, donde cada árbol aprende a dividir los datos de entrenamiento en subgrupos cada vez más homogéneos, optimizando en este caso la minimización del error entre los valores predichos y los reales [26]. La diferencia principal respecto a la clasificación está en la etapa final. Mientras que en clasificación cada árbol "vota" por una clase y gana la mayoría, en regresión cada árbol produce una estimación numérica, y el resultado final es el promedio de todas ellas, lo que hace la predicción más estable y menos sensible a valores atípicos. A esto se suma la aleatoriedad introducida durante la construcción de cada árbol, tanto en la selección de muestras como en las variables consideradas en cada división, garantizando así que los árboles sean diversos entre sí y que el modelo en conjunto reduzca el riesgo de sobreajuste [27].

2.1.5.1.3. K-Nearest Neighbors Regressor

El KNN aplicado a regresión funciona de forma muy similar a su versión de clasificación, con la diferencia de que, en lugar de predecir una categoría, predice un valor numérico continuo. Su funcionamiento consiste en almacenar todos los ejemplos de entrenamiento y, cuando llega una nueva muestra, identificar los k ejemplos más similares a ella dentro del espacio de características, usando para ello la métrica de distancia euclidiana. Una vez identificados esos vecinos, el valor predicho se obtiene calculando el promedio de sus valores objetivo [12]. La calidad de la predicción depende directamente del valor de k, es decir, un k pequeño hace que el modelo sea muy sensible a ejemplos individuales y al ruido, mientras que un k grande suaviza la predicción, pero puede perder detalles importantes del fenómeno que se quiere modelar [27].

2.1.5.1.4. MLP Regressor

El funcionamiento del Perceptrón Multicapa consiste en pasar la información de entrada a través de una serie de capas de neuronas conectadas entre sí, donde cada neurona combina sus entradas con unos pesos aprendidos y aplica una función de activación no lineal como ReLU, lo que le permite capturar relaciones complejas entre las variables. La diferencia principal respecto a su uso en clasificación está en la capa de salida. En regresión no se aplica ninguna función que limite el rango de la predicción, permitiendo que el modelo produzca cualquier valor numérico. Para aprender, el modelo usa el algoritmo de retropropagación, que calcula qué tanto se equivocó en cada predicción y ajusta los pesos de todas las neuronas en consecuencia, repitiendo este proceso iterativamente hasta minimizar el error entre los valores predichos y los reales [27].

2.1.5.2. Modelo de clasificación

La clasificación es una técnica de aprendizaje automático que predice la categoría o clase a la que pertenece una nueva muestra de datos. Según Han et al. [28], este proceso consta de dos etapas. Primero, se construye un modelo usando datos de entrenamiento donde el algoritmo aprende patrones y relaciones entre las características de los datos y sus clases correspondientes. Segundo, el modelo entrenado se utiliza para predecir las etiquetas de clase de datos nuevos no vistos anteriormente, siendo los datos de prueba. Por ejemplo, clasificar correos electrónicos como “spam” o “no spam” es una aplicación común de clasificación [29].

2.1.5.2.1. Support Vector Machine (SVM)

La Máquina de Vectores de Soporte es un modelo de aprendizaje supervisado cuyo objetivo es encontrar la frontera que mejor separa las clases dentro del espacio de características, maximizando la distancia entre dicha frontera y los ejemplos más cercanos de cada clase [12]. Esta frontera se posiciona en el punto medio entre los ejemplos límite de cada clase, llamados vectores de soporte, que son los únicos datos que determinan su posición. Cuando las clases no pueden separarse con una línea o plano simple, la SVM proyecta los datos a un espacio de mayor dimensión donde sí es posible encontrar esa separación. Para manejar casos donde los datos de distintas clases se mezclan o presentan ruido, el modelo permite que algunos ejemplos queden del lado incorrecto de la frontera, pero los penaliza proporcionalmente a qué tan lejos están de donde deberían estar, lo que lo hace robusto frente a valores atípicos [26], [27].

2.1.5.2.2. Random Forest

Random Forest es un modelo de aprendizaje supervisado que combina múltiples árboles de decisión para producir una predicción más robusta y confiable que la de un solo árbol. Cada árbol se construye dividiendo los datos de entrenamiento en subgrupos progresivamente más homogéneos, seleccionando en cada división la pregunta que mejor separa las clases [27]. La clave del modelo está en la aleatoriedad introducida durante su construcción. Cada árbol se entrena con un subconjunto aleatorio de los datos y considera solo un subconjunto aleatorio de características en cada división, lo que garantiza que los árboles sean diversos entre sí y que el modelo en conjunto sea menos sensible al ruido y al sobreajuste que un árbol individual [26].

Durante la predicción, cada árbol emite su propia clasificación y el resultado final corresponde a la clase más "votada" entre todos ellos. El desempeño del modelo depende principalmente de tres hiperparámetros siendo, la profundidad máxima de los árboles, el número total de árboles en el bosque y el número de características consideradas en cada división. Estas características lo convierten en una opción

ampliamente utilizada cuando se requiere un modelo robusto frente a datos ruidosos y con buena capacidad de generalización [26].

2.1.5.2.3. K-Nearest Neighbors (KNN)

KNN es una técnica de aprendizaje supervisado simple y no paramétrica, lo que significa que no asume ninguna forma predefinida para la distribución de los datos ni aprende parámetros durante el entrenamiento. En cambio, almacena todos los ejemplos de entrenamiento en memoria y los usa directamente al momento de clasificar una nueva muestra [12]. Su funcionamiento se basa en la idea de que ejemplos similares tienden a estar cerca unos de otros en el espacio de características. Cuando llega una nueva muestra, el algoritmo calcula su distancia a todos los ejemplos almacenados, usando comúnmente la distancia euclidiana, identifica los k más cercanos y asigna la clase más frecuente entre ellos mediante "votación" mayoritaria.

El desempeño del modelo depende críticamente del valor de k, es decir, un k muy pequeño hace que el modelo sea sensible al ruido y produzca fronteras de decisión irregulares, mientras que un k muy grande suaviza excesivamente esas fronteras y puede ignorar patrones importantes en clases minoritarias. Por esta razón, el valor de k suele ajustarse mediante técnicas como la validación cruzada [27].

2.1.5.2.4. Perceptrón multicapa (MLP)

El Perceptrón Multicapa es un modelo de aprendizaje supervisado que organiza neuronas artificiales en múltiples capas completamente conectadas, por ejemplo, una capa de entrada, una o más capas ocultas y una capa de salida [12]. Su funcionamiento consiste en pasar la información de entrada a través de estas capas, donde cada neurona combina sus entradas con unos pesos aprendidos y aplica una función de activación no lineal como ReLU, lo que le permite capturar relaciones complejas que un modelo lineal no podría resolver. Para aprender, el modelo utiliza el algoritmo de retropropagación, que calcula el error entre la predicción y el valor real, y lo propaga hacia atrás por la red para ajustar los pesos de todas las neuronas iterativamente hasta minimizar ese error. En este caso, el resultado final corresponde a la neurona de la capa de salida con mayor activación, que indica la clase predicha [12].

2.1.6. Aprendizaje Profundo

2.1.6.1. Redes Neuronales Convolucionales (CNN)

Las Redes Neuronales Convolucionales son un tipo de modelo de inteligencia artificial especializado en analizar imágenes. Su funcionamiento se basa en una serie de capas apiladas secuencialmente, donde cada una cumple una función específica dentro del proceso de aprendizaje [29]. Como se ilustra en la Fig. 4, una CNN típica está compuesta por capas de convolución, que aplican filtros para detectar patrones visuales como bordes, colores o texturas, capas de pooling, que reducen el tamaño de la información manteniendo los patrones más relevantes, también una capa de conexión completa, que combina toda la información extraída, y finalmente una capa de salida con una función Softmax, que asigna una probabilidad a cada clase posible para generar la clasificación final [30]. Gracias a que aplican los mismos filtros en diferentes partes de la imagen, son más eficientes, necesitan menos cálculos y logran identificar elementos, aunque cambien ligeramente de posición [29]. Por estas razones, las CNNs se han convertido en la técnica más usada en visión por computador, porque aprenden directamente de los datos sin depender de que una persona defina qué características deben buscar.

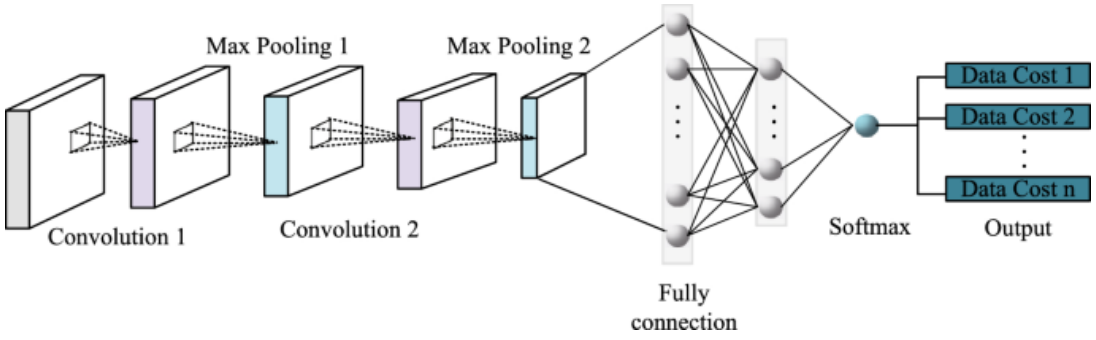


Fig. 4. Estructura general de una Red Neuronal Convolucional (CNN). Obtenido de [30].

2.1.6.1.1. Mobilenet-V3

MobileNetV3 es una arquitectura de red neuronal diseñada para ejecutarse de forma eficiente en dispositivos móviles, buscando el mejor equilibrio posible entre precisión y velocidad de procesamiento. Su base son las convoluciones separables en profundidad, que dividen la convolución tradicional en dos operaciones ligeras. La primera filtra espacialmente cada canal de entrada por separado, y la segunda combina los resultados para generar nuevas características [31]. Esta división reduce el costo computacional hasta nueve veces respecto a una convolución estándar manteniendo una precisión similar. La arquitectura también incorpora un bloque de construcción donde los datos pasan por tres etapas. Primero se amplía su representación para tener más información disponible al momento de aplicar las transformaciones, luego se procesan en esa representación, y finalmente se comprimen de vuelta a un tamaño más compacto. Esto permite que la red aprenda transformaciones más ricas sin que el costo computacional sea excesivo. [32].

Para mejorar la capacidad de la red sin aumentar significativamente su costo, MobileNetV3 incluye módulos de Squeeze-and-Excitation, que funcionan como un mecanismo de atención por canal. Estos módulos primero resumen la información de cada canal en un único valor mediante un promedio global, y luego aprenden qué canales son más relevantes para la tarea y les asignan mayor peso, permitiendo que la red se enfoque en las características más informativas [33].

La Fig. 5 muestra la arquitectura completa de MobileNetV3, donde se puede ver cómo la imagen de entrada va pasando por una serie de bloques encadenados que la transforman progresivamente. En cada bloque, la información se procesa siguiendo la estructura descrita anteriormente, se amplía, se filtra y se comprime. A medida que avanza por la red, las dimensiones espaciales de la imagen se van reduciendo mientras que la profundidad de la representación aumenta, permitiendo que el modelo capture características cada vez más abstractas. Al final, una última capa de procesamiento condensa toda esa información para producir la predicción [34].

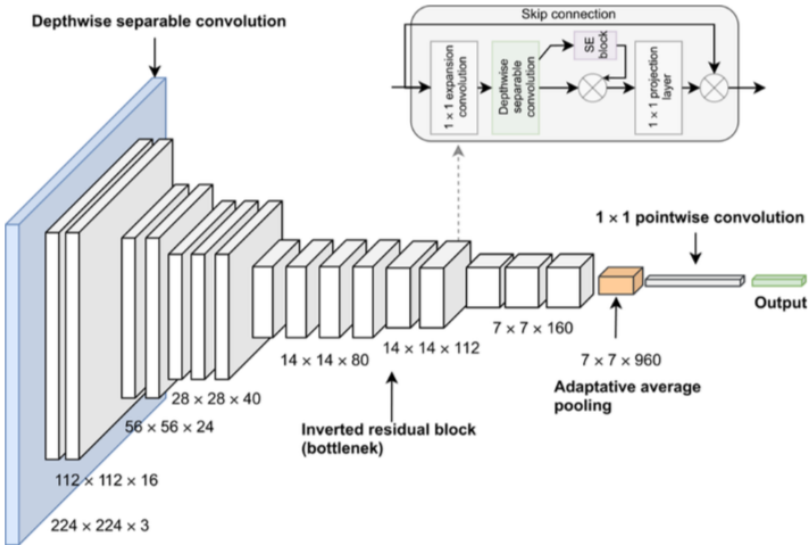


Fig. 5. Arquitectura de MobileNetV3. Obtenido de [34].

2.1.6.1.2. VGG16

VGG16 es una arquitectura de red neuronal profunda desarrollada por el Visual Geometry Group de la Universidad de Oxford, la cual sigue el principio de que aumentar la profundidad de la red es la clave para mejorar la precisión en el reconocimiento de imágenes. A diferencia de arquitecturas anteriores que usaban filtros grandes en las primeras capas, VGG16 utiliza exclusivamente filtros convolucionales pequeños de 3×3 a lo largo de toda la red, lo que le permite alcanzar 16 capas con pesos, 13 convolucionales y 3 completamente conectadas [35]. Usar filtros pequeños en lugar de grandes tiene dos ventajas importantes. Primero, apilar varios filtros de 3×3 logra el mismo efecto que un filtro grande, por ejemplo, tres capas de 3×3 equivalen en alcance a una sola de 7×7, pero introduciendo más funciones de activación intermedias, lo que permite que

la red aprenda patrones más complejos y discriminativos. Segundo, reduce considerablemente el número de parámetros. Tres capas de 3×3 utilizan un 81% menos de pesos que una sola capa de 7×7 con el mismo número de canales, lo que actúa como una forma de regularización que ayuda a evitar el sobreajuste [36].

La Fig. 6 muestra la arquitectura completa de VGG16. Como se puede apreciar, los datos de entrada pasan por cinco bloques convolucionales organizados en secuencia, donde cada bloque aplica varias convoluciones seguidas de una capa de agrupación que reduce progresivamente las dimensiones espaciales de la imagen mientras el número de canales aumenta de 64 hasta 512. Una vez extraídas las características, la información pasa por tres capas completamente conectadas, las dos primeras con 4.096 neuronas cada una, y la última con tantas neuronas como clases tenga el problema, culminando en una función softmax que convierte las activaciones en probabilidades para cada categoría [35].

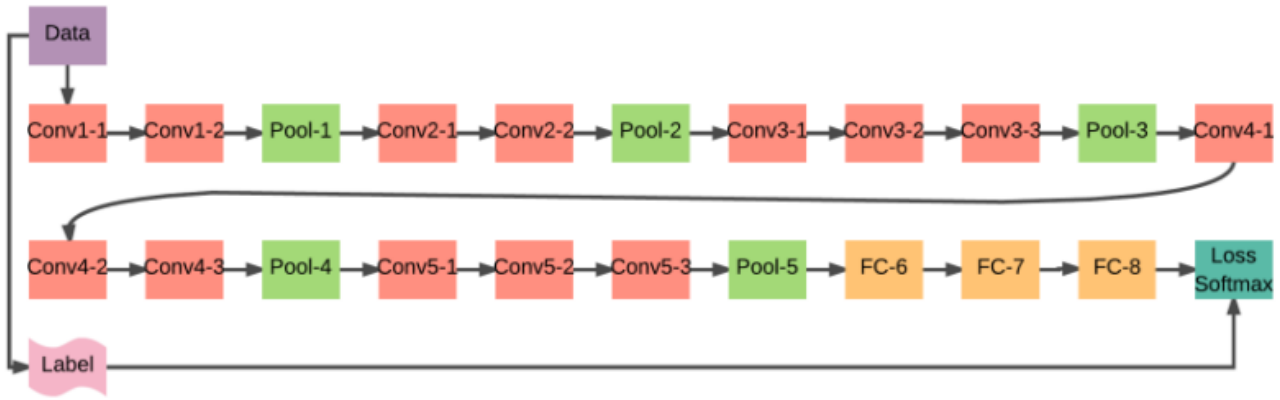


Fig. 6. Arquitectura VGG16. Obtenido de [35].

2.1.6.1.3. EfficientNet-B3

EfficientNet-B3 es una arquitectura de red neuronal convolucional cuya característica principal es la forma en que crece y se hace más poderosa. A diferencia de otros modelos que mejoran su capacidad simplemente añadiendo más capas, más canales o usando imágenes más grandes de forma independiente, EfficientNet propone hacer las tres cosas al mismo tiempo y de forma equilibrada [37]. La razón es que estas tres dimensiones están relacionadas. Si se le da al modelo una imagen más grande para analizar, también necesita más capas para procesar, regiones más amplias y más canales para capturar mayor detalle. Escalarlas juntas de forma proporcional es lo que hace que este modelo sea más eficiente que otros de tamaño similar.

Internamente, el modelo se construye a partir de bloques que combinan tres ideas principales. La primera es dividir la operación de convolución en pasos más simples y ligeros, reduciendo la cantidad de cálculos necesarios sin perder capacidad de aprendizaje. La segunda son las conexiones residuales, que permiten que la información salte directamente de una parte del bloque a otra, facilitando el aprendizaje en redes muy profundas [38]. La tercera son los módulos Squeeze-and-Excitation, que funcionan como un

sistema de atención, es decir, la red analiza todas las características que extrajo y aprende cuáles son más útiles para la tarea, asignándoles automáticamente mayor importancia y reduciendo el peso de las menos relevantes. Finalmente, en lugar de usar la función de activación ReLU tradicional, el modelo utiliza Swish, una función matemática más suave que permite al modelo aprender relaciones más complejas entre los datos con mayor facilidad [37].

La Fig. 7 muestra la arquitectura completa de EfficientNet-B3, donde se puede ver cómo la imagen de entrada va pasando por una larga secuencia de bloques que la transforman progresivamente, reduciendo su tamaño espacial mientras extraen características cada vez más abstractas. Al final, una última etapa de procesamiento condensa toda esa información para producir la predicción final. Con aproximadamente 12 millones de parámetros, EfficientNet-B3 logra una precisión muy competitiva siendo considerablemente más liviano que otros modelos con resultados similares [39].

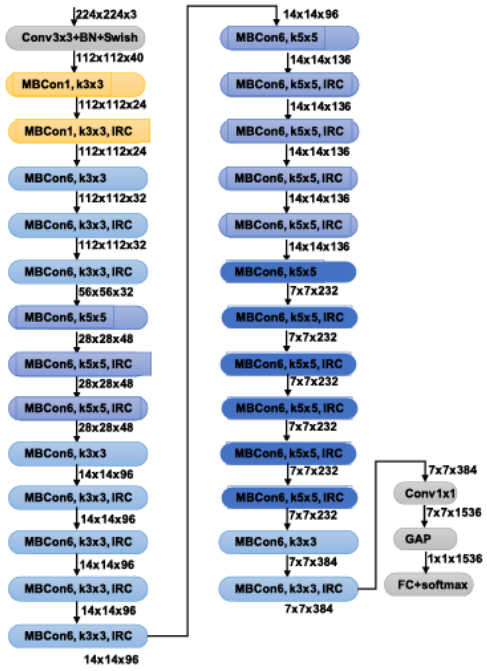


Fig. 7. Arquitectura EfficientNet-B3. Obtenido de [39].

2.1.6.1.4. ResNet-18

ResNet-18 es una arquitectura de red neuronal convolucional compuesta por 18 capas con pesos, cuyo aporte principal es la solución a un problema que aparece al construir redes muy profundas. Cuando se añaden demasiadas capas, en lugar de mejorar, el modelo empieza a rendir peor. Esto ocurre porque a medida que el error se propaga hacia atrás durante el entrenamiento, se va debilitando y las primeras capas dejan de aprender correctamente [40]. Para resolver esto, ResNet introduce los bloques residuales, que son bloques de capas que incluyen una conexión directa entre la entrada y la salida del bloque, como se puede ver en la Fig. 8. Esta

conexión, llamada shortcut o atajo, permite que la información original llegue directamente a la salida sin pasar por todas las transformaciones intermedias, lo que facilita el aprendizaje y evita que el error se pierda en el camino [41]. De esta forma, en lugar de que cada bloque aprenda la transformación completa desde cero, solo necesita aprender la diferencia entre la entrada y la salida deseada, lo cual es una tarea mucho más sencilla.

En cuanto a su estructura, y como se puede apreciar en la Fig. 8, la red comienza con una capa convolucional inicial que reduce el tamaño espacial de la imagen a la mitad, seguida de una capa de agrupación que la reduce nuevamente. Después, la red se organiza en cuatro etapas, cada una compuesta por dos bloques residuales, donde las flechas curvas que se ven en la figura representan precisamente esas conexiones de atajo que saltan cada bloque. A lo largo de estas etapas, el tamaño espacial de la imagen se va reduciendo progresivamente mientras que el número de canales aumenta, pasando de 64 a 128, luego a 256 y finalmente a 512, lo que permite que el modelo capture características cada vez más abstractas y complejas [40]. Adicionalmente, el modelo aplica normalización por lotes después de cada convolución, lo que estabiliza el entrenamiento y hace que el modelo converja más rápido [42]. La arquitectura finaliza con una capa de agrupación global que condensa toda la información extraída, seguida de una capa completamente conectada que produce la clasificación final.

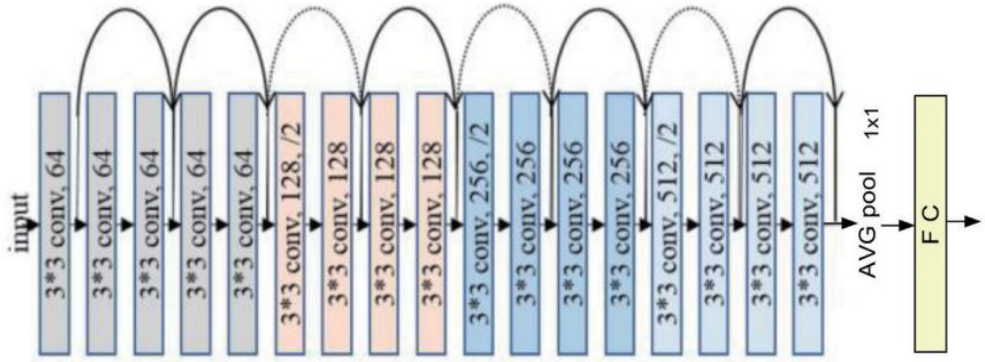


Fig. 8. Arquitectura ResNet-18. Obtenido de [43].

2.1.6.2. Aprendizaje por Transferencia

El aprendizaje por transferencia es una técnica que consiste en reutilizar un modelo que ya fue entrenado para resolver una tarea, y adaptarlo para resolver una tarea diferente pero relacionada. La idea como tal es que el conocimiento adquirido durante el primer entrenamiento, como por ejemplo la capacidad de reconocer bordes, formas o texturas en imágenes, no se descarta, sino que sirve como punto de partida para aprender la nueva tarea. Esto es especialmente útil cuando el conjunto de datos disponible para el nuevo

problema es pequeño, ya que en lugar de aprender todo desde cero, el modelo solo necesita ajustar sus parámetros internos para adaptarse a la nueva tarea [44].

2.1.6.3. Arquitecturas Preentrenadas

Las arquitecturas preentrenadas son modelos de redes neuronales que ya fueron entrenados con grandes conjuntos de imágenes, siendo el más común ImageNet. Gracias a ese entrenamiento previo, estos modelos tienen la capacidad de identificar patrones visuales en imágenes, lo que los convierte en una base sólida para resolver nuevos problemas de clasificación mediante aprendizaje por transferencia [45]. Entre las arquitecturas más comúnmente usadas se encuentran ResNet, AlexNet y MobileNet, cada una con un diseño distinto orientado a lograr buenos resultados en diferentes condiciones, ya sea priorizando la precisión, la eficiencia computacional o la capacidad de funcionar en dispositivos con recursos limitados [46].

2.1.6.4. Fine-Tuning

El Fine-Tuning es una técnica que consiste en tomar un modelo que ya fue entrenado para una tarea y adaptarlo para que funcione bien en una tarea diferente, sin necesidad de entrenarlo desde cero. Su base es el aprendizaje por transferencia, que parte de la idea de que el conocimiento que un modelo adquirió resolviendo un problema puede ser útil para resolver otro problema relacionado [47]. Esto es posible porque en una red neuronal convolucional las primeras capas aprenden características muy generales, como bordes, texturas y formas básicas, que son útiles en casi cualquier tarea visual. Las capas más profundas, en cambio, aprenden características más específicas de la tarea para la que fue entrenado originalmente el modelo [47]. Para aprovechar esto, el Fine-Tuning permite elegir qué partes del modelo se mantienen fijas y cuáles se vuelven a entrenar. Las capas iniciales generalmente se congelan, es decir, sus pesos no se modifican durante el nuevo entrenamiento, conservando así el conocimiento general que ya adquirieron. El ajuste se concentra entonces en las capas finales, que son las que deben especializarse para la nueva tarea [47].

2.1.7. Métricas de Evaluación

Una métrica de evaluación es una función que cuantifica el desempeño de un modelo al comparar sus predicciones con los valores reales de referencia, considerando un conjunto de datos específico y los supuestos establecidos [48].

2.1.7.1. Métricas para Regresión

2.1.7.1.1. MAE – Error medio absoluto

El MAE calcula el promedio de las diferencias absolutas entre el valor predicho y el valor real para cada muestra. En el contexto de predicción de vida útil, representa directamente cuántos días se equivoca el modelo en promedio. Al usar el valor absoluto en lugar del cuadrado del error, todas las diferencias tienen el mismo peso sin importar su tamaño, lo que lo hace una métrica robusta frente a valores atípicos [49]. Su cálculo se expresa como:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|P_i - O_i|$$

Donde n es el número total de muestras, P_i es el valor real y O_i es el valor predicho por el modelo [49].

2.1.7.1.2. SMAPE – Error porcentual absoluto medio simétrico

El SMAPE es una métrica que mide el error del modelo expresándolo como un porcentaje. Su principal característica es que penaliza de la misma manera cuando el modelo predice un valor más alto que el real y cuando predice un valor más bajo, convirtiéndolo en una medida más equilibrada del error. Su resultado se encuentra en un rango de 0% a 200%, donde un valor cercano a 0% indica que las predicciones del modelo son muy cercanas a los valores reales [50].

$$SMAPE = \frac{100\%}{n}\sum_{i=1}^{n}\frac{2|y_i - \hat{y}_i|}{|y_i| + |\hat{y}_i|}$$

Donde n es el número total de muestras, y_i es el valor real y $\hat{y}_i$ es el valor predicho por el modelo [50].

2.1.7.1.3. RMSE – Raíz del error cuadrático medio

El RMSE es la raíz de la media de los errores al cuadrado. Al elevar los errores al cuadrado, penaliza con mayor peso las predicciones que se alejan significativamente de los valores reales, por lo que resulta útil para evidenciar el riesgo de cometer errores graves [51]. Cuando el costo de un error grande en la predicción es desproporcionado, el RMSE complementa al MAE al resaltar estos casos críticos. Si RMSE > MAE, esto indica la presencia de valores atípicos o subgrupos problemáticos en el conjunto de datos que ameritan un diagnóstico más detallado [48]. Su cálculo se expresa como:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Donde n es el número total de muestras, y_i es el valor real y $\hat{y}_i$ es el valor predicho por el modelo [50].

2.1.7.1.4. R^2 y R^2 ajustado – Coeficiente de determinación

El R^2 mide qué proporción de la variación total en los datos logra explicar el modelo. Para calcularlo, compara los errores del modelo con los errores que se obtendrían si simplemente se predijera el valor promedio para todos los casos. Un valor de R^2 cercano a 1 indica que el modelo explica bien los datos, mientras que un valor cercano a 0 indica que el modelo no captura los patrones presentes [52]. Su cálculo se expresa como:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

Donde y_i es el valor real, $\hat{y}_i$ es el valor predicho y $\bar{y}_i$ es el promedio de todos los valores reales [52]. Por otro lado, el R^2 ajustado incorpora una penalización según el número de variables del modelo y el tamaño de la muestra, lo que lo hace más adecuado para comparar modelos con distinta complejidad, ya que el R^2 estándar siempre aumenta cuando se agregan más variables, incluso si no aportan valor real al modelo [52]. Su cálculo se expresa como:

$$R^2{}_{adj} = 1 - \frac{(1 - R^2)(n - 1)}{n - v - 1}$$

Donde n es el número total de muestras y v es el número de variables predictoras del modelo [50].

2.1.7.2. Métricas para Clasificación

2.1.7.2.1. Matriz de Confusión

La matriz de confusión es una tabla que cruza las clases reales con las clases predichas por el modelo, permitiendo visualizar tanto los aciertos como los diferentes tipos de errores cometidos en cada categoría [53].

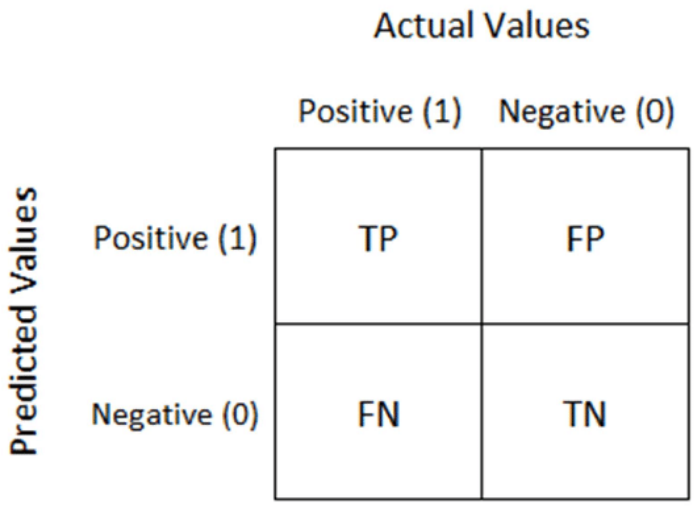


Fig. 9. Estructura de la matriz de confusión con sus cuatro cuadrantes: verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN). Obtenido de [54].

2.1.7.2.2. Precisión y Cobertura "Recall" por clase

La precisión evalúa qué tan confiables son las predicciones del modelo para una clase determinada. Específicamente, mide qué proporción de las instancias que el modelo clasificó como pertenecientes a una clase realmente lo eran. Un valor alto de precisión indica que cuando el modelo dice que algo pertenece a una clase, raramente se equivoca [55]. Su cálculo se expresa como:

$$Precision = \frac{TP}{TP + FP}$$

Donde TP son los verdaderos positivos, es decir, las instancias correctamente clasificadas en esa clase, y FP son los falsos positivos, es decir, las instancias que el modelo clasificó en esa clase pero que realmente pertenecían a otra [55]. Por otro lado, el Recall evalúa qué tan completo es el modelo a la hora de detectar todas las instancias de una clase. Mide qué proporción de las instancias que realmente pertenecían a una clase fueron efectivamente identificadas por el modelo. Un valor alto de Recall indica que el modelo logra encontrar la mayoría de los casos reales de esa clase sin dejar muchos sin detectar [55]. Su cálculo se expresa como:

$$Recall = \frac{TP}{TP + FN}$$

Donde TP son los verdaderos positivos y FN son los falsos negativos, es decir, las instancias que realmente pertenecían a esa clase pero que el modelo clasificó incorrectamente en otra [55].

2.1.7.2.3. F1-score

El F1-score combina la precisión y el Recall en una sola métrica calculando su media armónica. A diferencia de un promedio simple, la media armónica penaliza los casos en que una de las dos métricas es muy baja, lo que obliga a que el modelo tenga un buen desempeño en ambas para obtener un F1 alto [55]. Su cálculo se expresa como:

$$F1 = 2 * \frac{(Precision * Recall)}{Precision + Recall}$$

Cuando el problema involucra más de dos clases, se calcula el F1 de forma independiente para cada clase y luego se combinan mediante un método de promedio. Los dos métodos más utilizados son el macro y el weighted [56].

El F1 macro calcula el F1 de cada clase por separado y luego promedia los resultados asignando el mismo peso a todas las clases, independientemente de cuántas muestras tenga cada una. Esto garantiza que el modelo funcione bien en todas las categorías, incluso en aquellas con menos muestras [56]. Su cálculo se expresa como:

$$F1_{macro} = \frac{1}{C}\sum_{c=1}^{C} F1_c$$

Donde C es el número total de clases y $F1_c$ es el F1-score de la clase c [56]. Luego, está el F1 weighted, que también calcula el F1 de cada clase por separado, pero a diferencia del macro, no trata todas las clases como igualmente importantes. Aquí las clases con más muestras en el conjunto de datos tienen mayor influencia en el resultado final que las clases con pocas muestras. Esta versión refleja mejor el comportamiento del modelo en condiciones reales donde algunas clases son más frecuentes que otras [56]. Su cálculo se expresa como:

$$F1_{weighted} = \sum_{i=1}^{N} w_i * F1_c$$

Donde N es el número total de clases, $F1_c$ es el F1-score de cada clase i, y w_i es el peso asignado a cada clase, calculado como:

$$w_i = \frac{TP_i + FN_i}{\sum_{j=1}^{N}(TP_j + FN_j)}$$

Donde TP_i son los verdaderos positivos y FN_i son los falsos negativos de la clase i, y el denominador representa la suma de todas las instancias reales de todas las clases [56].

2.1.7.2.4. Accuracy (Global y por clase)

El Accuracy es la proporción total de predicciones correctas sobre el conjunto de evaluación, es decir, cuántas instancias clasificó correctamente el modelo del total evaluado. Esta métrica es intuitiva y útil como resumen general del desempeño del modelo. Aunque el conjunto de datos esté balanceado, es importante complementar el accuracy global con el accuracy calculado individualmente por cada clase. Esto permite identificar si el modelo tiene dificultades específicas con alguna etapa en particular. Por ejemplo, el modelo podría tener un buen accuracy global, pero presentar problemas específicos al distinguir entre clases similares [57]. Su cálculo se expresa como:

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

Donde TP son los verdaderos positivos, FP los falsos positivos, FN los falsos negativos, y TN son los verdaderos negativos, es decir, las instancias que el modelo clasificó correctamente como no pertenecientes a una clase [57].

2.2. Trabajos Relacionados

La aplicación de técnicas de visión por computador en la agricultura ha tomado relevancia por su capacidad de reducir pérdidas y optimizar la distribución de productos frescos. En particular, la predicción de la vida útil de frutas mediante métodos no invasivos ha sido abordada en diversas investigaciones recientes. Darwish [58] propuso un modelo basado en redes neuronales convolucionales para la clasificación de frutas a partir de imágenes digitales, demostrando la capacidad de estos modelos para superar las limitaciones de los métodos tradicionales de inspección visual.

Por otra parte, Nandan G [59] propuso un proceso basado en visión por computador para extraer características de color en frutas y construir modelos de predicción de vida útil. Su investigación, aplicada en bananos como prueba de concepto, demostró que a partir de características actuales es posible estimar características futuras, permitiendo decisiones de distribución más precisas, reducción de desperdicios y eliminación de la subjetividad en los sistemas tradicionales de clasificación.

Finalmente, Shahzaib [60] desarrolló *RipeTrack*, usando un sistema móvil que combina espectroscopía y aprendizaje profundo con cámaras de smartphones para estimar tanto el estado de madurez como el tiempo de vida restante de aguacates y peras. Con precisiones superiores al 90%, el sistema probó ser aplicable en entornos reales como supermercados y hogares, destacando la viabilidad de soluciones no invasivas y de bajo costo. Estos trabajos sientan las bases para el presente proyecto, que busca aprovechar cámaras convencionales y modelos de aprendizaje profundo para clasificar aguacates Hass según su estado

de madurez y estimar su vida útil, contribuyendo a disminuir las pérdidas económicas y el desperdicio alimentario en el contexto local.

Estos antecedentes muestran que las técnicas de visión por computador y aprendizaje profundo han sido aplicadas con éxito en la clasificación de frutas y en la estimación de su vida útil, aportando soluciones objetivas y más confiables que los métodos visuales tradicionales. El trabajo de Darwish [58] evidenció la capacidad de las CNN para clasificar frutas a partir de imágenes, lo cual aporta una base sólida para la tarea de reconocimiento del estado de madurez de los aguacates Hass. La investigación de Nandan G [59] demostró que es posible predecir la vida útil de los frutos a partir de sus características visuales, lo que valida la idea de que el estado actual puede ser usado como predictor de su evolución en el tiempo. Finalmente, el desarrollo de RipeTrack por Shahzaib [60] confirma la viabilidad de sistemas prácticos, incluso con hardware accesible como cámaras de smartphones, integrando predicción de madurez y vida útil en un mismo flujo. Estos antecedentes aportan referentes técnicos y metodológicos que guían el desarrollo de este proyecto. Sin embargo, aún existe un vacío específico en soluciones orientadas al aguacate Hass en el contexto colombiano, considerando condiciones reales de supermercados de Cali, donde el desperdicio es un problema económico y logístico relevante.

3. DESARROLLO DEL PROYECTO

3.1. Preparación y preprocesamiento del conjunto de Datos

A continuación, se describe el proceso llevado a cabo para la identificación del conjunto de datos y el preprocesamiento de las imágenes, con el fin de garantizar una entrada de datos consistentes y adecuada para el entrenamiento de los modelos de clasificación y regresión.

3.1.1. Identificación y selección del conjunto de datos

Con el propósito de implementar técnicas de aprendizaje de maquina y desarrollar una metodología que permita la clasificación del estado de madurez del aguacate Hass y la predicción de su tiempo de vida útil en días, fue necesario identificar un conjunto de datos especifico y adecuado a lo que se quería hacer. Este conjunto debía de contener imágenes de aguacates en diferentes etapas de maduración, capturadas de forma que permitiera extraer características visuales relevantes como el color, la textura, y la apariencia superficial. Asimismo, la idea es que cada imagen estuviese anotada con su respectivo estado de madurez y con información que permitiera estimar el tiempo de deterioro del fruto, lo cual termino siendo un reto por las especificaciones que se necesitaban.

Para identificar el conjunto de datos más apropiado se realizó una búsqueda en repositorios públicos de datos académicos, como Kaggle y Mendeley Data. La búsqueda se orientó específicamente a conjunto de datos de aguacates Hass, ya que la mayoría de los trabajos previos relacionados con clasificación de frutas usaban dataset genéricos, donde unos no incluían información sobre la vida útil, y otros no estaban diseñados específicamente para este fruto en particular. Luego de revisar las opciones disponibles, que en ese momento eran pocas, se encontró y se seleccionó el conjunto de datos *"'Hass' Avocado Ripening Photographic Dataset*" [61], disponible en Mendeley Data. Este fue construido con un total de 478 aguacates Hass adquiridos tres días después de su cosecha, documentando su proceso de maduración mediante fotografías diarias tomadas desde dos lados opuestos de cada fruto. El resultado fue un conjunto de 14.710 fotografías etiquetadas, cada una de 800x800 pixeles en formato .jpg. Además, los aguacates estaban organizados en tres grupos según sus condiciones de almacenamiento. El primer grupo, denominado T10, correspondía a aguacates almacenados a 10 °C con una humedad relativa del 85%. El segundo grupo, T20, se almacenó a 20 °C con la misma humedad relativa. Finalmente, el grupo Tamb agrupó los aguacates conservados bajo condiciones ambientales, es decir, sin control de temperatura ni humedad.

Cada fotografía fue asociada a un índice de madurez de cinco etapas: 1. Verde, 2. Pre–Maduro, 3. Maduro Fase 1, 4. Maduro Fase 2, y 5. Podrido. La cuarta etapa marca el fin de la vida útil del aguacate, lo cual significa que cualquier muestra clasificada en la quinta etapa se encuentra en un estado no consumible. El dataset incluye además una hoja de cálculo con el nombre de cada fotografía, el número de muestra al que pertenece, el grupo de almacenamiento, la clasificación según su madurez y el día del experimento en que fue tomado. Esta información permitió calcular para cada imagen, cuántos días le restaban al aguacate alcanzar el estado Podrido, dato que se usó como variable objetivo para el modelo de regresión.

3.1.2. Análisis y depuración del conjunto de datos

Una vez identificado el conjunto de datos, se procedió a seleccionar las muestras correspondientes al grupo "Tamb", ya que estas representan mejor escenario real de los supermercados y distribuidores en el contexto colombiano, donde los frutos normalmente se almacenan y exhiben sin refrigeración ni control de humedad. Asimismo, las imágenes del dataset presentan una calidad uniforme y consistente, por el hecho de que cada fotografía muestra un aguacate completo, centrado, sobre un fondo blanco liso y con una buena iluminación, como se puede apreciar en la Fig. 10, las cuales facilitaron tanto la extracción de características como el entrenamiento de los modelos.

Fig. 10. Imagen representativa del conjunto de datos de la clase Verde. Obtenido de [61].

Tras la selección del grupo Tamb, el conjunto de datos quedó conformado por 2.934 imágenes, distribuidas entre las cinco clases, 504 imágenes de la etapa Verde, 694 de Premaduro, 470 de Maduro Fase 1, 654 de Maduro Fase 2, y 712 de Podrido. También, una decisión importante se tomó en esta etapa, y fue la reducción de cinco a cuatro clases de madurez. Tras analizar y observar las etapas Maduro Fase 1 y Maduro Fase 2, se determinó que ambas corresponden a aguacates en condiciones de consumo y con características

parecidas. Al fin y al cabo, separar estas dos etapas no aportaría mucha información útil para el usuario final, y hasta de pronto podría dificultar el aprendizaje del modelo por distinguir categorías visualmente muy similares, así que ambas clases fueron unificadas en una clase denominada como Maduro y quedaron con 1.124 imágenes. En la Fig. 11 se muestra un ejemplo representativo de cada una de estas cuatro clases, permitiendo apreciar los cambios de color y su apariencia a lo largo del proceso de maduración.

Fig. 11. Ejemplos representativos de las cuatro clases de madurez de una muestra seleccionada del conjunto de datos. Obtenido de [61].

3.1.3. Segmentación de los aguacates para los modelos clásicos de aprendizaje de máquina

A diferencia de los modelos de aprendizaje profundo, que reciben las imágenes directamente como entrada, los modelos clásicos de aprendizaje de máquina requieren que la información de cada imagen sea representada como un vector de valores numéricos. Para garantizar que las características extraídas correspondieran únicamente al aguacate y que no incluyera el fondo blanco de la imagen, se realizó una etapa de segmentación previa a la extracción de características, la cual se aplicó solamente a las imágenes destinadas a los modelos clásicos.

Para la segmentación se usó la librería rembg [16], una herramienta de eliminación de fondo basada en el modelo $U^2 - Net$, siendo una arquitectura de red neuronal diseñada para detectar el objeto principal dentro de la imagen y separarlo del fondo mediante una máscara binaria. Se eligió más que todo por su buena precisión y en que puede operar sin un costo computacional excesivo. En la Fig. 12 se puede observar una muestra de aguacate del conjunto de datos sin haber aplicado la segmentación, mientras que la Fig. 13 es el

resultado de haber aplicado la segmentación sobre la misma imagen, donde el fondo blanco es reemplazado por uno negro, conservando únicamente la región del aguacate para la extracción posterior de características.

Fig. 12. Muestra de aguacates de la clase Verde sin Segmentación. Obtenido de [61].

Fig. 13. Muestra de aguacates de la clase Verde con Segmentación. Obtenido de [61].

3.1.4. Preprocesamiento de imágenes para los modelos de aprendizaje profundo

Para los modelos de aprendizaje profundo, las imágenes originales del dataset fueron utilizadas directamente como entrada, sin aplicar la segmentación de fondo descrita anteriormente. No obstante, fue necesario aplicar transformaciones de preprocesamiento para garantizar que las imágenes fueran compatibles con los requerimientos de cada arquitectura preentrenada.

Redimensionamiento: dado que cada arquitectura fue diseñada para recibir imágenes de un tamaño especifico, las imágenes debieron ser escaladas antes de ser ingresadas al modelo. Para ResNet-18, VGG16 y MobileNetV3, las imágenes fueron redimensionadas a 256x256 píxeles y luego recortadas al centro a 224x224 píxeles [62]-[64]. Para EfficientNet-B3, fueron redimensionadas a 320x320 píxeles y recortadas al centro a 300x300 píxeles [65]. Este recorte central busca eliminar los bordes menos informativos de la imagen y enfocarse en la región central donde normalmente se encuentra el objeto [13].

Normalización: los valores de los píxeles, originalmente en el rango [0,255], fueron primeramente escalados al rango [0.0,1.0] y luego se normalizaron usando los valores de media = [0.485, 0.456, 0.406] y desviación estándar = [0.229, 0.224, 0.225], correspondientes al conjunto ImageNet con el que ambas arquitecturas fueron preentrenadas [62]-[65]. Aplicar esto garantiza que las imágenes tengan una distribución de valores compatible con el conocimiento aprendido por los modelos [14].

Aumento de Datos: para reducir el riesgo de sobreajuste, se generaron versiones transformadas de cada imagen únicamente durante el entrenamiento. Las transformaciones aplicadas fueron, desplazamientos aleatorios de hasta 20% en ambos ejes, simulando que el aguacate no siempre aparezca perfectamente centrado en la fotografía; rotaciones aleatorias de hasta 30° en cualquier dirección; variación de brillo, contraste y saturación de hasta 20%, esto para simular más que todo diferencias en las condiciones de iluminación al momento de capturar la imagen; e implementar un desenfoque gaussiano con una probabilidad del 20%, para imitar imágenes con menor nitidez. Cabe recalcar que el aumento de datos se aplicó mediante la técnica conocida como on-the-fly, que consiste en generar y aplicar transformaciones en tiempo real durante el entrenamiento, sin crear ni almacenar imágenes adicionales en disco. Esto significa que el tamaño del dataset original se mantiene igual antes y después del aumento, y lo que varía es la diversidad de versiones que el modelo observa de cada imagen a lo largo de las épocas de entrenamiento.

3.1.5. Extracción de características para los modelos clásicos de aprendizaje de máquina

A diferencia de los modelos de aprendizaje profundo, que aprenden directamente a partir de las imágenes del conjunto de datos, los modelos clásicos de aprendizaje de máquina como SVM, Random Forest, KNN y MLP no puede procesar imágenes en bruto. Por eso fue necesario realizar una extracción de características, es decir, transformar cada imagen en un conjunto de valores numéricos que describan sus propiedades visuales más importantes a tener en cuenta. Como en este caso se usó las imágenes segmentadas, únicamente los pixeles del aguacate fueron considerados en este cálculo, excluyendo el fondo negro.

Como punto de partida, se construyó un primer vector de 31 características por imagen, organizadas en tres grupos. En primer lugar, las imágenes fueron convertidas a los espacios de color HSV, RGB y CIELab. A partir del espacio HSV, se extrajo un histograma del canal de tonalidad H dividido en 10 intervalos de 9 bins. Luego, se calcularon estadísticas de color, correspondientes a la media y la desviación estándar de cada canal en los tres espacios de color. Finalmente, se obtuvieron características de textura mediante la Matriz de Co-ocurrencia de Niveles de Gris (GLCM), aplicada tanto sobre las imágenes en escala de grises como sobre cada espacio de color, extrayendo métricas como ASM, contraste, correlación, homogeneidad y entropía.

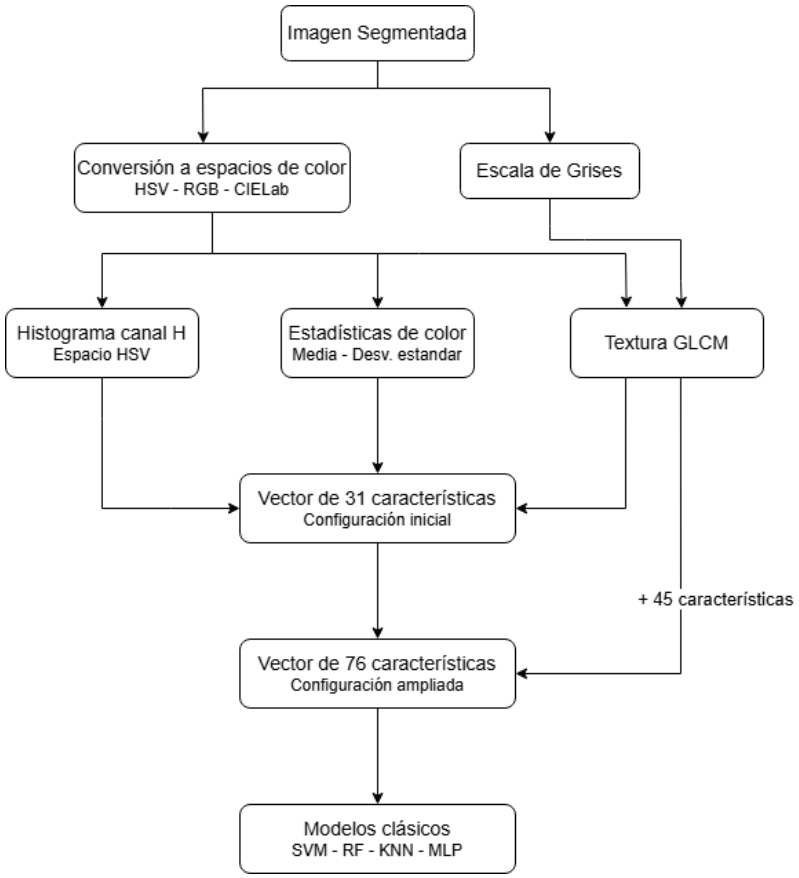


Fig. 14. Flujo del proceso de extracción de características.

La Fig. 14 ilustra el pipeline de extracción de características implementado para los modelos clásicos de aprendizaje automático. Como se puede observar, el proceso parte de la imagen segmentada del aguacate, la cual es convertida a los espacios de color HSV, RGB y CIELab, y adicionalmente a escala de grises. A partir de estas representaciones se extraen en paralelo tres grupos de características, el histograma del canal H del espacio HSV, las estadísticas de color calculadas como la media y desviación estándar por canal, y las características de textura obtenidas mediante la matriz GLCM. Estos tres grupos se concatenan para conformar el vector inicial de 31 características, el cual fue posteriormente ampliado a 76 características en una segunda configuración, donde cada característica de textura GLCM fue aplicada a cada canal de cada espacio de color. Ambos vectores fueron usados como entrada a los modelos clásicos.

Entonces, tras entrenar y evaluar los modelos clásicos con el vector inicial, los resultados obtenidos no fueron suficientemente satisfactorios, lo cual conllevó a optar por una ampliación del conjunto de características. Se identificó que la textura calculada únicamente en escala de grises podría estar perdiendo información relevante en los distintos canales de color, por el hecho de que la maduración del aguacate Hass involucra cambios tanto de color como de textura superficial que posiblemente podrían manifestarse de forma distinta en cada espacio de color. Así que, como se dijo anteriormente, se decidió ampliar el cálculo del GLCM aplicándolo también sobre cada canal individual de los espacios RGB, HSV y CIELab, incorporando así 45 características adicionales.

Este proceso dio como resultado un vector ampliado de 76 características por imagen. En la Tabla 1 se presenta un resumen de cómo se distribuyen estas características entre los 3 grupos que las componen.

Tabla 1. Resumen de las 76 características extraídas por imagen para los modelos clásicos de aprendizaje de máquina.

Grupo de características	Espacio / Canal	Características	Total
Estadísticas de Color	RGB, HSV, CIELab	Media y desviación estándar por canal	16
Textura GLCM	Gris, RGB, HSV, CIELab	ASM, contraste, correlación, homogeneidad, entropía	50
Histogramas de Color	HSV – Canal H	Distribución por bins	10
Total Características			76

El primer grupo corresponde a las características estadísticas de color. En el espacio RGB se calcularon la media y la desviación estándar de cada canal, más dos razones entre canales siendo el canal G dividido entre el canal R y el canal G dividido entre el canal B, las cuales capturan el balance del verde frente a los demás canales ante variaciones de iluminación entre imágenes. En el espacio HSV se calculó la media de los tres canales H, S, V, pero la desviación estándar únicamente del canal V, dado que es el que mejor

refleja la variabilidad del brillo superficial del fruto durante la maduración. En el espacio CIELab se calcularon la media y la desviación estándar de los canales L* y a*, capturando así la luminosidad y el comportamiento del eje Verde-Rojo, asociado al cambio de color del aguacate durante su maduración.

El segundo grupo corresponde a las características de textura, calculadas mediante el GLCM [22]-[23]. A diferencia del vector inicial, el cálculo se hizo no solo sobre la imagen en escala de grises, sino que también sobre cada canal de los espacios RGB, HSV y CIELab como se dijo anteriormente.

El tercer grupo corresponde a los histogramas de color, construidos sobre el canal de tonalidad H del espacio HSV, esto con el fin de analizar las variaciones cromáticas asociadas a los diferentes estados de maduración del aguacate Hass. La elección de este canal se fundamentó en que el Hue permite representar la tonalidad predominante de la imagen, lo cual resulta útil para identificar cambios de color durante el proceso de maduración del fruto [24]. Para abordar lo anterior, a cada imagen se construyó un histograma del canal Hue, que permite representar con qué frecuencia aparecen los distintos rangos de tonalidad en la región del aguacate. Este histograma fue normalizado para que los valores obtenidos no dependieran del tamaño de la imagen ni de la cantidad de píxeles analizados, sino que expresaran una distribución relativa de los colores presentes.

Con el fin de determinar la configuración más adecuada, se realizaron pruebas con histogramas de 20, 30 y 45 bins, tal como se muestra en la Fig. 15, evaluando cómo variaba la distribución de los valores de Hue entre las distintas clases de madurez. A partir de estas pruebas se observó que el histograma de 20 bins ofrecía la representación más adecuada para los propósitos del proyecto. Esta configuración logró conservar la información más relevante de la distribución del color sin generar una cantidad excesiva de características. Al mismo tiempo, esta configuración permitió distinguir diferencias entre las clases de madurez, ya que las variaciones en la distribución normalizada del Hue seguían siendo perceptibles en los rangos más representativos, como se puede apreciar en la Fig. 16.

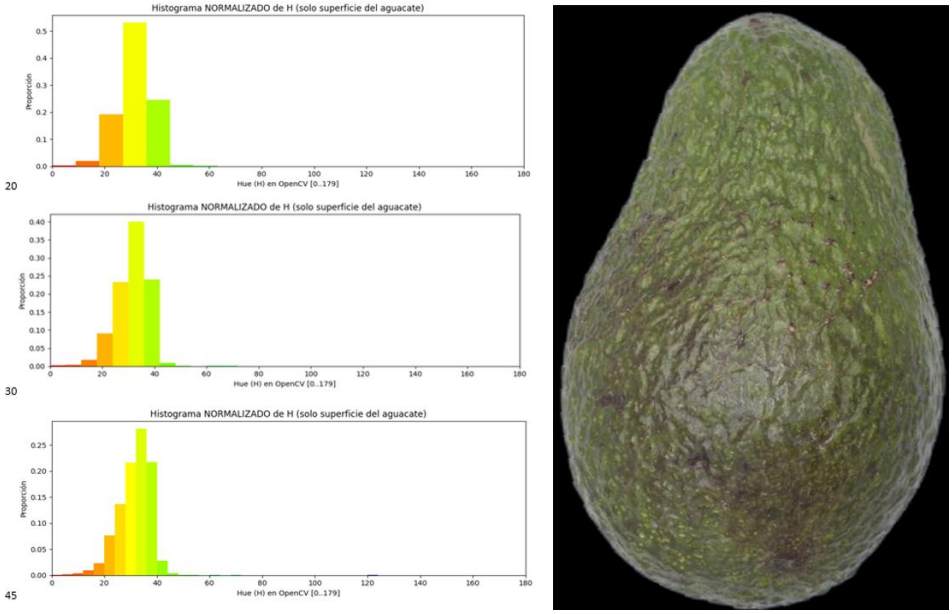


Fig.15 . Muestra de aguacate de la clase premaduro segmentado, junto a sus histogramas normalizados del canal Hue, con diferentes cantidades de bins.

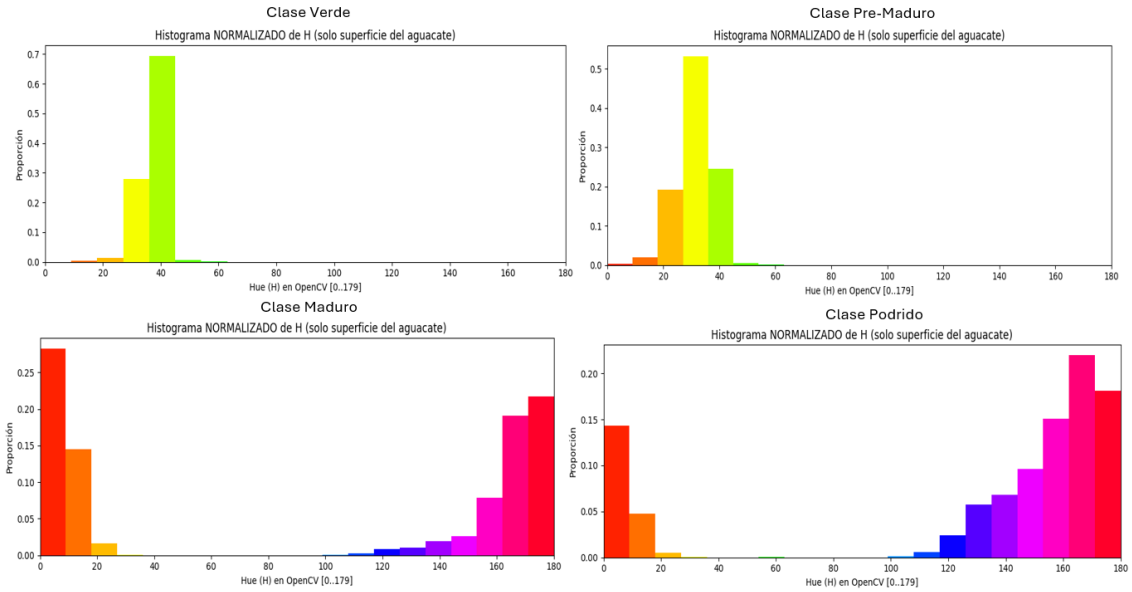


Fig.16 . Ejemplos de los histogramas normalizados sobre el canal Hue por cada clase.

Los 20 bins resultaron suficientes para representar los cambios de color asociados a cada etapa de madurez del aguacate Hass. A partir del análisis de los histogramas normalizados obtenidos, se identificaron los rangos del canal Hue con mayor capacidad para diferenciar las clases, siendo estos los intervalos entre 0° y 54°, y entre 144° y 180°, que corresponden a 10 de los 20 bins. Dado que estos rangos concentraban la información más relevante para distinguir los estados de madurez, se decidió utilizar únicamente estos 10 bins como características finales en lugar del histograma completo.

3.2. Diseño e implementación de modelos de clasificación y regresión

A continuación, se describe el proceso de diseño, configuración y entrenamiento de los modelos implementados en el proyecto. Para abordar los dos problemas planteados, siendo la clasificación del estado de madurez y la predicción en días de vida útil del fruto, se adoptaron dos enfoques. Primero, se probaron 4 modelos clásicos de aprendizaje de máquina, SVM, Random Forest, KNN y MLP, los cuales operan sobre el vector de características extraídas descrito en la sección 3.1.5. Segundo, se implementaron 4 arquitecturas de aprendizaje profundo basadas en redes neuronales convolucionales preentrenadas, siendo ResNet-18, EfficientNet-B3, VGG16 y MobileNetV3, las cuales procesan las imágenes directamente como entrada. En ambos enfoques se llevaron a cabo versiones adaptadas tanto para clasificación como para regresión, lo que resultó en un total de 16 configuraciones de modelos evaluadas a lo largo del proyecto. Cabe recalcar que los recursos usados se pueden observar en una carpeta compartida de Google Drive que se encuentra en la sección 6 de los anexos.

3.2.1. Estrategia general de la implementación

La selección de los modelos y la configuración general del proceso de entrenamiento no fue arbitraria, sino que se apoyó en una revisión exhaustiva de trabajos previos con problemas similares. En cuanto a los modelos clásicos, estudios de clasificación y predicción de vida útil de frutas han reportado buenos resultados con SVM, por ejemplo, se pudo alcanzar un accuracy del 82.55% en clasificación de madurez de aguacate [66] y un R^2 de 0.818 en predicción de vida útil [67], o el Random forest que obtuvo un R^2 de 0.860 en la misma tarea de regresión [67] y buenos resultados en clasificación de cultivos a partir de características GLCM. También, se encontró buenos resultados usando tanto KNN [68]-[69] como MLP [68], lo cual motivó a escogerlos para el proyecto. En cuanto a los modelos de aprendizaje profundo, trabajos específicos sobre aguacates Hass han usado con éxito ResNet-18, con un accuracy del 88% en clasificación de cinco etapas de madurez [70]. También está el modelo VGG16, que consiguió resultados cercanos al 90% de clasificación de madurez [71], y asimismo otros modelos como EfficientNet-B3 [72] y MobileNetV3 [72] reportaron precisiones superiores al 90%, consolidando así la decisión de escoger estas arquitecturas para realizar las pruebas.

La división de los datos también se fundamentó en los artículos revisados. Trabajos como el de Xavier et al. [70] y el de Nuanmeesri [71] utilizaron una partición de 70% para entrenamiento, 15% para validación y 15% para prueba sobre datasets de aguacates Hass. Esta misma división fue aplicada al proyecto. El 70% de las imágenes se destinó al entrenamiento, que es donde el modelo aprende los patrones. El 15% se usó como conjunto de validación, que permite revisar cómo va evolucionando el modelo durante el entrenamiento y detectar si hay sobreajuste. El otro 15% se reservó como conjunto de prueba, que es el que determina el

desempeño final del modelo sobre imágenes que nunca vio durante su desarrollo, garantizando así una evaluación objetiva e imparcial.

3.2.2. Modelos clásicos de aprendizaje de máquina

Los modelos clásicos reciben como entrada un vector de características extraídas de cada imagen segmentada. Como se describió en la sección 3.1, se trabajó con dos versiones, uno de 31 características, el cual se usó para las primeras pruebas, y uno ampliado de 76 características. Dado que las características tienen escalas y rangos distintos entre sí, como por ejemplo valores de media de color en un rango de [0, 255] frente a valores de los histogramas de color más pequeños, fue necesario aplicar una estandarización previa mediante StandarScaler de Scikit-learn [73], que transforma cada característica para que tenga media 0 y desviación estándar 1.

Para la búsqueda de los mejores hiperparámetros se utilizó GridSearchCV [74] con validación cruzada de 5 folds, evaluando distintas combinaciones de parámetros y seleccionando la configuración que maximizara el F1-macro en clasificación y minimizara el MAE en regresión. Los mejores hiperparámetros encontrados se usaron para el entrenamiento y evaluación final de cada modelo.

Modelos de Clasificación

La SVM fue configurada con kernel RBF, que permite capturar las relaciones no lineales entre las características [66]. Tras la búsqueda de hiperparámetros, los valores más efectivos encontrados fueron C = 1.000.000 y gamma = 0.001, donde C controla el equilibrio entre maximizar el margen de separación y minimizar los errores, y gamma determina el radio de influencia de cada punto de entrenamiento [75]. Se activó el parámetro “class_weight= “balanced”” para compensar el desbalance entre clases y la estrategia “decision_function_shape= “ovr”” para manejar el problema multiclase [73].

El Random Forest Classifier fue configurado con 200 árboles, con un criterio de entropía, una profundidad máxima de 10 niveles, el parámetro “class_weight= “balanced_subsample”” para manejar el desbalance dentro de cada árbol, y se activó el cálculo del error out-of-bag (OOB) como estimación adicional del desempeño durante el entrenamiento [76].

El KNN Classifier fue configurado con k = 10 vecinos, una ponderación por distancia y métrica Manhattan [77]. La ponderación por distancia hace que los vecinos más cercanos tengan mayor influencia sobre la predicción que los más lejanos, lo cual es útil cuando las clases no están claramente separadas. La métrica Manhattan fue seleccionada por su robustez ante características con distribuciones asimétricas. Para compensar el desbalance se aplicó SMOTE [78] solamente sobre el conjunto de entrenamiento, generando muestras sintéticas de las clases minoritarias antes de entrenar el modelo.

El MLP Classifier fue configurado con dos capas ocultas de tamaño (64, 32), función de activación ReLU, optimizador Adam, término de regularización L2 con Alpha = 0.01, y un máximo de 800 iteraciones [79]. La arquitectura de dos capas permitió capturar las relaciones no lineales entre las características extraídas sin añadir complejidad que pudiese dificultar el entrenamiento. Al igual que KNN, se aplicó SMOTE en el conjunto de entrenamiento.

La Figura 17 resume de forma esquemática el pipeline completo seguido por los modelos clásicos de clasificación, desde la entrada del vector de características hasta la evaluación final.

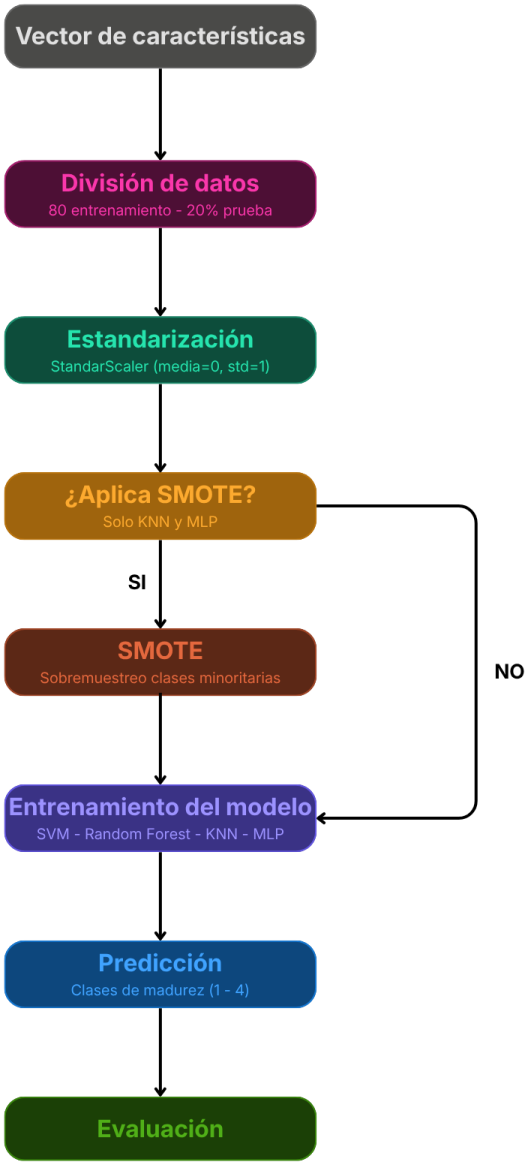


Fig. 17. Pipeline completo de los modelos de clasificación.

Modelos de Regresión

Para construir el dataset de regresión, se partió del mismo CSV de características usado en clasificación. A cada imagen se le calculó la variable objetivo "days_ramaining", que representa cuántos días le restaban al aguacate hasta alcanzar el estado Podrido. Este valor fue calculado identificando para cada aguacate el primer día en que su estado de madurez alcanzó el valor 4, y restando a ese día el día en que fue tomada cada fotografía. Para los aguacates que no registraron el estado Podrido dentro del experimento, se utilizó el día 9 como referencia máxima. Formalmente, se puede ver de esta forma. Sea d_i el día en que fue tomada la fotografía de la imagen i, y sea $d_{podrido}(a)$ el primer día en que el aguacate a alcanzó el índice de madurez 4, siendo el estado podrido, por lo tanto, la variable objetivo se define como:

$$days\_remaining_i = \max(0, d_{podrido}(a) - d_i)$$

La función max(0, …) asegura que days_remaining no tome valores negativos, considerando como cero aquellos casos en los que la fecha de la fotografía supera el día de podredumbre registrado, lo que indica que el aguacate ya está podrido.

El SVR fue configurado con kernel RBF, C = 30, gamma = "scale" y épsilon = 0.4 [80]. El valor del C permite al modelo cierta flexibilidad para ajustarse a los datos sin penalizar los errores fuera de la banda de tolerancia definida por épsilon, mientras que el valor de gamma ajusta automáticamente la influencia de cada punto de entrenamiento en función del número de características y la varianza de los datos.

El Random Forest Regressor fue configurado con 200 árboles, un criterio de error absoluto medio (MAE) para medir la calidad de las divisiones y una profundidad máxima de 10 niveles [81]. El uso del MAE como criterio de división lo hace más robusto ante valores atípicos en la variable objetivo, dado que no amplifica el efecto de predicciones muy alejadas al valor real como si lo hiciese el error cuadrático medio.

El KNN Regressor fue configurado con k = 10 vecinos, ponderación por distancia y la métrica Manhattan [82]. En regresión, el modelo predice el valor de la variable objetivo como un promedio ponderado de los valores de sus k vecinos más cercanos, donde los vecinos próximos tienen mayor peso en la predicción final.

El MLP Regressor fue configurado con tres capas ocultas de tamaño (64, 32, 16), función de activación ReLU, optimizador Adam, término de regularización L2 con alpha = 0.001 y un máximo de 800 iteraciones [83]. La arquitectura de tres capas permite capturar relaciones más complejas entre las características y el valor continuo de días restantes. La salida del modelo es una única neurona sin función de activación, produciendo directamente el valor numérico predicho.

La Figura 18 resume el pipeline seguido por los modelos de regresión, el cual difiere del de clasificación por la incorporación del cálculo de days_remaining como variable objetivo y la ausencia de SMOTE, dado que la variable de salida es continua.

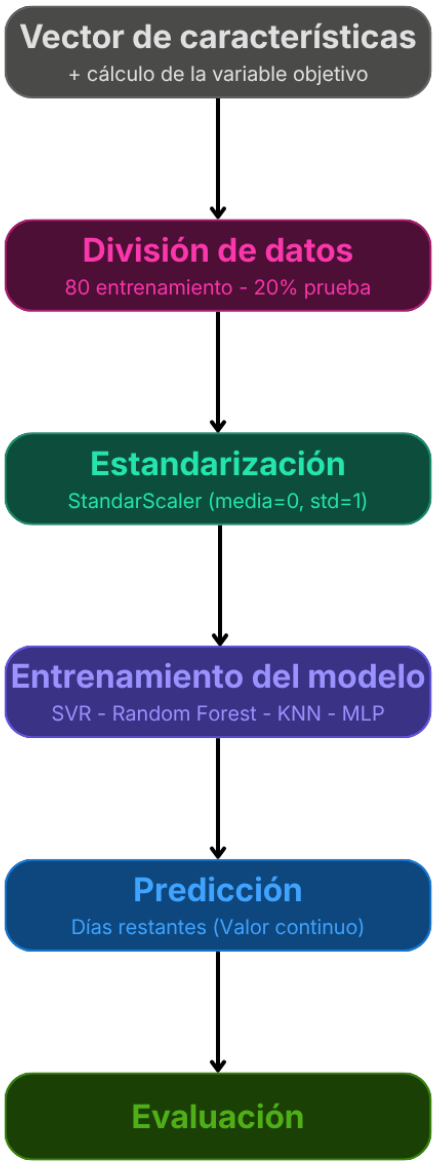


Fig. 18. Pipeline completo de los modelos de regresión.

3.2.3. Modelos de aprendizaje profundo

Los modelos de aprendizaje profundo fueron adaptados mediante aprendizaje por transferencia, reutilizando el conocimiento visual adquirido durante el preentrenamiento en ImageNet y reemplazando únicamente la capa de salida final para adaptarla al problema específico del aguacate Hass.

Preprocesamiento por arquitectura

Cada arquitectura requiere un tamaño de entrada especifico según su diseño original. ResNet-18 [62], VGG16 [63] y MobileNetV3 Large [64] reciben imágenes de 224x224 pixeles, por lo que las imágenes fueron redimensionadas a 256x256 y luego recortadas al centro a 224x224. EfficientNet-B3 [65] requiere de imágenes de 300x300 píxeles, por lo que se redimensionaron a 320x320 píxeles y se recortaron al centro a 300x300 píxeles. En todos los casos, los valores de los píxeles fueron escalados al rango [0.0, 1.0] y normalizados con media = [0.485, 0.456, 0.406] y desviación estándar = [0.229, 0.224, 0.225], correspondientes a los valores de ImageNet con los que fueron preentrenados.

Configuración de la cabeza y la capa de salida para clasificación y regresión

En todos los modelos se reemplazó la capa de salida original por una nueva cabeza con la misma estructura, tal como se ilustra en la Figura 19. La cabeza inicia con una capa de Dropout con tasa de 0.4, lo que significa que durante el entrenamiento el 40% de las neuronas se desactivan aleatoriamente en cada paso, forzando al modelo a no depender de conexiones específicas y así reducir el riesgo de sobreajuste. Luego, se aplica una capa densa que reduce esa representación a 256 neuronas, seguida de una activación ReLU, que permite al modelo aprender patrones más complejos a partir de las características extraídas. Después, se aplica un segundo Dropout de 0.2, más suave que el primero dado que la representación ya es más compacta y solo requiere de una regularización más ligera antes de llegar a la salida. Finalmente se añade la capa de salida, cuya configuración varía según la tarea. Para clasificación tiene 4 neuronas correspondientes a las cuatro clases de madurez, y usa CrossEntropy como función de perdida. Para regresión la capa de salida es una sola neurona que produce directamente el valor continuo de días restantes, y usa MSELoss como función de perdida.

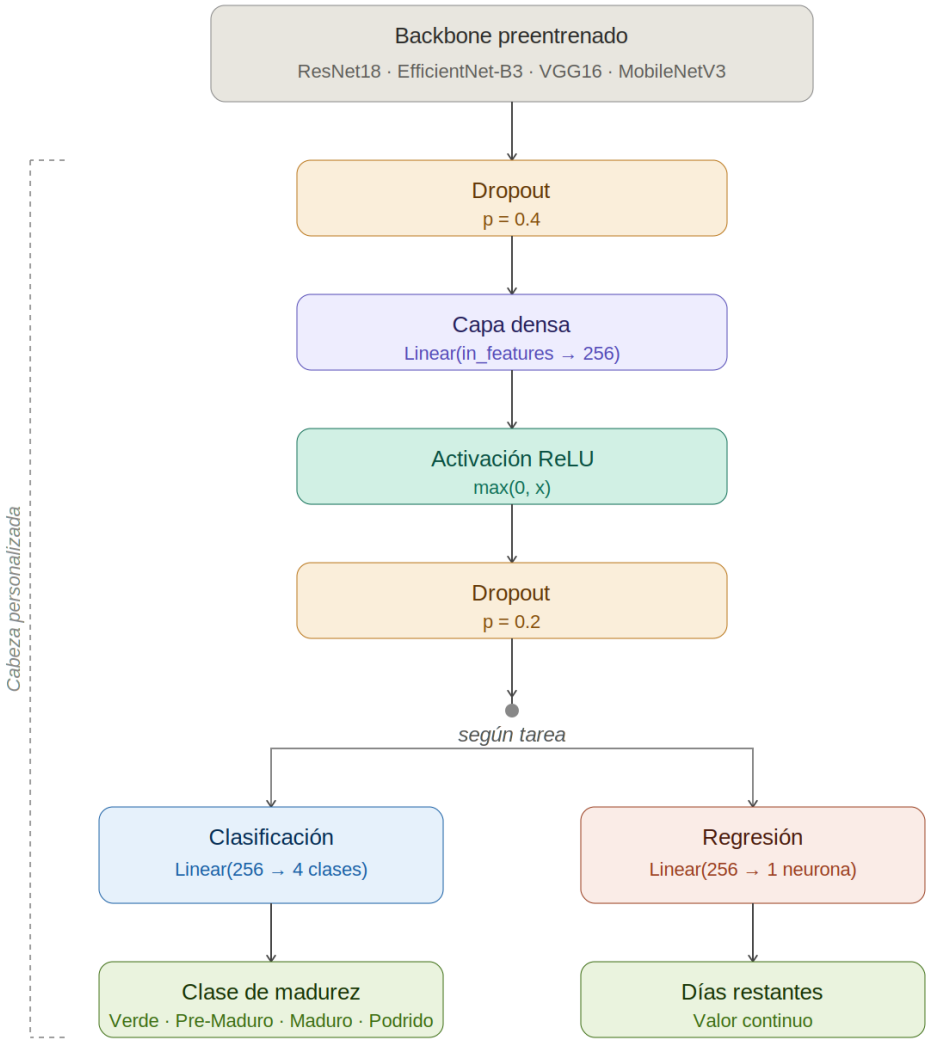


Fig. 19. Cabeza y capa de salida para clasificación y regresión

Entrenamiento de los modelos

El entrenamiento de todos los modelos siguió una estrategia de dos fases. En la primera fase se congeló completamente el backbone preentrenado y se entrenó únicamente la cabeza durante 10 épocas con una tasa de aprendizaje de 0.001 y un weight decay de 0.0001, permitiendo así que los nuevos pesos se estabilizaran antes de tocar las capas preentrenadas. En la segunda fase se liberaron las capas del backbone según la configuración del fine-tuning en cada modelo, y se continuó el entrenamiento con una tasa de aprendizaje un poco más conservadora de 0.0001 durante un máximo de entre 30-40 épocas para clasificación y regresión. En ambas fases se usó el optimizador Adam y el scheduler ReduceLROnPlateau, el cual reduce la tasa de aprendizaje a la mitad cuando el desempeño de validación no mejora durante 3 épocas consecutivas.

Experimentos de fine-tuning

Con el propósito de identificar qué proporción del backbone preentrenado convenía liberar para que cada modelo se adaptara mejor a las características visuales del aguacate Hass sin perder el conocimiento adquirido en ImageNet, se hicieron algunos experimentos variando el porcentaje del backbone descongelado durante la segunda fase de entrenamiento para las cuatro arquitecturas. Se comenzó probando el backbone completamente congelado, luego fue aumentando a 25%, 50% y 75%.

En ResNet-18, el porcentaje se aplica sobre sus cuatro bloques, comenzando siempre por los más profundos que capturan características más específicas, por lo que al 25% liberó el ultimo bloque, al 50% liberó los últimos dos, y al 75% los tres últimos.

En EfficientNet-B3, el porcentaje se aplica sobre sus 9 bloques de características, liberando en cada configuración los bloques más profundos, siendo 3 bloques al 25%, 5 al 50% y 7 al 75%.

En VGG16, el porcentaje se aplica sobre sus 5 bloques convolucionales, liberando así 2 bloques al 25%, 3 bloques al 50%, y 4 bloques al 75%, siempre comenzando por los más profundos.

En MobileNetV3 Large, el porcentaje se aplica sobre sus 16 bloques convolucionales, liberando 4 bloques al 25%, 8 bloques al 50%, y 12 al 75%.

Cabe recalcar que, en todas las configuraciones, independientemente de cuántas capas del backbone se liberen, la cabeza personalizada siempre se entrena, ya que es la parte del modelo que debe aprender directamente las características del aguacate Hass. Para un mejor entendimiento de lo que se hizo, la Figura 20 ilustra las cuatro configuraciones de fine-tuning aplicadas, tomando como ejemplo ResNet-18, donde se ve cómo se van liberando progresivamente los bloques del backbone comenzando siempre por los más profundos.

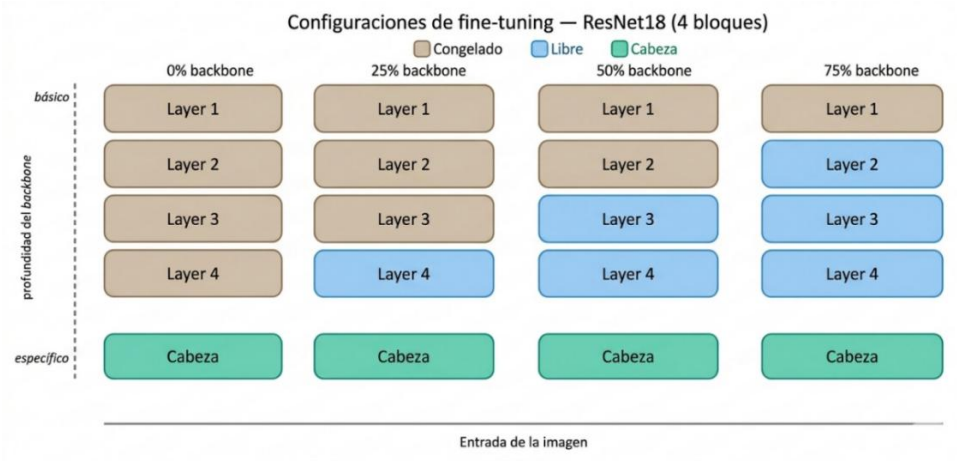


Fig. 20. Configuraciones de fine-tuning del backbone de ResNet-18 según el porcentaje de capas descongeladas.

3.3. Evaluación del desempeño de los modelos

En esta sección se presentan y analizan los resultados de los modelos implementados para clasificar el estado de madurez del aguacate Hass y predecir el tiempo de vida útil restante en días. Para determinar qué enfoque generaliza mejor sobre imágenes no vistas durante el entrenamiento, se evaluaron dos tipos de modelos. Primero, modelos clásicos de aprendizaje automático que utilizan vectores de características extraídas a partir de las imágenes segmentadas. Segundo, modelos de aprendizaje profundo basados en redes neuronales convolucionales preentrenadas, que procesan las imágenes directamente y aprenden representaciones visuales de forma automática. La evaluación fue independiente para cada tarea. En clasificación se usaron métricas como accuracy, precision, recall, F1-score y matriz de confusión, para analizar el desempeño global. En regresión se emplearon MAE, RMSE, SMAPE, R^2 y R^2 ajustado, para cuantificar el error en días y revisar qué tan preciso estuvo.

3.3.1. Evaluación de los modelos clásicos de aprendizaje automático

Como primera aproximación, se evaluaron modelos clásicos de aprendizaje automático con el fin de determinar si las características visuales extraídas eran suficientes para resolver el problema principal. Los modelos clásicos evaluados fueron Support Vector Machine, Random Forest, K-Nearest Neighbors y Perceptrón Multicapa, tanto en su versión de clasificación como de regresión.

Resultados de clasificación con modelos clásicos

Para la tarea de clasificación, los modelos clásicos fueron evaluados sobre las cuatro clases de madurez ya definidas previamente, que son Verde, Premaduro, Maduro y Podrido, y se realizaron dos evaluaciones. Primero se usó el vector inicial de 31 características y luego se utilizó el vector ampliado de 76 características, el cual incorporó descriptores adicionales de textura calculados sobre diferentes canales de color. La Tabla 2 presenta los resultados obtenidos por los modelos clásicos de clasificación para ambas configuraciones de características.

Tabla 2. Resultados de las métricas de clasificación obtenidos por los modelos clásicos usando 31 y 76 características.

Modelo	Nº de características	Accuracy	Precision	Recall	F1-score
SVM RBF	31	0.7641	0.7586	0.7746	0.7648
SVM RBF	76	0.7385	0.7302	0.7430	0.7355
Random Forest	31	0.7607	0.7631	0.7745	0.7672
Random Forest	**76**	**0.7846**	**0.7915**	**0.8054**	**0.7955**
KNN	31	0.7179	0.7219	0.7457	0.7279

KNN	76	0.7111	0.7218	0.7432	0.7215
MLP	31	0.7863	0.7825	0.7885	0.7847
MLP	76	0.7726	0.7738	0.7769	0.7751

A partir de los resultados obtenidos, se observa que la ampliación del vector de características no produjo una mejora en todos los modelos. En el caso de SVM, KNN y MLP, el desempeño fue superior con el vector de 31 características, por lo que el incremento de características introdujo información redundante que dificultó la separación entre clases. Por el contrario, Random Forest sí se benefició del vector ampliado de 76 características, pasando de un F1-score de 0.7672 a 0.7955, y con un accuracy de 0.7846, lo que lo convierte en el mejor modelo clásico evaluado.

Analizando sus métricas, el Random Forest con 76 características alcanzó una precisión de 0.7846, lo que indica que aproximadamente el 78% de las predicciones realizadas fueron correctas. Su recall de 0.8054 es especialmente relevante en este contexto, ya que significa que el modelo fue capaz de identificar correctamente el 80% de los aguacates de cada clase, reduciendo así el riesgo de errores críticos como clasificar un aguacate podrido como maduro. El F1-score de 0.7955 refleja un balance adecuado entre ambas métricas, confirmando que el modelo no sacrificó una a expensas de la otra. Aunque el MLP con 31 características obtuvo el mayor accuracy con 0.7863, se priorizó en este caso el F1-score porque permite evaluar de manera más equilibrada el desempeño del modelo, especialmente cuando interesa que todas las clases sean reconocidas adecuadamente y no únicamente maximizar el porcentaje global de aciertos.

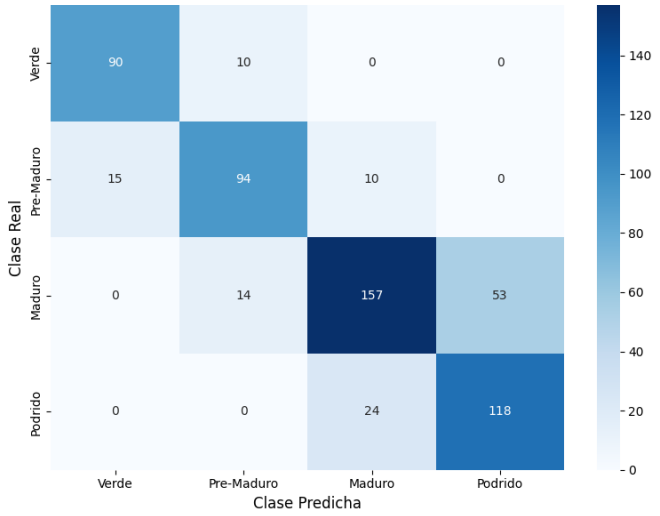


Fig. 21. Matriz de confusion Random Forest con 76 características.

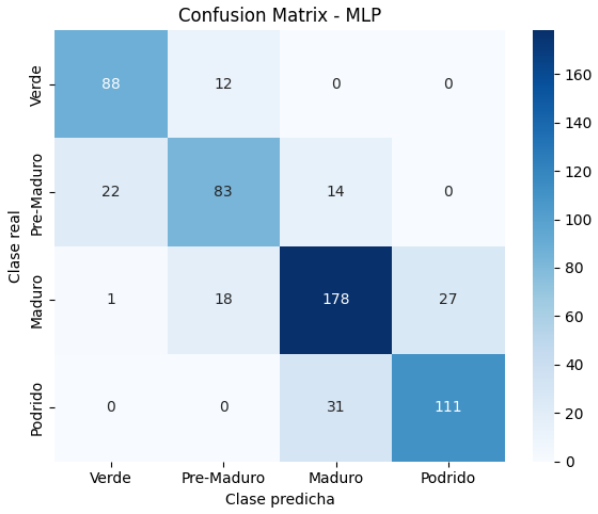


Fig. 22. Matriz de confusion MLP con 31 características

Con el fin de mantener la claridad, se presentan únicamente las matrices de confusión de los dos modelos con mejor desempeño. Random Forest con 76 características, presentada en la figura 21, por obtener el mayor F1-score, y MLP con 31 características, presentada en la figura 22, por alcanzar el mayor accuracy. Las matrices restantes se incluyen en la sección 2 de los anexos.

En el Random Forest, las clases Verde y Premaduro fueron reconocidas con buen desempeño, clasificando correctamente 90 y 94 imágenes respectivamente. Las confusiones entre estas dos clases son esperables dado que los cambios visuales entre estados de madurez cercanos son graduales y no siempre fácilmente distinguibles. La mayor dificultad se concentró en la frontera entre Maduro y Podrido, 53 imágenes reales de la clase Maduro fueron clasificadas como Podrido, y 24 imágenes de Podrido fueron clasificadas como Maduro.

Igualmente, el MLP mostró un comportamiento similar. Aunque clasificó correctamente 178 imágenes de la clase Maduro, 27 de ellas fueron confundidas con Podrido, y 31 imágenes reales de Podrido fueron clasificadas como Maduro, esto confirma que la confusión entre los estados finales de maduración es una limitación común en los modelos clásicos evaluados, y no exclusiva de una arquitectura particular. Al final, esto evidencia que las características extraídas no logran capturar con suficiente precisión las diferencias visuales en los estados finales de maduración, donde la cáscara del aguacate comparte tonalidades oscuras y texturas similares entre ambas clases.

En general, los modelos clásicos lograron establecer una línea base aceptable para la clasificación del estado de madurez, alcanzando valores de accuracy cercanos al 78% en los mejores casos. Este resultado confirma que las características extraídas, especialmente las relacionadas con el color, la textura y la distribución del canal Hue, contienen información relevante para diferenciar las etapas de maduración del aguacate Hass, validando que existe una relación entre los cambios visuales del fruto y su estado de madurez. Sin embargo, las matrices de confusión evidencian que estos modelos presentan dificultades para distinguir clases visualmente similares. Cabe recalcar que el aguacate Hass no transita entre estados de forma abrupta, sino mediante cambios progresivos de color, textura y apariencia. Esto hace que imágenes de etapas cercanas compartan patrones visuales parecidos, dificultando la construcción de fronteras de decisión claras entre clases. Esta dificultad fue especialmente evidente entre las clases Maduro y Podrido como se mencionó anteriormente, donde las diferencias pueden depender de detalles como manchas oscuras, variaciones irregulares de tonalidad o patrones de textura no uniformes.

Resultados de regresión con modelos clásicos

Para la tarea de regresión, los modelos clásicos fueron evaluados con el objetivo de predecir el número de días restantes hasta que el aguacate alcanzara el estado de deterioro completo. Al igual que en la tarea de clasificación, los modelos fueron evaluados usando dos representaciones del conjunto de datos, el vector inicial de 31 características y el vector ampliado de 76 características. Esta comparación permitió analizar si la incorporación de características adicionales de textura y color también favorecía la predicción continua de vida útil.

Tabla 3. Resultados de las métricas de regresión obtenidos por los modelos clásicos usando 31 y 76 características

Modelo	Nº de características	MAE	RMSE	SMAPE	R^2	R^2 ajustado
SVR RBF	31	0.6551	0.8606	66.60%	0.9135	0.9086
SVR RBF	**76**	**0.6023**	**0.7759**	**63.48%**	**0.9297**	**0.9192**
Random Forest Regressor	31	0.7004	0.9646	66.18%	0.8913	0.8852
Random Forest Regressor	76	0.6987	0.9587	65.88%	0.8926	0.8766
KNN Regressor	31	0.7540	0.9735	66.13%	0.8893	0.8831
KNN Regressor	76	0.7462	0.9491	63.87%	0.8948	0.8790
MLP Regressor	31	0.7240	0.9320	67.03%	0.8985	0.8928
MLP Regressor	76	0.7239	0.9325	66.73%	0.8984	0.8832

Los resultados de la Tabla 3 muestran que el SVR con kernel RBF y 76 características fue el modelo con mejor desempeño en la tarea de regresión. Su MAE de 0.6023 días indica que en promedio sus predicciones se desviaron menos de un día respecto al valor real de vida útil restante, lo cual es relevante para el contexto del proyecto, ya que un error de esta magnitud puede considerarse aceptable para apoyar decisiones prácticas sobre la gestión y consumo del aguacate Hass. Así mismo, su RMSE de 0.7759 días confirma que el modelo no solo tuvo un buen desempeño promedio, sino que también limitó la presencia de errores grandes en sus predicciones. El R^2 de 0.9297 indica que el modelo logró explicar aproximadamente el 93% de la variabilidad en los días restantes de vida útil, y el R^2 ajustado de 0.9192 confirma que este alto poder explicativo se mantiene incluso penalizando la complejidad asociada al uso de más variables. La ampliación del vector de 31 a 76 características produjo una mejora clara en el SVR, reduciendo el MAE de 0.6551 a 0.6023 días y el RMSE de 0.8606 a 0.7759 días, lo que indica que los descriptores adicionales aportaron información útil para estimar la vida útil restante.

En los demás modelos, el impacto del vector ampliado fue más limitado. El Random Forest Regressor mostró una mejora marginal en MAE y RMSE, pasando de 0.7004 a 0.6987 días y de 0.9646 a 0.9587 días respectivamente, pero su R^2 ajustado disminuyó de 0.8852 a 0.8766, lo que sugiere que las características adicionales introdujeron complejidad sin aportar una mejora, posiblemente porque algunas de las nuevas variables resultaron redundantes o poco informativas para este modelo. El KNN Regressor también mejoró ligeramente con 76 características, reduciendo su MAE de 0.7540 a 0.7462 días y su SMAPE de 66.13% a 63.87%, pero su desempeño siguió siendo inferior al SVR y al MLP. Esto puede explicarse por la naturaleza de este modelo ya que al depender directamente de la distancia entre muestras en el espacio de características, el aumento de dimensionalidad puede dificultar la identificación de vecinos similares cuando no todas las variables aportan información importante. El MLP Regressor, por su parte, mostró resultados prácticamente idénticos entre ambas configuraciones, con un MAE de 0.7240 y 0.7239 días respectivamente y un R^2 estable cercano a 0.898, lo que indica que este modelo ya había capturado la información más relevante con el vector inicial de 31 características y no logró beneficiarse del vector ampliado. En términos generales, los resultados de regresión con modelos clásicos son prometedores, especialmente los del SVR, cuyo R^2 superior al 92% sugiere que las características visuales extraídas manualmente sí guardan una relación con la vida útil del aguacate Hass.

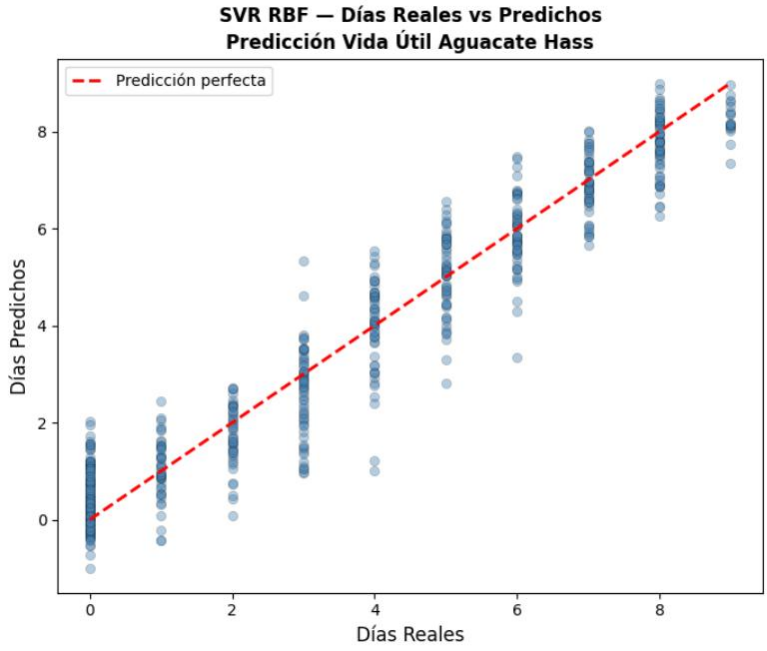


Fig. 23. Gráfica de dispersión del SVR con 76 características.

Ahora, al analizar los gráficos de dispersión entre los días reales y los días predichos, se toma como referencia principal el de la Fig. 23, correspondiente al SVR con 76 características por ser el modelo con mejor desempeño general en regresión. Las gráficas de los demás modelos siguen un patrón similar y se incluyen en la sección 3 de los anexos con más detalle. En el gráfico del SVR se observa una tendencia positiva clara, a medida que aumenta el valor real de días restantes, la predicción del modelo también aumenta de forma proporcional, confirmando que las características visuales extraídas contienen información relevante sobre el avance del proceso de maduración. No obstante, se aprecia cierta dispersión alrededor de la línea de predicción perfecta, especialmente en valores intermedios del rango de días, lo cual evidencia que aunque los modelos capturan bien la tendencia general, presentan dificultades para estimar con exactitud algunos casos específicos, y esto se explica por la naturaleza del problema. Dos aguacates visualmente similares pueden tener valores distintos de vida útil restante porque la maduración del aguacate Hass es un proceso gradual y continuo, donde pequeños cambios de color, textura o deterioro superficial pueden corresponder a diferencias temporales que las características extraídas mediante procesamiento de imágenes no logran capturar con suficiente precisión.

En términos generales, los resultados obtenidos con los modelos clásicos para regresión muestran un desempeño superior al observado en clasificación. Mientras que en clasificación los mejores modelos alcanzaron valores de accuracy cercanos al 78%, en regresión se obtuvieron valores de R^2 superiores a 0.90 en el caso del SVR, lo que indica una buena capacidad para modelar la relación entre las características extraídas y los días restantes de vida útil. Esto puede explicarse porque la variable objetivo de regresión sigue una progresión temporal asociada al proceso de maduración, y varias características visuales, especialmente las relacionadas con color y textura, cambian de forma relativamente continua durante dicho proceso. A pesar de estos resultados favorables, los modelos clásicos de regresión siguen presentando la misma limitación general observada en clasificación que es que no trabajan directamente con la imagen, sino con el vector de características extraído de ella. Aunque el vector de características permite capturar información relevante, como tonalidades predominantes, variabilidad de color y patrones de textura, no conserva por completo la distribución espacial de los cambios visuales sobre la superficie del fruto. Por esta razón los resultados de regresión también justifican el hecho de hacer una evaluación posterior de modelos de aprendizaje profundo, con el fin de determinar si las redes convolucionales pueden aprender representaciones visuales más complejas directamente desde las imágenes originales.

3.3.2. Evaluación de los modelos de aprendizaje profundo CNN

Una vez evaluados los modelos clásicos, se procedió a analizar el desempeño de modelos de aprendizaje profundo basados en redes neuronales convolucionales, con el propósito de determinar si estas arquitecturas permitían superar las limitaciones de la pérdida de información espacial y la dependencia de características extraídas por uno. A diferencia de los modelos clásicos, las redes convolucionales reciben la imagen directamente como entrada y aprenden por sí mismas qué patrones visuales son relevantes para la tarea, lo que les permite capturar más información y distinguir mejor los estados de madurez cercanos. Los modelos evaluados fueron ResNet-18, EfficientNet-B3, VGG16 y MobileNetV3 Large, todos adaptados mediante aprendizaje por transferencia, reemplazando su capa de salida original por una cabeza personalizada para cada tarea. Adicionalmente, se realizaron experimentos de fine-tuning sobre los modelos, variando el porcentaje del backbone descongelado, para identificar el nivel de adaptación que maximizará el desempeño sin provocar sobreajuste. Las métricas de evaluación se mantuvieron iguales a las utilizadas con los modelos clásicos, permitiendo una comparación directa entre ambos enfoques.

Resultados de clasificación con modelos CNN

Para la tarea de clasificación se evaluó el efecto del aumento de datos sobre el desempeño de los modelos. Por esta razón cada arquitectura fue entrenada y evaluada bajo dos configuraciones, una primera sin aumento de datos y una segunda con aumento de datos, las cuales incorpora transformaciones aleatorias durante el entrenamiento, tales como rotaciones, desplazamientos, variaciones de brillo, contraste, saturación, distorsión de perspectiva y desenfoque gaussiano. Estas transformaciones fueron descritas previamente en la sección de preprocesamiento y tuvieron como objetivo mejorar la capacidad de generalización de los modelos frente a posibles variaciones en la captura de imágenes.

Tabla 4. Resultados de clasificación obtenidos por las arquitecturas CNN con y sin aumento de datos

Arquitectura	Aumento de datos	Accuracy	Precision	Recall	F1-score
ResNet-18	**No**	**0.8727**	**0.8763**	**0.8727**	**0.8741**
ResNet-18	Sí	0.8591	0.8585	0.8591	0.8586
EfficientNet-B3	No	0.8568	0.8641	0.8568	0.8592
EfficientNet-B3	**Sí**	**0.8727**	**0.8761**	**0.8727**	**0.8735**
VGG16	No	0.8477	0.8584	0.8477	0.8503
VGG16	Sí	0.8523	0.8580	0.8523	0.8530
MobileNetV3	No	0.8273	0.8343	0.8273	0.8294
MobileNetV3	Sí	0.8114	0.8216	0.8114	0.8105

Los resultados de la Tabla 4 muestran que los modelos CNN superaron claramente a los modelos clásicos en la tarea de clasificación. Mientras que el mejor modelo clásico alcanzó un F1-score de 0.7955, las arquitecturas convolucionales superaron 0.85 en la mayoría de los casos, evidenciando su mayor capacidad para capturar patrones visuales del proceso de maduración. El mejor desempeño correspondió a ResNet-18 sin aumento de datos, con un accuracy de 0.8727 y F1-score de 0.8741, seguido de cerca por EfficientNet-B3 con aumento de datos, que alcanzó un accuracy de 0.8727 y F1-score de 0.8735, por lo que ambas configuraciones pueden considerarse las más competitivas en esta tarea.

El efecto del aumento de datos no fue uniforme entre arquitecturas. En EfficientNet-B3 y VGG16 produjo mejoras, incrementando el F1-score de 0.8592 a 0.8735 y de 0.8503 a 0.8530 respectivamente, lo que sugiere que las transformaciones aplicadas favorecieron su capacidad de generalización. En cambio, ResNet-18 y MobileNetV3 mostraron una reducción en el desempeño al aplicarlo. Esto no implica que el aumento de datos sea perjudicial en términos generales, sino que su efecto depende de la arquitectura y de las características del conjunto de datos. En este caso, dado que las imágenes presentan aguacates centrados sobre fondo blanco en condiciones controladas, transformaciones como las mencionadas anteriormente, pueden alejar las imágenes aumentadas de la distribución real del conjunto de prueba, dificultando así el aprendizaje en lugar de favorecerlo.

Otro punto importante a tratar son las curvas de pérdida y accuracy por época, como las que se muestran en la Fig. 24, que permiten identificar la presencia de sobreajuste durante el entrenamiento. En general se observa una diferencia considerable entre el desempeño en entrenamiento y en validación, mientras la curva de entrenamiento mejora de forma sostenida, la de validación tiende a estabilizarse o a oscilar, lo que indica que los modelos aprenden bien las imágenes del conjunto de entrenamiento, pero no siempre logran trasladar ese mismo desempeño a los datos de prueba. Este comportamiento es especialmente visible en las configuraciones sin aumento de datos, donde la brecha entre ambas curvas es más amplia, sugiriendo que el modelo memoriza patrones específicos del conjunto de entrenamiento en lugar de generalizar.

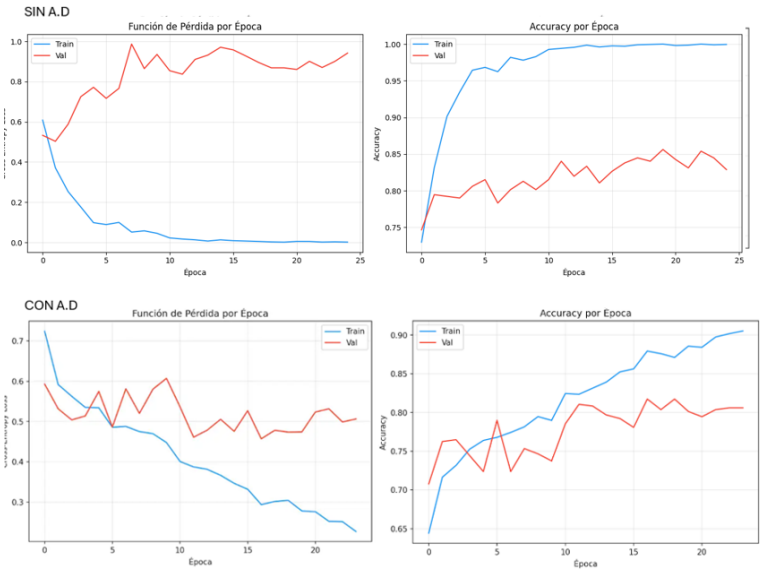


Fig. 24. Gráfica de Perdida y Accuracy por época de ResNet-18. En los dos bloques de arriba no se usó aumento de datos y en los dos bloques de abajo se hizo lo contrario.

En el caso de ResNet-18 sin aumento de datos, como se observa en la Fig. 24, la curva de entrenamiento muestra una disminución rápida de la pérdida hasta valores cercanos a cero, mientras que la pérdida de validación aumenta y se mantiene alta. De forma similar, el accuracy de entrenamiento se aproxima al 100% mientras que el de validación se estabiliza en valores inferiores, lo cual da a entender que hay sobreajuste. El modelo memoriza las imágenes de entrenamiento, pero generaliza con menor efectividad. A pesar de esto, su desempeño en el conjunto de prueba fue uno de los mejores entre todas las configuraciones evaluadas, lo que sugiere que aunque existe sobreajuste en las curvas de entrenamiento, el modelo conserva una buena capacidad predictiva final, posiblemente porque la distribución del conjunto de prueba es suficientemente similar a la del entrenamiento.

Por otro lado, con aumento de datos, el sobreajuste se reduce, aproximadamente 10 puntos porcentuales. La pérdida de entrenamiento disminuye y el accuracy de entrenamiento no alcanza valores tan extremos. Sin embargo, sigue existiendo una brecha entre las curvas de entrenamiento y validación, lo que indica que el aumento de datos ayudó a regularizar el entrenamiento sin eliminar completamente el sobreajuste.

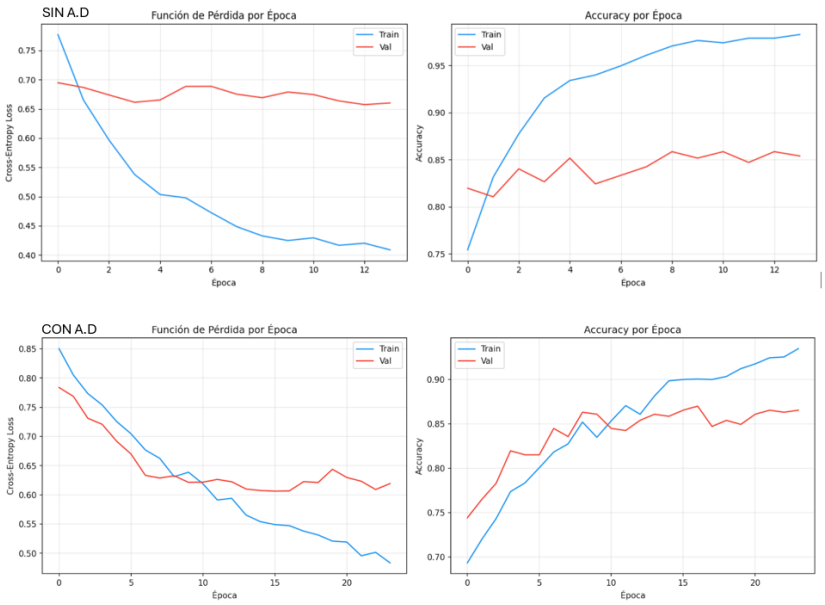


Fig. 25. Gráfica de Perdida y Accuracy por época de EfficientNet-B3. En los dos bloques de arriba no se usó aumento de datos y en los dos bloques de abajo se hizo lo contrario.

En EfficientNet-B3 sin aumento de datos como se muestra en la Fig. 25, se nota una separación marcada entre las curvas de entrenamiento y validación, donde el accuracy de entrenamiento sigue aumentando mientras el de validación se estabiliza, lo que indica un sobreajuste. Al incorporar aumento de datos, el comportamiento cambia notablemente. Ambas curvas se mantienen más cercanas durante el entrenamiento y el accuracy de validación mejora, reduciendo el sobreajuste considerablemente. Esta configuración obtuvo uno de los mejores resultados del experimento, confirmando que EfficientNet-B3 se benefició del aumento de datos tanto en desempeño como en capacidad de generalización.

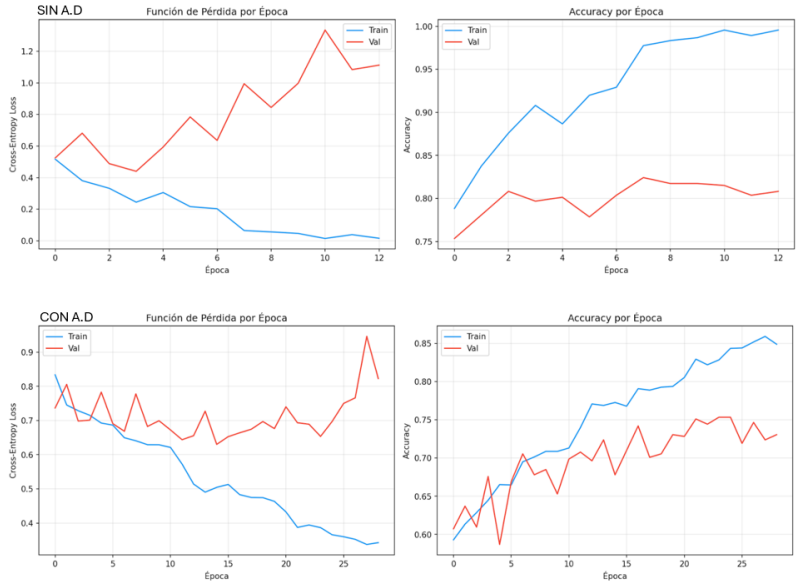


Fig. 26. Gráfica de Perdida y Accuracy por época de VGG16. En los dos bloques de arriba no se usó aumento de datos y en los dos bloques de abajo se hizo lo contrario.

En VGG16 sin aumento de datos como se muestra en la Fig. 26, se evidencian señales claras de sobreajuste. La pérdida de entrenamiento disminuye mientras que la de validación aumenta considerablemente en varias épocas, y el accuracy de entrenamiento alcanza valores muy altos mientras el de validación se mantiene por debajo, lo que indica que el modelo memorizó patrones del conjunto de entrenamiento sin lograr generalizarlos. Al incorporar aumento de datos se observa una ligera mejora en el conjunto de prueba y una reducción parcial del sobreajuste, aunque la brecha entre ambas curvas siguió siendo evidente, VGG16 no tuvo los resultados esperados en comparación con EfficientNet-B3.

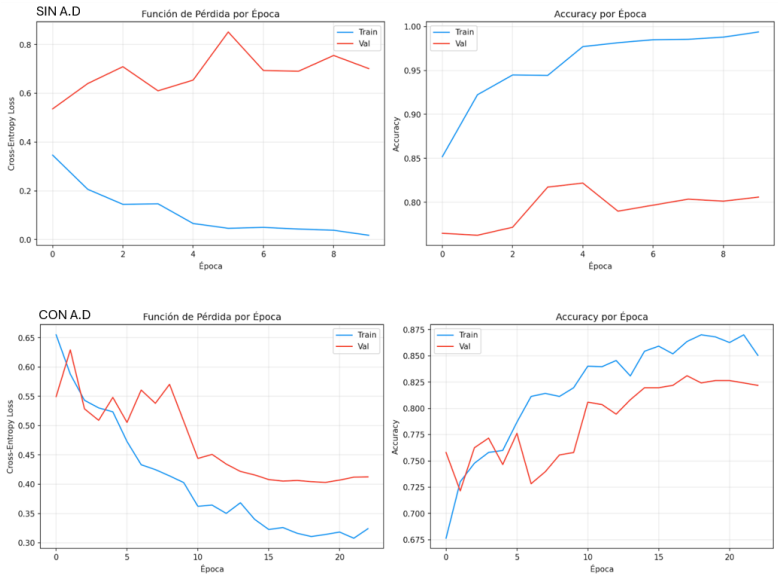


Fig. 27. Gráfica de Perdida y Accuracy por época de MobileNetV3. En los dos bloques de arriba no se usó aumento de datos y en los dos bloques de abajo se hizo lo contrario.

Como se muestra en la Fig. 27, MobileNetV3 sin aumento de datos tuvo sobreajuste. La pérdida de entrenamiento descendió rápidamente hasta valores muy bajos mientras que la de validación se mantuvo alta y con oscilaciones, y el accuracy de entrenamiento se aproximó al 100% frente a un accuracy de validación estacionado alrededor del 80%, lo que indica que el modelo memorizó los patrones del conjunto de entrenamiento sin lograr generalizarlos. Luego, al incorporar aumento de datos el comportamiento mejoró. La pérdida de entrenamiento descendió lentamente, la de validación se mantuvo más próxima a ella, y la brecha entre las precisiones se redujo significativamente, por lo que las transformaciones ayudaron a la regularización de los datos. Sin embargo, esta reducción del sobreajuste no se tradujo en una mejora del desempeño final, ya que el F1-score pasó de 0.8294 sin aumento de datos a 0.8105 con aumento. Entonces, al fin y al cabo, las transformaciones aplicadas si bien controlaron la brecha entre curvas, pudieron haber dificultado el aprendizaje de patrones discriminantes.

El análisis anterior evidenció que el sobreajuste fue un factor recurrente en todos los modelos, pero su impacto no fue uniforme. Si bien el aumento de datos ayudó a reducir la separación entre las curvas de entrenamiento y validación, esto no siempre se tradujo en una mejora en el conjunto de prueba. En EfficientNet-B3 sí permitió obtener un mejor equilibrio entre generalización y desempeño, mientras que en ResNet-18 y MobileNetV3 la reducción del sobreajuste estuvo acompañada de una caída en las métricas finales. Esto demuestra que una menor brecha entre curvas no garantiza un mejor modelo, ya que las transformaciones aplicadas deben conservar los patrones visuales relevantes para la tarea. Bajo este criterio, considerando el desempeño en test, la estabilidad durante la validación y el grado de sobreajuste observado, ResNet-18 sin aumento de datos y EfficientNet-B3 con aumento de datos se consolidan como las configuraciones más relevantes para la clasificación de madurez del aguacate Hass.

Ahora bien, pasando al análisis de las matrices de confusión, estas permiten examinar con mayor detalle el comportamiento de cada modelo por clase. Este análisis se centró en las dos configuraciones con mejor desempeño general, ResNet-18 sin aumento de datos y EfficientNet-B3 con aumento de datos, las cuales alcanzaron resultados muy similares en el conjunto de prueba. Dado ese estrecho margen, resulta relevante comparar no solo sus métricas globales sino también el tipo de errores cometidos por cada una.

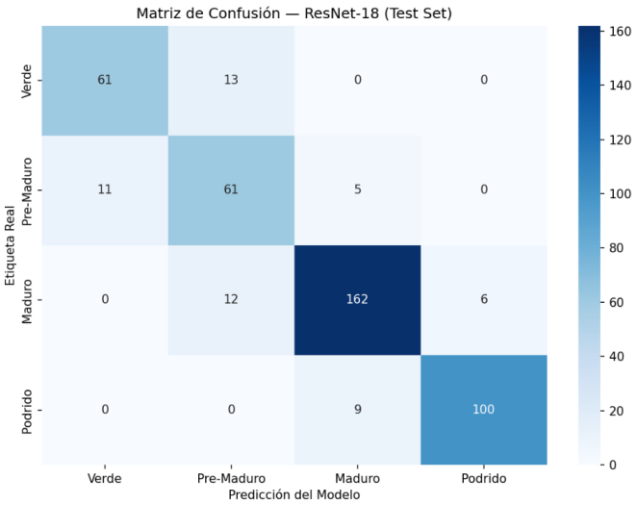


Fig. 28. Matriz de confusion ResNet-18 sin aumento de datos

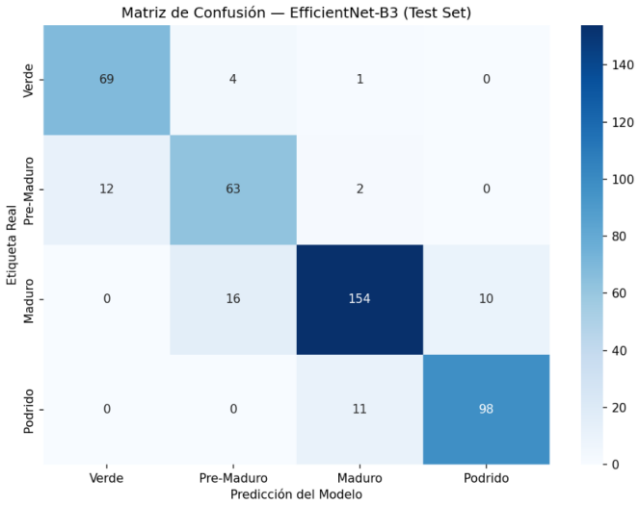


Fig. 29. Matriz de confusion EfficientNet-B3 con aumento de datos

En ResNet-18 sin aumento de datos (Fig. 28), los errores se concentraron principalmente entre clases adyacentes del proceso de maduración. En los estados iniciales, 13 imágenes verdes fueron clasificadas como Premaduro y 11 Pre-Maduras como Verde, lo que refleja la dificultad de separar etapas visualmente cercanas donde predominan tonalidades verdosas y cambios sutiles en la superficie del fruto. En los estados finales, la confusión fue menor, 6 imágenes de Maduro fueron clasificadas como Podrido y 10 de Podrido como

Maduro, lo que indica que, si bien persiste cierta ambigüedad entre estas clases, el modelo logró una separación razonablemente adecuada entre frutos aptos para consumo y frutos deteriorados. En términos generales, ResNet-18 mostró un patrón de error coherente con la naturaleza continua del proceso de maduración, cometiendo confusiones principalmente entre etapas contiguas y prácticamente ningún error entre clases distantes como Verde y Maduro o Verde y Podrido.

EfficientNet-B3 con aumento de datos (Fig. 29) presentó un comportamiento ligeramente superior en los estados iniciales, reduciendo los errores en la clase Verde respecto a ResNet-18, con solo 4 imágenes verdes clasificadas como Premaduro frente a las 13 de ResNet-18. Sin embargo, en los estados finales mostró una mayor dificultad para identificar correctamente la clase Podrido, con 16 imágenes podridas clasificadas como Maduro y 11 imágenes maduras clasificadas como Podrido. Este comportamiento indica que EfficientNet-B3 tendió a ser más conservador al asignar la etiqueta Podrido, lo cual reduce el riesgo de descartar erróneamente frutos aún consumibles, pero incrementa la probabilidad de clasificar frutos deteriorados como aptos.

La comparación entre ambos modelos revela que sus errores siguen una lógica visual coherente. Las confusiones se presentan casi exclusivamente entre clases adyacentes del proceso de maduración, como Verde–Premaduro y Maduro–Podrido, y prácticamente nunca entre estados extremos como Verde y Podrido. Este comportamiento es consistente con la naturaleza gradual de la maduración del aguacate Hass, donde las diferencias visuales entre etapas contiguas son sutiles. Las demás arquitecturas evaluadas como VGG16 y MobileNetV3, cuyos resultados se presentan en la misma sección 2 de los anexos, mostraron distribuciones de error similares, concentrando sus confusiones también en clases cercanas. Sin embargo, presentaron menor precisión global y un mayor número de errores en clases específicas. VGG16 alcanzó un desempeño competitivo, pero no logró igualar la estabilidad de ResNet-18 ni de EfficientNet-B3, mientras que MobileNetV3 obtuvo el desempeño más bajo entre las arquitecturas CNN evaluadas, con confusiones particularmente elevadas en la clase Maduro, donde una cantidad considerable de imágenes fue clasificada como Podrido en la configuración con aumento de datos.

En términos generales, las matrices de confusión evidencian que las CNN superaron a los modelos clásicos no solo en métricas globales, sino en la calidad de sus errores. Al aprender patrones directamente desde la imagen, sus confusiones fueron más coherentes con la progresión natural de la maduración que las de los modelos clásicos. No obstante, los resultados también dejan claro que tanto la selección de la arquitectura como el uso del aumento de datos deben analizarse cuidadosamente, ya que su impacto varía según el modelo.

Resultados de regresión con modelos CNN

Para la tarea de regresión, las arquitecturas CNN fueron adaptadas para predecir directamente el número de días restantes hasta el deterioro completo del aguacate Hass, por lo que la última capa de cada arquitectura fue reemplazada por una única neurona encargada de estimar los días de vida útil restantes. Cada arquitectura fue evaluada bajo las mismas dos configuraciones usadas en clasificación, con y sin aumento de datos, con el fin de analizar si las transformaciones aplicadas durante el entrenamiento favorecían también la predicción continua o si, por el contrario, afectaban la capacidad del modelo para estimar con precisión los días restantes.

Tabla 5. Resultados de regresión obtenidos por las arquitecturas CNN con y sin aumento de datos.

Arquitectura		Aumento de datos	MAE	SMAPE	RMSE	R^2	R^2 ajustado
ResNet-18		No	0.4174	62.42%	0.5718	0.9602	0.9601
ResNet-18		Sí	0.5066	66.06%	0.6949	0.9413	0.9411
EfficientNet-B3		No	0.4185	63.31%	0.5727	0.9601	0.9600
EfficientNet-B3		**Sí**	**0.4169**	**61.97%**	**0.5686**	**0.9607**	**0.9606**
VGG16		No	0.4672	65.02%	0.6392	0.9503	0.9502
VGG16		Sí	0.7139	70.45%	0.9160	0.8979	0.8977
MobileNetV3		No	0.4789	64.26%	0.6426	0.9498	0.9496
MobileNetV3		Sí	0.6426	67.71%	0.8361	0.9150	0.9148

A partir de los resultados mostrados en la Tabla 5, las arquitecturas CNN presentaron un desempeño alto en la predicción de vida útil. Las mejores configuraciones alcanzaron errores promedio inferiores a medio día y valores de R^2 cercanos a 0.96, lo cual indica que los modelos lograron capturar adecuadamente la relación entre la apariencia visual del aguacate y los días restantes hasta su deterioro. Este resultado es importante porque demuestra que las imágenes contienen información suficiente para estimar no solo una clase de madurez, sino también poder predecir en cuánto tiempo el fruto se pudriría.

El mejor desempeño fue obtenido por EfficientNet-B3 con aumento de datos, con un MAE de 0.4169 días, un RMSE de 0.5686 y un R^2 de 0.9607. El MAE indica que, en promedio, el modelo se equivocó menos de medio día al estimar la vida útil restante del aguacate, lo cual es un margen de error bajo considerando que el proceso de maduración se mide en días enteros. El RMSE de 0.5686, al ser solo ligeramente superior al MAE, sugiere que los errores atípicos fueron poco frecuentes y que el modelo mantuvo una predicción estable a lo largo del conjunto de prueba. El R^2 de 0.9607 indica que el modelo explicó aproximadamente el 96% de la variabilidad en los días de vida útil restantes, resultado que refleja que el modelo aprendió a estimar los días de vida útil a partir de la apariencia visual del aguacate con una precisión muy alta. De forma muy

cercana, ResNet-18 sin aumento de datos alcanzó un MAE de 0.4174 días, un RMSE de 0.5718 y un R^2 de 0.9602. La diferencia entre ambas configuraciones fue mínima en todas las métricas, por lo que ambos modelos pueden considerarse para usarse en esta tarea, aunque EfficientNet-B3 presentó unos resultados ligeramente superiores.

Frente a los modelos clásicos de regresión, las CNN lograron una mejora considerable. El mejor modelo clásico, SVR con 76 características, obtuvo un MAE de 0.6023 días y un R^2 de 0.9297, mientras que EfficientNet-B3 con aumento de datos redujo el error promedio a 0.4169 días y elevó el R^2 a 0.9607, lo que confirma que el aprendizaje profundo permitió construir una representación más completa del proceso de maduración al aprender directamente desde la imagen, sin depender de características de color, textura e histogramas.

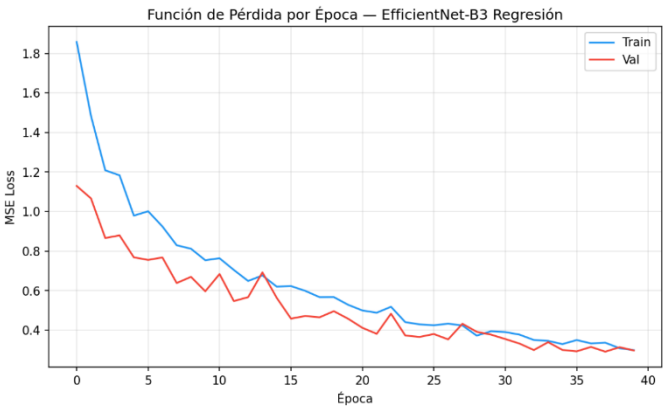


Fig. 30. Gráfica de Perdida MSE EfficientNet-B3 con Aumento de datos.

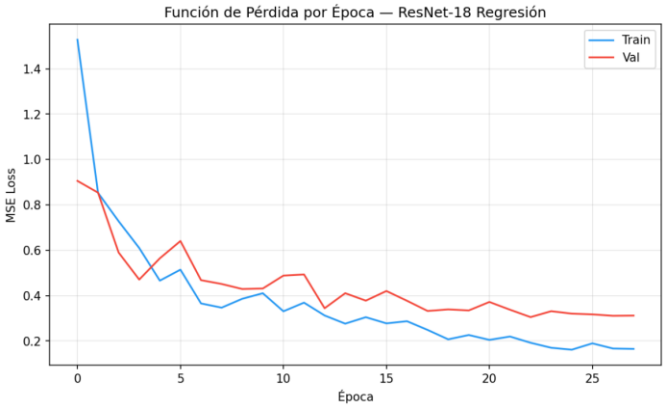


Fig. 31. Gráfica de Perdida MSE ResNet18 sin Aumento de datos.

Las curvas de pérdida por época complementan este análisis. Los modelos con mejor desempeño, presentados en la Fig. 30 y Fig. 31, presentaron una disminución progresiva y consistente tanto en la pérdida de entrenamiento como en la de validación, lo que indica que lograron aprender de forma estable la relación entre la apariencia visual del aguacate y los días de vida útil restantes. En contraste, las configuraciones con menor desempeño, siendo VGG16 y MobileNetV3 con aumento de datos, mostraron pérdidas de validación más altas o con mayor variabilidad a lo largo del entrenamiento, causando así una mayor dificultad para converger, aunque sin superar un loss de MSE de 0.3. Esto confirma que en la tarea de regresión no basta con reducir la pérdida en entrenamiento, sino que es igualmente necesario mantener una pérdida de validación baja y consistente para garantizar una estimación confiable. Las curvas del resto de arquitecturas se presentan en la sección 4 de los anexos.

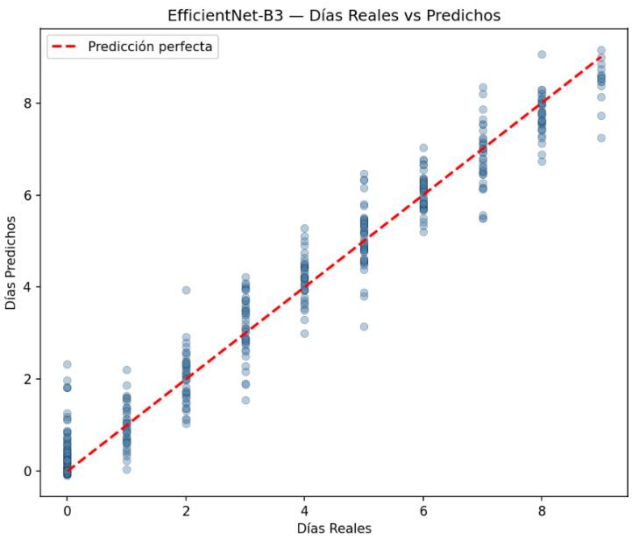


Fig. 32. Gráfica de dispersión de días EfficientNet-B3 con Aumento de datos.

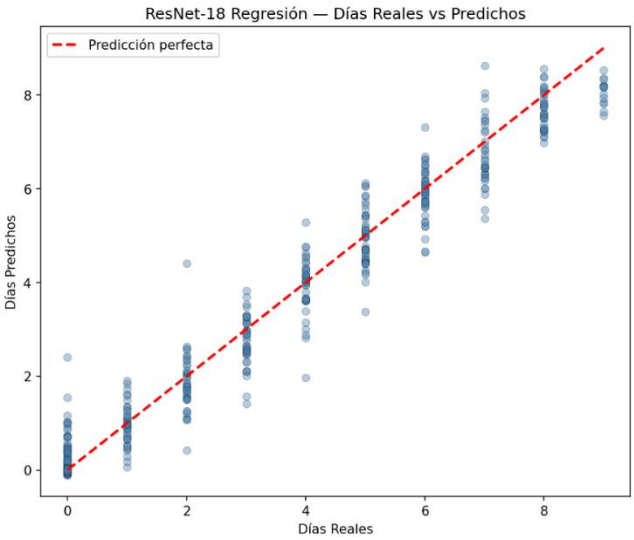


Fig. 33. Gráfica de dispersión de días ResNet18 sin Aumento de datos.

Las gráficas de días reales frente a días predichos permiten evaluar visualmente la precisión de cada modelo a través de la distribución de los puntos alrededor de la línea de predicción perfecta. EfficientNet-B3 con aumento de datos (Fig. 32) y ResNet-18 sin aumento de datos (Fig. 33), las dos configuraciones con menor MAE y mayor R^2, mostraron la mayoría de sus puntos distribuidos cerca de la diagonal a lo largo de todo el rango de días, lo que indica que ambas arquitecturas lograron aprender patrones visuales asociados al avance de la maduración. El buen comportamiento a lo largo de todo el rango confirma además que los modelos no se limitaron a predecir correctamente los valores extremos, sino que también aproximaron adecuadamente los estados intermedios del proceso de maduración.

No obstante, ambos modelos presentan desviaciones puntuales respecto a la diagonal, particularmente en valores cercanos a cero días y en algunos estados intermedios. Estas desviaciones son esperables dada la naturaleza del problema, ya que cuando un aguacate está a punto de deteriorarse, su apariencia visual es muy similar a la de un fruto que ya está completamente podrido, lo que dificulta que el modelo distinga con precisión cuántos días exactos le quedan, mientras que en los días intermedios dos aguacates pueden presentar una apariencia muy similar, pero corresponder a valores de vida útil distintos.

En conjunto, los resultados confirman que las mejores arquitecturas no solo obtuvieron buenos resultados numéricos, sino que también mostraron una relación visual consistente entre predicciones y valores reales. Tanto EfficientNet-B3 con aumento de datos como ResNet-18 sin aumento de datos lograron aproximar adecuadamente la vida útil restante del aguacate Hass, reforzando nuevamente la idea de que las CNN pueden aprender representaciones visuales útiles para estimar una variable continua, superando las limitaciones de los modelos clásicos.

3.3.3. Resultados de la experimentación con Fine-Tuning

Con el propósito de evaluar si permitirle al modelo ajustar sus pesos preentrenados mejoraba el desempeño en la clasificación de madurez del aguacate Hass, se llevaron a cabo experimentos de fine-tuning sobre las mismas cuatro arquitecturas trabajadas anteriormente. Para cada modelo se probaron cuatro configuraciones variando el porcentaje del backbone descongelado en la segunda etapa del entrenamiento siendo, backbone completamente congelado, donde el modelo actúa únicamente como extractor de características, y tres configuraciones adicionales en las que se liberó progresivamente el 25%, 50% y 75% de las capas del backbone. El objetivo principal fue determinar si descongelar parcialmente el backbone en la segunda etapa producía mejoras en accuracy, precisión o F1-score en comparación con mantenerlo completamente congelado o descongelarlo por completo.

Tabla 6. Resultados de las métricas de clasificación con diferentes porcentajes de descongelamiento en la segunda etapa de fine-tuning en los cuatro modelos.

Modelo	% Capas descongeladas	Accuracy	Precision	Recall	F1-score
EfficientNet-B3	0%	71.82%	0.7346	0.7182	0.7086
EfficientNet-B3	25%	83.41%	0.8373	0.8341	0.8331
EfficientNet-B3	50%	85.91%	0.8616	0.8591	0.8597
EfficientNet-B3	**75%**	**86.59%**	**0.8695**	**0.8659**	**0.8666**
ResNet-18	0%	73.41%	0.7461	0.7341	0.7301
ResNet-18	25%	81.59%	0.8151	0.8159	0.8141
ResNet-18	50%	78.18%	0.7911	0.7818	0.7786
ResNet-18	75%	81.14%	0.8104	0.8114	0.8085
VGG16	0%	73.86%	0.7464	0.7386	0.7398
VGG16	25%	80.45%	0.8099	0.8045	0.8039
VGG16	50%	84.32%	0.8441	0.8432	0.8392
VGG16	**75%**	**85.91%**	**0.8631**	**0.8591**	**0.8595**
MobileNetV3 Large	0%	71.14%	0.7348	0.7114	0.7023
MobileNetV3 Large	25%	82.50%	0.8286	0.8250	0.8253
MobileNetV3 Large	50%	84.77%	0.8499	0.8477	0.8483
MobileNetV3 Large	75%	83.41%	0.8365	0.8341	0.8347

Los resultados en la Tabla 6 muestran que el fine-tuning tuvo un impacto positivo en el desempeño de todos los modelos evaluados. En términos generales, se observa una mejora al pasar de un backbone completamente congelado a un 25% descongelado, lo que confirma que permitir que una parte de la red preentrenada se adapte al nuevo dominio es clave para capturar mejor las características visuales específicas del aguacate Hass. Este primer nivel de ajuste produjo incrementos notables en todas las métricas evaluadas.

Sin embargo, a medida que se incrementa el porcentaje de capas descongeladas, el comportamiento de los modelos deja de ser uniforme y depende ya de la arquitectura. EfficientNet-B3 y VGG16 mostraron una mejora progresiva conforme aumentaba el fine-tuning, alcanzando sus mejores resultados al 75% de capas descongeladas. Esto indica que estas arquitecturas tienen mayor capacidad para beneficiarse de un ajuste más profundo sin perder generalización, y la mejora se reflejó en métricas como en precisión, recall y F1-score, mostrando que no estuvo sesgada hacia ninguna clase en particular, sino que mejoró el reconocimiento equilibrado de todas las clases.

En contraste, ResNet-18 y MobileNetV3 Large presentaron un comportamiento menos estable. ResNet-18 alcanzó su mejor desempeño al 25% y comenzó a degradarse con niveles mayores de fine-tuning, lo que puede explicarse por su arquitectura más simple y menos profunda. MobileNetV3 Large, aunque más moderno, está diseñado para ser ligero y eficiente, lo cual hace que sus capas compactas sean más sensibles a ajustes profundos, posiblemente generando así inestabilidad cuando el fine-tuning es excesivo. Su mejor desempeño se alcanzó en configuraciones intermedias del 50%.

A grandes rasgos, estos resultados confirman que el fine-tuning es una estrategia a tener en cuenta para mejorar el desempeño de algunos de los modelos de aprendizaje profundo, pero su efectividad no depende de descongelar la mayor cantidad posible de capas, sino de encontrar el punto óptimo en el que el modelo logra adaptarse al problema sin comprometer su capacidad de generalización.

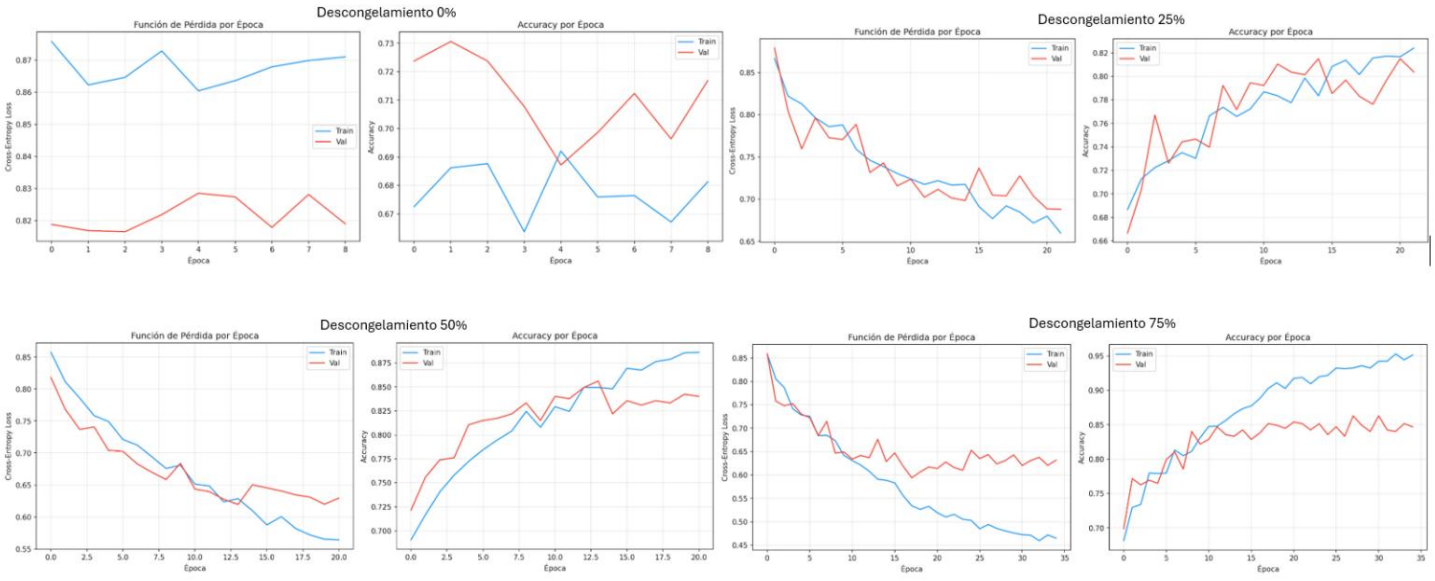


Fig. 34. Gráficas de Perdida y Accuracy por época de EfficientNet-B3 con distintos porcentajes de descongelamiento del backbone.

El análisis de las curvas de pérdida y accuracy de EfficientNet-B3, mostradas en la Fig. 34, evidencia que al incrementar el porcentaje de capas descongeladas el entrenamiento se vuelve un poco más estable, la pérdida disminuye consistentemente en ambos conjuntos y el accuracy de validación mejora progresivamente, manteniéndose cercano al de entrenamiento con una diferencia generalmente menor a 10 puntos porcentuales. Este comportamiento indica un sobreajuste moderado y controlado, lo que refleja un buen equilibrio entre aprendizaje y generalización. VGG16 presenta una tendencia similar, aunque con mejoras ligeramente menores. En conjunto, ambos modelos confirman que el fine-tuning progresivo permite mejorar el desempeño sin comprometer la estabilidad del entrenamiento.

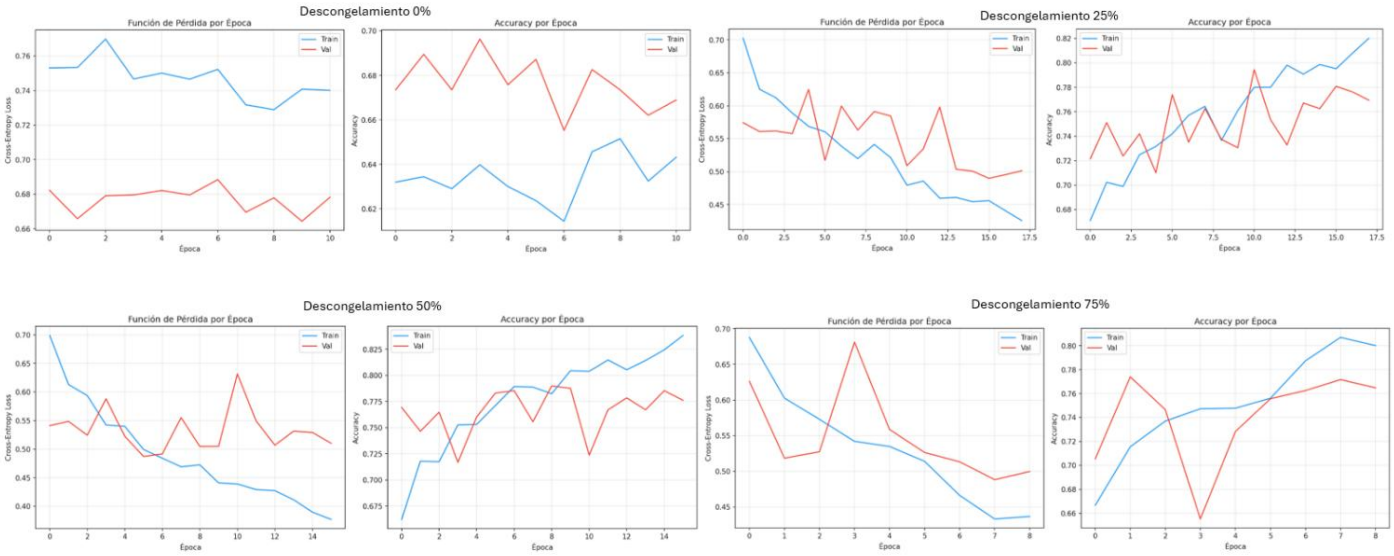


Fig. 35. Gráficas de Perdida y Accuracy por época de ResNet-18 con distintos porcentajes de descongelamiento del backbone.

Por otro lado, las gráficas de ResNet-18, mostradas en la Fig. 35, evidencian un comportamiento más inestable a medida que aumenta el grado de fine-tuning. Aunque la pérdida de entrenamiento disminuye de forma continua, la pérdida de validación presenta oscilaciones y el accuracy de validación no mantiene una mejora sostenida. La configuración con 25% de capas descongeladas muestra el mejor equilibrio entre aprendizaje y estabilidad, mientras que a partir del 50% comienzan a aparecer signos claros de sobreajuste, el modelo sigue mejorando en entrenamiento, pero no en validación, y la brecha entre ambas curvas se amplía. MobileNetV3 Large presenta una tendencia similar, aunque con menor grado de inestabilidad y más tardía, mejorando hasta el 50% y mostrando una leve caída al llegar al 75%, sus gráficas se incluyen en la sección 5 de los anexos. En ambos casos, la divergencia entre las curvas de entrenamiento y validación confirman que estos modelos no se benefician de un fine-tuning profundo, sino que alcanzan su mejor desempeño en configuraciones intermedias donde logran adaptarse al dominio sin perder capacidad de generalización, a

diferencia de EfficientNet-B3 y VGG16 que sí mostraron estabilidad y mejora progresiva con niveles más altos de ajuste.

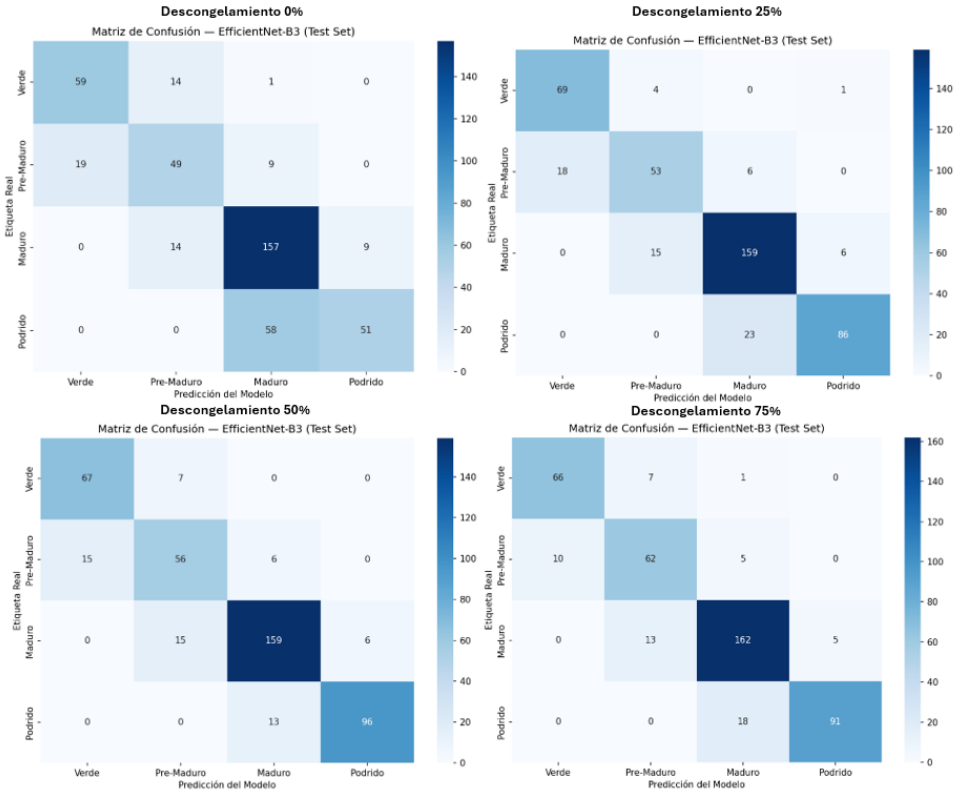


Fig. 36. Matriz de confusion EfficientNet-B3 con distintos porcentajes de descongelamiento del backbone.

El análisis de las matrices de confusión se centra en EfficientNet-B3 por ser el modelo con mejor desempeño global, mientras que las matrices de los demás modelos se incluyen en la sección 2 de los anexos. Como se aprecia en la Fig. 36, el fine-tuning mejora progresivamente la capacidad del modelo para diferenciar entre clases. En la configuración sin descongelamiento, el modelo presenta algunos errores, especialmente entre Maduro y Podrido, donde las clasificaciones incorrectas en esta última clase superan a las correctas, lo que refleja una baja capacidad discriminativa cuando el backbone permanece completamente congelado. Al aplicar el 25% de descongelamiento se observa una mejora en todas las clases. La más destacable es la reducción de errores en Podrido, lo cual sugiere que el modelo comienza a capturar mejor las características visuales del dominio. Esta tendencia continúa en el 50%, donde la diagonal de la matriz se vuelve más dominante, y se consolida en el 75%, configuración en la que se alcanza el mejor desempeño, con alta precisión en todas las clases y una disminución de errores respecto a las evaluaciones previas. En todas las configuraciones, los errores más frecuentes se presentan entre clases adyacentes, lo cual es esperable dado que el aguacate Hass no cambia de un estado a otro de forma repentina, sino mediante transiciones visuales graduales que dificultan la separación entre etapas cercanas.

En síntesis, los experimentos de fine-tuning confirmaron que el ajuste progresivo de las capas preentrenadas mejora el desempeño de los modelos, pero su efectividad depende de la arquitectura. EfficientNet-B3 y VGG16 se beneficiaron consistentemente de un mayor grado de descongelamiento, mejorando en todas las métricas de forma sostenida hasta alcanzar sus mejores resultados con el 75% de capas ajustadas, por lo que estas arquitecturas tienen mayor capacidad para adaptar sus representaciones internas al dominio del aguacate Hass sin perder estabilidad. ResNet-18 y MobileNetV3, en cambio, alcanzaron su mejor desempeño en configuraciones intermedias del 25% y 50%, y mostraron deterioro al aumentar el ajuste, evidenciando que un fine-tuning excesivo introduce inestabilidad y afecta su capacidad de generalización. Esto demuestra que un mayor porcentaje de capas descongeladas no garantiza mejores resultados, sino que existe un punto óptimo que varía según la profundidad y capacidad de cada arquitectura para absorber modificaciones sin perder el conocimiento adquirido durante el preentrenamiento.

3.3.4. Elección de los modelos para clasificación y regresión

A partir del proceso experimental desarrollado en esta sección, que abarcó modelos clásicos de aprendizaje automático, arquitecturas CNN con y sin aumento de datos, y estrategias de fine-tuning, se estableció el modelo final para las dos tareas planteadas en el proyecto. Tanto para la clasificación del estado de madurez como para la predicción de vida útil restante, EfficientNet-B3 con aumento de datos y con el 100% del backbone descongelado en la segunda etapa de fine-tuning se consolida como la mejor alternativa. La implementación en código de las arquitecturas seleccionadas se puede ver en la sección 6 de los anexos.

Para la tarea de clasificación, EfficientNet-B3 con aumento de datos y backbone completamente descongelado alcanzó un accuracy de 0.8727 y un F1-score de 0.8735, posicionándose entre los mejores modelos evaluados. Sin embargo, la elección no se fundamenta únicamente en estas métricas, sino en el comportamiento global del modelo durante el entrenamiento. Las gráficas de pérdida y accuracy por época (Fig. 25 y Fig. 30) muestran curvas de entrenamiento y validación cercanas entre sí, por lo que el modelo aprendió patrones relevantes sin memorizar los datos de entrenamiento. Este equilibrio es importante a tener en cuenta, porque refleja una buena capacidad de generalización. En contraste, ResNet-18, a pesar de alcanzar métricas similares, presentó un comportamiento menos estable, especialmente en la pérdida, con una separación más marcada entre las curvas de entrenamiento y validación que evidencia sobreajuste y reduce su confiabilidad en escenarios reales.

Por otra parte, la interpretación de las matrices de confusión de EfficientNet-B3, presentadas en las Fig. 29 y 36, refuerzan esta decisión. Los errores del modelo se concentran entre clases adyacentes como se ha venido mencionando, y se reducen progresivamente a medida que se descongelan más capas del backbone, lo cual es un patrón que debería de esperarse por parte del modelo, dado que la maduración del aguacate Hass

es un proceso gradual, y las diferencias visuales entre estados cercanos son ligeramente sutiles. Que el modelo falle precisamente en estas transiciones, y no entre estados extremos, evidencia que ha aprendido una representación coherente del fenómeno.

En la tarea de regresión, los resultados de la Tabla 5 confirman la superioridad de EfficientNet-B3 con aumento de datos, alcanzando un MAE de 0.4169 días y un R^2 de 0.9607. Un error promedio inferior a medio día es una precisión alta considerando la naturaleza del problema. Este desempeño se valida visualmente en la gráfica de dispersión de valores reales vs. predichos en la Fig. 32, donde la mayoría de los puntos se distribuyen muy cerca de la diagonal a lo largo de todo el rango de valores, lo cual confirma que el modelo mantiene una relación entre la apariencia visual del aguacate y su vida útil estimada, más allá de los resultados obtenidos por las métricas.

Adicionalmente, las curvas de pérdida (Fig. 30) muestran valores estables y cercanos entre sí para entrenamiento y validación, reflejando así un muy buen desempeño de generalización. Cabe destacar que este comportamiento no se limita a los extremos del proceso de maduración, sino que se extiende a los estados intermedios, siendo los más difíciles de predecir por la similitud visual entre muestras, lo cual indica que el modelo tiene buena capacidad para capturar patrones complejos asociados a la evolución temporal del fruto. Con base en estos resultados, EfficientNet-B3 se consolida como el modelo seleccionado para ambas tareas y el más viable para su integración en el desarrollo de la interfaz.

3.4. Desarrollo de la interfaz funcional

A continuación, se describe el proceso de diseño e implementación de la interfaz funcional desarrollada para el proyecto, denominado como AvoCheck. Esta interfaz tiene como propósito permitir a cualquier usuario poder tomar o adjuntar una fotografía de un aguacate Hass y obtener su estado de madurez y el tiempo estimado de vida útil.

La Fig. 37 presenta un diagrama general del sistema, en el cual se puede apreciar cómo los distintos componentes interactúan entre sí. El frontend se ejecuta localmente en una PC y es expuesto a través de un enlace público generado con ngrok, permitiendo así el acceso desde dispositivos externos. Luego, cuando un usuario carga una imagen desde la interfaz, esta es enviada al backend, el cual se encuentra desplegado en Google Colab y también es expuesto mediante un segundo enlace de ngrok. En el backend se realiza el procesamiento de la imagen usando los pesos de los modelos almacenados en Google Drive, y una vez obtenidas las predicciones, el backend retorna los resultados al frontend para que el usuario los visualice.

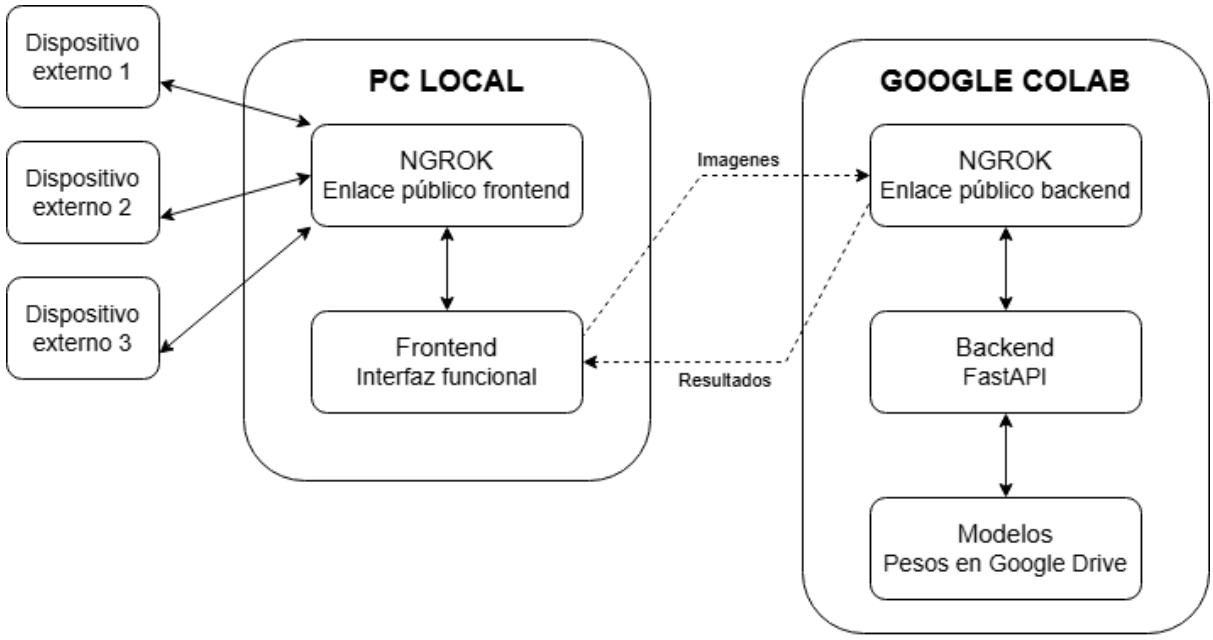


Fig. 37. Diagrama general de la arquitectura del sistema.

3.4.1. Diseño de la interfaz

Antes de comenzar con el desarrollo de la interfaz, se elaboró un diagrama de actividades (Fig. 38) que describe el flujo de interacción entre el usuario y el sistema, desde el momento en que accede a la web hasta que obtiene los resultados de la predicción. Este diagrama sirvió como guía para definir las funcionalidades necesarias y el orden lógico de las pantallas. A partir de él, se construyó un pequeño Mock Up (Fig. 39-40) que representa visualmente la apariencia de la interfaz tanto en dispositivos móviles como en computador.

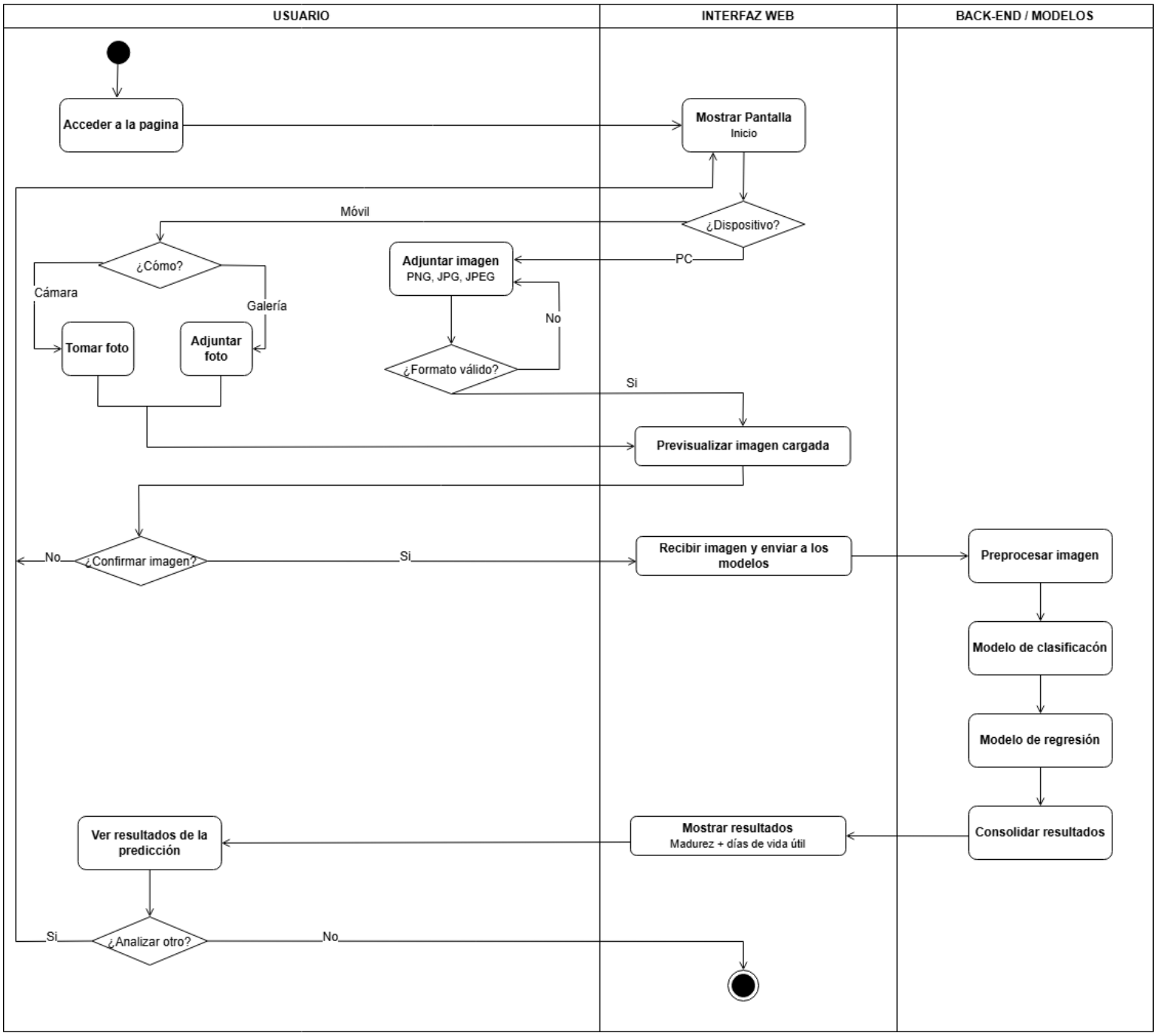


Fig. 38. Diagrama de actividades de AvoCheck. Flujo de interacción entre el usuario y el sistema.

La interfaz fue diseñada con dos pantallas principales. La primera es la pantalla de inicio, donde el usuario puede subir o tomar la fotografía del aguacate, cuyo diseño se puede apreciar en la Fig. 39 tanto en su versión móvil como en su versión de escritorio. La segunda es la pantalla de resultados, donde se muestra el estado de madurez clasificado, los días de vida útil restantes y el porcentaje de precisión de la predicción. Su diseño se muestra en la Fig. 40 en ambas versiones.

Fig. 39. Pantalla de inicio de AvoCheck en versión móvil y escritorio.

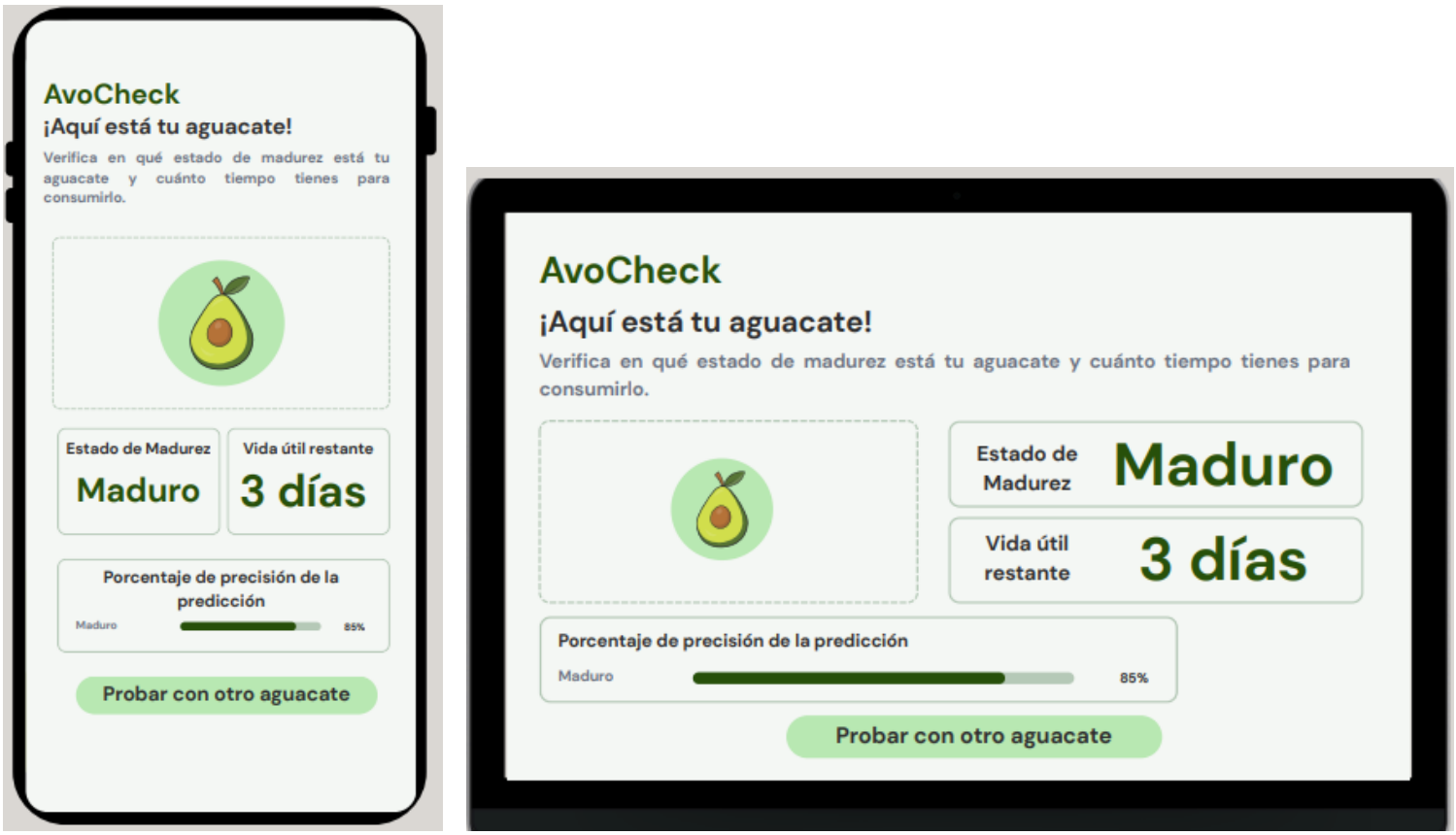


Fig. 40. Pantalla de resultados de AvoCheck en versión móvil y escritorio.

3.4.2. Estructura del sistema

La solución se estructuró en dos componentes, un frontend encargado de la interfaz del usuario, y un backend encargado de procesar las imágenes y ejecutar las predicciones de los modelos entrenados.

El frontend fue desarrollado con React y Vite, usando JavaScript como lenguaje principal y Tailwind CSS para los estilos. Con estas herramientas se construyó una interfaz responsiva que se adapta tanto a dispositivos móviles como a computadores.

El backend fue implementado con FastAPI, siendo un framework de Python para la construcción de APIs. Dado que los pesos de los modelos entrenados se encontraban almacenados en Google Drive, el backend fue desplegado directamente en Google Colab, lo que facilitó su carga y ejecución sin necesidad de un servidor externo.

Ahora, para que tanto la interfaz como el backend fueran accesibles desde cualquier punto de acceso de internet, se usó ngrok, una herramienta que genera enlaces públicos temporales hacia servicios que están corriendo localmente o en la nube. La razón de usar esta herramienta fue que inicialmente la interfaz se mostraba mediante la opción --host de Vite, la cual genera una dirección IP pública para que cualquier dispositivo pudiese acceder, sin embargo, se identificó que, en algunas redes públicas como por ejemplo en la universidad, bloqueaban el acceso a ese tipo de direcciones, así que por eso se utilizó ngrok para resolver ese problema. Cabe recalcar que se configuraron dos enlaces, uno para acceder a la interfaz desde cualquier dispositivo y otro para que la interfaz se pudiera comunicar con el backend y enviarle las imágenes a analizar.

3.4.3. Implementación del frontend

El frontend fue desarrollado como una aplicación de página única con React y Vite. La aplicación cuenta con dos rutas principales, siendo la página de inicio y la página de resultados. La URL del backend expuesta por ngrok se configura manualmente mediante una variable de entorno, lo que facilita actualizar el endpoint sin modificar el código fuente cada vez que se reinicia el servidor Colab.

En la pantalla de inicio, el usuario puede interactuar con una zona de carga que acepta imágenes en formato PNG, JPG, JPEG, ya sea arrastrándolas directamente, seleccionándolas desde el explorador de archivos o tomando una foto con la cámara del dispositivo móvil. Una vez ya seleccionada la imagen, ésta se muestra como vista previa y se habilita el botón de analizar el aguacate. Al presionarlo, la imagen se envía al backend mediante una solicitud HTTP POST y la aplicación navega automáticamente a la pantalla de resultados con la respuesta recibida.

En la pantalla de resultados se presenta de forma clara el estado de madurez predicho, los días de vida útil restantes y una barra de progreso que muestra el porcentaje de precisión del modelo para la clase predicha. Desde esta pantalla el usuario puede volver al inicio para analizar otro aguacate si lo necesita. El desarrollo del frontend se puede entender más a fondo en el repositorio que se compartió que está en la sección 6 de los anexos.

3.4.4. Implementación del backend

El backend se implementó como un servidor FastAPI ejecutado desde un notebook de Google Colab. Al iniciar, el servidor carga en memoria los pesos del modelo EfficientNet-B3 entrenado para clasificación y para regresión, ambos previamente guardados en Google Drive. Una vez cargados, el servidor queda a la espera de solicitudes.

El endpoint principal es POST/predecir, el cual recibe la imagen del aguacate enviada desde el frontend, aplica el mismo preprocesamiento usado durante el entrenamiento, siendo el redimensionamiento a 300x300 píxeles y la normalización con los valores de ImageNet, lo cual garantiza que la imagen llegue al modelo en el formato que ésta espera para producir las predicciones correctas. La imagen pasa por los dos modelos de forma secuencial. Primero, el modelo de clasificación devuelve la clase predicha junto con su porcentaje de precisión, y el modelo de regresión devuelve el número de días restantes de vida útil estimado. Ambos resultados se empaquetan en una respuesta que es enviada de vuelta al frontend para ser mostrada al usuario. La implementación de esta conexión con la interfaz se hizo en un Colab y se puede ver en la sección 6 de los anexos

3.4.5. Limitaciones del sistema

Si bien la interfaz funcional desarrollada cumple con el objetivo de permitir al usuario analizar un aguacate Hass de forma sencilla, es importante reconocer algunas limitaciones a tener en cuenta al momento de usar o evaluar el sistema.

En primer lugar, la interfaz no cuenta con un mecanismo de validación que verifique si la imagen enviada corresponde efectivamente a un aguacate Hass. Esto significa que, si el usuario sube una foto de cualquier objeto, el sistema igualmente procesará la imagen y entregará una predicción que carecerá de sentido. Por ejemplo, si se sube una foto de una manzana o incluso de un perro, el modelo de igual forma clasificará la imagen en una de las cuatro categorías de madurez como si fuese un aguacate. La responsabilidad de subir una imagen adecuada recae en el usuario.

En segundo lugar, el sistema depende de que la sesión de Google Colab esté activa y en ejecución. Si la sesión se cierra por inactividad, el backend deja de funcionar y la interfaz no puede procesar ninguna solicitud. Por ejemplo, si se intenta analizar un aguacate horas después de que el servidor fue iniciado y la sesión de Colab expiró, la aplicación mostrara un error de conexión sin poder entregar ningún resultado.

En tercer lugar, la calidad de la fotografía influye en la precisión de la predicción. El modelo fue entrenado con imágenes capturadas bajo condiciones controladas de iluminación y fondo uniforme, por lo tanto, una foto tomada de noche con la linterna del celular, con el aguacate sobre una superficie verde similar al color de su cáscara, o con el fruto cortado a la mitad del encuadre, puede llevar al modelo a dar resultados erróneos.

Finalmente, dado que el modelo fue entrenado exclusivamente con aguacates almacenados a temperatura ambiente, la predicción de días de vida útil puede no ser precisa si el aguacate analizado estuvo refrigerado. Por ejemplo, un aguacate que lleva varios días en la nevera puede verse visualmente como maduro, pero tener más días de vida útil de los que el modelo estimaría, ya que el frio ralentiza el proceso de maduración de una forma que el modelo no aprendió a reconocer.

4. CONCLUSIONES Y TRABAJOS FUTUROS

4.1. Conclusiones

El objetivo principal de este proyecto fue desarrollar un sistema capaz de clasificar el estado de madurez del aguacate Hass y predecir su tiempo de vida útil restante en días mediante técnicas de visión por computador y aprendizaje profundo. Este objetivo fue cumplido satisfactoriamente, logrando resultados positivos en ambas tareas a través de un proceso experimental que abarcó desde la preparación del conjunto de datos hasta el despliegue de una interfaz funcional.

El proceso experimental permitió comparar dos enfoques, modelos clásicos de aprendizaje automático basados en características extraídas mediante técnicas de procesamiento de imágenes, y arquitecturas CNN preentrenadas adaptadas mediante aprendizaje por transferencia y fine-tuning. Los resultados confirmaron que las redes convolucionales superaron a los modelos clásicos en ambas tareas, no solo por alcanzar mejores métricas, sino porque sus errores fueron más coherentes con la naturaleza gradual del proceso de maduración del aguacate Hass, concentrándose principalmente entre clases adyacentes y nunca entre estados extremos. Este comportamiento evidencia que las CNN lograron aprender representaciones visuales más ricas directamente desde las imágenes, superando las limitaciones de los métodos de extracción de características basados en procesamiento de imágenes. Dentro de las arquitecturas evaluadas, los experimentos de fine-tuning mostraron que descongelar progresivamente el backbone preentrenado mejoró el desempeño, pero su efectividad dependió de cada arquitectura. EfficientNet-B3 se benefició de descongelar una mayor proporción del backbone, mejorando sus métricas de forma progresiva hasta alcanzar su mejor desempeño con el 75% de capas ajustadas, mostrando curvas de entrenamiento y validación estables y cercanas entre sí, lo que refleja un buen equilibrio entre aprendizaje y generalización. En contraste, ResNet-18, aunque alcanzó métricas similares en clasificación, presentó un comportamiento menos estable con mayor sobreajuste, lo que reduce su confiabilidad en escenarios reales.

En la tarea de regresión, EfficientNet-B3 también demostró superioridad, estimando la vida útil restante con un error promedio inferior a medio día y explicando el 96% de la variabilidad en los días restantes, resultado que confirma que la apariencia visual del aguacate contiene información suficiente para predecir no solo su estado de madurez sino también cuántos días le quedan antes de deteriorarse completamente. Con base en todos estos resultados, EfficientNet-B3 con aumento de datos y el 100% del backbone descongelado en la segunda etapa de fine-tuning se consolidó como el modelo final para ambas tareas.

Los modelos seleccionados fueron integrados en AvoCheck, una interfaz web funcional desarrollada con React y Vite, que permite a cualquier usuario subir o tomar una fotografía de un aguacate Hass y obtener de forma inmediata su estado de madurez y los días de vida útil restantes. Esta solución cumple con el alcance planteado al inicio del proyecto y demuestra que es posible construir una herramienta práctica y accesible que apoye la toma de decisiones sobre el estado del aguacate Hass, contribuyendo a la posibilidad de reducir el desperdicio en la cadena de distribución y comercialización en el contexto colombiano.

4.2. Trabajos Futuros

Uno de los trabajos futuros que se puede realizar sobre este proyecto es la construcción de una base de datos propia y más robusta. Aunque el conjunto de datos usado fue útil y está bien estructurado, fue recolectado en condiciones controladas de laboratorio, con aguacates fotografiados sobre fondo blanco e iluminación uniforme. Una base de datos futura podría incluir imágenes capturadas en condiciones más reales, con variaciones de iluminación, fondos diversos, diferentes ángulos y distintas condiciones de captura. Ampliar la cantidad de frutos documentados más allá de los 478 aguacates originales permitiría mejorar la capacidad de generalización de los modelos y hacerlos más representativos del contexto real. También sería interesante incorporar aguacates de otras variedades además del Hass, para enriquecer la diversidad del conjunto y abrir la posibilidad de desarrollar sistemas que trabajen con múltiples tipos de fruto.

Desde el punto de vista del despliegue, la interfaz AvoCheck desarrollada en este proyecto es un prototipo funcional con varias limitaciones, ya que depende de una sesión activa en Google Colab y no incluye validación de la imagen de entrada. Un trabajo futuro relevante sería migrar el backend a una infraestructura de nube más estable, como AWS, Google Cloud o Azure, que permita mantener el sistema activo de forma continua y soportar múltiples usuarios simultáneamente. Además, implementar un módulo de validación que verifique que la imagen subida corresponde efectivamente a un aguacate antes de procesarla, posiblemente mediante un clasificador binario, eliminaría una de las principales limitaciones del sistema actual. Con estas mejoras, el sistema podría evolucionar hacia una herramienta de uso real en entornos como supermercados, distribuidoras o puntos de venta, donde el personal podría fotografiar los aguacates directamente desde sus dispositivos para obtener en segundos una estimación de su estado de madurez y vida útil restante, facilitando la toma de decisiones sobre la rotación de inventario o la reducción de pérdidas por deterioro.

Finalmente, este trabajo puede servir como punto de partida para otros investigadores y desarrolladores que quieran abordar problemas similares. El proceso está documentado con suficiente detalle como para replicarlo, adaptarlo o extenderlo sin partir desde cero. Los experimentos comparativos realizados sobre las cuatro arquitecturas con distintos porcentajes de descongelamiento constituyen una referencia útil que no siempre está disponible en la literatura con este nivel de detalle. Quienes quieran retomar el proyecto directamente podrían concentrarse en las limitaciones que quedaron abiertas, como mejorar la validación de entrada en la interfaz, reemplazar el esquema de despliegue actual, evaluar si los modelos generalizan correctamente ante imágenes capturadas fuera de las condiciones controladas del dataset, y explorar su extensión hacia otros tipos de frutos en el contexto de la visión por computador aplicada a la agricultura.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] Gustavsson, J. & Cederberg, Christel & Sonesson, Ulf & Otterdijk, R. & Meybeck, Alexandre. (2011). Global Food Losses and Food Waste- Extent, Causes and Prevention. Disponible en: https://www.researchgate.net/publication/285683189_Global_Food_Losses_and_Food_Waste-_Extent_Causes_and_Prevention

[2] Herrera-González, Juan & Bautista-Baños, Silvia & Salazar-Garcia, Samuel & Gutierrez-Martinez, Porfirio. (2020). Situación actual del manejo poscosecha y de enfermedades fungosas del aguacate 'Hass' para exportación en Michoacán. Revista Mexicana de Ciencias Agrícolas. 11. 1647-1660. Disponible en: https://www.researchgate.net/publication/345390068_Situacion_actual_del_manejo_poscosecha_y_de_enfermedades_fungosas_del_aguacate_'Hass'_para_exportacion_en_Michoacan

[3] Ministerio de Agricultura y Desarrollo Rural - Colombia (2021). Estadísticas de producción agrícola. Sistema de Información de Gestión y Desempeño de Organizaciones de Cadenas (SIOC). Disponible en: https://sioc.minagricultura.gov.co/Aguacate/Pages/default.aspx

[4] Sandra Landahl, Leon A. Terry. Non-destructive discrimination of avocado fruit ripeness using laser Doppler vibrometry. Biosystems Engineering. Volume 194. (2020). ISSN 1537-5110. Disponible en: https://doi.org/10.1016/j.biosystemseng.2020.04.001.

[5] Andreas Kamilaris, Francesc X. Prenafeta-Boldú. Deep learning in agriculture: A survey. Computers and Electronics in Agriculture. Volume 147. (2018). ISSN 0168-1699. Disponible en: https://doi.org/10.1016/j.compag.2018.02.016.

[6] Shahzaib, M. RipeTrack: assessing fruit ripeness and remaining lifetime using smartphones. (Thesis) M.Sc. National University of Sciences and Technology. Islamabad, Pakistán. (2024). Disponible en: https://summit.sfu.ca/item/38514

[7] M. Dreher and A. Davenport, "Hass Avocado Composition and Potential Health Effects," Critical Reviews in Food Science and Nutrition, vol. 53, no. 7, pp. 738-750, 2013. Disponible en: https://www.tandfonline.com/doi/full/10.1080/10408398.2011.556759

[8] E. Hershkovitz et al., "The role of the embryo and ethylene in avocado fruit mesocarp discoloration," Journal of Experimental Botany, vol. 60, no. 3, pp. 791-799, 2009. Disponible en: https://pmc.ncbi.nlm.nih.gov/articles/PMC2652053/

[9] Mattheis, J. (USDA/ARS Tree Fruit Research Laboratory). "Fruit Maturity and Quality", (2016). Disponible en: https://s3-us-west-2.amazonaws.com/treefruit.wsu.edu/wp-content/uploads/2016/01/15-Mattheis_Fruit-Maturity-Quality.pdf?

[10] Vázquez Carranza, Ariel. (2023). Materiality, Wittgenstein, and avocados. Sensorial inspections in commercial interactions. 19. 56-95. 10.24275/sling.v19n37.03. Disponible en: https://www.researchgate.net/figure/Espectro-de-los-estados-de-madurez-del-aguacate-Hass_fig1_383323541

[11] S. V. Mahadevkar et al., "A Review on Machine Learning Styles in Computer Vision—Techniques and Future Directions," in IEEE Access, vol. 10, pp. 107293-107329, 2022, doi: 10.1109/ACCESS.2022.3209825. Disponible en: https://ieeexplore.ieee.org/document/9903420

[12] R. C. Gonzalez y R. E. Woods, *Digital Image Processing*, 4th ed., Pearson, (2018).

[13] H. Talebi y P. Milanfar, "Learning to Resize Images for Computer Vision Tasks," en *Proc. IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. Disponible en: https://arxiv.org/abs/2103.09950

[14] M. Afsharpour, T. Zoughi, M. Deypir y S. Zoqi, "Robust Deep Learning Method for Fruit Decay Detection and Plant Identification: Enhancing Food Security and Quality Control," *Frontiers in Plant Science*, vol. 15, 2024. doi: 10.3389/fpls.2024.1366395. Disponible en: https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2024.1366395/full

[15] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *Journal of Big Data*, (2019). Disponible en: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0

[16] X. Qin, Z. Zhang, C. Huang, M. Dehghan, O. R. Zaiane y M. Jagersand, "U2-Net: Going Deeper with Nested U-Structure for Salient Object Detection," *Pattern Recognition*, vol. 106, p. 107404, 2020. doi: 10.1016/j.patcog.2020.107404.
Disponible en: https://www.sciencedirect.com/science/article/abs/pii/S0031320320302077

[17] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16,2002. Disponible en: https://www.jair.org/index.php/jair/article/view/10302

[18] M. Muratbekova et al., "Color Models in Image Processing: A Review and Experimental Comparison," *arXiv preprint*, arXiv:2510.00584, 2025. Disponible en: https://arxiv.org/abs/2510.00584

[19] Pardede, Jasman & Husada, Milda & Hermana, Asep & Rumapea, Sri. (2019). Fruit Ripeness Based on RGB, HSV, HSL, L*a*b* Color Feature Using SVM. 1-5. 10.1109/ICoSNIKOM48755.2019.9111486. Disponible en: https://www.researchgate.net/publication/342058112_Fruit_Ripeness_Based_on_RGB_HSV_HSL_Lab_Color_Feature_Using_SVM

[20] A. Lazaro, M. Boada, R. Villarino y D. Girbau, "Color Measurement and Analysis of Fruit with a Battery-Less NFC Sensor," *Sensors*, vol. 19, no. 7, p. 1741, 2019. doi: 10.3390/s19071741. Disponible en: https://pmc.ncbi.nlm.nih.gov/articles/PMC6479777/

[21] J. F. Reyes, E. Contreras, C. Correa y P. Melin, "Image analysis of real-time classification of cherry fruit from colour features," *Journal of Agricultural Engineering*, vol. LII, 2021. doi: 10.4081/jae.2021.1160. Disponible en: https://www.researchgate.net/publication/357305353

[22] A. R. Zubair y O. Alo, "Grey Level Co-occurrence Matrix (GLCM) Based Second Order Statistics for Image Texture Analysis," *arXiv preprint*, arXiv:2403.04038, 2024. Disponible en: https://arxiv.org/abs/2403.04038

[23] N. Iqbal, R. Mumtaz, U. Shafi y S. M. H. Zaidi, "Gray level co-occurrence matrix (GLCM) texture based crop classification using low altitude remote sensing platforms," *PeerJ Computer Science*, vol. 7, e536, 2021. doi: 10.7717/peerj-cs.536. Disponible en: https://peerj.com/articles/cs-536/

[24] D. Zhang, G. Lu, et al., "Classification of Fruits Using Computer Vision and a Multiclass Support Vector Machine," *Advances in Multimedia*, vol. 2012, pp. 1–13, 2012. Disponible en: https://pmc.ncbi.nlm.nih.gov/articles/PMC3478854/

[25] X. Font. Técnicas de clasificación supervised learning, Material didáctico Analítica de Datos - Módulo 4, Universitat Oberta de Catalunya, Barcelona, España, (2019). Disponible en: https://openaccess.uoc.edu/bitstream/10609/147174/9/AnaliticaDeDatos_Modulo4_TecnicasDeClasificacionSupervisedLearning.pdf

[26] R. Szeliski, *Computer Vision: Algorithms and Applications*, 2.ª ed. Cham, Suiza: Springer Nature Switzerland AG, 2022.

[27] K. Ikeuchi, Ed., *Computer Vision: A Reference Guide*, 2.ª ed. Cham, Suiza: Springer Nature Switzerland AG, 2021. Disponible en: https://link.springer.com/referencework/10.1007/978-3-030-63416-2

[28] Han J, Kamber M, Pei J. Data mining: concepts and techniques. New York: Elsevier; 2011. Disponible en: https://books.google.com.co/books?hl=es&lr=&id=NR1oEAAAQBAJ&oi=fnd&pg=PP1&ots=_N6LRLyio2&sig=dTF7ADANyHvGrob6breMhQBwaHI&redir_esc=y#v=onepage&q&f=false

[29] A. W. Harley, K. G. Derpanis and I. Kokkinos, "Segmentation-Aware Convolutional Networks Using Local Attention Masks," *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Italy, (2017). Disponible en : https://ieeexplore.ieee.org/document/8237800

[30] X. Zhao, L. Wang, Y. Zhang, X. Han y M. Deveci, "A review of convolutional neural networks in computer vision," *Artificial Intelligence Review*, vol. 57, art. 99, 2024. doi: 10.1007/s10462-024-10721-6. Disponible en: https://link.springer.com/article/10.1007/s10462-024-10721-6

[31] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, y H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv*, 2017. Disponible en: https://arxiv.org/abs/1704.04861

[32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, y L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," en *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018. Disponible en: https://arxiv.org/abs/1801.04381

[33] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, Q. V. Le, y H. Adam, "Searching for MobileNetV3," en *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, 2019. Disponible en: https://arxiv.org/abs/1905.02244

[34] T. Shahriar, "Comparative Analysis of Lightweight Deep Learning Models for Memory-Constrained Devices, Sep. 2025. Disponible en: https://arxiv.org/pdf/2505.03303

[35] H. Qassim, D. Feinzimer y A. Verma, "Residual Squeeze VGG16", *arXiv preprint arXiv*, 2017. Disponible en: https://arxiv.org/abs/1705.03004

[36] K. Simonyan y A. Zisserman, "Very deep convolutional networks for large-scale image recognition", en *International Conference on Learning Representations (ICLR)*, San Diego, CA, EE. UU. 2015. Disponible en: https://arxiv.org/abs/1409.1556

[37] M. Tan y Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks", en *Proceedings of the 36th International Conference on Machine Learning*, Long Beach, California, PMLR 97, 2019. Disponible en: https://arxiv.org/pdf/1905.11946

[38] H. Ali, N. Shifa, R. Benlamri, A. A. Farooque y R. Yaqub, "A fine tuned EfficientNet-B0 convolutional neural network for accurate and efficient classification of apple leaf diseases", *Scientific Reports*, 2025. Disponible en: https://pmc.ncbi.nlm.nih.gov/articles/PMC12267516

[39] Alhichri, Haikel & Alsuwayed, Asma & Bazi, Yakoub & Ammour, Nassim & Alajlan, Naif. (2021). Classification of Remote Sensing Images Using EfficientNet-B3 CNN Model with Attention. IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3051085. Disponible en: https://www.researchgate.net/publication/348470984_Classification_of_Remote_Sensing_Images_Using_EfficientNet-B3_CNN_Model_with_Attention

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition", 2015, arXiv:1512.03385. Disponible en: https://arxiv.org/abs/1512.03385

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Identity Mappings in Deep Residual Networks", 2016, arXiv:1603.05027v3. Disponible en: https://arxiv.org/abs/1603.05027

[42] S. A. Hasanah, A. A. Pravitasari, A. S. Abdullah, I. N. Yulita, and M. H. Asnawi, "A Deep Learning Review of ResNet Architecture for Lung Disease Identification in CXR Image," *Appl Sci*, vol 13, 2023. Disponible en: https://www.mdpi.com/2076-3417/13/24/13111

[43] Wang, Ping & Tseng, Hsien-Wei & Chen, Tzu-Ching & Hsia, Chih-Hsien. "Deep Convolutional Neural Network for Coffee Bean Inspection", 2021.

[44] N. Abou Baker, N. Zengeler y U. Handmann, "A Transfer Learning Evaluation of Deep Neural Networks for Image Classification," *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 22–41, 2022. doi: 10.3390/make4010002. Disponible en: https://www.mdpi.com/2504-4990/4/1/2

[45] C. O. Ukwuoma et al., "Recent Advancements in Fruit Detection and Classification Using Deep Learning Techniques," *Mathematical Problems in Engineering*, vol. 2022, 2022. doi: 10.1155/2022/9210947. Disponible en: https://onlinelibrary.wiley.com/doi/10.1155/2022/9210947

[46] Y. Gulzar, "Fruit Image Classification Model Based on MobileNetV2 with Deep Transfer Learning Technique," *Sustainability*, vol. 15, no. 3, p. 1906, 2023. doi: 10.3390/su15031906. Disponible en: https://www.mdpi.com/2071-1050/15/3/1906

[47] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2014. Disponible en: https://arxiv.org/abs/1411.1792

[48] R. J. Hyndman and A. B. Koehler, “Another look at measures of forecast accuracy,” *International Journal of Forecasting*, vol. 22, no. 4, pp. 679–688, 2006. doi: 10.1016/j.ijforecast.2006.03.001. Disponible en: https://doi.org/10.1016/j.ijforecast.2006.03.001

[49] C. J. Willmott and K. Matsuura, “Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance,” *Climate Research*, vol. 30 , 2005. Disponible: https://www.int-res.com/abstracts/cr/v30/p79-82/

[50] A. Fetanat, M. Tayebi y S. Ahmadi, "A critical review on selecting performance evaluation metrics for supervised machine learning models in wastewater quality prediction," *Journal of Environmental Chemical Engineering*, vol. 13, no. 6, p. 115843, 2025. doi: 10.1016/j.jece.2025.115843. Disponible en: https://www.sciencedirect.com/science/article/pii/S2213343725043714

[51] T. Chai and R. R. Draxler, “Root mean square error (RMSE) or mean absolute error (MAE)?—Arguments against avoiding RMSE,” *Geoscientific Model Development*, vol. 7, 2014. Disponible en: https://gmd.copernicus.org/articles/7/1247/2014/

[52] D. Chicco, M. J. Warrens y G. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," *PeerJ Computer Science*, vol. 7, e623, 2021. doi: 10.7717/peerj-cs.623. Disponible en: https://pmc.ncbi.nlm.nih.gov/articles/PMC8279135/

[53] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Information Processing & Management*, vol. 45, no. 4,2009. Disponible en: https://doi.org/10.1016/j.ipm.2009.03.002

[54] S. Pushpa y K. Bhargavi, "Logistic Regression Based Human Activities Recognition," *International Journal of Recent Technology and Engineering (IJRTE)*, vol. 8, no. 6, pp. 4353–4357, 2020. doi: 10.35940/ijrte.F7754.038620. Disponible en: https://www.researchgate.net/publication/341106657_LOGISTIC_REGRESSION_BASED_HUMAN_ACTIVITIES_RECOGNITION

[55] L. Cabrera-Sánchez et al., "Evaluation metrics and statistical tests for machine learning," *Scientific Reports*, vol. 14, p. 6086, 2024. doi: 10.1038/s41598-024-56706-x. Disponible en: https://pmc.ncbi.nlm.nih.gov/articles/PMC10937649/

[56] T. Aljedaani et al., "Performance Metrics for Multilabel Emotion Classification: Comparing Micro, Macro, and Weighted F1-Scores," *Applied Sciences*, vol. 14, no. 21, p. 9863, 2024. doi: 10.3390/app14219863. Disponible en: https://www.mdpi.com/2076-3417/14/21/9863

[57] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Information Processing & Management*, vol. 45, no. 4,2009. Disponible en: https://doi.org/10.1016/j.ipm.2009.03.002

[58] M. Darwish, *Fruit Classification Using Convolutional Neural Networks*, M.S. Thesis, Near East University, Nicosia, (2020). Disponible en: https://docs.neu.edu.tr/library/6916765672.pdf

[59] Nandan. G. Thor, *Using Computer Vision Techniques to Build a Predictive Model of Fruit Shelf-Life*, M.S. Thesis, California Polytechnic State University, (2017). Disponible en: https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=2947&context=theses

[60] Shahzaib, M. RipeTrack: assessing fruit ripeness and remaining lifetime using smartphones. (Thesis) M.Sc. National University of Sciences and Technology. Islamabad, Pakistán, (2024). Disponible en: https://summit.sfu.ca/item/38514

[61] Xavier, Pedro; Rodrigues, Pedro; L. M. Silva, Cristina (2024), "'Hass' Avocado Ripening Photographic Dataset", Mendeley Data, V1, doi: 10.17632/3xd9n945v8.1. Disponible en: https://data.mendeley.com/datasets/3xd9n945v8/1

[62] TorchVision Contributors, "resnet18 — Torchvision documentation", PyTorch, 2024. Disponible en: https://docs.pytorch.org/vision/main/models/generated/torchvision.models.resnet18.html

[63] TorchVision Contributors, "vgg16 — Torchvision documentation", PyTorch, 2024. Disponible en: https://docs.pytorch.org/vision/main/models/generated/torchvision.models.vgg16.html

[64] TorchVision Contributors, "mobilenet_v3_large — Torchvision documentation", PyTorch, 2024. Disponible en: https://docs.pytorch.org/vision/main/models/generated/torchvision.models.mobilenet_v3_large.html

[65] TorchVision Contributors, "efficientnet_b3 — Torchvision documentation", PyTorch, 2024. Disponible en: https://docs.pytorch.org/vision/main/models/generated/torchvision.models.efficientnet_b3.html

[66] K. Crisannaufal y W. F. Al Maki, "Optimizing Support Vector Machine for Avocado Ripeness Classification Using Moth Flame Optimization," *Journal of Electronics, Electromedical Engineering, and Medical Informatics*, vol. 7, no. 2, pp. 330–340, 2025. doi: 10.35882/jeeemi.v7i2.652. Disponible en: https://jeeemi.org/index.php/jeeemi/article/download/652/227

[67] I.-H. Lee, Z. Li y L. Ma, "Explainable AI and mobile imaging for non-destructive avocado ripeness and internal quality assessment to reduce food waste," *Current Research in Food Science*, vol. 11, p. 101196, 2025. doi: 10.1016/j.crfs.2025.101196. Disponible en: https://pmc.ncbi.nlm.nih.gov/articles/PMC12476098/

[68] M. A. Taha et al., "Accurate prediction of avocado ripeness using image processing and machine learning models (multi-layer perceptron, support vector regression, and K-nearest neighbour)," *International Journal of Food Science and Technology*, vol. 61, no. 1, vvag005, 2025. doi: 10.1093/ijfst/vvag005. Disponible en: https://academic.oup.com/ijfst/article/61/1/vvag005/8425003

[69] A. M. A. Efendi, Sriani y M. S. Hasibuan, "Classification of Watermelon Ripeness Levels Using HSV Color Space Transformation and K-Nearest Neighbor Method," *Journal of Computer Networks, Architecture and High Performance Computing*, vol. 6, no. 3, pp. 934–940, Jul. 2024. doi: 10.47709/cnapc.v6i3.3999. Disponible en: https://jurnal.itscience.org/index.php/CNAPC/article/view/3999

[70] P. Xavier, P. Rodrigues y C. L. M. Silva, "Shelf-Life Management and Ripening Assessment of 'Hass' Avocado (*Persea americana*) Using Deep Learning Approaches," *Foods*, vol. 13, no. 8, p. 1150, 2024. doi: 10.3390/foods13081150. Disponible en: https://www.mdpi.com/2304-8158/13/8/1150

[71] S. Nuanmeesri, "Utilization of Multi-Channel Hybrid Deep Neural Networks for Avocado Ripeness Classification," *Engineering, Technology & Applied Science Research*, vol. 14, no. 4, pp. 14862–14867, 2024. doi: 10.48084/etasr.7651. Disponible en: https://etasr.com/index.php/ETASR/article/view/7651

[72] S. Nuanmeesri, "Enhanced hybrid attention deep learning for avocado ripeness classification on resource constrained devices," *Scientific Reports*, vol. 15, p. 3719, 2025. doi: 10.1038/s41598-025-87173-7. Disponible en: https://pmc.ncbi.nlm.nih.gov/articles/PMC11779919/

[73] Scikit-learn developers, "Support Vector Machines," Scikit-learn documentation, 2024. Disponible en: https://scikit-learn.org/stable/modules/svm.html

[74] Scikit-learn developers, "sklearn.model_selection.GridSearchCV," Scikit-learn documentation, 2024. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[75] Scikit-learn developers, "RBF SVM parameters," Scikit-learn documentation, 2024. Disponible en: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html

[76] Scikit-learn Contributors, "RandomForestClassifier — scikit-learn documentation," Scikit-learn, 2024. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[77] Scikit-learn Contributors, "KNeighborsClassifier — scikit-learn documentation," Scikit-learn, 2024. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[78] N. V. Chawla, K. W. Bowyer, L. O. Hall y W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002. doi: 10.1613/jair.953. Disponible en: https://www.jair.org/index.php/jair/article/view/10302

[79] Scikit-learn Contributors, "MLPClassifier — scikit-learn documentation," Scikit-learn, 2024. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

[80] Scikit-learn developers, "sklearn.svm.SVR," Scikit-learn documentation, 2024. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

[81] Scikit-learn Contributors, "RandomForestRegressor — scikit-learn documentation," Scikit-learn, 2024. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[82] Scikit-learn Contributors, "KNeighborsRegressor — scikit-learn documentation," Scikit-learn, 2024. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsRegressor.html

[83] Scikit-learn Contributors, "MLPRegressor — scikit-learn documentation," Scikit-learn, 2024. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

ÍNDICE DE ANEXOS

1. Tabla de características extraídas. 93
2. Matrices de Confusión 96
2.1. Matrices de confusión de los modelos clásicos de clasificación de Machine Learning. 96
2.2. Matrices de confusión de los modelos de redes neuronales convolucionales de clasificación. 97
2.3. Matrices de confusión de los modelos de redes neuronales convolucionales de clasificación en el experimento de Fine-Tuning. 98
3. Gráficas de dispersión 101
3.1. Gráficas de dispersión de los modelos clásicos de regresión de Machine Learning. 101
3.2. Gráficas de dispersión de los modelos de redes neuronales convolucionales de regresión. 102
4. Gráficas de pérdida de modelos de redes neuronales convolucionales. 103
5. Gráficas de pérdida de modelos de redes neuronales convolucionales en el experimento de Fine-Tuning. 105
6. Recursos del proyecto 107

6. ANEXOS

1. Tabla de características extraídas.

Tabla que presenta la lista completa de las 76 características extraídas de cada imagen para los modelos clásicos de aprendizaje de máquina, organizadas por grupo y espacio de color.

#	Nombre	Grupo	Espacio / Canal	Descripción
1	mean_R	Estadística de color	RGB - Canal R	Media de intensidad del canal rojo
2	mean_G	Estadística de color	RGB - Canal G	Media de intensidad del canal verde
3	mean_B	Estadística de color	RGB - Canal B	Media de intensidad del canal azul
4	std_R	Estadística de color	RGB - Canal R	Desviación estandar del canal rojo
5	std_G	Estadística de color	RGB - Canal G	Desviación estandar del canal verde
6	std_B	Estadística de color	RGB - Canal B	Desviación estandar del canal azul
7	rad_R	Estadística de color	RGB - Canales R/G	Balance relativo entre verde y rojo
8	rad_B	Estadística de color	RGB - Canales B/G	Balance relativo entre verde y azul
9	mean_H	Estadística de color	HSV - Canal H	Media de la tonalidad del aguacate
10	mean_S	Estadística de color	HSV - Canal S	Media de la saturación del color
11	mean_V	Estadística de color	HSV - Canal V	Media del brillo general
12	std_V	Estadística de color	HSV - Canal V	Desviación estandar del brillo
13	mean_L	Estadística de color	CIELab - Canal L*	Media de la luminosidad
14	std_L	Estadística de color	CIELab - Canal L*	Desviacion estandar de la luminosidad
15	mean_a	Estadística de color	CIELab - Canal a*	Media del eje verde-rojo
16	std_a	Estadística de color	CIELab - Canal a*	Desviación estandar del eje verde-rojo
17	glcm_asm	Textura GLCM	Escala de Grises	Mide uniformidad de la textura
18	glcm_contast	Textura GLCM	Escala de Grises	Variación de intensidad entre píxeles vecinos
19	glcm_correlation	Textura GLCM	Escala de Grises	Patrón de continuidad entre píxeles vecinos
20	glcm_homoge	Textura GLCM	Escala de Grises	Similitud entre píxeles vecinos
21	glcm_entropy	Textura GLCM	Escala de Grises	Desorden de la textura
22	glcm_R_asm	Textura GLCM	RGB - Canal R	ASM sobre canal rojo
23	glcm_R_contrast	Textura GLCM	RGB - Canal R	Constraste sobre canal rojo

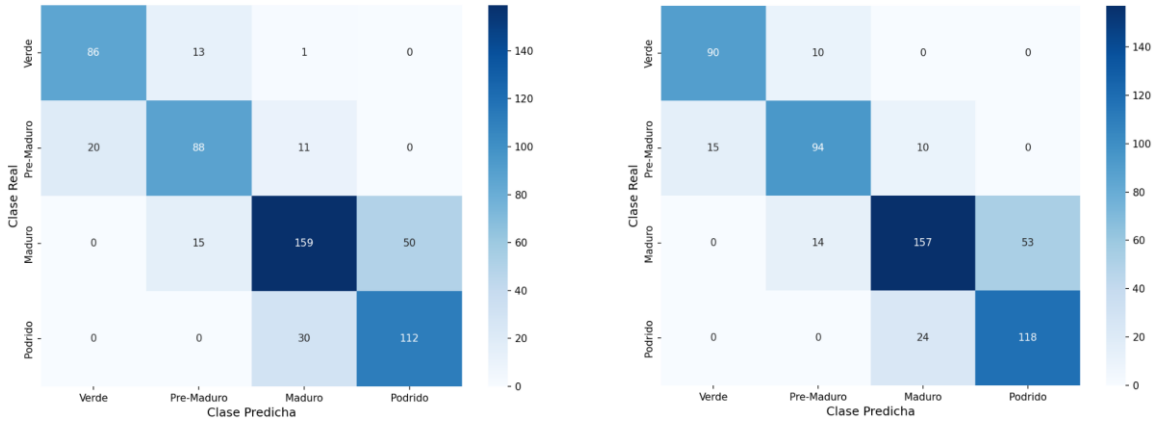
24	glcm_R_correlation	Textura GLCM	RGB - Canal R	Correlación sobre canal rojo
25	glcm_R_homoge	Textura GLCM	RGB - Canal R	Homogeneidad sobre canal rojo
26	glcm_R_entropy	Textura GLCM	RGB - Canal R	Entropía sobre canal rojo
27	glcm_G_asm	Textura GLCM	RGB - Canal G	ASM sobre canal verde
28	glcm_G_contrast	Textura GLCM	RGB - Canal G	Constraste sobre canal verde
29	glcm_G_correlation	Textura GLCM	RGB - Canal G	Correlación sobre canal verde
30	glcm_G_homoge	Textura GLCM	RGB - Canal G	Homogeneidad sobre canal verde
31	glcm_G_entropy	Textura GLCM	RGB - Canal G	Entropía sobre canal verde
32	glcm_B_asm	Textura GLCM	RGB - Canal B	ASM sobre canal azul
33	glcm_B_contrast	Textura GLCM	RGB - Canal B	Constraste sobre canal azul
34	glcm_B_correlation	Textura GLCM	RGB - Canal B	Correlación sobre canal azul
35	glcm_B_homoge	Textura GLCM	RGB - Canal B	Homogeneidad sobre canal azul
36	glcm_B_entropy	Textura GLCM	RGB - Canal B	Entropía sobre canal azul
37	glcm_H_asm	Textura GLCM	HSV - Canal H	ASM sobre canal de tonalidad
38	glcm_H_contrast	Textura GLCM	HSV - Canal H	Constraste sobre canal de tonalidad
39	glcm_H_correlation	Textura GLCM	HSV - Canal H	Correlación sobre canal de tonalidad
40	glcm_H_homoge	Textura GLCM	HSV - Canal H	Homogeneidad sobre canal de tonalidad
41	glcm_H_entropy	Textura GLCM	HSV - Canal H	Entropía sobre canal de tonalidad
42	glcm_S_asm	Textura GLCM	HSV - Canal S	ASM sobre canal de saturación
43	glcm_S_contrast	Textura GLCM	HSV - Canal S	Constraste sobre canal de saturación
44	glcm_S_correlation	Textura GLCM	HSV - Canal S	Correlación sobre canal de saturación
45	glcm_S_homoge	Textura GLCM	HSV - Canal S	Homogeneidad sobre canal de saturación
46	glcm_S_entropy	Textura GLCM	HSV - Canal S	Entropía sobre canal de saturación
47	glcm_V_asm	Textura GLCM	HSV - Canal V	ASM sobre canal de brillo
48	glcm_V_contrast	Textura GLCM	HSV - Canal V	Constraste sobre canal de brillo
49	glcm_V_correlation	Textura GLCM	HSV - Canal V	Correlación sobre canal de brillo
50	glcm_V_homoge	Textura GLCM	HSV - Canal V	Homogeneidad sobre canal de brillo
51	glcm_V_entropy	Textura GLCM	HSV - Canal V	Entropía sobre canal de brillo

52	glcm_L_asm	Textura GLCM	CIELab - Canal L*	ASM sobre canal de luminosidad
53	glcm_L_contrast	Textura GLCM	CIELab - Canal L*	Constraste sobre canal de luminosidad
54	glcm_L_correlation	Textura GLCM	CIELab - Canal L*	Correlación sobre canal de luminosidad
55	glcm_L_homoge	Textura GLCM	CIELab - Canal L*	Homogeneidad sobre canal de luminosidad
56	glcm_L_entropy	Textura GLCM	CIELab - Canal L*	Entropía sobre canal de luminosidad
57	glcm_a_asm	Textura GLCM	CIELab - Canal a*	ASM sobre el eje verde-rojo
58	glcm_a_contrast	Textura GLCM	CIELab - Canal a*	Constraste sobre el eje verde-rojo
59	glcm_a_correlation	Textura GLCM	CIELab - Canal a*	Correlación sobre el eje verde-rojo
60	glcm_a_homoge	Textura GLCM	CIELab - Canal a*	Homogeneidad sobre el eje verde-rojo
61	glcm_a_entropy	Textura GLCM	CIELab - Canal a*	Entropía sobre el eje verde-rojo
62	glcm_b_asm	Textura GLCM	CIELab - Canal b*	ASM sobre el eje azul-amarillo
63	glcm_b_contrast	Textura GLCM	CIELab - Canal b*	Constraste sobre el eje azul-amarillo
64	glcm_b_correlation	Textura GLCM	CIELab - Canal b*	Correlación sobreel eje azul-amarillo
65	glcm_b_homoge	Textura GLCM	CIELab - Canal b*	Homogeneidad sobre el eje azul-amarillo
66	glcm_b_entropy	Textura GLCM	CIELab - Canal b*	Entropía sobre el eje azul-amarillo
67	hist_H_0_9	Histograma	HSV - Canal H	Proporción de píxeles con tono entre 0-9
68	hist_H_9_18	Histograma	HSV - Canal H	Proporción de píxeles con tono entre 9-18
69	hist_H_18_27	Histograma	HSV - Canal H	Proporción de píxeles con tono entre 18-27
70	hist_H_27_36	Histograma	HSV - Canal H	Proporción de píxeles con tono entre 27-36
71	hist_H_36_45	Histograma	HSV - Canal H	Proporción de píxeles con tono entre 36-45
72	hist_H_45_54	Histograma	HSV - Canal H	Proporción de píxeles con tono entre 45-54
73	hist_H_144_153	Histograma	HSV - Canal H	Proporción de píxeles con tono entre 144-153
74	hist_H_153_162	Histograma	HSV - Canal H	Proporción de píxeles con tono entre 153-162
75	hist_H_162_171	Histograma	HSV - Canal H	Proporción de píxeles con tono entre 162-171
76	hist_H_171_180	Histograma	HSV - Canal H	Proporción de píxeles con tono entre 171-180

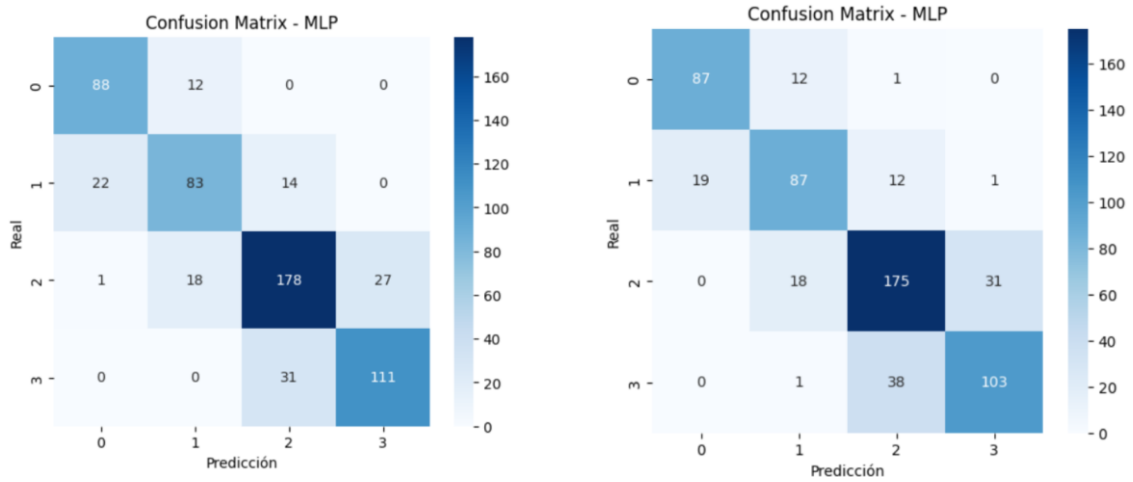
2. Matrices de Confusión

2.1. Matrices de confusión de los modelos clásicos de clasificación de Machine Learning.

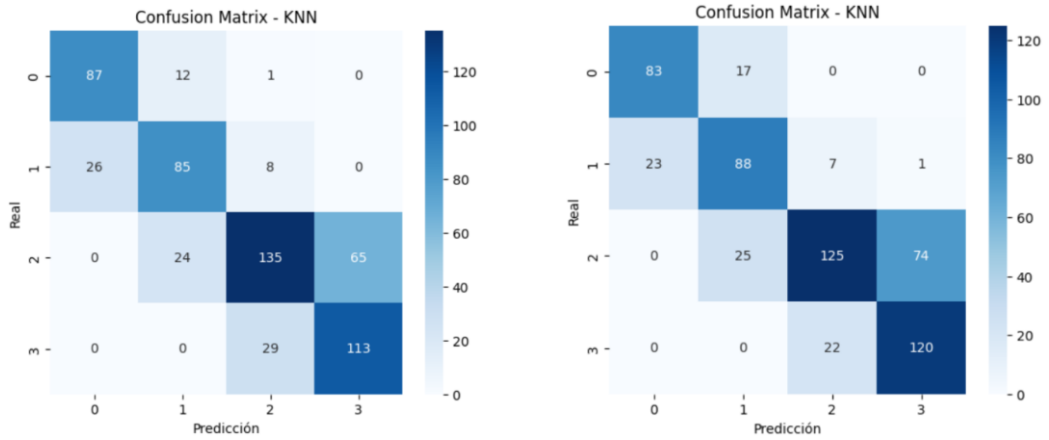
- Matrices de confusión de Random Forest con 31 características (izquierda) y 76 características (derecha).



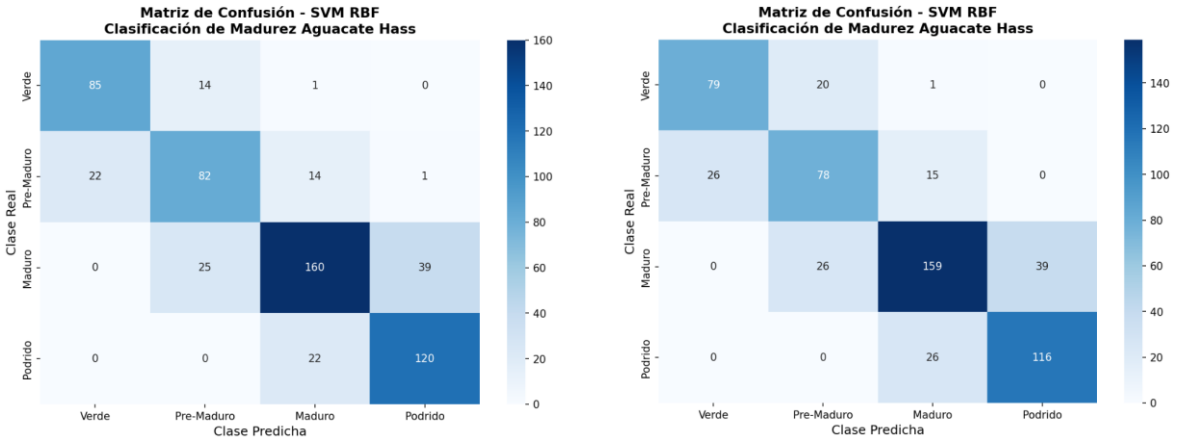
- Matrices de confusión de MLP con 31 características (izquierda) y 76 características (derecha).



- Matrices de confusión de KNN con 31 características (izquierda) y 76 características (derecha).

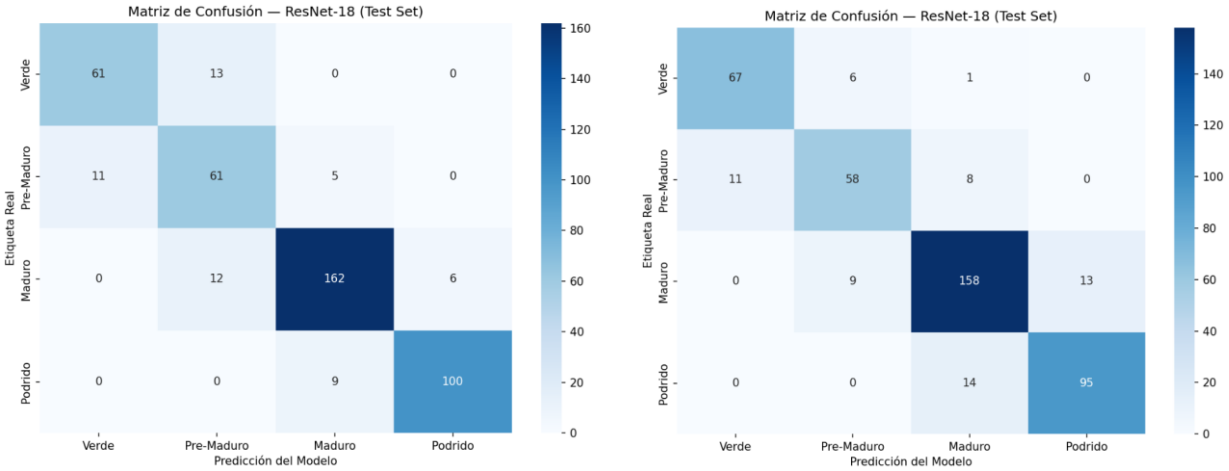


- Matrices de confusión de SVM RBF con 31 características (izquierda) y 76 características (derecha).

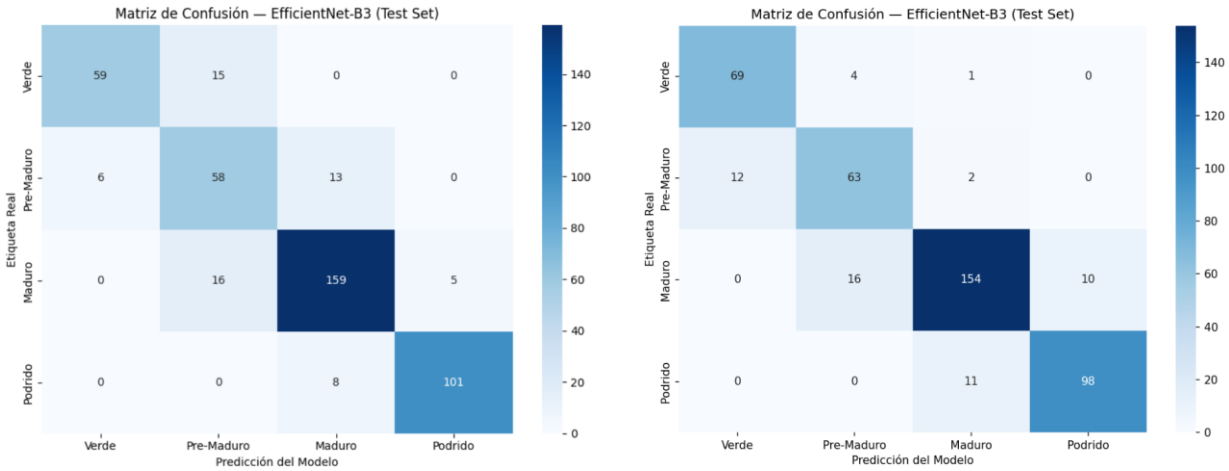


2.2. Matrices de confusión de los modelos de redes neuronales convolucionales de clasificación.

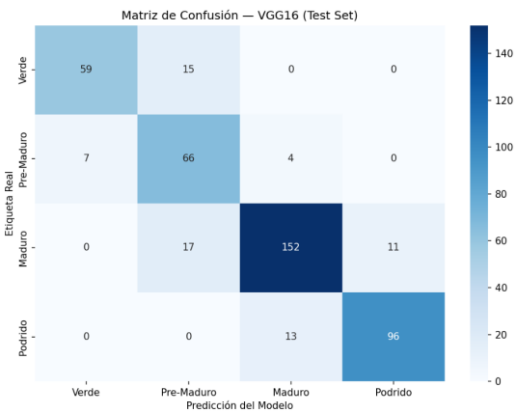
- Matrices de confusión de ResNet-18 sin (izquierda) y con aumento de datos (derecha).

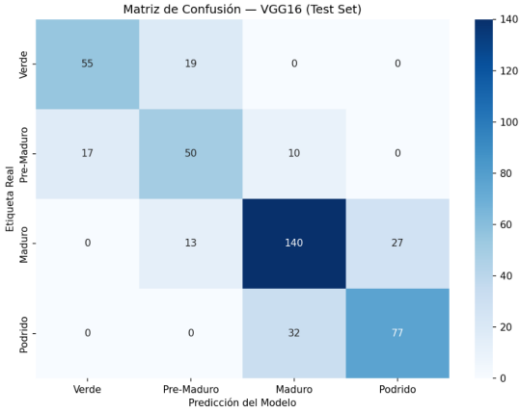


- Matrices de confusión de EfficientNet-B3 sin (izquierda) y con aumento de datos (derecha).

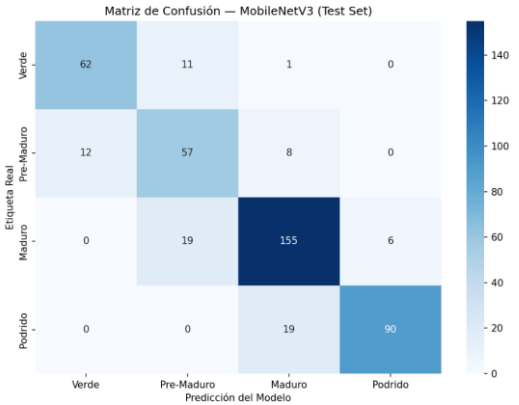


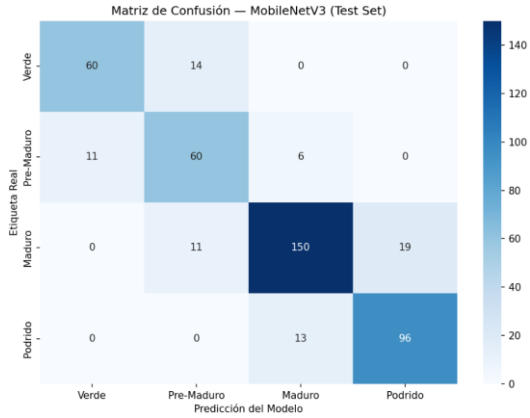
- Matrices de confusión de VGG16 sin (izquierda) y con aumento de datos (derecha).





- Matrices de confusión de MobileNetV3 sin (izquierda) y con aumento de datos (derecha).

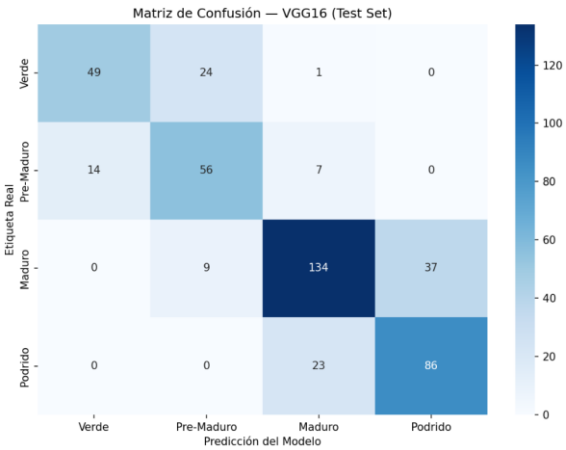




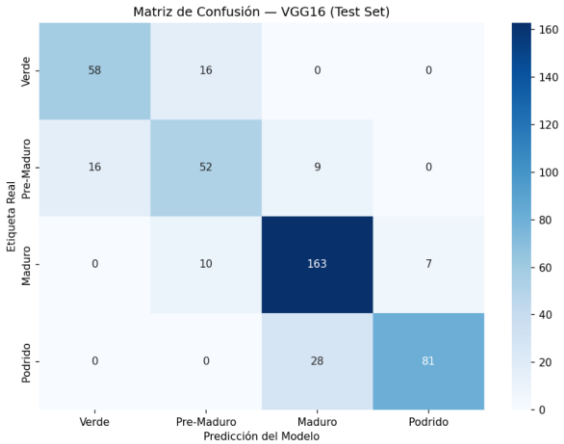
2.3. Matrices de confusión de los modelos de redes neuronales convolucionales de clasificación en el experimento de Fine-Tuning.

- Matrices de confusión de VGG16 en distintos porcentajes de descongelamiento del backbone.

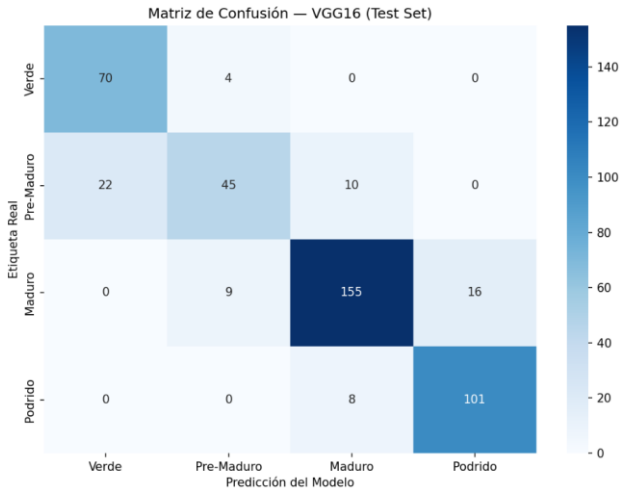
a) 0% de descongelamiento



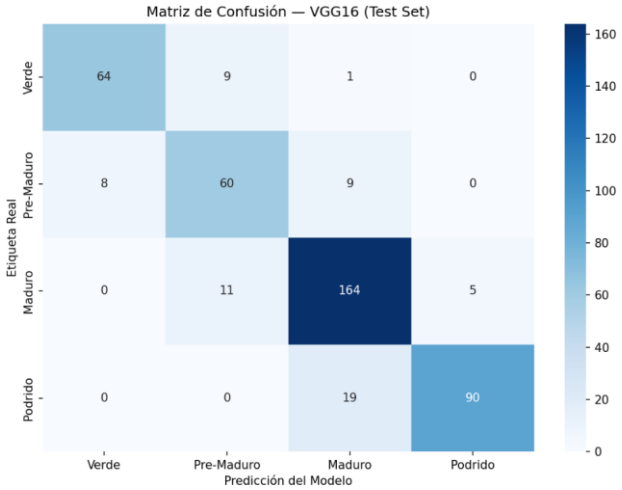
b) 25% de descongelamiento



c) 50% de descongelamiento

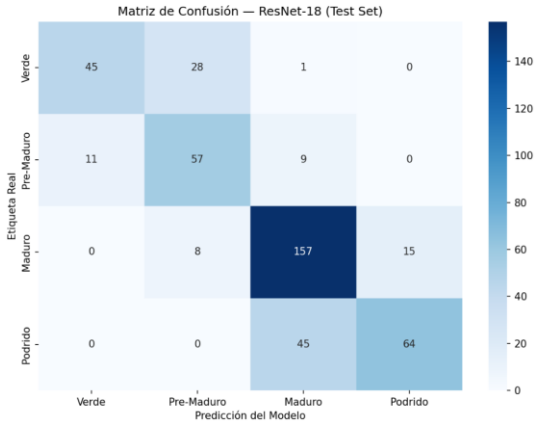


d) 75% de descongelamiento

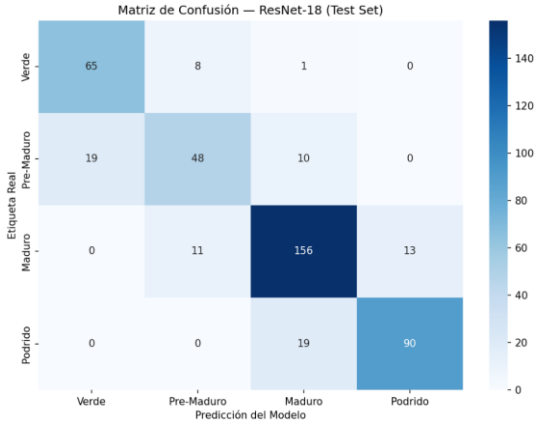


- Matrices de confusión de ResNet-18 en distintos porcentajes de descongelamiento del backbone.

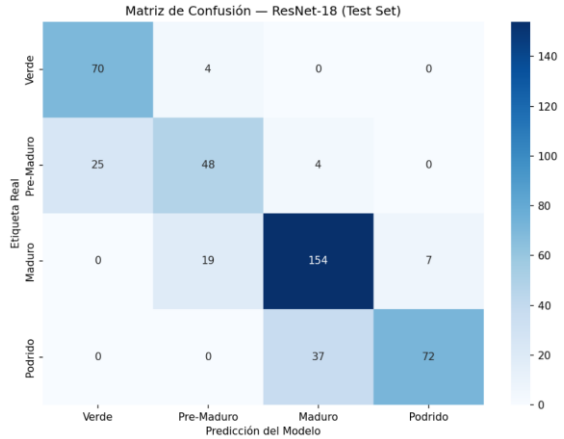
a) 0% de descongelamiento



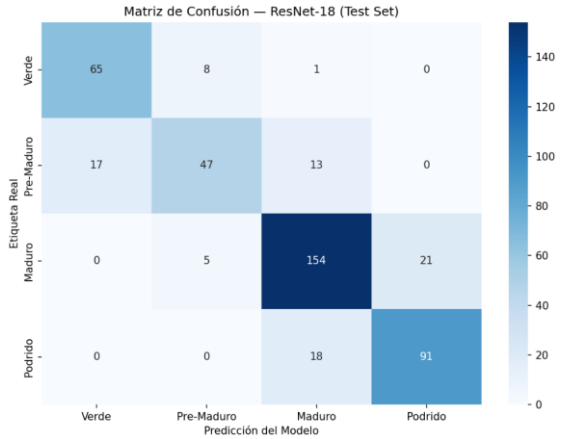
b) 25% de descongelamiento



c) 50% de descongelamiento

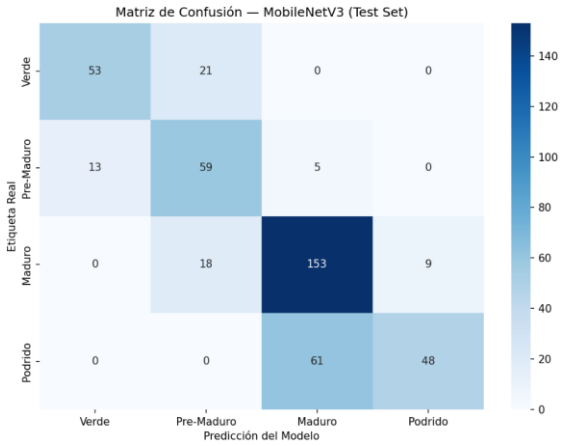


d) 75% de descongelamiento

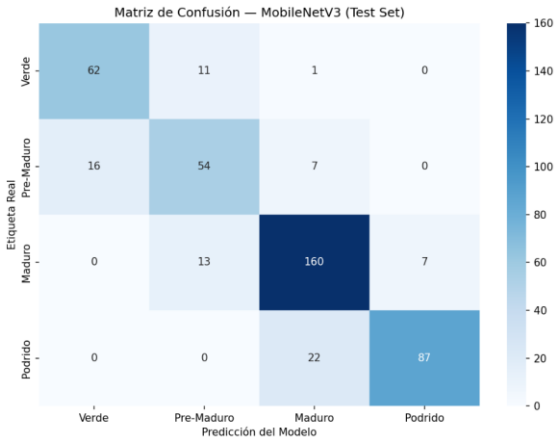


- Matrices de confusión de MobileNetV3 en distintos porcentajes de descongelamiento del backbone.

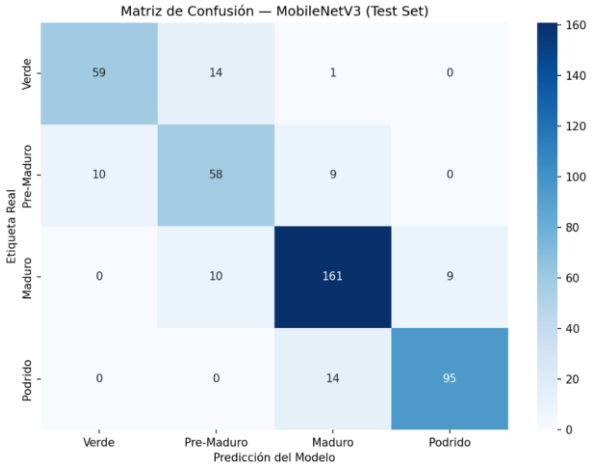
a) 0% de descongelamiento



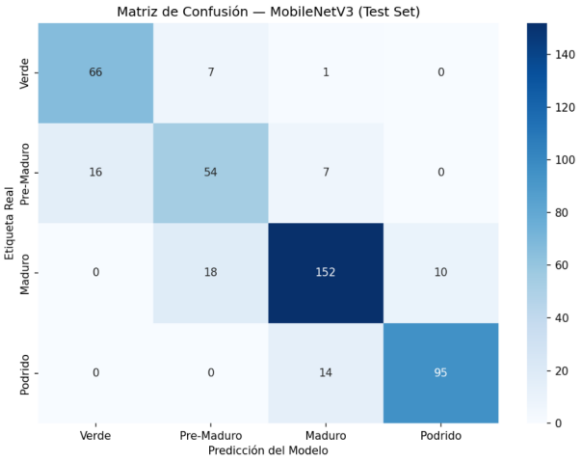
b) 25% de descongelamiento



c) 50% de descongelamiento



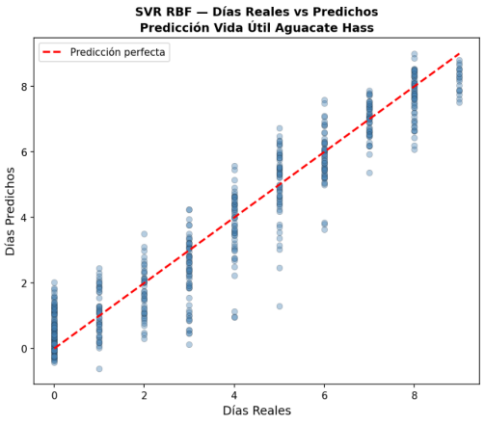
d) 75% de descongelamiento

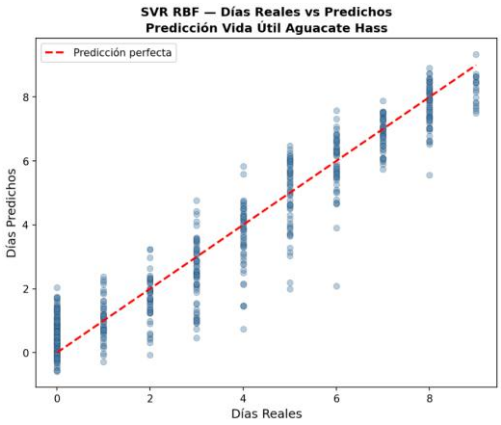


3. Gráficas de dispersión

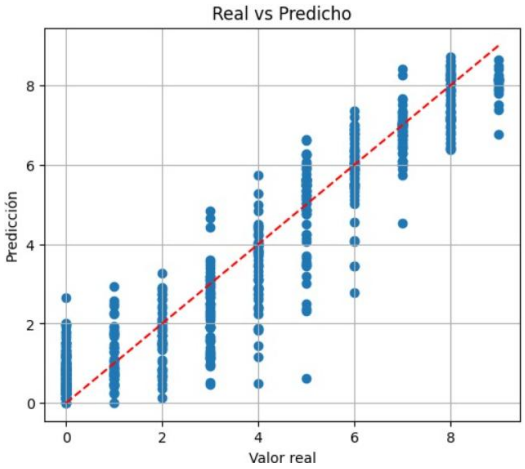
3.1. Gráficas de dispersión de los modelos clásicos de regresión de Machine Learning.

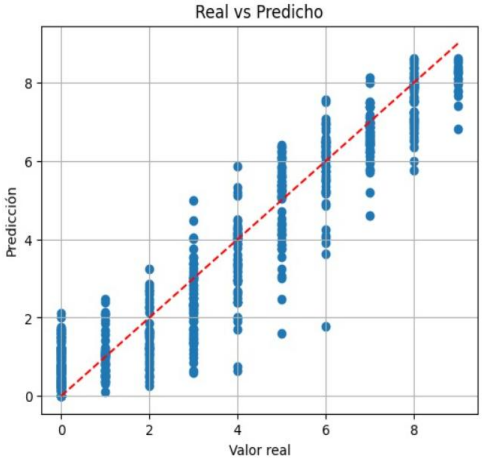
- Gráficas de dispersión del SVR con 31 características (izquierda) y 76 características (derecha).



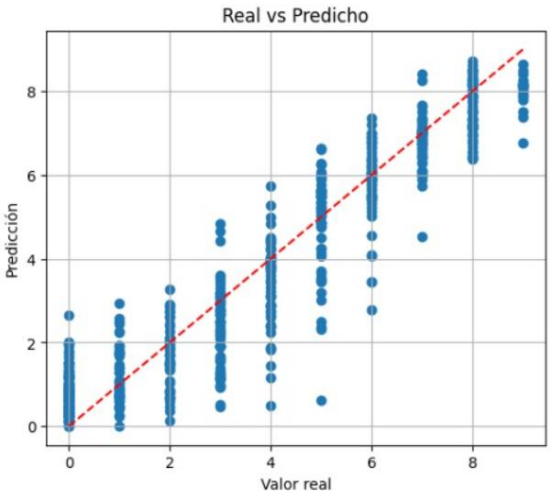


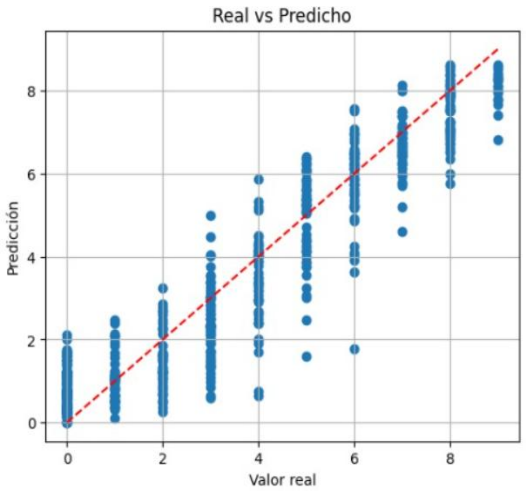
- Gráficas de dispersión del KNN con 31 características (izquierda) y 76 características (derecha).



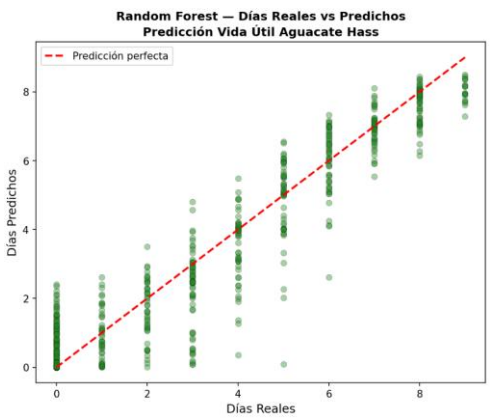


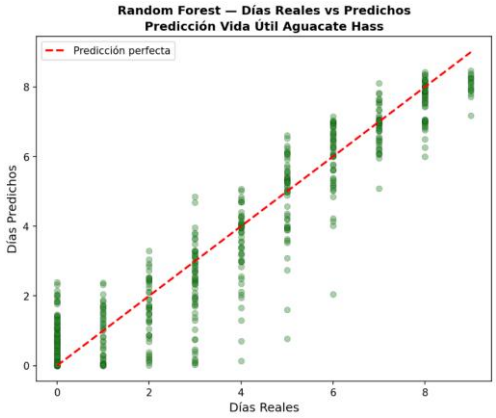
- Gráficas de dispersión del MLP con 31 características (izquierda) y 76 características (derecha).





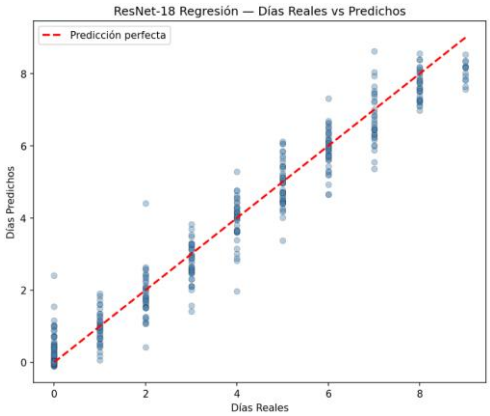
- Gráficas de dispersión del Random Forest con 31 características (izquierda) y 76 características (derecha).

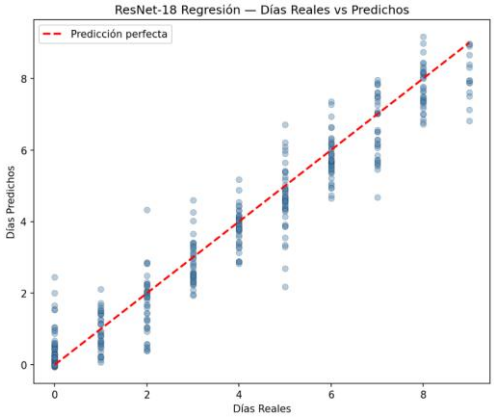




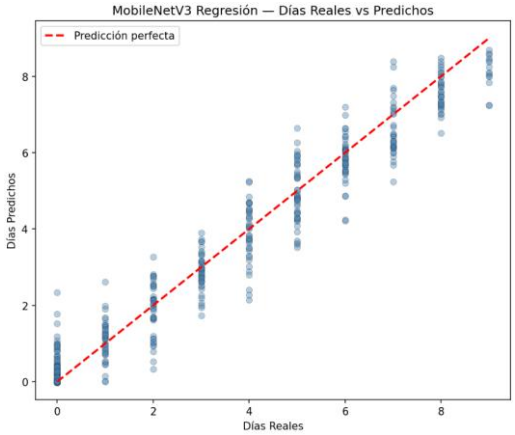
3.2. Gráficas de dispersión de los modelos de redes neuronales convolucionales de regresión.

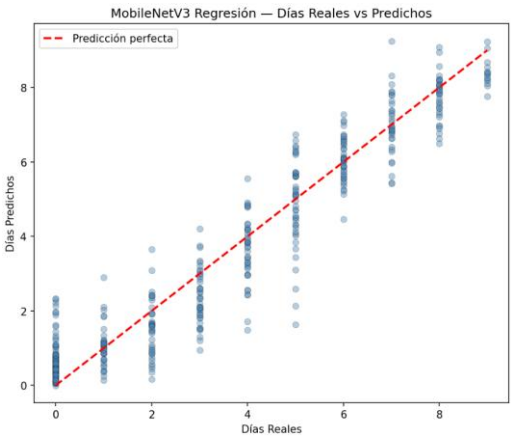
- Gráficas de dispersión de ResNet-18 sin (izquierda) y con aumento de datos (derecha).



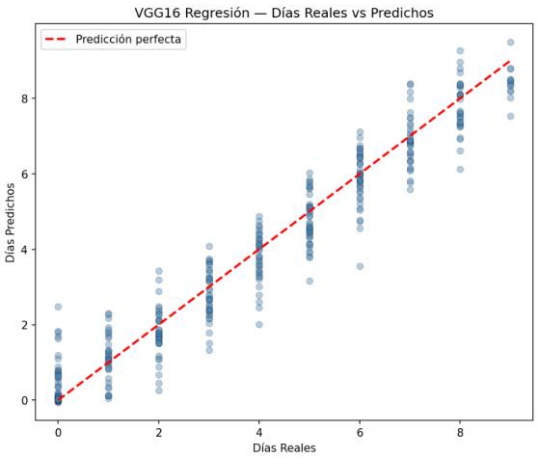


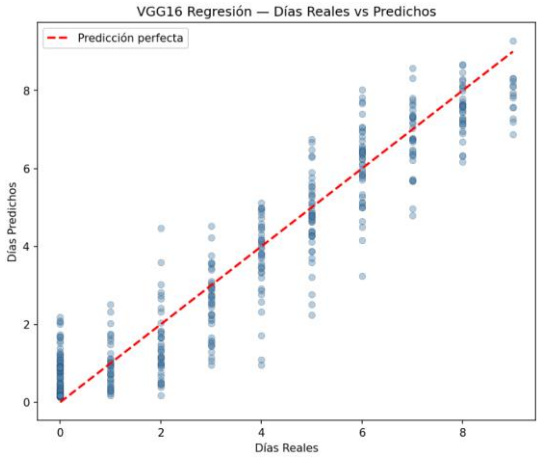
- Gráficas de dispersión de MobileNetV3 sin (izquierda) y con aumento de datos (derecha).



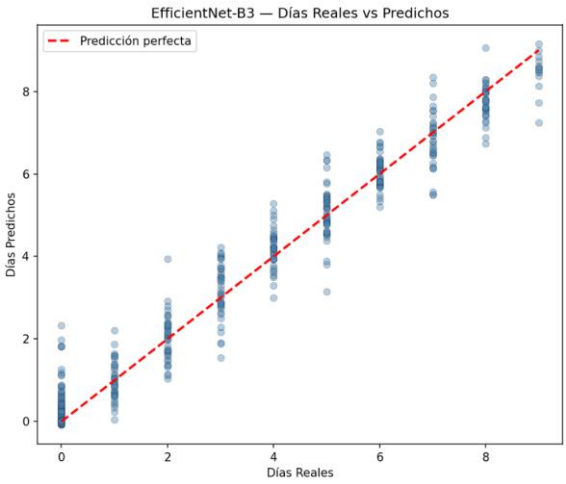


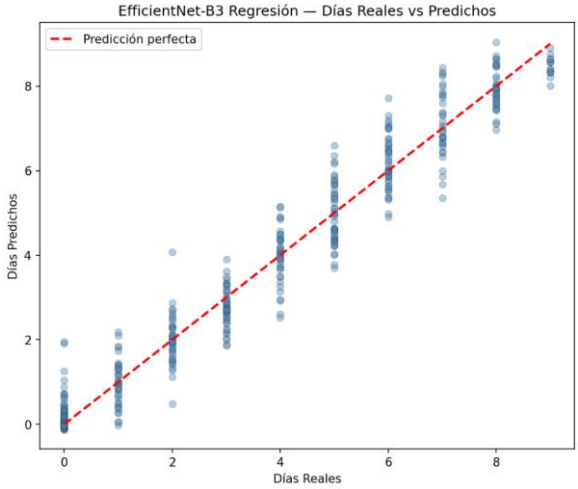
- Gráficas de dispersión de VGG16 sin (izquierda) y con aumento de datos (derecha).





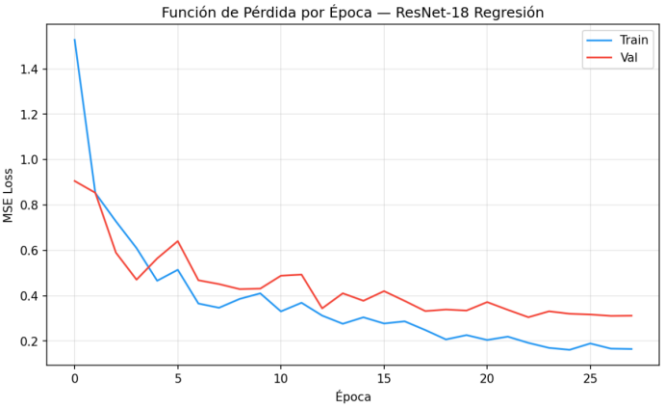
- Gráficas de dispersión de EfficientNet-B3 sin (izquierda) y con aumento de datos (derecha).

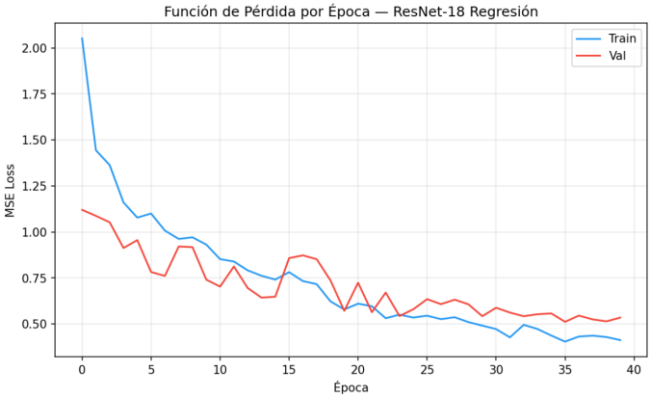




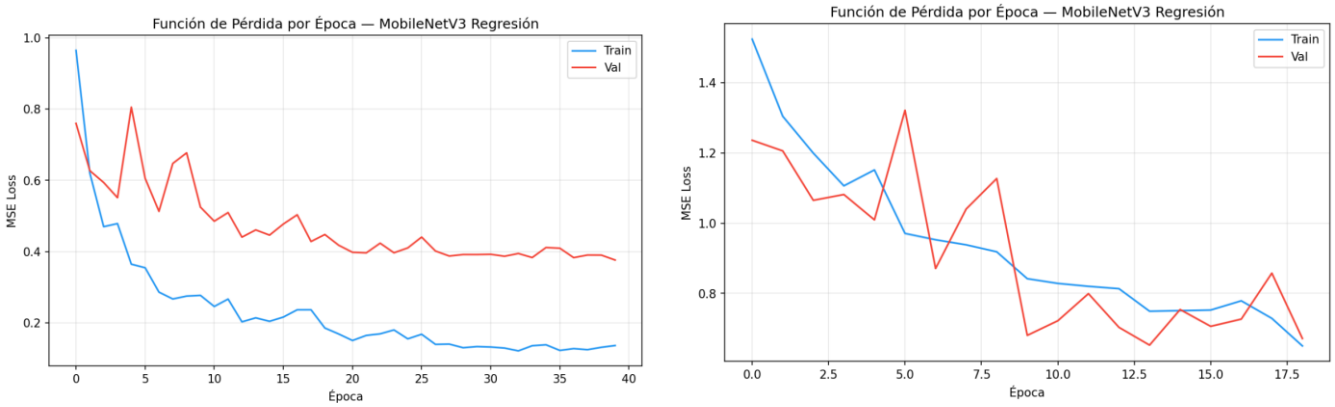
4. Gráficas de pérdida de modelos de redes neuronales convolucionales.

- Gráficas de pérdida MSE de ResNet-18 sin (izquierda) y con aumento de datos (derecha).

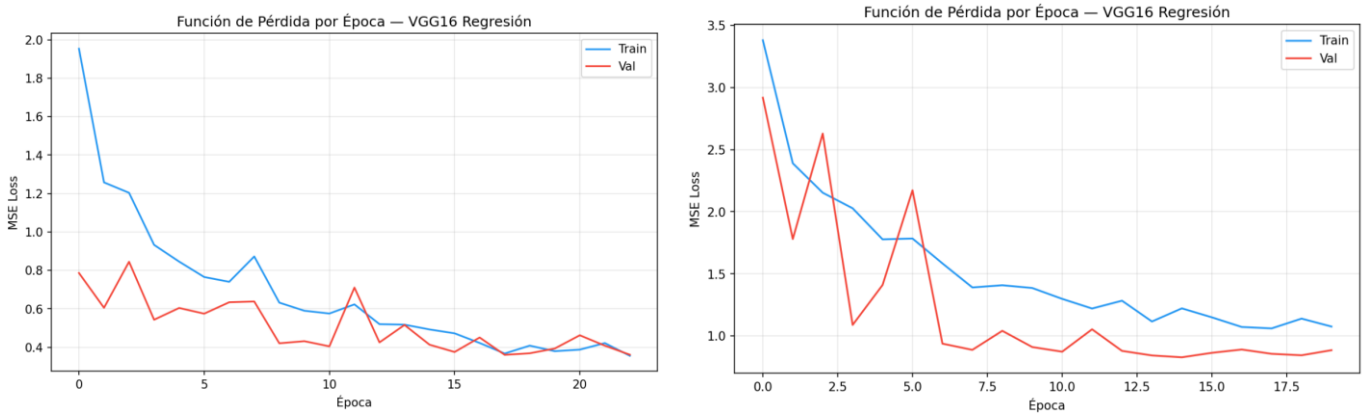




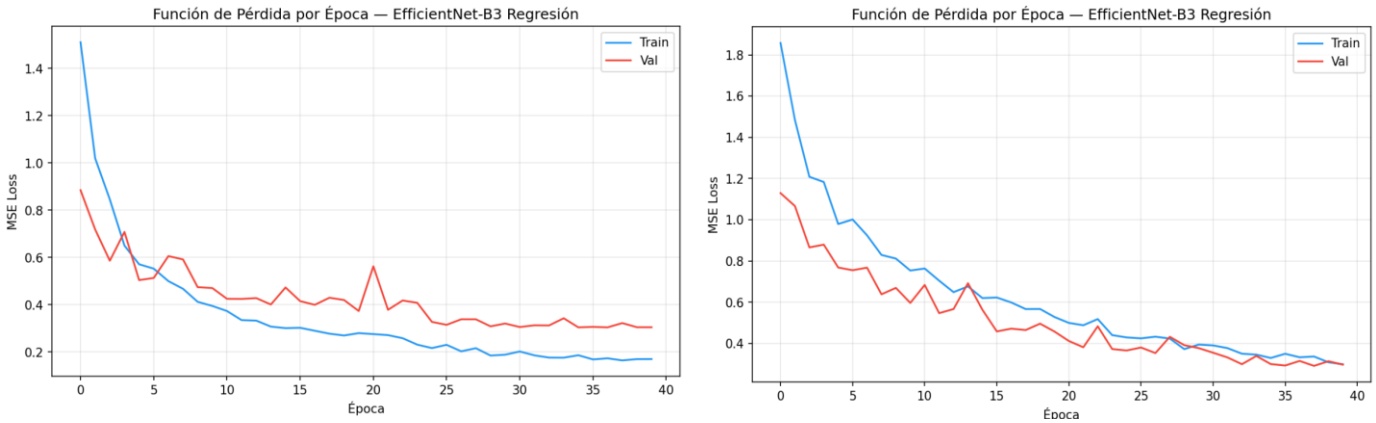
- Gráficas de pérdida MSE de MobileNetV3 sin (izquierda) y con aumento de datos (derecha).



- Gráficas de pérdida MSE de VGG16 sin (izquierda) y con aumento de datos (derecha).



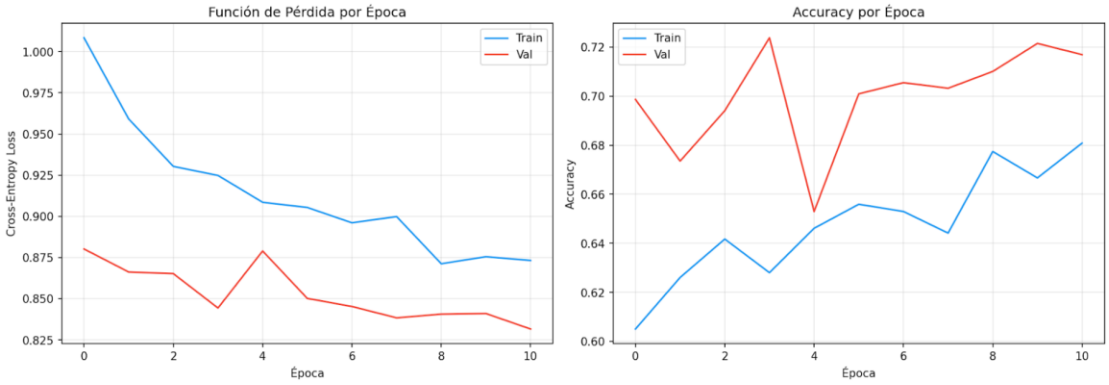
- Gráficas de pérdida MSE de EfficientNet-B3 sin (izquierda) y con aumento de datos (derecha).



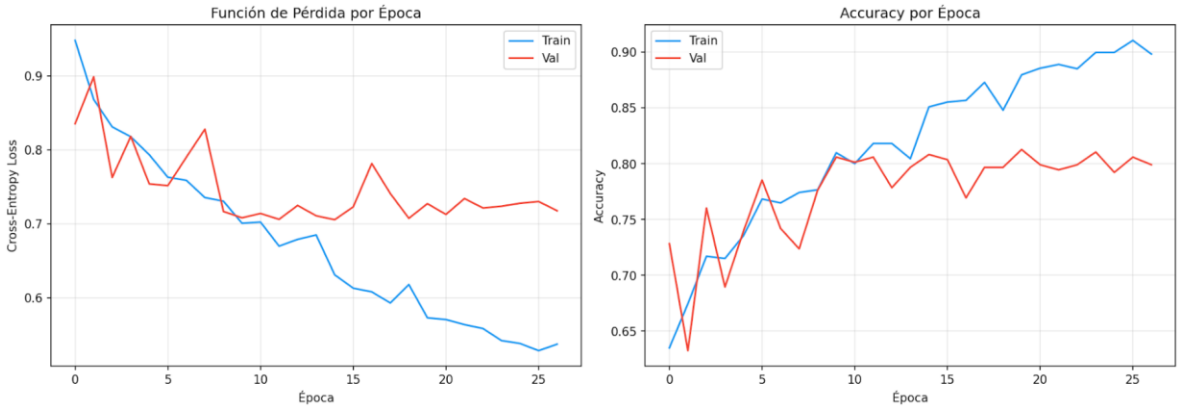
5. Gráficas de pérdida de modelos de redes neuronales convolucionales en el experimento de Fine-Tuning.

- Gráficas de perdida y accuracy por época de VGG en distintos porcentajes de descongelamiento del backbone.

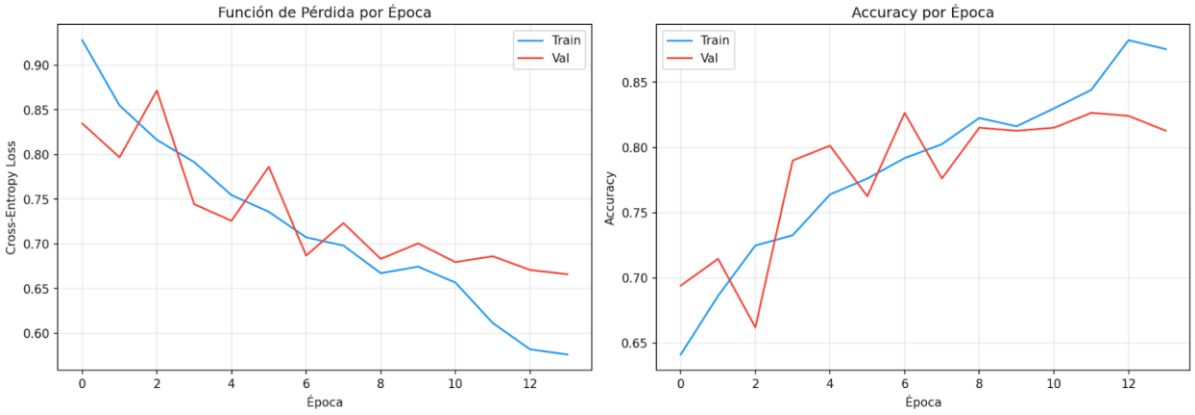
a) 0% de descongelamiento



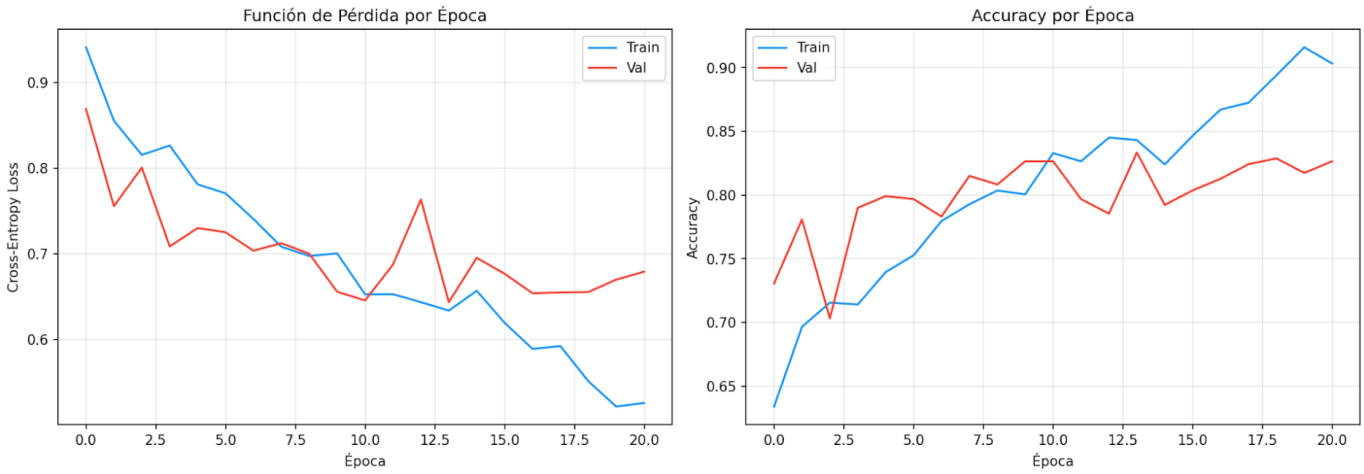
b) 25% de descongelamiento



c) 50% de descongelamiento

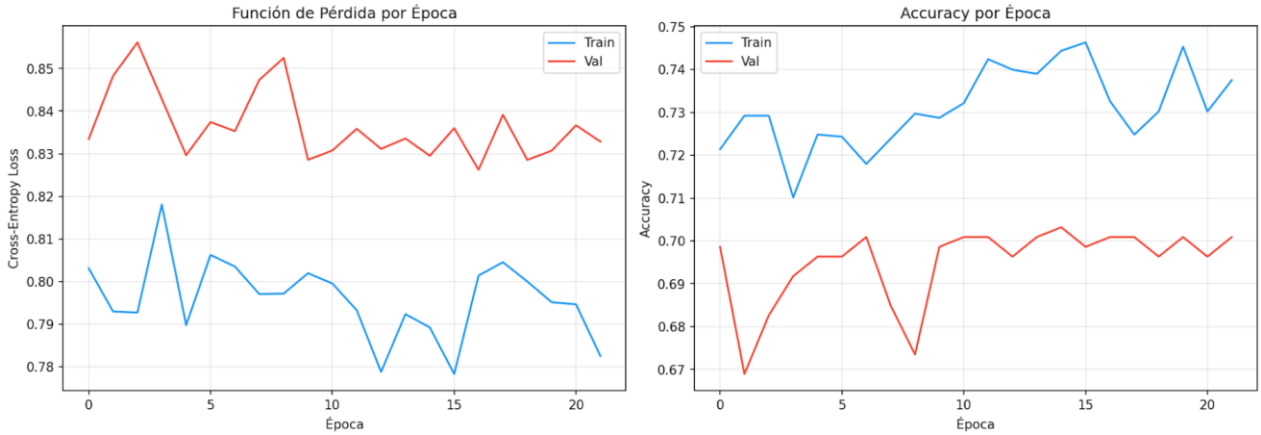


d) 75% de descongelamiento

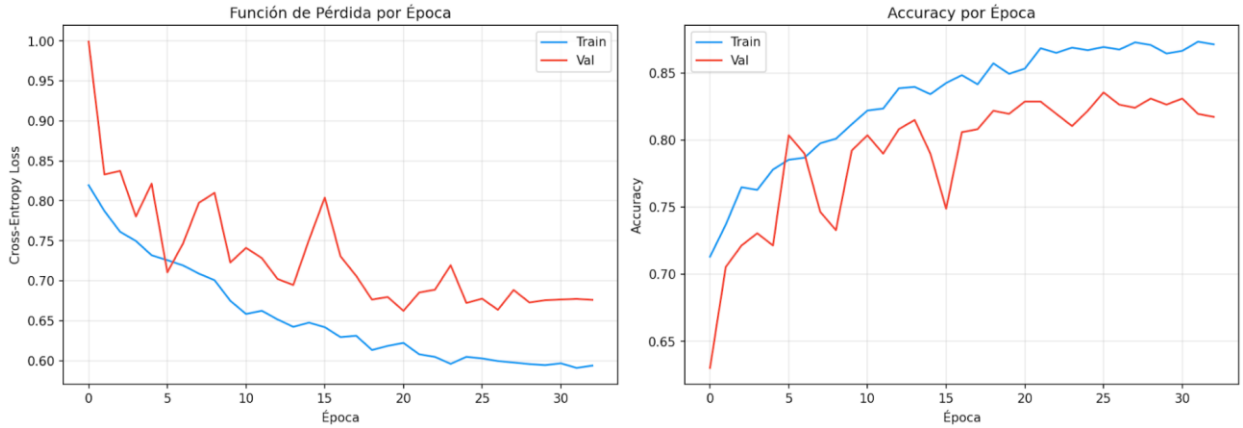


- Gráficas de perdida y accuracy por época de MobileNetV3 en distintos porcentajes de descongelamiento del backbone.

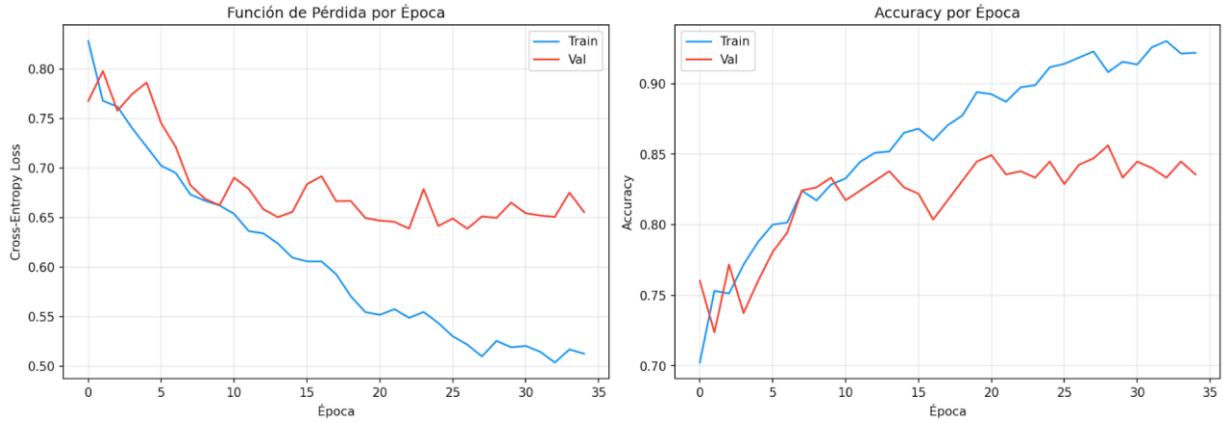
a) 0% de descongelamiento



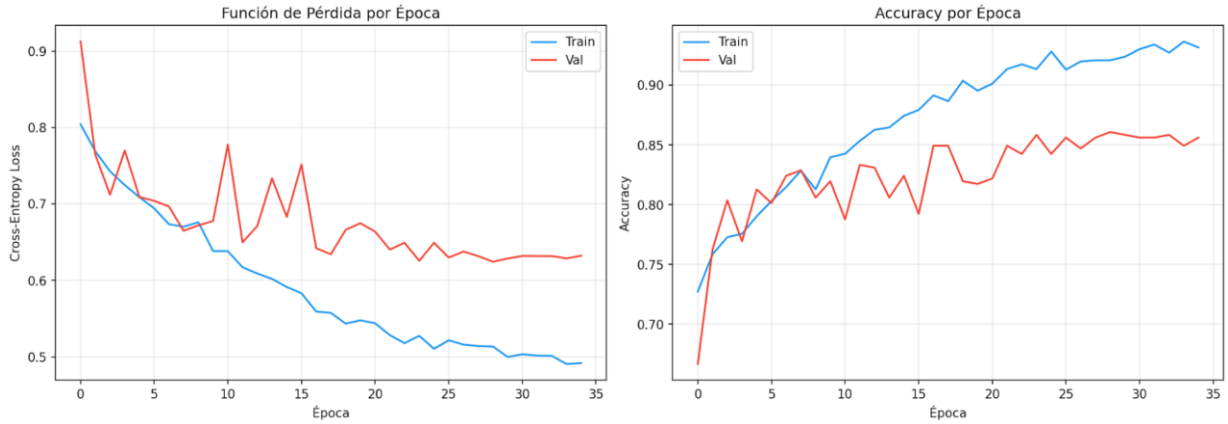
b) 25% de descongelamiento



c) 50% de descongelamiento



d) 75% de descongelamiento



6. Recursos del proyecto

- Link para acceder a la carpeta del proyecto con todos los recursos usados.

https://drive.google.com/drive/folders/1SGAg7lalvdxCw01vBBRykGEAk1aEo8vw?usp=sharing

- Link al Colab donde se encuentran los modelos seleccionados para la interfaz y su conexión con ella.

https://colab.research.google.com/drive/13gHwHkS3wrfDnngv7CVR-zWvjcKiYCet?usp=sharing

- Link del repositorio que se usó para la elaboración de la interfaz.

https://github.com/SebasEE77/Avocado_Classifier#