Detección de enfermedades en maíz mediante IA optimizada para implementación en dispositivos de borde

Luis E. Macea Zabaleta[1], Andrés F. Flórez Olivera[1], Nicolás Castillo Ojeda[1],
Oscar J. González Zárate[1,2]
[1] *Maestría en Inteligencia Artificial - Universidad de los Andes, Bogotá - Colombia*
[2] *Center for Embedded Devices and Research in Digital Agriculture (CEDRA), São Leopoldo - RS, Brazil*

***Resumen*—Las enfermedades foliares del maíz —roya común, mancha gris y tizón foliar— generan pérdidas relevantes y su diagnóstico manual es lento, subjetivo, especialmente en zonas rurales con conectividad limitada. Este trabajo propone un sistema de diagnóstico automatizado basado en visión por computador pensado para dispositivos de borde (Edge Computing), capaz de operar sin dependencia de la nube y con recursos computacionales restringidos. La solución consiste en un clasificador multiclase optimizado para inferencia offline. La metodología comparó cuatro arquitecturas de aprendizaje profundo DL (Deep Learning) —MobileNetV3-Large, EfficientNet-LiteB0, MobileViT y Plant Mobile Vision Transformer (PMVT)— bajo aprendizaje por transferencia TL (Transfer Learning) con ajuste fino progresivo. El desarrollo del entrenamiento se implementó en Python con TensorFlow y el seguimiento del ciclo de vida de cada modelo se hizo mediante MLflow, siguiendo la metodología MLOps (Machine Learning Operations); la viabilidad edge se validó midiendo exactitud, recall por clase y latencia.**

De acuerdo con los resultados obtenidos, *MobileNetV3-Large* mostró el mejor desempeño, con una precisión de prueba cercana a 0.92 y un *recall* por clase ≥ 0.83. Por ello, únicamente este modelo fue seleccionado para avanzar hacia el proceso de optimización orientado a dispositivos de borde. Tras poda y cuantización de rango dinámico, el modelo seleccionado obtuvo $\approx 0,92$ de precisión, con fuerte reducción de tamaño y latencia, cumpliendo restricciones de hardware embebido. Esto facilita diagnósticos rápidos y objetivos en campo, favoreciendo la implementación de agricultura de precisión en zonas con baja conectividad.

Palabras clave— Agricultura de precisión; Visión por computador; Deep learning; Edge computing; Transfer learning.

I. Introducción

El maíz es un cultivo estratégico para la seguridad alimentaria global y regional. Su productividad se ve afectada por diversas enfermedades foliares capaces de generar pérdidas significativas en el rendimiento [1]. La detección tradicional, basada en la inspección visual realizada por expertos, es un proceso subjetivo, lento y difícil de escalar en cultivos extensivos [2].

Esta dependencia del juicio humano limita la estandarización del diagnóstico y retrasa intervenciones oportunas, lo que incrementa los costos y reduce la eficiencia productiva [2], [1].

A estas limitaciones se suma que muchas soluciones digitales disponibles actuales requieren conectividad estable o recursos computacionales elevados, condiciones operativas poco asequibles en zonas rurales, especialmente de Suramérica [3], [4]. En este contexto, la computación en el borde (*Edge Computing*), emerge como una alternativa pertinente para habilitar diagnósticos rápidos y objetivos en tiempo casi real, reducir costos de infraestructuras y facilitar despliegues distribuidos en entornos con conectividad restringida [5], [6]. Este enfoque es especialmente relevante para aplicaciones de agricultura de precisión donde el objetivo es apoyar al productor mediante sistemas eficientes, portátiles y operativos fuera de la nube.

Frente a estos retos, se hace necesario avanzar hacia modelos de visión por computadora, que mantengan un adecuado nivel de precisión diagnóstica y simultáneamente, puedan ejecutarse en dispositivos de borde con recursos limitados. Asimismo, persiste la necesidad de sistematizar procesos de curación de datos, evaluar arquitecturas eficientes del estado del arte y aplicar métodos de compresión —como poda y cuantización— que permitan reducir la huella computacional, sin comprometer el desempeño [6].

En respuesta a este panorama, este trabajo presenta una solución basada en aprendizaje profundo orientada a la clasificación de enfermedades foliares del maíz y optimizada para ejecución en el borde. El estudio integra datos provenientes de repositorios públicos, evalúa arquitecturas de Deep Learning modernas bajo transferencia de aprendizaje y explora técnicas de optimización para habilitar inferencia fuera de la nube. Los resultados obtenidos permiten analizar el potencial de un enfoque *edge* para apoyar el diagnóstico fitopatológico en regiones agrícolas.

Contribuciones: Este trabajo aporta: (i) un protocolo de preparación de datos trazable y robusto; (ii) un *benchmark* de modelos orientado a dispositivos de borde; (iii) una arquitectura y un proceso de entrenamiento reproducibles mediante aprendizaje por transferencia; y (iv) una ruta de optimización (poda/cuantización) para un despliegue operativo en terreno. En conjunto, estos elementos ofrecen una base técnica sólida para habilitar diagnóstico fitopatológico *offline* y decisiones agronómicas oportunas en contextos de recursos restringidos.

II. REVISIÓN DEL ESTADO DEL ARTE

A. Etapa de revisión

La detección automatizada de enfermedades en cultivos con visión por computador ha avanzado impulsada por la disponibilidad de datos abiertos, la madurez metodológica y computacional del desarrollo de redes neuronales, y la necesidad de soluciones escalables para la agricultura de precisión [2], [7], [8]. En el cultivo del maíz, la relevancia económica está documentada por estimaciones de pérdida de rendimiento asociadas a enfermedades, lo que subraya la urgencia de diagnósticos oportunos y confiables [1]. En paralelo, la agenda de agricultura digital y *Edge AI* plantea que llevar el cómputo al borde, habilita decisiones en tiempo casi real, reduce costos de infraestructura y posibilita despliegues distribuidos en entornos con conectividad limitada—un ajuste adaptable para contextos rurales de Suramérica [3], [4], [9], [5], [6].

Un punto de partida técnico clave fue *PlantVillage*, que permitió entrenar *CNN* clásicas (p. ej., *AlexNet* y GoogLeNet) con precisiones cercanas al $\approx 99\%$ en condiciones controladas. Sin embargo, estas evaluaciones se realizaron sobre imágenes con fondos uniformes e iluminación estable, lo que limita la validez externa ante la variabilidad propia de las condiciones reales de campo [2].

Esta brecha motiva la transición hacia enfoques mas robustos basados en *transfer learning* sobre arquitecturas preentrenadas—comúnmente *VGG*, *ResNet* o *EfficientNet*— que permiten acelerar el entrenamiento y mejorar generalización sin sacrificar desempeño [10]. En paralelo, se han explorado mecanismos de auto-atención adaptados a maíz con el fin de capturar patrones mas complejos en condiciones reales [11].

Los estudios recientes en campo refuerzan esta necesidad: la mezcla de enfermedades, la variabilidad lumínica y heterogeneidad ambiental, los cuales deterioran el rendimiento de los modelos desarrollados debido a su entrenamiento efectuado en entornos controlados, plantean la urgencia de soluciones más resilientes ante condiciones variables [7], [8].

Con miras al despliegue en dispositivos de borde, se ha consolidado el uso de arquitecturas ligeras (familias *MobileNet* y *EfficientNet-Lite*) y variantes móviles de *Transformers*; en particular, propuestas como *PMVT* demuestran que es viable sostener buen rendimiento en terminales móviles, favoreciendo casos de uso con recursos restringidos [12]. En paralelo, la literatura en agricultura digital y *Edge AI* destacan el uso de técnicas de optimización -como la poda y cuantización- orientadas a reducir la latencia, el consumo energético y la huella de memoria, condiciones indispensables para la ejecución en *hardware* embebido [9], [5], [6].

Los ecosistemas actuales de datos y herramientas optimizan la reproducibilidad y la comparación rigurosa entre modelos. Repositorios abiertos como *Kaggle* y *Roboflow* permiten obtener conjuntos de datos verificables, mientras que la integración de marcos como *TensorFlow* para el entrenamiento de redes neuronales y sistemas de trazabilidad como *MLflow* garantiza comparaciones objetivas de configuraciones y métricas, fortaleciendo así la transparencia experimental [13], [14], [15], [16], [17], [18].

B. Brechas y análisis

A partir de la revisión del estado del arte, es posible identificar tres vacíos que motivan el presente trabajo: (i) dependencia de infraestructura centralizada, donde muchas soluciones digitales asumen conectividad o cómputo elevado; (ii) optimización insuficiente para el borde, con escasos estudios que documenten de forma sistemática técnicas de compresión posteriores al entrenamiento incluyendo variantes de -poda/cuantización- y su efecto en conjunto sobre latencia y memoria; y (iii) validación limitada en condiciones reales, donde la variabilidad lumínica, climática y morfológica compromete los resultados reportados en laboratorio [8], [2], [5], [6]. En respuesta a estas brechas, este trabajo se enfoca en el desarrollo de un modelo ligero y reproducible, evaluado bajo criterios de cómputo en el borde tales como precisión, *recall* y latencia. Se emplea *transfer learning* sobre arquitecturas eficientes y se aplican técnicas de compresión para habilitar la inferencia local en dispositivos embebidos sin dependencia de la nube. Este enfoque es pensado a llevarse hacia escenarios agrícolas de Suramérica con conectividad limitada, para los cuales se proyectan futuras pruebas en campo y el uso de herramientas de interpretación del modelo.

III. METODOLOGÍA

El procesamiento y desarrollo del sistema se implementó siguiendo un enfoque orientado al cómputo en el borde (*Edge Computing*), priorizando el uso de modelos ligeros con arquitecturas eficientes, baja latencia y una huella de memoria reducida, en consonancia con la literatura actual en agricultura digital [3], [4]. El *backbone* de la solución se restringe a arquitecturas diseñadas específicamente para entornos con recursos limitados: *MobileNetV3-Large*, *EfficientNet-LiteB0*, *MobileViT* y *PMVT*. Estas redes presentan un número reducido de parámetros y tamaños compatibles con el despliegue *on-device*, características que permiten integrar eficientemente el entrenamiento mediante *transfer learning* y facilitan la posterior optimización mediante poda y cuantización [8], [12], [11].

Se utiliza aprendizaje por transferencia para acelerar la convergencia y mejorar la generalización utilizando datos del dominio agrícola [10]. Para garantizar la viabilidad en el dispositivo, la metodología incorpora, posterior al entrenamiento base, un proceso de optimización de dos etapas: primero, la aplicación de poda (*pruning*) y, posteriormente, la cuantización (*quantization*). Estas técnicas se evalúan controladamente para medir su impacto en la precisión, *recall* por clase y latencia. Finalmente, el modelo resultante se convierte a formato *TensorFlow Lite*, generando un ejecutable ligero (*.tflite*) idóneo para dispositivos de borde.

A. Recolección de datos

Para la conformación del conjunto de datos se emplearon imágenes en formato *RGB* de hojas de maíz provenientes de

dos repositorios abiertos, alcanzando un volumen inicial de 7.835 muestras. Este total se compone de 4.188 imágenes extraídas del *Corn Leaf Disease Dataset* [13], [14], [15] y 3.647 imágenes pertenecientes al conjunto *Corn Diseases* [16].

B. Análisis exploratorio de datos

Siguiendo la nomenclatura de trabajos de referencia [2], [11], se realizó una unificación taxonómica de las etiquetas en cuatro clases: `Healthy`, `Common_Rust`, `Gray_Leaf_Spot` y `Blight`. Esta estandarización evita ambigüedades entre fuentes, facilita la evaluación comparativa y se complementó con la ejecución de *hashing perceptual* para identificar y eliminar duplicados, evitando así fugas de información entre particiones.

Durante la exploración se identificaron limitaciones críticas en las fuentes originales. Por un lado, el conjunto de *Roboflow* incluía imágenes previamente aumentadas con rotaciones y reflexiones sin documentación explícita y carecía totalmente de la clase `Healthy`, impidiendo el acceso al volumen completo planeado inicialmente; debido a esta pre-aumentación, dicho material no se consideró apto para las pruebas y su uso se restringió cuidadosamente al entrenamiento. Por otro lado, el conjunto de datos crudo de *Kaggle* presentó un desbalance significativo donde la clase minoritaria, `Gray_Leaf_Spot`, contaba únicamente con 574 imágenes, lo que representa una diferencia superior al 40% en comparación con las demás clases.

C. Preprocesamiento

Para garantizar la reproducibilidad y obtener una estimación no sesgada del desempeño, se efectuó una partición estratificada de los datos en carpetas de entrenamiento (*train*), validación (*val*) y prueba (*test*). Dada la disparidad en la naturaleza de las fuentes, específicamente entre las imágenes de Roboflow con aumento previo y las de Kaggle sin él, se diseñó una estrategia de conformación diferenciada. Para los conjuntos de validación y prueba se utilizó exclusivamente el conjunto de *Kaggle* como base, dado que contiene imágenes originales sin alteraciones artificiales, asegurando así una evaluación sobre datos reales.

Por el contrario, al subconjunto de entrenamiento se le aplicó aumentación de datos (*data augmentation*) para corregir el desbalance y mejorar la generalización. Es crucial destacar que en este proceso se excluyeron alteraciones cromáticas, limitándose a transformaciones espaciales (rotaciones, traslaciones y reflexiones). Esta decisión obedece a que la uniformidad cromática es un factor diagnóstico clave en patologías foliares; por ejemplo, las hojas sanas presentan una distribución de verde uniforme, y modificar el color, tono o contraste podría introducir artefactos que degraden la capacidad del modelo para identificar clorosis o lesiones específicas, un riesgo documentado en estudios de campo [8].

Como resultado, el conjunto de datos final comprendió un total de 9.928 imágenes con una distribución completamente balanceada para evitar sesgos hacia clases mayoritarias, tal como se detalla en la Tabla I. Esta estructura final asignó 1.488 imágenes para la fase de prueba y 1.488 para validación (correspondientes a 372 por clase en cada caso), destinando el volumen restante de 6.952 imágenes a la fase de entrenamiento (1.738 por clase).

Tabla I
TABLA DE DISTRIBUCIÓN DE IMÁGENES POR CLASE Y CARPETAS TRAS EL PREPROCESAMIENTO

Clase	Train	Val	Test
Common Rust	1738	372	372
Gray Leaf Spot	1738	372	372
Blight	1738	372	372
Healthy	1738	372	372

D. Pipeline de características

El pipeline de entrada se construyó utilizando el artefacto `image_dataset_from_directory` de TensorFlow, generando un `tf.data.Dataset` etiquetado y por lotes. Todas las imágenes fueron redimensionadas a 224×224 píxeles para asegurar la compatibilidad con los *backbones* móviles y controlar el uso de memoria. Posteriormente, se aplicó una normalización re-escalando los valores de intensidad al rango $[0, 1]$, lo cual reduce la variabilidad numérica, estabiliza los gradientes y favorece la convergencia. Para optimizar el rendimiento computacional, se implementó `prefetch(tf.data.AUTOTUNE)`, permitiendo solapar el procesamiento de la GPU con la preparación de datos en la CPU, y se preservó el arreglo de nombres de clases (`class_names`) antes de cualquier optimización para garantizar la trazabilidad correcta en las matrices de confusión.

1) Selección de modelos y entrenamiento: Se evaluaron cuatro arquitecturas del estado del arte orientadas a escenarios móviles: *MobileNetV3-Large*, *EfficientNet-LiteB0*, *MobileViT* y *PMVT*. La selección se basó en su eficiencia demostrada en el equilibrio entre precisión, latencia y memoria [8], [12], [11], adoptando *transfer learning* con pesos de ImageNet para maximizar la eficiencia de los datos. El protocolo experimental estandarizado se desplegó en una infraestructura Google Colab con GPU T4/L4, utilizando el optimizador *Adam* ($lr = 1e{-}4$), un tamaño de lote de 64 y un horizonte inicial de 20 épocas. Asimismo, se configuraron los *callbacks EarlyStopping* (paciencia 10, monitoreando *val_loss*), *ReduceLROnPlateau* (factor 0.5) y *ModelCheckpoint* para el control del entrenamiento.

En la fase de selección, cuyo objetivo fue identificar el comportamiento más robusto y no la optimización de hiperparámetros, se sometió a cada arquitectura a múltiples configuraciones. La Figura 1 presenta el análisis estadístico comparativo derivado de estas pruebas.

Los resultados evidenciaron que *MobileNetV3-Large* ofrece el mejor rendimiento global. En términos de precisión y estabilidad, alcanzó un *accuracy* promedio de ≈ 0.915 con la menor desviación estándar (Figura 1b-c), superando a *EfficientNet* (≈ 0.755), *MobileViT* (≈ 0.718) y *PMVT* (≈ 0.641), además de presentar la mejor relación *loss/accuracy* en prueba (Figura 1d). Respecto al *recall* en la clase crítica `Gray_Leaf_Spot`, *MobileNetV3-Large* obtuvo un mínimo

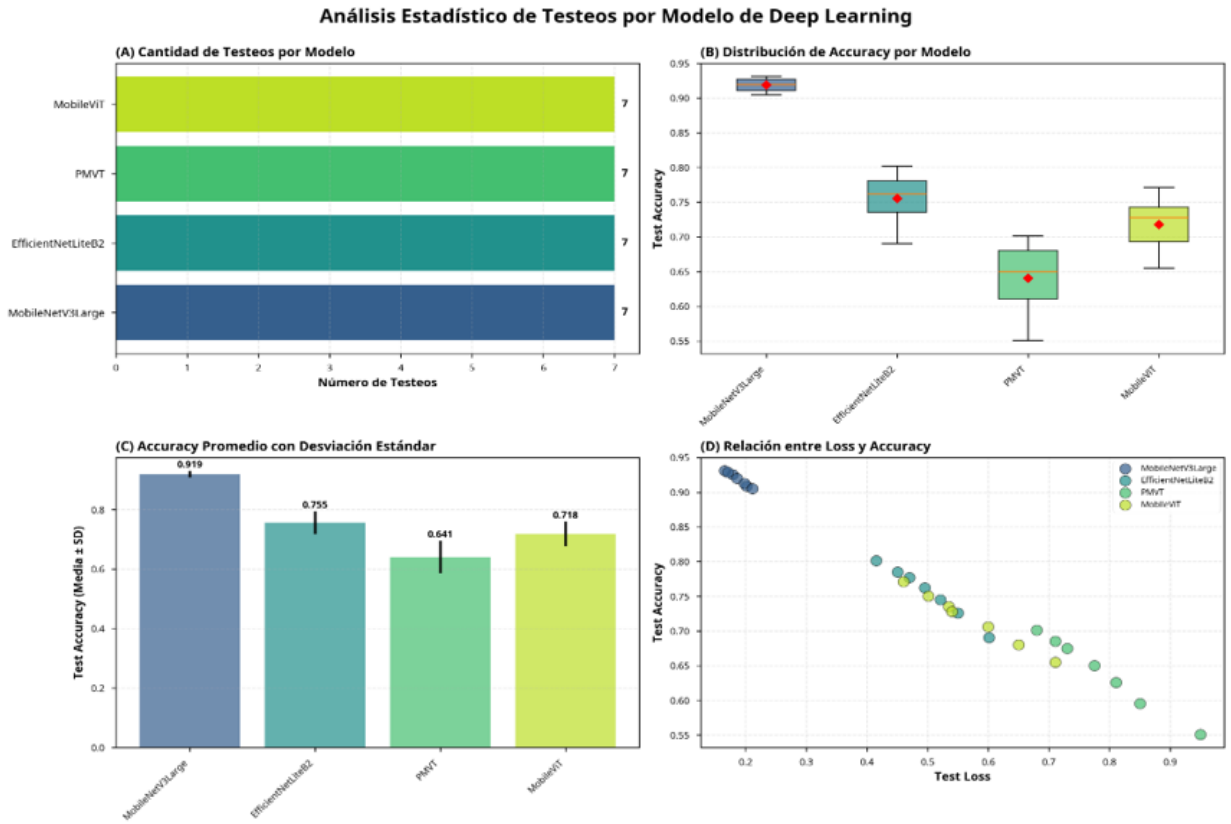

Fig. 1. Análisis estadístico comparativo de las arquitecturas. (a) Evaluaciones por modelo. (b) Distribución del accuracy. (c) Precisión promedio y desviación estándar. (d) Relación loss/accuracy.

de ≈ 0.853, superando el objetivo de 0.85, mientras que los demás modelos mostraron un desempeño deficiente oscilando entre ≈ 0.435 (PMVT) y ≈ 0.582 (EfficientNet). Finalmente, en cuanto a la eficiencia en borde mostrada en la Figura 2, este modelo presenta una huella de memoria significativamente menor (≈ 13 MB frente a ≈ 33 MB de los otros modelos) y mayor estabilidad entre ejecuciones. Debido a su exactitud superior, robustez frente a clases desafiantes y eficiencia computacional, *MobileNetV3-Large* fue seleccionada como la arquitectura base.

E. Entrenamiento de MobileNetV3-Large

Sobre la arquitectura base seleccionada, se desarrolló un pipeline de entrenamiento en dos fases para garantizar la estabilidad y convergencia. La primera fase, denominada de calentamiento, consistió en un entrenamiento con el *backbone* congelado durante 5 épocas para estabilizar las nuevas capas superiores. Posteriormente, en la segunda fase de *fine-tuning*, se procedió al descongelamiento progresivo a partir de la capa 100, aplicando una reducción de la tasa de aprendizaje ($\times 10$) y extendiendo el entrenamiento por 15 épocas adicionales.

La arquitectura final (Figura 3) utiliza *MobileNetV3-Large* (preentrenado en ImageNet, `include_top=False`, `pooling='avg'`) seguido de un cabezal de clasificación diseñado específicamente para el dominio. Este cabezal consta de dos capas densas (512 y 256 neuronas) con Normalización

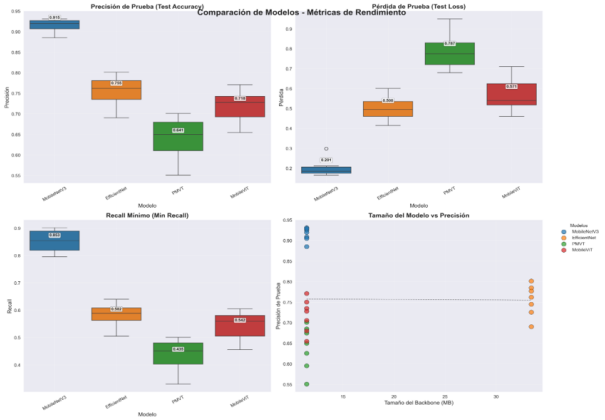

Fig. 2. Resultados comparativos de desempeño y huella de memoria de los modelos evaluados.

por Lotes (*Batch Normalization*) y regularización *Dropout* (0.5 y 0.3), finalizando en una capa *Softmax* de 4 neuronas. Dicha configuración añade un total de 625.924 parámetros entrenables, tal como se detalla en la Tabla II.

1) Evaluación del modelo para transición a edge: El proceso de entrenamiento mostró una convergencia adecuada desde las etapas iniciales (Figura 4), atribuible a la calidad del preprocesamiento y la calibración de la arquitectura; la mínima

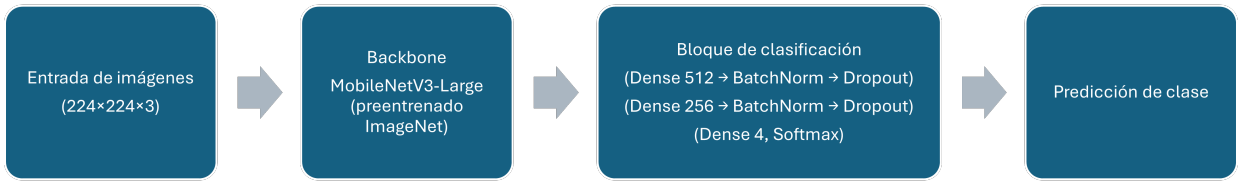

Fig. 3. Diagrama de bloques del modelo basado en MobileNetV3-Large y el flujo de clasificación.

Tabla II
DETALLE DE LA ARQUITECTURA DEL CABEZAL DE CLASIFICACIÓN Y PARÁMETROS.

Layer (type)	Output Shape	Param #
MobileNetV3Large (Functional)	(None, 960)	2,996,352
dense (Dense)	(None, 512)	492,032
batch_normalization (Batch Normalization)	(None, 512)	2,048
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131,328
batch_normalization_1 (Batch Normalization)	(None, 256)	1,024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 4)	1,028

Tabla III
MÉTRICAS DETALLADAS DE RENDIMIENTO EN ENTRENAMIENTO Y VALIDACIÓN.

	precision	recall	f1-score	support
Blight	0.9017	0.8387	0.8691	372
Common_Rust	0.9811	0.9785	0.9798	372
Gray_Leaf_Spot	0.8485	0.9032	0.8750	372
Healthy	0.9920	1.0000	0.9960	372
accuracy			0.9301	1488
macro avg	0.9308	0.9301	0.9300	1488
weighted avg	0.9308	0.9301	0.9300	1488

diferencia entre las curvas de entrenamiento y validación confirma la efectividad de las estrategias de regularización implementadas.

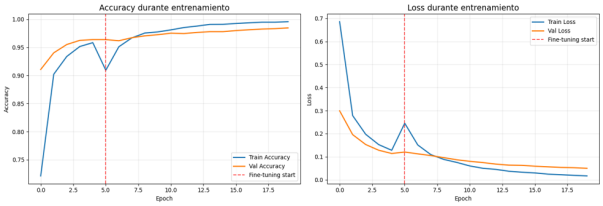

Fig. 4. Curvas de pérdida y accuracy durante el entrenamiento y validación.

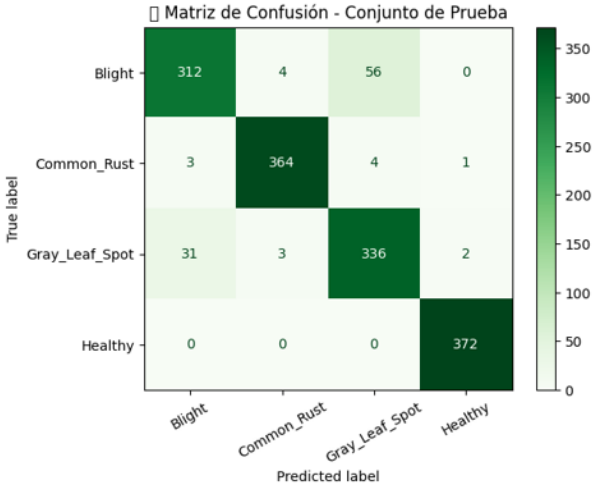

Fig. 5. Matriz de confusión del conjunto de prueba aplicado al modelo MobileNetV3-Large final.

El modelo final alcanzó una precisión promedio de $\approx 93.08\%$ y un recall por clase alrededor del 93.01% (Tabla III). El archivo resultante en formato `.keras` tiene un peso de 40.78 MB, validando su idoneidad preliminar para dispositivos de borde [6].

Por su parte, la matriz de confusión sobre el conjunto de prueba (Figura 5) revela un desempeño excelente en las clases `Healthy` y `Common_Rust`, mientras que los errores residuales se concentran en el solapamiento morfológico entre `Blight` y `Gray_Leaf_Spot`, un desafío documentado en la literatura [8].

F. Optimización para el borde: poda y cuantización

Con el objetivo de reducir la huella de memoria y latencia sin degradar la capacidad diagnóstica [19], [9], [6], se aplicaron técnicas de optimización sobre el modelo entrenado. El procedimiento inició con la poda, donde se clonó el modelo base y se aplicó una esparsidad objetivo del 50% sobre las capas densas para eliminar pesos redundantes. A continuación, se generó una versión adicional del modelo podado sometida a un ajuste fino (*fine-tuning*) para recuperar la potencial pérdida de precisión, culminando con la cuantización de rango dinámico o DRQ (*Dynamic Range Quantization*). Esta técnica reduce la precisión de los pesos y activaciones de punto flotante (`float32`) a enteros de 8 bits (`int8`), disminuyendo

el tamaño del modelo con un impacto mínimo en la exactitud.

Para explorar la ruta más eficaz, se generaron y evaluaron tres versiones cuantizadas finales, comparándolas bajo métricas estándar (*accuracy*, *precision*, *recall*, *f1-score*, tamaño en MB y tiempo de inferencia en ms): el modelo original cuantizado, el modelo podado y cuantizado, y el modelo con poda, *fine-tuning* y cuantización.

G. Reproducibilidad y disponibilidad de recursos

El código fuente, los pesos preentrenados del modelo y el corpus de datos utilizados se encuentran disponibles en el repositorio del proyecto: GitHub, bajo licencia privada de los autores.

IV. RESULTADOS Y EVALUACIÓN

Con el fin de realizar una revisión objetiva de los resultados, se implementó una serie de pruebas sobre los modelos obtenidos en cada etapa del proceso. Para ello se empleó el conjunto de prueba (*test*), el cual, como se detalló en la metodología, no incluye imágenes aumentadas y no fue utilizado durante el entrenamiento. Este conjunto consta de 372 imágenes por cada una de las cuatro categorías, garantizando una evaluación imparcial del desempeño.

Previo al análisis de las técnicas de optimización, se verificó la estabilidad del entrenamiento del modelo base. Las curvas de aprendizaje y validación mostraron comportamientos consistentes sin indicios de sobreajuste prematuro, confirmando que las estrategias de regularización y el protocolo experimental permitieron una convergencia adecuada. Esta validación preliminar justificó el avance hacia las etapas de optimización orientadas al despliegue en el borde: poda y cuantización.

A. Poda

La aplicación de la técnica de poda sobre el modelo base generó un impacto positivo en la eficiencia sin comprometer significativamente la capacidad predictiva. experimentó una variación marginal de ≈ 0.27 puntos porcentuales (p.p.), pasando de $\approx 0,9234$ en el modelo original a $\approx 0,9207$ en el modelo podado, como se observa en la Figura 6. En contraparte, se lograron mejoras sustanciales en los recursos: el tamaño del archivo se redujo un 64.5%, descendiendo de 40.78 MB a 14.49 MB, y la latencia media por imagen disminuyó de ≈ 29.41 ms a ≈ 22.81 ms. La esparsidad total alcanzó un 49.9%, validando la efectividad de la técnica para eliminar pesos redundantes.

Posteriormente, se exploró la aplicación de un *fine-tuning* breve sobre el modelo podado. Si bien este ajuste redujo drásticamente la latencia a ≈ 4.19 ms, conllevó una degradación notable del desempeño: el *accuracy* cayó a $\approx 0,8884$ y las métricas de *precision*, *recall* y *f1-score* disminuyeron entre 3.3 y 3.5 p.p. respecto al modelo original. Adicionalmente, el tamaño del archivo permaneció en 40.78 MB, dado que la reducción física de tamaño requiere una etapa de reconstrucción (*stripping*) que no fue efectiva en esta variante tras el re-entrenamiento.

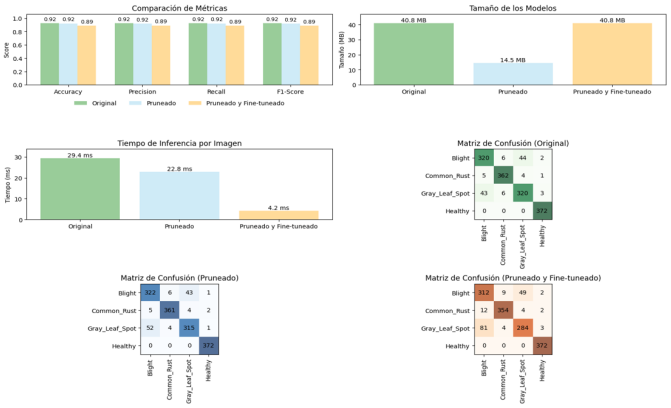

Fig. 6. Comparativo de resultados de los modelos original, podado y podado con *fine-tuning*.

El análisis de las matrices de confusión reveló patrones consistentes en los tres modelos (original, podado y podado con *fine-tuning*). Las clases *Healthy* y *Common_Rust* mantuvieron un desempeño superior, mientras que las confusiones se concentraron entre *Blight* y *Gray_Leaf_Spot*, comportamiento atribuible al solapamiento morfológico de sus lesiones.

En conclusión para esta etapa, el modelo podado sin *fine-tuning* se posicionó como el candidato más equilibrado, logrando reducir la huella de memoria y mejorar la latencia preservando la exactitud, a diferencia de la variante con ajuste fino que sacrificó demasiada precisión.

B. Cuantización

Partiendo de los resultados previos, se evaluó la cuantización de rango dinámico (*DRQ*) aplicada a tres rutas de procesamiento: el modelo original, el modelo podado y el modelo podado con *fine-tuning*. El objetivo principal consistió en convertir los modelos a formato *TensorFlow Lite* (*.tflite*) para analizar su viabilidad técnica en el despliegue en el borde.

1) Análisis de métricas y tamaño: La cuantización demostró ser una técnica robusta para la conservación del rendimiento tanto en el modelo original como en el podado, manteniendo métricas agregadas en torno a ≈ 0.92 como se observa en la Figura 7. Por el contrario, la variante podada con *fine-tuning* experimentó una caída generalizada en todas las métricas de evaluación (*accuracy*, *precision*, *recall*, *f1-score*), situándose cerca de ≈ 0.89, tal como se detalla en la Tabla IV.

Un hallazgo relevante en este proceso fue la convergencia en el tamaño de almacenamiento. Tras la conversión a *.tflite*, los tres modelos resultantes presentaron un peso aproximado de 3.7 MB. Esto indica que, al aplicar la técnica de cuantización DRQ, el proceso de *pruning* previo no aportó ventajas adicionales significativas en términos de la huella de memoria final del ejecutable.

2) Análisis de latencia: Contrario a la expectativa inicial de que el *pruning* mejoraría la velocidad de procesamiento, el modelo original cuantizado registró la menor latencia promedio (≈ 15.2 ms). Las variantes podadas mostraron tiempos

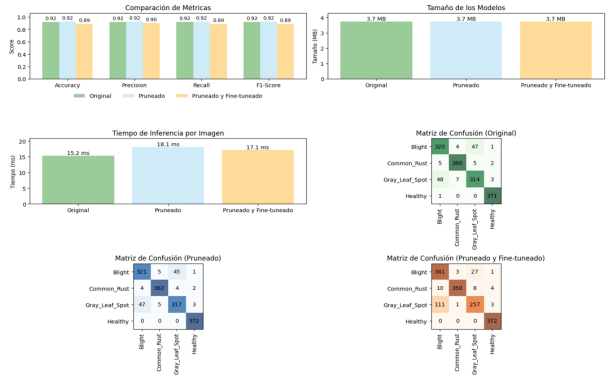

Fig. 7. Comparativo de resultados de cuantización sobre los modelos original, podado y podado con *fine-tuning*.

Tabla IV
DESEMPEÑO COMPARATIVO DE LOS MODELOS TRAS CUANTIZACIÓN

Modelo	**Accuracy**	**Precision**	**Recall**	**F1-score**
Original	0.92	0.92	0.92	0.92
Podado	0.92	0.92	0.92	0.92
Podado + *Fine-Tuning*	0.89	0.90	0.89	0.89

de inferencia ligeramente superiores, comportamiento que se observa en la Tabla V. Cabe señalar que estas pruebas se realizaron con *XNNPACK* deshabilitado por motivos de estabilidad, por lo que los resultados representan el comportamiento bajo esta configuración específica de hardware y software.

Tabla V
LATENCIA PROMEDIO POR IMAGEN PARA LOS MODELOS CUANTIZADOS.

Modelo	**Latencia por imagen (ms)**
Original	15.2
Podado	18.1
Podado + *Fine-Tuning*	17.1

3) Comportamiento por clase y discusión: El análisis de las matrices de confusión ratificó la estabilidad del modelo original y del podado tras la cuantización. La clase `Healthy` mostró una detección casi perfecta, mientras que `Common_Rust` mantuvo niveles elevados de acierto. No obstante, las confusiones principales persistieron entre las clases `Blight` y `Gray_Leaf_Spot`. Específicamente, en el modelo podado con *fine-tuning*, si bien mejoró ligeramente la detección de `Blight`, se degradó notablemente la separabilidad de `Gray_Leaf_Spot`, lo que desaconseja su uso sin un reentrenamiento dirigido.

Al contrastar estos resultados con el estado del arte, el *accuracy* de $\approx$ 0.92 alcanzado por el modelo cuantizado resulta competitivo frente a la literatura de diagnóstico foliar, que reporta rangos típicos entre 0.85 y 0.95 [8], [11], [12]. Esto confirma la viabilidad de la solución propuesta para operar en condiciones reales.

4) Selección final: Los resultados evidencian que la cuantización captura la mayor parte de los beneficios de reducción de tamaño (llegando a 3.7 MB) sin necesidad de poda previa. Dado que el *pruning* no mejoró la latencia ni redujo adicionalmente el tamaño tras la cuantización, y considerando que el *fine-tuning* degradó la precisión, se concluye que el modelo original cuantizado representa la alternativa más equilibrada para el despliegue en el borde. Esta configuración ofrece el mejor compromiso entre precisión ($\approx$ 0.92), velocidad ($\approx$ 15.2 ms) y tamaño compacto (3.7 MB), cumpliendo satisfactoriamente con los requerimientos de eficiencia y desempeño del proyecto.

V. DISCUSIÓN

Los resultados obtenidos confirman la viabilidad técnica de un sistema de diagnóstico de enfermedades foliares de maíz diseñado para ejecutarse en dispositivos de borde (*Edge Computing*) y operar sin dependencia de conectividad constante. A continuación se discuten los principales hallazgos en relación con la metodología adoptada, las estrategias de optimización, los patrones de error y su impacto agronómico.

A. Validación metodológica y selección de arquitectura

El énfasis en la preparación de datos constituye un pilar central de la validez de los resultados. La unificación taxonómica de etiquetas, el control de duplicados mediante *hashing* perceptual y, especialmente, la separación estricta entre imágenes originales y material aumentado en las particiones de validación y prueba reducen el riesgo de fuga de datos y proporcionan estimaciones más realistas que las de estudios basados en configuraciones de laboratorio idealizadas, como *PlantVillage* [2].

En este contexto más exigente, la elección de *MobileNetV3-Large* como arquitectura base se justifica empíricamente: fue el modelo con mejor balance entre exactitud promedio, estabilidad entre repeticiones y huella de memoria, superando a alternativas como *EfficientNet-LiteB0*, *MobileViT* y *PMVT* tanto en desempeño global como en sensibilidad para la clase minoritaria. Este comportamiento es coherente con trabajos que señalan a las familias *MobileNet* y *EfficientNet-Lite* como candidatas naturales para escenarios de recursos limitados [8], [12], [11]. La trazabilidad sistemática de experimentos en *MLflow* refuerza, además, la reproducibilidad del flujo propuesto dentro de un ciclo *MLOps*.

B. Poda, cuantización y transición al borde

Las estrategias de optimización aplicadas ofrecen diferentes perspectivas sobre el papel de la poda y la cuantización en este tipo de aplicaciones. En formato nativo (`.keras`), la poda post-entrenamiento sobre las capas densas cumplió el objetivo teórico de reducir la complejidad del modelo: se obtuvo una disminución sustancial del tamaño del archivo y de la latencia, con una penalización mínima en las métricas globales. En este estadio, la técnica resulta interesante para contextos donde el modelo deba almacenarse y ejecutarse directamente en marcos *TensorFlow* convencionales.

Sin embargo, tras la conversión a *TensorFlow Lite* con cuantización de rango dinámico, los tres modelos evaluados (original, podado y podado con *fine-tuning*) convergieron a

un tamaño muy similar en torno a unos pocos megabytes. Este hallazgo indica que, bajo la configuración considerada, la cuantización captura prácticamente todo el beneficio de compresión en disco, de modo que la poda deja de aportar ventajas de huella de memoria una vez aplicado *DRQ*.

La comparación de tiempos de inferencia refuerza esta idea: el modelo original cuantizado resultó ser el más rápido en el entorno evaluado, mientras que las variantes con poda presentaron latencias ligeramente superiores y, en el caso del modelo podado con *fine-tuning*, una pérdida apreciable de desempeño global. Este comportamiento es consistente con estudios que advierten que la combinación de altos niveles de esparsidad y reentrenamiento parcial puede introducir inestabilidad cuando el volumen de datos o la representación de ciertas clases es limitada. En conjunto, los resultados respaldan al modelo original cuantizado como punto de operación preferente para despliegue en el borde, reservando la poda para escenarios donde se persiga una reducción adicional en formato nativo o se disponga de un conjunto de entrenamiento más amplio para reentrenamientos dirigidos.

C. Patrones de error y retos pendientes

El análisis de las matrices de confusión muestra un patrón de error alineado con la literatura reciente sobre diagnóstico foliar en condiciones reales demostrado en [7], [8]. Las clases `Healthy` y `Common_Rust` presentan un comportamiento estable y confiable, mientras que las confusiones se concentran sistemáticamente en el pareado `Blight`–`Gray_Leaf_Spot`. Esta dificultad es conocida tanto en evaluaciones humanas como automáticas debido a la similitud morfológica de las lesiones cuando coexisten patologías y variaciones de iluminación en campo [8].

Además, `Gray_Leaf_Spot` es la clase con menor representatividad en el corpus consolidado, lo que contribuye a una menor sensibilidad relativa. Esto sugiere que las mejoras futuras no deberían enfocarse únicamente en arquitectura o compresión, sino también en una ampliación y curaduría dirigida de los datos para esta clase, así como en estrategias de entrenamiento sensibles al coste que penalicen con mayor fuerza los falsos negativos en enfermedades críticas.

D. Impacto práctico en agricultura de precisión

Desde una perspectiva aplicada, el modelo original cuantizado proporciona una solución compatible con dispositivos de borde de bajo consumo y coste razonable. Su tamaño reducido y la latencia observada permiten integrar la inferencia local en flujos de trabajo habituales de la agricultura de precisión, como inspecciones visuales con dispositivos portátiles, monitoreo remoto mediante cámaras embarcadas o apoyo a operaciones de pulverización dirigida.

En términos agronómicos, la posibilidad de ejecutar el modelo directamente en el borde —sin necesidad de conectividad estable ni infraestructura centralizada— facilita la toma de decisiones durante recorridos de lote y monitoreos rutinarios, reduciendo la dependencia de especialistas, acortando los tiempos de respuesta frente a brotes y estandarizando criterios de diagnóstico entre productores con distintos niveles de formación. Estas capacidades están alineadas con la agenda de agricultura digital y *Edge AI* para contextos con infraestructura limitada.

E. Limitaciones del estudio

El estudio presenta limitaciones que deben considerarse al interpretar los resultados. La validación se realizó sobre un corpus balanceado y cuidadosamente controlado, lo cual es adecuado para valorar la metodología, pero no captura toda la variabilidad lumínica, de fondos y de estados fenológicos que se observa en campo. La clase `Gray_Leaf_Spot` sigue subrepresentada, lo que afecta la sensibilidad del sistema para esta patología específica.

Por otro lado, las mediciones de latencia corresponden a una configuración de hardware y *runtime* concreta, con *XNNPACK* deshabilitado por estabilidad, de modo que los tiempos reportados deben entenderse como un escenario de referencia y no como valores universales. Finalmente, el impacto energético se infiere de forma indirecta a partir de la complejidad del modelo y no a partir de medidas directas de consumo.

F. Trabajo futuro

A partir de estos resultados, se pueden definir varias líneas de trabajo futuro. En primer lugar, resulta prioritario validar el modelo en condiciones reales mediante campañas multisitio y multitemporales en zonas rurales de Suramérica con diferentes condiciones climáticas y de manejo, evaluando tanto el comportamiento predictivo como la aceptabilidad por parte de técnicos y productores.

En segundo lugar, se plantea ampliar y curar el conjunto de datos, incrementando la representación de `Gray_Leaf_Spot` y de casos difíciles (*hard negatives*) para reducir la confusión con `Blight`, e incorporar estrategias de entrenamiento coste-sensibles o pérdidas focalizadas orientadas a minimizar falsos negativos en enfermedades de alto impacto.

Finalmente, en el plano de la optimización, se propone explorar esquemas de cuantización más agresivos (p. ej., cuantización entera con conjuntos de calibración representativos), evaluar distintos delegados y *runtimes* en hardware heterogéneo y realizar mediciones directas de consumo energético, con el fin de caracterizar mejor el compromiso entre precisión, latencia y energía en diferentes plataformas de borde.

G. Propuesta de dispositivos para Edge Computing

Teniendo en cuenta que uno de los objetivos del presente trabajo no es solo explorar un modelo de clasificación para la detección de enfermedades en hojas de maíz, sino también evaluar su viabilidad de despliegue en dispositivos de borde sin conectividad, resulta pertinente realizar una comparación preliminar de las plataformas de *hardware* potencialmente utilizables. El primer reto en esta selección es la amplitud del universo de dispositivos de borde; por ello, con el fin de

Tabla VI
COMPARACIÓN RESUMIDA DE PLATAFORMAS HARDWARE PARA LA EJECUCIÓN DEL MODELO EN EL BORDE.

Plataforma	**Tipo de hardware**	**Capacidad CNN 224×224**
Jetson Nano	CPU+GPU (SBC)	Alta
Kria KV260	CPU+FPGA (SoM de visión)	Alta
STM32MP257F-EV1	MPU Linux (Cortex-A35)	Media–alta
STM32N6570-DK	MCU+NPU (kit Edge-AI)	Media
Nucleo-N657XQ-Q	MCU+NPU (Nucleo)	Media / limitada
Nexys Video Artix-7	FPGA pura	Alta (teórica)
Nucleo-U545RE-Q	MCU sin NPU/GPU	Baja
PSoC 6 AI	MCU sin NPU (TinyML)	Baja

acotarlo y alinearlo con los objetivos de apoyo al instituto CEDRA, se optó por analizar ocho dispositivos que se encuentran actualmente en su inventario.

Además de esta restricción práctica, se tuvieron en cuenta las características del modelo propuesto, esto es, una *CNN* con entrada de imágenes de resolución 224×224×3 y un tamaño de aproximadamente $\approx 3,7$ MB. En la **Tabla VI** se sintetiza la capacidad relativa de cada plataforma para tareas de visión por computador. Las plataformas con *GPU* o lógica reconfigurable (JETSON NANO y KRIA KV260) ofrecen la menor latencia de inferencia, aunque a costa de una mayor complejidad de integración y requisitos de *hardware*. El STM32MP257F-EV1 y el STM32N6570-DK representan un compromiso entre rendimiento e integrabilidad en entornos industriales, por lo que se perfilan como las opciones más adecuadas para desplegar el modelo en escenarios de agricultura de precisión. Por su parte, las placas basadas únicamente en microcontroladores (NUCLEO-U545, PSOC 6) quedan restringidas a modelos TinyML y se descartan para la arquitectura propuesta.

Finalmente, dado que el modelo base ocupa del orden de 3,7 MB, su despliegue resulta viable en plataformas con sistema operativo y memoria externa (JETSON NANO, KRIA KV260, STM32MP257F-EV1, STM32N6570-DK), mientras que las placas basadas exclusivamente en microcontroladores sin memoria externa significativa (NUCLEO-U545RE-Q, PSOC 6 AI) se descartan al no disponer de suficiente memoria de programa ni *RAM* para almacenar los pesos y las activaciones de una *CNN* con entrada de 224×224 píxeles.

VI. CONCLUSIONES

Este trabajo presentó un flujo completo para el diseño, entrenamiento y optimización de un modelo de diagnóstico de enfermedades foliares de maíz orientado a dispositivos de borde. La propuesta integra buenas prácticas de preparación de datos, aprendizaje por transferencia y MLOps, y demuestra que es posible alcanzar desempeños competitivos bajo restricciones realistas de computación y memoria.

En primer lugar, el *pipeline* de datos —basado en la unificación taxonómica, el control de duplicados y una partición estratificada que separa estrictamente material aumentado de los conjuntos de validación y prueba— permitió obtener estimaciones de desempeño más cercanas a las condiciones de uso real que las derivadas de escenarios puramente de laboratorio. Sobre esta base, *MobileNetV3-Large* se consolidó como la arquitectura más adecuada, al combinar una exactitud global en el orden de los bajos 90 % con un *recall* por clase que supera los umbrales mínimos definidos y una huella de memoria moderada. El patrón de error concentrado en el pareado `Blight–Gray_Leaf_Spot` pone de relieve la necesidad de estrategias específicas para clases difíciles, pero no compromete la utilidad global del sistema.

En segundo lugar, las etapas de poda y cuantización aportan una lectura clara sobre el compromiso entre compresión y desempeño. La poda post-entrenamiento permitió una reducción nativa del tamaño del modelo del orden del 65 % y una mejora de la latencia sin deterioro relevante de las métricas, lo que la hace atractiva cuando se trabaja directamente en formato `.keras`. No obstante, tras la cuantización de rango dinámico en *TensorFlow Lite*, los distintos modelos convergieron a un tamaño similar de unos 3.7 MB, y el modelo original cuantizado ofreció simultáneamente la menor latencia y las mejores métricas globales. La combinación de poda y *fine-tuning* no resultó ventajosa en este escenario, ya que redujo la precisión y afectó especialmente a la clase `Gray_Leaf_Spot` sin aportar beneficios adicionales en huella de memoria.

En consecuencia, la conclusión técnica central es que, bajo las condiciones evaluadas, el modelo original cuantizado constituye la opción más equilibrada para despliegue en el borde, al ofrecer un compromiso favorable entre precisión, tamaño reducido y tiempos de respuesta compatibles con aplicaciones de campo. Este resultado matiza la expectativa de que la poda mejore sistemáticamente la latencia una vez aplicada cuantización, y sugiere que las decisiones de optimización deben evaluarse de forma conjunta y contextualizada al *runtime* y al hardware objetivo.

Finalmente, desde la perspectiva agronómica, la posibilidad de ejecutar el modelo en dispositivos de borde abre una vía concreta para apoyar el diagnóstico fitosanitario en regiones con infraestructura limitada. Al proporcionar inferencia local, rápida y estandarizada, la solución propuesta tiene el potencial de reducir la dependencia de especialistas, mejorar la oportunidad de las intervenciones y contribuir a prácticas de agricultura de precisión más sostenibles. Como trabajo futuro se plantea ampliar y diversificar el corpus de entrenamiento, profundizar en estrategias orientadas a reducir falsos negativos en enfermedades críticas, y validar la solución en campañas de campo multisitio y en un espectro más amplio de plataformas de *hardware* embebido.

RECONOCIMIENTOS

Se extiende un agradecimiento especial al Dr. Carlos Julio González Aguilera, coordinador técnico de proyectos del *Center for Embedded Devices and Research in Digital Agriculture* (CEDRA, Brasil), por su supervisión técnica y guía metodológica a lo largo del desarrollo de este proyecto. Asimismo, a la Dra. Haydemar Núñez, directora del programa de Maestría en Inteligencia Artificial de la Universidad de los Andes (Colombia), por su revisión crítica y orientaciones que enriquecieron sustancialmente este trabajo; y al equipo docente

de la Maestría por su acompañamiento académico durante el proceso formativo.

Este trabajo ha sido financiado parcialmente por el proyecto ”IA na borda”, apoyado por el Centro de Dispositivos Integrados e Investigación en Agricultura Digital (CEDRA) de SENAI-RS, con recursos financieros del PPI IoT/Manufatura 4.0 / PPI HardwareBR del MCTI, bajo la subvención número 056/2023, firmada con EMBRAPII.

REFERENCIAS

[1] D. S. Mueller, K. A. Wise, A. J. Sisson, T. W. Allen, G. C. Bergstrom, K. M. Bissonnette, C. A. Bradley, E. Byamukama, M. I. Chilvers, A. A. Collins, P. D. Esker, T. R. Faske, A. J. Friskop, A. K. Hagan, R. W. Heiniger, C. A. Hollier, T. Isakeit, T. A. Jackson-Ziems, D. J. Jardine, H. M. Kelly, N. M. Kleczewski, A. M. Koehler, S. R. Koenning, D. K. Malvick, H. L. Mehl, R. F. Meyer, P. A. Paul, A. J. Peltier, P. P. Price, A. E. Robertson, G. W. Roth, E. J. Sikora, D. L. Smith, C. A. Tande, D. E. P. Telenko, A. U. Tenuta, L. D. Thiessen, and W. J. Wiebold, “Corn yield loss estimates due to diseases in the united states and ontario, canada, from 2016 to 2019,” *Plant Health Progress*, vol. 21, no. 4, p. 238–247, Jan. 2020. [Online]. Available: http://dx.doi.org/10.1094/PHP-05-20-0038-RS

[2] S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using deep learning for image-based plant disease detection,” *Frontiers in Plant Science*, vol. 7, Sep. 2016. [Online]. Available: http://dx.doi.org/10.3389/fpls.2016.01419

[3] R. Abiri, N. Rizan, S. K. Balasundram, A. B. Shahbazi, and H. Abdul-Hamid, “Application of digital technologies for ensuring agricultural productivity,” *Heliyon*, vol. 9, no. 12, p. e22601, Dec. 2023. [Online]. Available: http://dx.doi.org/10.1016/j.heliyon.2023.e22601

[4] B. Basso and J. Antle, “Digital agriculture to design sustainable agricultural systems,” *Nature Sustainability*, vol. 3, no. 4, p. 254–256, Apr. 2020. [Online]. Available: http://dx.doi.org/10.1038/s41893-020-0510-0

[5] M. O’Grady, D. Langton, and G. O’Hare, “Edge computing: A tractable model for smart agriculture?” *Artificial Intelligence in Agriculture*, vol. 3, p. 42–51, Sep. 2019. [Online]. Available: http://dx.doi.org/10.1016/j.aiia.2019.12.001

[6] R. Singh and S. S. Gill, “Edge ai: A survey,” *Internet of Things and Cyber-Physical Systems*, vol. 3, p. 71–92, 2023. [Online]. Available: http://dx.doi.org/10.1016/j.iotcps.2023.02.004

[7] M. Arsenovic, M. Karanovic, S. Sladojevic, A. Anderla, and D. Stefanovic, “Solving current limitations of deep learning based approaches for plant disease detection,” *Symmetry*, vol. 11, no. 7, p. 939, Jul. 2019. [Online]. Available: http://dx.doi.org/10.3390/sym11070939

[8] H. A. Craze, N. Pillay, F. Joubert, and D. K. Berger, “Deep learning diagnostics of gray leaf spot in maize under mixed disease field conditions,” *Plants*, vol. 11, no. 15, p. 1942, Jul. 2022. [Online]. Available: http://dx.doi.org/10.3390/plants11151942

[9] *Agricultura de Precisão:: Um Novo Olhar na Era Digital*. Editora Cubo, 2024. [Online]. Available: http://dx.doi.org/10.4322/978-65-86819-38-0.1000001

[10] F. CHOLLET, *Deep Learning with Python*. AManning, 2021.

[11] X. Qian, C. Zhang, L. Chen, and K. Li, “Deep learning-based identification of maize leaf diseases is improved by an attention mechanism: Self-attention,” *Frontiers in Plant Science*, vol. 13, Apr. 2022. [Online]. Available: http://dx.doi.org/10.3389/fpls.2022.864486

[12] G. Li, Y. Wang, Q. Zhao, P. Yuan, and B. Chang, “Pmvt: a lightweight vision transformer for plant disease identification on mobile devices,” *Frontiers in Plant Science*, vol. 14, Sep. 2023. [Online]. Available: http://dx.doi.org/10.3389/fpls.2023.1256773

[13] Corn or maize leaf disease dataset. Accessed: 2025-11-06. [Online]. Available: https://www.kaggle.com/datasets/smaranjitghose/corn-or-maize-leaf-disease-dataset

[14] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, and N. Batra, “Plantdoc: A dataset for visual plant disease detection,” 2019. [Online]. Available: https://arxiv.org/abs/1911.10317

[15] A. P. J, “Data for: Identification of plant leaf diseases using a 9-layer deep convolutional neural network,” 2019. [Online]. Available: https://data.mendeley.com/datasets/tywbtsjrjv/1

[16] Corn diseases dataset. Accessed: 2025-07-16. [Online]. Available: https://universe.roboflow.com/corn-disease-7/corn-diseases-oxojk

[17] Tensorflow. Accessed: 2025-11-06. [Online]. Available: https://www.tensorflow.org

[18] Mlflow. Accessed: 2025-11-06. [Online]. Available: https://mlflow.org

[19] M. R. Cherubin, J. M. Damian, T. R. Tavares, R. G. Trevisan, A. F. Colaço, M. T. Eitelwein, M. Martello, R. Y. Inamasu, O. H. d. C. Pias, and J. P. Molin, “Precision agriculture in brazil: The trajectory of 25 years of scientific research,” *Agriculture*, vol. 12, no. 11, p. 1882, Nov. 2022. [Online]. Available: http://dx.doi.org/10.3390/agriculture12111882