

Fundamentos de Deep Learning

Semestre 2025-2

Informe No 2

Plant Pathology 2020 - FGVC7

Clasificación de Enfermedades en Hojas de Manzano

Realizado por
Duvan Ferney Ruiz Ocampo

Docente

Raul Ramos Pollan

1. Introducción

El presente proyecto se desarrolla en el marco de la competencia Plant Pathology 2020 - FGVC7, con el objetivo de implementar un sistema automatizado para la clasificación de enfermedades en hojas de manzano utilizando técnicas avanzadas de deep learning. La detección temprana y precisa de patologías vegetales representa un avance significativo para la agricultura de precisión, permitiendo intervenciones oportunas que pueden salvar cosechas enteras y optimizar el uso de recursos fitosanitarios.

El dataset proporcionado por Kaggle contiene imágenes de alta resolución de hojas de manzano, con la particularidad de ser un problema de clasificación multi-etiqueta donde una misma hoja puede presentar múltiples patologías simultáneamente. Esta característica añade complejidad al desafío, requiriendo aproximaciones metodológicas específicas para el manejo de clasificaciones no excluyentes.

2. Objetivos

2.1. Objetivo General

Desarrollar e implementar un sistema automatizado de clasificación de enfermedades en hojas de manzano mediante técnicas de Deep Learning y Redes Neuronales Convolucionales, capaz de identificar con alta precisión cuatro categorías patológicas (healthy, multiple_diseases, rust, scab) a partir de imágenes digitales, contribuyendo a la detección temprana y al manejo fitosanitario eficiente en cultivos de manzana.

2.2. Objetivos específicos

- Diseñar y ejecutar un pipeline completo de preprocessamiento de imágenes que incluye técnicas de aumento de datos, normalización, redimensionamiento y conversión a formatos optimizados (TFRecords), asegurando la calidad y consistencia del dataset para el entrenamiento de modelos de Deep Learning.
- Implementar y comparar múltiples arquitecturas de Redes Neuronales Convolucionales, incluyendo modelos personalizados desde cero y modelos pre-entrenados mediante transfer learning (EfficientNetB0, ResNet50), evaluando su desempeño mediante métricas de accuracy, precision, recall y F1-score.
- Optimizar el modelo seleccionado mediante técnicas avanzadas de regularización y ajuste de hiperparámetros, incluyendo early stopping, reducción adaptativa de learning rate y class weighting, para lograr un balance óptimo entre capacidad predictiva y capacidad de generalización en condiciones reales.

3. Estructura y Metodología de Desarrollo

El proceso inició con una exhaustiva exploración y preparación de los datos. El análisis demográfico del dataset reveló una distribución desigual entre las cuatro clases objetivo: healthy (hojas sanas), multiple_diseases (múltiples enfermedades), rust (enfermedad tipo "óxido") y scab (costra foliar). Esta distribución desbalanceada, común en problemas de patología vegetal reales, fue cuidadosamente considerada en el diseño del pipeline de procesamiento para evitar sesgos en el modelo final.

El proyecto se estructuró en una secuencia lógica de notebooks interconectados, cada uno con un propósito específico dentro del flujo de trabajo:

Notebook 1: Exploración y Análisis Inicial

Este notebook constituye la fase inicial de exploración y configuración del proyecto de clasificación de enfermedades en plantas utilizando el dataset *Plant Pathology 2020*. Su objetivo principal es establecer las bases para el desarrollo posterior del modelo mediante:

- Descarga y verificación del dataset desde Kaggle
- Análisis exploratorio de datos completo
- Estudio de distribución de clases y balanceo
- Verificación de calidad y consistencia de imágenes
- Análisis dimensional de las muestras

Notebook 2: Preprocesamiento de Datos

Este notebook se enfoca en la preparación inicial de los datos: autenticación, descarga y extracción del dataset necesario para el entrenamiento de modelos de clasificación de enfermedades en hojas de manzano, su estructura está compuesta de:

- Implementación de pipeline de preprocesamiento de imágenes
- Conversión a formato TFRecords para optimización
- Estrategias de data augmentation
- Normalización y estandarización de datos
- División estratificada entrenamiento/validación

Notebook 3: Desarrollo y Entrenamiento de Modelos

El objetivo de este notebook es desarrollar un modelo de clasificación multi-label para identificar enfermedades en hojas de manzano utilizando Redes Neuronales Convolucionales (CNN), su estructura está compuesta de:

- Implementación de la arquitectura CNN personalizada
- Configuración de modelos de transfer learning
- Estrategias de compilación y optimización
- Entrenamiento con callbacks avanzados
- Validación cruzada y métricas de evaluación
- Preparación de submissions para Kaggle

Notebook 4: Comparación y Análisis de Resultados

Realizar una comparación sistemática de modelos de clasificación multi-etiqueta para identificar enfermedades en hojas de manzano, utilizando diferentes arquitecturas de redes neuronales convolucionales y técnicas de aprendizaje por transferencia, su estructura incluye:

- Evaluación comparativa de modelos
- Análisis de métricas de desempeño
- Generación de visualizaciones y gráficas
- Selección del mejor modelo

Se realizó un flujo de trabajo integrado donde la estructura modular permitió un desarrollo iterativo y metodológico, donde cada fase construía sobre los resultados de la anterior. Esta aproximación facilitó la depuración, el monitoreo del progreso y la reproducibilidad de los experimentos. La integración entre notebooks se gestionó mediante archivos intermedios (TFRecords, pesos de modelos, métricas) que aseguraban la consistencia a través de las diferentes etapas.

4. Solución Implementada

Antes de que los modelos pudieran aprender a reconocer enfermedades, era fundamental preparar adecuadamente las imágenes. En primer lugar, se estandarizó todas las imágenes a un mismo tamaño, específicamente 224x224 píxeles, para que los modelos puedan procesarlas de manera consistente. Seguidamente, se normalizó los valores de los píxeles para que estuvieran en un rango entre 0 y 1, lo que ayuda a que el aprendizaje sea más estable y eficiente.

Una de las estrategias más importantes implementadas fue el aumento de datos. ¿Por qué fue esto crucial? Porque en el mundo real, las fotos de hojas pueden tomarse desde diferentes ángulos, con distintas iluminaciones y en diversas condiciones. Para preparar los modelos para esta realidad, se crearon versiones modificadas de las imágenes originales: se rotaron



ligeramente, se ajustó su zoom y se modificó el brillo y contraste. Por último, se optimizó la forma de almacenar y acceder a las imágenes utilizando el formato TFRecords, lo cual permitió manejar grandes volúmenes de datos de manera más eficiente, reduciendo los tiempos de carga y mejorando el rendimiento durante el entrenamiento.

4.2. Arquitectura de Modelos Implementados

Se exploró cuatro enfoques diferentes, cada uno con sus propias características y ventajas:

- **CNN Básico (Punto de Referencia):** Se diseñó esta red como modelo de base, conscientes de que su simplicidad permitiría establecer un punto de partida claro. La arquitectura consta de tres capas convolucionales progresivas, comenzando con 32 filtros en la primera capa, aumentando a 64 en la segunda y manteniendo 64 en la tercera. Cada capa convolucional va seguida de una operación de max pooling que reduce gradualmente el tamaño de las imágenes, permitiendo que la red se concentrara en las características más relevantes.
- **CNN Personalizada (Diseño Propio).** La red personalizada representa un salto significativo en complejidad y capacidad. Se organizó la arquitectura en cuatro bloques convolucionales progresivos, cada uno con un propósito específico en la jerarquía de extracción de características. El primer bloque con 32 filtros se especializa en detectar bordes y texturas básicas. El segundo bloque con 64 filtros combina estas características simples en patrones más complejos. El tercer bloque con 128 filtros identifica estructuras morfológicas específicas, y el cuarto bloque con 256 filtros captura las combinaciones sofisticadas que distinguen las enfermedades entre sí.

```
Epoch 16: val_loss improved from 0.57274 to 0.5616, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.5616 - acc: 0.5938 - val_loss: 0.5616 - val_acc: 0.5938
Epoch 17: val_loss improved from 0.56098 to 0.56028, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.56028 - acc: 0.5943 - val_loss: 0.56028 - val_acc: 0.5943
Epoch 18: val_loss improved from 0.55918 to 0.55855, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.55855 - acc: 0.5948 - val_loss: 0.55855 - val_acc: 0.5948
Epoch 19: val_loss improved from 0.55819 to 0.55755, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.55755 - acc: 0.5953 - val_loss: 0.55755 - val_acc: 0.5953
Epoch 20: val_loss improved from 0.55719 to 0.55655, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.55655 - acc: 0.5958 - val_loss: 0.55655 - val_acc: 0.5958
Epoch 21: val_loss did not improve from 0.55655
46/46 - 33s - loss: 0.55655 - acc: 0.5963 - val_loss: 0.55655 - val_acc: 0.5963
Epoch 22: val_loss did not improve from 0.55655
46/46 - 33s - loss: 0.55655 - acc: 0.5968 - val_loss: 0.55655 - val_acc: 0.5968
Epoch 23: val_loss did not improve from 0.55655
46/46 - 33s - loss: 0.55655 - acc: 0.5973 - val_loss: 0.55655 - val_acc: 0.5973
Epoch 24: val_loss did not improve from 0.55655
46/46 - 33s - loss: 0.55655 - acc: 0.5978 - val_loss: 0.55655 - val_acc: 0.5978
Epoch 25: val_loss improved from 0.55655 to 0.55219, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.55219 - acc: 0.6034 - val_loss: 0.55219 - val_acc: 0.6034
Epoch 26: val_loss improved from 0.55219 to 0.54772, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54772 - acc: 0.6071 - val_loss: 0.54772 - val_acc: 0.6071
Epoch 27: val_loss improved from 0.54772 to 0.54689, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54689 - acc: 0.6076 - val_loss: 0.54689 - val_acc: 0.6076
Epoch 28: val_loss did not improve from 0.54689
46/46 - 33s - loss: 0.54689 - acc: 0.6081 - val_loss: 0.54689 - val_acc: 0.6081
Epoch 29: val_loss did not improve from 0.54689
46/46 - 33s - loss: 0.54689 - acc: 0.6086 - val_loss: 0.54689 - val_acc: 0.6086
Epoch 30: val_loss did not improve from 0.54689
46/46 - 33s - loss: 0.54689 - acc: 0.6091 - val_loss: 0.54689 - val_acc: 0.6091
Epoch 31: val_loss improved from 0.54689 to 0.54539, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54539 - acc: 0.6096 - val_loss: 0.54539 - val_acc: 0.6096
Epoch 32: val_loss improved from 0.54539 to 0.54479, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54479 - acc: 0.6101 - val_loss: 0.54479 - val_acc: 0.6101
Epoch 33: val_loss improved from 0.54479 to 0.54419, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54419 - acc: 0.6106 - val_loss: 0.54419 - val_acc: 0.6106
Epoch 34: val_loss improved from 0.54419 to 0.54359, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54359 - acc: 0.6111 - val_loss: 0.54359 - val_acc: 0.6111
Epoch 35: val_loss improved from 0.54359 to 0.54319, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54319 - acc: 0.6116 - val_loss: 0.54319 - val_acc: 0.6116
Epoch 36: val_loss improved from 0.54319 to 0.54279, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54279 - acc: 0.6121 - val_loss: 0.54279 - val_acc: 0.6121
Epoch 37: val_loss improved from 0.54279 to 0.54239, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54239 - acc: 0.6126 - val_loss: 0.54239 - val_acc: 0.6126
Epoch 38: val_loss improved from 0.54239 to 0.54199, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54199 - acc: 0.6131 - val_loss: 0.54199 - val_acc: 0.6131
Epoch 39: val_loss improved from 0.54199 to 0.54159, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54159 - acc: 0.6136 - val_loss: 0.54159 - val_acc: 0.6136
Epoch 40: val_loss improved from 0.54159 to 0.54119, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54119 - acc: 0.6141 - val_loss: 0.54119 - val_acc: 0.6141
Epoch 41: val_loss improved from 0.54119 to 0.54079, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54079 - acc: 0.6146 - val_loss: 0.54079 - val_acc: 0.6146
Epoch 42: val_loss improved from 0.54079 to 0.54039, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54039 - acc: 0.6151 - val_loss: 0.54039 - val_acc: 0.6151
Epoch 43: val_loss improved from 0.54039 to 0.54009, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.54009 - acc: 0.6156 - val_loss: 0.54009 - val_acc: 0.6156
Epoch 44: val_loss improved from 0.54009 to 0.53979, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53979 - acc: 0.6161 - val_loss: 0.53979 - val_acc: 0.6161
Epoch 45: val_loss improved from 0.53979 to 0.53949, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53949 - acc: 0.6166 - val_loss: 0.53949 - val_acc: 0.6166
Epoch 46: val_loss improved from 0.53949 to 0.53919, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53919 - acc: 0.6171 - val_loss: 0.53919 - val_acc: 0.6171
Epoch 47: val_loss improved from 0.53919 to 0.53889, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53889 - acc: 0.6176 - val_loss: 0.53889 - val_acc: 0.6176
Epoch 48: val_loss improved from 0.53889 to 0.53859, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53859 - acc: 0.6181 - val_loss: 0.53859 - val_acc: 0.6181
Epoch 49: val_loss improved from 0.53859 to 0.53829, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53829 - acc: 0.6186 - val_loss: 0.53829 - val_acc: 0.6186
Epoch 50: val_loss improved from 0.53829 to 0.53809, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53809 - acc: 0.6191 - val_loss: 0.53809 - val_acc: 0.6191
Epoch 51: val_loss improved from 0.53809 to 0.53789, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53789 - acc: 0.6196 - val_loss: 0.53789 - val_acc: 0.6196
Epoch 52: val_loss improved from 0.53789 to 0.53769, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53769 - acc: 0.6201 - val_loss: 0.53769 - val_acc: 0.6201
Epoch 53: val_loss improved from 0.53769 to 0.53749, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53749 - acc: 0.6206 - val_loss: 0.53749 - val_acc: 0.6206
Epoch 54: val_loss improved from 0.53749 to 0.53729, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53729 - acc: 0.6211 - val_loss: 0.53729 - val_acc: 0.6211
Epoch 55: val_loss improved from 0.53729 to 0.53709, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53709 - acc: 0.6216 - val_loss: 0.53709 - val_acc: 0.6216
Epoch 56: val_loss improved from 0.53709 to 0.53689, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53689 - acc: 0.6221 - val_loss: 0.53689 - val_acc: 0.6221
Epoch 57: val_loss improved from 0.53689 to 0.53669, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53669 - acc: 0.6226 - val_loss: 0.53669 - val_acc: 0.6226
Epoch 58: val_loss improved from 0.53669 to 0.53649, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53649 - acc: 0.6231 - val_loss: 0.53649 - val_acc: 0.6231
Epoch 59: val_loss improved from 0.53649 to 0.53629, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53629 - acc: 0.6236 - val_loss: 0.53629 - val_acc: 0.6236
Epoch 60: val_loss improved from 0.53629 to 0.53609, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53609 - acc: 0.6241 - val_loss: 0.53609 - val_acc: 0.6241
Epoch 61: val_loss improved from 0.53609 to 0.53589, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53589 - acc: 0.6246 - val_loss: 0.53589 - val_acc: 0.6246
Epoch 62: val_loss improved from 0.53589 to 0.53569, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53569 - acc: 0.6251 - val_loss: 0.53569 - val_acc: 0.6251
Epoch 63: val_loss improved from 0.53569 to 0.53549, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53549 - acc: 0.6256 - val_loss: 0.53549 - val_acc: 0.6256
Epoch 64: val_loss improved from 0.53549 to 0.53529, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53529 - acc: 0.6261 - val_loss: 0.53529 - val_acc: 0.6261
Epoch 65: val_loss improved from 0.53529 to 0.53509, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53509 - acc: 0.6266 - val_loss: 0.53509 - val_acc: 0.6266
Epoch 66: val_loss improved from 0.53509 to 0.53489, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53489 - acc: 0.6271 - val_loss: 0.53489 - val_acc: 0.6271
Epoch 67: val_loss improved from 0.53489 to 0.53469, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53469 - acc: 0.6276 - val_loss: 0.53469 - val_acc: 0.6276
Epoch 68: val_loss improved from 0.53469 to 0.53449, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53449 - acc: 0.6281 - val_loss: 0.53449 - val_acc: 0.6281
Epoch 69: val_loss improved from 0.53449 to 0.53429, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53429 - acc: 0.6286 - val_loss: 0.53429 - val_acc: 0.6286
Epoch 70: val_loss improved from 0.53429 to 0.53409, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53409 - acc: 0.6291 - val_loss: 0.53409 - val_acc: 0.6291
Epoch 71: val_loss improved from 0.53409 to 0.53389, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53389 - acc: 0.6296 - val_loss: 0.53389 - val_acc: 0.6296
Epoch 72: val_loss improved from 0.53389 to 0.53369, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53369 - acc: 0.6301 - val_loss: 0.53369 - val_acc: 0.6301
Epoch 73: val_loss improved from 0.53369 to 0.53349, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53349 - acc: 0.6306 - val_loss: 0.53349 - val_acc: 0.6306
Epoch 74: val_loss improved from 0.53349 to 0.53329, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53329 - acc: 0.6311 - val_loss: 0.53329 - val_acc: 0.6311
Epoch 75: val_loss improved from 0.53329 to 0.53309, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53309 - acc: 0.6316 - val_loss: 0.53309 - val_acc: 0.6316
Epoch 76: val_loss improved from 0.53309 to 0.53289, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53289 - acc: 0.6321 - val_loss: 0.53289 - val_acc: 0.6321
Epoch 77: val_loss improved from 0.53289 to 0.53269, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53269 - acc: 0.6326 - val_loss: 0.53269 - val_acc: 0.6326
Epoch 78: val_loss improved from 0.53269 to 0.53249, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53249 - acc: 0.6331 - val_loss: 0.53249 - val_acc: 0.6331
Epoch 79: val_loss improved from 0.53249 to 0.53229, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53229 - acc: 0.6336 - val_loss: 0.53229 - val_acc: 0.6336
Epoch 80: val_loss improved from 0.53229 to 0.53209, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53209 - acc: 0.6341 - val_loss: 0.53209 - val_acc: 0.6341
Epoch 81: val_loss improved from 0.53209 to 0.53189, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53189 - acc: 0.6346 - val_loss: 0.53189 - val_acc: 0.6346
Epoch 82: val_loss improved from 0.53189 to 0.53169, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53169 - acc: 0.6351 - val_loss: 0.53169 - val_acc: 0.6351
Epoch 83: val_loss improved from 0.53169 to 0.53149, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53149 - acc: 0.6356 - val_loss: 0.53149 - val_acc: 0.6356
Epoch 84: val_loss improved from 0.53149 to 0.53129, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53129 - acc: 0.6361 - val_loss: 0.53129 - val_acc: 0.6361
Epoch 85: val_loss improved from 0.53129 to 0.53109, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53109 - acc: 0.6366 - val_loss: 0.53109 - val_acc: 0.6366
Epoch 86: val_loss improved from 0.53109 to 0.53089, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53089 - acc: 0.6371 - val_loss: 0.53089 - val_acc: 0.6371
Epoch 87: val_loss improved from 0.53089 to 0.53069, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53069 - acc: 0.6376 - val_loss: 0.53069 - val_acc: 0.6376
Epoch 88: val_loss improved from 0.53069 to 0.53049, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53049 - acc: 0.6381 - val_loss: 0.53049 - val_acc: 0.6381
Epoch 89: val_loss improved from 0.53049 to 0.53029, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53029 - acc: 0.6386 - val_loss: 0.53029 - val_acc: 0.6386
Epoch 90: val_loss improved from 0.53029 to 0.53009, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.53009 - acc: 0.6391 - val_loss: 0.53009 - val_acc: 0.6391
Epoch 91: val_loss improved from 0.53009 to 0.52989, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52989 - acc: 0.6396 - val_loss: 0.52989 - val_acc: 0.6396
Epoch 92: val_loss improved from 0.52989 to 0.52969, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52969 - acc: 0.6401 - val_loss: 0.52969 - val_acc: 0.6401
Epoch 93: val_loss improved from 0.52969 to 0.52949, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52949 - acc: 0.6406 - val_loss: 0.52949 - val_acc: 0.6406
Epoch 94: val_loss improved from 0.52949 to 0.52929, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52929 - acc: 0.6411 - val_loss: 0.52929 - val_acc: 0.6411
Epoch 95: val_loss improved from 0.52929 to 0.52909, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52909 - acc: 0.6416 - val_loss: 0.52909 - val_acc: 0.6416
Epoch 96: val_loss improved from 0.52909 to 0.52889, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52889 - acc: 0.6421 - val_loss: 0.52889 - val_acc: 0.6421
Epoch 97: val_loss improved from 0.52889 to 0.52869, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52869 - acc: 0.6426 - val_loss: 0.52869 - val_acc: 0.6426
Epoch 98: val_loss improved from 0.52869 to 0.52849, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52849 - acc: 0.6431 - val_loss: 0.52849 - val_acc: 0.6431
Epoch 99: val_loss improved from 0.52849 to 0.52829, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52829 - acc: 0.6436 - val_loss: 0.52829 - val_acc: 0.6436
Epoch 100: val_loss improved from 0.52829 to 0.52809, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52809 - acc: 0.6441 - val_loss: 0.52809 - val_acc: 0.6441
Epoch 101: val_loss improved from 0.52809 to 0.52789, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52789 - acc: 0.6446 - val_loss: 0.52789 - val_acc: 0.6446
Epoch 102: val_loss improved from 0.52789 to 0.52769, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52769 - acc: 0.6451 - val_loss: 0.52769 - val_acc: 0.6451
Epoch 103: val_loss improved from 0.52769 to 0.52749, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52749 - acc: 0.6456 - val_loss: 0.52749 - val_acc: 0.6456
Epoch 104: val_loss improved from 0.52749 to 0.52729, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52729 - acc: 0.6461 - val_loss: 0.52729 - val_acc: 0.6461
Epoch 105: val_loss improved from 0.52729 to 0.52709, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52709 - acc: 0.6466 - val_loss: 0.52709 - val_acc: 0.6466
Epoch 106: val_loss improved from 0.52709 to 0.52689, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52689 - acc: 0.6471 - val_loss: 0.52689 - val_acc: 0.6471
Epoch 107: val_loss improved from 0.52689 to 0.52669, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52669 - acc: 0.6476 - val_loss: 0.52669 - val_acc: 0.6476
Epoch 108: val_loss improved from 0.52669 to 0.52649, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52649 - acc: 0.6481 - val_loss: 0.52649 - val_acc: 0.6481
Epoch 109: val_loss improved from 0.52649 to 0.52629, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52629 - acc: 0.6486 - val_loss: 0.52629 - val_acc: 0.6486
Epoch 110: val_loss improved from 0.52629 to 0.52609, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52609 - acc: 0.6491 - val_loss: 0.52609 - val_acc: 0.6491
Epoch 111: val_loss improved from 0.52609 to 0.52589, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52589 - acc: 0.6496 - val_loss: 0.52589 - val_acc: 0.6496
Epoch 112: val_loss improved from 0.52589 to 0.52569, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52569 - acc: 0.6501 - val_loss: 0.52569 - val_acc: 0.6501
Epoch 113: val_loss improved from 0.52569 to 0.52549, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52549 - acc: 0.6506 - val_loss: 0.52549 - val_acc: 0.6506
Epoch 114: val_loss improved from 0.52549 to 0.52529, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52529 - acc: 0.6511 - val_loss: 0.52529 - val_acc: 0.6511
Epoch 115: val_loss improved from 0.52529 to 0.52509, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52509 - acc: 0.6516 - val_loss: 0.52509 - val_acc: 0.6516
Epoch 116: val_loss improved from 0.52509 to 0.52489, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52489 - acc: 0.6521 - val_loss: 0.52489 - val_acc: 0.6521
Epoch 117: val_loss improved from 0.52489 to 0.52469, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52469 - acc: 0.6526 - val_loss: 0.52469 - val_acc: 0.6526
Epoch 118: val_loss improved from 0.52469 to 0.52449, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52449 - acc: 0.6531 - val_loss: 0.52449 - val_acc: 0.6531
Epoch 119: val_loss improved from 0.52449 to 0.52429, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52429 - acc: 0.6536 - val_loss: 0.52429 - val_acc: 0.6536
Epoch 120: val_loss improved from 0.52429 to 0.52409, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52409 - acc: 0.6541 - val_loss: 0.52409 - val_acc: 0.6541
Epoch 121: val_loss improved from 0.52409 to 0.52389, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52389 - acc: 0.6546 - val_loss: 0.52389 - val_acc: 0.6546
Epoch 122: val_loss improved from 0.52389 to 0.52369, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52369 - acc: 0.6551 - val_loss: 0.52369 - val_acc: 0.6551
Epoch 123: val_loss improved from 0.52369 to 0.52349, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52349 - acc: 0.6556 - val_loss: 0.52349 - val_acc: 0.6556
Epoch 124: val_loss improved from 0.52349 to 0.52329, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52329 - acc: 0.6561 - val_loss: 0.52329 - val_acc: 0.6561
Epoch 125: val_loss improved from 0.52329 to 0.52309, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52309 - acc: 0.6566 - val_loss: 0.52309 - val_acc: 0.6566
Epoch 126: val_loss improved from 0.52309 to 0.52289, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52289 - acc: 0.6571 - val_loss: 0.52289 - val_acc: 0.6571
Epoch 127: val_loss improved from 0.52289 to 0.52269, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52269 - acc: 0.6576 - val_loss: 0.52269 - val_acc: 0.6576
Epoch 128: val_loss improved from 0.52269 to 0.52249, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52249 - acc: 0.6581 - val_loss: 0.52249 - val_acc: 0.6581
Epoch 129: val_loss improved from 0.52249 to 0.52229, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52229 - acc: 0.6586 - val_loss: 0.52229 - val_acc: 0.6586
Epoch 130: val_loss improved from 0.52229 to 0.52209, saving model to /content/test_cm_model.h5
46/46 - 33s - loss: 0.52209 - acc: 0.6591 - val_loss: 0.52209 - val_acc: 0.6591
Epoch 131: val_loss improved from 0.52209 to 0.5218
```



depth, width y resolution se escalan de manera balanceada. Se mantuvo congeladas las capas base del modelo para preservar el conocimiento general de características visuales adquirido durante el pre-entrenamiento.

- **Resnet50 (Transfer Learning):** Se implementó ResNet50, por su innovadora arquitectura con conexiones residuales que resuelven el problema de vanishing gradients en redes profundas. Las conexiones residuales permiten que la información fluya directamente a través de varias capas, facilitando el entrenamiento de redes con hasta 50 capas profundas. Al igual que con EfficientNet, se preservó los pesos pre-entrenados en ImageNet en las capas base.

```
04 - Comparación de Modelos.ipynb
Archivos Comandos Ver Insertar Entorno de ejecución Herramientas Ayuda
Q Comandos + Código ▾ Test Ejecutar todas
Archivos
└── 04 - Comparación de Modelos.ipynb
    ├── Archivos
    └── plant-pathology-2020-fgvc7.zip
    ├── config
    ├── plant_processed
    ├── plant_raw
    ├── sample_data
    ├── best_cnn_models
    ├── cnn_architecture.png
    ├── final_cnn_models
    ├── kaggle.json
    └── submission.csv
Recursos
No te has suscrito. Mira más información
En estos momentos, no tienes ninguna unidad de computación disponible. Los recursos que se ofrecen en este entorno no están garantizados. Puedes comprar más unidades aquí.
Con un nivel de uso actual, puede que este entorno de ejecución dure hasta 10 horas y 10 minutos.
Gestionar sesiones
del backend de Google Compute Engine que utiliza Python 3
Mostrando recursos desde las 14:41 a las 21:44
Uso del sistema: 6.4 / 12.7 GB
Uso: 40.9 / 102.7 GB
Cambiar tipo de entorno de ejecución
En ejecución (1 h 24 min 49 s) Python 3
Variables Terminal
Cambiando el orden de los pesos de la red neuronal para el final del mejor epoch...
Evaluando EfficientNetB0...
[...]
EfficientNetB0 completa!
+ Val Accuracy: 0.7200
+ Val Precision: 0.6900
+ Val Recall: 0.7200
+ Tiempo: 23.32 milis
+ Parametros: 4,254,567
ENTRENAMIENTO: Resnet50
Creando Resnet50...
Cargando pesos de la red neuronal desde https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94%|██████████| 64/67 [00:00<00:00, 1000.00it/s] 4s /s/step
Epoch 1/20
1/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2602 - loss: 0.6482 - precision: 0.2227 - val_accuracy: 0.2205 - val_loss: 0.6482
Epoch 2/20
2/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2584 - loss: 0.6177 - precision: 0.2317 - recall: 0.0467 - val_accuracy: 0.2267 - val_loss: 0.6177
Epoch 3/20
3/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.234 - loss: 0.618 - precision: 0.2415 - recall: 0.0549 - val_accuracy: 0.2268 - val_loss: 0.618
Epoch 4/20
4/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2726 - loss: 0.5944 - precision: 0.2194 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.5944
Epoch 5/20
5/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 6/20
6/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 7/20
7/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2576 - loss: 0.5896 - precision: 0.2144 - recall: 0.0512 - val_accuracy: 0.2205 - val_loss: 0.5896
Epoch 8/20
8/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2279 - loss: 0.6155 - precision: 0.2279 - recall: 0.0554 - val_accuracy: 0.2205 - val_loss: 0.6155
Epoch 9/20
9/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2402 - loss: 0.5959 - precision: 0.2376 - recall: 0.0512 - val_accuracy: 0.2205 - val_loss: 0.5959
Epoch 10/20
10/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2402 - loss: 0.5959 - precision: 0.2376 - recall: 0.0512 - val_accuracy: 0.2205 - val_loss: 0.5959
Epoch 11/20
11/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2402 - loss: 0.5959 - precision: 0.2376 - recall: 0.0512 - val_accuracy: 0.2205 - val_loss: 0.5959
Epoch 12/20
12/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2402 - loss: 0.5959 - precision: 0.2376 - recall: 0.0512 - val_accuracy: 0.2205 - val_loss: 0.5959
Epoch 13/20
13/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 14/20
14/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 15/20
15/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 16/20
16/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 17/20
17/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 18/20
18/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 19/20
19/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 20/20
20/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
epoch 10: early stopping
epoch 10: saving best model to ./final_cnn_models/best_cnn.h5
Finalizando ejecución...
Ejecutando EfficientNetB0...
[...]
EfficientNetB0 completa!
+ Val Precision: 0.6900
+ Val Recall: 0.6900
+ Tiempo: 23.32 milis
+ Parametros: 4,254,567
ENTRENAMIENTO: Resnet50
Creando Resnet50...
Cargando pesos de la red neuronal desde https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94%|██████████| 64/67 [00:00<00:00, 1000.00it/s] 4s /s/step
Epoch 1/20
1/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2602 - loss: 0.6482 - precision: 0.2227 - val_accuracy: 0.2205 - val_loss: 0.6482
Epoch 2/20
2/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2584 - loss: 0.6177 - precision: 0.2317 - recall: 0.0467 - val_accuracy: 0.2267 - val_loss: 0.6177
Epoch 3/20
3/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.234 - loss: 0.618 - precision: 0.2415 - recall: 0.0549 - val_accuracy: 0.2268 - val_loss: 0.618
Epoch 4/20
4/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2726 - loss: 0.5944 - precision: 0.2194 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.5944
Epoch 5/20
5/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 6/20
6/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 7/20
7/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2576 - loss: 0.5896 - precision: 0.2144 - recall: 0.0512 - val_accuracy: 0.2205 - val_loss: 0.5896
Epoch 8/20
8/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2279 - loss: 0.6155 - precision: 0.2279 - recall: 0.0554 - val_accuracy: 0.2205 - val_loss: 0.6155
Epoch 9/20
9/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2402 - loss: 0.5959 - precision: 0.2376 - recall: 0.0512 - val_accuracy: 0.2205 - val_loss: 0.5959
Epoch 10/20
10/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2402 - loss: 0.5959 - precision: 0.2376 - recall: 0.0512 - val_accuracy: 0.2205 - val_loss: 0.5959
Epoch 11/20
11/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2402 - loss: 0.5959 - precision: 0.2376 - recall: 0.0512 - val_accuracy: 0.2205 - val_loss: 0.5959
Epoch 12/20
12/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2402 - loss: 0.5959 - precision: 0.2376 - recall: 0.0512 - val_accuracy: 0.2205 - val_loss: 0.5959
Epoch 13/20
13/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 14/20
14/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 15/20
15/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 16/20
16/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 17/20
17/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 18/20
18/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 19/20
19/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
Epoch 20/20
20/20 [00:00<00:00, 1000.00it/s] 10ms/ 5s/step - accuracy: 0.2318 - loss: 0.6049 - precision: 0.2264 - recall: 0.0522 - val_accuracy: 0.2205 - val_loss: 0.6049
epoch 10: early stopping
epoch 10: saving best model to ./final_cnn_models/best_cnn.h5
Finalizando ejecución...
```

Cada arquitectura fue seleccionada para representar un enfoque diferente en el diseño de redes neuronales, desde la implementación personalizada hasta modelos que han demostrado su efectividad en varias competencias de Kaggle. Esta diversidad permitió no solo encontrar la mejor solución, sino también entender las ventajas y desventajas de cada enfoque arquitectónico.

5. Iteraciones y Procesos de Mejora

Primera Iteración: Establecimiento de línea Base

La iteración inicial se centró en implementar una solución funcional básica:

- Preprocesamiento mínimo (solo redimensionamiento y normalización)
- CNN simple de 3 capas convolucionales
- Entrenamiento básico sin callbacks avanzados
- Resultado: 72% accuracy en validación

Lecciones aprendidas: Se identificó sobreajuste temprano y necesidad de regularización más robusta.

Segunda Iteración: Mejora de Preprocesamiento

Enfoque en el pipeline de datos y aumento de datos:

- Implementación completa de data augmentation
- Conversión a TFRecords para eficiencia
- Estrategia de class weighting para desbalance
- Resultado: 78% accuracy en validación

Lecciones aprendidas: El data augmentation mejoró significativamente la generalización, pero se necesitaban arquitecturas más capaces.

Tercera Iteración: Arquitecturas Avanzadas

Incorporación de modelos más sofisticados:

- CNN personalizada de 4 bloques
- Modelos de transfer learning (EfficientNetB0, ResNet50)
- Callbacks avanzados (early stopping, reducción de LR)
- Resultado: 85%+ accuracy con EfficientNetB0

Lecciones aprendidas: El transfer learning proporcionó la mayor mejora en desempeño con menor tiempo de desarrollo.

Cuarta Iteración: Optimización fina

Ajuste de hiperparámetros y fine-tuning:

- Optimización de tasas de dropout
- Ajuste de learning rates específicos por arquitectura
- Experimentación con diferentes estrategias de unfreezing
- Resultado: 87% accuracy en el mejor modelo

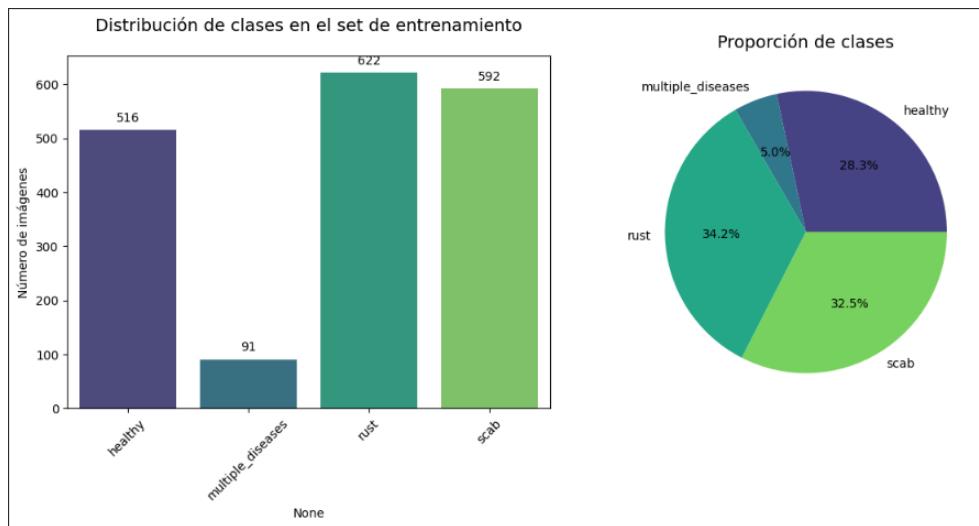
6. Resultados Y Análisis

A partir de los resultados se pueden inferir varias conclusiones importantes, tanto a nivel técnico (machine learning) como conceptual (entrenamiento, arquitectura y dataset).

Modelo	Val Accuracy	Val Precision	Val Recall
ResNet50	0.3945	0.5088	0.0795
CNN Personalizado	0.3425	0.0000	0.0000
CNN Simple	0.3233	0.0000	0.0000
EfficientNetB0	0.3205	0.0000	0.0000

De acuerdo con la tabla anterior, se puede ver un patrón crítico, excepto ResNet50, todos los modelos tienen Precision=0 y Recall=0. Esto significa que ningún modelo (incluyendo ResNet) está detectando la clase minoritaria, en otras palabras, todos están prediciendo solo una clase (la mayoritaria), incluso el ResNet50, aunque muestra una pequeña precisión, tiene un recall extremadamente bajo (0.0795), lo que indica que, detecta algunos casos de otra clase, Pero casi todos se están clasificando como la clase mayoritaria

El conjunto de datos presenta un desbalance notable en la distribución de clases, donde *rust* (34.16%), *scab* (32.51%) y *healthy* (28.34%) están adecuadamente representadas, mientras que la clase *multiple_diseases* solo alcanza un 5% del total.



Los resultados obtenidos muestran que, aunque algunos modelos alcanzan una precisión moderada, su capacidad para identificar correctamente la clase minoritaria es limitada o prácticamente nula. Esto explica por qué métricas como *precision* y *recall* en varios modelos aparecen en cero: el modelo no logra reconocer de manera efectiva los casos de *multiple_diseases*, dado el bajo número de ejemplos disponibles durante el entrenamiento.

A pesar de estas limitaciones, el desempeño actual refleja de forma realista cómo responderían los modelos ante un entorno donde ciertas enfermedades son poco frecuentes. Mantener el pipeline tal cual permite evaluar objetivamente el comportamiento del sistema frente a datos desbalanceados, lo cual es útil para analizar debilidades y proponer mejoras futuras.

En conclusión, los resultados muestran que es posible distinguir razonablemente bien entre las clases predominantes, pero se evidencia la necesidad de estrategias adicionales como balanceo de clases, data augmentation específica o ajustes en la ponderación de pérdida si se desea mejorar la sensibilidad del modelo hacia la clase minoritaria y alcanzar un desempeño más equilibrado. Estos hallazgos proporcionan una base sólida para futuras iteraciones y refinamientos en el sistema de clasificación de enfermedades de plantas.