|  |  |
| --- | --- |
| **Identificación de enfermedades de la planta de yuca a partir del análisis de las fotografías de sus hojas** | |
|  | |
|  | **Manolo S. Muñoz E.**  Nombre del Programa  Maestría en Ciencia de Datos  Área de trabajo final  Machine Learning / Deep Learning / Natural Language Processing / Deep Reinforcement Learning    **Tutor/a de TF**  Jordi de la Torre Gallart  **Profesor/a responsable de la asignatura**  Ismael Benito Altamirano    Fecha Entrega  29/12/2024 |

[Llicència de Creative Commons](http://creativecommons.org/licenses/by-nc-nd/3.0/es/)

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](http://creativecommons.org/licenses/by-nc-nd/3.0/es/)

**Licencias alternativas (elegir alguna de las siguientes y sustituir la de la página anterior)**

**A) Creative Commons:**

[Llicència de Creative Commons](http://creativecommons.org/licenses/by-nc-nd/3.0/es/)

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](http://creativecommons.org/licenses/by-nc-nd/3.0/es/)

[Llicència de Creative Commons](http://creativecommons.org/licenses/by-nc-sa/3.0/es/)

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-CompartirIgual [3.0 España de Creative Commons](http://creativecommons.org/licenses/by-nc-sa/3.0/es/)

[Llicència de Creative Commons](http://creativecommons.org/licenses/by-nc/3.0/es/)

Esta obra está sujeta a una licencia de Reconocimiento-NoComercial [3.0 España de Creative Commons](http://creativecommons.org/licenses/by-nc/3.0/es/)

[Llicència de Creative Commons](http://creativecommons.org/licenses/by-nd/3.0/es/)

Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada [3.0 España de Creative Commons](http://creativecommons.org/licenses/by-nd/3.0/es/)

[Llicència de Creative Commons](http://creativecommons.org/licenses/by-sa/3.0/es/)

Esta obra está sujeta a una licencia de Reconocimiento-CompartirIgual [3.0 España de Creative Commons](http://creativecommons.org/licenses/by-sa/3.0/es/)

[Llicència de Creative Commons](http://creativecommons.org/licenses/by/3.0/es/)

Esta obra está sujeta a una licencia de Reconocimiento [3.0 España de Creative Commons](http://creativecommons.org/licenses/by/3.0/es/)

**B) GNU Free Documentation License (GNU FDL)**

Copyright © 2024 – MANOLO S. MUÑOZ E.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.

A copy of the license is included in the section entitled "GNU Free Documentation License".

**C) Copyright**

© (el autor/a)

Reservados todos los derechos. Está prohibido la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la impresión, la reprografía, el microfilme, el tratamiento informático o cualquier otro sistema, así como la distribución de ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.

Ficha del Trabajo Final

|  |  |
| --- | --- |
| **Título del trabajo:** | Identificación de enfermedades de la planta de yuca a partir del análisis de las fotografías de sus hojas |
| **Nombre del autor/a:** | Manolo Muñoz Espinoza |
| **Nombre del Tutor/a de TF:** | Jordi de la Torre Gallart |
| **Nombre del/de la PRA:** | Ismael Benito Altamirano |
| **Fecha de entrega:** | 22/12/2024 |
| **Titulación o programa:** | Máster en Ciencia de Datos |
| **Área del Trabajo Final:** | Machine Learning / Deep Learning / Natural Language Processing / Deep Reinforcement Learning |
| **Idioma del trabajo:** | Castellano |
| **Palabras clave** | aprendizaje profundo, detección, imágenes |
| **Resumen del Trabajo** | |
| La yuca (*Manihot esculenta)* es un cultivo importante en muchas regiones del mundo, lamentablemente su rendimiento se ve afectado por la presencia de múltiples agentes fitopatógenos que afectan las hojas y por ende la actividad fotosintética, así como su productividad. Por ello se hace necesario contar con herramientas eficientes para la detección temprana de dichas enfermedades. La presente investigación tiene como objetivo identificar cuatro enfermedades foliares en plantas de yuca a partir del análisis de fotografías de hojas con síntomas, mediante el uso de visión por computadora y aprendizaje profundo (deep learning). Para dar solución a este reto, se utilizaron un conjunto de datos constituidos por 5,656 imágenes de etiquetadas, las cuales fueron recolectadas en Uganda, quienes fueron etiquetadas en cinco categorías. Se creará un modelo de clasificación de imágenes, el cual será entrenado por medio de redes neuronales convolucionales (CNNs) para identificar y diferenciar hojas enfermas y sanas.  Al modelo resultante se lo validará mediante métricas de desempeño. | |
| **Abstract** | |
| Cassava (Manihot esculenta) is a significant crop in many regions worldwide; unfortunately, its yield is impacted by multiple phytopathogens that affect the leaves and, consequently, photosynthetic activity and overall productivity. Therefore, efficient tools for the early detection of these diseases are necessary. This research aims to identify four foliar diseases in cassava plants through the analysis of leaf images showing symptoms, using computer vision and deep learning techniques. To address this challenge, a dataset comprising 5,656 labeled images was used, collected in Uganda and categorized into five classes. An image classification model will be developed and trained using convolutional neural networks (CNNs) to distinguish between healthy and diseased leaves. The resulting model will be validated using performance metrics. | |

Índice

Contenido

[1. Introducción 4](#_Toc185239215)

[1.1 Justificación del Interés y Relevancia de la Propuesta 4](#_Toc185239216)

[1.2 Explicación de la motivación personal 5](#_Toc185239217)

[1.3 Definición de los objetivos (principales y secundarios) 6](#_Toc185239218)

[1.3.1 Objetivo principal: 6](#_Toc185239219)

[1.3.2 Objetivos Secundarios: 6](#_Toc185239220)

[1.4 Descripción de la metodología empleada en el desarrollo del proyecto 6](#_Toc185239221)

[1.4.1 Preparación de Datos 6](#_Toc185239222)

[1.4.2 Desarrollo del Modelo de Red Neuronal Convolucional (CNN) 7](#_Toc185239223)

[1.4.3 Evaluación del Modelo 7](#_Toc185239224)

[1.4.4 Análisis de Resultados y Documentación 8](#_Toc185239225)

[2. Estado del arte o análisis de mercado del proyecto 9](#_Toc185239226)

[2.1 Planteamiento y competencias 9](#_Toc185239227)

[3. Proceso de implementación 12](#_Toc185239228)

[3.1 Estructura del conjunto de datos 12](#_Toc185239229)

[3.2 Propiedades de las imágenes 13](#_Toc185239230)

[3.3 Análisis y Diseño de la Solución 15](#_Toc185239231)

[3.3.1 Introducción 15](#_Toc185239232)

[3.3.2 Preparación de los Datos 18](#_Toc185239233)

[3.3.3 Preprocesamiento de los datos de entrada 18](#_Toc185239234)

[4. Implementación de la Solución 20](#_Toc185239235)

[4.1 Modelo 1- Red Neuronal Convolucional 20](#_Toc185239236)

[4.1.1 Definición del modelo 20](#_Toc185239237)

[4.1.2 Compilación del Modelo 22](#_Toc185239238)

[4.1.3 Entrenamiento del Modelo 22](#_Toc185239239)

[4.2 Modelo 2: Red Neuronal Convolucional con Dropout 23](#_Toc185239240)

[4.3 Modelo 3: Transfer Learning 24](#_Toc185239241)

[4.3.1 Configuración del modelo EfficientNetB4 24](#_Toc185239242)

[4.3.2 Definición del modelo 25](#_Toc185239243)

[4.3.3 Compilación del modelo 26](#_Toc185239244)

[4.3.4 Callbacks: optimizando el proceso de ajuste 26](#_Toc185239245)

[4.3.5 Entrenamiento del Modelo 27](#_Toc185239246)

[5. Resultados 29](#_Toc185239247)

[5.1 Métricas de evaluación 29](#_Toc185239248)

[5.1.1 Exactitud (Accuracy) por clase 29](#_Toc185239249)

[5.1.2 Sensibilidad o Recall por clase 29](#_Toc185239250)

[5.1.3 Precisión (Precision) por clase 29](#_Toc185239251)

[5.1.4 F1 Score por clase 30](#_Toc185239252)

[5.2 Métricas Modelo 1 30](#_Toc185239253)

[5.3 Métricas Modelo 2 32](#_Toc185239254)

[5.3 Métricas Modelo 3 34](#_Toc185239255)

[6. Planificación o plan de investigación del proyecto 36](#_Toc185239256)

[7. Glosario 37](#_Toc185239257)

[8. Bibliografía 37](#_Toc185239258)

[9. Anexos 39](#_Toc185239259)

Lista de Figuras

[Figura 1. Distribución de clases 13](#_Toc185726664)

[Figura 2. Resolución de las imágenes 15](#_Toc185726665)

[Figura 3. Arquitectura básica de una RNC 16](#_Toc185726666)

[Figura 4 Arquitectura RNC 18](#_Toc185726667)

[Figura 5 Escalado de modelos 18](#_Toc185726668)

[Figura 6 Tamaño del Modelo vs Imagenet Accuracy 19](#_Toc185726669)

[Figura 7 Parámetros de cada arquitectura de EfficientNet 20](#_Toc185726670)

[Figura 8 Red de referencia EffientNet-B0 20](#_Toc185726671)

[Figura 9. Accuracy modelo CNN 34](#_Toc185726672)

[Figura 10. Pérdida modelo CNN 35](#_Toc185726673)

[Figura 11. Métricas modelo CNN 35](#_Toc185726674)

[Figura12. Gráfica matriz de confusión, modelo CNN 37](#_Toc185726675)

[Figura 13. Accuracy modelo CNN con Dropout 38](#_Toc185726676)

[Figura 14. Pérdida modelo CNN con Dropout 39](#_Toc185726677)

[Figura 15. Métricas modelo CNN con Dropout 39](#_Toc185726678)

[Figura 16. Gráfica matriz de confusión, modelo CNN con Dropout 41](#_Toc185726679)

[Figura 17. Accuracy modelo EfficientNet-B0 42](#_Toc185726680)

[Figura 18. Pérdida modelo EfficientNet-B0 43](#_Toc185726681)

[Figura 19. Métricas modelo EfficientNet-B0 43](#_Toc185726682)

[Figura 20. Gráfica matriz de confusión, modelo EfficientNet-B0 45](#_Toc185726683)

[Figura 21. Planificación 49](#_Toc185726684)

# Introducción

La yuca es un cultivo clave para la seguridad alimentaria en África, es cultivada por pequeños agricultores porque puede soportar condiciones difíciles. Al menos el 80% de las granjas familiares en África subsahariana cultivan esta raíz almidonada, pero las enfermedades virales son las principales causas de los bajos rendimientos productivos. Con la ayuda de la ciencia de datos, es posible identificar enfermedades comunes para que puedan tratarse. Los métodos existentes de detección de enfermedades requieren que los agricultores soliciten la ayuda de expertos agrícolas financiados por el gobierno para inspeccionar y diagnosticar visualmente las plantas. Este método es laborioso y costoso.

En este proyecto se utilizará un conjunto de datos de 21,367 imágenes etiquetadas recolectadas en Uganda. La mayoría de las imágenes se obtuvieron de agricultores que tomaron fotos de sus cultivos, estas fueron anotadas por expertos del Instituto Nacional de Investigación de Recursos de Cultivos (NaCRRI) en colaboración con el laboratorio de IA de la Universidad de Makerere, Kampala. Está en un formato que representa de manera más realista lo que los agricultores necesitarían diagnosticar en la vida real.

El proyecto consiste en clasificar cada imagen de yuca en cuatro categorías de enfermedades y una quinta categoría que indica una hoja sana. Este algoritmo puede permitir a los agricultores identificar rápidamente las plantas enfermas, lo que podría salvar sus cultivos antes de que se causen daños irreparables.

## 1.1 Justificación del Interés y Relevancia de la Propuesta

La yuca es un cultivo importante en la seguridad alimentaria y en los ingresos económicos de millones de personas en América Latina, África y Asia, en el año 2022 la producción a nivel mundial fue de 26,3 Mha, con un rendimiento de 11 t /ha (Cock & Connor, 2021). Sin embargo, la productividad de este cultivo se ve amenazada por enfermedades (Igwe, 2023), que deben ser controladas a tiempo, caso contrario pueden ocasionar pérdidas en el rendimiento y la calidad del producto.

La detección temprana y precisa de las enfermedades, es crucial para mitigar los efectos en el cultivo, lamentablemente, no se dispone de herramientas tecnológicas que ayuden con este problema, se sigue dependiendo del factor humano mediante la inspección manual, lo que conlleva a un gran costo y proclive a errores humanos.

El desarrollo de un sistema automatizado para identificar enfermedades en la yuca, basado en imágenes, representa una solución innovadora y eficiente frente a este problema.

La combinación de visión por computadora y deep learning, tiene el potencial de proporcionar un diagnóstico preciso y en tiempo real, lo que permitirá a los agricultores obtener información inmediata sobre la salud de su cultivo y tomar las medidas correctivas de manera oportuna.

Esta propuesta está alineada al auge de la agricultura de precisión, ya que mediante el uso de tecnologías avanzadas buscamos optimizar los recursos y mejorar la productividad agrícola, lo que conlleva a un impacto positivo en la economía agrícola.

La relevancia de esta propuesta radica en su potencial para mejorar significativamente la detección de enfermedades en un cultivo importante, lo que contribuye al bienestar económico y alimentario de los agricultores dedicados a este cultivo, además que impulsa el uso de inteligencia artificial en el sector agrícola.

## 1.2 Explicación de la motivación personal

Como docente de la Universidad Técnica de Ambato, de la Facultad de Ciencias Agropecuarias, he tenido la oportunidad de observar los desafíos que enfrentan los agricultores en el manejo de diferentes cultivos. A lo largo de mi experiencia académica, he presenciado como las enfermedades foliares pueden causar mucho daño en plantaciones enteras, afectado la producción y el sustento de muchas familias rurales que dependen de sus cultivos.

Bajo estas circunstancias, me he interesado profundamente en el desarrollo de soluciones para contrarrestar el impacto de enfermedades y mejorar la capacidad de respuesta de los agricultores.

La idea de aplicar técnicas avanzadas de inteligencia artificial y aprendizaje profundo en la detección de enfermedades surge, en parte, de mi deseo de contribuir a la mejora del sector agrícola ecuatoriano y mundial, combinando mi formación técnica y mi experiencia en la docencia. En un mundo cada vez más digitalizado, considero que la fusión de la agricultura y la tecnología es una de las herramientas más poderosas que podemos ofrecer a nuestros estudiantes y a la comunidad agrícola en general.

Asimismo, la motivación detrás de este trabajo radica en la posibilidad de llevar el conocimiento teórico que imparto en las aulas a una aplicación práctica y tangible. Quiero demostrar a mis estudiantes y colegas que la ciencia de datos no es un campo aislado, sino que puede integrarse eficazmente en la agricultura para resolver problemas reales. El proyecto me permitirá no solo profundizar en mis conocimientos sobre técnicas de visión por computadora y aprendizaje profundo, sino también abrir una nueva vía de investigación dentro de la facultad, donde podremos desarrollar y promover tecnologías que transformen la manera en que se manejan las enfermedades de los cultivos.

Por último, creo firmemente en el poder de la educación aplicada y en la importancia de inspirar a las nuevas generaciones de profesionales agropecuarios a que exploren y adopten las herramientas tecnológicas disponibles. Este proyecto tiene el potencial de servir como un modelo de cómo la ciencia de datos puede ser una aliada clave en la mejora de la agricultura sostenible y eficiente, un mensaje que me esfuerzo por transmitir en mi labor docente.

## 1.3 Definición de los objetivos (principales y secundarios)

### 1.3.1 Objetivo principal:

Identificar enfermedades foliares que afectan a (*Manihot esculenta*)a partir del análisis profundo de imágenes mediante redes neuronales convolucionales para la detección temprana de enfermedades en campo.

### 1.3.2 Objetivos Secundarios:

* Preprocesar las imágenes de enfermedades foliares mediante la normalización y aumento de datos para el entrenamiento del modelo.
* Entrenar un modelo de redes neuronales convolucionales (CNN) mediante aprendizaje transferido para mejorar la precisión y reducir el tiempo de entrenamiento.
* Evaluar el accuracy y la pérdida, para validar la efectividad en la identificación correcta de enfermedades foliares en la yuca del modelo creado.

## 1.4 Descripción de la metodología empleada en el desarrollo del proyecto

El desarrollo de este proyecto seguirá una metodología estructurada en varias etapas, desde la preparación de los datos hasta la implementación del sistema final. A continuación, se describen las fases principales:

### 1.4.1 Preparación de Datos

Una vez recolectadas las imágenes, se llevará a cabo un proceso de preprocesamiento de los datos, que incluirá las siguientes acciones:

**Normalización de Imágenes:** Para garantizar que todas las imágenes tengan las mismas dimensiones y características básicas, las imágenes se normalizarán a un tamaño estándar.

**Aumento de Datos (Data Augmentation):** Para incrementar la diversidad de las imágenes, se aplicarán técnicas de aumento de datos, esto con el objetivo de aumentar el conjunto de datos. Esto ayudará a mejorar la capacidad del modelo de generalizar.

**División del Conjunto de Datos:** El conjunto de datos se dividirá en dos subconjuntos: entrenamiento (80%) y validación (20%).

### 1.4.2 Desarrollo del Modelo de Red Neuronal Convolucional (CNN)

La segunda fase consistirá en el diseño y entrenamiento del modelo basado en redes neuronales convolucionales (CNNs) para la clasificación de las enfermedades. El proceso se llevará a cabo en los siguientes pasos:

**Transfer Learning:** Se utilizará un modelo preentrenado, que ha demostrado ser eficaz en la clasificación de imágenes. Este modelo será adaptado a la tarea específica de identificación de enfermedades foliares de la yuca mediante la modificación de sus capas superiores.

**Entrenamiento del Modelo:** El modelo será entrenado con el conjunto de datos de entrenamiento, ajustando hiperparámetros clave como el tamaño del batch, el learning rate y el número de épocas para mejorar su rendimiento.

### 1.4.3 Evaluación del Modelo

Una vez entrenado, el modelo será evaluado utilizando el conjunto de datos de prueba. La evaluación se llevará a cabo con base en las siguientes métricas:

**Precisión (Accuracy):** La proporción de predicciones correctas sobre el total de predicciones realizadas.

**Recall:** La capacidad del modelo para detectar correctamente las imágenes de hojas enfermas.

**F1-Score:** Una métrica combinada que tiene en cuenta tanto la precisión como el recall.

Si las métricas de evaluación no cumplen con los estándares esperados, se ajustarán los hiperparámetros del modelo y se realizarán iteraciones adicionales de entrenamiento.

### 1.4.4 Análisis de Resultados y Documentación

Finalmente, se realizará un análisis exhaustivo de los resultados obtenidos en cada fase del proyecto. Se documentarán tanto los aciertos como los desafíos encontrados a lo largo del proceso, y se proporcionarán sugerencias para trabajos futuros que busquen mejorar el modelo o expandir la metodología a otros cultivos.

**Tecnologías y Herramientas que se utilizarán**

Python para el desarrollo del modelo de redes neuronales utilizando librerías como TensorFlow y Keras.

Kaggle, Google Colab o servidores GPU para el entrenamiento del modelo.

Esta metodología asegura un enfoque riguroso y sistemático, centrado en el uso de técnicas de vanguardia en inteligencia artificial, con un énfasis práctico en la aplicación real de los resultados.

# Estado del arte o análisis de mercado del proyecto

## 2.1 Planteamiento y competencias

La yuca (*Manihot esculenta*), comúnmente llamada mandioca o cassava, es un cultivo tropical de raíz comestible originario de América del Sur. Esta planta, que pertenece a la familia Euphorbiaceae, se ha convertido en un alimento básico en regiones tropicales y subtropicales de África, Asia y América Latina, proporcionando una fuente significativa de energía alimentaria (A. Burns et al., 2010), es considerado un alimento básico en muchas regiones, por lo que es consumida por más de un billón de personas en el mundo (Adebayo, 2023).

Los tubérculos de la yuca, que destacan por su alto contenido de almidón, son una fuente económica de carbohidratos para millones de personas en regiones con acceso limitado a otros alimentos. Sin embargo, la yuca es baja en proteínas y otros nutrientes esenciales, por lo que su consumo suele complementarse con otros alimentos para mejorar la calidad nutricional de la dieta(Montagnac et al., 2009).

Un aspecto crucial de la yuca es que ciertas variedades contienen compuestos tóxicos cianogénicos, que requieren procesamiento para hacerla segura para el consumo. Estos compuestos, si no son eliminados adecuadamente mediante técnicas de preparación como la cocción, pueden provocar efectos adversos en la salud, razón por la cual se recomienda procesar la yuca antes de ingerirla (A. E. Burns et al., 2012).

El aprovechamiento de la yuca se presenta en diversas formas, una de ellas es en la producción de etanol, alcohol, medicinas alimento para animales entre otros productos (Fathima et al., 2023).

Asimismo, la cáscara de la yuca es otro subproducto, se lo usa para en la producción de bio-briquettes (Fitriyano et al., 2023).

A partir de los residuos de la yuca, se obtuvo biocarbono, el cual se lo puede utilizar para incrementar el pH del suelo, además el alto contenido energético hace que sea óptimo para aplicaciones energéticas (Alonso-Gómez et al., 2024).

También se puede aprovechar las hojas de yuca, las cuales se pueden convertir en abono con aproximadamente 3% de nitrógeno, además se puede usar como fuente de nitrógeno para fermentación de etanol (Karuna et al., 2023)

La producción de yuca para consumo humano a nivel mundial es importante, solo está por debajo del arroz, trigo y arroz; por ejemplo, en el año 2021, Nigeria llegó a una producción de 60.8 millones de toneladas métricas de este cultivo (ReportLinker, 2022).

Sin embargo, diversas enfermedades afectan gravemente su producción, lo que conlleva a pérdidas de consideración en el rendimiento agrícola(Igwe, 2023).

Enfermedades como el tizón bacteriano, raya marrón, moteado verde, mosaico, son algunas de las enfermedades que afectan este cultivo.

El tizón bacteriano en la yuca es causado principalmente por Xanthomonas axonopodis y se propaga a través de lluvias intensas, salpicaduras de agua y heridas en las plantas. Esta bacteria infecta mediante herramientas contaminadas y fragmentos vegetativos, lo cual facilita la diseminación en campos húmedos o durante el riego (Restrepo & Verdier, 1997).

La raya marrón de la yuca es causada por el virus de la raya marrón de la yuca (CsCMV) y se transmite a través de los materiales vegetativos infectados. Además, puede propagarse a través de prácticas agrícolas como el uso de esquejes infectados, facilitando la transmisión entre plantas (Legg & Fauquet, 2004).

El moteado verde es una enfermedad viral que afecta principalmente a las hojas de la yuca. Este virus se propaga mediante áfidos, en particular Aphis gossypii, que son insectos vectores comunes en climas tropicales. La transmisión se facilita cuando los áfidos se alimentan de plantas infectadas y luego se trasladan a plantas sanas (Thresh & Cooter, 2005).

El mosaico de la yuca es causado por varios virus de la familia Geminiviridae y se transmite principalmente mediante áfidos, especialmente Bemisia tabaci. Esta enfermedad se propaga rápidamente a través de esquejes infectados y por contacto con herramientas contaminadas, afectando tanto la producción como la calidad del cultivo (Legg & Fauquet, 2004).

En este contexto, un aliado para la automatización en la detección de enfermedades es la agricultura de precisión, que en las últimas décadas ha ganado importancia mediante el uso de tecnologías avanzadas, por lo que se presenta como una solución óptima y prometedora (Karunathilake et al., 2023).

En los últimos años, la aplicación de inteligencia artificial (IA), y en particular el uso de visión por computadora ha mostrado un gran potencial para la agricultura de precisión(Ghazal et al., 2024).

Mediante el uso herramientas de redes neuronales (RN), RN convolucionales (RNC) se analizan datos visuales masivos, facilitando la identificación y clasificación de problemas como enfermedades foliares y plagas (Paul et al., 2023).

Este enfoque automatizado mejora la precisión en el diagnóstico, así también reduce la intervención humana, lo que es crucial en regiones con recursos y accesos limitados.

Diversas investigaciones han abordado problemáticas en la detección de enfermedades foliares mediante RNC. En un estudio (Shrestha et al., 2020), mediante imágenes de muestra, se detectaron enfermedades foliares mediante diferentes algoritmos basados en RCN, en las que se etiquetaron 12 enfermedades y 3 etiquetas con hojas sanas, logrando una precisión en la prueba e in 88.80%.

Similarmente (Marzougui et al., 2020), utilizó Deep learning con la arquitectura ResNet y data augmentation utilizando imágenes con dos clases, hojas enfermas y sanas, los resultados obtenidos fueron del 98.96% de precisión.

Con relación a la yuca, (Alado, 2024) utilizó Deep learning usando la técnica de Squeezenet RNC para clasificar enfermedades basados en imágenes, etiquetadas con cuatro categorías, esta investigación logró una precisión del 92% además la matriz de confusión logró un buen desempeño en la identificación de hojas enfermas.

Mediante la fusión de las técnicas de Random Forest y RNC, (Rajora et al., 2024) presentó la identificación de enfermades de yuca a nivel de raíz con un dataset de 6,820 imágenes, creando un modelo sofisticado alcanzando una precisión del 97.79%.

La arquitectura EfficientNet es ampliamente utilizada en la clasificación automática de imágenes, al utilizar artificial driving-EfficientNet (AD-ENet), evaluaron el desempeño en términos de diferentes medidas de evaluación, obteniendo un accuracy del 99.91% (Kotwal et al., 2024).

Aunque los avances en el uso de redes neuronales para la identificación de enfermedades agrícolas han sido significativos, aún existen varias limitaciones. En primer lugar, la mayoría de los estudios se han centrado en cultivos de alto valor comercial como el maíz y el tomate, dejando de lado cultivos esenciales como la yuca.

Además, muchos de los modelos desarrollados requieren gran capacidad de procesamiento, lo que dificulta su implementación en dispositivos móviles o entornos con recursos limitados. Asimismo, la mayoría de los conjuntos de datos utilizados provienen de imágenes tomadas en condiciones controladas, lo que puede afectar la generalización de los modelos en condiciones de campo.

Dado el impacto económico de las enfermedades de la yuca y la escasez de investigaciones centradas en este cultivo, se hace evidente la necesidad de desarrollar sistemas automáticos que faciliten su diagnóstico en campo.

Mi investigación se centrará en la identificación de enfermedades foliares de la yuca mediante un conjunto de datos extenso y diverso.

# Proceso de implementación

Vamos a analizar y conocer el conjunto de datos, previo al diseño de la solución.

## 3.1 Estructura del conjunto de datos

El conjunto de datos de entrada consta de 5,656 imágenes etiquetadas procedentes de Uganda, de plantaciones de yuca.

Los datos constan de dos carpetas, una carpeta de entrenamiento que contiene 5 subcarpetas que contienen las imágenes respectivas para las diferentes 5 clases (cmd, healthy, cgm, cbsd, cbb) y una carpeta de prueba que contiene 3774 imágenes de prueba.

1. cmd: 2658
2. cbb: 466
3. cbsd: 1443
4. healthy: 316
5. cgm: 773

|  |
| --- |
| Gráfico, Gráfico de barras  Descripción generada automáticamente |

Figura 1. Distribución de clases

El enlace del conjunto de datos es el siguiente:

* <https://www.kaggle.com/c/cassava-disease/data>

## 3.2 Propiedades de las imágenes

Se extraen algunas propiedades de las imágenes, esto nos servirá para poder manipular de mejor manera en la etapa del diseño e implementación.

Resolución de las imágenes:

* Subcarpeta: cmd
* Tamaño mínimo: (499, 500)
* Tamaño máximo: (1622, 500)
* Subcarpeta: cbb
* Tamaño mínimo: (424, 500)
* Tamaño máximo: (500, 1071)
* Subcarpeta: cbsd
* Tamaño mínimo: (434, 500)
* Tamaño máximo: (500, 1000)
* Subcarpeta: healthy
* Tamaño mínimo: (500, 500)
* Tamaño máximo: (1000, 500)
* Subcarpeta: cgm
* Tamaño mínimo: (456, 500)
* Tamaño máximo: (1386, 500)

|  |
| --- |
|  |

Figura 2. Resolución de las imágenes

## 3.3 Análisis y Diseño de la Solución

### 3.3.1 Introducción

Ahora abordaremos el tipo de solución que se le dará a la clasificación automática de imágenes, en el capítulo 2 se mencionaron la Redes Neuronales Convolucionales que son el pilar fundamental para solucionar problemas de clasificación de imágenes, así como arquitecturas como Resnet, que también as solucionar el problema descrito.

Existen algunas arquitecturas de redes neuronales para la clasificación de imágenes, para este trabajo se ha considerado la arquitectura conocida como EfficientNet.

Es necesario abordar las Redes Neuronales Convolucionales (CNNs), las cuales han revolucionado el campo del deep learning, diseñada específicamente para tareas de visión por computadora. Esta arquitectura se inspira en el procesamiento visual del cerebro humano, estas redes son capaces de extraer y aprender patrones jerárquicos, mediante la aplicación de filtros o kernels con el fin de detectar patrones, bordes, formas, texturas. a partir de los datos, desde bordes simples hasta estructuras complejas como objetos completos (Lecun et al., 2015) (Goodfellow, 2016).

Una Red Neuronal Convolucional básica (RNC) trabaja con 3 tipos de capas (Alsaleh & Perkgoz, 2023):

* Capa convolucional
* Capa de agrupamiento (pooling)
* Capas totalmente conectadas (Fully.Connnected Layers)

|  |
| --- |
|  |

Figura 3. Arquitectura básica de una RNC

En la figura 4 (Sahai, 2024), se detalla de mejor manera los componentes claves de una Red Neuronal Convolucional que son (Seseri, 2023) :

**Capas Convolucionales:**

* Filtros (Kernels): Pequeñas matrices que se deslizan sobre la imagen de entrada para detectar características específicas como bordes, texturas y patrones.
* Operación de Convolución: Multiplica el filtro con regiones de la imagen de entrada, produciendo mapas de características que resaltan la presencia de las características detectadas.
* Invariancia a Traslaciones: Gracias a la convolución, las CNNs pueden reconocer características independientemente de su posición en la imagen.

**Funciones de Activación:**

* ReLU (Unidad Lineal Rectificada): Introduce no linealidad en el modelo, permitiendo que la red aprenda relaciones complejas. La función ReLU activa solo las neuronas con valores positivos, acelerando el entrenamiento y mitigando problemas como el desvanecimiento del gradiente.

**Capas de Agrupamiento (Pooling):**

* Max Pooling y Average Pooling: Reducen la dimensionalidad de los mapas de características, disminuyendo la cantidad de parámetros y el costo computacional, además de ayudar a controlar el sobreajuste.
* Reducción de la Resolución Espacial: Preserva las características más importantes mientras reduce la complejidad de los datos.

**Capas Totalmente Conectadas (Fully Connected Layers):**

* Clasificación Final: Después de las capas convolucionales y de pooling, las capas totalmente conectadas interpretan las características extraídas para realizar tareas de clasificación o regresión.
* Integración de Características: Combinan las características aprendidas para identificar patrones complejos y tomar decisiones finales sobre la salida del modelo.

|  |
| --- |
|  |

Figura 4. Arquitectura RNC

### Arquitectura EfficientNet

EfficientNet es una arquitectura de red neuronal que optimiza el uso de recursos computacionales, esta arquitectura introduce un enfoque sistemático para escalar redes neuronales, optimizando simultáneamente la profundidad, anchura y resolución de la red (Tan & Le, 2019).

|  |
| --- |
|  |

Figura 5. Escalado de modelos

La mejora de esta arquitectura es que realiza una gran innovación en el tratamiento de imágenes, ya que escala tres factores clave al mismo tiempo:

1. **Profundidad (depth α):** Esto se refiere al número de capas que tiene la red neuronal, mientras más profunda, el modelo puede aprender características de las imágenes con mucho más detalle.
2. **Ancho (width β):** Esto se refiere al número de neuronas que existen en cada capa. Un mayor ancho permite captar más información dentro de una misma capa.
3. **Resolución (resolution ϒ):** Se refiere al nivel de detalle de las imágenes que analiza la red, imágenes con mucha más resolución, permite detectar detalles más finos.

Antes de esta arquitectura, se escalaban estos tres factores de manera separada. Con EfficientNet se propone una estrategia balanceada que lleva el nombre del método de escalado compuesto, lo que permite ajustar los tres factores de forma conjunta y óptima.

|  |
| --- |
|  |

Figura 6. Tamaño del Modelo vs Imagenet Accuracy

Al hacer uso de EfficientNet, estamos utilizando una técnica llamada transfer learning, la cual consiste en utilizar modelos previamente entrenados, lo que nos ayuda a mejorar el tiempo de entrenamiento y mejorar el desempeño de nuestro propio modelo (Punuri et al., 2023).

Cada una de las arquitecturas de EfficientNet tiene sus respectivos parámetros, sobre todo la resolución (Merino et al., 2021).

|  |
| --- |
|  |

Figura 7. Parámetros de cada arquitectura de EfficientNet

Para este caso, se utiliza la arquitectura B0, con una resolución de entrada de (224,224).

|  |
| --- |
|  |

Figura 8. Red de referencia EffientNet-B0

### 3.3.2 Preparación de los Datos

Se establecieron hiperparámetros esenciales para entrenar un modelo de aprendizaje profundo en el análisis de imágenes, en función de la arquitectura EfficientNet-B0, para cada uno de los modelos, la resolución de imagen de entrada será de 224x224 pixeles.

* BATCH\_SIZE = 32:

El batch size (tamaño del lote) es el número de muestras o ejemplos que se procesan juntos en una única iteración durante el entrenamiento de una red neuronal.

Es un hiperparámetro que define cuántas muestras verá la red antes de realizar una actualización de los pesos, determina cuánta memoria se utilizará durante el entrenamiento, afecta directamente a la dinámica del proceso de optimización

El tamaño de 32 es el balance perfecto entre memora y velocidad, es suficiente mente grande para permitir un aprovechamiento del paralelismo en la GPU que se ha usa, además al ser una potencia de 2 optimiza los cálculos computacionales.

### 3.3.3 Preprocesamiento de los datos de entrada

El aumento de datos (data augmentation) es una transformación de los datos de entrada, lo que permite tener más datos para entrenar el modelo, aumentando la robustez del modelo, al expandir efectivamente el conjunto de datos y evitar el sobreajuste. Al entrenar con datos variados, la red es capaz de generalizar mejor a nuevos datos.

A continuación de detalla las transformaciones a realizarse, este aumento artificial de datos lo realizaremos con la función ImageDataGenerator.

#### 3.3.3.1 Transformaciones para aumento de datos

* ImageDataGenerator: Es una clase de Keras que permite realizar data augmentation y preprocesamiento en tiempo real.
* rescale=1.0/255

Factor de escala que se aplica a cada píxel de la imagen, normaliza los valores de píxeles del rango [0-255] al rango [0-1], esto lo hacemos debido que las redes neuronales funcionan mejor con valores normalizados cercanos a cero, mejorando la estabilidad numérica y la velocidad de convergencia

* rotation\_range=20

Rango en grados para rotaciones aleatoria, rota la imagen aleatoriamente entre -20 y +20 grados, esto ayuda a que el modelo sea robusto ante diferentes orientaciones de los objetos en las imágenes.

* width\_shift\_range=0.2 y height\_shift\_range=0.2

Fracción del ancho/alto total para desplazar la imagen, desplaza la imagen horizontalmente y verticalmente hasta un 20% de su dimensión, esto hace el modelo más robusto a objetos que aparecen en diferentes posiciones de la imagen.

* shear\_range=0.2

Intensidad de la transformación de cizallamiento, aplica una transformación que inclina la imagen, esto ayuda con el reconocimiento de objetos desde diferentes ángulos y perspectivas.

* zoom\_range=0.2

Rango para el zoom aleatorio, aplica zoom aleatorio entre [0.8, 1.2], esto ayuda mejora la robustez del modelo ante objetos de diferentes tamaños.

* horizontal\_flip=True

Volteo horizontal aleatorio de las imágenes, voltea aleatoriamente las imágenes de izquierda a derecha, esto es útil para objetos que pueden aparecer en ambas orientaciones.

* fill\_mode='nearest'

Estrategia para rellenar píxeles nuevos tras las transformaciones, rellena los espacios nuevos con el valor del píxel más cercano, esta es una opción segura que evita artefactos visuales extraños.

* validation\_split=0.2

Fracción de imágenes reservadas para validación, separa el 20% de los datos para validación, esto permite evaluar el rendimiento del modelo en datos no vistos durante el entrenamiento.

#### 3.3.3.2 Aplicación de las transformaciones a los datos de entrenamiento y validación

Las imágenes serán cargadas desde un directorio y se prepararán para el entrenamiento y validación.

* train\_datagen.flow\_from\_directory

Carga imágenes desde el directorio especificado (ruta\_train) y aplica las transformaciones definidas en el generador train\_datagen.

* DATASET\_DIR

Es la ruta al directorio principal que contiene subcarpetas organizadas por clases. Cada subcarpeta debe corresponder a una etiqueta de clase.

* target\_size=IMG\_SIZE

Dimensiones a las que se redimensionarán todas las imágenes, asegura que todas las imágenes tengan el mismo tamaño. Las CNNs requieren que todas las imágenes de entrada tengan las mismas dimensiones.

* class\_mode='categorical'

Formato de las etiquetas de salida, convierte las etiquetas a formato one-hot encoding, esto es necesario para problemas de clasificación multiclase con categorical\_crossentropy como función de pérdida.

* subset='training'/'validation'

Especifica si se usan los datos para entrenamiento o validación, separa los datos según el validation\_split definido, esto permite mantener conjuntos de datos separados pero consistentes.

# Implementación de la Solución

En esta fase, detallaremos el diseño y entrenamiento del modelo basado en redes neuronales convolucionales (CNNs) y transfer learning utilizando el modelo entrenado EfficientNet.

## 4.1 Modelo 1- Red Neuronal Convolucional (CNN)

Compilaremos y configuraremos el modelo de CNN con la ayuda de TensorFlow/Keras de la siguiente manera:

### 4.1.1 Definición del modelo

**Primera capa convolucional: Conv2D(32, (3, 3), activation='relu')**

Filtros (32): Número de kernels de convolución diferentes

Es un buen punto de partida que permite detectar características básicas sin consumir demasiados recursos, luego se duplicará, esto con el fin de detectar características más complejas.

Tamaño del kernel (3, 3): Ventana de convolución

Este es el tamaño más pequeño que puede capturar direccionalidad, balancea bien la capacidad de detección de características con el costo computacional.

ReLU: Función de activación que introduce no-linealidad, esto reviene el problema del desvanecimiento del gradiente además de que computacionalmente es más eficiente.

MaxPooling2D((2, 2))

Reduce la dimensionalidad espacial a la mitad además de reducir el número de parámetros, esto hacer que el modelo sea más robusto a pequeñas traslaciones, así como también ayuda a prevenir el sobreajuste.

**Segunda capa: Conv2D(64, (3, 3))**

Se duplica el número de filtros (64), permitiendo detectar características más complejas, compensando la reducción de dimensionalidad espacial del pooling.

**Tercera capa: Conv2D(128, (3, 3))**

Se vuelve a duplicar filtros (128), esto aumenta la capacidad de aprender patrones más abstractos, se sigue el patrón común de incrementar filtros mientras se reduce dimensionalidad.

**Flatten()**

Realizamos el llamado aplanamiento, esto se hace convirtiendo los mapas de características 2D en un vector 1D, esto es necesario para conectar con capas densas.

**Dense(128, activation='relu')**

Primera capa completamente conectada

128 neuronas: suficientes para procesar las características extraídas, esto mantiene el modelo relativamente ligero.

ReLU como activación por las mismas razones anteriores

**Dense(num\_classes, activation='softmax')**

Capa de salida, convierte las salidas en probabilidades, suma 1 entre todas las clases, ideal para clasificación multiclase

### 4.1.2 Compilación del Modelo

**optimizer='adam'**

Adam es un algoritmo de optimización que combina las ventajas de dos otros optimizadores (RMSprop y Momentum)

**Características principales:**

Tasa de aprendizaje adaptativa para cada parámetro

Utiliza estimaciones de momentos de primer y segundo orden

Autoadapta la tasa de aprendizaje durante el entrenamiento

Este optimizador funciona bien en la mayoría de los problemas, requiere poca configuración manual, converge más rápido que SGD simple, maneja bien problemas con gradientes ruidosos.

**loss='categorical\_crossentropy'**

Función de pérdida que mide la diferencia entre las predicciones y las etiquetas reales.

**Características:**

Diseñada específicamente para clasificación multiclase

Trabaja con etiquetas en formato one-hot encoding

Penaliza más fuertemente las predicciones incorrectas con alta confianza

Esta función de pérdida es ideal para problemas de clasificación con más de dos clases, proporciona gradientes útiles para el entrenamiento además es compatible con la activación softmax en la capa de salida.

**metrics=['accuracy'].** Métricas que se monitorizarán durante el entrenamiento

El Accuracy (precisión), mide el porcentaje de predicciones correcta ,fácil de interpretar, se calcula tanto para entrenamiento como validación

Esta métrica proporciona una medida intuitiva del rendimiento, permite monitorear el progreso del entrenamiento, es útil para detectar sobreajuste (comparando train vs. validation)

### 

### 4.1.3 Entrenamiento del Modelo

**train\_data y validation\_data**

Datos de entrenamiento y validación respectivamente

Proporcionan los datos con los que el modelo aprenderá y se evaluará, la separación permite evaluar el rendimiento en datos no vistos, ayuda a detectar sobreajuste, permite monitorear la generalización del modelo

**epochs**

Número de veces que el modelo procesará todo el conjunto de datos

Determina cuánto tiempo entrenará el modelo.

Esto es importante porque, con muy pocas epochs se cae en subajuste, lo contrario, demasiadas

epochs sería un posible sobreajuste

Se debe monitorear la validación para encontrar el punto óptimo

**class\_weight**

Diccionario que asigna diferentes pesos a cada clase, este conjunto de datos es bastante desbalanceado.

'cmd': 1.0, # Peso base

'cbb': 4.0, # 4 veces más importante

'cbsd': 2.0, # 2 veces más importante

'healthy': 5.0, # 5 veces más importante

'cgm': 3.0 # 3 veces más importante

Se decidió utilizar estos pesos para compensar el desbalance en el dataset, esto da más importancia a las clases con menos muestras o más críticas.

## 4.2 Modelo 2: Red Neuronal Convolucional con Dropout

Ahora utilizaremos el mismo modelo descrito en el 4.1.1 definición del modelo con la única diferencia que añadiremos capas de Dropout con el objetivo de reducir el sobreajuste (overfitting) al entrenamiento.

La función de estas capas es desactivar aleatoriamente un porcentaje de neuronas de la capa interior, esto previene que el modelo dependa demasiado de neuronas específicas, forzándolo a aprender características más generalizadas.

**layers.Dropout(0.5)**

Técnica de regularización que "apaga" aleatoriamente un porcentaje de neuronas durante el entrenamiento

El valor 0.5 significa que, en cada paso de entrenamiento, el 50% de las neuronas se desactivan aleatoriamente, esto previene el sobreajuste, hace el modelo más robusto, funciona como un conjunto de redes implícito.

**Ubicación estratégica del Dropout:**

**Después de Conv2D(128)**

Reduce la dependencia de características específicas, evita que el modelo se especialice demasiado en patrones particulares, ayuda a mantener la generalización en las características de alto nivel

**Antes de la última capa Dense**

Previene la co-adaptación entre neuronas, reduce la dependencia de características específicas, especialmente importante antes de la clasificación final

## 4.3 Modelo 3: Transfer Learning

Utilizamos el modelo basado en **EfficientNetB0** utilizando *Transfer Learning,* lo configuramos de la siguiente forma:

### 4.3.1 Configuración del modelo EfficientNetB0

**EfficientNetB0()**

Es una arquitectura CNN preentrenada que destaca por su eficiencia y rendimiento

**Características principales:**

Optimizada mediante búsqueda de arquitectura neural (NAS)

Balance óptimo entre profundidad, anchura y resolución

Menor número de parámetros que otras arquitecturas similares

**Análisis de los parámetros:**

**weights='imagenet'**

Carga los pesos preentrenados en el dataset ImageNet

Con esto se provecha el conocimiento aprendido de millones de imágenes

Las características de bajo nivel son útiles para la mayoría de las tareas

Acelera significativamente el entrenamiento

**include\_top=False**

Elimina las capas completamente conectadas finales del modelo original

Permite adaptar el modelo a un número diferente de clases

Mantiene solo las capas convolucionales (extractor de características)

Permite agregar nuestras propias capas de clasificación

**input\_shape=(IMG\_SIZE, IMG\_SIZE, 3)**

Define el tamaño de entrada de las imágenes

**Consideraciones:**

Debe ser compatible con la arquitectura, (224,224)

Balance entre resolución y recursos computacionales

3 canales para imágenes RGB

**base\_modelb0.trainable = True**

Permite que los pesos del modelo base sean actualizados

**Implicaciones:**

Fine-tuning completo del modelo

Mayor capacidad de adaptación al nuevo dominio

Requiere más recursos y datos

### 4.3.2 Definición del modelo

**base\_modelb0**

Modelo EfficientNetB0 preentrenado que actúa como extractor de características

Extrae características jerárquicas de las imágenes

Al utilizar en la primera capa, se aprovecha las características aprendidas de ImageNet

Proporciona una base sólida para la transferencia de aprendizaje

**GlobalAveragePooling2D()**

Reduce los mapas de características a un vector promediando espacialmente

Permite reducir drásticamente el número de parámetros, proporciona invarianza a traslaciones

Es importante utilizarlo poque es más eficiente que Flatten(), reduce el riesgo de sobreajuste, mantiene la información espacial importante, no tiene parámetros entrenables.

**Dense(5, activation='softmax')**

Capa final de clasificación con 5 neuronas (una por clase)

Esta capa permite realizar la clasificación final, convierte las características en probabilidades para cada clase

Se utilizar softmax para garantiza que las salidas sumen 1, proporciona probabilidades interpretables, ideal para clasificación multiclase

### 4.3.3 Compilación del modelo

El método .compile() se utiliza para configurar el modelo antes de entrenarlo. Especifica:

El optimizador: Define cómo se ajustan los pesos del modelo durante el entrenamiento.

La función de pérdida: Indica qué métrica debe minimizar el modelo.

Las métricas: Permiten evaluar el rendimiento del modelo durante el entrenamiento y la validación.

En este caso, el modelo es de la familia EfficientNet, una arquitectura conocida por su eficiencia en tareas de clasificación de imágenes.

Se utiliza el optimizador Adam, que es una combinación de los algoritmos de optimización RMSProp y SGD con momentos. Es popular por su eficiencia y por ajustarse automáticamente a diferentes problemas de aprendizaje.

learning\_rate=0.001: Es la tasa de aprendizaje, que controla el tamaño de los pasos que da el optimizador al ajustar los pesos del modelo.

La pérdida y las métricas similares a los modelos anteriores.

### 4.3.4 Callbacks: optimizando el proceso de ajuste

* ReduceLROnPlateau

Este callback ajusta dinámicamente la tasa de aprendizaje (learning rate) cuando la mejora del modelo se estanca.

Parámetros:

* monitor='val\_loss':

Monitorea la pérdida en el conjunto de validación.

Reduce la tasa de aprendizaje si la pérdida no mejora.

* factor=0.2:

Reduce la tasa de aprendizaje multiplicándola por 0.2.

* patience=2:
* EarlyStopping

Detiene el entrenamiento del modelo si no mejora durante varias épocas consecutivas, para evitar gastar recursos innecesarios.

Parámetros:

* monitor='val\_loss':

Evalúa la pérdida en el conjunto de validación para determinar si el entrenamiento debe continuar.

* patience=3:

Permite 3 épocas consecutivas sin mejora antes de detener el entrenamiento.

### 4.3.5 Entrenamiento del Modelo

Realizamos el entrenamiento como en los anteriores modelos, con la única diferencia que se incluye los callbacks.

* callbacks=[lr\_reduction, early\_stopping]:
* lr\_reduction:

Ajusta dinámicamente la tasa de aprendizaje si la pérdida de validación se estanca.

* early\_stopping:

Detiene el entrenamiento si la pérdida de validación no mejora después de 5 épocas consecutivas.

# Resultados

En este capítulo vamos a presentar los resultados obtenidos a partir de los diferentes modelos que hemos generado y entrenado, esto para cumplir el objetivo secundario tres, en donde se debe evaluar la accuracy, precisión, recall, el F-1 Score.

## 5.1 Métricas de evaluación

### 5.1.1 Exactitud (Accuracy) por clase

### 

Proporción de predicciones correctas sobre el total de muestras. Esta es una métrica básica y su interpretación es sencilla.

Pero esta métrica puede ser engañosa cuando tenemos un conjunto de datos desbalanceado.

### 5.1.2 Sensibilidad o Recall por clase

Proporción de muestras correctamente clasificadas como positivas para una clase específica sobre todas las muestras de esa clase.

### 5.1.3 Precisión (Precision) por clase

Proporción de predicciones correctas para una clase específica entre todas las predicciones realizadas de esa clase.

### 5.1.4 F1 Score por clase

Promedio armónico entre precisión y sensibilidad, se lo utiliza cuando existe un desbalance entre estas dos métricas.

## 5.2 Métricas Modelo CNN

|  |
| --- |
|  |

Figura 9. Accuracy modelo CNN

|  |
| --- |
|  |

Figura 10. Pérdida modelo CNN

|  |
| --- |
|  |

Figura 11. Métricas modelo CNN

**Análisis por clase (figura 11):**

**cmd (Cassava Mosaic Disease):**

Precision: 0.12 (12%) - Baja precisión en identificar casos positivos

Recall: 0.04 (4%) - Muy baja tasa de detección de casos reales

F1-score: 0.06 (6%) - Rendimiento general muy pobre

Support: 93 muestras - Clase minoritaria

**cbb (Cassava Bacterial Blight):**

Precision: 0.24 (24%)

Recall: 0.36 (36%)

F1-score: 0.29 (29%)

Support: 288 muestras - Segunda clase más numerosa

**cbsd (Cassava Brown Streak Disease):**

Precision: 0.07 (7%) - Muy baja precisión

Recall: 0.01 (1%) - Casi nula detección de casos reales

F1-score: 0.02 (2%)

Support: 154 muestras

**healthy:**

Precision: 0.49 (49%) - Mejor precisión entre todas las clases

Recall: 0.52 (52%) - Mejor recall entre todas las clases

F1-score: 0.50 (50%) - Mejor rendimiento general

Support: 531 muestras - Clase mayoritaria

**cgm (Casava Green Mite):**

Precision: 0.02 (2%)

Recall: 0.02 (2%)

F1-score: 0.02 (2%)

Support: 63 muestras - Clase más minoritaria

**Métricas globales:**

Accuracy global: 0.35 (35%)

Macro avg: 0.19 (19%) - Promedio simple de todas las clases

Weighted avg: 0.32 (32%) - Promedio ponderado por el número de muestras

|  |
| --- |
|  |

Figura12. Gráfica matriz de confusión, modelo CNN

**Análisis por clase:**

**Clase 0 (cmd):**

Predicciones correctas: 4 (diagonal).

Principales errores: Clasificados como clase 3 (healthy) (46 casos) y clase 1 (cbb) (36 casos).

**Clase 1 (cbb) :**

Predicciones correctas: 105.

Principales errores: Clasificados como clase 3 (healthy) (145 casos).

**Clase 2 (cbsd):**

Predicciones correctas: 75.

Principales errores: Clasificados como clase 3 (healthy) (67 casos).

**Clase 3 (healthy):**

Predicciones correctas: 278.

Principales errores: Clasificados como clase 1 (cbb) (202 casos).

**Clase 4 (cgm):**

Predicciones correctas: 1.

Principales errores: Clasificados como clase 3 (healthy) (28 casos) y clase 1 (cbb) (33 casos).

## 5.3 Métricas Modelo CNN con Dropout

|  |
| --- |
|  |

Figura 13. Accuracy modelo CNN con Dropout

|  |
| --- |
|  |

Figura 14. Pérdida modelo CNN con Dropout

|  |
| --- |
|  |

Figura 15. Métricas modelo CNN con Dropout

**Análisis por clase (figura 15):**

**cmd (93 muestras):**

Mejoró ligeramente la precisión: 0.14 (vs 0.12 anterior)

Empeoró el recall: 0.01 (vs 0.04 anterior)

Empeoró F1-score: 0.02 (vs 0.06 anterior)

**cbb (288 muestras):**

Similar precisión: 0.25 (vs 0.24 anterior)

Ligera mejora en recall: 0.38 (vs 0.36 anterior)

Ligera mejora en F1-score: 0.30 (vs 0.29 anterior)

**cbsd (154 muestras):**

Empeoró dramáticamente: precisión, recall y F1-score cayeron a 0.00

Indica que el modelo no está detectando ningún caso de esta enfermedad

**healthy (531 muestras):**

Ligera caída en precisión: 0.46 (vs 0.49 anterior)

Mejora significativa en recall: 0.59 (vs 0.52 anterior)

Mejora en F1-score: 0.52 (vs 0.50 anterior)

**cgm (63 muestras):**

Empeoró a 0.00 en todas las métricas (vs 0.02 anterior)

El modelo no está detectando esta enfermedad en absoluto

**Métricas globales:**

Accuracy: Ligera mejora a 0.37 (vs 0.35 anterior)

Macro avg: Ligera caída a 0.17 (vs 0.19 anterior)

Weighted avg: Similar en 0.32

|  |
| --- |
|  |

Figura 16. Gráfica matriz de confusión, modelo CNN con Dropout

**Clase 0 (cmd):**

Predicciones correctas: 1.

Principales errores: Clasificados como clase 1 (cbb) (37) y clase 3 (healthy) (55).

El modelo tiene problemas para reconocer esta clase.

**Clase 1 (cbb):**

Predicciones correctas: 108.

Principales errores: Clasificados como clase 3 (healthy) (179).

Aunque tiene varias predicciones correctas, hay una alta confusión con la clase 3 (healthy).

**Clase 2 (cbsd):**

Predicciones correctas: 93.

Principales errores: Clasificados como clase 3 (healthy) (57).

Desempeño moderado, pero también muestra confusión significativa con la clase 3 (healthy).

**Clase 3 (healthy):**

Predicciones correctas: 314 (el mayor número de aciertos).

Principales errores: Clasificados como clase 1 (cbb) (216).

Es la clase con mejor desempeño, pero presenta confusiones considerables con la clase 1 (cbb).

**Clase 4 (cgm):**

Predicciones correctas: 42.

Principales errores: Clasificados como clase 3 (healthy) (21).

Aunque el rendimiento es bajo, muestra menos confusión comparada con otras clases.

## 5.3 Métricas Modelo EfficientNet-B0

|  |
| --- |
|  |

Figura 17. Accuracy modelo EfficientNet-B0

|  |
| --- |
|  |

Figura 18. Pérdida modelo EfficientNet-B0

|  |
| --- |
|  |

Figura 19. Métricas modelo EfficientNet-B0

**Análisis por clase (figura 19):**

**cmd (93 muestras):**

Precisión bajó a 0.07 (vs 0.14 anterior)

Recall mejoró a 0.06 (vs 0.01 anterior)

F1-score mejoró a 0.07 (vs 0.02 anterior)

Muestra un rendimiento más equilibrado, aunque bajo

**cbb (288 muestras):**

Precisión similar: 0.24 (vs 0.25 anterior)

Recall bajó a 0.24 (vs 0.38 anterior)

F1-score bajó a 0.24 (vs 0.30 anterior)

Muestra métricas consistentes entre sí

**cbsd (154 muestras):**

Mejoró significativamente: de 0.00 a 0.11 en precisión

Recall de 0.10 (vs 0.00 anterior)

F1-score mejoró a 0.11 (de 0.00)

Recuperó la capacidad de detección

**healthy (531 muestras):**

Precisión se mantiene: 0.46

Recall bajó a 0.47 (vs 0.59 anterior)

F1-score bajó a 0.46 (vs 0.52 anterior)

Mantiene el mejor rendimiento general

**cgm (63 muestras):**

Mejoró de 0.00 a 0.03 en todas las métricas

Aunque bajo, al menos detecta algunos casos

**Métricas globales:**

Accuracy bajó a 0.30 (vs 0.37 anterior)

Macro avg: 0.18 (similar al anterior)

Weighted avg: 0.30 (vs 0.32 anterior)

|  |
| --- |
|  |

Figura 20. Gráfica matriz de confusión, modelo EfficientNet-B0

**Análisis clase por clase**

**Clase 0 (cmd):**

Correcto: 6 predicciones correctas.

Errores: Principalmente clasificadas como clase 3 (healthy) (50 casos).

Bajo desempeño, con más ejemplos clasificados incorrectamente que correctamente.

**Clase 1 (cbb):**

Correcto: 69 predicciones correctas.

Errores: Principalmente clasificadas como clase 3 (healthy) (139 casos) y clase 2 (cbsd) (41 casos).

Muestra una fuerte confusión con la clase 3 (healthy).

**Clase 2 (cbsd):**

Correcto: 15 predicciones correctas.

Errores: Confusión significativa con clase 3 (healthy) (76 casos) y clase 1 (cmd) (42 casos).

Bajo rendimiento en esta clase.

**Clase 3 (healthy):**

Correcto: 250 predicciones correctas.

Errores: Principalmente confundidas con clase 1 (cbb) (141 casos) y clase 2 (cbsd) (69 casos).

Es la clase con el mejor desempeño en términos de predicciones correctas.

**Clase 4 (cgm):**

Correcto: 2 predicciones correctas.

Errores: Confundidas principalmente con clase 3 (healthy) (32 casos) y clase 1 (cbb) (19 casos).

Rendimiento muy bajo en esta clase.

# Conclusiones

De acuerdo con el análisis de este conjunto de datos, se ha llegado a las siguientes conclusiones.

* Al ser un conjunto de datos desbalanceado, se deberían tratar los datos con algunas técnicas de balanceo de datos, técnicas como:
* Oversampling de clases minoritarias.
* Undersampling de clases mayoritarias.
* Técnicas como SMOTE
* Considerar técnicas de data augmentation específicas por clases.
* Evaluar otros tipos de arquitecturas.
* Refinar los class\_weights.
* Considerar el entrenamiento por etapas.
* Implementar validación cruzadas.
* Usar técnicas de early stopping mucho más sofisticados.

**Resultados**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Modelo | epochs | accuracy | loss | val\_accuracy | val\_loss |
| Modelo CNN | 20 | 0.6447 | 0.9577 | 0.6484 | 0.9693 |
| Modelo CNN con Dropout | 20 | 0.6354 | 1.0351 | 0.6068 | 1.0093 |
| Modelo EfficientNet-B0 | 10 | 0.9595 | 0.1156 | 0.8618 | 0.5324 |

Es necesario precisar que, a pesar del modelo presentado, es necesario seguir mejorando con el objetivo de obtener resultados mucho más confiables para que estos sean utilizados productores.

# Líneas de trabajo futuras

En función de los problemas detectados con este trabajo de fin de máster, han surgido algunas líneas de trabajo futuras.

1. Con el uso de la arquitectura YOLO (You only look once), se podrían entrenar modelos con la consideración de usar clasificación y detección de objetos, se podría iniciar con 200 fotografías de cada clase, utilizar por ejemplo Roboflow o LabelImg para generar archivos tipo YOLO la cual tiene la característica de almacenar la clase, el centro del cuadro que enmarca el objeto y las dimensiones de este cuadro para etiquetar las clases específicas y entrenar dicho modelo.
2. Extender el modelo para detectar múltiples enfermedades en una misma hoja o fruto (detección multicategoría).
3. Explorar el uso de YOLO combinado con modelos de segmentación como Mask R-CNN para identificar áreas afectadas con mayor precisión.
4. Desarrollar una solución híbrida que combine detección y análisis temporal para monitorear la evolución de las enfermedades.

# Planificación o plan de investigación del proyecto

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Actividad** | **Meses** | | | |
| **Octubre** | **Noviembre** | **Diciembre** | **Enero** |
| Definición y planificación de trabajo final | x |  |  |  |
| Estado del are o análisis del mercado del proyecto |  | x |  |  |
| Diseño e Implementación del proyecto |  |  | x |  |
| Redacción de la documentación del TFM |  |  | x |  |
| Defensa del proyecto |  |  |  | x |

Figura 21. Planificación

# Glosario

Deep Learning (Aprendizaje Profundo):

Rama del aprendizaje automático que utiliza redes neuronales artificiales con múltiples capas para modelar y resolver problemas complejos.

Clasificación de Imágenes:

Tarea del aprendizaje profundo que consiste en asignar etiquetas o categorías a una imagen en función de su contenido.

Red Neuronal Convolucional (CNN):

Tipo de red neuronal diseñada para procesar datos estructurados en forma de grillas, como imágenes, utilizando operaciones de convolución para extraer características.

Convolución:

Operación matemática que combina un filtro con una región de la imagen para extraer características locales como bordes o texturas.

Pooling:

Operación que reduce la dimensionalidad de las características extraídas, manteniendo la información más relevante. Los tipos comunes son el max pooling y el average pooling.

Capa Densa (Fully Connected Layer):

Capa de una red neuronal en la que cada neurona está conectada a todas las neuronas de la capa anterior, utilizada para realizar la clasificación final.

Función de Activación:

Función que introduce no linealidad en la salida de una neurona. Ejemplos comunes incluyen ReLU, Sigmoid y Tanh.

ReLU (Rectified Linear Unit):

Función de activación que devuelve el valor de entrada si es positivo y cero si es negativo. Es ampliamente utilizada en redes profundas.

Backpropagation (Retropropagación):

Algoritmo utilizado para entrenar redes neuronales ajustando los pesos mediante la propagación del error desde la salida hacia las capas anteriores.

Learning Rate (Tasa de Aprendizaje):

Hiperparámetro que controla la magnitud de los ajustes en los pesos durante el entrenamiento.

Batch:

Conjunto de ejemplos de entrenamiento procesados juntos en una iteración del entrenamiento.

Overfitting (Sobreajuste):

Situación en la que un modelo se ajusta demasiado a los datos de entrenamiento, perdiendo capacidad de generalización en datos nuevos.

Regularización:

Técnica utilizada para prevenir el sobreajuste penalizando los pesos altos en la función de pérdida. Ejemplos comunes incluyen L1, L2 y Dropout.

Dropout:

Técnica de regularización que desactiva aleatoriamente algunas neuronas durante el entrenamiento para mejorar la generalización.

Softmax:

Función utilizada en la capa de salida de modelos de clasificación multiclase para convertir los valores en probabilidades.

Función de Pérdida:

Métrica que cuantifica la diferencia entre las predicciones del modelo y los valores reales. En clasificación, la pérdida de entropía cruzada es comúnmente usada.

Dataset (Conjunto de Datos):

Colección de ejemplos utilizados para entrenar y evaluar modelos de aprendizaje profundo. Suele dividirse en entrenamiento, validación y prueba.

Augmentación de Datos:

Técnica para aumentar la diversidad del conjunto de datos aplicando transformaciones como rotaciones, volteos y ajustes de brillo a las imágenes.

Transfer Learning (Aprendizaje por Transferencia):

Técnica que reutiliza un modelo previamente entrenado en una tarea similar para acelerar y mejorar el rendimiento en una nueva tarea.

GPU (Unidad de Procesamiento Gráfico):

Hardware especializado que acelera el cálculo de operaciones en aprendizaje profundo, especialmente las matrices y tensores.

Epoch:

Iteración completa sobre el conjunto de datos de entrenamiento durante el proceso de entrenamiento del modelo.

Precision (Precisión):

Métrica de evaluación que mide la proporción de predicciones correctas entre todas las predicciones positivas.

Recall (Exhaustividad):

Métrica de evaluación que mide la proporción de predicciones positivas correctas en relación con todas las instancias positivas reales.

Matriz de Confusión:

Tabla utilizada para evaluar el rendimiento de un modelo de clasificación al comparar predicciones con etiquetas reales.

Cross-Entropy Loss (Pérdida de Entropía Cruzada):

Función de pérdida comúnmente utilizada en clasificación para cuantificar la diferencia entre las probabilidades predichas y las reales.

Gradient Descent (Descenso por Gradiente):

Algoritmo de optimización utilizado para minimizar la función de pérdida ajustando iterativamente los pesos del modelo.

Optimizer (Optimizador):

Algoritmo que regula el proceso de ajuste de los pesos del modelo durante el entrenamiento. Ejemplos: SGD, Adam, RMSprop.

# Bibliografía

Adebayo, W. G. (2023). Cassava production in africa: A panel analysis of the drivers and trends. *Heliyon*, *9*(9). https://doi.org/10.1016/j.heliyon.2023.e19939

Alado, D. (2024). Cassava Disease Classification Using Squeezenet CNN Technique. *2024 IEEE 15th Control and System Graduate Research Colloquium (ICSGRC)*, 1–5. https://doi.org/10.1109/ICSGRC62081.2024.10691308

Alonso-Gómez, L. A., Celis-Carmona, D. D., Rodríguez-Sánchez, Y. F., Castro-Ladino, J. R., & Solarte-Toro, J. C. (2024). Biochar production from cassava waste biomass: A techno-economic development approach in the Colombian context. *Bioresource Technology Reports*, *26*, 101872. https://doi.org/10.1016/J.BITEB.2024.101872

Alsaleh, A., & Perkgoz, C. (2023). A space and time efficient convolutional neural network for age group estimation from facial images. *PeerJ Computer Science*, *9*, e1395. https://doi.org/10.7717/peerj-cs.1395

Burns, A. E., Bradbury, J. H., Cavagnaro, T. R., & Gleadow, R. M. (2012). Total cyanide content of cassava food products in Australia. *Journal of Food Composition and Analysis*, *25*(1), 79–82. https://doi.org/10.1016/J.JFCA.2011.06.005

Burns, A., Gleadow, R., Cliff, J., Zacarias, A., & Cavagnaro, T. (2010). *Cassava: The Drought, War and Famine Crop in a Changing World*. *2*, 3572–3607. https://doi.org/10.3390/su2113572

Cock, J. H., & Connor, D. J. (2021). Cassava. *Crop Physiology Case Histories for Major Crops*, 588–633. https://doi.org/10.1016/B978-0-12-819194-1.00019-0

Fathima, A. A., Sanitha, M., Tripathi, L., & Muiruri, S. (2023). Cassava (Manihot esculenta) dual use for food and bioenergy: A review. *Food and Energy Security*, *12*(1). https://doi.org/10.1002/FES3.380

Fitriyano, G., Ismiyati, I., Purnawan, I., Othman, R., & Ramadhan, R. F. (2023). Mini Review: Potential Utilization of Cassava Peel Waste as Raw Material for Bio Briquettes Production in Indonesia. *E3S Web of Conferences*, *432*. https://doi.org/10.1051/E3SCONF/202343200011

Ghazal, S., Munir, A., & Qureshi, W. S. (2024). Computer vision in smart agriculture and precision farming: Techniques and applications. *Artificial Intelligence in Agriculture*, *13*, 64–83. https://doi.org/10.1016/J.AIIA.2024.06.004

Goodfellow, I. (2016). *Deep learning*.

Igwe, D. O. (2023). *Diseases of Cassava*. 1–37. https://doi.org/10.1007/978-3-030-35512-8\_31-1

Karuna, N., Jindapang, P., Saengphenchan, R., Panpedthan, J., & Supasorn, S. (2023). Cassava Leaves as an Alternative Nitrogen Source for Ethanol Fermentation. *Bioenerg*, *16*, 835–842. https://doi.org/10.1007/s12155-022-10473-7

Karunathilake, E. M. B. M., Le, A. T., Heo, S., Chung, Y. S., & Mansoor, S. (2023). The Path to Smart Farming: Innovations and Opportunities in Precision Agriculture. In *Agriculture (Switzerland)* (Vol. 13, Issue 8). Multidisciplinary Digital Publishing Institute (MDPI). https://doi.org/10.3390/agriculture13081593

Kotwal, J. G., Kashyap, R., & Shafi, P. M. (2024). Artificial Driving based EfficientNet for Automatic Plant Leaf Disease Classification. *Multimedia Tools and Applications*, *83*(13), 38209–38240. https://doi.org/10.1007/S11042-023-16882-W/METRICS

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. In *Nature* (Vol. 521, Issue 7553, pp. 436–444). Nature Publishing Group. https://doi.org/10.1038/nature14539

Legg, J. P., & Fauquet, C. M. (2004). Cassava mosaic geminiviruses in Africa. *Plant Mol Biol*, *56*, 585–599.

Marzougui, F., Elleuch, M., & Kherallah, M. (2020). A Deep CNN Approach for Plant Disease Detection. *2020 21st International Arab Conference on Information Technology (ACIT)*, 1–6. https://doi.org/10.1109/ACIT50332.2020.9300072

Merino, I., Azpiazu, J., Remazeilles, A., & Sierra, B. (2021). 3D convolutional neural networks initialized from pretrained 2D convolutional neural networks for classification of industrial parts. *Sensors (Switzerland)*, *21*(4), 1–18. https://doi.org/10.3390/S21041078

Montagnac, J. A., Davis, C. R., & Tanumihardjo, S. A. (2009). Nutritional Value of Cassava for Use as a Staple Food and Recent Advances for Improvement. *Comprehensive Reviews in Food Science and Food Safety*, *8*(3), 181–194. https://doi.org/10.1111/J.1541-4337.2009.00077.X

Paul, S. G., Biswas, A. A., Saha, A., Zulfiker, M. S., Ritu, N. A., Zahan, I., Rahman, M., & Islam, M. A. (2023). A real-time application-based convolutional neural network approach for tomato leaf disease classification. *Array*, *19*, 100313. https://doi.org/10.1016/J.ARRAY.2023.100313

Punuri, S. B., Kuanar, S. K. , Kolhar, M. , MishraT. K., Alameen, A. , Mohapatra, H. , & Mishra, S. R. (2023). *Efficient Net-XGBoost: An Implementation for Facial Emotion Recognition Using Transfer Learning.* *11(3)*, 776. https://doi.org/https://doi.org/10.3390/math11030776

Rajora, R., Banerjee, D., Upadhyay, D., Dangi, S., & Sunil, G. (2024). Precision in Cassava Root Disease Diagnosis: Merging CNN and Random Forest Techniques. *2024 Asia Pacific Conference on Innovation in Technology (APCIT)*, 1–6. https://doi.org/10.1109/APCIT62007.2024.10673432

ReportLinker. (2022). *Global Cassava Trends in 2022*. Https://Www.Reportlinker.Com/Clp/Global/3978.

Restrepo, S., & Verdier, V. (1997). Geographical Differentiation of the Population of Xanthomonas axonopodis pv. manihotis in Colombia. *Applied and Environmental Microbiology*, *63(11)*, 4427–4434.

Sahai, M. (2024, January). *Convolutional Neural Networks – Definition, Architecture, Types, Applications, and more*. Https://Www.Analytixlabs.Co.in/Blog/Convolutional-Neural-Network/.

Seseri, R. (2023, June 16). *What are CNNs?* Https://Www.Linkedin.Com/Pulse/Ai-Atlas-16-Convolutional-Neural-Networks-Cnns-Rudina-Seseri/.

Shrestha, G., Deepsikha, Das, M., & Dey, N. (2020). Plant Disease Detection Using CNN. *2020 IEEE Applied Signal Processing Conference (ASPCON)*, 109–113. https://doi.org/10.1109/ASPCON49795.2020.9276722

Tan, M., & Le, Q. V. (2019). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*.

Thresh, J. M., & Cooter, R. J. (2005). Strategies for controlling cassava mosaic virus disease in Africa. *Plant Pathology*, *54*(5), 587–614. https://doi.org/10.1111/J.1365-3059.2005.01282.X

# 

# Anexos

**ANEXO 1**

Enlace github

* https://github.com/manolomunoz651/tesis\_UOC

**ANEXO 2**

|  |  |
| --- | --- |
| Real: | cbb |
| Modelo: | Imagen que contiene Aplicación |

|  |  |
| --- | --- |
| Real: | cbsd |
| Modelo: | Gráfico  Descripción generada automáticamente con confianza media |

|  |  |
| --- | --- |
| Real: | cmd |
| Modelo: | Imagen que contiene Gráfico |

|  |  |
| --- | --- |
| Real: | cgm |
| Modelo: |  |

|  |  |
| --- | --- |
| Real: | sana |
| Modelo: | Imagen que contiene Gráfico |

|  |  |
| --- | --- |
| Real: | cbb |
| Modelo: |  |

|  |  |
| --- | --- |
| Real: | cbsd |
| Modelo: | Imagen que contiene Interfaz de usuario gráfica  Descripción generada automáticamente |

|  |  |
| --- | --- |
| Real: | cgm |
| Modelo: | Interfaz de usuario gráfica |

|  |  |
| --- | --- |
| Real: | cmd |
| Modelo: | Gráfico |

|  |  |
| --- | --- |
| Real: | healthy |
| Modelo: |  |

|  |  |
| --- | --- |
| Real: | cbb |
| Modelo: | Interfaz de usuario gráfica  Descripción generada automáticamente con confianza media |

|  |  |
| --- | --- |
| Real: | cbsd |
| Modelo: |  |

|  |  |
| --- | --- |
| Real: | cgm |
| Modelo: | Imagen que contiene Mapa |

|  |  |
| --- | --- |
| Real: | cmd |
| Modelo: | Interfaz de usuario gráfica, Aplicación  Descripción generada automáticamente |

|  |  |
| --- | --- |
| Real: | healthy |
| Modelo: | Imagen que contiene Interfaz de usuario gráfica |