Final Report

PlantStatus

Antonio Rangel Hugo Manuel

López Rosales Ivan Tonathiu

17 July 2025

Coacalco de Berriozábal, México state, México

Abstract

PlantStatus is an intelligent system developed to identify plant species and diagnose diseases through leaf image analysis. Its goal is to automate agricultural diagnostics in an accessible, fast, and accurate manner, particularly in resource-limited contexts. The prototype was designed in four stages: exploratory dataset analysis, image preprocessing, training of a deep learning model, and real-time testing with a camera.

During the exploratory analysis, over 160,000 images were identified, with varied formats (.jpeg, .png) and class imbalances that were later corrected. Preprocessing involved converting and resizing images to 224x224 pixels, generating class weights, and applying data augmentation. For training, a pretrained ResNet18 convolutional neural network was used, optimized with techniques such as early stopping and stratified validation. The system was evaluated in real-time using a webcam and successfully classified different species and diseases under practical conditions.

The results demonstrate that PlantStatus is technically feasible and economically accessible. It represents an innovative solution within engineering applied to precision agriculture and opens possibilities for future developments aimed at improving crop management through artificial intelligence.

Background/Introduction

In agriculture, one of the greatest challenges faced by producers is the early and accurate detection of plant diseases. Losses caused by pests can significantly impact production, increasing costs and reducing crop yields. Traditionally, disease diagnosis has relied on the knowledge of farmers or agronomic experts, which involves high technical consulting costs and significant margins of error due to the subjectivity of the analysis.

With advancements in information technologies and artificial intelligence, computer vision-based systems have been developed to identify patterns in images for classification and diagnostic tasks. Various studies have shown that convolutional neural networks (CNNs) can be trained to recognize different plant species and diagnose diseases through the analysis of leaf photographs. PlantStatus is a solution that employs a deep learning model trained with thousands of images of healthy and diseased leaves. Using a compact neural network based on ResNet18 and data augmentation techniques, PlantStatus can classify plant species in real time and detect the presence of diseases, providing a fast and visual diagnosis through the use of a camera.

The need driving this project stems from the lack of accessible tools for AI-assisted agricultural diagnostics, particularly in rural areas or farming communities with limited resources. PlantStatus aims to enhance precision in crop management, reducing losses and facilitating data-driven agricultural decisions.

This work builds on prior research in image classification within digital agriculture, incorporating elements such as class balancing, image preprocessing, GPU (CUDA) acceleration for learning, and real-time processing. With this, PlantStatus positions itself as an innovative and sustainable solution with real-world applications for the agri-food sector.

Engineering Fundamentals

El The development of PlantStatus is grounded in key principles of computational engineering, artificial intelligence, and computer vision, integrating theoretical and practical knowledge to address a real-world agricultural challenge: automated plant diagnostics through image analysis.

From the perspective of software engineering and intelligent systems, structured methods were applied for the modular design of the prototype, including project segmentation into phases (analysis, preprocessing, training, and testing) and the use of modern development tools such as PyTorch and OpenCV. Computational resource efficiency was prioritized through GPU (CUDA) acceleration and parallel processing techniques for large-scale image analysis.

In terms of machine learning engineering, the base model used was a ResNet18 convolutional neural network, selected for its residual architecture that enhances information flow and mitigates the vanishing gradient problem in deep networks. This model was tailored to the dataset's characteristics using techniques such as transfer learning (pretraining), data augmentation, and class balancing with inverse weighting.

Regarding computer vision, fundamental principles like adaptive thresholding, color-based region segmentation, and contour detection were employed, enabling the system to identify leaves in real time via a camera. These processes rely on morphological operations and spatial transformations of digital images, widely used in biomedical engineering, robotics, and control systems.

The prototype's design also considers hardware engineering and compatibility aspects, adapting the system to run on a standard personal computer with realistic specifications and using a standard webcam. This demonstrates the project's feasibility even without specialized industrial equipment.

In terms of standards, the model's development follows industry best practices for software and AI research, including the use of clean datasets, cross-validation, data set separation, and tracking of key metrics. Ethical principles are also upheld by working exclusively with public data and maintaining transparency throughout the process.

Finally, the project's technical innovation lies in integrating these technologies into a functional, accessible, and reproducible system. PlantStatus not only applies classic

engineering principles but also aligns with current trends in precision agriculture, promoting sustainable, intelligent, and adaptable solutions for small-scale farmers.

Prototype Design

The PlantStatus prototype was designed as a tool capable of identifying plant species and diagnosing diseases through leaf image analysis. Its primary goal is to automate visual agricultural diagnostics in an accessible and real-time manner.

The prototype's development was structured into four key stages:

1. **Exploratory Analysis**: The dataset's structure was thoroughly examined, identifying image formats (.png, .jpg, .jpeg) and evaluating the quality and class balance by plant species and condition. Images with size variations were detected and discarded, ensuring a reliable dataset for subsequent processing.
2. **Preprocessing**: In this phase, all images were standardized to .jpg format and resized to 224x224 pixels. Data augmentation techniques (rotation, flipping, cropping, etc.) were applied to enhance model robustness. Additionally, a class balancing system using inverse weighting was implemented to prevent bias during training.
3. **Data Splitting and Model Training**: The model used was a pretrained ResNet18 convolutional neural network with ImageNet weights. It was trained using the CrossEntropy loss function, the AdamW optimizer, and techniques such as ReduceLROnPlateau and early stopping to prevent overfitting and improve overall performance.
4. **Real-Time Testing**: The trained model was integrated into a live classification system using a Logitech C920 camera. With the aid of computer vision techniques like adaptive thresholding and contour detection, leaves within the camera's field of view are identified. Detected regions are classified by the model, displaying the plant species and any disease presence on the screen, along with a confidence percentage.

Technical Specifications of the Prototype

- **Computer**: AMD Ryzen 5 5600G processor, 16 GB DDR4 RAM, SSD for faster performance, and a dedicated NVIDIA GeForce RTX 3050 graphics card with 6 GB VRAM.
- **Camera**: Logitech C920 (1080p resolution, autofocus).
- **Plants Used for Testing**: Apple, peach, and green grape leaves.

Tools and Technologies Used

- **Model**: Pretrained ResNet18.
- **Libraries**: PyTorch, torchvision, CuPy, tqdm, PIL, NumPy, OpenCV (cv2).
- **Acceleration**: GPU with CUDA support.
- **Techniques Implemented**: WeightedRandomSampler, DataLoader, data augmentation, balanced class analysis.

This modular and scalable design allows PlantStatus to be adapted to new crops, diseases, or field conditions without compromising functionality or accuracy.

Implemented Methodology

In the development of the PlantStatus system, a quantitative and experimental methodology was implemented, focused on creating, training, and validating an artificial intelligence model capable of identifying a plant's health status from leaf images. The research was structured in progressive phases, covering everything from exploratory data analysis to real-time model implementation.

First, a verification of the computational environment was conducted, ensuring the availability of resources such as a CUDA-compatible GPU using PyTorch libraries, which accelerated model training. Subsequently, the dataset was validated and structured, classifying images by plant species, image type (color, grayscale, and segmented), and health status. Libraries such as os, PIL, and NumPy were used to explore file extensions, resolutions, and potential corrupted images, ensuring data integrity.

In the preprocessing phase, tasks included format conversion, standardized resizing to 224x224 pixels, and class balancing through proportional weight assignment based on the number of images per category. Images were also subjected to data augmentation techniques (rotation, cropping, flipping, translation, etc.) to enhance the model's generalization ability.

The base model adopted was a ResNet18 convolutional neural network with pretrained ImageNet weights, fine-tuned for the project's specific classes. Training was conducted using the CrossEntropy loss function and the AdamW optimizer, complemented by techniques like ReduceLROnPlateau and early stopping to prevent overfitting. For efficient data handling, WeightedRandomSampler and DataLoader were used, enabling balanced image selection during training.

In the evaluation phase, a stratified split of the dataset was applied into training, validation, and test sets (70%, 15%, and 15%, respectively), ensuring class proportionality in each subset. The obtained metrics (accuracy, loss, etc.) were recorded and analyzed to assess the model's performance in real-world conditions.

Finally, a real-time classification system was implemented using a camera and libraries such as cv2 and torchvision. Captured images are processed live, detecting leaves through color segmentation, thresholding, and contour detection. The model classifies the species and disease, displaying the results along with prediction confidence. This solution is stable, efficient, and adaptable to various scenarios, laying the foundation for future practical applications in the agricultural sector.

Adaptive Resilience

Problem 1: Small Dataset
The initial dataset was limited, with few plant species and conditions.
Solution: Searched Kaggle and other platforms, eventually finding a comprehensive dataset with over 160,000 images, offering diverse plants and conditions.

Problem 2: Lack of GPU
No GPU was available for development.
Solution: Used Google Colab, an online platform, to develop and test the project without needing a powerful PC.

Problem 3: Large Dataset Upload Issues
The 160,000-image dataset was too large for Google Colab to upload directly.
Solution: Uploaded the dataset to Google Drive, extracted the ZIP in Colab, and accessed it from there, despite needing to reconnect Drive repeatedly.

Problem 4: Slow Processing in Google Colab
Colab was slow, and Google Drive storage was nearly full (6 GB of 10 GB free).
Solution: Purchased an NVIDIA GPU with 6 GB VRAM and switched to Cursor, an AI-enhanced IDE based on Visual Studio Code.

Problem 5: GPU Detection Issues
The system couldn't detect the GPU for processing.
Solution: Installed CUDA 12.9, resolved compatibility issues, and verified GPU detection with code.

Problem 6: Poor Dataset Organization
Plant conditions were mixed in folders, risking class detection issues.
Solution: Reorganized the dataset into folders by plant species and condition.

Problem 7: Preprocessing Bottleneck
Large batches and 6-8 workers caused a bottleneck during image resizing.
Solution: Reduced batch sizes and removed workers, slowing processing but eliminating the bottleneck.

Problem 8: Lengthy Model Training
Training with thousands of images would take days or a week.
Solution: Reduced the number of images and images per epoch to shorten training time.

Problem 9: Inaccurate Leaf Recognition
The model failed to identify plant leaves correctly, likely due to lighting or

environmental factors.
Solution: Adjusted recognition to use a bounding box, tweaked brightness, applied a green color filter, and tested in better-lit environments with fewer objects.

Engineering Innovation

PlantStatus represents an innovative proposal in the field of engineering applied to agriculture, integrating computer vision, deep learning, and real-time processing into a compact, accessible, and highly practical system. Its primary contribution lies in automating diagnostics through artificial intelligence, enabling the identification of both plant species and potential diseases through visual analysis of their leaves, all using a standard camera.

One of the project's most notable innovations is the development of a real-time classification system, which combines a pretrained ResNet18 neural network with a leaf detection pipeline based on segmentation, adaptive thresholding, and contour techniques. This approach ensures smooth performance on devices with limited resources, eliminating the need for complex or costly infrastructure.

At the methodological level, advanced image preprocessing techniques were implemented, including class balancing by species and condition, standardized format conversion, efficient resizing, and extensive data augmentation through geometric and color transformations. Additionally, a weighted data loading system was designed to significantly improve fairness in model training, enhancing accuracy without requiring an overload of new images in the dataset.

Another significant contribution is the stabilization of real-time predictions, a mechanism that minimizes result fluctuations during camera use, improving user experience and ensuring diagnostic reliability. This technique is implemented directly in the visualization interface, making the system not only functional but also intuitive and practical for field use.

PlantStatus also stands out for its scalability and adaptability. Thanks to its modular design and reliance on open-source libraries such as PyTorch, OpenCV, and PIL, the system can be expanded to include new crops or diseases without requiring a complete redesign of the architecture. This level of flexibility makes it a tool with potential for implementation across various agricultural sectors and regions with specific needs.

Collectively, these technical and methodological innovations position the project as a benchmark for how engineering can serve essential productive sectors. PlantStatus not only proposes improvements to current agricultural monitoring practices but also paves the way for future research in precision agriculture, technological sustainability, and rural automation.

Financial Feasibility

PlantStatus was designed with a focus on technical and economic efficiency, prioritizing the use of accessible, low-cost resources without compromising the quality of results. This section details the costs associated with the prototype's execution, as well as an evaluation of its profitability and long-term sustainability.

Direct Project Costs

Below is a breakdown of the expenses incurred for the development and validation of the prototype:

Item	Cost (MXN)
GeForce RTX 3050 Graphics Card	$4,384
Grape Plant	$50
Peach Plant	$250
2 Plant Pots	$100
Total	**$4,784**

Profitability and Sustainability

The investment made is significantly low compared to other commercial agricultural diagnostic solutions, which may require specialized sensors, industrial equipment, or subscriptions to private platforms. PlantStatus offers a scalable and reusable alternative, as once the model is trained and the system is set up, the cost of operating and replicating the prototype is minimal. Additionally, since the software can run on any GPU-compatible PC with a standard camera, the system is potentially transferable to agricultural communities with basic infrastructure, reinforcing its social and economic viability.

Financial Risks and Mitigation Strategies

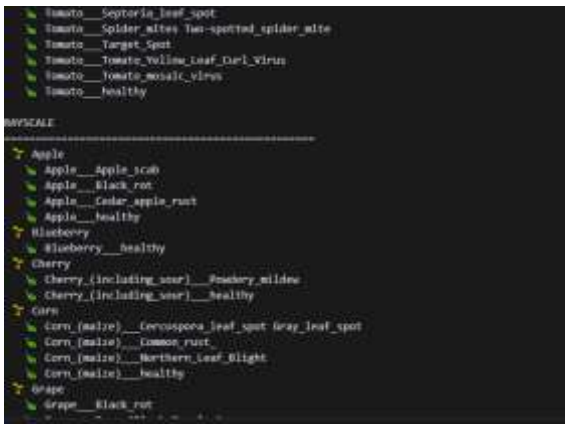
The main identified financial risks include:

- **Technological Obsolescence**: This can be mitigated by using open-source libraries and continuously updating the model.
- **Dependence on Dedicated Hardware (GPU)**: This can be reduced with future optimizations that allow execution on devices without graphics acceleration (CPU or mobile platforms).
- **Cost of Expanding the Dataset**: This can be mitigated through collaborations with agricultural institutions, open-access image banks, or synthetic data generation (data augmentation).

Outcome Presentation

1. Exploratory Data Analysis

1. A total of approximately 160,000 images were identified in the dataset, ensuring a robust foundation for model training.

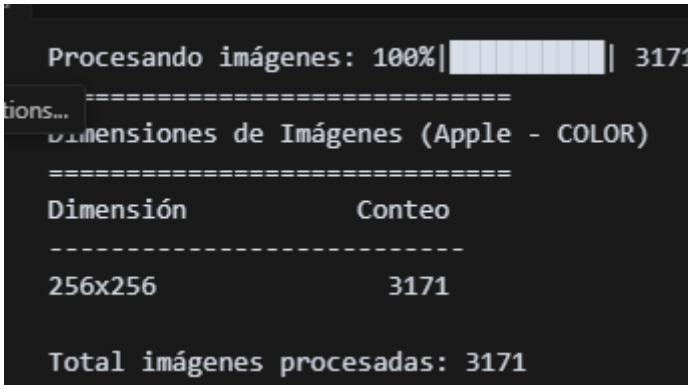


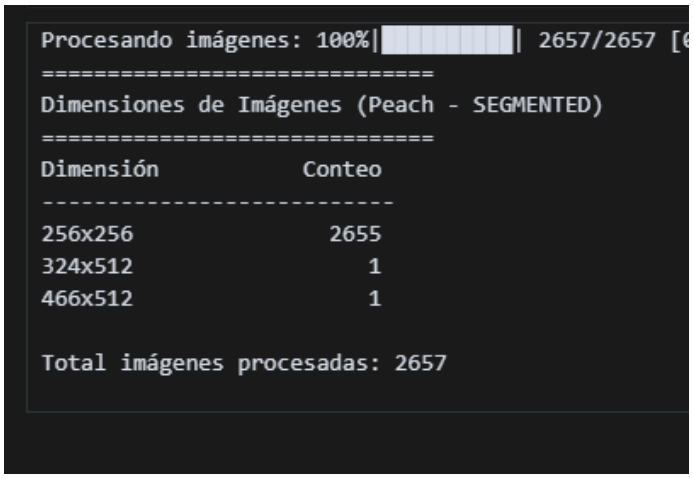
2. Two predominant image extensions were detected: .jpeg and .png, which were recorded and normalized in subsequent stages.

```
==============================
Extensiones de Imágenes
==============================
Extensión      Conteo
----------------------
.jpeg               2
.jpg           162912
.png                2

Total archivos: 162916
```

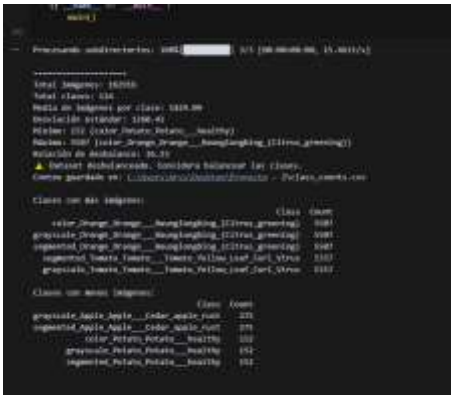
3. Most images had a standard resolution of 256x256 pixels, although other variable dimensions were also found depending on the plant and image type.





4. After a thorough analysis, no corrupted or empty images were found, indicating a clean and usable dataset without the need for additional filtering.

5. A class imbalance was observed, with some categories having significantly more images than others, which was addressed during the preprocessing stage.



2. Preprocessing

1. All images were converted to the .jpg format, standardizing the dataset and facilitating subsequent processing.

```
Procesando directorios: 160it [00:00, 218.58it/s]

====================
Imágenes .jpeg convertidas: 0
Imágenes .png convertidas: 0
Total archivos procesados: 0
No se encontraron imágenes .png o .jpeg.
```

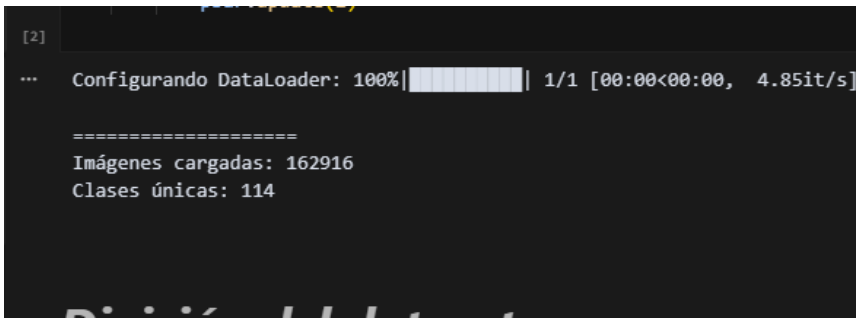
2. All images were resized to 224x224 pixels, as this is the expected input for the ResNet18 model.

```
Recolectando imágenes: 160it [00:01, 137.58it/s]
Redimensionando imágenes: 100%|██████████| 162916/162916 [04:07<00:00, 658.40it/s]

====================
Imágenes redimensionadas: 162916
Errores: 0
```

3. Weights were calculated for each class based on the number of images, assigning higher weights to underrepresented classes to improve balance during training.

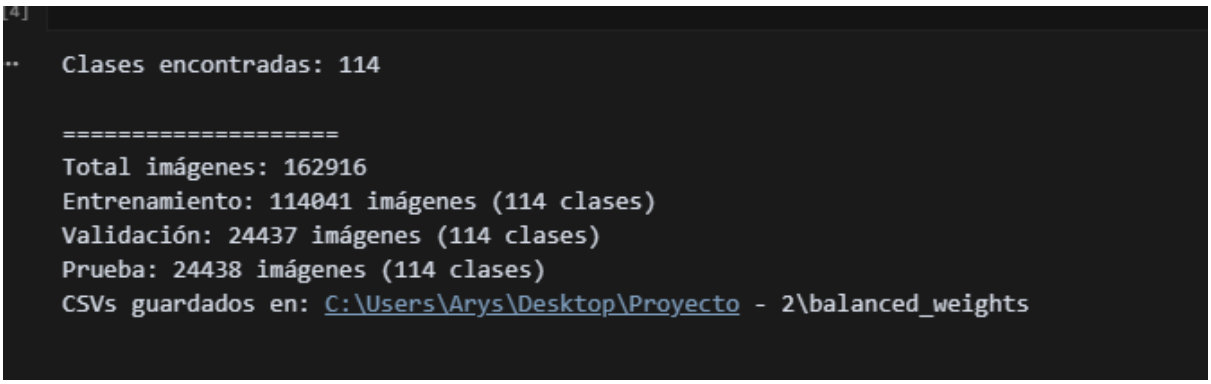
4. These weights were directly linked to the images via a CSV file, enabling their use with techniques such as WeightedRandomSampler in the DataLoader.



3. División y Entrenamiento del Modelo

1. The dataset of 160,000 images was stratified into three subsets:

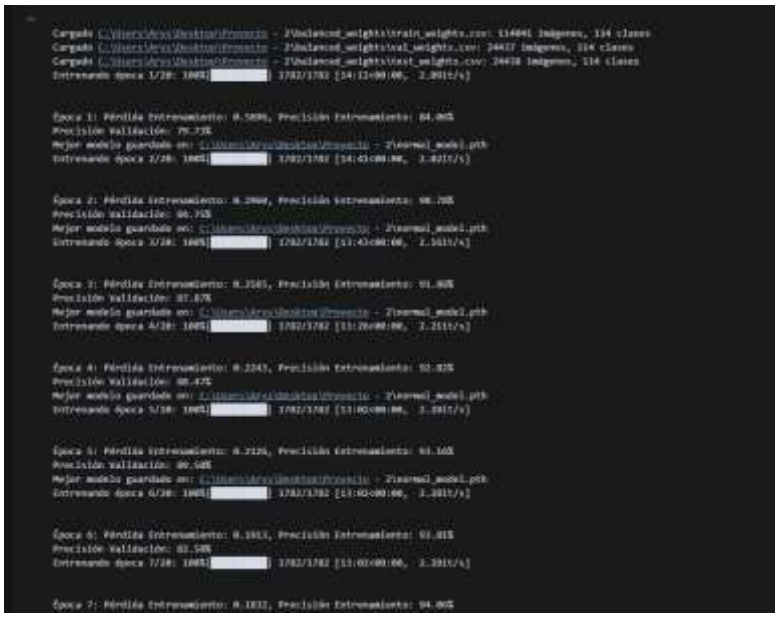
- 70% for training,
- 15% for validation,
- 15% for testing.

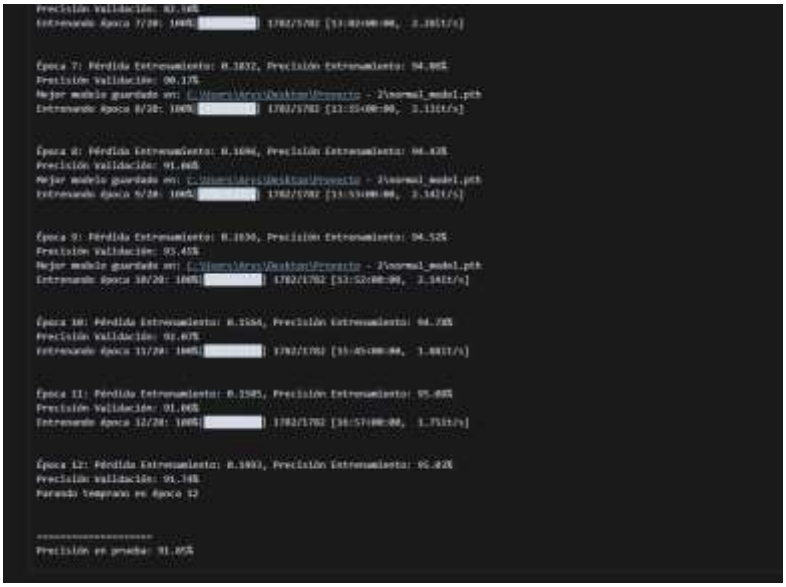


```
C:\Users\Arys\AppData\Local\Temp\ipykernel_19564\2653019098.py:26: FutureWarning: DataF
  df = df.groupby('Class', group_keys=False).apply(lambda x: x.sample(frac=sample_fract
Se ha muestreado el 30.0% del dataset para pruebas
Clases encontradas: 114

====================
Total imágenes: 293258
Entrenamiento: 205280 imágenes (114 clases)
Validación: 43988 imágenes (114 clases)
Prueba: 43990 imágenes (114 clases)
CSVs guardados en: C:\Users\Arys\Desktop\Proyecto - 2\augmented_weights
```

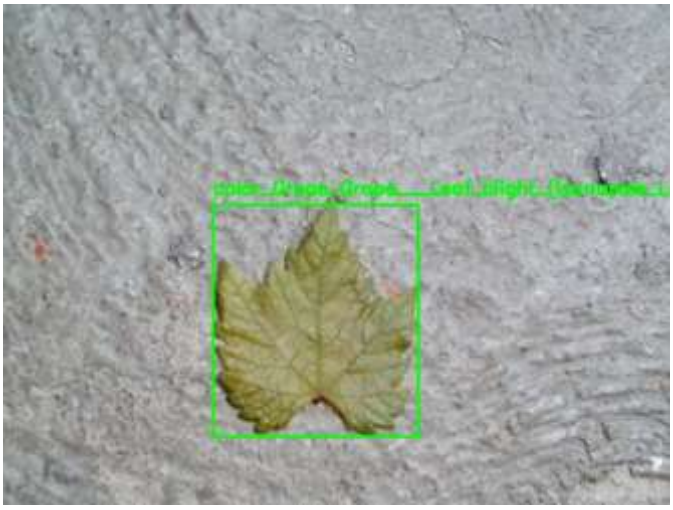
2. The model was trained over 20 epochs; however, early stopping was applied, halting training before all epochs were completed to prevent overfitting and preserve the best possible model.

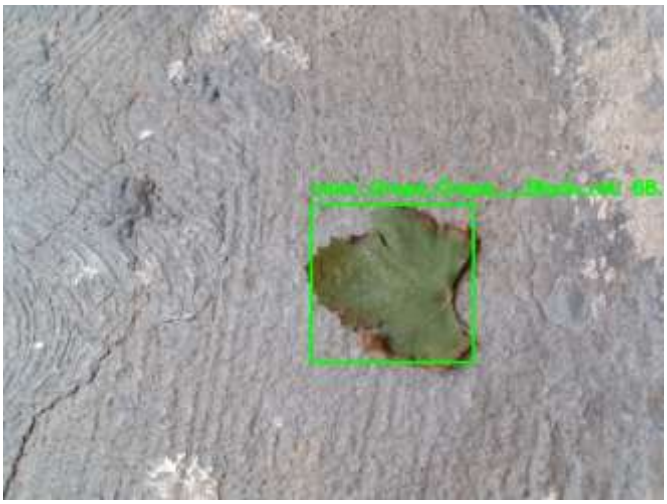



Precisión Validación: 82.58%
Entrenando época 7/20: 100%| | 1782/1782 [11:02<00:00, 2.69it/s]
Época 7: Pérdida Entrenamiento: 0.1832, Precisión Entrenamiento: 94.00%
Precisión Validación: 90.17%
Mejor modelo guardado en: C:\Users\Aero\Desktop\Proyecto - 2\normal_model.pth
Entrenando época 8/20: 100%| | 1782/1782 [11:55<00:00, 2.11it/s]
Época 8: Pérdida Entrenamiento: 0.1696, Precisión Entrenamiento: 94.42%
Precisión Validación: 91.06%
Mejor modelo guardado en: C:\Users\Aero\Desktop\Proyecto - 2\normal_model.pth
Entrenando época 9/20: 100%| | 1782/1782 [11:53<00:00, 2.34it/s]
Época 9: Pérdida Entrenamiento: 0.1634, Precisión Entrenamiento: 94.52%
Precisión Validación: 93.45%
Mejor modelo guardado en: C:\Users\Aero\Desktop\Proyecto - 2\normal_model.pth
Entrenando época 10/20: 100%| | 1782/1782 [11:52<00:00, 2.34it/s]
Época 10: Pérdida Entrenamiento: 0.1564, Precisión Entrenamiento: 94.78%
Precisión Validación: 92.07%
Entrenando época 11/20: 100%| | 1782/1782 [11:45<00:00, 1.88it/s]
Época 11: Pérdida Entrenamiento: 0.1505, Precisión Entrenamiento: 95.00%
Precisión Validación: 91.84%
Entrenando época 12/20: 100%| | 1782/1782 [10:57<00:00, 1.71it/s]
Época 12: Pérdida Entrenamiento: 0.1493, Precisión Entrenamiento: 95.02%
Precisión Validación: 91.34%
Parando temprano en época 12
==========
Precisión en prueba: 91.05%

4. Testing

1. Real-time tests were conducted using a camera to verify the system's performance under practical conditions. The model successfully detected leaves, classified them, and displayed the results live with a high confidence rate.





2. Two versions of the model were developed and evaluated:
    - A model trained exclusively with original data.
    - A second model trained with a combination of original and augmented data, improving generalization to images with visual variations.

Results Analysis

The results obtained from our models demonstrated high accuracy during the training, validation, and testing phases. Specifically, both models achieved over 90% accuracy on the test set, which is a highly competitive performance when compared to similar models. This strong performance highlights the model's ability to correctly identify both the type of plant and its health condition in most cases.

One of the main advantages of the system is its potential for real-time disease detection using a simple camera setup. However, there are also some limitations. The model requires controlled environments to perform optimally, as the presence of unrelated objects or varying lighting conditions can negatively impact its accuracy. Additionally, the model currently depends on a .pth file and must be executed from a device with access to a camera, which may limit its portability and ease of deployment.

As for future applications, this model could be integrated into automated drones or autonomous navigation systems in agricultural fields. Such systems could continuously monitor crop health, reduce maintenance costs, and improve efficiency when compared to manual inspection by humans.

References

Santillana Quesada, S. (2022). Introducción a las redes neuronales para el tratamiento de imágenes. (Trabajo de Fin de Grado). Universidad de Granada, Granada, España. Recuperado de

GeeksforGeeks. (2025, 21 abril). ResNet18 from Scratch Using PyTorch. GeeksforGeeks. https://www.geeksforgeeks.org/deep-learning/resnet18-from-scratch-using-pytorch/

Bossan, S. P. A. J. B. (s. f.). torch.compile and Diffusers: A Hands-On Guide to Peak Performance – PyTorch. https://pytorch.org/blog/torch-compile-and-diffusers-a-hands-on-guide-to-peak-performance/

OpenCV: Introduction. (s. f.). https://docs.opencv.org/4.x/d1/dfb/intro.html

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using Deep Learning for Image-Based Plant Disease Detection. Frontiers In Plant Science, 7. https://doi.org/10.3389/fpls.2016.01419

Hayes, A. (2025, 30 mayo). How stratified random sampling works, with examples. Investopedia. https://www.investopedia.com/terms/stratified_random_sampling.asp

PlantVillage Dataset. (2019, 1 septiembre). Kaggle. https://www.kaggle.com/datasets/abdallahalidev/plantvillage-dataset/data