Deep learning Project Report

Master's in Information Systems

Engineering and Artificial Intelligence

Under the Theme

Detection and Classification of Plant Diseases

Prepared by:

Mlle. BELHAJ Ikram.

Mlle . ABDELLAH Qatre.En.Nada

Mlle. TAGMOUTI Ghita

Mlle. CHAOUI Marwa

Supervised by:

Pr. Abdelatif Hafid

Defended on: **30/05/2025**

Academic Year: **2024 - 2025**

ACKNOWLEDGMENTS

We would like to express our deepest gratitude to Mr. Abdelatif Hafid, Professor of Deep Learning, for his guidance throughout this project entitled "*Detection and Classification of Plant Diseases Using Deep Neural Networks.*"

Mr. Hafid has transmitted to us, with both rigor and passion, the theoretical and practical foundations of Deep Learning, placing particular emphasis on the importance of project-based learning and a deep understanding of neural network architectures. Thanks to his clear explanations, his sense of pedagogy, and his availability, we were able to approach this project with confidence, curiosity, and ambition.

His expertise in the field of Artificial Intelligence, along with his ability to guide our thinking while granting us great autonomy, has been a valuable asset in the success of this work. He encouraged us to dig deeper, explore various technical paths, question our choices, and always aim for a better understanding of the challenges related to AI applications in agriculture.

This project has not only allowed us to apply advanced knowledge in Deep Learning (particularly CNNs and transfer learning), but also to become aware of the potential impact of these technologies in solving real-world problems, such as the automated detection of plant diseases — a major issue for modern agriculture.

We therefore extend our warmest thanks to Mr. Hafid for his kindness, patience, and the richness of his feedback throughout this work, which has been for us an enriching and highly motivating learning experience.

Chapter 1:

General Introduction

1.1. Context and Motivation

Modern agriculture, in constant evolution, faces numerous challenges, particularly those related to crop diseases. Among the most widespread and economically important crops is the apple tree, which plays a significant role in global fruit production. However, this crop is vulnerable to many foliar diseases, such as apple scab, rust, and black rot. These infections can lead to considerable losses in terms of yield, fruit quality, and even economic viability for farmers.

Traditionally, the detection of plant diseases relies on visual inspection by experts or experienced farmers. Although common, this method has several limitations: it is subjective, time-consuming, and often requires in-depth agronomic knowledge. Furthermore, in rural or developing areas, access to plant health specialists is limited, making prevention and treatment more complex.

In this context, Artificial Intelligence (AI), and more specifically Deep Learning, offers promising prospects. Thanks to recent advances in computer vision, it is now possible to develop systems capable of automatically detecting and classifying plant diseases from images with a high degree of accuracy. These systems not only save time but also assist farmers in decision-making, while reducing the overuse of pesticides.

This project aligns with that objective. It aims to design and train convolutional neural network (CNN) models to automatically detect foliar diseases of apple trees from a dataset of segmented images. The work is based on the use of an open-source dataset from the PlantVillage platform, widely used in AI-assisted agricultural research.

1.2. Problem Statement

In light of this context, the central problem we aim to address is the following:

"How can we design an intelligent, accurate, and accessible system to automatically detect and classify apple leaf diseases from images, using deep learning techniques, especially convolutional neural networks (CNNs)?"

This problem raises several technical and scientific challenges:

- **Data quality**: The images must be well-segmented and representative of different diseases.
- **Model generalization**: The model must not only recognize examples seen during training but also perform well on new, unseen images.

- **Interpretability**: The system should be able to explain its results to non-specialist users (farmers, technicians).
- **Efficiency**: The model should ensure fast predictions suitable for real-world use (mobile, drones, sensors, etc.).

1.3. Project Objectives

The general objective of this project is to develop an automatic classification solution for apple leaf diseases using segmented images. To achieve this, several specific goals have been defined:

1. Preprocess and explore image data from the PlantVillage dataset, particularly segmented apple leaf images.
2. Design a binary classification model to distinguish between healthy and diseased leaves.
3. Design a multiclass classification model to accurately identify the type of disease affecting the leaf (including: apple scab, rust, black rot).
4. Evaluate the model performances using standard metrics (accuracy, recall, precision, F1-score), ROC/AUC curves, and confusion matrices.
5. Visualize results using loss and accuracy graphs, validation curves, and interpret misclassifications.
6. Propose treatment recommendations or appropriate prevention measures for each detected disease, based on the model's outputs.

1.4. Methodology and Approach

The work carried out in this project follows a structured scientific approach, divided into several successive phases:

1. **Data collection**: Use of a subset of the segmented PlantVillage dataset, containing images of apple leaves belonging to multiple classes.
2. **Data exploration and preparation**: Cleaning, resizing, normalization, and data augmentation to enrich the dataset.
3. **CNN Modeling**:
    - **Model 1**: Binary classification (healthy vs. diseased)
    - **Model 2**: Multiclass classification (scab, rust, black rot, healthy)
4. **Model training and validation**: Tuning of hyperparameters (number of layers, filters, learning rate, epochs), and monitoring of loss and accuracy curves.

5. **Evaluation and visualization**: Generation of confusion matrices, testing on unseen data, and visualization of classification errors.
6. **Interpretation and recommendations**: Propose suitable agricultural solutions for each detected disease, based on existing agronomic knowledge.

Chapter 2:

Data Preparation and Exploration

2.1. Dataset Presentation

The dataset used in this project comes from the PlantVillage database, a reference in the field of automated plant disease detection. This database contains thousands of images of plant leaves (e.g., tomato, apple, grapevine, corn, etc.) in various conditions (healthy or affected by different diseases).

For our project, we extracted only the **segmented images of apple leaves**, which allows us to focus on the leaf surface without background elements that could interfere with classification. The selected classes are:

- Apple Scab
- Cedar Apple Rust
- Apple Black Rot
- Healthy (non-infected leaves)

Each image is labeled according to the health status of the leaf, which facilitates supervised classification tasks.

2.2. Data Loading and Organization

The images are stored in a directory structure organized by folders, with each folder corresponding to a class. We used the Python `os` library to retrieve paths, `PIL` for image loading, and `matplotlib` for visualization.

Example structure:

markdown
Copier le code

```
/data/apple_dataset_segmented/
  ├── Apple___Apple_scab/
  ├── Apple___Cedar_apple_rust/
  ├── Apple___Black_rot/
  └── Apple___healthy/
```

All images were resized to **128×128 pixels** to standardize the input size for the CNN model. The dataset was then split into three sets:

- **Training set**: 70% of images
- **Validation set**: 15%
- **Test set**: 15%

This split helps control overfitting and ensures proper evaluation of model performance.

2.3. Image Preprocessing

Before being used as input to a CNN model, the images were subjected to the following steps:

- **Resizing**: Standardizing all images to (128, 128)
- **RGB conversion**: Some images were in grayscale and were converted to RGB
- **Normalization**: Pixel values divided by 255 to scale into the [0,1] range
- **Data Augmentation** to simulate natural variations in image capture:
    - Random rotation (±20°)
    - Zooming (±10%)
    - Horizontal/vertical shifting
    - Horizontal flipping
    - Brightness adjustment

Data augmentation improves model robustness by increasing the variability in the training data.

2.4. Exploratory Data Analysis (EDA)

Initial exploration of the dataset helped us understand class distribution and image diversity.

Class distribution:

Class	Approx. Number of Images
Apple Scab	~630
Cedar Apple Rust	~620
Black Rot	~620
Healthy	~620

The dataset is **well balanced**, which is favorable for effective supervised learning without major class bias.

Sample images display:

Examples of leaves per class:

- **Apple Scab**: circular dark spots, deformation
- **Cedar Apple Rust**: orange circles with yellow halo
- **Black Rot**: black concentric lesions
- **Healthy**: uniformly green leaf

These visual differences are distinct enough for a CNN to learn to differentiate them.

2.5. Label Encoding

To make labels usable by the classification model, a numerical encoding was applied:

Label (Text)	Encoding
Apple Scab	0
Cedar Apple Rust	1
Black Rot	2
Healthy	3

A **One-Hot Encoding** is used for the multiclass classification, so the model outputs a probability vector across the four classes.

2.6. Summary of Tools Used

Here is an overview of the main libraries and tools used for data preparation:

Tool / Library	Main Usage
Pandas	Metadata handling
Matplotlib / Seaborn	Statistical and graphical visualization
TensorFlow / Keras	Dataset preparation, data augmentation
NumPy	Numerical operations
PIL	Image loading and processing
scikit-learn (sklearn)	Label encoding, dataset splitting

2.7. Conclusion

This data preparation and exploration phase allowed us to:

- Understand the dataset's structure and diversity
- Standardize image input for compatibility with CNNs
- Prepare properly split datasets for training, validation, and testing
- Apply data augmentation techniques to enhance learning diversity
- Identify key visual characteristics of each disease, aiding future model interpretation

Chapter 3:

Design of the Convolutional Neural Network Model

3.1 Introduction

In this chapter, we discuss the design of a Convolutional Neural Network (CNN) model for classifying plant leaf images into two categories: diseased and healthy. We detail the model architecture, hyperparameter choices, and the training and validation process used to evaluate its performance.

3.2 Building the CNN Model

The CNN model was built using the TensorFlow library and its Keras API, which allows for easy definition of deep architectures tailored to image processing tasks.

The chosen architecture is a **sequential network** composed of the following layers:

- **Convolutional Layers (Conv2D)**: These layers automatically extract key features from the images by applying filters that detect patterns such as edges, textures, or shapes.
  We used three convolutional blocks with an increasing number of filters (32, 64, 128), each followed by a **MaxPooling** layer that reduces the spatial dimensions of the feature maps, helping to decrease the number of parameters and extract robust information.
- **Flatten Layer**: This layer flattens the 2D outputs of the convolutions into a 1D vector to connect with the fully connected (dense) layers.
- **Dense Layers**: These are classic neural network layers where each neuron is connected to all neurons in the previous layer. We used a hidden dense layer with 128 neurons and ReLU activation, followed by a **Dropout layer** (rate: 0.5) for regularization by randomly deactivating neurons during training to prevent overfitting.
- **Output Layer**: The final layer is a dense layer with **softmax activation**, used to output a probability distribution over the two target classes.

Model Architecture Summary

Layer	Parameters	Description
Conv2D	32 filters (3x3), ReLU activation	Feature extraction
MaxPooling2D	Pool size (2x2)	Spatial dimension reduction
Conv2D	64 filters (3x3), ReLU activation	Feature extraction
MaxPooling2D	Pool size (2x2)	Spatial dimension reduction
Conv2D	128 filters (3x3), ReLU activation	Feature extraction
MaxPooling2D	Pool size (2x2)	Spatial dimension reduction
Flatten	–	Convert 2D output to 1D vector
Dense	128 neurons, ReLU activation	Fully connected hidden layer
Dropout	Rate 0.5	Regularization
Dense	2 neurons, softmax activation	Binary classification output

3.3 Model Compilation

The model was compiled using the **Adam optimizer**, an adaptive and efficient variant of stochastic gradient descent, commonly used in deep learning.

The **loss function** used was categorical_crossentropy, which is appropriate for multiclass classification problems with one-hot encoded labels (in this case, two classes).

The **performance metric** used was **accuracy**, which represents the percentage of correct predictions.

3.4 Model Training

Training the model involves adjusting the weights of the neurons to minimize the loss function on the training set.

The following parameters were selected:

- **Number of epochs**: 3 (this can be increased for better performance, though it requires more computation time)
- **Batch size**: 32 (number of samples processed before weights are updated)

The model was trained using the **fit()** method from Keras, which takes the training data (X_train, y_train) and validation data (X_test, y_test) as input. Validation allows us to monitor the model's performance on unseen data and detect potential overfitting.

Training progress was displayed for each epoch, including loss and accuracy values on both training and validation sets.

3.5 Validation and Performance Evaluation

After training, the model's performance was evaluated on the **test set** to measure its ability to generalize to new, unseen data.

The key metrics used were:

- **Accuracy**: Percentage of correctly classified images
- **Loss**: Value of the loss function on the test set

For deeper performance analysis, we can also explore:

- **Confusion matrix**
- **ROC curves**
- **Precision, recall, and F1-score**

3.6 Conclusion

Building and training the CNN model is a crucial step in the image classification process. This simple yet effective architecture lays the groundwork for further analysis, where hyperparameters can be fine-tuned and more complex architectures can be explored to improve the model's accuracy and robustness.

Chapter 4:

In-depth Evaluation of Model Performance

4.1 Evaluation Objective

The primary objective of this chapter is to rigorously analyze the performance of the two convolutional neural network (CNN) models developed in this project:

- A **binary classification model** designed to distinguish healthy leaves from diseased ones.
- A **multiclass classification model** aimed at accurately identifying the specific type of disease affecting a leaf among three common pathologies (apple scab, cedar apple rust, black rot), as well as the healthy class.

The evaluation is based on several complementary axes:

- **Standard metrics**: accuracy, precision, recall, F1-score.
- **Visualizations**: confusion matrices, learning curves, ROC curve.
- **Practical considerations**: real-world implications for farmers and field deployment.
- **Therapeutic recommendations** associated with the predictions of the multiclass model.

4.2 Evaluation of the Binary Model

4.2.1 Quantitative Results

The binary model achieves an overall **accuracy of 91%** on the test set. The table below summarizes the scores by class:

Classe	Précision	Rappel	F1-score	Support
Sain	0.95	0.86	0.90	199
Malade	0.88	0.95	0.92	216

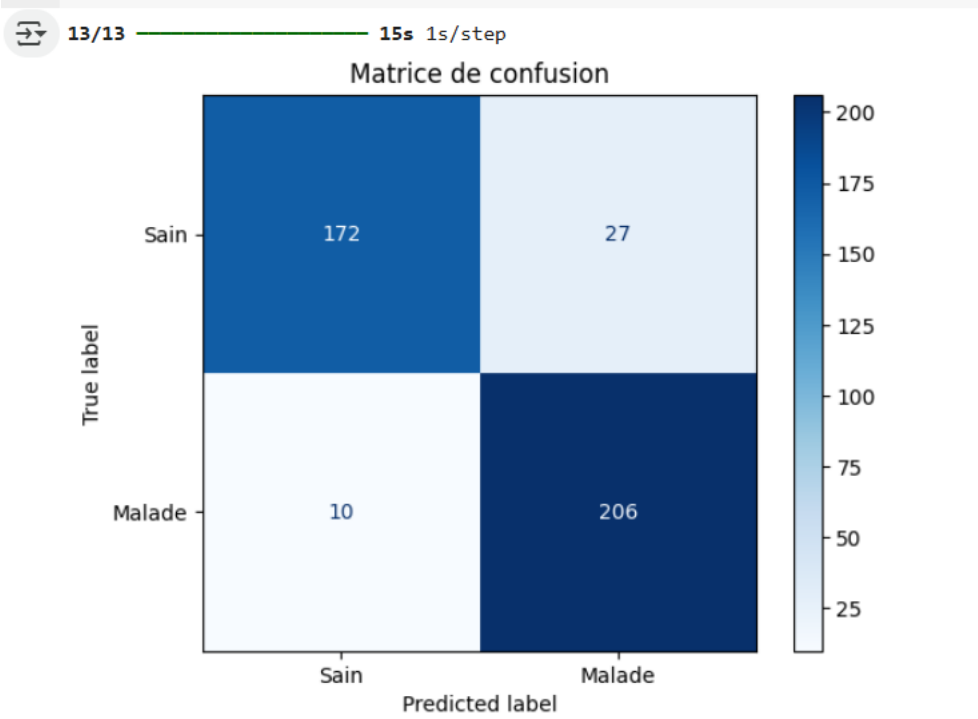
📌 **Interpretation**:

- The model performs excellently in detecting diseased leaves, with a **recall of 0.95**, which is crucial for preventing the spread of infections.
- There is a slight tendency to mistakenly classify healthy leaves as diseased (**false positives**), which remains acceptable in a preventive screening approach.

4.2.2 Confusion Matrix

	Prédit : Sain	**Prédit : Malade**
Réel : Sain	172 (VN)	27 (FP)
Réel : Malade	10 (FN)	206 (VP)

- **False negatives (FN)**: Only 10, which is excellent for an early detection system.
- **False positives (FP)**: Slight confusion, but acceptable from a practical perspective.



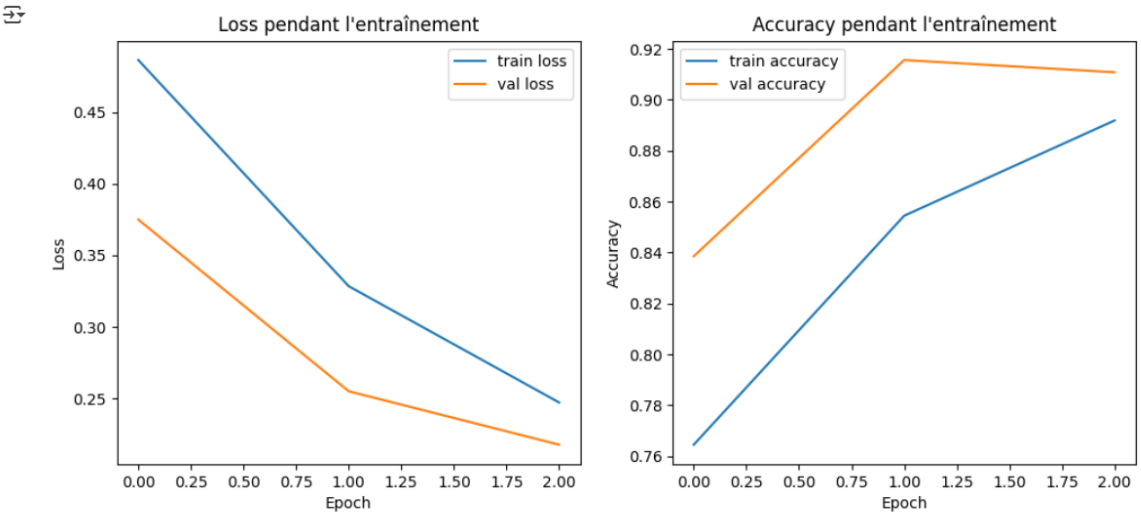
.

4.2.3 Learning Curves

The loss and accuracy curves show **rapid convergence by the 2nd epoch**, with stable performance across all epochs:

- No overfitting observed.
- Good alignment between training and validation sets.

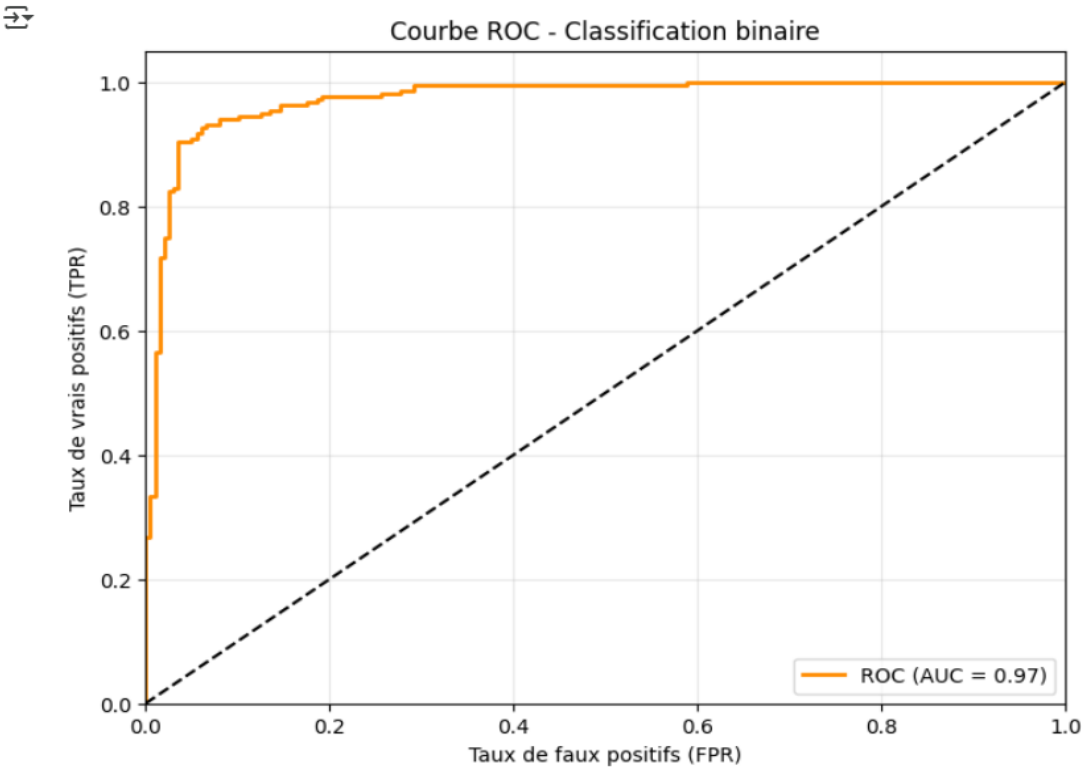
This indicates that the model generalizes well, even with a limited number of training epochs.



4.2.4 ROC Curve and AUC

The ROC curve for the binary model shows an **area under the curve (AUC) close to 1.0**, indicating excellent separability between healthy and diseased classes.

- This result allows for tuning the classification threshold based on the desired emphasis on **recall** or **precision**, depending on the application context.



4.3 Evaluation of the Multiclass Model

4.3.1 Training Results

Performance evolution during training is as follows:

Époque	Accuracy Entraînement	Accuracy Validation
1	62.38 %	79.04 %
2	81.22 %	83.86 %
3	83.94 %	85.06 %

- The model significantly improves accuracy across all classes.
- No signs of stagnation or overfitting.

4.3.2 Rapport de Classification

Classe	Précision	Rappel	F1-score	Support
Apple-Apple_scab	0.83	0.89	0.86	80
Apple-healthy	0.94	0.83	0.88	200
Apple-Cedar_apple_rust	0.69	0.73	0.71	55
Apple-Black_rot	0.81	0.95	0.87	80
Accuracy Globale			**0.85**	**415**

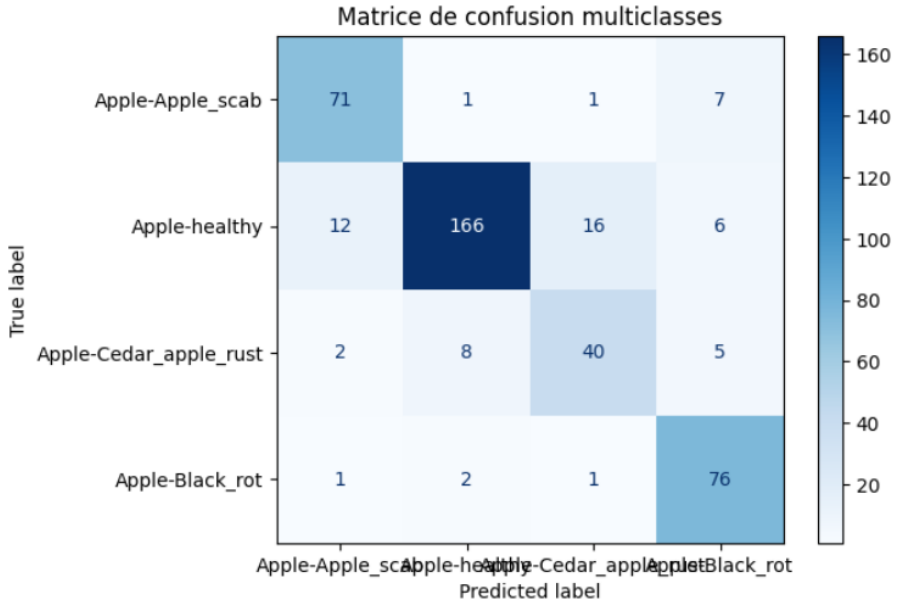
Key Observations:

- **Cedar apple rust** is the most challenging class, likely due to visual similarities with other diseases or with healthy leaves.
- **Black rot** and **apple scab** are well distinguished.
- The model remains effective for the healthy class, enhancing diagnostic reliability.

4.3.3 Confusion Matrix – Qualitative Analysis

- Most confusions involve **cedar apple rust**, often mistaken for healthy or scab leaves.
- Visually distinct classes (Black rot, Apple scab) are well captured by the model.
- The **healthy class** sometimes suffers from false positives (confused with scab or rust).

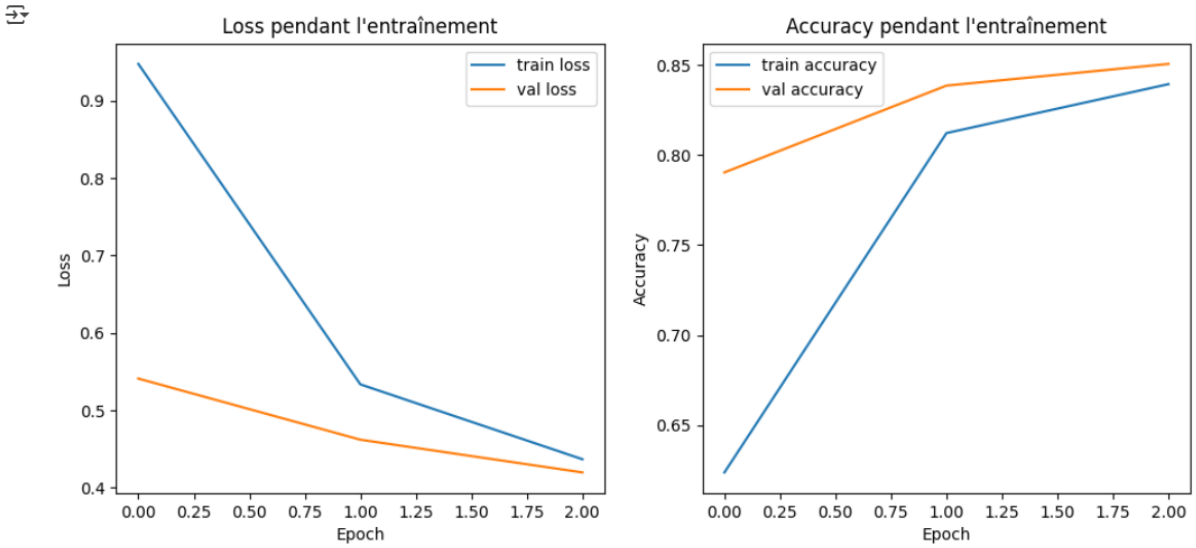
```
13/13 ━━━━━━━━━━━━━━━━━━━━ 13s 983ms/step
```



4.3.4 Learning Curves

The loss and accuracy curves show:

- A consistent improvement without early plateau.
- A small gap between training and validation, indicating **good generalization**.



4.3.5 Therapeutic Recommendations

One of the major advantages of the multiclass model is enabling **targeted action**. Below are the automatically generated recommendations based on the predicted class:

Classe détectée	Recommandation proposée
Apple-healthy	Aucun traitement – feuille saine
Apple-Black_rot	Taille des parties infectées + fongicide (thiophanate-méthyl)
Apple-Cedar_apple_rust	Éloignement des genévriers + fongicide (myclobutanil)
Apple-Apple_scab	Traitement biologique ou captane

4.4 Comparison of the Two Models

Critère	Modèle Binaire	Modèle Multiclasse
Objectif	Sain / Malade	Type de maladie (4 classes)
Accuracy globale	91 %	85 %
Métrique clé	Rappel sur "malade" = 0.95	F1-score moyen = 0.83
Utilité métier	Détection rapide	Diagnostic précis + traitement
Recommandation possible ?	Non	Oui
Complexité	Faible	Moyenne

✅ **Conclusion**:

- The **binary model** is ideal for fast pre-screening, particularly in early analysis phases.
- The **multiclass model** is better suited for detailed expert diagnosis with actionable recommendations.

4.5 Limitations and Areas for Improvement

4.5.1 Observed Limitations

- **Visual confusion** between certain classes (especially cedar apple rust vs healthy).
- **Low robustness** in real-world conditions (lighting variations, noise, blur).
- **Insufficient training duration** (only 3 epochs).

4.5.2 Améliorations Futures

Axe	Proposition technique
Données	Collecte d'images en conditions réelles (lumière naturelle, bruit)
Modèle	Tester des architectures avancées : ResNet, EfficientNet, ViT
Entraînement	Augmenter les époques, early stopping, scheduler
Explicabilité	Intégrer Grad-CAM ou LIME pour visualiser les zones influentes
Déploiement	Export vers TensorFlow Lite / ONNX pour intégration mobile

4.6 General Evaluation Conclusion

The in-depth evaluation shows that:

- The **binary model** is fast, accurate, and perfectly suited for automated pre-diagnostic stages. It reliably filters out healthy cases.
- The **multiclass model**, though more complex, provides detailed identification of pathologies and targeted therapeutic recommendations, aligning with the needs of a **decision support system for agriculture**.
- **Technical improvements** (fine-tuning, data enrichment, advanced architectures) will enhance robustness and adaptability in real-world scenarios.

Chapter 5:

Visual Analysis of Predictions and Agronomic Recommendations

5.1 Introduction

This chapter aims to complement the numerical evaluation of the multiclass model's performance (as presented in Chapter 4) with a visual and interpretative analysis of the predictions generated from the test dataset.

The goal is to verify the relevance of the classifications performed under conditions close to real-world usage and to demonstrate the added business value of such a system in an agricultural context. This analysis follows an applicative validation and field reliability perspective.

5.2 Prediction Methodology

The CNN model trained to classify into 4 categories (Apple-Apple_scab, Apple-Black_rot, Apple-Cedar_apple_rust, Apple-healthy) was tested on a subset of the test dataset.

The steps followed are as follows:

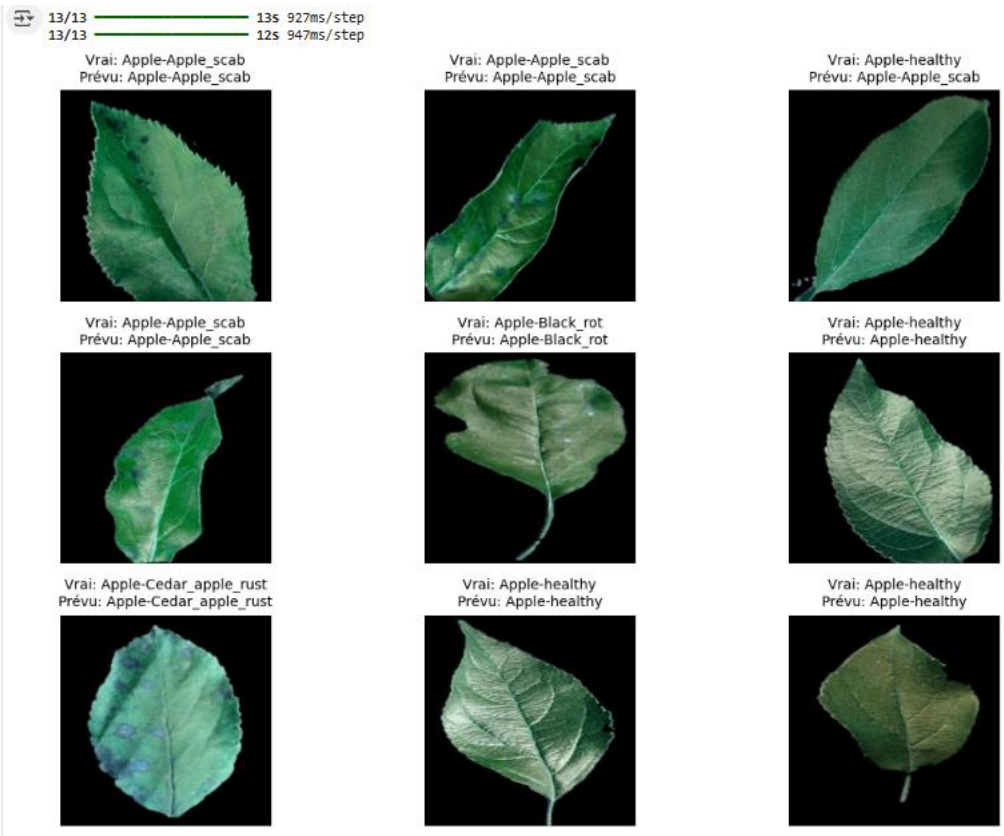
1. Random selection of 9 leaf images not seen during training.
2. Prediction of the class using the model (model.predict).
3. Display of the image with the true label and the predicted label.
4. Automatic generation of a treatment recommendation based on the detected disease.

The entire process was coded in Python using TensorFlow and Matplotlib, enabling simple but effective visualization.

5.3 Visual Analysis of Predictions

The following two figures illustrate the predictions made by the multiclass model on real leaf images from the test dataset.

Figure 5.1 – Visual Predictions (Series 1)

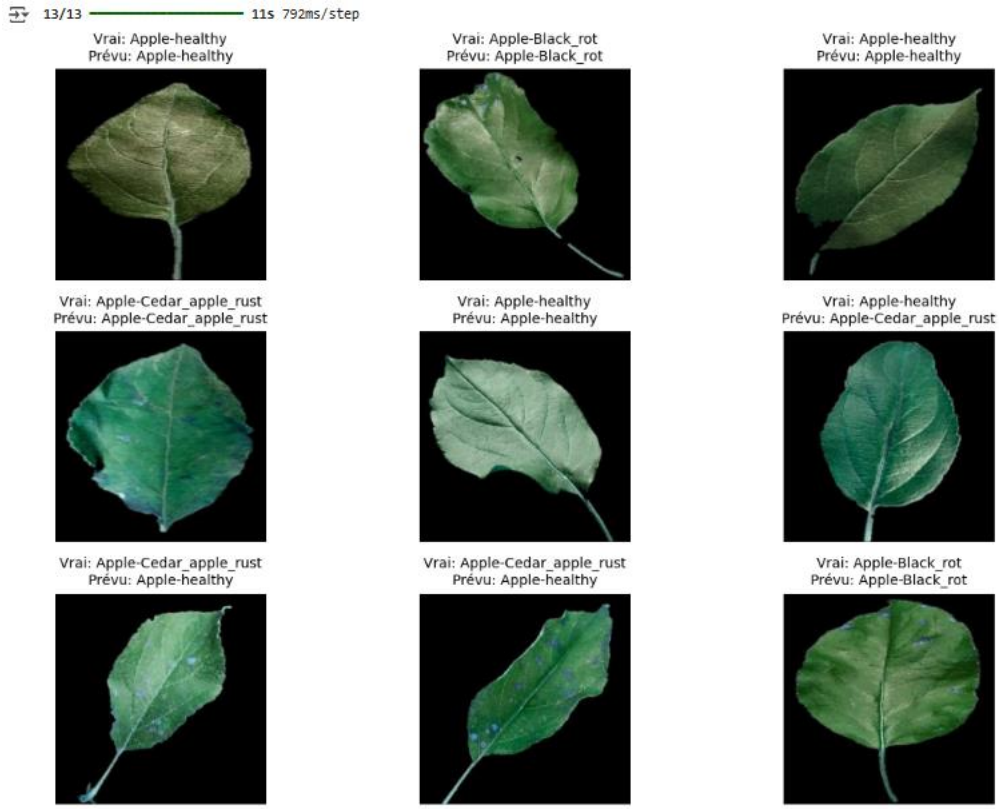


Caption: Random sample of predictions (9 leaves). The title of each image shows the true class ("True") and the predicted class ("Predicted"). Most predictions are correct, though some misclassifications are observed (e.g., healthy leaf predicted as rust-infected).

Observations:

- Healthy leaves are generally well identified (good color rendering, absence of lesions).
- Some false positives for Cedar apple rust: likely caused by natural variations in color or texture simulating anomalies.
- Diseased leaves with clear symptoms are correctly classified.

Figure 5.2 – Visual Predictions (Series 2)



Caption: Second series of predictions on the test set. Some confusions are visible between Apple_scab and healthy. The distinction between Apple_scab and rust remains sometimes delicate, probably due to similar leaf textures.

5.4 Analysis of Classification Errors

5.4.1 False Positives (Disease wrongly predicted)

- Often observed on healthy leaves showing reflections, pronounced veins, or natural spots.
- Risk: unnecessary treatment → product waste.

5.4.2 False Negatives (Disease not detected)

- Rare but critical: infected leaf classified as healthy.
- Possible causes:
    - Visually subtle symptoms,
    - Blurry or poorly lit image,
    - Symptoms underrepresented during training.
    -

5.4.3 Inter-Class Confusion

- Cedar apple rust is often confused with Apple_scab or Healthy.
- Likely explanation: rust areas not always clearly visible (depending on angle or lighting).
  □ Possible improvement: apply an explainability technique like Grad-CAM to understand which image regions influence the prediction.

5.5 Automatic Generation of Recommendations

Beyond classification, the model is enhanced by an applied interpretation module, providing specific phytosanitary recommendations according to the predicted class. Here are the automatically generated recommendations for the previous predictions:

Image	Predicted Class	Recommended Treatment
1	Apple-healthy	✅ No treatment necessary. Continuous monitoring.
2	Apple-Black_rot	💊 Prune infected branches. Apply thiophanate-methyl fungicide.
3	Apple-healthy	✅ No treatment necessary. Regular monitoring.
4	Apple-Cedar_apple_rust	⚠□ Remove surrounding junipers. Treat with myclobutanil fungicide.
5	Apple-healthy	✅ No treatment necessary.
6	Apple-Cedar_apple_rust	⚠□ Remove infected leaves. Treat with systemic fungicide (myclobutanil).
7	Apple-healthy	✅ No treatment.
8	Apple-healthy	✅ No treatment.
9	Apple-Black_rot	💊 Cut infected areas. Apply an effective fungicide.

5.6 Agronomic Interest and Field Applicability

This system offers concrete advantages for farmers:

- Smart diagnostic assistance, fast and accessible.
- Usable on mobile devices (with export to TensorFlow Lite or ONNX).
- Reduction of unnecessary treatments, saving resources.
- Better management of foliar diseases, environmentally friendly.

Possible integration:

- Agricultural diagnostic mobile app,
- Surveillance drone with onboard recognition,
- Web application for phytosanitary monitoring with suggestions.

5.7 Improvement Prospects

Area	**Proposed Improvement**
Data	Add images taken in real field conditions (outdoor, varied lighting).
Robustness	Integrate image quality filters (sharpness, exposure) to reduce errors.
Explainability	Visualize CNN activations with Grad-CAM to strengthen user trust.
Interface	Develop a user interface with audio feedback and geolocation.
Integrated Advice	Link each prediction to a simplified information sheet for non-expert users.

5.8 Conclusion

This chapter shows that visual prediction combined with agronomic interpretation offers a powerful framework for an intelligent decision support system. The visual results confirm the statistically observed performances.
The integration of concrete recommendations goes beyond simple diagnosis towards autonomous and contextualized decision-making.
Although still improvable, this model shows strong potential for real-world use in precision agriculture, provided it is tested under field conditions and integrated into an ergonomic software solution.

Conclusion

This report has presented the complete development cycle of a convolutional neural network model designed for the classification of plant leaf diseases, from data preparation to performance evaluation and visual interpretation. Through both quantitative metrics and qualitative analyses, the model demonstrated solid predictive capabilities, particularly in distinguishing between healthy and diseased leaves.

Beyond the technical accuracy, the integration of automated agronomic recommendations highlights the added value of artificial intelligence in supporting real-world agricultural decision-making. The proposed solution not only improves disease diagnosis efficiency but also promotes more sustainable and targeted phytosanitary interventions.

While promising, the system remains open to improvement. Future work will focus on increasing robustness in real-world conditions, enhancing model explainability, and developing user-friendly interfaces for deployment in mobile or embedded environments.

Overall, this project lays the foundation for an intelligent decision support tool that bridges the gap between deep learning and precision agriculture, with the potential to positively impact crop health monitoring and resource optimization on the field.