
RECONNAISSANCE DES PLANTES

Membres de l'équipe projet : Bernadette GASMI, Jean-Baptiste MACK, Morgan PERCHEC, Lionel SCHNEIDER

5 décembre 2025



Jean-Baptiste MACK



Morgan PERCHEC



Bernadette GASMI



Lionel SCHNEIDER



Résumé : Dans le cadre de notre formation DataScientist, nous mettons en pratiques les techniques d'intelligence artificielle enseignées avec un passage obligé par le Machine Learning pour progressivement conclure vers le Deep Learning.

Mots clés : Machine Learning, Deep Learning, Classification, Métriques

Table des matières

1.	Introduction	6
1.1.	Contexte et enjeux du projet	6
1.2.	Scénario	6
1.2.1.	Objectifs.....	7
1.2.2.	Revue littéraire	7
1.3.	Organisation du document	8
1.4.	Organisation du projet.....	8
1.5.	Environnement de développement	9
2.	Analyse exploratoire des datasets.....	10
2.1.	Sélection du dataset	10
2.2.	Exploration du dataset PlantVillage/Segmented	11
2.3.	Pré-processing	14
3.	Modélisation	14
3.1.	Machine Learning	14
3.1.1.	Méthodologie	14
3.1.2.	Extraire les caractéristiques	15
3.1.3.	Explorer les caractéristiques extraites	16
3.1.4.	Split train/valid/test	18
3.1.5.	Rééchantillonnage	18
3.1.6.	Pré-traitements	19
3.1.6.1.	Data augmentation	19
3.1.6.2.	Standardisation / normalisation des features	20
3.1.7.	Sélection des meilleures caractéristiques	20
3.1.8.	Entrainement des modèles et évaluations	21

3.2. Deep Learning	24
3.2.1. Méthodologie	24
3.2.2. Définition des critères de sélection	24
3.2.3. Exploration d'architectures DL	25
3.2.4. Évaluation comparative	32
3.2.5. Interprétabilité	35
3.2.5.1. Vérifier la pertinence des prédictions du modèle en phase d'inférence	36
3.2.5.2. Comparer l'attention du réseau entre les tâches de classification	37
3.2.5.3. Analyser l'influence d'une couleur de fond uni différente	37
3.2.5.4. Analyser l'inférence sur de nouvelles photos « in wild »	39
3.2.6. Sélection et recommandations	40
3.2.7. Limites et perspectives	41
4. Conclusion	43
5. Retour d'expérience	44
6. Bibliographie	44
7. Annexes	45
7.1. Description du code	45
7.2. Analyse exploratoire des caractéristiques extraites	45
7.3. Les méthodes pour analyser l'importance des features	54
7.4. Comparaison des caractéristiques des modèles ML	55
7.5. Matrice de confusion des modèles ML	55
7.6. Métriques trackées des architectures DL	56
7.7. Caractéristiques des features	63

Table des illustrations

Figure 1 : Scénario global	6
Figure 2: organisation du document.....	8
Figure 3 : Sélection du dataset pour premier cycle.....	10
Figure 4 : Comparatif des datasets	10
Figure 5 : extrait des images du dataset Plantvillage/segmented	11
Figure 6 : Schéma de principe des activités à mener pour ML (sans DL).....	15
Figure 7 : Caractéristiques extraites des feuilles de plantes	15
Figure 8 : Nouvelle répartition avec la data augmentation	19
Figure 9 : Importance globale des variables (SHAP) — Top 25.....	20
Figure 10: Importance des features par classe (SHAP) — Top-5 classes	21
Figure 11 : Performances des modèles ML sur l'ensemble de test	22
Figure 12 : Performances des modèles ML par classe.....	23
Figure 13 : Critères de sélection des architectures.....	25
Figure 14 : Comparatif des caractéristiques des modèles pré-entraînés choisis	26
Figure 15 : Synthèse des performances des 9 architectures	32
Figure 16 : comparaison et corrélation Accuracy et Macro F1-Score par architecture - cas 1	33
Figure 17 : classement final des 6 architectures pour l'identification de l'espèce	33
Figure 18 : comparaison et corrélation Accuracy et Macro F1-Score par architecture - cas 2	34
Figure 19 : classement final des 6 architectures pour l'identification de la maladie	34
Figure 20 : comparaison et corrélation Accuracy et Macro F1-Score par architecture - cas 3	35
Figure 21 : classement final des 6 architectures pour l'identification complète	35
Figure 22 : Synthèse des critères évalués pour les architectures retenues	41
Figure 23 : synthèse des performances obtenues par cas	41

Acronymes et sigles

Ce tableau présente les acronymes et sigles employés dans ce rapport.

Acronyme	Signification
----------	---------------

ML	Machine Learning
DL	Deep Learning
IA	Intelligence Artificielle

1. Introduction

1.1. Contexte et enjeux du projet

Les plantes occupent une place essentielle dans notre vie quotidienne, que ce soit pour alimenter les populations, protéger la biodiversité ou embellir nos paysages. Pourtant, leur santé est régulièrement menacée par des maladies, des parasites ou des conditions climatiques défavorables, entraînant des pertes économiques colossales — estimées à plusieurs centaines de milliards de dollars par an selon la FAO — et compromettant la sécurité alimentaire mondiale. En parallèle, l'engouement du grand public pour la reconnaissance des espèces et la compréhension de leur état de santé ne cesse de croître, comme le montre l'adoption massive d'outils collaboratifs tels que PlantNet.

Ce projet s'inscrit dans cette dynamique, développé dans le cadre d'une formation en data science. Il consiste à concevoir une solution basée sur des techniques d'intelligence artificielle pour identifier automatiquement les espèces végétales, évaluer leur santé et diagnostiquer d'éventuelles maladies. À travers ce travail, nous mettons en pratique des méthodes avancées d'apprentissage automatique tout en nous confrontant à des enjeux réels.

Nous sommes trois collaborateurs sans expertise dans ce domaine et un architecte IA qui possède des compétences dans l'industrialisation du Deep Learning.

1.2. Scénario

La figure suivante illustre le scénario global établi à partir de la mission du projet.

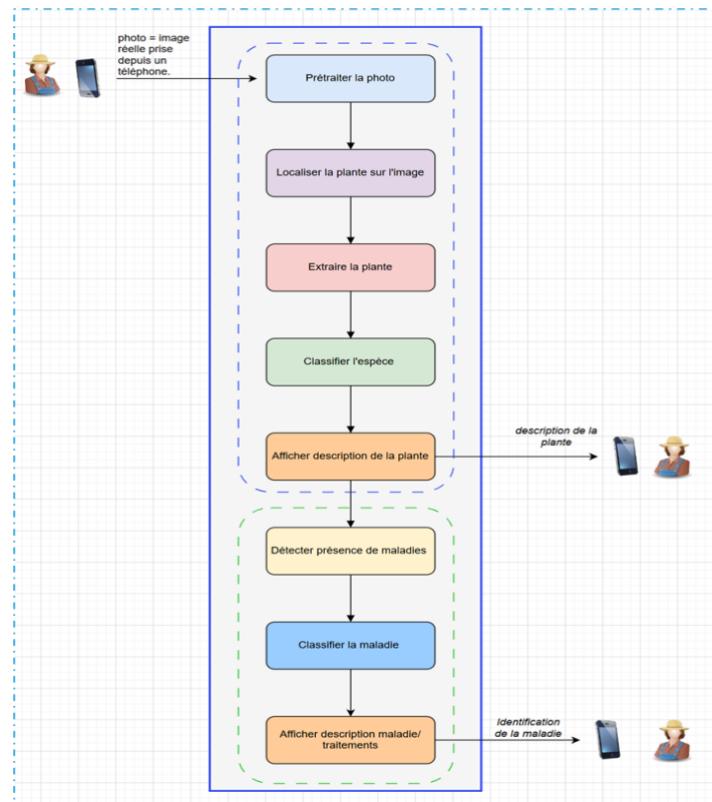


Figure 1 : Scénario global

Parmi les scénarios envisagés, « Détection de plantes invasives ou nuisibles parmi des plantules de maïs », « Surveillance de la croissance de plantes de maïs », « Identifier une espèce », « identifier la maladie de la plante », nous avons retenu le scénario suivant : A partir d'une photo d'une feuille de plante **cadrée** :

- Identifier la classe/espèce de la plante
- Dire si la plante est malade ou saine
- Si elle est malade, donner le nom de la maladie

Les autres étapes n'ont pas pu être mises en place faut de temps.

1.2.1. Objectifs

Le scénario nous a amené à définir les 3 objectifs suivants :

Objectif 1 – Classification de l'espèce – Quelle est cette plante ?

Développer un modèle (multi-classe) capable de reconnaître automatiquement l'espèce de chaque feuille (orange, maïs, tomates, etc.) à partir de ses caractéristiques de forme, couleur et texture...

Objectif 2 – Détection de l'état de santé – La plante est-elle malade ?

Mettre en place un classifieur binaire pour distinguer les feuilles saines de celles présentant des symptômes de maladie.

Objectif 3 – Identification du type de maladie – Quelle est la maladie de la plante ?

Pour les feuilles diagnostiquées « malade », développer un modèle (multi-classe) déterminant précisément la maladie (tavelure, mildiou, rouille, etc.)

Notre projet se situe donc dans un contexte d'apprentissage supervisé. Nos variables cibles sont : le nom de la plante, l'état de santé et le nom de la maladie. Les variables explicatives seront les caractéristiques des feuilles (forme, couleur, contour...).

1.2.2. Revue littéraire

Nous avons effectué un état des lieux des applications, des datasets et des critères de qualité compte tenu de nos objectifs. Nous avons exploré plusieurs publications. Celle qui a retenu notre attention est une [revue systématique](#) des méthodes les plus performantes de Machine Learning, de traitement d'image, et d'algorithmes de Deep Learning appliqués à la reconnaissance des espèces végétales, publiée en 2024. Elle confirme le Deep Learning (DL) comme technique la plus performante. Cependant, lors du passage de jalon n°1, il a été convenu d'effectuer tout le cycle de vie du projet (étapes Machine Learning, Deep Learning) dans un esprit de mise en application des techniques apprises dans le cursus DataScientist.

Grâce à la revue littéraire et à l'état des lieux réalisés, nous avons pu :

- Cerner les enjeux du projet et définir ses objectifs de manière précise, tout en identifiant les risques associés, compte tenu de la complexité du sujet.
- Sélectionner les datasets les plus adaptés, en fonction de leurs caractéristiques et de leur pertinence pour l'étude.

- Présélectionner des modèles de Machine Learning et de Deep Learning, en vue de leur application dans le cadre du projet.
- Établir une méthodologie claire, encadrant les différentes étapes de la démarche.

1.3. Organisation du document

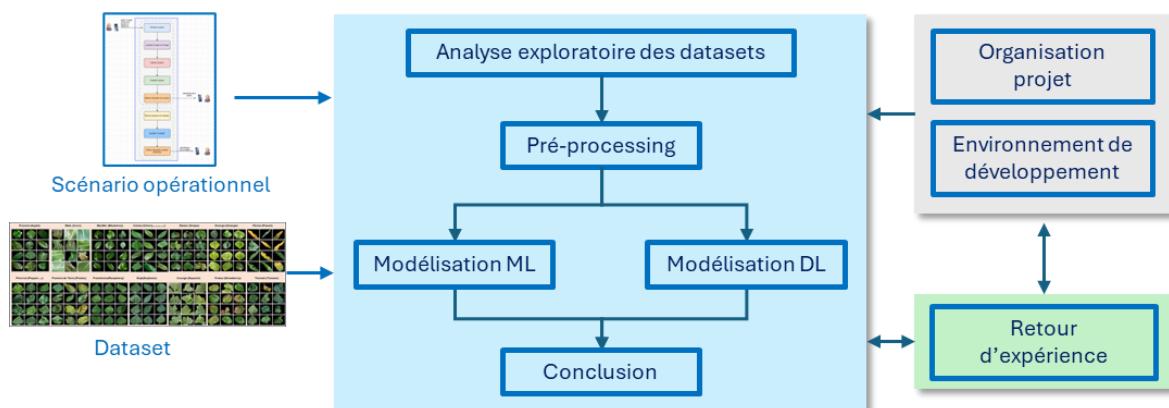


Figure 2: organisation du document

Ce schéma synthétise l'organisation générale du document. Les chapitres suivants décrivent d'abord l'organisation du projet puis l'environnement de développement, qui constituent le cadre dans lequel le travail a été mené. Le cœur du document suit une chaîne de traitement structurée : exploration du dataset, puis pré-processing (nettoyage, préparation des labels, transformations éventuelles). Les données préparées sont ensuite exploitées selon deux approches distinctes de modélisation, en ML d'une part et en DL d'autre part. **L'objectif n'est pas de réaliser une étude comparative entre ces deux familles de méthodes – la littérature couvrant déjà largement ce sujet – mais d'explorer un spectre de modèles suffisamment large pour mettre en pratique les notions abordées dans la formation de DataScientist. Dans ce cadre, le périmètre étudié en ML reste volontairement plus restreint que celui exploré en DL, comme cela sera détaillé dans les chapitres concernés.** Les résultats obtenus sont ensuite synthétisés dans une section de conclusion. Enfin, un retour d'expérience vient clore le rapport en discutant les difficultés rencontrées, les limites de l'étude et les pistes d'amélioration pour l'organisation du projet et l'environnement technique.

1.4. Organisation du projet

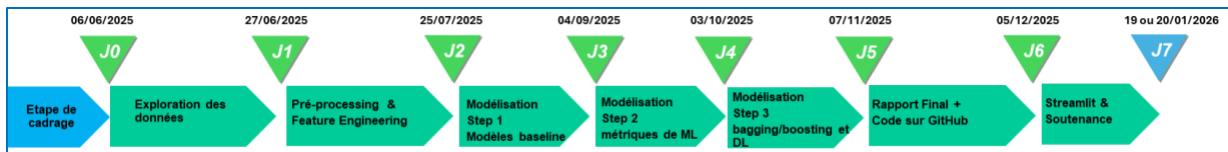
Les points clés de la gestion du projet :

Réunions : Hebdomadaire et au besoin (pair programming, installations).

Pratique : Tout au long du projet, pour les étapes d'analyse exploratoire des datasets, de modélisation Machine Learning (ML) et Deep Learning (DL), nous avons favorisé : une phase d'exploration informelle par chaque membre de l'équipe et une phase de synthèse élaborée par l'équipe. C'était le meilleur moyen de s'assurer de la mise en application de notre formation et de l'émergence des idées.

Cycle de vie du projet et revues

Le cycle de vie du projet est le suivant :



L'avancement et l'orientation du projet s'est appuyé sur les présentations de nos travaux et les décisions prises lors de chaque revue avec notre référent.

Risques projet

Risque 1 : Beaucoup d'itérations

- Cause : Le démarrage du projet arrive trop tôt par rapport au planning des apprentissages nécessaires pour le projet.
- Mitigation : Lire les modules des cours par anticipation

Risque 2 : Des jalons non tenus

- Cause : La performance de nos PC
- Mitigation : VM de DataScientest, Google Colab, Changer de PC

Risque 3 : Erreurs probables sur les interprétations

- Cause : La durée entre jalons très courte
- Mitigation : Les recommandations des analyses exploratoires ne seront pas toutes intégrées mais elles seront révisées régulièrement.

1.5. Environnement de développement

L'environnement de développement du projet s'appuie principalement sur **Visual Studio Code**, associé à **Git** pour le versionnement et le travail collaboratif : chaque membre de l'équipe a d'abord exploré ses idées sur sa propre branche avant intégration dans la branche principale.

Les dépendances sont gérées avec **Conda** via un fichier env, qui définit notamment **Python 3.11**.

Les principales librairies utilisées sont scikit-learn, pandas, matplotlib, TensorFlow. Les versions utilisées ont pu différer pour cause de GPU récents.

Nous avons privilégié des scripts Python structurés (.py) plutôt que des notebooks, afin de faciliter la factorisation du code, sa réutilisation et les revues de code.

Selon les ressources matérielles disponibles, certains membres ont travaillé sur des machines locales équipées de GPU en s'appuyant sur **Docker** pour garantir la reproductibilité des environnements, tandis que d'autres ont utilisé **GitHub Codespaces** pour accéder à des ressources GPU dans le cloud.

Nous avons également travaillé avec l'outil **Windsurf**, ce qui nous a permis de nous concentrer sur des tâches à plus forte valeur ajoutée, pour se rapprocher des pratiques en entreprise.

L'arborescence détaillée du dépôt GitHub est décrite dans un fichier **README.md** afin de faciliter la prise en main du projet.

En fin de projet, à la suite du cours sur le suivi d'expériences, nous avons mis en place le traçage systématique des entraînements et des métriques avec **MLflow** déployé sur Azure, ce qui a amélioré la traçabilité, la comparaison et la reproductibilité de nos expérimentations.

2. Analyse exploratoire des datasets

2.1. Sélection du dataset

A partir des 6 datasets proposés par DataScientest, nous avons effectué plusieurs sélections successives basées sur des explorations afin de n'en retenir qu'un, PlantVillage.

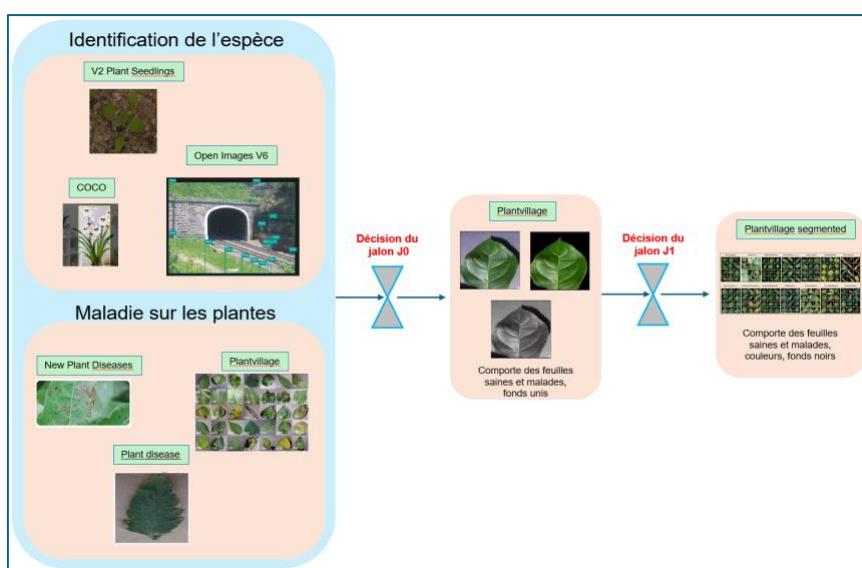


Figure 3 : Sélection du dataset pour premier cycle

Les 3 datasets V2 Plant Seedlings, Open Images V6 et COCO contenant des images de plantes avec des environnements différents et peu d'espèces communes entre ces 3 datasets, sont éliminés car le scénario part d'une photo feuille cadrée sur fond uni. Le tableau suivant compare les 3 autres datasets permettant la détection des maladies.

Caractéristique	PlantVillage	Plant Disease	New Plant Diseases
Taille du dataset	~54,000 images 38 classes (espèce + maladie ou healthy)	~54,000 images	~88,000 images
Nombre d'espèces	14 espèces de plantes	38 classes de maladies sur plusieurs espèces	> 60 espèces de plantes
Nombre de maladies	26 maladies (avec healthy classes)	>50 maladies (certaines plantes avec plusieurs maladies)	~120 maladies (certaines classes rares)
Type d'image	Images avec des fonds unis	Images avec des fonds unis	Images avec des fonds unis
Annotation	Espèce + maladie + <u>healthy</u>	Espèce + maladie	Espèce + maladie + conditions environnementales
Format	JPG / PNG	JPG	JPG / JPEG / PNG
Accessibilité	Ouvert (Kaggle, GitHub)	Ouvert (Kaggle)	Ouvert (Google Dataset Search, Zenodo, etc.)

Figure 4 : Comparatif des datasets

Plant Disease est éliminé car pour un même nombre d'images, il fournit plus de types de maladies. Cela n'apporte rien à notre scénario. New Plant Disease est créé à l'aide d'une augmentation hors ligne de PlantVillage de 34000 images supplémentaires. Notre analyse

exploratoire a montré que certaines espèces majoritaires ont été augmentées plus que d'autres pour couvrir un objectif non précisé dans la littérature. Notre choix se porte sur PlantVillage qui cadre bien à notre scénario. Sa structure est détaillée dans le chapitre suivant.

2.2. Exploration du dataset PlantVillage/Segmented

Le dataset PlantVillage est structuré en 3 dossiers : color, grayscale et segmented. Chaque dossier comporte 54306 images, réparties dans 38 sous-dossiers (les 38 classes). Il y a 14 espèces de plantes et 20 classes de maladies. La taille du dossier est de 593 Mo. La variante color correspond aux images RGB d'origine, où la feuille apparaît avec son fond (souvent simple), tandis que la variante segmented contient les feuilles segmentées, le fond ayant été supprimé et remplacé par un fond noir. Nous avons démarré avec la variante Segmented, constituant un point d'entrée plus simple, compte tenu de notre absence de pratique préalable en ML et en DL.

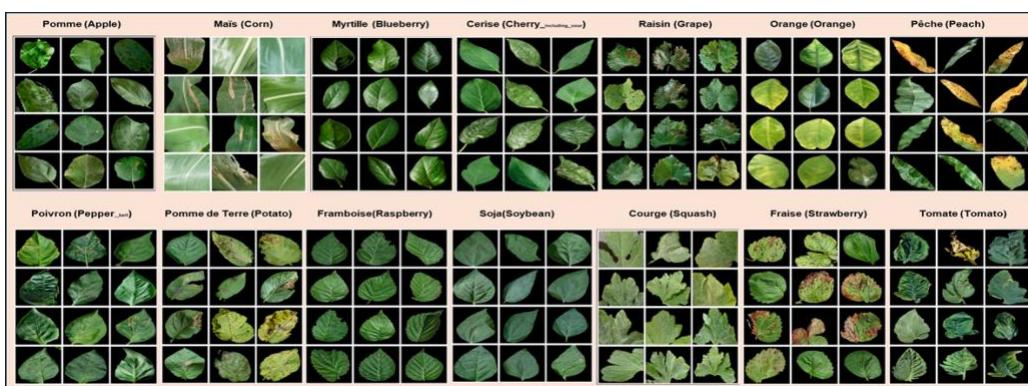
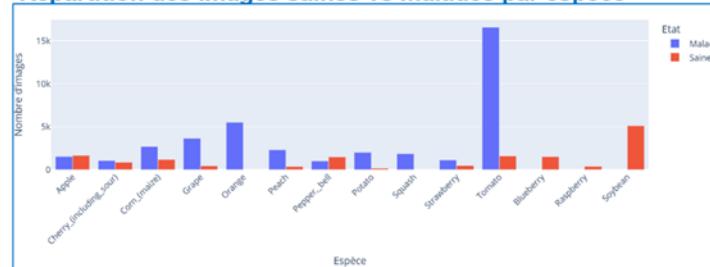


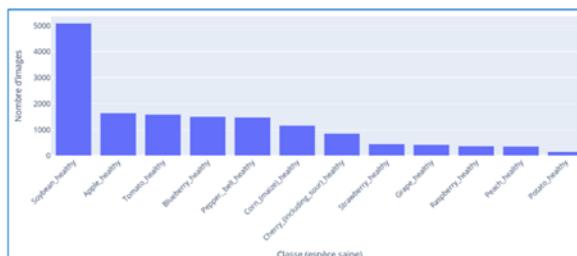
Figure 5 : extrait des images du dataset Plantvillage/segmented

Répartition des images saines vs malades par espèce, Distribution des classes saines, Distribution des classes malades

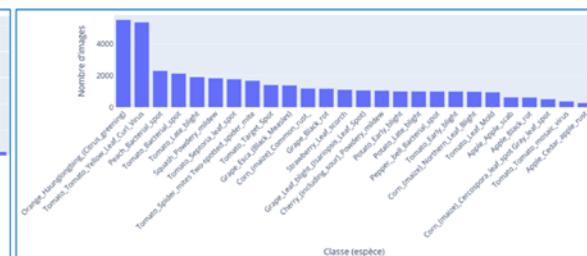
Répartition des images saines vs malades par espèce



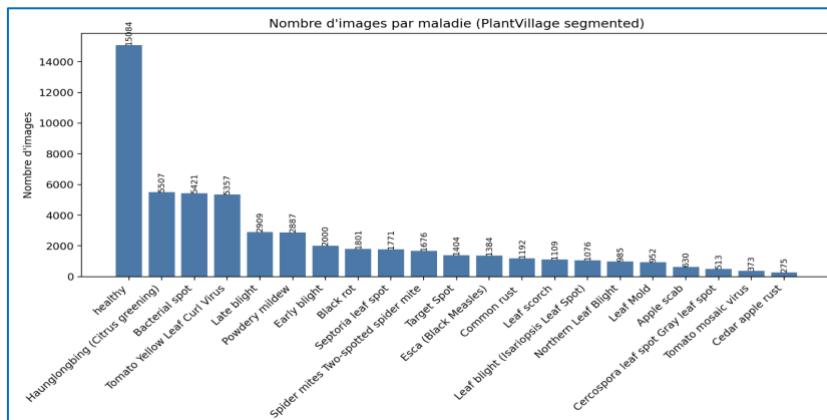
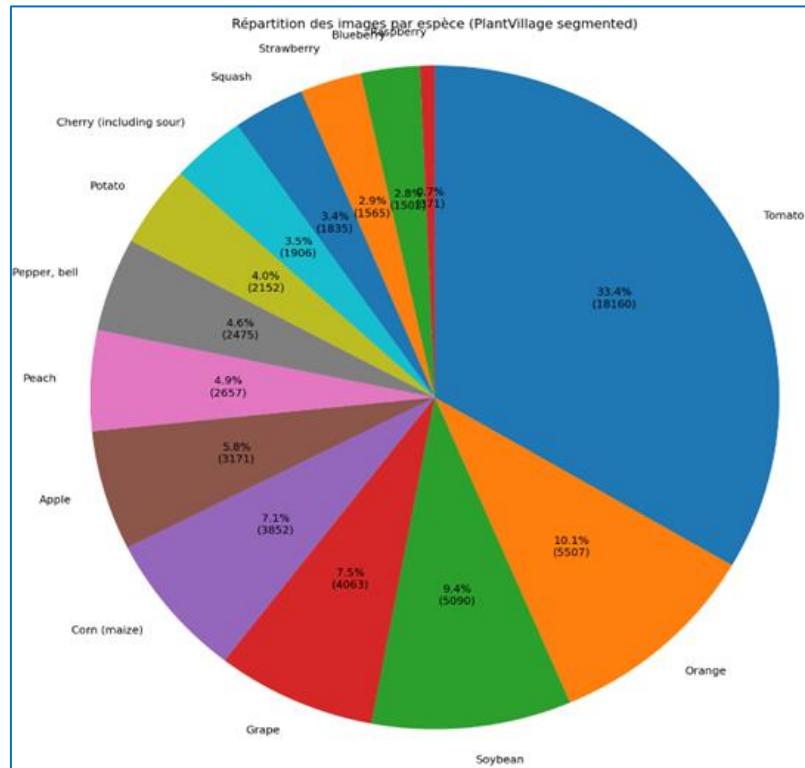
Distribution des classes saines"



Distribution des classes malades"



Nombre d'images par espèce et distribution des classes maladie :



Observations :

- Le dataset montre un très fort déséquilibre entre espèces : Tomato domine avec près de 18 000 images, tandis que Raspberry n'en compte qu'environ 300. Orange et Grape figurent aussi parmi les plus abondantes, alors que Blueberry ou Peach restent peu représentées.
- Au sein de certaines espèces (Tomato, Orange, Grape...), les images malades sont majoritaires, alors que pour d'autres (Pepper_bell, Potato, Squash...) les exemples sains l'emportent. Les maladies comme Orange_Huanglongbing ou Tomato_Yellow_Leaf_Curl_Virus totalisent chacune près de 5 000 images.
- En revanche, de nombreuses maladies rares forment une longue traîne avec moins de 300–500 images.
- Le dataset présente une structure où chaque maladie est associée à une seule espèce.

Risques :

Risques liés au déséquilibre des données :

- Biais vers les classes majoritaires : La surreprésentation de certaines espèces (par exemple Tomato) et, plus généralement, des classes majoritaires pousse le modèle à privilégier ces catégories au détriment des classes rares (par exemple Raspberry), pour lesquelles la diversité d'exemples est insuffisante pour capturer des caractéristiques visuelles spécifiques, ce qui augmente les erreurs de classification.

Risques liés à l'intégration de l'espèce et de l'état de santé dans le classifieur "maladie" :

- Biais croisé espèce/santé → maladie : Le modèle peut "apprendre par raccourci" que certaines maladies collent à certaines espèces/états (ex. Tomato souvent malade, Pepper souvent sain) et s'appuyer davantage sur ces méta-information que sur les indices visuels des lésions.
- Seuils et calibrations non spécifiques à l'espèce : Le modèle utilise un classifieur de maladie unique pour toutes les espèces, avec les mêmes frontières de décision et les mêmes seuils healthy/malade. Or la répartition des symptômes et la fréquence de "healthy vs diseased" varient fortement d'une espèce à l'autre.
- Ignorance des maladies rares : Les maladies fréquentes dominent l'apprentissage et la décision ; les catégories rares (souvent spécifiques à une espèce) voient leur rappel chuter, augmentant faux négatifs et faux positifs (confusion vers 3–5 maladies majeures).
- Propagation d'erreurs et décalage entraînement/inférence : Si l'espèce/santé est prédite en amont et servie en entrée, toute erreur se répercute sur la maladie.
- Combinaisons espèce–maladie figées : seules certaines paires espèce–maladie sont présentes. Le modèle apprend implicitement que "les autres combinaisons n'existent pas". Cela limite les prédictions absurdes sur le dataset, mais pose problème si, en situation réelle, une maladie apparaît sur une espèce qui n'était pas observée.
- Mauvaise séparation "healthy" vs maladies : Lorsque la classe "healthy" est traitée comme une maladie de plus dans la tête de classification (ou que l'état sain/malade est injecté comme feature), le fort déséquilibre entre feuilles saines et malades selon les espèces peut déformer les frontières de décision.

Recommandations :

Les mesures suivantes visent à corriger les biais, améliorer la généralisation du modèle, et garantir une meilleure détection des classes minoritaires :

- Privilégier un split stratifié afin de préserver les proportions réelles des espèces et des classes saines/malades.
- Appliquer un sous-échantillonnage des classes surreprésentées et un sur-échantillonnage ou une augmentation de données pour les espèces et classes sous-représentées.
- Utiliser des class weights équilibrés pour renforcer la prise en compte des catégories rares et corriger les biais liés aux déséquilibres.
- Organiser la prédiction en plusieurs niveaux au lieu d'une seule tête globale. Par exemple :
 - d'abord une étape simple (par espèce) pour décider "healthy vs malade" ;
 - puis, seulement sur les feuilles malades, une étape multiclasse pour choisir la maladie correspondante.
- On peut aussi fournir explicitement le nom l'espèce en entrée du classificateur de maladie.
- Métriques adaptées : f1-score par classe, recall, confusion matrix plutôt qu'accuracy brute.

Extraction des labels

Pour couvrir les 3 objectifs du projet, nous avons identifié les cibles suivantes : espèce (species), santé(healthy), maladie(disease).

2.3. Pré-processing

L'analyse exploratoire a montré qu'un pré-processing du dataset PlantVillage/Segmented était indispensable pour nos modélisations.

Dans notre travail, les modèles de Deep Learning sont entraînés sur la variante color, tandis que les approches ML classiques exploitent des descripteurs calculés sur des versions segmentées des feuilles.

Ce paragraphe présente le pré-processing **commun** aux deux types de modélisation, appliqué aux 2 dossiers segmented et color:

Nettoyage des données : Appliquer les mêmes règles de filtrage dans les deux pipelines :

- Nous avons détecté et supprimé 18 images quasi noires (dont une complètement noire).
- Nous avons détecté et supprimé 21 doublons.
- Redimensionnement uniforme des images : Sur les 54267 images, 4 images ont été redimensionnées à 256 x 256, toutes les autres étant en 256x 256.
- L'extension .jpg a un nombre très élevé d'images, soit 54302 images et les extensions .jpeg et .png ont chacune seulement 2 images. Leur extension sera modifiée.
- Nous avons supprimé 10 images inexploitables (morceaux de feuilles)

Le même mapping a été effectué entre images et labels species / disease / healthy pour que les modèles ML et DL apprennent exactement la même tâche.

3. Modélisation

Sachant que le Deep Learning fournit les meilleurs résultats, nous n'avons pas approfondi l'exploration des approches de Machine Learning, ni mené de comparaisons détaillées entre plusieurs modèles ML pour sélectionner une solution optimale, réservant ces efforts d'optimisation au Deep Learning.

Par abus de langage, nous emploierons dans la suite du rapport le terme ML pour désigner le Machine Learning hors approches de Deep Learning.

3.1. Machine Learning

Comme déjà mentionné, la variante segmented de PlantVillage est utilisée pour ML.

3.1.1. Méthodologie

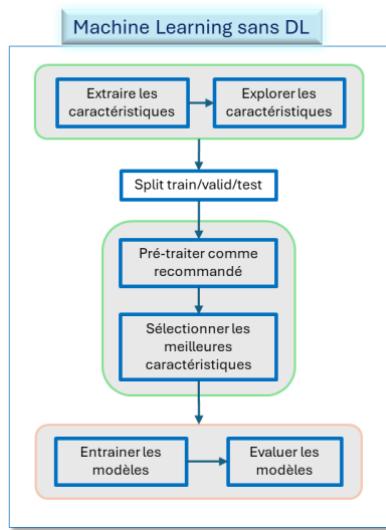


Figure 6 : Schéma de principe des activités à mener pour ML (sans DL)

Les modèles ML ne traitent pas directement les pixels bruts d'une image. Il faut extraire des caractéristiques numériques, pour les utiliser comme entrées et effectuer une exploration statistique. Un rééchantillonnage (sur le train uniquement) sera nécessaire. Les données extraites sont divisées en trois ensembles distincts : train, valid, test. Avant l'entraînement, deux pré-traitements clés sont appliqués : Augmentation des données (uniquement sur train) et Standardisation/ normalisation des caractéristiques. Puis les caractéristiques les plus informatives sont sélectionnées avec SHAP. Les modèles sont entraînés sur train, ajustés sur valid et évalués sur test. Des métriques adaptées (précision, rappel, F1-score, etc.) permettent de mesurer leurs performances et de sélectionner la meilleure approche.

3.1.2. Extraire les caractéristiques

Pour chaque image, nous avons extrait des descripteurs visant à capturer différents aspects visuels discriminants. Ces caractéristiques sont regroupées en plusieurs catégories :

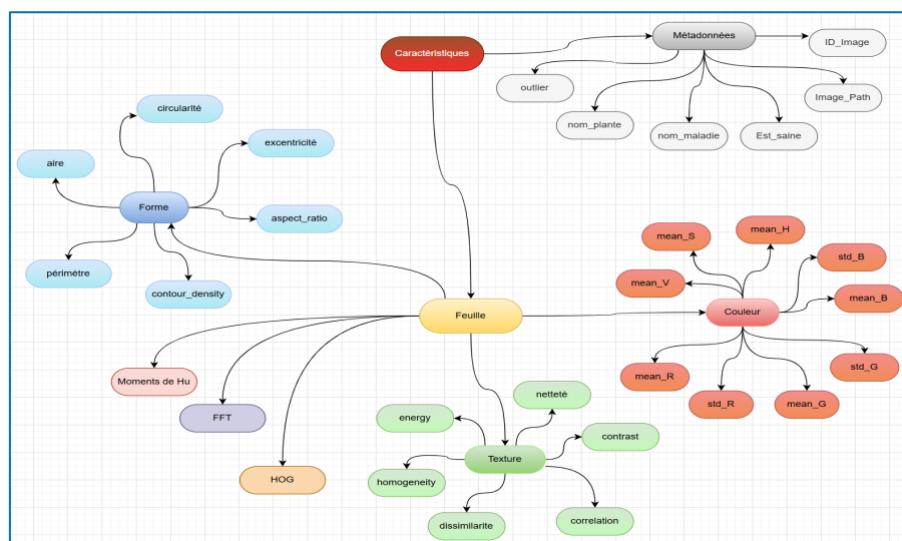


Figure 7 : Caractéristiques extraites des feuilles de plantes

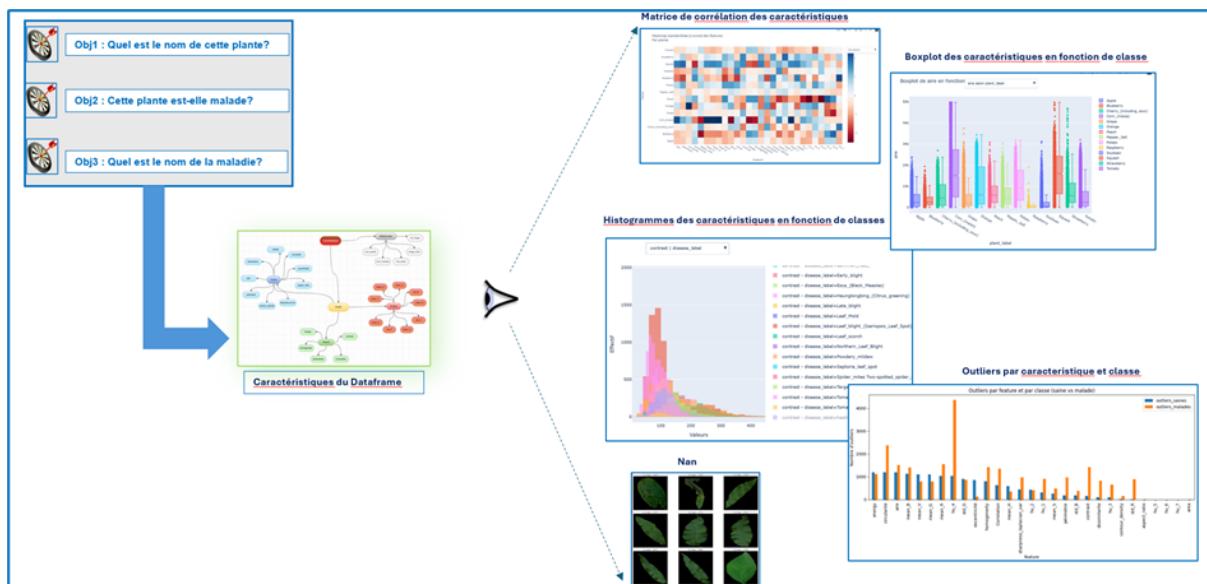
- **Caractéristiques morphologiques** : superficie (aire), périmètre, circularité, excentricité, rapport d'aspect (aspect ratio) et densité de contours. Ces indicateurs décrivent la forme globale des objets présents sur l'image.
- **Caractéristiques colorimétriques** : moyennes et écarts-types des canaux RGB (mean/std_R, G, B), ainsi que les moyennes des composantes HSV (mean_H, S, V), permettant de représenter les couleurs dominantes et leur variation.
- **Caractéristiques de texture** : netteté, contraste, energy, homogeneity, dissimilarity, correlation, calculées à partir de matrices de co-occurrence, décrivent les variations locales d'intensité dans l'image.
- **Descripteurs invariants** : les moments de Hu (hu_1 à hu_7) permettent de capturer la forme de manière invariante à la rotation, à la translation et au changement d'échelle.
- **Descripteurs fréquentiels** : les coefficients extraits de la transformée de Fourier (fft_energy, fft_entropy, low/high frequency power) communiquent une information sur la répartition spectrale des détails dans l'image.
- **Descripteurs de gradient** : les descripteurs HOG (moyenne, écart-type, entropie) résument les orientations dominantes des gradients, utiles pour capturer les structures visuelles.

Ces descripteurs sont concaténés pour former un vecteur unique par image, servant ensuite d'entrée aux algorithmes de classification. Les caractéristiques générées sont consignées dans un tableau (annexe [7.7](#)), précisant pour chaque descripteur : sa source ou librairie d'origine, ainsi que sa fonction ou utilité dans le cadre de notre projet.

Certaines variables sont directement issues du traitement d'image (par exemple via OpenCV, NumPy, skimage.feature ou la transformée de Fourier), tandis que d'autres jouent un rôle de support (identifiant de la plante, label, cible, ou métadonnée de structure). Cette organisation facilite l'analyse, la traçabilité, ainsi que la future sélection des features les plus discriminantes pour la phase de classification.

3.1.3. Explorer les caractéristiques extraites

L'analyse exploratoire des données a permis d'examiner la qualité, la structure et les spécificités des images. Cette étape est cruciale pour identifier les défis potentiels et orienter les choix méthodologiques pour la modélisation. Le détail de l'analyse exploratoire se trouve en annexe [7.2](#).



Gestion des Données Manquantes et Outliers

- **Données manquantes** : 9 images présentaient des valeurs manquantes (NaN) et ont été supprimées. Ces images restent accessibles pour une analyse ultérieure si nécessaire.
- **Outliers** : La méthode **IQR** a révélé un taux élevé d'outliers (40,34% des données), principalement concentrés dans la classe "malade". Certaines caractéristiques comme la *circularité*, *fft_entropy*, et *hu_4* présentaient une surreprésentation marquée d'outliers pour les plantes malades.
- **Risques** : Ces outliers pourraient biaiser les modèles sensibles aux valeurs extrêmes ou refléter une hétérogénéité biologique légitime.
- **Recommandations** :
 - Tester l'impact de leur suppression ou transformation (ex. : winsorisation) sur les performances des modèles.
 - Privilégier des modèles robustes (ex. : Random Forest, SVM avec noyau RBF) ou des techniques de normalisation adaptées.

Distribution des Caractéristiques par Objectif

Objectif 1 : Identification de la Plante

- **Observations** : Les distributions des caractéristiques (ex. : mean_G, mean_H) varient significativement entre espèces, suggérant un fort potentiel discriminatif pour la classification. Certaines variables (ex. : canaux de couleur, moments de Hu) présentent une redondance, tandis que d'autres sont peu informatives pour certaines classes.
- **Risques** : Déséquilibre entre espèces risque de biaiser le modèle vers les classes majoritaires.
- **Recommandations** :
 - Sélection de variables pour réduire la redondance (ex. : Analyse de Corrélation, PCA).
 - Techniques de rééchantillonnage (ex. : SMOTE) ou pondération des classes pour atténuer le déséquilibre.

Objectif 2 : Détection de l'État de Santé (Sain vs. Malade)

- **Observations** : Les variables morphologiques (aire, périmètre) et colorimétriques (mean_R, mean_G) montrent des différences nettes entre plantes saines et malades. Les plantes malades présentent souvent une aire réduite et des valeurs de couleur distinctes.
- **Recommandations** : Prioriser ces caractéristiques pour construire un modèle binaire robuste. Utiliser des visualisations interactives (ex. : boxplots) pour affiner la sélection des variables les plus discriminantes.

Objectif 3 : Diagnostic Spécifique de la Maladie

- **Observations** : Forte disparité dans la représentation des maladies : certaines sont surreprésentées (ex. : Apple_scab), tandis que d'autres ne comptent que quelques exemples. Des caractéristiques comme la *contour_density* permettent de distinguer certaines maladies (ex. : Apple_scab vs. Black_rot).
- **Risques** : Déséquilibre extrême entre maladies, risquant de limiter la détection des pathologies rares.
- **Recommandations** :
 - Data Augmentation ciblée pour les maladies sous-représentées.
 - Modèles hiérarchiques : Classifier d'abord l'espèce, puis la maladie pour améliorer la précision.

Relations entre Caractéristiques

- **Corrélations** : Les heatmaps ont révélé des groupes de variables fortement corrélées (ex. : canaux de couleur, moments de Hu), ainsi que des profils distincts entre espèces et maladies. Les plantes saines présentent des profils de caractéristiques plus homogènes, tandis que les plantes malades montrent des déviations marquées pour certaines variables.
- **Risques** : La corrélation entre variables pourrait entraîner un surapprentissage. A l'intérieur d'une même "classe" (ex. une maladie donnée), les images sont très différentes entre elles. Le modèle doit donc apprendre une frontière de décision unique pour couvrir des sous-cas très variés, ce qui complique la généralisation des modèles.

- **Recommandations :**

- Réduction de dimensionnalité (PCA, t-SNE) pour visualiser et sélectionner les variables les plus pertinentes.
- Ingénierie de caractéristiques pour combiner les variables redondantes en indices synthétiques (ex. : ratio de couleurs, indices de texture).

Conclusions

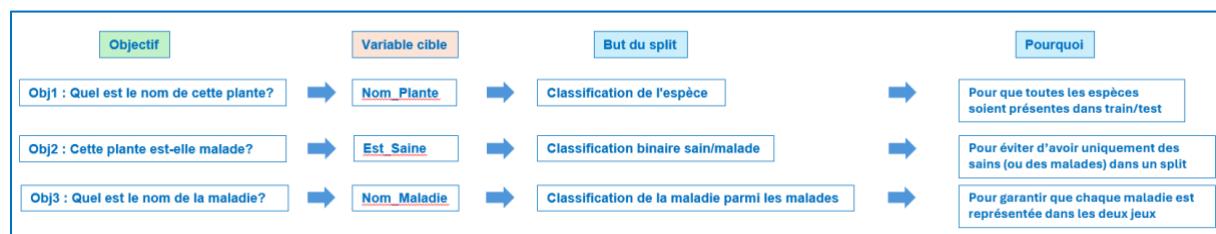
- Les caractéristiques morphologiques et colorimétriques sont fortement discriminantes pour distinguer les plantes saines des malades.
- Le déséquilibre des classes (espèces/maladies) nécessite des stratégies de rééchantillonnage ou des modèles adaptés.
- La présence d'outliers et de redondances impose une préparation rigoureuse des données avant modélisation.

Les boxplots et heatmaps générés, livrés au jalon n°2 nous ont offert une base solide pour affiner l'analyse et guider le choix des caractéristiques les plus pertinentes.

3.1.4. Split train/valid/test

Pour garantir la qualité de nos modèles, nous avons séparé le dataset en trois parties : un ensemble d'entraînement pour apprendre, un ensemble de validation pour choisir les meilleurs hyperparamètres et améliorer les modèles, et un ensemble de test pour évaluer les performances réelles des modèles entraînés sur des données totalement nouvelles

Le déséquilibre des classes constaté nous impose par ailleurs des découpages du jeu de données (train, val, test) stratifiés, c'est-à-dire en respectant la proportion des classes cibles pour chaque objectif :



3.1.5. Rééchantillonnage

Face au déséquilibre de classe, nous avons testé plusieurs techniques de rééchantillonnage en fonction des modèles ML testés :

- RandomOverSampler vs SMOTE pour SVM-RBF : les meilleurs couples sont RandomOverSampler+StandardScaler et

SMOTE+StandardScaler ($f1_{macro}$ CV ≈ 0.8864), alors que RobustScaler est en retrait (~0.866).

Nous avons créé un fichier csv comprenant le résultat de rééchantillonnage après le split, sur train pour équilibrer les classes :

- oversampling ciblé des espèces/maladies minoritaires
- undersampling léger des classes sur-représentées

Remarque : Compte tenu des modèles ML que nous avons utilisé (SVM, Logistic Regression, XGBoost et Extra-Trees), nous n'avons pas eu besoin d'utiliser le fichier csv rééchantillonné. Ces modèles ML possèdent souvent un paramètre (`class_weight`, ...) pour gérer le déséquilibre automatiquement, qui est souvent plus robuste que le rééchantillonnage artificiel. Le fichier csv a été utilisé pour SVM-RBF avec des résultats moindres.

3.1.6. Pré-traitements

3.1.6.1. Data augmentation

Nous avons réalisé de la data-augmentation, sur train uniquement, afin de réduire le déséquilibre des classes, enrichir la diversité des cas et limiter le sur-apprentissage.

Nous avons appliqué une combinaison des méthodes suivantes : Flip, Rotation, Crop aléatoire, Translation, Modification de la luminosité, contraste, bruit, Zoom in/out, Transformation couleur, Gaussian blur, sharpness.

Nous avons maintenant 91770 images

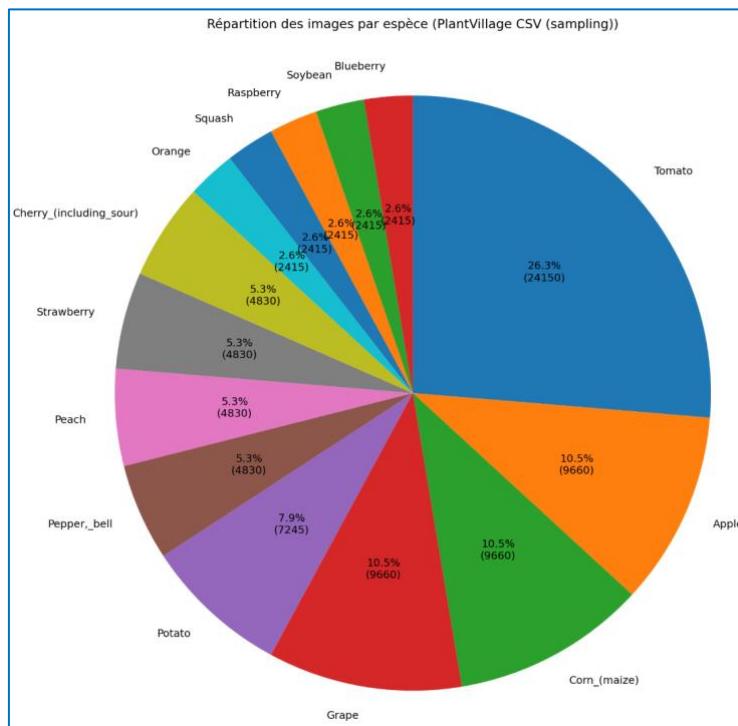


Figure 8 : Nouvelle répartition avec la data augmentation

3.1.6.2. Standardisation / normalisation des features

Lors de l'analyse exploratoire, nous avons observé un taux d'outliers élevé pour certaines variables (hu_4, circularité, aire, mean_R, mean_B, etc.) ce qui implique que les méthodes "classiques" de scaling (standardisation, min-max) seront fortement influencées par ces valeurs extrêmes. Afin de garantir que nos modèles ne soient pas biaisés par des valeurs extrêmes ou des ordres de grandeurs différents nous avons :

- Appliqué un RobustScaler fit uniquement sur l'ensemble d'entraînement
- Transformé ensuite les ensembles de validation et de test avec ce même scaler préalablement ajusté.

L'analyse des distributions de certaines variables montrent que certaines variables avaient des ordres de grandeur très différents (ex: aire entre 0 et 50000, circularité entre 0 et 1).

3.1.7. Sélection des meilleures caractéristiques

Cette phase consiste à ne retenir que les caractéristiques les plus pertinentes en réduisant la dimension du dataset, en éliminant le bruit ou les variables non informatives, en améliorant la performance et la vitesse des modèles. Nous avons comparé plusieurs familles de méthodes de sélection de variables (filtre, wrapper, intégré) pour mesurer l'importance des caractéristiques (voir liste des méthodes annexe [7.3](#)), à l'aide d'un pipeline par sélecteur avec standardisation (RobustScaler), validation croisée stratifiée (StratifiedKFold) et un seul classifieur commun : RandomForestClassifier pour rendre les comparaisons cohérentes. Raisons du choix : robustesse au sur-apprentissage, gestion des classes déséquilibrées (class_weight='balanced') et capacité à modéliser des relations non linéaires. Les métriques suivies incluent accuracy, F1 macro, précision et rappel. Les deux graphiques suivants présentent les résultats obtenus.

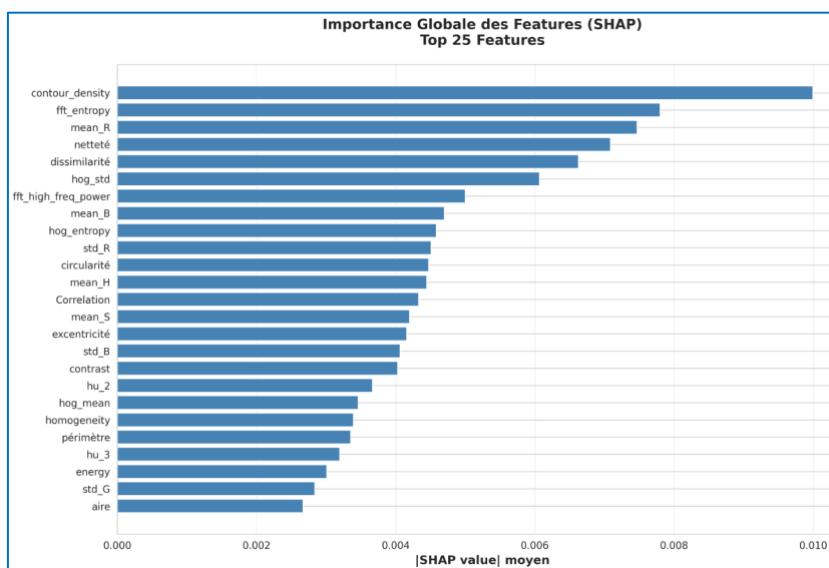


Figure 9 : Importance globale des variables (SHAP) — Top 25

La densité de contour (contour_density) domine très nettement l'importance globale des features, avec une valeur SHAP moyenne de 30% supérieure à la deuxième feature la plus importante. Les features de fréquence spectrale (fft_entropy) et de couleur (mean_R, mean_B) occupent le trio de tête, confirmant que la forme des lésions et leurs

caractéristiques colorimétriques sont les indicateurs les plus discriminants pour la classification des maladies.

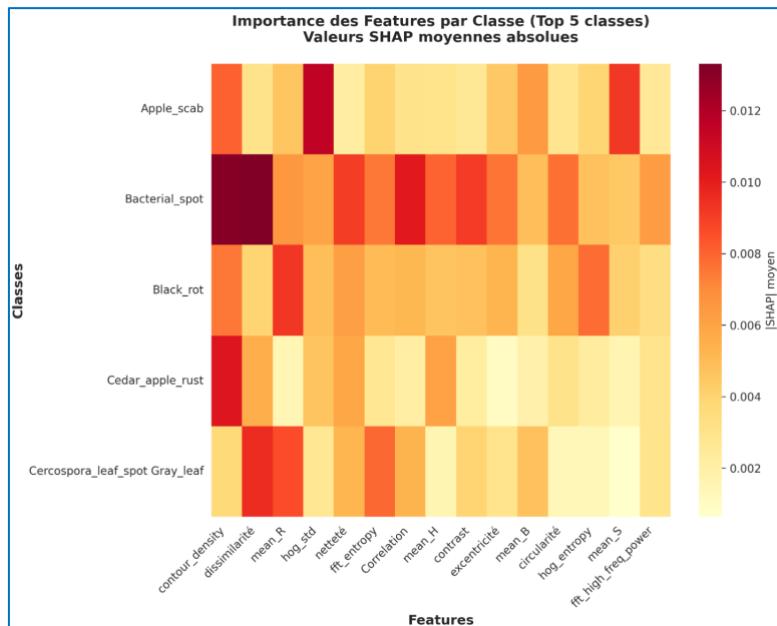


Figure 10: Importance des features par classe (SHAP) — Top-5 classes

L'analyse par classe révèle des signatures de features distinctes pour chaque maladie : par exemple, hog_std (texture) est extrêmement discriminant pour Apple_scab (valeur SHAP ~0.013) mais beaucoup moins pour les autres maladies. À l'inverse, contour_density présente une importance élevée et relativement uniforme pour plusieurs maladies, indiquant qu'il s'agit d'une feature généraliste importante pour détecter les anomalies foliaires. Cette variabilité confirme que différentes maladies se manifestent par des combinaisons spécifiques de caractéristiques visuelles (forme, couleur, texture).

En synthèse, ces 2 graphiques nous apprennent :

- Qu'aucune catégorie unique ne suffit - il faut combiner : la géométrie (contour, circularité, excentricité), la couleur (mean_R, mean_B, std_R), la texture (HOG descriptors) et la fréquence (FFT entropy et power)
- Que chaque classe de maladie s'appuie sur un sous-ensemble différent de features, justifiant l'utilisation d'un modèle complexe capable de capturer ces patterns spécifiques
- Que les 34 features extraites sont toutes pertinentes, aucune n'est totalement négligeable (même les moins importantes dans le top 25 contribuent).

3.1.8. Entrainement des modèles et évaluations

Nous avons réparti le travail en équipe avec un modèle par membre (SVM, XGBoost, Extra-Trees, Régression Logistique) pour la classification des plantes. Chaque membre a adopté sa propre méthodologie : optimisation poussée des hyperparamètres (exploration bayésienne, GridSearch), configurations variées, ou traitement simultané des trois objectifs (espèce, santé, maladie). Pour permettre une comparaison équitable, nous présentons ici uniquement les résultats de l'objectif 1 : identification de l'espèce, permettant d'évaluer transversalement les performances et la robustesse de chaque approche.

Méthodologie commune :

- Dataset: plantvillage/segmented nettoyé
- Standardisation/normalisation des caractéristiques
- Split Train/Val/Test (stratifié)
- Validation croisée stratifiée (ex. : 5-fold) pour éviter le surapprentissage.

Métriques choisies

Pour la classification binaire (plante saine vs malade) : précision, le rappel, et le F1-score. Ces métriques permettent de mesurer respectivement la capacité du modèle à éviter les faux positifs, à détecter correctement les cas positifs (malades), et à trouver un équilibre entre les deux, ce qui est crucial pour limiter les erreurs coûteuses (ex. : fausse détection de santé pour une plante malade).

Pour les problèmes multi-classes (ex. : identification de la plante, de la maladie), l'accuracy globale peut être trompeuse en raison des déséquilibres ; nous utiliserons donc la moyenne des F1-scores par classe (macro-F1) pour garantir une évaluation équitable de toutes les catégories, y compris les maladies rares.

Enfin, la matrice de confusion permettra d'identifier les confusions fréquentes entre classes. Ces métriques, combinées à une validation croisée stratifiée, assureront une évaluation rigoureuse et adaptée aux spécificités des données, notamment la présence d'outliers et de classes minoritaires.

Interprétation des Performances Globales

Les résultats sont présentés dans le tableau comparatif synthétisant les performances pour l'**objectif 1 : Classification de la plante** :

Modèle	Accuracy	Précision (macro avg)	Rappel (macro avg)	F1-score (macro avg)
SVM(RBFVC)	0.9370	0.9271	0.9207	0.9237
XGBoosT	0.9038	0.9051	0.8654	0.8839
Régression Logistique	0.8615	0.8462	0.8214	0.8328
Extra-Trees	0.8310	0.8607	0.7405	0.7863

Figure 11 : Performances des modèles ML sur l'ensemble de test

Le SVM-RBF démontre une excellente capacité de généralisation sur l'ensemble des 14 classes d'espèces. L'équilibre entre précision et rappel témoigne de sa robustesse face à la complexité et au déséquilibre du dataset. XGBoost occupe la deuxième position avec environ 4 points en-dessous du SVM-RBF, avec un rappel légèrement plus faible suggérant quelques difficultés à identifier certaines classes minoritaires. Enfin, Extra-Trees affiche les résultats les plus faibles, avec un rappel particulièrement faible indiquant des difficultés importantes à reconnaître correctement l'ensemble des espèces.

Interprétation des Performances par Classe

Pour une analyse plus fine, les performances ont été détaillées par classe de plante comme détaillé par la figure suivante :

	<u>precision</u>	<u>recall</u>	f1-score	support		<u>precision</u>	<u>recall</u>	f1-score	support
Apple	0.8777	0.8861	0.8819	632	Apple	0.7288	0.7057	0.7170	632
Blueberry	0.9628	0.9500	0.9564	300	Blueberry	0.8845	0.8933	0.8889	300
Cherry_(including_sour)	0.8898	0.8688	0.8792	381	Cherry_(including_sour)	0.7425	0.7113	0.7265	381
Corn_(maize)	0.9730	0.9857	0.9793	769	Corn_(maize)	0.9673	0.9623	0.9648	769
Grape	0.9526	0.9643	0.9584	812	Grape	0.8613	0.8645	0.8629	812
Orange	0.9596	0.9500	0.9548	1101	Orange	0.8807	0.8783	0.8795	1101
Peach	0.9371	0.9248	0.9309	532	Peach	0.8950	0.8496	0.8717	532
Pepper,_bell	0.8972	0.8300	0.8623	494	Pepper,_bell	0.8825	0.6417	0.7132	494
Potato	0.8897	0.8628	0.8760	430	Potato	0.8133	0.7698	0.7909	430
Raspberry	0.8933	0.9054	0.8993	74	Raspberry	0.8529	0.7838	0.8169	74
Soybean	0.9289	0.9371	0.9330	1018	Soybean	0.8335	0.8851	0.8585	1018
Squash	0.9617	0.9591	0.9604	367	Squash	0.9836	0.8937	0.8986	367
Strawberry	0.9183	0.9073	0.9088	313	Strawberry	0.8000	0.7412	0.7695	313
Tomato	0.9455	0.9578	0.9516	3626	Tomato	0.8813	0.9192	0.8998	3626
accuracy			0.9370	10849	accuracy			0.8615	10849
macro avg	0.9271	0.9207	0.9237	10849	macro avg	0.8462	0.8214	0.8328	10849
weighted avg	0.9368	0.9370	0.9368	10849	weighted avg	0.8603	0.8615	0.8601	10849

SVM-RBF

Logistique régression

	<u>precision</u>	<u>recall</u>	f1-score	support
Apple	0.8218	0.7737	0.7971	632
Blueberry	0.9401	0.8900	0.9144	300
Cherry_(including_sour)	0.8764	0.8005	0.8368	381
Corn_(maize)	0.9778	0.9727	0.9752	769
Grape	0.8917	0.9027	0.8972	812
Orange	0.9258	0.9183	0.9220	1101
Peach	0.9480	0.8910	0.9186	532
Pepper,_bell	0.8400	0.6802	0.7517	494
Potato	0.8936	0.7814	0.8337	430
Raspberry	0.9516	0.7973	0.8676	74
Soybean	0.8899	0.9214	0.9054	1018
quash	0.9227	0.9428	0.9326	367
Strawberry	0.8397	0.7700	0.8033	313
Tomato	0.8941	0.9589	0.9253	3626
accuracy			0.8996	10849
macro avg	0.9010	0.8572	0.8772	10849
weighted avg	0.8991	0.8996	0.8981	10849

XGBoost

	<u>precision</u>	<u>recall</u>	f1-score	support
Apple	0.8631	0.5585	0.6782	632
Blueberry	0.9176	0.5200	0.6638	300
Cherry_(including_sour)	0.8399	0.7297	0.7809	381
Corn_(maize)	0.9331	0.9610	0.9468	769
Grape	0.8154	0.8103	0.8128	812
Orange	0.8773	0.8765	0.8769	1101
Peach	0.9362	0.7726	0.8465	532
Pepper,_bell	0.7657	0.4433	0.5615	494
Potato	0.8148	0.6651	0.7324	430
Raspberry	0.9388	0.6216	0.7488	74
Soybean	0.8373	0.9047	0.8697	1018
Squash	0.9081	0.8883	0.8981	367
Strawberry	0.8189	0.6645	0.7337	313
Tomato	0.7835	0.9512	0.8592	3626
accuracy			0.8310	10849
macro avg	0.8607	0.7405	0.7863	10849
weighted avg	0.8356	0.8310	0.8238	10849

Extra-Trees

Figure 12 : Performances des modèles ML par classe

L'analyse par classe révèle : des performances globalement élevées mais contrastées selon les modèles et selon les espèces. Le SVM-RBF maintient des F1 élevés et très homogènes sur l'ensemble des 14 espèces, y compris sur des classes minoritaires comme Raspberry, confirmant sa robustesse face au déséquilibre du jeu de données. XGBoost reste performant mais montre un rappel plus variable, avec des F1 plus faibles sur certaines espèces intermédiaires, alors qu'il reconnaît très bien les grandes classes telles que Tomato ou Corn. Il en va de même pour la régression logistique: elle distingue correctement les espèces les mieux séparables (Corn, Tomato), tout en peinant davantage sur des classes plus ambiguës comme Apple, Pepper_bell ou Strawberry. Extra-Trees est le plus affecté par le déséquilibre, avec des rappels particulièrement bas sur plusieurs classes peu représentées (Blueberry, Raspberry, Strawberry, Pepper_bell), ce qui se traduit par des F1 nettement inférieurs. La comparaison des moyennes macro et pondérées confirme ainsi que seul le SVM-RBF parvient à préserver un bon compromis entre toutes les espèces, tandis que les autres modèles restent fortement influencés par les classes majoritaires. On trouvera dans l'annexe 7.4 un tableau de comparaison des caractéristiques des modèles et en annexe 7.5 les matrices de confusion.

Discussion des Résultats

Identifier automatiquement les espèces de plantes est réalisable avec une accuracy supérieure à 93% et un F1 Macro 92% (SVM-RBF). La supériorité du SVM avec noyau RBF s'explique par sa capacité à capturer des relations non-linéaires complexes, essentielles pour distinguer les subtilités morphologiques entre espèces. L'écart de 4 points avec XGBoost montre que le SVM-RBF tire mieux parti des caractéristiques extraites que XGBoost..

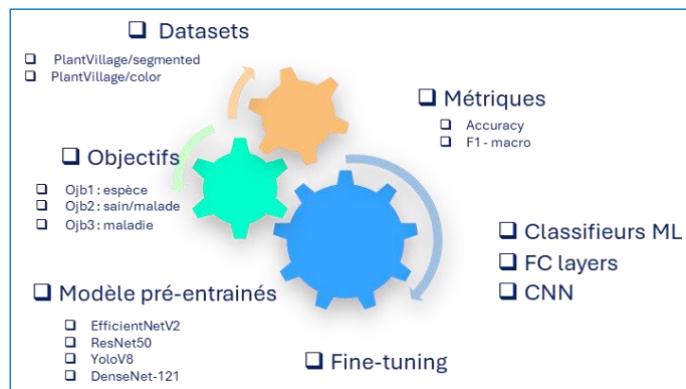
Les difficultés sur certaines classes (Pepper_bell, Potato, Strawberry) suggèrent des pistes d'amélioration : augmentation ciblée des données. Le déséquilibre des classes reste un défi malgré l'optimisation des hyperparamètres et aux poids de classes ajustés de SVM-RBF.

Bien que XGBoost et Extra-Trees pourraient bénéficier d'une optimisation plus poussée (exploration bayésienne, GridSearch exhaustif), le SVM-RBF constitue le choix optimal pour l'identification d'espèces végétales, offrant un excellent compromis entre performance globale et équité entre classes.

3.2. Deep Learning

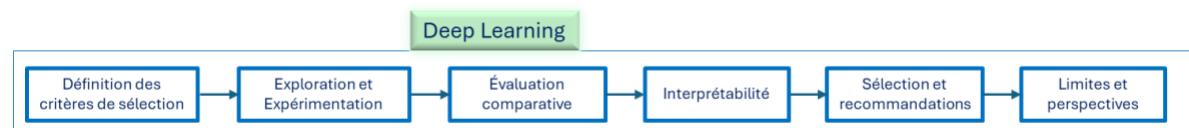
Dans le cadre de notre formation, nous avons procédé en deux étapes :

Etape1 : chaque membre de l'équipe a expérimenté les techniques de Deep Learning pour se confronter aux diverses problématiques en jouant sur les leviers suivants :



Etape 2 : Nous avons pratiqué une démarche structurée présentée dans ce qui suit :

3.2.1. Méthodologie



Afin de comprendre le fonctionnement du Deep Learning et ses défis, nous avons explorés 9 architectures. Pour effectuer une évaluation comparative, nous avons restreint le nombre d'architectures pour couvrir notre scénario avec 3 cas. Enfin, la sélection et recommandation est une projection « s'il fallait faire un déploiement ».

3.2.2. Définition des critères de sélection

Les architectures explorées retenues seront évaluées sur les critères suivants :

Classe de critère	Critère	Justification
Critères métier	Précision (Macro-F1)	Mesure la capacité du modèle à bien prédire toutes les classes, y compris les rares. Essentiel pour un diagnostic fiable.
	Précision (Accuracy)	Pourcentage de prédictions correctes. Indicateur simple mais peut être trompeur si déséquilibre de classes. À compléter par le Macro-F1.
	Généralisation (écart val/test)	Différence entre performances validation et test. Écart faible (<2%) = modèle robuste qui généralise bien. Écart élevé (>5%) = surapprentissage (overfitting).
	Couverture opérationnelle	Capacité à répondre aux 3 cas d'usage métier : (1) diagnostic maladie si plante connue, (2) identification espèce, (3) diagnostic complet. Détermine la polyvalence du modèle déployé.
Critères techniques	Coût d'inférence (FLOPs)	FLOPs (relatif) ≈ nombre de passes du backbone par image. Plus de FLOPs = plus de calculs = batterie consommée. Critique pour déploiement mobile/edge (smartphones agriculteurs, drones).
	Coût d'inférence (latence)	Temps réel pour obtenir une prédiction (en millisecondes). Impacte l'expérience utilisateur : <100 ms = fluide, >500 ms = frustrant.
	Coût d'entraînement (temps)	Important pour itérer rapidement lors des expérimentations et améliorer le modèle.
	Coût d'entraînement (GPU)	Ressources matérielles nécessaires (VRAM, type de GPU). Détermine le budget cloud ou la nécessité d'un GPU local.
	Complexité (paramètres)	Nombre de poids à stocker. Impact sur la taille du fichier modèle (déploiement) et le temps de chargement.
	Complexité (maintenabilité)	Facilité à comprendre, modifier et debugger le code. Important pour collaboration et évolution future.
Critères pédagogiques	Concepts explorés	Variété des techniques deep learning testées
	Apprentissages acquis	Compétences concrètes grâce au projet
Critères additionnels	Interprétabilité	Capacité à expliquer les prédictions du modèle (ex: Grad-CAM pour visualiser zones d'attention).
	Besoins en données	Quantité d'images annotées minimale pour entraîner efficacement.

Figure 13 : Critères de sélection des architectures

Pour rappel, le scénario se compose de 3 cas :

- Cas 1 — Identification d'espèce : Seule l'espèce doit être identifiée sans diagnostic de maladie (ex: Botaniste identifie juste l'espèce)
- Cas 2 — Diagnostic ciblé : L'espèce de la plante est connue, seule l'identification de la maladie est requise (ex: Agriculteur connaît sa plante, veut le diagnostic)
- Cas 3 — Diagnostic complet : L'espèce et la maladie sont inconnues, nécessitant une identification complète (ex: Application tout public sans connaissance préalable)

3.2.3. Exploration d'architectures DL

Nous avons choisi d'utiliser le transfert d'apprentissage car les modèles sont déjà entraînés sur des millions d'images pour détecter des motifs génériques (contours, textures, formes). C'est un gain de temps et de ressources. Le tableau suivant présente les caractéristiques des modèles pré-entraînés retenus pour nos explorations individuelles. Nous avons sélectionné **EfficientNetV2S** car il offre un excellent compromis entre performance et efficacité, avec une précision Top-1 de 83,9% sur ImageNet, surpassant ResNet50 et DenseNet-121 tout en utilisant moins de paramètres (21,5M contre 25,6M et 8M respectivement). Son efficacité computationnelle est remarquable, avec des GFLOPs optimisés (8,4 pour 224x224) et une latence GPU réduite (5-8 ms), ce qui le rend adapté à nos travaux qui nécessitent rapidité et compte tenu de nos ressources limitées. Malgré une taille de modèle plus légère (~86 MB), il

atteint une meilleure précision Top-5 (96,7%) que YOLOv8n-cls, idéal pour des tâches de classification exigeantes.

Caractéristique	EfficientNetV2-S	ResNet50	YOLOv8n-cls*	DenseNet-121
Année	2021	2015	2023	2017
Auteurs/Org	Google Brain	Microsoft Research	Ultralytics	Cornell/Facebook
Paramètres (M)	21.5	25.6	2.7	8.0
Taille modèle (MB)	~86	~102	~11	~32
GFLOPs (224×224)	8.4	4.1	4.2	2.9
GFLOPs (256×256)	~10.8	~5.3	~5.4	~3.7
Taille vecteur sortie	1280	2048	1024	1024
Top-1 Acc ImageNet	83.9%	76.1%	69.0%	74.4%
Top-5 Acc ImageNet	96.7%	93.0%	88.3%	92.0%
Latence CPU (ms)	60-80	40-50	25-35	30-40
Latence GPU (ms)	5-8	3-5	2-4	3-5
Taille entrée	384×384 (optim.)	224×224	224×224	224×224
Profondeur (layers)	~150	50	~100	121

Figure 14 : Comparatif des caractéristiques des modèles pré-entraînés choisis

Protocole expérimental

- Le même dataset PlantVillage/color
- Splits identiques pour tous les modèles (fichier pv_color_splits.json)
- Hyperparamètres fixés : learning rate, batch size, augmentation
- Le même Backbone pré-entraîné EfficientNetV2S (ImageNet)
- Métriques trackées : Loss, Accuracy, Macro-F1, matrice de confusion
- Seuil de Sur-apprentissage : équilibré $|Accuracy - Macro-F1| \leq 0,002$. Entre 0,002 et 0,005: acceptable mais à surveiller. $>0,005$: déséquilibre notable à investiguer.

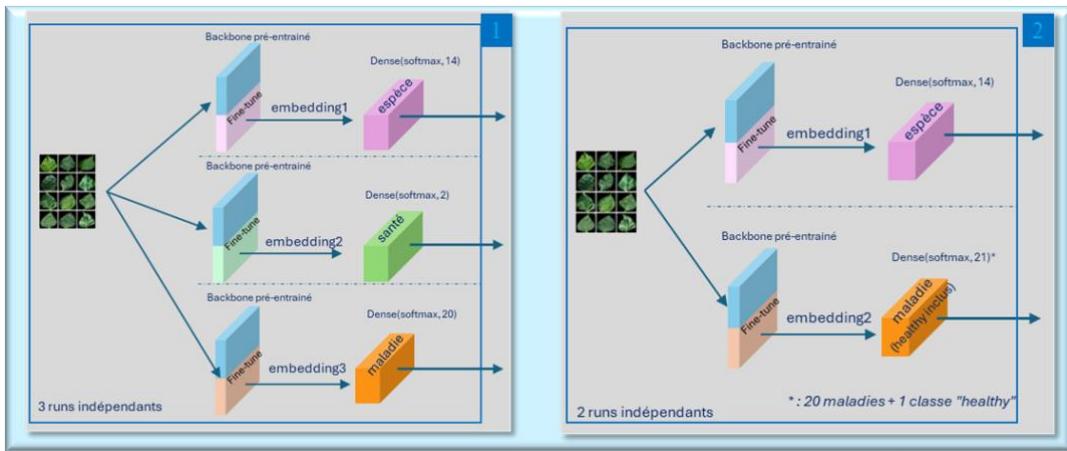
Remarque : nous avons créé une variante data augmented du dataset PlantVillage/color, qui a été testée uniquement sur les deux architectures sélectionnées.

Entrainements réalisés

- 1 run par architecture (préparation, stage 1, stage 2, évaluation finale)
- Logs sont sauvegardés (courbes, history.csv)
- Checkpoint du meilleur modèle : (val_macro_f1)

Les architectures explorées sont issues de nos nombreuses discussions et de nos lectures. Pour les présenter, elles sont réparties en 2 groupes : les architectures dont le backbone pré-entraîné est dédié à chaque objectif (tête), et les architectures dont le backbone pré-entraîné est partagé entre plusieurs objectifs. Pour éviter les répétitions : les classes considérées sont : 14 espèces (species), 2 états de santé(healthy), 20 maladies(disease). Parcours du dataset, extraction des features par le backbone, constitution des splits train/val/test. Les courbes des métriques trackées sont en annexe [7.6](#).

Backbone pré-entraîné dédié à chaque objectif :



Architecture 1 :

Architecture spécialisée : Trois modèles CNN indépendants, chacun dédié à une seule tâche (species, health, disease). Chaque modèle comprend un backbone pré-entraîné et une tête de classification Dense adaptée au nombre de classes.

Workflow : Chaque modèle s'entraîne en 2 phases sur le même dataset d'entrée : Phase 1 : backbone gelé avec entraînement de la tête uniquement; Phase 2 : fine-tuning des dernières couches du backbone pour adapter les features ImageNet aux spécificités du dataset.

Avantages : Simplicité (1 tâche = 1 modèle), absence de conflits entre tâches (pas de compromis dans l'optimisation), performances maximales par tâche (spécialisation totale), interprétabilité facilitée (1 objectif clair par modèle).

Limites : Triplication des ressources (3 backbones à stocker et maintenir), inférences multiples pour cas d'usage complexes, absence de synergie inter-tâches (pas de transfert d'apprentissage entre les 3 têtes), temps d'entraînement cumulé plus long (3 runs).

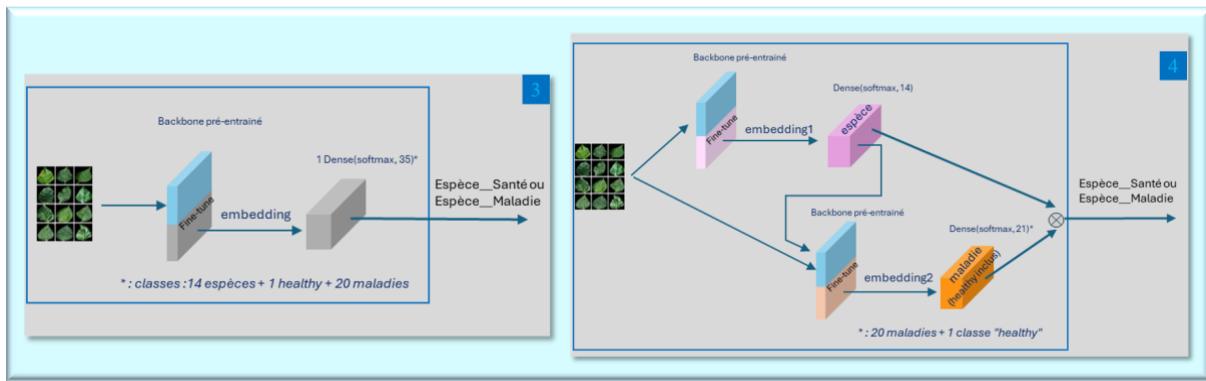
Architecture 2 :

Architecture : Deux modèles CNN indépendants, l'un pour l'espèce, l'autre pour l'état sanitaire complet : la classe "healthy" est intégrée comme une maladie spéciale.

Workflow : Deux runs mono-tâche. Le modèle species s'entraîne sur toutes les images (saines + malades). Le modèle disease_extended s'entraîne également sur toutes les images.

Avantages : Simplicité (2 têtes), uniformité (deux softmax multi-classe), diagnostic complet en 2 inférences (species + disease_extended), "healthy" est un état sanitaire comme les maladies.

Limites : Déséquilibre accru (classe "healthy" majoritaire), perte de la métrique binaire explicite healthy/diseased, interprétation plus ambiguë des prédictions mixtes (ex: 40% healthy, 35% early_blight).



Architecture 3 :

Architecture unifiée : Un modèle CNN pré-entraîné + 1 tête Dense softmax (35 classes). Étiquette combinée : chaque image est étiquetée par un couple “Espèce_Etat” (Tomato_healthy, Apple_scab...).

Workflow : Phase 1: backbone gelé, entraînement de la tête uniquement. Phase 2: fine-tuning partiel des dernières couches du backbone. Les labels sont pré-combinés en 35 classes.

Avantages : Un seul modèle, une seule inférence : plus simple à déployer et à utiliser. Synergie entre tâches : l'apprentissage capte directement les co-dépendances espèce↔maladie/santé.

Limites : Moins de spécialisation par tâche. Les classes rares peuvent être sous-apprises. Peu flexible : impossible de gérer des paires inédites (nouvelle espèce/maladie) sans réentraîner les 35 classes. Interprétabilité : plus dur d'isoler l'erreur (vient-elle de l'identification d'espèce ou de maladie ?).

Architecture 4 :

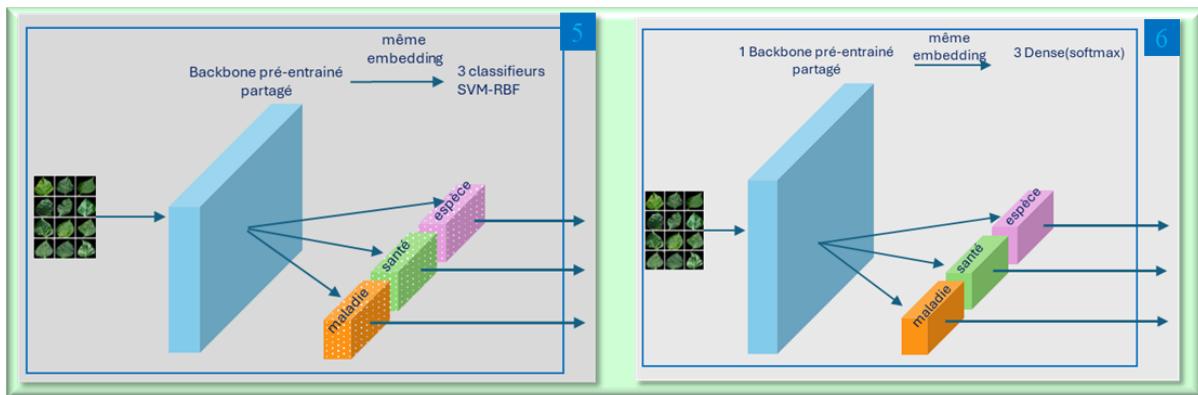
Architecture en cascade : Deux modèles CNN pré-entraînés sont chaînés. Un classificateur d'espèce extrait un embedding d'image et prédit l'espèce. Un classificateur de maladie global (21 classes, dont “healthy”) reçoit l'image + l'espèce True (étiquette d'espèce vraie (ground truth), pas l'espèce prédite) et applique une attention spatiale pour se focaliser sur les zones pertinentes, puis prédit la maladie. À l'inférence, la maladie est conditionnée par l'espèce prédite. La sortie finale se lit comme une étiquette composée “Espèce_Santé/Maladie”.

Workflow : Phase 1 : backbone gelé, entraînement de la tête. Phase 2 : fine-tuning partiel du backbone. Entraînement du modèle maladie en 2 phases, avec la même logique (tête puis fine-tuning), mais en lui fournissant l'espèce (True) en entrée pour stabiliser l'apprentissage. Évaluation en CASCADE - maladie avec espèce prédite (performance réelle de bout en bout).

Avantages : La prédiction d'espèce guide la maladie, réduisant les confusions entre espèces. L'attention spatiale sur la branche maladie aide à capturer les indices visuels pertinents. Modularité : possibilité d'améliorer séparément espèce ou maladie sans tout réentraîner.

Limites : Une espèce mal prédite dégrade la maladie. Le modèle maladie voit l'espèce (True) à l'entraînement mais la prédite en production. Latence accrue avec passes réseau successives. En cas d'espèce erronée, une maladie impossible peut être proposée.

Backbone pré-entraîné partagé entre plusieurs objectifs :



Architecture 5

Architecture “CNN + SVM” : Un backbone CNN pré-entraîné (gelé) transforme chaque image en vecteur d’embeddings (features). Des classificateurs SVM (espèce, santé, maladie) sont entraînés sur ces embeddings.

Workflow : Sauvegarde des vecteurs + labels. Puis chargement des embeddings, entraînement de trois têtes SVM: Espèce: multi-classe, Santé: binaire (healthy vs diseased), Maladie : soit global (multi-classe, malades uniquement), soit par espèce (un SVM par espèce, malades uniquement).

Avantages : Entraînement très rapide des SVM; itérations légères (on réutilise les embeddings). Simplicité opérationnelle : séparation claire “features gelées” / “classificateurs”; facile de remplacer le backbone ou de réentraîner seulement les SVM.

Limites : Les features restent génériques : pas d’adaptation conjointe aux tâches du dataset. Cohérence multi-tâches limitée.

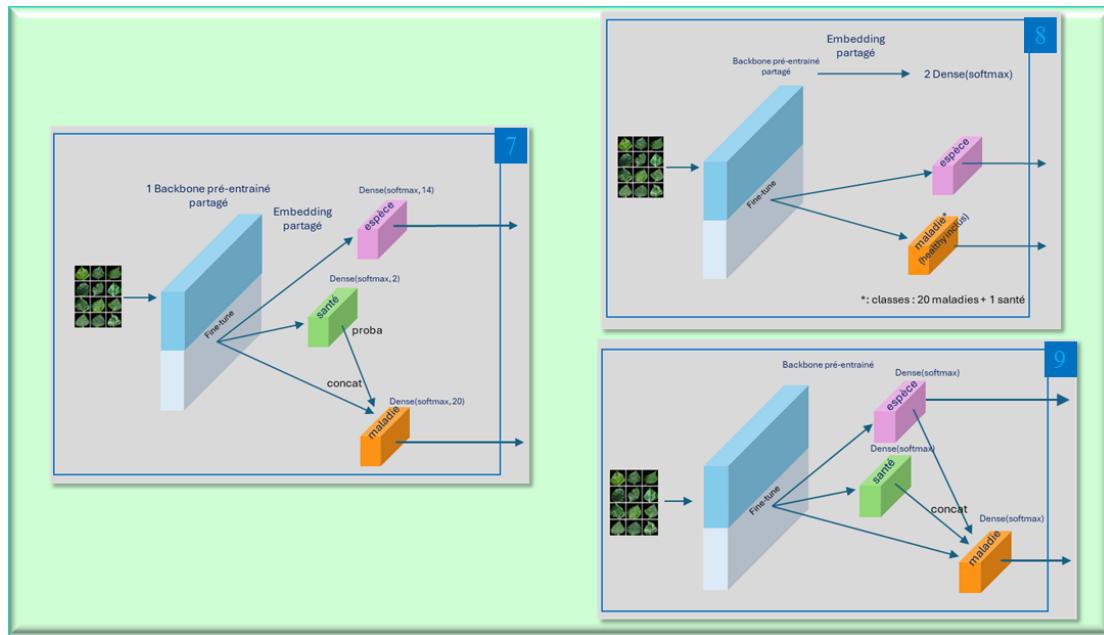
Architecture 6

Architecture multi-tâche unifiée : Un seul backbone CNN pré-entraîné, partagé, produit un embedding commun, puis trois têtes de classification parallèles: Espèce, Santé, Maladie. La tête “maladie” est optimisée sur les images malades, afin d’éviter de perturber l’apprentissage par des exemples “healthy”.

Workflow : Une seule phase “têtes seules” avec backbone gelé (pertes pondérées par tête).

Avantages : Les trois tâches se renforcent (l’espèce et la santé aident la maladie). Un seul backbone à entraîner ; une seule inférence pour obtenir espèce, santé, maladie. Contrôle des compromis : pondérations de pertes par tête pour équilibrer objectifs (espèce/santé/maladie).

Limites : Conflits d’optimisation : objectifs parfois concurrents ; sensibilité aux pondérations des pertes. Malgré la tête dédiée, les maladies peu représentées restent difficiles. On n’active pas le fine-tuning, les features ImageNet peuvent rester trop génériques, la Loss du cas fine-tuning était catastrophique. Couplage des tâches : une mauvaise modélisation de l’espèce/santé peut impacter indirectement la maladie (via le partage d’embedding).



Architecture 7

Architecture multi-tâche à 2 têtes: Un backbone CNN pré-entraîné partagé produit un embedding commun. Tête espèce: multi-classe. Tête maladie: multi-classe hors “healthy”, activée uniquement pour les échantillons malades. Santé (healthy/diseased) n'est pas une sortie directe: un signal santé auxiliaire interne (probabilité “malade”) est appris et injecté comme feature dans la tête maladie pour l'aider à se focaliser sur les cas réellement malades.

Workflow : Phase 1: entraînement des têtes avec backbone gelé (pondérations de pertes, l'échantillon tagué “healthy” n'entraîne pas la tête maladie). Phase 2: fine-tuning partiel des couches hautes du backbone pour adapter les features au domaine.

Avantages : Une seule passe backbone pour deux tâches; coût d'inférence réduit. L'injection de la probabilité “malade” et le masquage de perte évitent que les “healthy” perturbent la tête maladie. Synergie utile: l'embedding partagé bénéficie des signaux espèce et santé auxiliaire, améliorant la discrimination des maladies. Equilibre des objectifs via pondérations des pertes;

Limites : Pas de sortie santé explicite: pas de score/label “healthy vs diseased” livrable tel quel (signal interne non calibré pour un usage direct). Dépendance au signal santé: si le signal auxiliaire est biaisé, la tête maladie peut sur- ou sous-activer certaines classes. Conflits d'optimisation: partage d'un même backbone pour deux objectifs; sensibilité aux pondérations et au fine-tuning. Classes rares: malgré le masquage des “healthy”, les maladies peu représentées restent difficiles.

Architecture 8

Architecture multi-tâche simplifiée (2 têtes) : Un seul backbone CNN pré-entraîné, partagé, et deux têtes parallèles: Espèce, Disease qui inclut explicitement healthy. Pas de tête “santé” dédiée, pas de masquage d'échantillons: toutes les images (saines et malades) entraînent les deux têtes.

Workflow : Préparation: splits stratifiés par dossier de classe, pipeline tf.data avec augmentations légères et prétraitement compatible EfficientNetV2S. Entraînement: Phase 1:

entraînement des têtes avec backbone gelé (pondérations de pertes, label smoothing). Phase 2: fine-tuning partiel du haut du backbone (option gradient clipping) pour adapter les features au domaine. Évaluation: accuracy et macro-F1 pour l'espèce et pour disease_all (les 21 classes), matrices de confusion, courbes, rapport Markdown. Inférence: une seule passe réseau → deux sorties simultanées: Espèce et Healthy/Maladie_k.

Avantages : Simplicité: pas de tête santé, pas de règles/mask; supervision uniforme sur toutes les images. Efficience: un seul backbone et une seule inférence pour obtenir espèce + santé/maladie. Cohérence de décision: healthy fait partie du même espace que les maladies → seuils et calibration unifiés au sein d'une softmax à 21 classes. Maintenance légère: pipeline standardisé (figures, rapports, checkpoints) et fine-tuning optionnel.

Limites : Déséquilibre "healthy": la classe healthy peut dominer et biaiser la tête disease_all, au détriment des maladies rares (macro-F1 crucial). Pas de conditionnement par espèce: la tête maladie n'est pas contrainte par l'espèce → risque de confusions inter-espèces. Seuils globaux: frontières de décision communes à toutes les espèces; calibration potentiellement sous-optimale pour des distributions très différentes selon l'espèce. Raccourci possible: le modèle peut exploiter des corrélations de fond (espèce, contexte) plutôt que des lésions fines si les données sont biaisées.

Architecture 9

Architecture conditionnée (Species + Health → Disease): Un backbone CNN pré-entraîné unique produit un embedding partagé. Tête espèce: multi-classe. Tête maladie: multi-classe (hors "healthy"), conditionnée par deux signaux additionnels: le vecteur de probabilités d'espèce. La probabilité interne d'être malade (tête santé auxiliaire non exposée). Les échantillons "healthy" n'entraînent pas la tête maladie (masque de perte).

Workflow : Phase 1: apprentissage des têtes avec backbone gelé, pondérations de pertes. Phase 2: fine-tuning partiel des couches hautes. La tête maladie est optimisée uniquement sur les images malades (healthy masqués).

Avantages : Conditionnement explicite: la maladie est guidée par l'info d'espèce et un indicateur de santé, ce qui réduit les confusions inter-espèces et focalise sur les cas réellement malades. Synergie multi-tâches: l'embedding partagé + signaux auxiliaires apportent un contexte fort au classifieur maladie. Efficience: un seul backbone à entraîner/déployer; une seule inférence pour obtenir espèce et maladie. Contrôle des compromis: pondérations de pertes par tête; fine-tuning optionnel selon budget/performance.

Limites : Propagation d'erreurs: une erreur d'espèce ou un biais du signal santé peut entraîner une mauvaise prédiction de maladie. Raccourcis/biais: le modèle peut sur-utiliser les a priori espèce/santé au détriment d'indices visuels fins si les données sont déséquilibrées. Pas de sortie santé livrable: la santé est un signal interne; si un score "healthy vs diseased" est requis, il faut une tête/mesure dédiée. Calibration sur "healthy": la tête maladie n'est pas entraînée sur les sains; ses sorties peuvent être peu informatives pour des images réellement "healthy" si utilisées seules.

Synthèse des performances des 9 architectures

Le surapprentissage a été vérifié sur toutes les architectures (courbes de Loss et courbes d'accuracy et de Macro-F1) . Le tableau suivant présente les performances obtenues :

Arch	Nom	Espèce		Maladie	
		Macro-F1	Accuracy	Macro-F1	Accuracy
1	Mono-tâche - 3 modèles indépendants	0.9987	0.9990	0.9901	0.9934
2	Mono-tâche - 2 modèles indépendants	0.9990	0.9990	0.9922	0.9945
3	Mono-tâche - 1 tête (35 classes)	0.9987	0.9988	0.9921	0.9954
4	Cascade - 2 modèles en série	0.9988	0.9989	0.9911	0.9953
5	CNN Embeddings + SVM	0.9909	0.9928	0.9545	0.9657
6	Multi-tâche - 3 têtes de sortie	0.9988	0.999	0.9889	0.9939
7	Multi-tâche - 2 têtes de sortie	0.9985	0.9988	0.9904	0.9941
8	Multi-tâche - 2 têtes +concat.santé	0.9986	0.9988	0.9908	0.9939
9	Multi-tâche - 2 têtes + concat. espèce + santé	0.9986	0.9988	0.9913	0.9952

Figure 15 : Synthèse des performances des 9 architectures

Les modèles atteignent des performances avec des Macro-F1 et Accuracy souvent supérieurs à 0.99, ce qui montre une grande précision dans la classification. Les architectures mono-tâche, notamment celles utilisant plusieurs modèles indépendants, démontrent une robustesse remarquable, idéale pour des tâches spécifiques et ciblées. Les approches multi-tâches, légèrement moins performantes sur certaines métriques, offrent une solution intégrée efficace pour gérer plusieurs objectifs simultanément. Le modèle "CNN Embeddings + SVM" présente des résultats inférieurs, soulignant les limites des méthodes hybrides face aux architectures deep learning pures.

Synthèse des coûts des 9 architectures

Archi	Description	FLOPs (relatif)	Latence (indicative)	Entrainement: temps	Entrainement: GPU	Paramètres (M)	Maintenabilité
1	Mono-tâche - 3 modèles indépendants	1× (Cas1/2) • 3× (Cas3)	faible (Cas1/2) • élevé (Cas3, 3 passes)	élevé (3 modèles)	modéré	136 (*3 en Cas3)	faible (3 pipelines)
2	Mono-tâche - 2 modèles indépendants	1×	faible	moyen	modéré	137	bonne
3	Mono-tâche - 1 tête (35 classes)	1×	faible	moyen	modéré	136	très bonne (pipeline simple)
4	Cascade - 2 modèles en série	2*	élevée (2 inférences)	élevé (2 modèles)	modéré à élevé	2 backbones (B0+B2)	moyenne (2 étages)
5	CNN Embeddings + SVM	1*	faible	faible (SVM backbone gelé)	modéré	136 (+SVM négl.)	moyenne (mix DL/ML)
6	Multi-tâche - 3 têtes de sortie	1*	faible	faible à moyen (pas de FT)	modéré à faible	137	bonne
7	Multi-tâche - 2 têtes de sortie	1*	faible	moyen	modéré	137	bonne
8	Multi-tâche - 2 têtes +concat.santé	1*	faible	moyen	modéré	137	bonne
9	Multi-tâche - 2 têtes + concat. espèce + santé	1*	faible	moyen	modéré	137	moyenne (logique hiérarchique)

Les architectures multi-tâche (archi 6 -> 9) se distinguent par leur équilibre optimal : légèreté (FLOPs et paramètres modérés), latence faible, et maintenabilité élevée, idéales pour un déploiement efficace. Les modèles mono-tâche (archi 2 et 3) offrent une simplicité et rapidité d'exécution, mais leur maintenabilité est limitée par la gestion de pipelines multiples. La cascade (archi 4) et CNN Embeddings + SVM (archi 5) sont plus coûteuses en ressources et en latency, avec une complexité accrue (2 backbones, SVM), ce qui les rend moins adaptées aux environnements contraints. Les architectures multi-tâche (archi 7,8,9) sont particulièrement avantageuses pour leur facilité d'entraînement et leur logique unifiée, réduisant les efforts de maintenance.

Décision

Nous excluons **l'architecture 4** car sa cascade en deux passes augmente nettement la latency et la complexité sans gain tangible face aux architectures 3/7/9 déjà proches de 0.99, avec en plus un risque de propagation d'erreurs. Nous écartons **l'architecture 6** car elle est nettement en retrait sur la maladie, ce qui dégrade la fiabilité opérationnelle face à des alternatives qui atteignent ≈ 0.99 et ≥ 0.989 de Macro-F1. Enfin, **l'architecture 8** n'apporte pas de bénéfice mesurable par rapport aux 7/9 et reste plus basse sur la maladie (≈ 0.986 de Macro-F1 vs ≥ 0.989), ne justifiant pas sa complexité supplémentaire.

3.2.4. Évaluation comparative

L'évaluation est présentée en fonction des 3 cas de notre scénario global.

Cas 1 — Identification d'espèce : Seule l'espèce doit être identifiée sans diagnostic de maladie (ex: Botaniste identifie juste l'espèce)

Pour les 6 architectures restantes, comparons Accuracy et Macro F1-Score ainsi que la corrélation Accuracy vs F1 pour valider la cohérence :

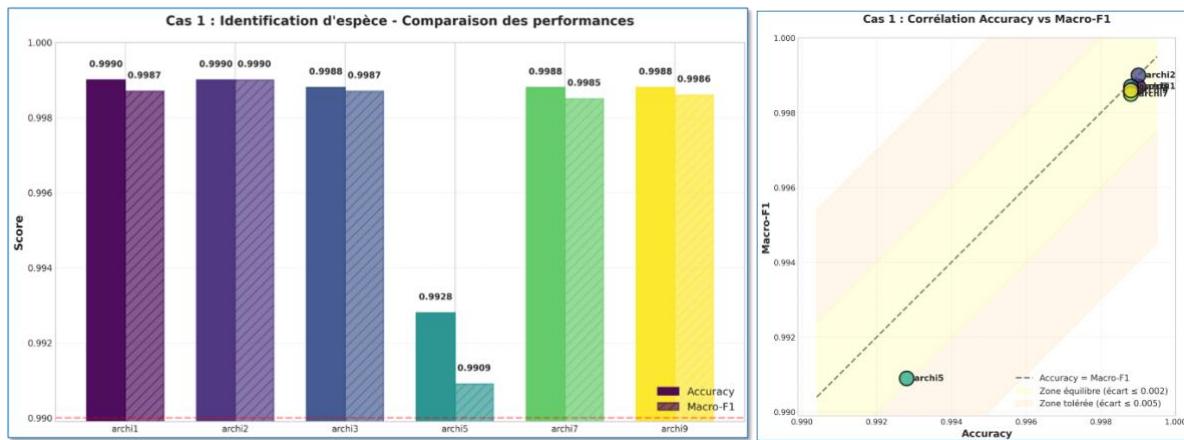


Figure 16 : comparaison et corrélation Accuracy et Macro F1-Score par architecture - cas 1

Toutes les architectures atteignent une précision quasi parfaite pour l'identification d'espèce, avec des écarts globaux très faibles entre elles. Les écarts Accuracy – Macro-F1 restent très limités pour archi1/2/3/7/9 ($\leq 0,0003$), ce qui traduit une performance très homogène entre espèces et un faible risque de biais de classe. Archi5 présente un écart un peu plus marqué (~0,002, avec 0,9928 d'Accuracy pour 0,9909 de Macro-F1), ce qui suggère que quelques espèces sont légèrement moins bien apprises, même si le niveau global de performance reste très élevé.

Voici le classement final des 6 architectures pour l'identification de l'espèce, basé sur le Macro F1 plus robuste que l'Accuracy.

Rang	Archi	Nom	Accuracy	Macro-F1
#1	archi2	Mono-tâche - 2 modèles indépendants	0.9990	0.9990
#2	archi1	Mono-tâche - 3 modèles indépendants	0.9990	0.9987
#3	archi3	Mono-tâche - 1 tête (35 classes)	0.9988	0.9987
#4	archi9	Multi-tâche - 2 têtes + concat. espèce + santé	0.9988	0.9986
#5	archi7	Multi-tâche - 2 têtes de sortie	0.9988	0.9985
#6	archi5	CNN Embeddings + SVM	0.9928	0.9909

Figure 17 : classement final des 6 architectures pour l'identification de l'espèce

Cas 2 — Diagnostique ciblé : L'espèce de la plante est connue, seule l'identification de la maladie est requise (ex: Agriculteur connaît sa plante, veut le diagnostic)

Pour les 6 architectures restantes, comparons Accuracy et Macro F1-Score ainsi que la corrélation Accuracy vs F1 pour valider la cohérence :

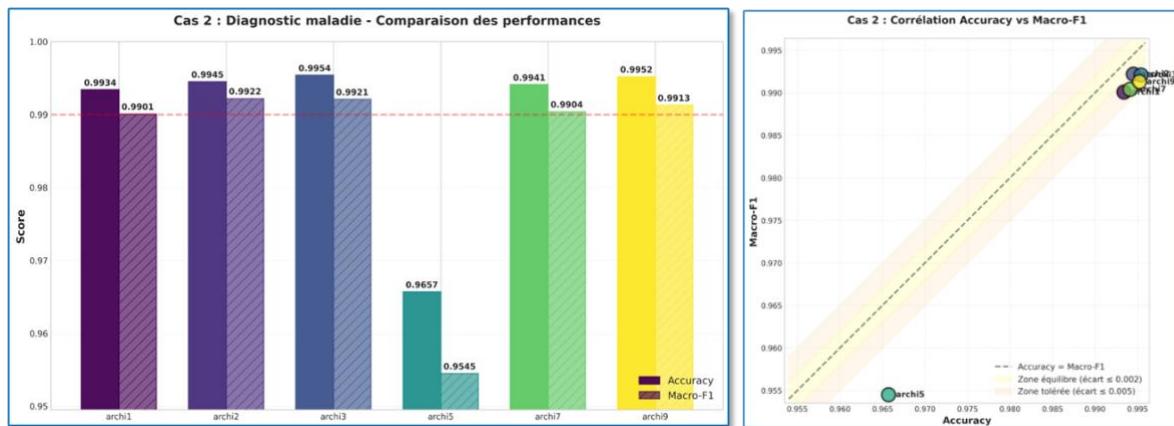


Figure 18 : comparaison et corrélation Accuracy et Macro F1-Score par architecture - cas 2

Les architectures archi3 et archi2 restent en tête : archi3 affiche désormais la meilleure Accuracy ($\approx 0,9954$) et archi2 le meilleur Macro-F1 ($\approx 0,9922$), avec archi9 pratiquement au même niveau sur les deux métriques. L'archi5 demeure nettement en retrait, avec un Macro-F1 bien inférieur aux autres modèles et une Accuracy qui passe sous le seuil de 0,97.

Les architectures archi1/2/3/7/9 restent proches de la diagonale et se situent dans la zone tolérée (écart Accuracy – Macro-F1 d'environ 0,002 à 0,004), ce qui traduit un comportement globalement équilibré même si l'Accuracy reste systématiquement légèrement supérieure au Macro-F1. En revanche, archi5 se retrouve nettement en dehors de la zone tolérée (écart $\approx 0,011$), signe que plusieurs maladies sont sensiblement moins bien reconnues. Cela renforce le risque de faux négatifs ou faux positifs, en particulier sur les maladies rares, pour ce modèle.

Voici le classement final des 6 architectures pour l'identification de l'espèce. Le classement est basé sur le Macro F1-Score, qui est plus robuste que l'Accuracy.

Rang	Archi	Nom	Accuracy	Macro-F1
#1	archi2	Mono-tâche - 2 modèles indépendants	0.9945	0.9922
#2	archi3	Mono-tâche - 1 tête (35 classes)	0.9954	0.9921
#3	archi9	Multi-tâche - 2 têtes + concat. espèce + santé	0.9952	0.9913
#4	archi7	Multi-tâche - 2 têtes de sortie	0.9941	0.9904
#5	archi1	Mono-tâche - 3 modèles indépendants	0.9934	0.9901
#6	archi5	CNN Embeddings + SVM	0.9657	0.9545

Figure 19 : classement final des 6 architectures pour l'identification de la maladie

Cas 3 — Diagnostic complet : L'espèce et la maladie sont inconnues, nécessitant une identification complète (ex : Application grand public sans connaissance préalable)

Pour les 6 architectures restantes, comparons Accuracy et Macro F1-Score ainsi que la corrélation Accuracy vs F1 pour valider la cohérence :

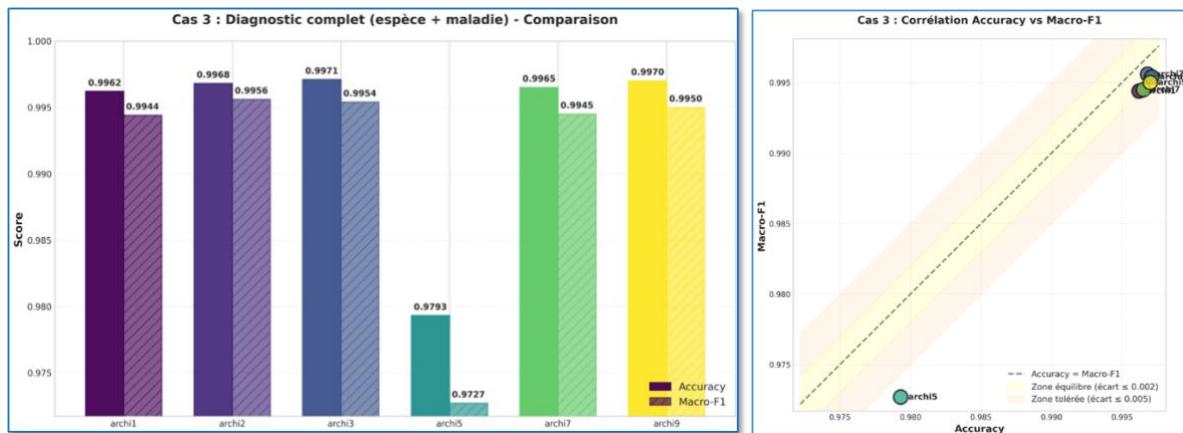


Figure 20 : comparaison et corrélation Accuracy et Macro F1-Score par architecture - cas 3

Pour un diagnostic bout-à-bout, archi2 apparaît comme le meilleur compromis (Accuracy $\approx 0,9968$, Macro-F1 $\approx 0,9956$), avec archi3 et archi9 quasiment au même niveau de performance. archi7 puis archi1 restent très proches. archi5 demeure nettement en retrait (Accuracy $< 0,98$ et Macro-F1 $\approx 0,973$) et reste à éviter pour un usage critique.

Les architectures archi1/2/3/7/9 se situent tous dans la bande d'équilibre (écart Accuracy – Macro-F1 $\leq 0,002$), avec des différences très limitées entre les deux métriques ($\approx 0,0012$ – $0,0020$). Cela indique un comportement globalement homogène entre classes pour ces modèles, avec peu de risque de biais marqué sur certaines maladies. Archi5 reste nettement en dehors de la zone tolérée (écart $\approx 0,0066 > 0,005$), ce qui traduit une hétérogénéité plus forte entre classes et un risque accru de faux négatifs/faux positifs sur certaines combinaisons espèce+maladie. Archi2, archi3, archi7 et archi9 sont de bons candidats “prêts à déployer” pour un flux bout-à-bout avec moins de surprises par classe, tandis que archi5 n'est pas recommandé sans amélioration ou recalibrage substantiel.

Voici le classement final des 6 architectures pour l'identification de l'espèce. Le classement est basé sur le Macro F1-Score, qui est plus robuste que l'Accuracy.

Rang	Archi	Nom	Accuracy	Macro-F1
#1	archi2	Mono-tâche - 2 modèles indépendants	0.9968	0.9956
#2	archi3	Mono-tâche - 1 tête (35 classes)	0.9971	0.9954
#3	archi9	Multi-tâche - 2 têtes + concat. espèce + santé	0.9970	0.9950
#4	archi7	Multi-tâche - 2 têtes de sortie	0.9965	0.9945
#5	archi1	Mono-tâche - 3 modèles indépendants	0.9962	0.9944
#6	archi5	CNN Embeddings + SVM	0.9793	0.9727

Figure 21 : classement final des 6 architectures pour l'identification complète

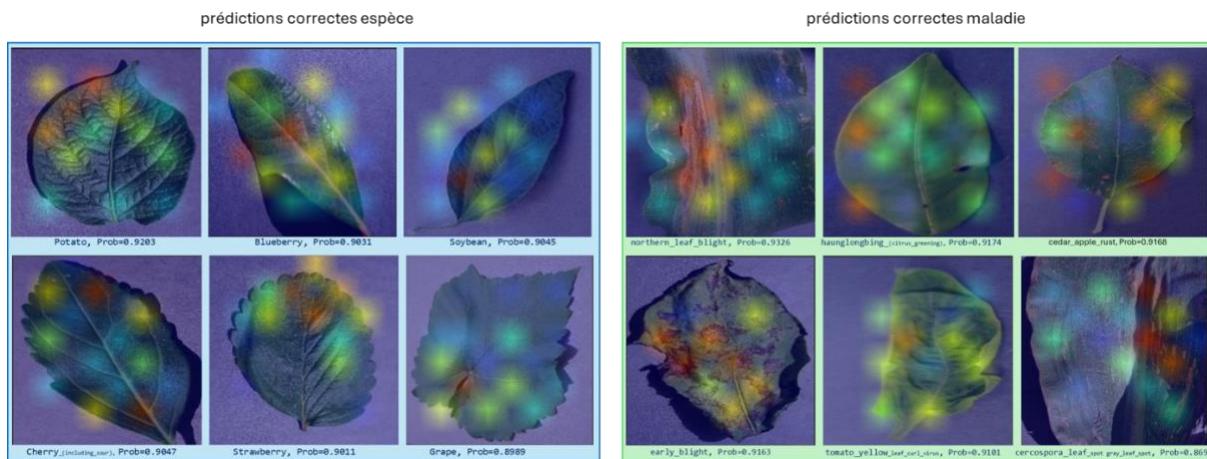
3.2.5. Interprétabilité

Nous décidons d'évaluer l'interprétabilité du modèle archi9 qui pourrait être déployé en production standard. Pour rappel, dans l'archi9, le backbone extrait des features communes, à partir desquels une tête principale prédit l'espèce, une tête auxiliaire estime la probabilité d'être malade, et une tête maladie combine ces features avec les probabilités d'espèce et de santé pour prédire la maladie. Nous utilisons Grad-CAM, qui produit des cartes de chaleur superposées aux images d'entrée et met en évidence les régions ayant le plus contribué à la prédiction. Cet outil nous permet les vérifications suivantes.

3.2.5.1. Vérifier la pertinence des prédictions du modèle en phase d'inférence

Grad-CAM sur prédictions correctes (species et disease) :

Lorsque le modèle prédit correctement l'espèce et la maladie, les cartes Grad-CAM montrent-elles que son attention se concentre bien sur la feuille et les lésions pertinentes ?

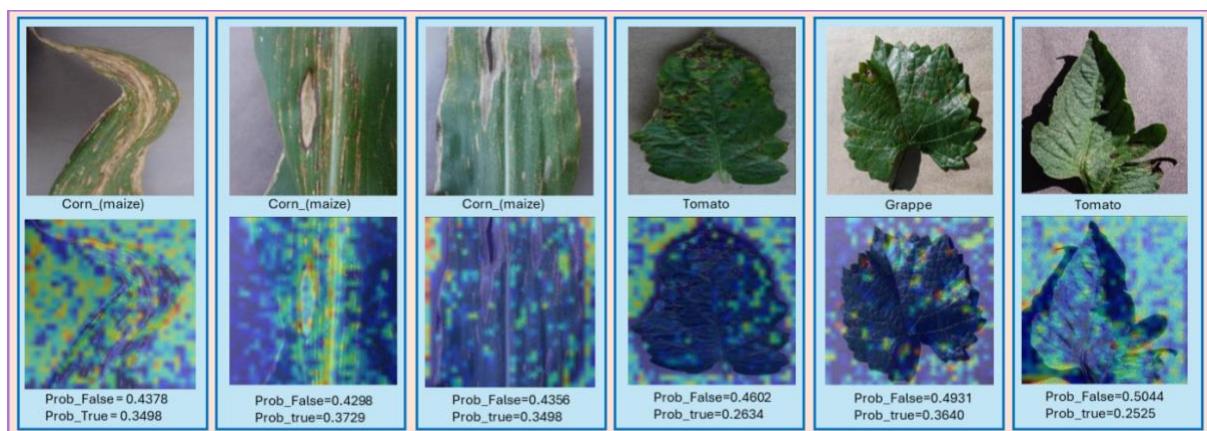


Pour les espèces, les cartes Grad-CAM montrent des foyers d'attention surtout répartis sur la feuille et les nervures, avec très peu d'activation sur le fond ou les supports, ce qui suggère que la tête espèce exploite bien la forme et la texture de la feuille. Les probabilités associées à ces prédictions ($\approx 0,90\text{--}0,92$) sont élevées et cohérentes avec ces cartes, indiquant une confiance du modèle supportée par des indices visuels.

Pour les maladies, les zones chaudes se concentrent majoritairement sur les régions de lésions, décolorations ou bords dégradés, plutôt que sur les zones saines ou le fond violet, ce qui est conforme à l'expertise visuelle attendue. Globalement, ces exemples corrects confirment que, dans des cas typiques, le modèle Archi9 base ses décisions sur des régions anatomiquement et pathologiquement pertinentes, plutôt que sur des artefacts de studio.

GradCAM sur erreurs de classification (disease, images malades) :

Les erreurs viennent-elles du fait que le modèle regarde ailleurs que les lésions, ou bien qu'il regarde les lésions mais se trompe de classe ?

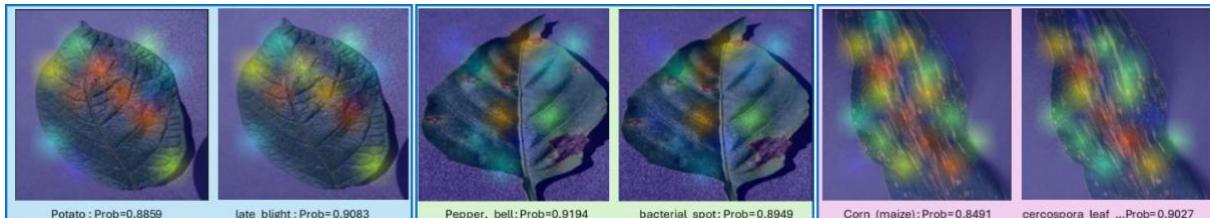


Les cartes Grad-CAM montrent que le modèle concentre globalement son attention sur la région de la feuille plutôt que sur le fond. Les activations restent cependant diffuses sur de larges zones de la feuille (parfois au-delà des lésions visibles), ce qui suggère que le modèle exploite des motifs globaux de texture/couleur plutôt qu'une localisation très précise des taches. Pour chaque image, la probabilité de la classe prédictive (fausse) est modérée ($\approx 0,43 - 0,50$) et la probabilité de la vraie classe reste non négligeable ($\approx 0,25 - 0,37$), ce qui traduit une forte incertitude et une compétition entre plusieurs maladies candidates. Globalement, l'expérience CAM + probabilités indique que le modèle regarde bien les feuilles mais manque de sélectivité fine sur les lésions, ce qui conduit à des frontières de décision floues entre certaines classes de maladies.

3.2.5.2. Comparer l'attention du réseau entre les tâches de classification

Lorsque l'on passe d'une tâche de classification à l'autre (espèce, maladie), les cartes Grad-CAM montrent-elles un déplacement significatif de l'attention du réseau vers des régions différentes de la feuille ?

Le schéma suivant présente pour 3 images les heatmap par paires tête espèce et tête maladie.



La comparaison des cartes Grad-CAM issues des têtes species et disease montre des zones d'attention très similaires, centrées sur les mêmes régions de la feuille. Elles mettent en évidence des caractéristiques liées à l'espèce (ex. forme des feuilles, nervures) plutôt que les symptômes de la maladie (ex. taches, nécroses). Ce résultat s'explique d'une part par l'architecture hiérarchique d'Archi9 et d'autre part par la forte corrélation espèce–maladie propre au dataset PlantVillage. En effet, les deux tâches apprennent à exploiter des signatures visuelles communes, ce qui complique la distinction entre « information d'espèce » et « information de maladie » au niveau des cartes d'attention. Ce résultat confirme le biais de corrélation espèce–maladie propre au dataset PlantVillage identifié dans l'analyse exploratoire de PlantVillage.

Qu'en est-il pour les autres architectures ?

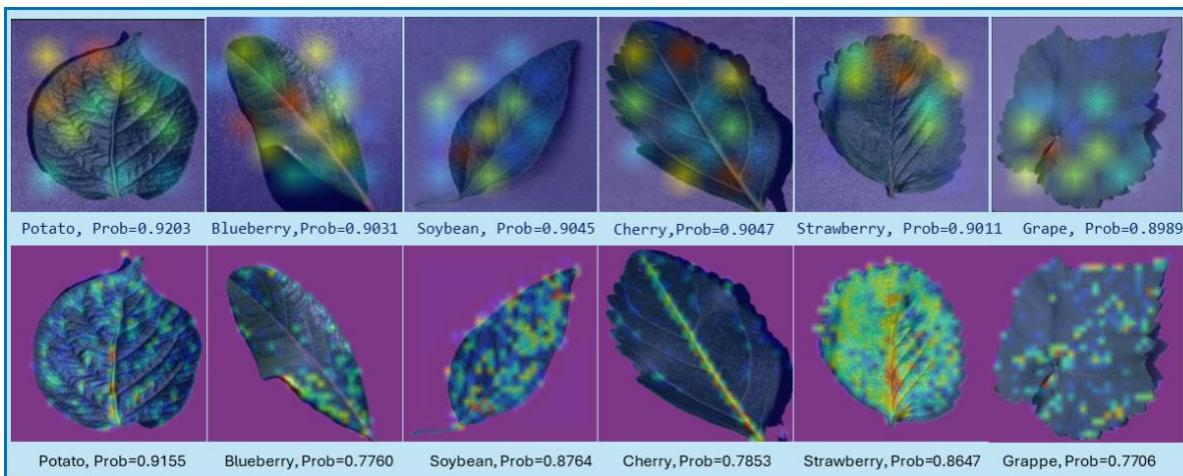
- Il est maximal pour les modèles qui codent directement les couples espèce_maladie ou conditionnent fortement (3, 4, 7, 9).
- Il est modéré mais présent pour les architectures multi-tâches partagées (6, 8).
- Il est un peu moins structurel mais toujours réel pour les modèles séparés (1, 2, 5).

3.2.5.3. Analyser l'influence d'une couleur de fond uni différente

Les Grad-CAM montrent-elles que le modèle s'appuie sur des indices de fond ou de prise de vue caractéristiques du dataset PlantVillage plutôt que sur des motifs pathologiques réellement liés à la maladie ?

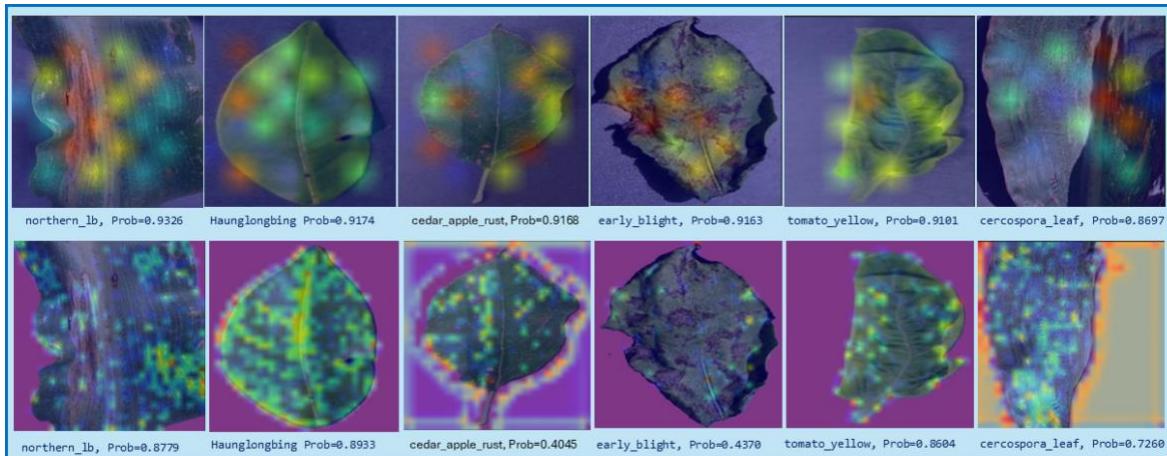
Nous avons changé et le fond beige des images § 3.2.5.1 prédictions correctes espèce et maladie en saumon. Comparons les Grad-CAM fond beige et fond saumon.

Prédictions correctes espèce:



Les cartes Grad-CAM demeurent centrées sur la forme et les nervures des feuilles, sans attraction notable vers le nouveau fond, ce qui indique que la tête espèce s'appuie principalement sur la morphologie de la feuille. Les prédictions d'espèce restent correctes avec des probabilités globalement élevées, même si l'on observe une baisse plus marquée pour certaines espèces comme Blueberry, Cherry et Grape.

Prédictions correctes maladie:



Les résultats montrent une sensibilité au changement de fond qui dépend fortement de la maladie : northern_lb, Haunglongbing, tomato_yellow et cercospora_leaf restent globalement stables, avec des probabilités encore élevées et des CAM centrées sur les lésions. En revanche, pour cedar_apple_rust et early_blight, la forte chute de probabilité et la diffusion des activations vers les bords et la zone de transition avec le fond saumon suggèrent un biais de contexte (fond/cadrage) plus marqué pour ces classes spécifiques.

Ces essais avec un fond saumon suggèrent une dépendance au contexte globalement modérée et variable selon les classes, mais faute de temps, nous ne pouvons exclure que d'autres variations de fond (couleurs, textures, cadrages) renforcent, atténuent ou au contraire ne modifient pas ces effets.

3.2.5.4. Analyser l'inférence sur de nouvelles photos « *in wild* »

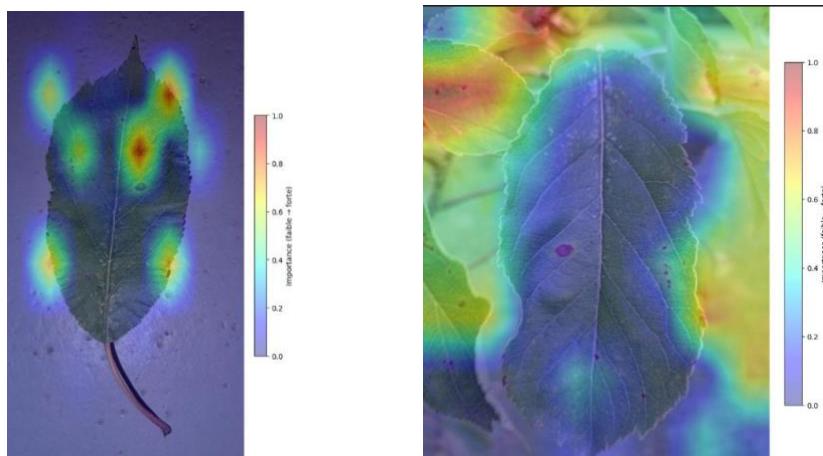
Evaluons la robustesse de notre modèle sur de nouvelles photos « *in wild* »:

Photos en conditions “wild” (prises directement dans une ferme, avec arrière-plan complexe, variations de lumière, feuilles partiellement cachées, salissures, etc.) : le modèle devient beaucoup plus fragile. Les biais appris lors de l’entraînement peuvent provoquer des erreurs systématiques, par exemple confondre des espèces ou ignorer certaines maladies. Ces résultats mettent en évidence la limite du modèle pour une utilisation générique, soulignant la nécessité de diversifier le dataset et d’entraîner le modèle sur des images plus représentatives du monde réel.

Pour illustrer ce point, voici un tableau récapitulatif des prédictions du modèle sur différentes images, avec le type de photo et les probabilités associées à chaque classe :

Image	Type de photo	Espèce (Top-1)	Probabilité Espèce	Maladie (Top-1)	Probabilité Maladie
20251013_200349.jpg	Fond uni	Apple	83.7%	Cedar apple rust	79.4%
20251013_171602.jpg	Wild	Squash	37.9%	Powdery mildew	28.0%
20251013_200400.jpg	Fond uni	Apple	76.5%	Cedar apple rust	50.1%
1000027883.jpg	Wild	Squash	65.1%	Powdery mildew dup	55.3%
potato.png	Wild	Strawberry	44.7%	Powdery mildew	41.4%
poivrons.png	Wild	Pepper, bell	26.2%	Septoria leaf spot	37.1%

On constate que les probabilités sont souvent plus faibles et moins fiables pour les images “wild”, ce qui illustre bien la fragilité du modèle lorsqu'il est appliqué hors du contexte PlantVillage.

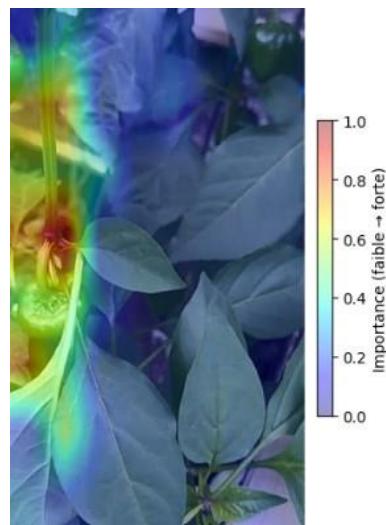


20251013_200349.jpg

20251013_171602.jpg



Poivrons.png



Poivrons.png

Les résultats d'inférence montrent que les erreurs de prédiction observées ne sont pas dues à une faiblesse intrinsèque du modèle, mais à un biais lié au contexte visuel des images. En effet, le modèle a été entraîné sur des feuilles avec un fond uniforme, tandis que certaines images de test présentent un fond 'wild' (naturel), introduisant des textures et des couleurs non vues pendant l'apprentissage. Ce phénomène, conduit le modèle à accorder une importance excessive au fond plutôt qu'aux caractéristiques de la feuille. Les heatmaps Grad-CAM confirment cette tendance lorsque les zones chaudes s'étendent au-delà de la feuille. Pour atténuer ce biais et améliorer la robustesse du modèle, plusieurs stratégies sont recommandées :

- Segmentation préalable des feuilles pour supprimer le fond avant la classification.
- Augmentation des données avec des fonds variés afin de simuler des conditions réelles.
- Fine-tuner le modèle avec des images "wild" afin de l'adapter aux nouveaux types de fonds et de contextes visuels.
- Contrôle systématique via Grad-CAM pour vérifier que l'attention du modèle se concentre sur la feuille et les symptômes.

La technique Grad-CAM permet de confirmer ou d'invalider certaines limitations que nous avons identifiées dans le chapitre sur la présentation des architectures. Nous constatons l'importance de la prise en compte du contexte visuel, essentielle pour garantir la fiabilité de nos modèles. Une approche combinant segmentation, enrichissement des données et interprétabilité permettra d'obtenir un modèle plus robuste et exploitable en conditions réelles.

3.2.6. Sélection et recommandations

La sélection et les recommandations se feront pour deux contextes de déploiement : Production standard - Applications professionnelles et Applications mobiles / Edge computing à l'aide des critères sélectionnés et les constatations fournies par les heatmap.

Compte tenu des classements précédents, nous ne retenons que les architectures : archi1, archi2, archi3, archi9. La synthèse des résultats de tous les critères est présentée ci-dessous avec le jeu de couleurs suivant :

- Vert clair : $F1 \geq 0.995$ (excellent).
- Jaune clair : $0.990 \leq F1 < 0.995$ (très bon).

Métrique	Multi-tâche - 2 têtes + concat. espèce + santé (Arch 9)	Mono-tâche - 3 modèles indépendants (Arch 1)	Mono-tâche - 1 tête (35 classes) (Arch 3)	Mono-tâche - 2 modèles indépendants (Arch 2)
F1 moyen	99.50%	99.44%	99.54%	99.56%
F1 species	99.86%	99.87%	99.87%	99.90%
F1 disease	99.13%	99.01%	99.21%	99.22%
Accuracy moyen	99.70%	99.62%	99.71%	99.68%
Écart Acc-F1	0.20%	0.18%	0.17%	0.12%
Paramètres	137M	408M	137M	137M
Coût relatif	1.0x	3.0x	1.0x	1.0x
Nb modèles	1	3	1	2
Inférences	3x	3x	1x	2x

Figure 22 : Synthèse des critères évalués pour les architectures retenues

Le tableau suivant nous présente la synthèse des performances obtenues par cas :

Rang	Arch	Nom	Cas1 (species)	Cas2 (disease)	Cas3 (complet)	Coût
1	Archi 3	Mono-tâche - 1 tête (35 classes)	99,87%	99,21%	99,54%	+
2	Archi 9	Multi-tâche - 2 têtes + concat. espèce + santé	99,86%	99,13%	99,50%	++
3	Archi 2	Mono-tâche - 2 modèles indépendants	99,90%	99,22%	99,56%	+++
4	Archi 1	Mono-tâche - 3 modèles indépendants	99,87%	99,01%	99,44%	++++

Figure 23 : synthèse des performances obtenues par cas

Pour un déploiement sur Applications mobiles / Edge computing, nous retenons l'architecture 3 (Mono 35 classes). Classée #2, elle atteint un F1 moyen de 99,53% — quasi identique à l'archi9 — avec (1 modèle, 1 inférence) qui réduit latence et complexité. Le tableau de synthèse indique un coût relatif de 1,0x (137M params) et un écart Accuracy–F1 contenu ($\approx 0,20\%$), offrant le meilleur compromis précision/complexité/latence pour un environnement contraint. La maintenance et l'intégration sont simplifiées grâce à la mono-tête. Exemple de profil d'utilisation recommandé : Applications mobiles (iOS/Android) avec contraintes de batterie, Systèmes embarqués (Raspberry Pi, Jetson Nano), Déploiement edge avec connectivité intermittente, budget : minimal (1 seul modèle). Avec l'hypothèse de déploiement : “Assistant IA pour un usage contrôlé, proche de PlantVillage (mêmes espèces, mêmes maladies, conditions assez proches)”.

Pour un déploiement en Production standard – Applications professionnelles, nous sélectionnons l'architecture 9. Elle présente un F1 moyen (99,55%) et des scores équilibrés sur les trois cas, tout en présentant l'écart Accuracy–F1 le plus faible ($\approx 0,15\%$), signe d'une bonne calibration et d'une robustesse opérationnelle. Avec 137M de paramètres et un seul modèle à maintenir, elle reste fiable malgré 3 inférences, grâce à la stabilité de son schéma multi-tâches. Exemple de profil d'utilisation recommandé : Applications web/serveur où un temps de réponse < 500 ms est acceptable, Systèmes de diagnostic professionnels pour botanistes, Déploiement cloud avec GPU, Budget d'hébergement : 1 modèle. Pour un déploiement pour un agriculteur : S'il envoie surtout des feuilles malades issues de situations très proches de PlantVillage, le biais espèce–maladie ne va pas forcément altérer les performances à court terme. Le modèle reste utilisable dans un contexte très contrôlé (PlantVillage-like). Cependant, ce biais rend le modèle fragile dès que les combinaisons espèce/maladie changent.

3.2.7. Limites et perspectives

Les limitations clés :

1. Validation terrain absente : Test en conditions agricoles réelles / Validation par experts botanistes / Pilote terrain avec agriculteurs / Feedback utilisateurs finaux. En conséquence, les performances présentées constituent une borne supérieure en conditions contrôlées.
2. Biais du dataset PlantVillage : Fond uniforme (blanc/vert uni)/ Éclairage contrôlé (studio)/ Feuilles isolées, détournées/ Angles standardisés. Conséquence, une chute de performance - 5% à -15% en conditions réelles.
3. Validation statistique limitée : nous avons effectué 5 runs sur l'architecture 7 uniquement pour des problèmes de performances, et les résultats montraient un modèle robuste. Pour les autres architectures : 1 seul run par architecture / Seed aléatoire unique / Pas de test statistique (Test de significativité ($p_value < 0,05$)). Conséquence : les résultats exacts peuvent varier ($\pm 0.1\text{--}0.2\%$)
4. Classes maladies difficiles identifiées : Faux négatifs : Maladie rare non détectée → propagation de la maladie. Faux positifs : Sur-traitement inutile (coût + écologie)

Perspectives d'amélioration

1. Amélioration et extension du dataset

- Augmenter la diversité des données : Élargir le jeu de données en ajoutant de nouvelles combinaisons espèce–maladie, davantage d'images de feuilles saines, ainsi que des contextes variés (prises de vue en extérieur, arrière-plans complexes), et en intégrant des jeux de données complémentaires comme PlantSeg, ..., qui contiennent des images en conditions réelles. Cela permettra de casser la corrélation directe espèce–maladie et d'entraîner des modèles plus robustes et généralisables. Analyser les classes difficiles pour identifier les lacunes dans les données actuelles.
- Rééquilibrage et augmentation : Utiliser GRAD-CAM comme outil de diagnostic pour guider l'augmentation et le rééquilibrage du dataset, en ciblant les classes sous-représentées ou mal classées.

2. Optimisation des modèles pré-entraînés

- Sélection plus poussée des modèles pré-entraînés : Expérimenter différents modèles pré-entraînés (ex. ResNet, ConvNeXt, Swin Transformer...) pour identifier celui qui offre le meilleur compromis entre précision, généralisation et ressource. Affiner la sélection en fonction des performances sur les classes difficiles et des heatmaps GRAD-CAM.
- Expérimenter avec des architectures alternatives : Tester des architectures comme Vision Transformer (ViT), qui pourraient mieux capturer les dépendances globales et locales dans les images de feuilles.

3. Amélioration des architectures et de l'entraînement

- Valider la robustesse des architectures : Effectuer 3 à 5 runs multiples avec des statistiques (moyenne, écart-type) pour vérifier la stabilité et la reproductibilité des performances des architectures retenues.
- Ajuster les têtes de classification : Utiliser GRAD-CAM pour analyser les zones de l'image utilisées par le modèle pour chaque prédiction. Cela permettra d'ajuster l'architecture ou les têtes de classification (repenser la décomposition des tâches, modifier le nombre ou le

type de têtes, ajuster les pertes associées aux têtes) afin que le modèle se concentre sur les caractéristiques pertinentes, indépendamment des biais du dataset.

- Optimiser les hyperparamètres : Utiliser les jeux de données de validation pour affiner les hyperparamètres (ex. taux d'apprentissage, taille des couches) et améliorer les performances globales.

4. Conclusion

La première phase du projet s'est appuyée sur une démarche méthodique, depuis l'analyse exploratoire des données brutes jusqu'à la sélection des caractéristiques les plus pertinentes. Le choix du dataset PlantVillage a permis une immersion progressive dans les outils de Machine Learning tout en maîtrisant la complexité du sujet. Grâce au travail d'ingénierie de caractéristiques et à une analyse approfondie, nous avons enrichi notre compréhension des données, mis en évidence leurs spécificités et leurs contraintes, et posé des bases solides pour la suite de l'étude.

Pour la phase de modélisation, nous avons exploré deux axes complémentaires : d'une part, des modèles classiques de Machine Learning (SVM, XGBoost, Extra-Trees, Régression Logistique) ; d'autre part, des architectures de Deep Learning. Parmi les modèles classiques, le SVM à noyau RBF s'est distingué comme le plus performant pour l'identification des 14 espèces, offrant un bon compromis précision/rappel et une certaine robustesse face au déséquilibre des classes. XGBoost et Extra-Trees ont également obtenu des résultats prometteurs, même si un réglage plus poussé des hyperparamètres aurait pu améliorer leurs performances. Les analyses par classe ont mis en lumière des disparités : d'excellents résultats pour Blueberry, Tomato et Grape, mais des difficultés persistantes sur des classes plus ambiguës (Pepper_bell, Potato, Strawberry), ouvrant la voie à des pistes d'amélioration ciblées (rééquilibrage, augmentation de données).

Afin d'appréhender la diversité des architectures de Deep Learning et de mieux comprendre leur comportement, nous avons défini et évalué neuf architectures distinctes. Cette phase d'exploration nous a permis d'identifier les forces et les limites de chaque approche. À l'issue d'une première sélection, trois architectures ont été écartées pour des raisons de performance ou d'adéquation avec nos objectifs. Les six restantes ont été évaluées de manière approfondie sur trois cas d'usage opérationnels, représentant des scénarios variés. Bien que le déploiement ne fasse pas partie des exigences du projet, cette analyse nous a conduits à retenir deux architectures parmi les plus performantes (archi9 et archi3), associées à un scénario précis, en vue d'un éventuel déploiement en production.

Nous avons également mis en évidence l'intérêt de combiner les métriques quantitatives (accuracy, F1-score) avec des visualisations de type Grad-CAM pour évaluer la performance réelle du modèle sur les lésions, au-delà des seuls chiffres. Nos résultats suggèrent toutefois que le modèle exploite en partie des corrélations espèce–maladie propres au dataset PlantVillage, comme l'indiquent des heatmaps Grad-CAM qui mettent parfois en avant des caractéristiques morphologiques des feuilles plutôt que les symptômes de la maladie. Ce biais, documenté dans la littérature récente, limite la capacité de généralisation du modèle à des environnements réels.

Malgré des résultats très performants, nous sommes conscients des limites de notre travail. Ce projet a néanmoins constitué une expérience extrêmement formatrice, où chaque difficulté rencontrée a été l'occasion d'apprendre, d'ajuster notre démarche et de renforcer notre compréhension des méthodes de Machine Learning et de Deep Learning.

5. Retour d'expérience

Dans ce projet, les difficultés rencontrées nous ont régulièrement obligées à chercher des solutions et à nous réorganiser. En tant que débutants dans le domaine, nous avons dû passer par de nombreuses itérations, faute de recul suffisant, et la prise en main de certains outils et bibliothèques s'est révélée plus complexe que prévu.

Le décalage entre le calendrier des cours et les besoins du projet a également pesé : certaines notions (par exemple MLflow) n'ont été abordées qu'après le moment où elles auraient été les plus utiles, ce qui a limité le périmètre des expériences systématiquement tracées. Nous avons également manqué de temps pour revenir sur les recommandations issues de l'analyse exploratoire et ajuster les modèles de manière plus approfondie.

Ce projet a été un formidable accélérateur de compréhension des cours : le fait de devoir mettre en pratique, presque en temps réel, des nouveaux concepts introduits a fortement consolidé nos acquis. Les revues mensuelles avec le référent ont joué un rôle important dans cette dynamique, en apportant un regard extérieur.

Sur le plan opérationnel, plusieurs contraintes techniques ont ralenti la progression de nos travaux. Les performances limitées de nos PC personnel ont freiné l'exécution de certains entraînements, et l'exploitation efficace des GPU n'a pas été immédiate, nécessitant des adaptations de code et une meilleure maîtrise de l'environnement. La dépendance à différents environnements (Colab, Codespaces, machines DataScientist) a parfois provoqué des interruptions ou des écarts de configuration.

Malgré cela, la mise en place d'un cycle de vie de projet structuré, avec des jalons clairement définis, a permis de travailler dans une relative sérénité : les objectifs ont été atteints, même si certaines tâches ont demandé plus de temps que prévu.

L'analyse des résultats obtenus avec les différentes architectures de DL testées a mis en évidence la difficulté à garantir une bonne capacité de généralisation en conditions réelles de déploiement dans le cadre trop large de notre scénario opérationnel. Cette expérience souligne l'importance de renforcer la collaboration avec les experts métiers, afin de valider en continu la pertinence des scénarios et des données mobilisées.

Le principal verrou scientifique du projet tient à la difficulté de construire des modèles généralisables à partir d'un dataset limité et biaisé comme PlantVillage, caractérisé par un contexte d'images artificiel (feuilles sur fond uniforme), une forte corrélation entre espèce et maladie et un décalage marqué avec les conditions réelles d'usage. Même si les performances obtenues sur ce dataset sont élevées, il reste délicat de garantir que les modèles apprennent des représentations robustes et transférables en situation réelle, ce qui limite la portée des conclusions.

6. Bibliographie

Publications :

- [2024] - [Trends in Machine and Deep Learning Techniques for Plant Disease Identification: A Systematic Review](#) - Diana-Carmen Rodríguez-Lira , Diana-Margarita Córdova-Esparza , José M. Álvarez-Alvarado Juan Terven , Julio-Alejandro Romero-González, JuvenalRodríguez-Reséndiz
- [2024] - [A systematic review of machine learning and deep learning approaches in plant species detection](#) - Deepti Barhate , Sunil Pathak, Bhupesh Kumar Singh , Amit Jain , Ashutosh Kumar Dubey

- [2024] - [A systematic review of deep learning techniques for plant diseases](#) - Ishak Pacal · Ismail Kunduracioglu · Mehmet Hakki Alma · Muhammet Deveci · Seifedine Kadry · Jan Nedoma · Vlastimil Slany · Radek Martinek
- [2024] - [A Systematic Literature Review of Machine Learning and Deep Learning Approaches for Spectral Image Classification in Agricultural Applications Using Aerial Photography](#) – Usman Khan, MuhammadKhalidKhan, MuhammadAyubLatif, MuhammadNaveed, MuhammadMansoorAlam, SalmanA.Khan, MazlihamMohdSu'ud

Livres :

- [2022] - [The StatQuest Illustrated Guide To Machine Learning](#) – Josh Starmer
- [2017] - [Machine Learning avec Scikit-Learn](#) – Aurélien Géron – Dunod
- [2017] - [Deep Learning avec TensorFlow](#) – Aurélien Géron – Dunod

Des recherches complémentaires ont été effectuées sur Internet afin de mieux comprendre les méthodes de classement, d'organisation et d'archivage de documents. Des ressources variées, comme des sites éducatifs, des forums spécialisés ou encore des blogs techniques, ont été consultés pour enrichir nos connaissances théoriques et pratiques. Par ailleurs, des outils d'intelligence artificielle tels que ChatGPT et Le chat ont été utilisés pour poser des questions précises, obtenir des explications claires, explorer différentes approches et affiner certaines idées.

7. Annexes

7.1. Description du code

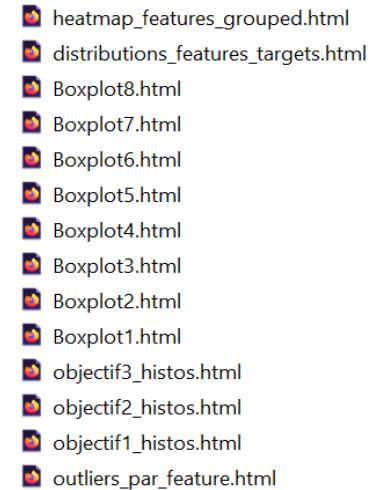
Voici le lien GitHub du dépôt : https://github.com/mackjb/ai_reco_plante.

Ce dépôt contient les expériences de reconnaissance de maladies de plantes (Machine Learning et Deep learning) autour du dataset PlantVillage.

Voici le lien ReadMe : https://github.com/mackjb/ai_reco_plante/blob/main/README.md

7.2. Analyse exploratoire des caractéristiques extraites

L'analyse exploratoire était supportée par des graphiques (histogrammes, boxplot, heatmap...), générés avec plotly et des menus déroulants. Ils sont enregistrés dans des fichiers HTML pour explorer plus précisément les caractéristiques :



Gestion des données manquantes

Nous avons détecté 9 images avec NaN. Dans un premier temps, nous les avons supprimées. On pourra toujours les trouver si besoin.

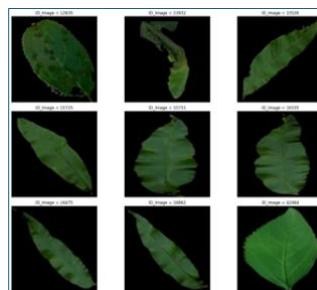


Figure : données manquantes

Identification des outliers : La détection des outliers est faite via la méthode IQR pour chaque variable. Nombre de lignes avec au moins un outlier: 21900.

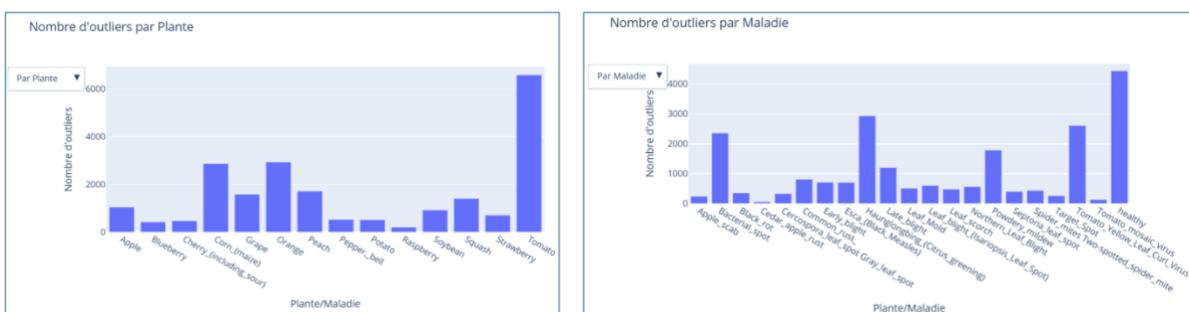


Figure : Identification des outliers par plante – identification des outliers par maladie

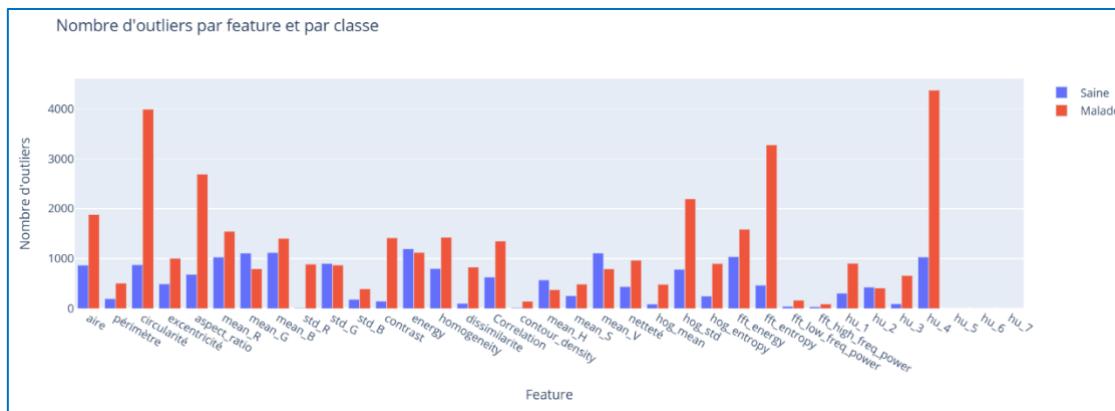


Figure : Identification des outliers par feature et par classe

Observations :

Le taux d'outliers est élevé : 40,34%. On constate que, pour la majorité des variables, la classe "malade" affiche significativement plus d'outliers que la classe "saine". Circularité, fft_entropy et hu_4, se distinguent avec un nombre élevé d'outliers chez les malades.

Risques :

La surreprésentation d'outliers chez les malades confirme l'hétérogénéité importante au sein de cette classe, mais cela peut aussi révéler des problèmes potentiels de qualité de données ou de mesures extrêmes spécifiques à certains cas qui risquent de biaiser les analyses statistiques et les modèles prédictifs, en faussant la détection des vraies différences entre classes. Enfin, la présence d'un nombre très faible ou nul d'outliers sur certaines features pourrait signaler des variables peu discriminantes ou mal calibrées. Les features avec beaucoup d'outliers risquent de perturber certains algorithmes sensibles aux valeurs extrêmes. Supprimer les outliers, sans discernement, risque d'enlever des cas légitimes et importants. Possible présence de valeurs aberrantes parmi les outliers détectés.

Conclusions :

Ces graphiques mettent en évidence l'importance de traiter et d'analyser spécifiquement les outliers. Avant toute modélisation, il sera essentiel d'investiguer l'origine de ces valeurs extrêmes, de décider de leur prise en compte (suppression, correction, transformation) ou utilisation de modèles robustes aux outliers. Analyse approfondie des outliers : Inspecter visuellement et statistiquement les valeurs extrêmes pour distinguer les vraies valeurs aberrantes (bruit, erreur) des observations atypiques mais valides. Tester l'impact du traitement des outliers sur la performance des modèles (avec/sans, transformation...). Prioriser les features discriminantes : Les variables qui montrent beaucoup d'outliers chez les malades pourraient être de bons marqueurs pour la classification santé/maladie.

Dans le chapitre sur les Représentations visuelles, les boxplots sont également des représentations pertinentes pour explorer la nature des outliers.

Distribution des caractéristiques

Un jeu d'histogrammes interactifs permet l'exploration de la distribution des caractéristiques suivant l'objectif auquel le modèle devra répondre.

Objectif1 : Quelle est cette plante ?

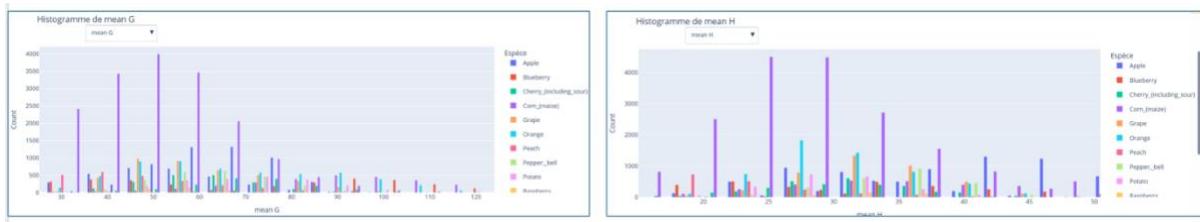


Figure : Distribution des caractéristiques pour objectif 1 "Quelle est cette plante?"

Observations :

Variabilité importante entre espèces : Les distributions de certaines features (aire, couleur, texture...) varient fortement d'une plante à l'autre, ce qui suggère que plusieurs caractéristiques sont potentiellement très discriminantes pour l'identification de la plante.

Features très redondantes ou corrélées : Certaines familles de variables (ex : canaux couleurs, moments de Hu) présentent des corrélations élevées entre elles, ce qui peut indiquer une certaine redondance à traiter lors de la sélection des variables.

Caractéristiques quasi constantes ou peu informatives pour certaines classes : Certaines variables varient peu selon l'espèce, ce qui les rend peu utiles pour la classification de la plante (par exemple, une texture ou une couleur presque identique dans toutes les classes).

Risques :

Un fort déséquilibre entre espèces peut biaiser le modèle, qui aura tendance à privilégier les classes majoritaires lors de la prédiction. Les espèces sous-représentées risquent d'être mal reconnues, augmentant le taux d'erreur pour ces classes. Les performances globales du modèle peuvent être trompeuses et masquer des faiblesses sur les minorités. Ce déséquilibre complique la généralisation et la robustesse de la solution sur l'ensemble des plantes du jeu de données.

Conclusions :

On observe que la variabilité de certaines caractéristiques entre espèces offre de bons leviers pour la classification. Toutefois, la redondance, la présence d'outliers et des variables peu informatives devront être prises en compte pour optimiser la robustesse et la performance du modèle pour l'objectif "identifier la plante".

Objectif2 : Cette plante est-elle saine ?

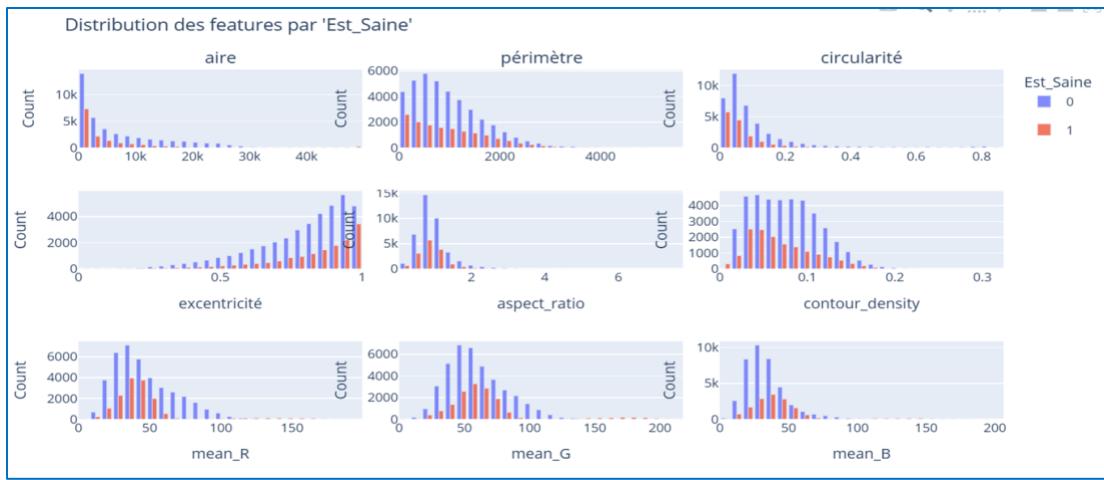


Figure : Distribution des caractéristiques pour objectif 2 (1/3)

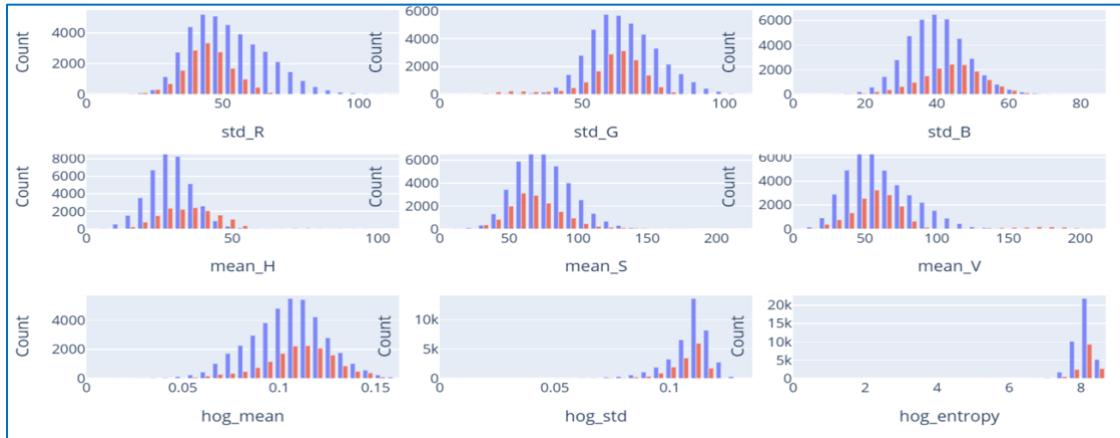


Figure : Distribution des caractéristiques pour objectif 2 (2/3)

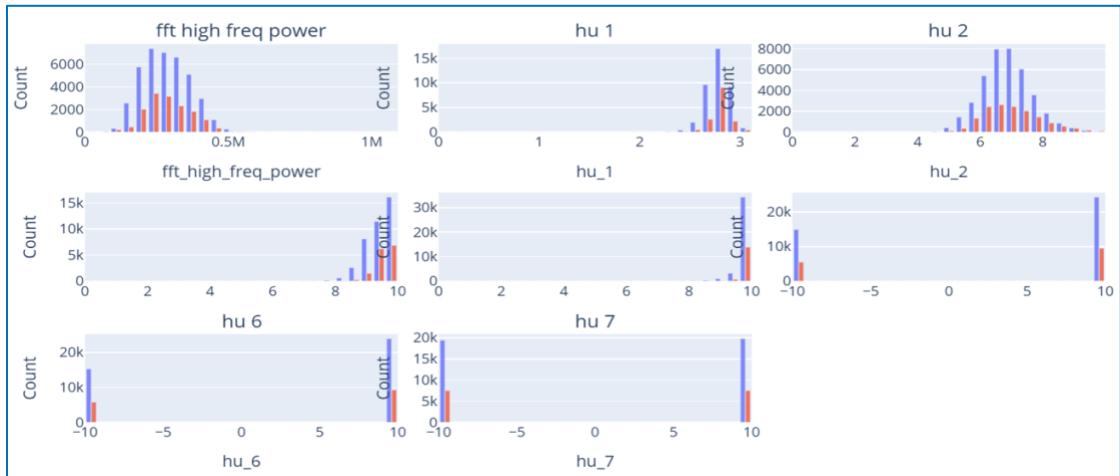


Figure : Distribution des caractéristiques pour objectif 2 (3/3)

Observations :

Les variables morphologiques (aire, périmètre, circularité) et colorimétriques (mean R, G, B) montrent des différences nettes entre les classes saines (bleu) et malades (rouge). Les plantes malades présentent souvent une aire et une circularité plus faibles, avec des valeurs de teinte (H) et saturation (S) distinctes. Les mesures de texture (hog, fft, hu) confirment cette séparation.

Risques :

Les plantes avec ces caractéristiques tendent vers une altération de structure et de couleur indicative de stress ou infection. Une mauvaise identification pourrait survenir si certaines valeurs se chevauchent, surtout dans les variables de texture.

Conclusions :

Le profil global du graphique correspond davantage aux distributions observées pour les plantes malades. Les écarts sur la couleur et la morphologie suggèrent un état non sain. Cette plante est donc probablement malade.

Objectif3 : Quelle est cette maladie ?

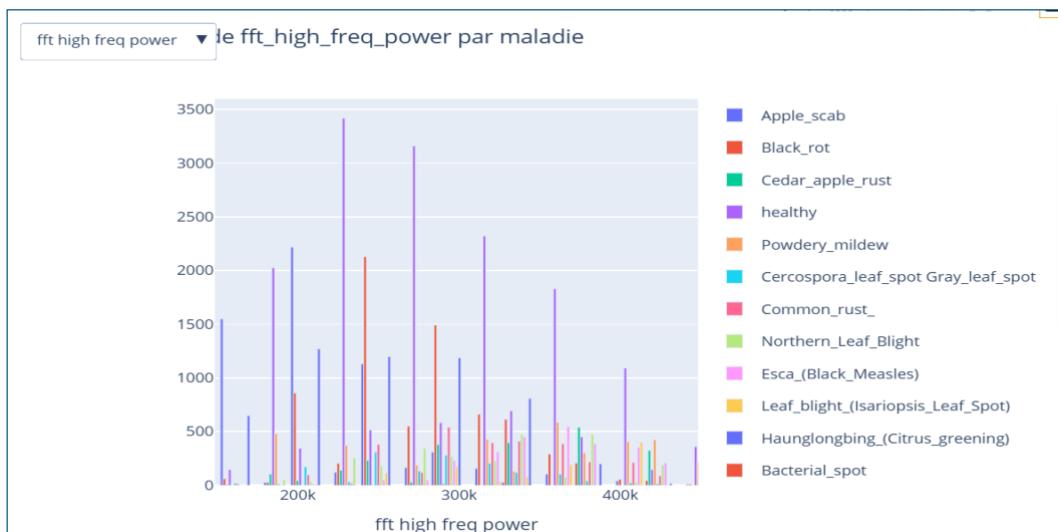


Figure : Distribution des caractéristiques pour objectif 3 "Quelle est la maladie ?"

Observations :

Les graphiques présentent la répartition des images par maladie, toutes plantes confondues. On constate que certaines maladies (ex : Apple_scab, Powdery_mildew, Leaf_blight_, Haunglongbing_) sont beaucoup plus représentées dans le jeu de données, tandis que d'autres ne comptent que quelques exemples. Ce déséquilibre entre maladies est très marqué, certaines pathologies rares n'apparaissant que sur une poignée d'images.

Risques :

Un fort déséquilibre entre maladies expose le modèle à un risque de biais : il apprendra surtout à reconnaître les maladies majoritaires et aura du mal à prédire correctement les maladies rares, même si elles sont importantes en pratique.

Conclusion :

Pour répondre correctement à l'objectif 3 (identifier la maladie), il sera essentiel de corriger ou d'atténuer ce déséquilibre (ex : data augmentation, sous-échantillonnage, pondérations lors de l'apprentissage). Cela garantira que le modèle accorde une attention équitable à chaque maladie et que la détection des maladies rares soit suffisamment fiable pour un usage terrain.

Représentations visuelles des caractéristiques :

Ces visualisations permettent d'identifier des variables clés qui séparent bien les plantes saines des malades. Elles permettent également de repérer les comportements atypiques (valeurs extrêmes) qui méritent une investigation (plante en transition de santé, mesure erronée, etc.). Aider au choix des variables pour un futur modèle de classification visant à prédire si une plante est malade. Le nom du dossier contenant les fichiers HTML des Boxplots : Boxplots

- **Boxplot1 et Boxplot2** : ils présentent une grande quantité de données et plusieurs classes, mais la lisibilité est réduite : difficile d'identifier visuellement des différences marquées entre malades et non-malades.
- **Boxplot3** : intéressant car il met en évidence des médianes différentes entre maladies et une variabilité interne bien visible, mais il est légèrement moins clair que Boxplot2 pour distinguer les distributions.
- **Boxplot4** : plus simple et lisible, mais les différences entre classes sont moins marquées, limitant son intérêt pour identifier une plante malade.
- **Boxplot5** : il met en évidence de fortes dispersions avec des valeurs extrêmes (outliers), mais sans séparation claire par maladie.
- **Boxplot6** : comparable au Boxplot3 en termes de clarté, il offre une bonne visualisation de la dispersion mais moins de contraste entre classes.
- **Boxplot7** : très riche en points et en variabilité, mais les différences entre classes sont moins nettes, rendant l'interprétation plus complexe.
- **Boxplot8** : plus condensé visuellement et facile à lire, mais la séparation des distributions entre maladies y est peu marquée.

Explorons un des boxplots : le **Boxplot2** particulièrement intéressant à commenter pour l'objectif 3 : “*Quelle est cette maladie ?*”.

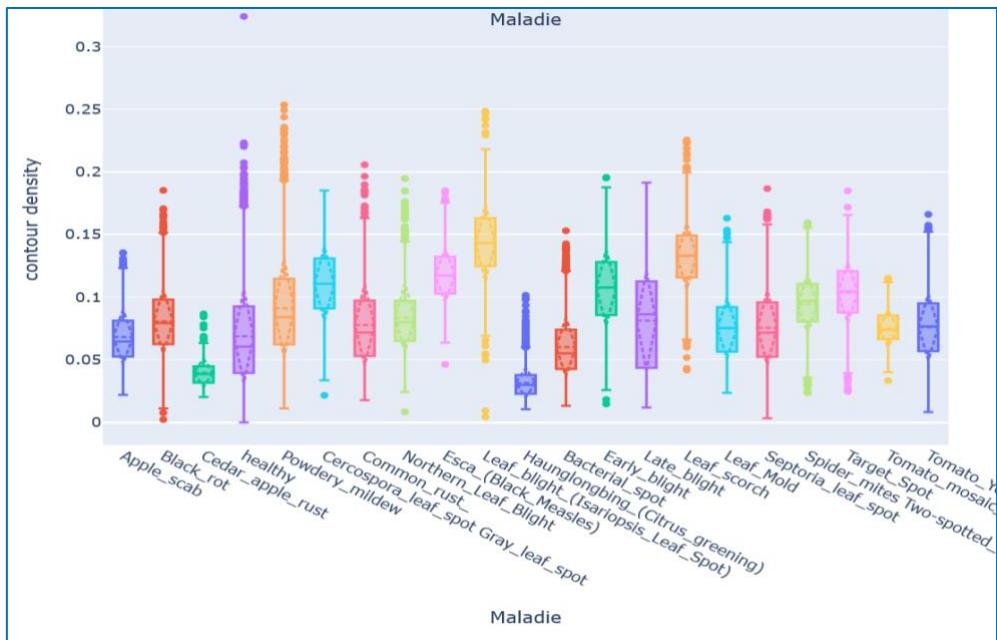


Figure : Boxplots de la densité de contours selon les différentes maladies

Le Boxplot2 met en évidence la variable `contour_density` pour distinguer deux maladies du pommier : `Apple_scab` et `Black_rot`. Ces maladies présentent des symptômes visuellement contrastés sur un même type de feuilles de pommier, limitant les biais liés aux différences d'espèces et facilitant l'interprétation des écarts de médiane et de dispersion. Cette comparaison illustre clairement la capacité des boxplots à révéler des variables discriminantes pour l'objectif 2, qui vise à isoler les caractéristiques clés selon le type de maladie. Un axe d'amélioration consistera à générer des boxplots par plante, afin d'explorer d'autres maladies (ex. `Leaf_blight` ou `Haunglongbing`) tout en conservant la cohérence inter-espèces.

Histogramme présentant la distribution des caractéristiques par type de plantes : Le nom du fichier HTML : `distributions_features_targets.html`

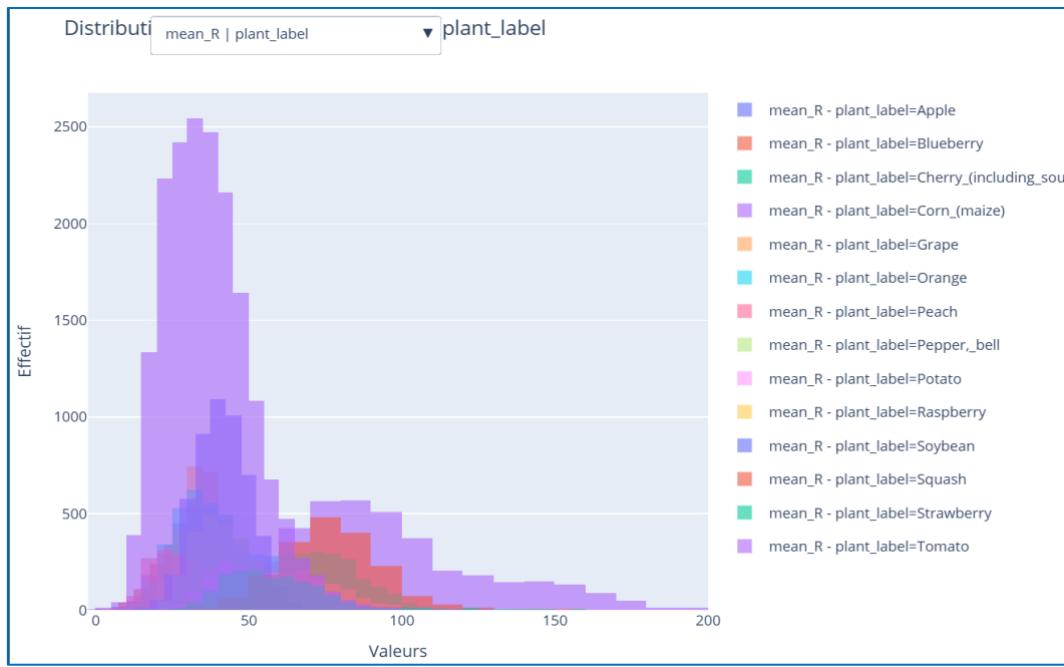


Figure : Histogramme de la distribution des caractéristiques par type de plantes

Relations entre les caractéristiques

Heatmaps présentant les corrélations entre features: Nom du fichier HTML:

heatmap_features_grouped.html

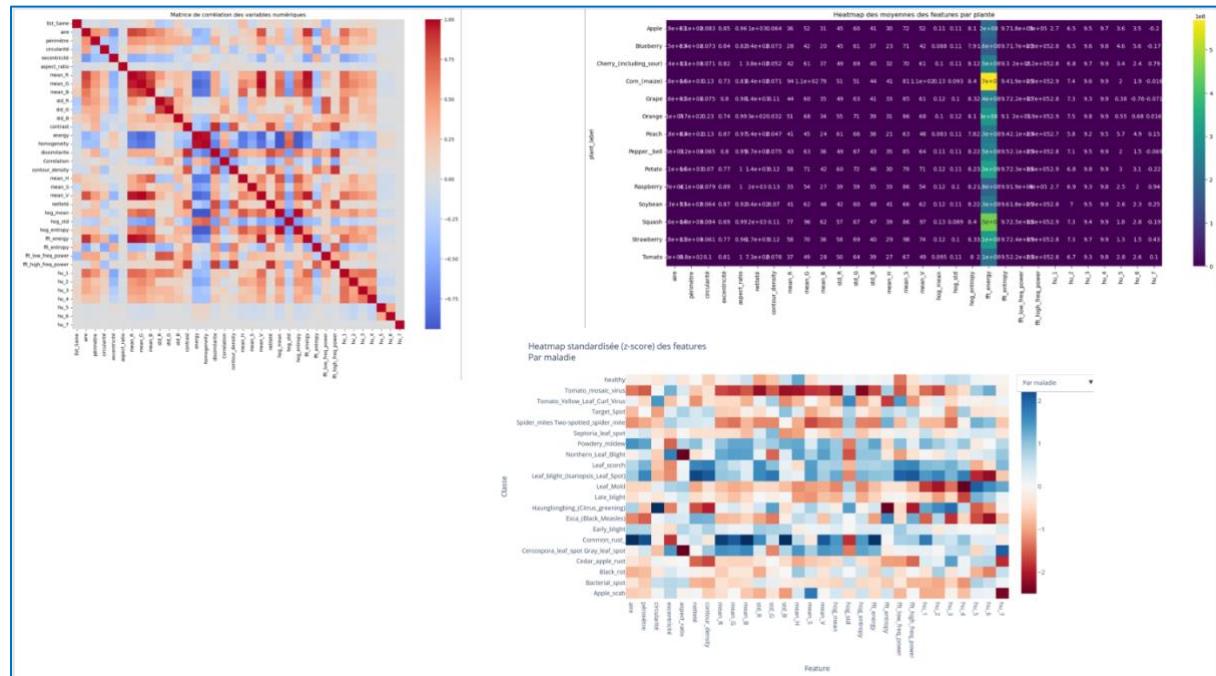


Figure : Heatmaps des corrélations entre features

Observations :

- On observe une grande variabilité des profils z-score des features entre les différentes espèces de plantes.
- Certaines plantes présentent des profils très contrastés (zones rouges/bleues marquées), ce qui indique des caractéristiques spécifiques fortes.
- Aucune plante n'a exactement la même "empreinte" de feature, c'est intéressant pour la classification d'espèces.
- Le contraste est beaucoup plus net : chaque feature est soit nettement au-dessus, soit nettement au-dessous de la moyenne globale.
- La plupart des features discriminent de façon binaire entre sain et malade : fort potentiel pour des modèles de classification binaire.
- Les profils "healthy" (sains) se distinguent assez bien, avec des scores souvent proches de la moyenne (blancs).
- Certaines maladies affichent des signatures très marquées sur certaines features, ce qui peut aider à leur identification automatique.

Risques :

- Certains groupes de features semblent évoluer ensemble. A priori, il y a de la redondance. Risque de surapprentissage
- Variabilité intra-classe : Pour certaines plantes ou maladies, la couleur des cases oscille beaucoup, ce qui peut indiquer une hétérogénéité importante dans la classe. Cela pourrait compliquer la classification fine.
- Effet de standardisation : Comme l'échelle est relative (z-score), une forte déviation sur une feature peut être causée soit par une vraie différence, soit par une variance très faible pour cette feature.
- Binarité excessive (statut sain/malade) : Si les différences sont trop franches et systématiques sur toutes les features, risque d'avoir des modèles "trop parfaits" sur le jeu d'entraînement et moins robustes en situation réelle

Conclusions :

- Sélection de features pertinentes : Utiliser l'information des heatmaps pour prioriser les features qui montrent des différences nettes entre classes (plante, maladie, statut) tout en surveillant les redondances.
- Analyse approfondie des profils atypiques : Vérifier pourquoi certaines plantes ou maladies ont des patterns très différents ou très fluctuants, et si cela correspond à des outliers, ou une vraie diversité biologique.
- Affinement des modèles :
 - Pour la classification binaire (sain/malade), de nombreuses features sont prometteuses : tester différents algorithmes.
 - Pour la classification multi-classes (espèce, maladie), travailler la réduction de dimensionnalité et l'ingénierie de features pour optimiser la séparation entre groupes.

L'Analyse en Composantes Principales (PCA) a été utilisée lors de la modélisation pour SVM et XGBoost mais sans apporter un réel bénéfice.

7.3. Les méthodes pour analyser l'importance des features

Voici un tableau de synthèse des différentes approches possibles pour analyser l'importance des features.

Méthode	Type	Principe	Classification binaire	Classification multi-classes	Avantages	Inconvénients
Variance Threshold	Filtre	Supprime les features dont la variance est inférieure à un seuil	✓	✓	Très rapide à exécuter	Ne tient pas compte de la cible
Corrélation (Pearson)	Filtre	Retire les features fortement corrélées entre elles	✓	✓	Réduit multicolinéarité	Seulement linéaire
Chi-2	Filtre	Test d'indépendance entre feature catégorielle et cible	✓	Limitée (binaires ou k-classes >2)	Simple et efficace pour variables discrètes	Ne gère pas bien les variables continues
ANOVA F-test	Filtre	Compare moyennes des groupes de la cible	-	✓	Approprié pour variables numériques	Hypothèses normales
Mutual Information	Filtre	Mesure l'information partagée entre feature et cible	✓	✓	Capture relations non linéaires	Plus coûteux qu'un test univarié
Recursive Feature Elimination	Wrapper	Enlève itérativement la moins importante via un estimateur	✓	✓	Prend en compte l'interaction features-cible	Lourd en calcul
Sélection séquentielle (SFS/SBS)	Wrapper	Ajoute ou retire séquentiellement les meilleures features	✓	✓	Flexible (forward/backward)	Risque de surapprentissage, coûteux
L1 Regularization (Lasso)	Intégré	Pénalise les coefficients pour en annuler certains	✓	✓ (via "one-vs-rest")	Rapide, intégré au modèle	Seuil dépendant du paramètre de pénalité
Importance d'arbre (Random Forest)	Intégré	Mesure la réduction d'impureté apportée par chaque feature	✓	✓	Gère variables mixtes, robuste au bruit	Biais vers variables à plus de niveaux
Gradient Boosting Importances	Intégré	Comme pour les forêts, mais via boosting	✓	✓	Meilleure précision souvent	Plus lent que Random Forest

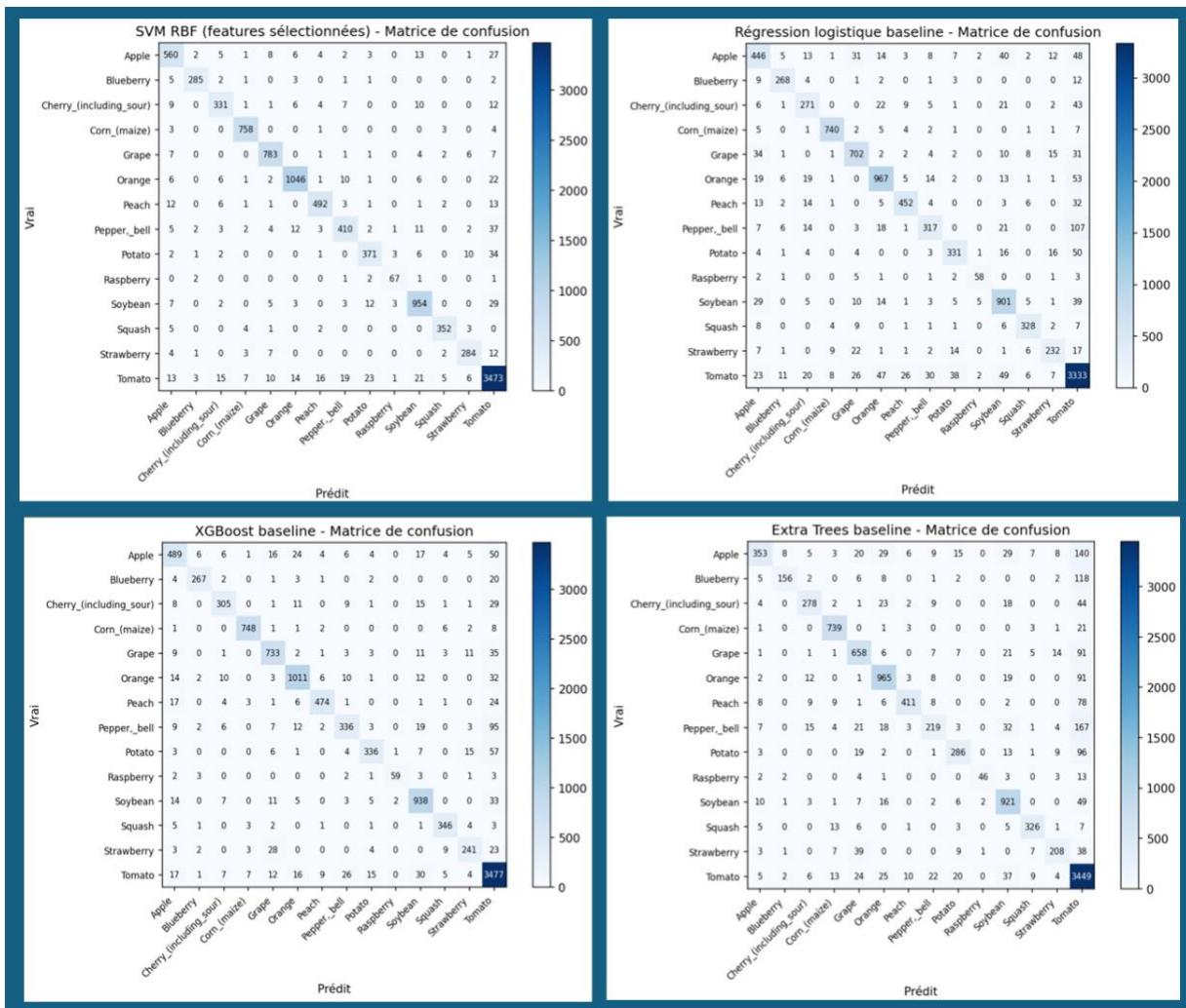
Tableau de synthèse des différentes approches possible pour analyser l'importance des features

- Filtre = sélection avant apprentissage, indépendante du modèle.
- Wrapper = sélection autour du modèle, itérative et évaluée par performance.
- Intégré = sélection pendant l'apprentissage, via des pénalités ou des métriques internes au modèle

7.4. Comparaison des caractéristiques des modèles ML

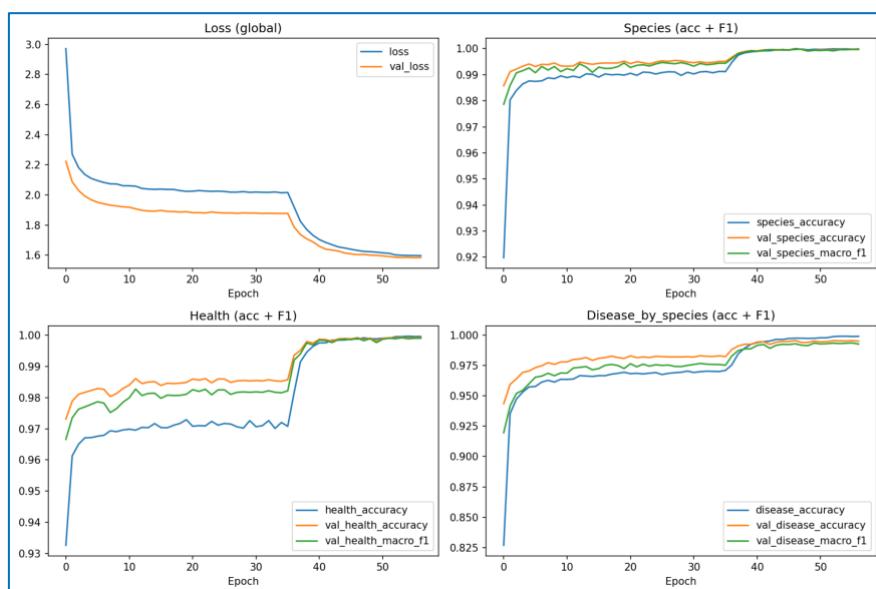
Caractéristique	SVM RBF	XGBoost	Extra Trees	Logistic Regression
Script	models/svm_rbf_plantvillage_features_selected_baseline.py	models/xgboost_baseline.py	models/Xtra_trees.py	models/Log_reg.py
Modèle	SVC(kernel="rbf", decision_function_shape="ovr")	XGBClassifier (GPU si dispo, device="cuda")	ExtraTreesClassifier	LogisticRegression (multinomial, lbfgs)
Pipeline	RobustScaler → SVC	RobustScaler → XGB	RobustScaler → ExtraTrees	RobustScaler → LogisticRegression
CSV (features)	dataset/plantVillage/csv/clean_data_plantvillage_segmented_all_with_features.csv			
Cible	species	species (encodée via LabelEncoder)	species	species
Grille hyperparamètres (résumé)	svm__C=[76.0]; svm__class_weight=[None, "balanced"]	xgb__n_estimators=[300]; xgb__max_depth=[6]; xgb__learning_rate=[0.1,0.05]; xgb__subsample=[0.8]; xgb__colsample_bytree=[0.8]	xtr__n_estimators=[300]; xtr__class_weight=[None, "balanced"]	logreg__C=[0.1, 1.0, 10.0]; logreg__class_weight=[None, "balanced"]
CV (GridSearchCV)	StratifiedKFold(n_splits=5, shuffle=True, random_state=42)	StratifiedKFold(n_splits=5, shuffle=True, random_state=42)	StratifiedKFold(n_splits=5, shuffle=True, random_state=42)	StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
Scoring / refit	{"balanced_accuracy", "f1_macro"}; refit="f1_macro"	{"balanced_accuracy", "f1_macro"}; refit="f1_macro"	{"balanced_accuracy", "f1_macro"}; refit="f1_macro"	{"balanced_accuracy", "f1_macro"}; refit="f1_macro"
n_jobs (CV)	-1	1 si GPU, sinon -1	1	1
Courbes de validation	C, gamma	n_estimators, max_depth	n_estimators, max_features	C
Repeated CV (seeds)	[1,2,3,4,5] (n_jobs=-1)	[1,2,3] (n_jobs=1 si GPU, sinon -1)	[1,2,3] (n_jobs=1)	[1,2,3] (n_jobs=1)
Bootstrap (B)	1000	500	500	500
Mesure(s) de complexité	Ratio vecteurs de support	n_estimators, max_depth, n_classes	n_estimators, profondeur moyenne	n_classes, n_features, coeff ₂
Artefacts principaux	rapport, cm.csv/html, cm_interactive, galerie erreurs, matrice interactive avec images	rapport, cm.csv/html, cm_interactive, label_mapping.json	rapport, cm.csv/html, cm_interactive	rapport, cm.csv/html, cm_interactive
Dossier résultats	results_modifiés/models/svm_rbf_baseleine_features_selected/	results_modifiés/models/xgb_baseleine/	results_modifiés/models/extra_trees_baseleine/	results_modifiés/models/logreg_baseleine/

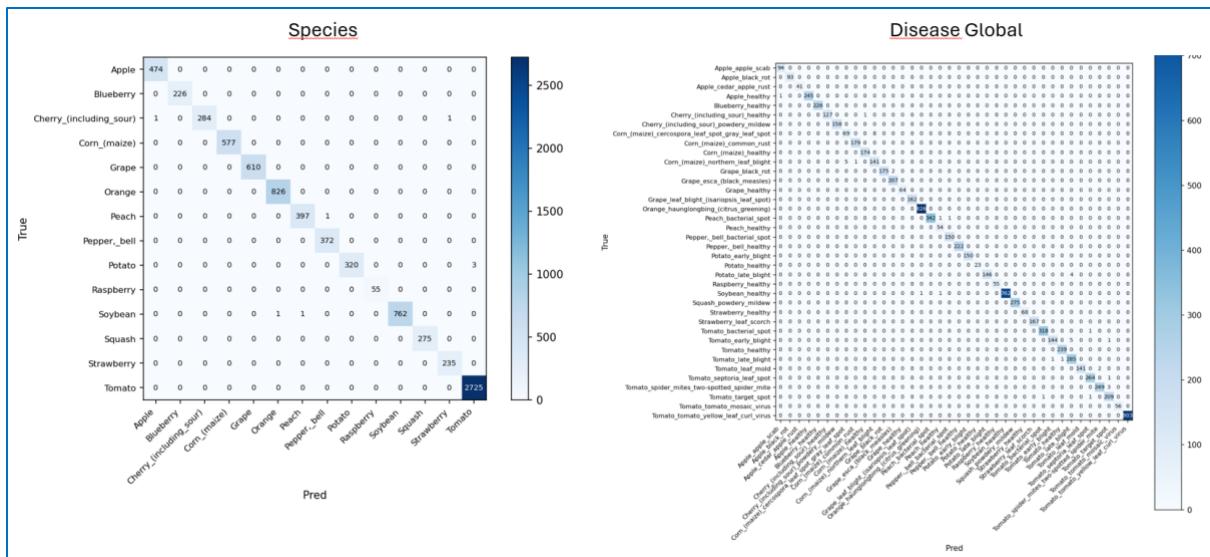
7.5. Matrice de confusion des modèles ML



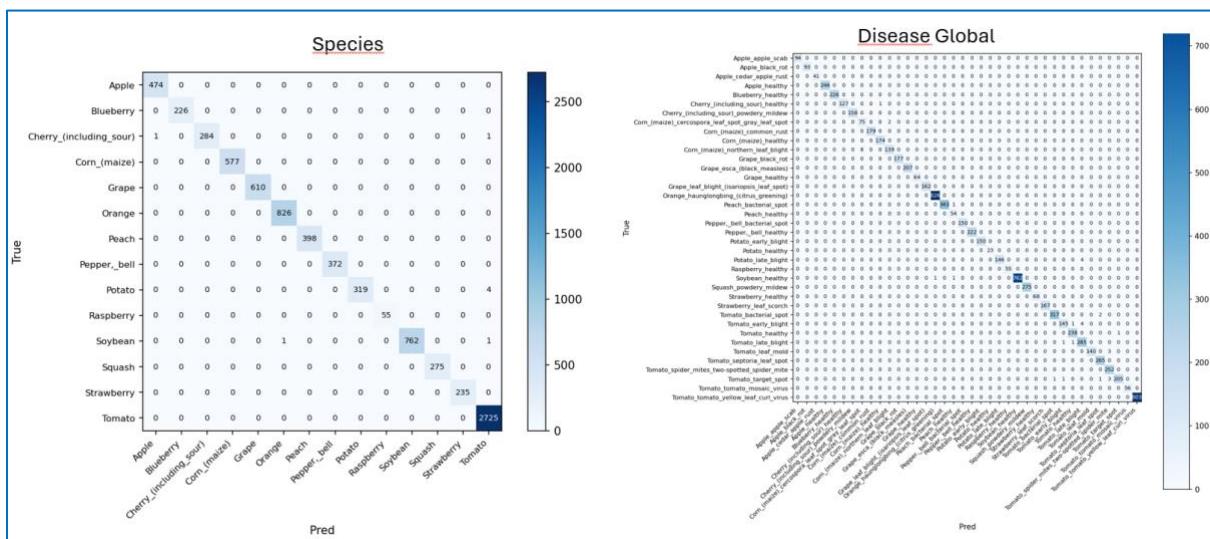
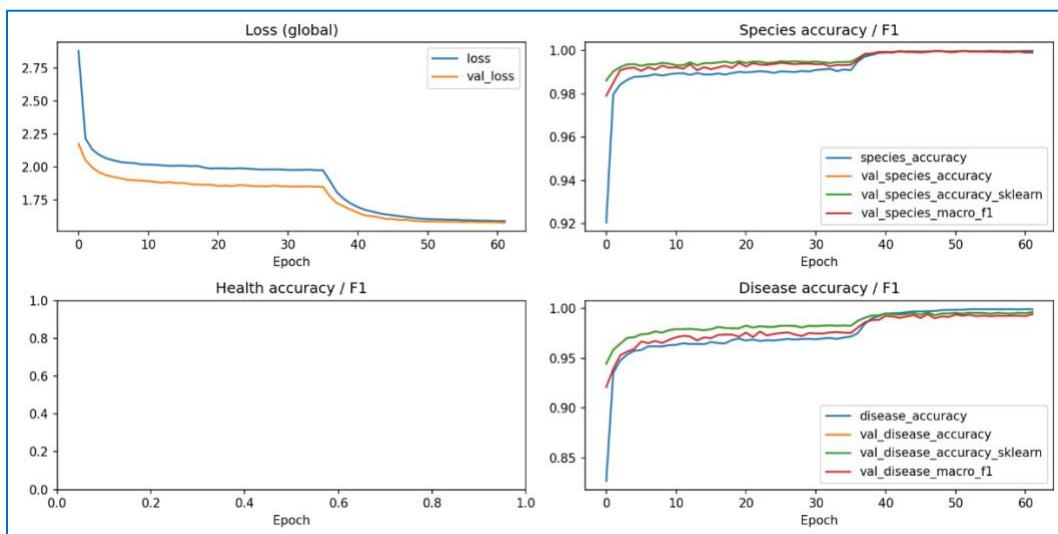
7.6. Métriques trackées des architectures DL

Architecture 1 :

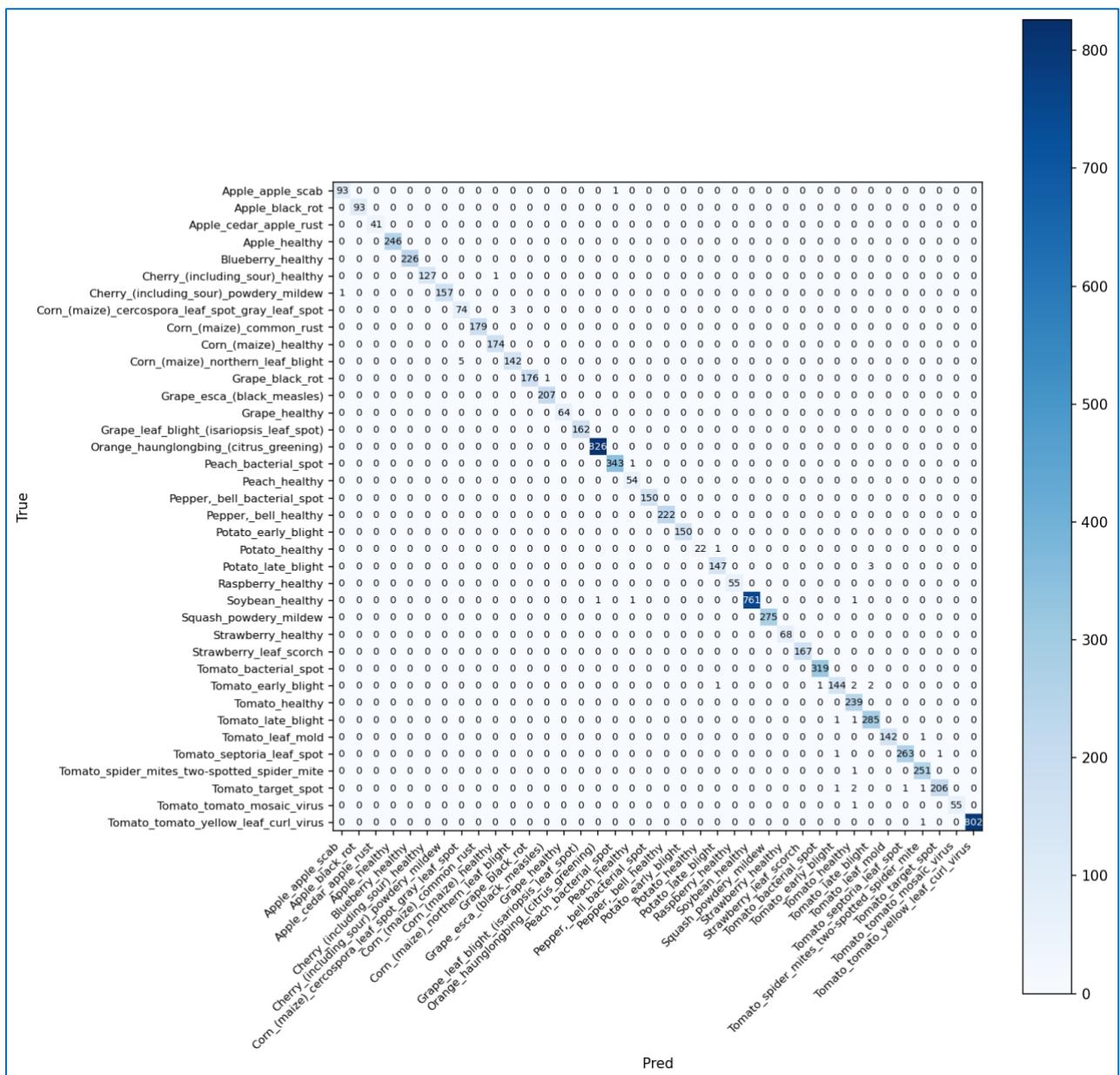
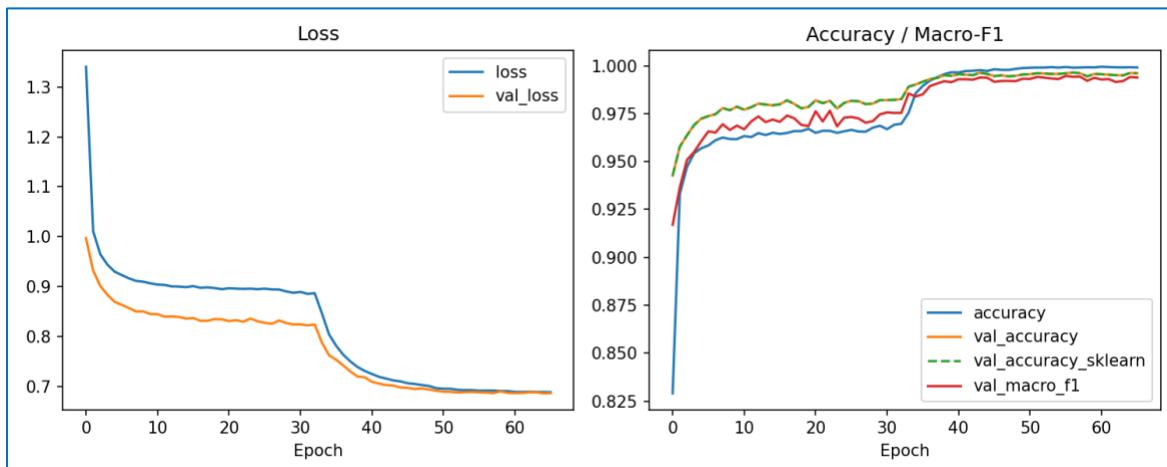




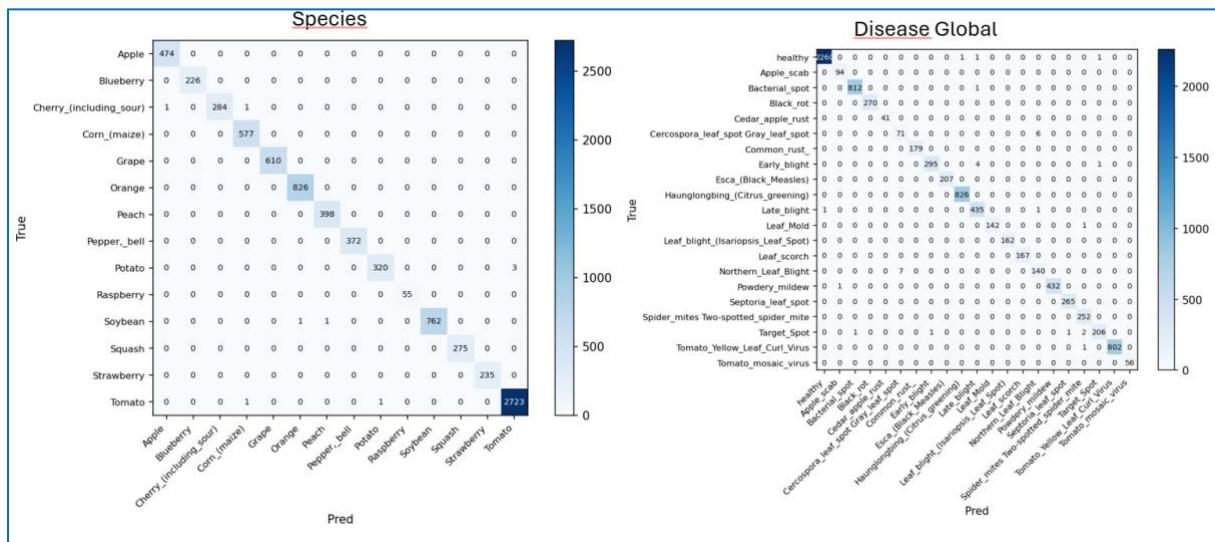
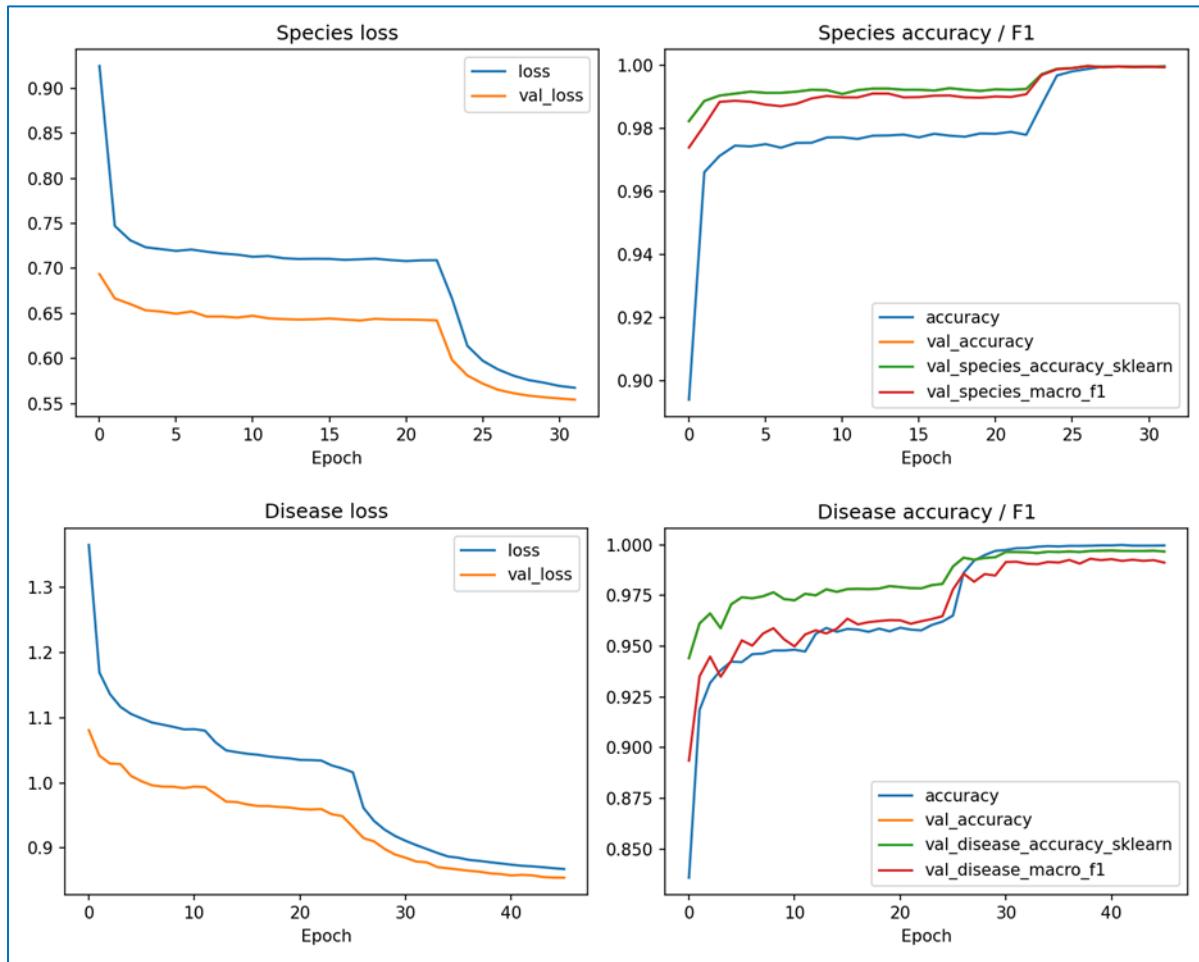
Architecture 2 :



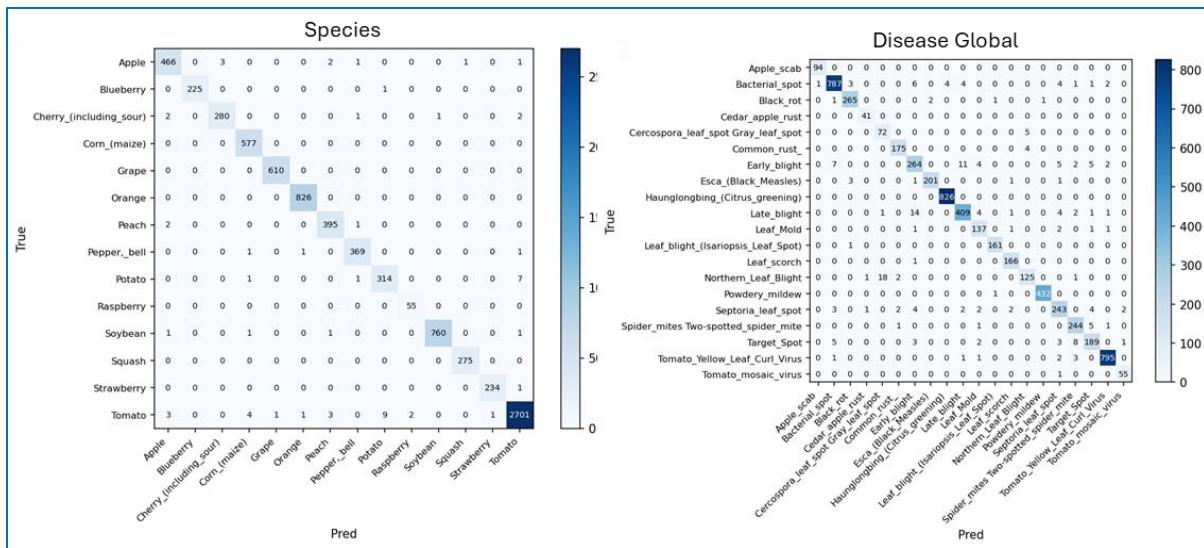
Architecture 3 :



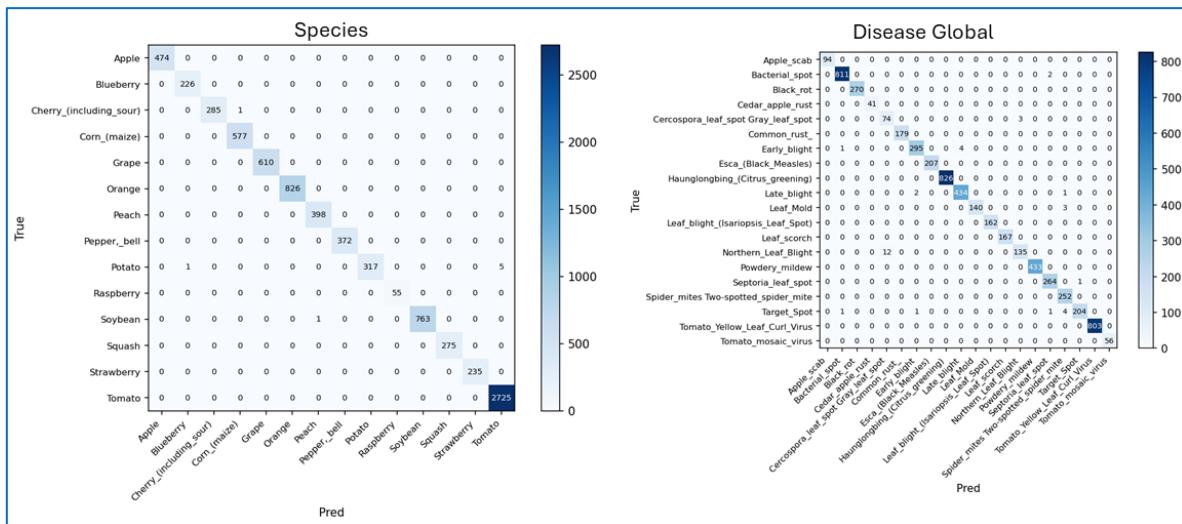
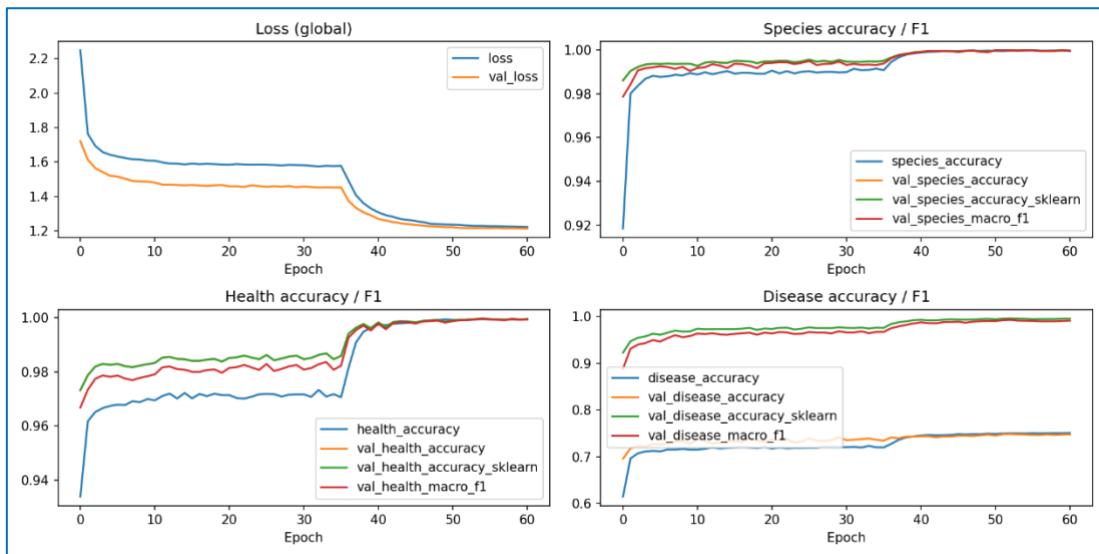
Architecture 4 :



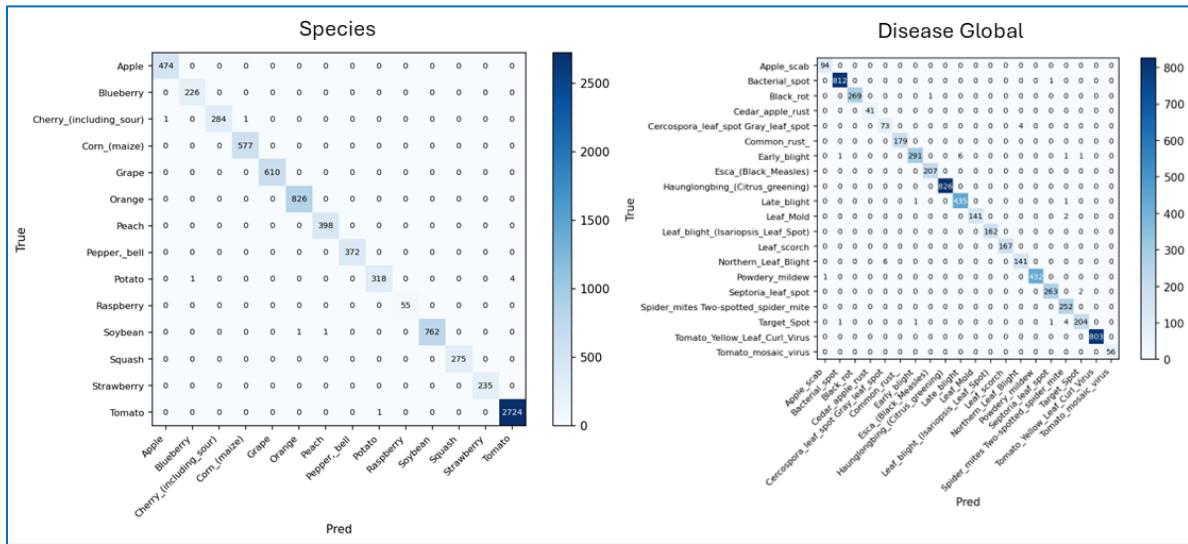
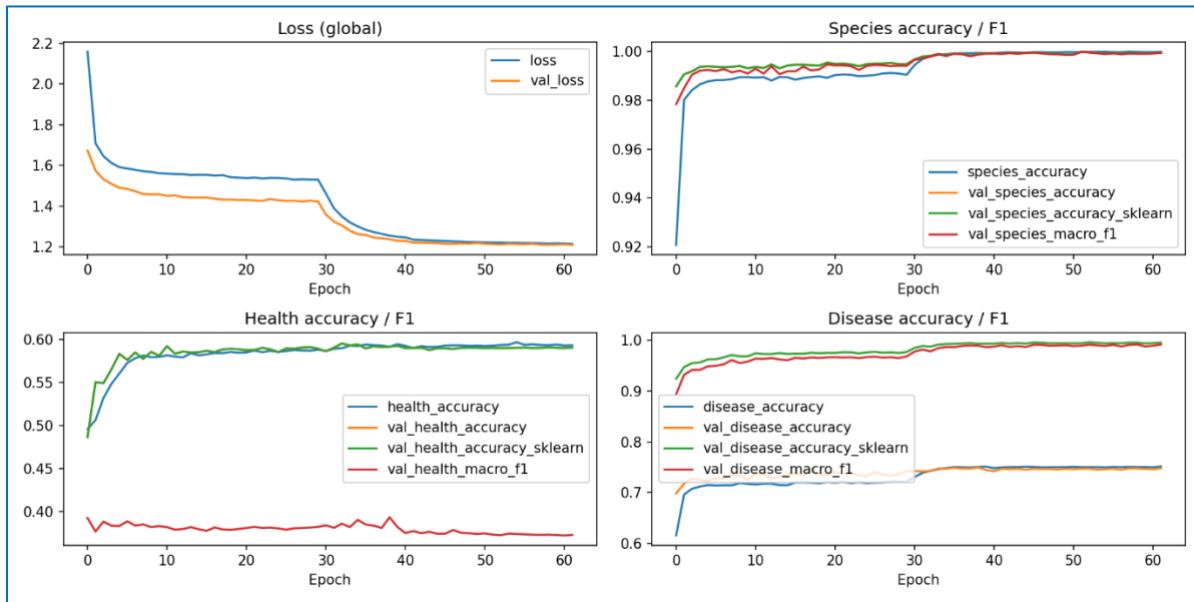
Architecture 5 :



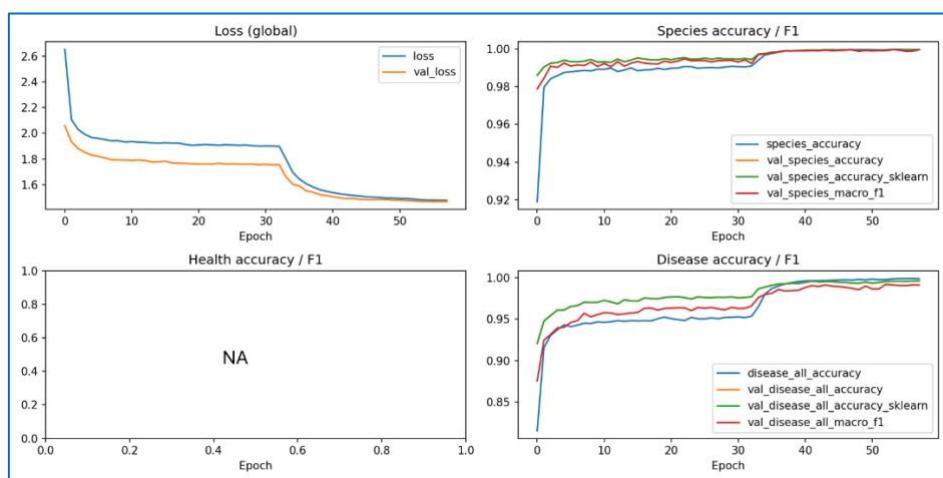
Architecture 6 :

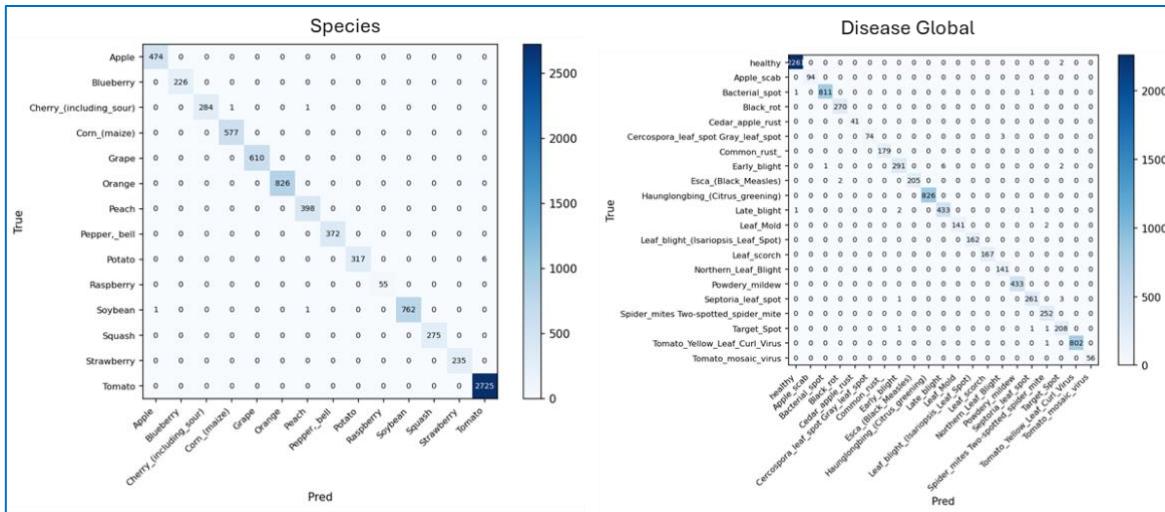


Architecture 7 :

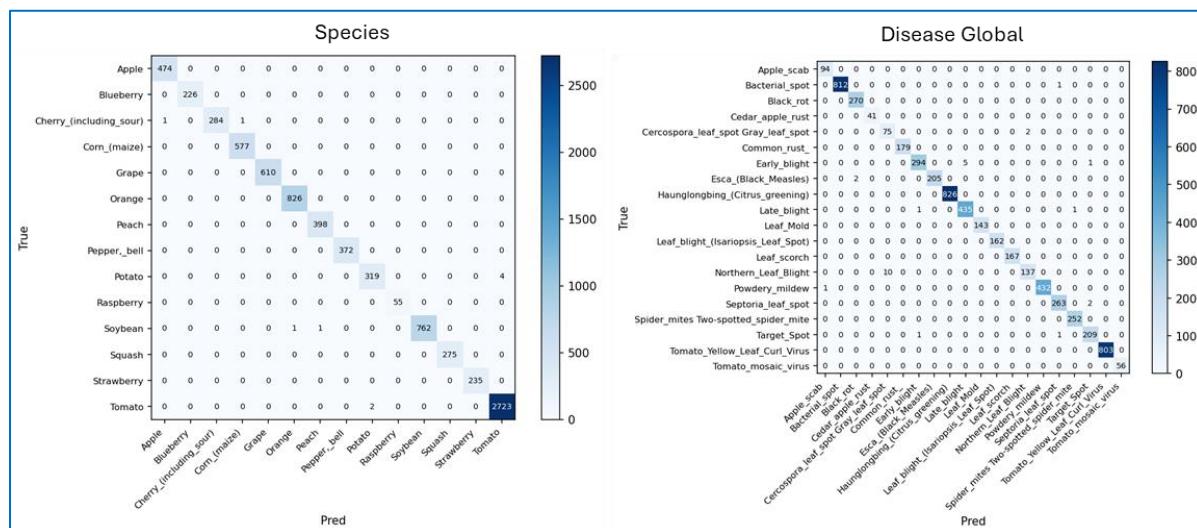
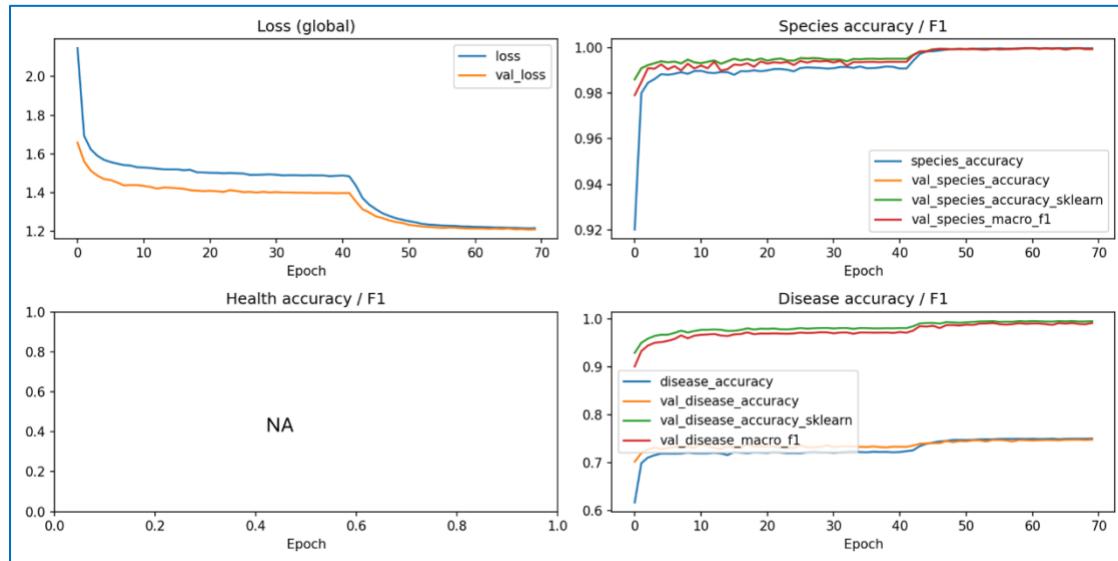


Architecture 8 :





Architecture 9 :



7.7. Caractéristiques des features

Les premières colonnes du tableau ne sont pas visibles car ce ne sont pas des features.

N° Col	Nom de la colonne	Description	Distribution des valeurs	Librairie / Source	Fonction
6	is_black	Flag indiquant si l'image est totalement noire (True ou Non (0))	Majorité à 0	OpenCV / NumPy	Si l'image est globalement noire ou indéfinissable (bruit, absence de données). Sert au filtrage.
7	dimension	Taille de la boîte englobante de la feuille	Continue, valeurs positives	OpenCV	Taille de l'image ou de la région d'intérêt.
8	aire	Surface de la feuille détectée via contour (pixel ²)	Corrélée à dimension	OpenCV	Surface occupée par la tache ou la plante détectée.
9	périmètre	Longueur du contour de la feuille (pixel)	Corrélée à aire	OpenCV	Longueur du contour de la forme détectée.
10	circularité	$4 \cdot \pi \cdot \text{aire} / (\text{périmètre}^2) : 1 = \text{parfait cercle}$ (Dimensionless)	Entre 0 et 1	OpenCV	Mesure de la ressemblance de la forme à un cercle : 1 pour un cercle parfait.
11	excentricité	Ratio excentricité de la région (0 = cercle, 1 = ligne) (Dimensionless)	Entre 0 et 1	OpenCV	Mesure de l'élongation (distance entre les foyers d'une ellipse).
12	aspect_ratio	Rapport largeur / hauteur de la boîte englobante (Dimensionless)	Valeurs > 0	OpenCV	Rapport largeur / hauteur de la forme ; indique si elle est étirée.
13	mean_R	Moyenne du canal Rouge de la feuille (Intensité normalisée (0–1))	Entre 0 et 1	OpenCV	Moyenne des intensités sur les canaux Rouge, Vert, Bleu. Donne la teinte dominante.
14	mean_G	Moyenne du canal Vert de la feuille (Intensité normalisée (0–1))	Entre 0 et 1	OpenCV	Moyenne des intensités sur les canaux Rouge, Vert, Bleu. Donne la teinte dominante.
15	mean_B	Moyenne du canal Bleu de la feuille (Intensité normalisée (0–1))	Entre 0 et 1	OpenCV	Moyenne des intensités sur les canaux Rouge, Vert, Bleu. Donne la teinte dominante.
16	std_R	Écart-type du canal Rouge (Intensité normalisée (0–1))	Entre 0 et 1	OpenCV / NumPy	Variation (écart-type) des couleurs — mesure la diversité colorimétrique.
17	std_G	Écart-type du canal Vert (Intensité normalisée (0–1))	Entre 0 et 1	OpenCV / NumPy	Variation (écart-type) des couleurs — mesure la diversité colorimétrique.
18	std_B	Écart-type du canal Bleu (Intensité normalisée (0–1))	Entre 0 et 1	OpenCV / NumPy	Variation (écart-type) des couleurs — mesure la diversité colorimétrique.
19	mean_H	Moyenne du canal Teinte (Hue) Dégrés (0–179)	0–179 degrés	OpenCV	Moyennes dans l'espace HSV (Teinte, Saturation, Valeur) : plus robustes aux variations de lumière.
20	mean_S	Moyenne du canal Saturation Intensité (0–255)	0–255	OpenCV	Moyennes dans l'espace HSV (Teinte, Saturation, Valeur) : plus robustes aux variations de lumière.
21	mean_V	Moyenne du canal Valeur (brightness) Intensité (0–255)	0–255	OpenCV	Moyennes dans l'espace HSV (Teinte, Saturation, Valeur) : plus robustes aux variations de lumière.
22	nettété	Variance du Laplacien (sharpness measure) Intensité ² (dimensionless)	Valeurs positives	OpenCV (Laplacian)	Mesure de la clarté des détails dans l'image (souvent calculée par des gradients ou Laplacien).
23	contrast	GLCM contrast : mesure de la variation locale de gris (Dimensionless)	Valeurs positives	skimage.feature.greycomatrix	Déférence locale d'intensité : élevé si beaucoup de variations entre pixels voisins.
24	energy	GLCM energy : somme des carrés de la matrice de cooccurrence (Dimensionless)	0–1	skimage.feature.greycomatrix	Homogénéité globale (somme des carrés des probabilités dans la GLCM).
25	homogeneity	GLCM homogeneity : voisinage uniforme vs varié (Dimensionless)	0–1	skimage.feature.greycomatrix	Inverse de la variance locale : élevé si les pixels voisins ont des valeurs proches.
26	dissimilarité	GLCM dissimilitude : somme des écarts pondérés (Dimensionless)	Valeurs positives	skimage.feature.greycomatrix	Mesure brute des écarts entre pixels voisins.
27	Correlation	GLCM correlation : dépendance linéaire des intensités voisines (Dimensionless)	-1 à 1	skimage.feature.greycomatrix	Corrélation entre intensités de pixels : détecte les motifs réguliers ou structurés.
28	contour_density	Densité du contour = périmètre / aire (Pixels ⁻¹) (dimensionless)	périmètre / aire	OpenCV	Densité des bords détectés ; élevé si beaucoup de transitions nettes (formes, nervures).
29	hu_1	7 Moments de Hu invariants (shape descriptors invariants) (Dimensionless)	Valeurs continues	OpenCV (Hu Moments)	Représente la variance (dispersion) globale. Sensible à la structure générale.
30	hu_2	7 Moments de Hu invariants (shape descriptors invariants) (Dimensionless)	Valeurs continues	OpenCV (Hu Moments)	Différencie les formes symétriques des asymétriques.
31	hu_3	7 Moments de Hu invariants (shape descriptors invariants) (Dimensionless)	Valeurs continues	OpenCV (Hu Moments)	Capte la régularité des contours (peut indiquer des anomalies locales).
32	hu_4	7 Moments de Hu invariants (shape descriptors invariants) (Dimensionless)	Valeurs continues	OpenCV (Hu Moments)	Sensible à l'asymétrie diagonale.
33	hu_5	7 Moments de Hu invariants (shape descriptors invariants) (Dimensionless)	Valeurs continues	OpenCV (Hu Moments)	Capte la torsion ou la déformation de la forme.
34	hu_6	7 Moments de Hu invariants (shape descriptors invariants) (Dimensionless)	Valeurs continues	OpenCV (Hu Moments)	Très sensible aux petites irrégularités — utile pour distinguer des textures fines.
35	hu_7	7 Moments de Hu invariants (shape descriptors invariants) (Dimensionless)	Valeurs continues	OpenCV (Hu Moments)	Combine plusieurs effets de symétrie ; utilisé pour distinguer des formes très similaires.
36	hog_mean	Moyenne des valeurs HOG (Histogramme des gradients orientés)	Valeurs positives	scikit-image (HOG)	Capture la variation locale des orientations de bords — reflète les textures et contours répétitifs.
37	hog_std	Écart-type des valeurs HOG	Valeurs positives	scikit-image (HOG)	Moyenne ou histogramme global des orientations — donne un résumé des formes présentes.
38	hog_entropy	Entropie des histogrammes HOG	Valeurs positives	scikit-image (HOG)	L'entropie des gradients (HOG) indique la diversité directionnelle — utile pour détecter des irrégularités.
39	fft_energy	Énergie spectrale de la transformée de Fourier	Valeurs positives	numpy.fft	Indique l'intensité globale des motifs périodiques — utile pour les textures régulières.
40	fft_entropy	Entropie spectrale (Transformée de Fourier)	Valeurs positives	numpy.fft	Mesurée par l'entropie du spectre — reflète la diversité des composantes fréquentielles.
41	fft_low_freq_power	Puissance dans la bande de basse fréquence (Transformée de Fourier)	Valeurs positives	numpy.fft (basses fréquences)	Concentration des basses fréquences — traduit les grandes structures de l'image.
42	fft_high_freq_power	Puissance dans la bande de haute fréquence (Transformée de Fourier)	Valeurs positives	numpy.fft (hautes fréquences)	Concentration des hautes fréquences — capte les contours nets et les détails fins.