



Projet Datascientest

Reconnaissance de plantes

Rapport final

Ela CIMEN
Yao N Guessan Roland KONAN
Yacine MADI SAID
Nicolas RINCÉ

09/06/2025

Table des matières

Partie 1 : Cadrage du projet.....	5
Contexte.....	5
Objectifs.....	5
Description des macro-objectifs.....	5
Localiser et classifier l'espèce d'une plante dans une image.....	5
Localiser et classifier la maladie éventuelle touchant la plante dans cette même image.....	5
Bâtir une application renvoyant les informations de ces classifications à l'utilisateur sur la base d'une photographie prise sur son smartphone.	6
Jalons du projet.....	6
Schéma de fonctionnement de l'application.....	6
Niveaux d'expertise	7
Niveaux d'expertise interne.....	7
Recours à expertise externe	7
Partie 2 : Exploration et DataViz.....	7
Jeux de données.....	7
Soumis et exploités	7
Soumis et non exploités	8
Constats.....	8
Pertinence.....	9
Variables	9
Description.....	10
Limitation des données.....	10
Partie 3 : Pré-Processing et feature engineering.....	11
Création du dataframe des caractéristiques.....	11
Nettoyage des données et étapes de traitement	11
Transformations des données.....	12
Traitement des images.....	12
Visualisations et Statistiques.....	14
Corrélation entre variables du dataframe.....	14
Relations avec les variables cibles	14
Distribution des données	14
Distribution des espèces dans le dataframe "high quality".....	14
Distribution des espèces dans le dataframe "low quality"	15
Mesure d'amélioration de la qualité des images.....	15
Analyses statistiques.....	17
Conclusions pré-modélisation	18
Partie 4 : Modélisation	18
Classification du problème.....	18
Type de problème de machine learning	18
Tâches de machine learning.....	18

Métriques de performance utilisées pour comparer les modèles	18
Métrique principale.....	18
Métriques secondaires.....	18
Choix du modèle et optimisation	19
Algorithmes testés	19
Résultats par modèle	19
1- KERAS	19
2- FastAI.....	23
3- Torch	24
4- AlexNet.....	26
Optimisation	29
Phase 1	29
Phase 2	35
Stratégies de traitement des images Cassava	41
1. Augmentation massive des classes minoritaires.....	41
2. Filtrage basé sur un score multi-critères	42
3. Relabellisation automatique basée sur le score.....	42
4. Fine-tuning des modèles sur les nouveaux datasets	43
5. Résultats obtenus.....	43
6. Analyse des résultats.....	44
7. Conclusion de l'étude d'amélioration sur Cassava	45
Étude de l'ensemblage des modèles.....	46
Conclusion sur le choix des ensembles de modèles.....	49
Partie 5 : Conclusions tirées	49
Difficultés rencontrées pendant le projet.....	49
Principal verrou scientifique rencontré.....	49
Difficultés détaillées par catégorie	49
Prévisionnel	49
Jeux de données.....	50
Compétences techniques/théoriques	50
Pertinence	50
IT	50
Autres.....	51
Bilan.....	51
Contribution principale dans l'atteinte des objectifs.....	51
Modifications depuis la dernière itération	51
Résultats obtenus et comparaison au benchmark.....	51
Performances atteintes	51
Comparaison aux benchmarks de la littérature	51
Atteinte des objectifs du projet.....	52
Objectif 1 : Classification des espèces (✓ Atteint).....	52
Objectif 2 : Détection des maladies (⚠ Partiellement atteint).....	52
Objectif 3 : Application Streamlit (✓ En cours)	52
Processus métier d'inscription	52
Agriculture de précision	52
Services aux professionnels.....	52
Applications grand public	52
Suite du projet	52

Pistes d'amélioration des performances	52
Amélioration des données.....	52
Optimisations architecturales	53
Techniques avancées.....	53
Déploiement et optimisation	53
Contribution à l'accroissement de la connaissance scientifique.....	53
Avancées méthodologiques.....	53
Connaissances domaine-spécifiques	53
Impact scientifique et sociétal	54

Partie 1 : Cadrage du projet

Contexte

La **cible du projet** est d'**offrir** à un individu, ne bénéficiant pas de l'expertise pour reconnaître une plante et sa maladie éventuelle, la **capacité** de l'identifier via l'appareil photo de son smartphone

- **Du point de vue technique**, il s'agit d'utiliser des algorithmes de machine learning, pour analyser les images et classifier les espèces de plantes, détecter les maladies et réaliser des analyses statistiques afin de déterminer, le cas échéant, les corrélations entre espèces de plantes et maladies associées.
- **Du point de vue économique**, l'application construite pourra aider un certain nombre d'acteurs (agriculteurs, distributeurs de plantes, paysagiste, ...) à détecter précocement les maladies et éviter des pertes de récoltes ou de chiffre d'affaires, et minimiser les coûts associés aux traitements tardifs.
- **Du point de vue scientifique**, l'utilisation de cette application pourrait offrir un support durable au recensement des espèces, renforcer l'échange de connaissances entre chercheurs, en particulier sur le maintien de la bio-diversité, sensibiliser écoliers, étudiants et grand public à la connaissance des espèces et leur conservation, être élargi à la recherche des causes des maladies à l'aide de données additionnelles (géographie, climat, ...)

Objectifs

Description des macro-objectifs

Localiser et classifier l'espèce d'une plante dans une image

La photographie initiale prise par l'utilisateur sera analysée au regard de critères d'acceptabilité comparables aux conditions d'entraînement de l'algorithme de classification (niveau de flou, luminosité, contraste, etc). Elle sera ensuite retraitée (format JPEG, RGB, redimensionnement et éventuellement retraitement du niveau de flou afin de faciliter le diagnostic). Ce premier algorithme de classification, entraîné sur la base totale des images décrite en page 14 et de métadonnées pouvant y être rattachées aura été testé sur une base constituée de 20% des images concernées avec un objectif d'accuracy de 95%. La base totale des images aura été probablement corrigée afin de minimiser les déséquilibres entre classes.

Localiser et classifier la maladie éventuelle touchant la plante dans cette même image

La photographie retraitée, ou la photographie d'origine subissant de nouveaux filtres, sera soumise à un second algorithme de classification, entraîné sur la base totale des images décrite en page 14 et de méta-données pouvant y être rattachées (telles que le niveau de bleu dans l'image qui est un premier prédicteur du caractère malade ou non de la plante) et testé sur une base constituée de 20% des images concernées avec un objectif d'accuracy de 95% des feuilles malades.

Bâtir une application renvoyant les informations de ces classifications à l'utilisateur sur la base d'une photographie prise sur son smartphone.

L'application sera développée sur Streamlit conformément au schéma de l'application présentée en Page 6 en veillant à des règles d'ergonomie de « base » (éviter les images de fond, contraste des couleurs pour faciliter la lecture, usage des couleurs limité à une gamme réduite, boutons de navigation explicites, etc.)

Jalons du projet

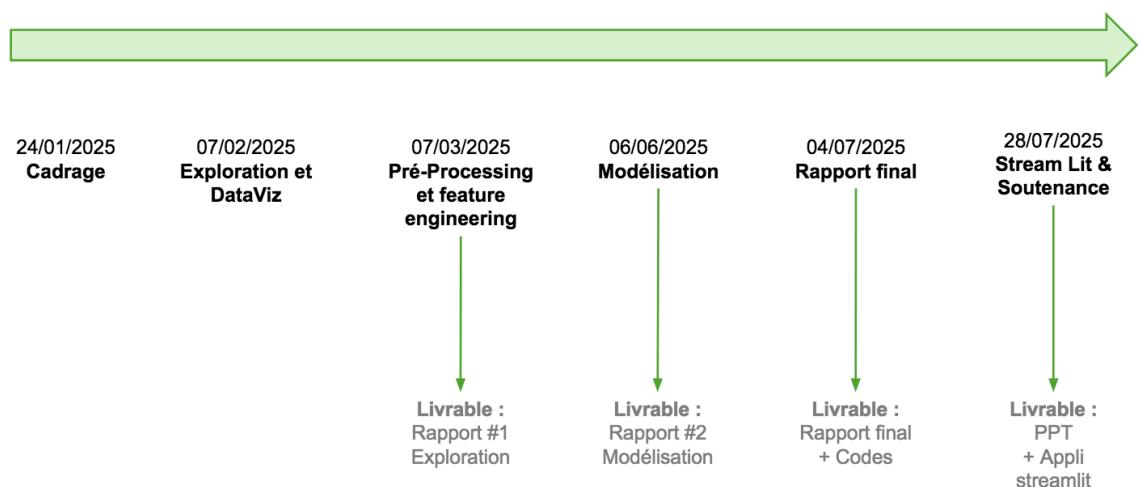
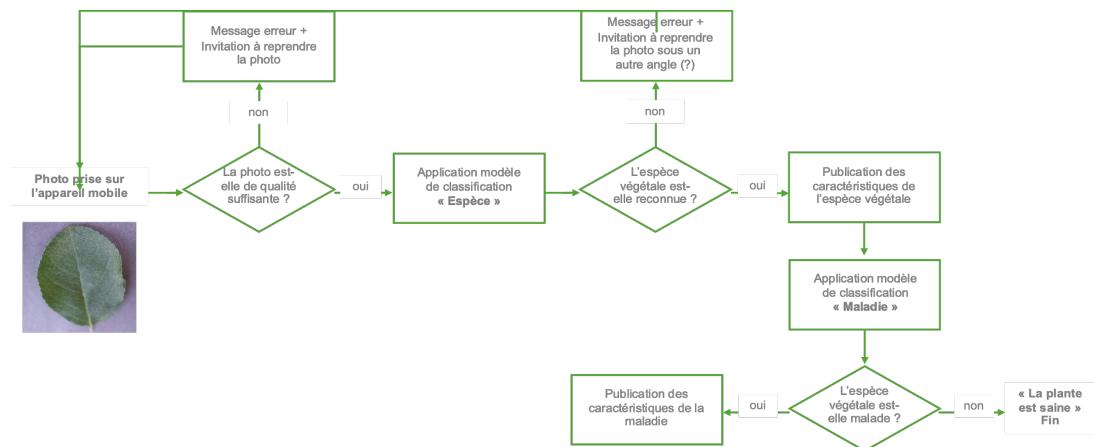


Schéma de fonctionnement de l'application



Niveaux d'expertise

Niveaux d'expertise interne

Nom	Algorithmie	Dév. Appli mobile	Statistiques	Gestion projet	Botanique
Ela CIMEN	2/5	0/5	3/5	4/5	3/5
Roland KONAN	4/5	2/5	3/5	4/5	2/5
Yacine MADI SAID	3/5	5/5	2/5	4/5	1/5
Nicolas RINCÉ	2/5	0/5	4/5	4/5	1/5

Recours à expertise externe

Nous n'avons pas eu recours à des experts externes à proprement parler (en-dehors de Damien évidemment !) mais avons réalisé de nombreuses recherches afin de nous guider, notamment sur les sujets suivants :

- Taille des images standard utilisée dans les algorithmes de reconnaissance d'image ;
- Détermination du niveau de flou d'une image et seuil d'acceptabilité pour la reconnaissance d'images (150 de Laplacienne) ;
- Retraitements des niveaux de flous ;
- Déséquilibre acceptable entre classes sur le fichier d'entraînement (1 à 10) ;

Nombre minimal d'images à soumettre à l'algorithme par classe (1.000).

Partie 2 : Exploration et DataViz

Jeux de données

Soumis et exploités

<https://www.kaggle.com/vbookshelf/v2-plant-seedlings-dataset>

- Arborescence de 12 dossiers pour 12 espèces de jeune pousse les plus répandue au Danemark
- Il n'y a qu'un seul dataset (non séparé en test/train) contenant 5539 images
- Volumétries restreintes aux images d'espèces végétales : 100%

<https://www.kaggle.com/vipoooool/new-plant-diseases-dataset>

- Fichier d'images de feuilles d'arbres fruitiers, de légumineuses et de céréales, malades ou saines
- Dataset d'entraînement de 70.295 images
- Dataset de test de 17.572 images
- Volumétries restreintes aux images d'espèces végétales : 100%

<https://www.kaggle.com/saroz014/plant-disease>

- Fichier d'images de feuilles d'arbres fruitiers, de légumineuses et de céréales, malades ou saines
- Dataset d'entraînement de 43.456 images
- Dataset de test de 10.849 images
- Volumétries restreintes aux images d'espèces végétales : 100%

<https://www.kaggle.com/abdallahhalidev/plantvillage-dataset>

- Fichier d'images de feuilles d'arbres fruitiers, de légumineuses et de céréales, malades ou saines
- 3 Datasets de 54305 images chacun avec les mêmes images en couleur, niveau de gris et prétraité.
- Volumétries restreintes aux images d'espèces végétales : 100%

Soumis et non exploités

<https://storage.googleapis.com/openimages/web/download.html> et

<https://cocodataset.org/#home>

- Non exploité car bases de données non spécifiquement axées sur les plantes, de volumétrie limitée sur cette dernière catégorie, sans discrimination entre plantes saines et plantes malades, dont le seul intérêt était la présence de masques qui ne serviront pas dans les algorithmes choisis.

Constats

- Les 3 dernières bases de données soumises et exploitées se sont révélées redondantes et provenant d'un même set d'images initial. Nous n'avons donc conservé que la base d'origine en fusionnant les données d'entraînement et de test (<https://www.kaggle.com/vipoooool/new-plant-diseases-dataset>)
- La première base nous est apparue trop limitée en nombre d'images (moins de 500 images par espèces), peu exploitable en raison de la part de l'image réservée au végétal (il s'agit de jeunes pousses) et trop spécifique (12 espèces provenant du Danemark). Nous l'avons donc abandonnée.
- Compte tenu de la limitation du dataset à des arbres fruitiers et de céréales et de légumineuses, nous avons recherché d'autres datasets permettant d'étendre la reconnaissance à d'autres plantes (canne à sucre, maïs et vigne) comme exposé dans le tableau ci-dessous. Comme pour le reste des datasets, ces images sont libres de droits.

<https://www.kaggle.com/datasets/nirmalsankalana/sugarcane-leaf-disease-dataset>

- Dataset d'images de feuilles de canne à sucre saine et malades
- 2569 images, séparées en 5 répertoires (1 par maladie et 1 pour celles en bonne santé)
- Volumétries restreintes aux images d'espèces végétales : 100%

<https://www.kaggle.com/datasets/nirmalsankalana/grape400-dataset>

- Datasets d'images de feuilles de vigne saines et malades
- 1600 images séparées en 4 répertoires (1 par maladie et 1 pour celles en bonne santé)
- Volumétries restreintes aux images d'espèces végétales : 100%

<https://www.kaggle.com/datasets/shuvokumarbasak4004/rose-leaf-disease-dataset>

- Datasets d'images de feuilles de rose saines et malades
- 14,910 images séparées en 3 répertoires (test, train, validation)
- Volumétries restreintes aux images d'espèces végétales : 100%

<https://www.kaggle.com/datasets/nirmalsankalana/cassava-leaf-disease-classification>

- Datasets d'images de feuilles de cassave saines et malades
- 21 397 images séparées en 5 répertoires (1 par maladie et 1 pour celles en bonne santé)
- Volumétries restreintes aux images d'espèces végétales : 100%

Pertinence

Variables

- La reconnaissance des plantes, puis du caractère sain ou malade, va s'appuyer sur les images sélectionnées. Le sujet de la pertinence des variables ne s'applique donc pas de la même manière que sur un problème de classification basé sur un fichier de données textuelles ou numériques. Les principales variables dans ce cadre seront les pixels de chaque image.
- Néanmoins, la sélection des images s'est appuyée dans une première phase sur un fichier .csv répertoriant les principales caractéristiques des images (cf. section suivante sur le pre-processing) : niveau de flou, luminosité, contraste, taille des images, type de l'image (.jpeg, .png, ...).
- C'est dans ce même fichier que nous avons logé les deux variables cibles : le nom de la plante, d'une part, le caractère sain ou malade d'autre part.
- Un troisième fichier de données devra être constitué pour abriter les données descriptives à afficher sur l'application streamlit.

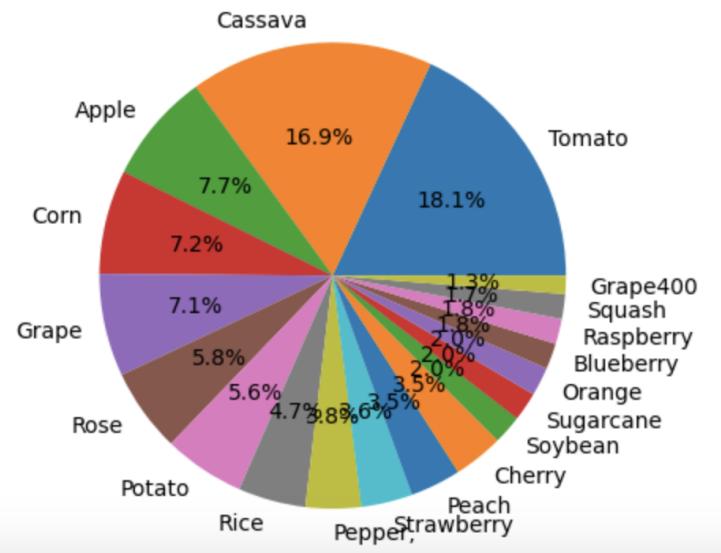
Description

Le dataframe global contient 126 668 images pour 19 espèces.

Distribution des especes Data frame global

	count	percentage
Species		
Tomato	22930	18.1%
Cassava	21397	16.89%
Apple	9714	7.67%
Corn	9145	7.22%
Grape	9027	7.13%
Rose	7351	5.8%
Potato	7128	5.63%
Rice	5932	4.68%
Pepper,	4876	3.85%
Strawberry	4498	3.55%
Peach	4457	3.52%
Cherry	4386	3.46%
Soybean	2527	1.99%
Sugarcane	2521	1.99%
Orange	2513	1.98%
Blueberry	2270	1.79%
Raspberry	2226	1.76%
Squash	2170	1.71%
Grape400	1600	1.26%

Distribution des especes Image dataset global



Limitation des données

- **Limitations liées au périmètre des données :** la première limitation concerne le périmètre des plantes et végétaux présents dans la base. Dès lors, l'application ne pourra pas être utilisée pour reconnaître « toutes » les espèces de plantes. La généralisation de l'application ne pourrait venir que de l'élargissement du périmètre des plantes concernées.
- **Limitations liées à l'hétérogénéité des données :** la seconde limitation concerne l'hétérogénéité des données étudiées, qui a mené à l'exclusion des images de jeunes pousses trop éloignées de la majorité de la base d'images que nous avons pu constituer, et insuffisamment nombreuses pour les intégrer dans le jeu d'entraînement. De même, cela limitera la portée de l'application.

- **Limitations liées au déséquilibre des classes :** la troisième limitation concerne le déséquilibre entre plantes avec une surreprésentation de certaines espèces dont en particulier les tomates.
- **Limitations liées à la qualité des données :** la quatrième limitation touche la qualité des données parmi les images retenues, et notamment leur degré de flou qui est primordial dans la reconnaissance des images
- **Limitations liées à la taille du dataset :** la cinquième limitation concerne la taille de l'ensemble des images ainsi constituées qui doit être limitée afin de tenir compte des temps de traitement de l'algorithme.

Partie 3 : Pré-Processing et feature engineering

Création du dataframe des caractéristiques

Afin de faciliter la sélection des images nous avons créé un dataframe contenant les caractéristiques de l'ensemble des images issues des dataset qui présente la structure suivante :

- FileName : nom du fichier,
- FilePath : chemin vers le dataset initial,
- Extension : extension de l'image
- Species : espèce,
- Disease : maladie,
- FileSize : taille de l'image,
- Width : largeur de l'image,
- Height : hauteur de l'image,
- Contrast : moyenne du contraste
- Luminosity: moyenne de la luminosité
- RedMean: moyenne de vert
- GreenMean : moyenne de vert
- BlueMean : moyenne de bleu
- Blur : moyenne de flou

Nettoyage des données et étapes de traitement

Définition des critères de sélection :

- Flou > 500
- Contraste > 2
- Luminosité < 200
- Nombre maximum d'image = 12500
- nombre minimum d'image = 1250

A partir du dataframe initial, on parcourt l'ensemble des images grâce au FilePath pour appliquer nos critères de qualité et créer un dataframe (**df_high_quality**) réunissant uniquement les images qui passent le test.

Ceux ne passant pas les critères de qualité vont être intégré à un dataframe différent (**df_low_quality**) pour être traité.

Transformations des données

Traitement des images

Plusieurs traitements sont appliqués aux images dans le but d'améliorer leur qualité. Les traitements incluent des ajustements sur la luminosité, le contraste, la netteté (flou et netteté), ainsi que des redimensionnements.

Les étapes sont détaillées ci-dessous :

1. Ajustement de la luminosité

La fonction `adjust_brightness(image)` permet de modifier la luminosité d'une image en utilisant la méthode `ImageEnhance.Brightness` de la bibliothèque `PIL`. Le facteur de luminosité, défini par `brightness_factor`, est appliqué à l'image. Après ajustement, la luminosité moyenne de l'image (calculée sur une conversion en niveaux de gris) est renvoyée pour référence.

2. Augmentation du contraste

La fonction `process_contrast(image)` augmente le contraste de l'image à l'aide de `ImageEnhance.Contrast`. Le facteur de contraste, défini par `contrast_factor`, est appliqué pour rendre les différences entre les pixels plus marquées. Après traitement, la fonction calcule et renvoie la variance de l'image en niveaux de gris (une mesure du contraste).

3. Calcul du flou (Netteté)

La fonction `calculate_blurriness(image)` utilise la méthode du Laplacien pour calculer le flou d'une image. Le flou est mesuré par la variance de la matrice du Laplacien sur l'image en niveaux de gris. Une faible variance indique un flou plus élevé, tandis qu'une haute variance indique une image plus nette.

4. Augmentation de la netteté

La fonction `increase_sharpness(image)` est utilisée pour améliorer la netteté de l'image. Cela se fait par l'application de la méthode `ImageEnhance.Sharpness`. Un facteur de netteté, `sharpness_factor`, est défini pour contrôler l'intensité de l'amélioration. Après avoir ajusté la netteté, la fonction calcule et renvoie également le niveau de flou après traitement pour évaluer l'effet de la netteté appliquée.

5. Redimensionnement de l'image

La fonction `resize_image(image, width, height)` redimensionne les images à une taille spécifiée (ici, 256x256 pixels). Cette opération est réalisée avec la méthode `resize()` de PIL en utilisant l'algorithme `LANCZOS`, qui permet de préserver la qualité de l'image lors du redimensionnement.

6. Configuration des paramètres globaux

Les facteurs de traitement sont définis avant le traitement :

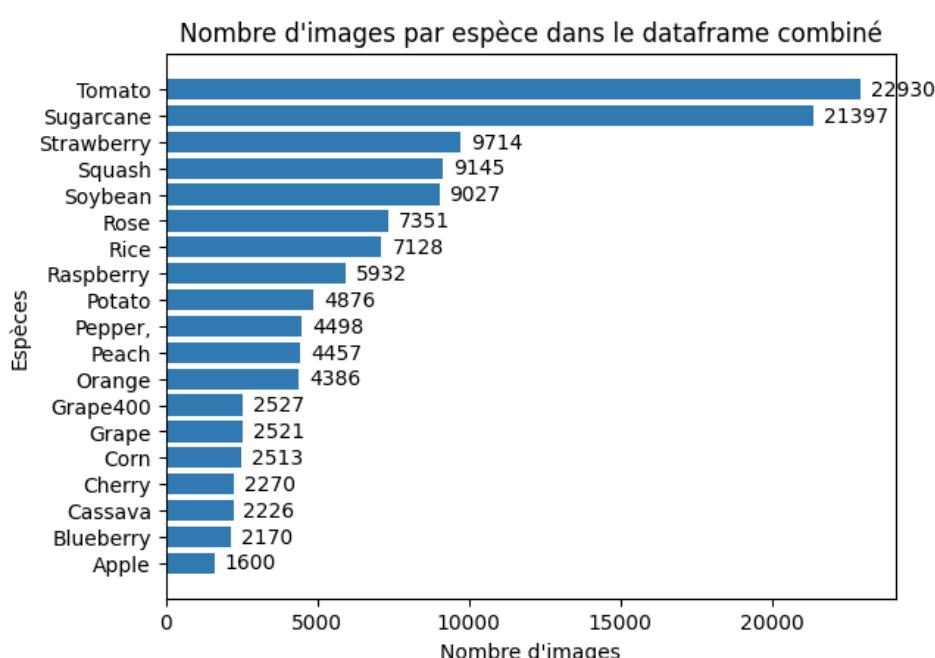
- `brightness_factor` : Ajuste la luminosité (ici fixé à 0.8 pour une légère réduction de la luminosité).
- `contrast_factor` : Augmente le contraste (ici fixé à 1.5 pour un contraste plus élevé).
- `sharpness_factor` : Améliore la netteté (ici fixé à 4.0 pour une forte amélioration de la netteté).
- Taille de redimensionnement : Les images sont redimensionnées à 256x256 pixels pour une normalisation de la taille.

7. Exécution et enregistrement des résultats

Une fois les images traitées, le DataFrame Low_quality_processed contenant les informations des images traitées est enregistré dans un fichier CSV. Ce fichier peut être utilisé pour des analyses ultérieures ou pour vérifier les modifications apportées à chaque image.

8. Combinaison dans un dataframe global

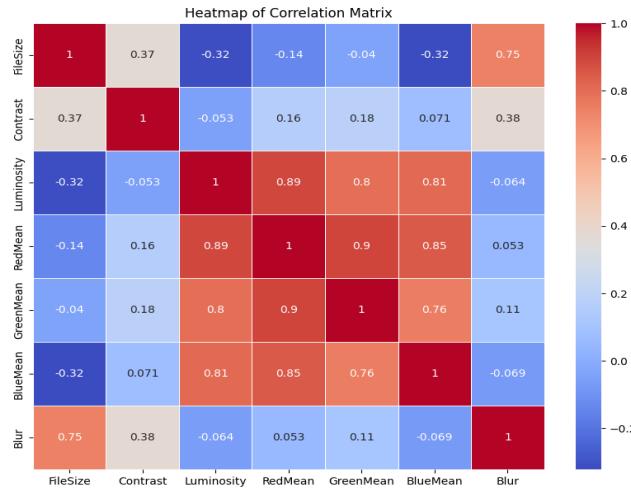
Les dataframes High_quality et Low_quality_processed sont combinés dans un nouveau dataframe df_recombined_quality.



Visualisations et Statistiques

Corrélation entre variables du dataframe

On constate que les variables de luminosité et de niveau de rouge, vert et bleu sont très bien corrélées. En revanche, à l'exception du couple ("Blur", "Filesize"), les autres variables sont peu corrélées entre elles.



Relations avec les variables cibles

Nous n'avons pas identifié de relations entre la variable 'species' et le reste des variables. En revanche, en première approche, nous avons noté une corrélation entre le niveau de "jaune" des images et la variable saine ou malade sur une partie du jeu de données (cf. Analyse statistiques).

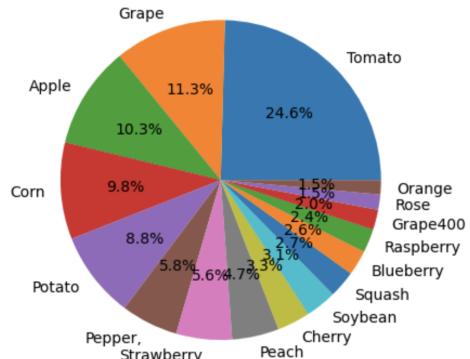
Distribution des données

Distribution des espèces dans le dataframe "high quality"

Distribution des especes Data frame high quality

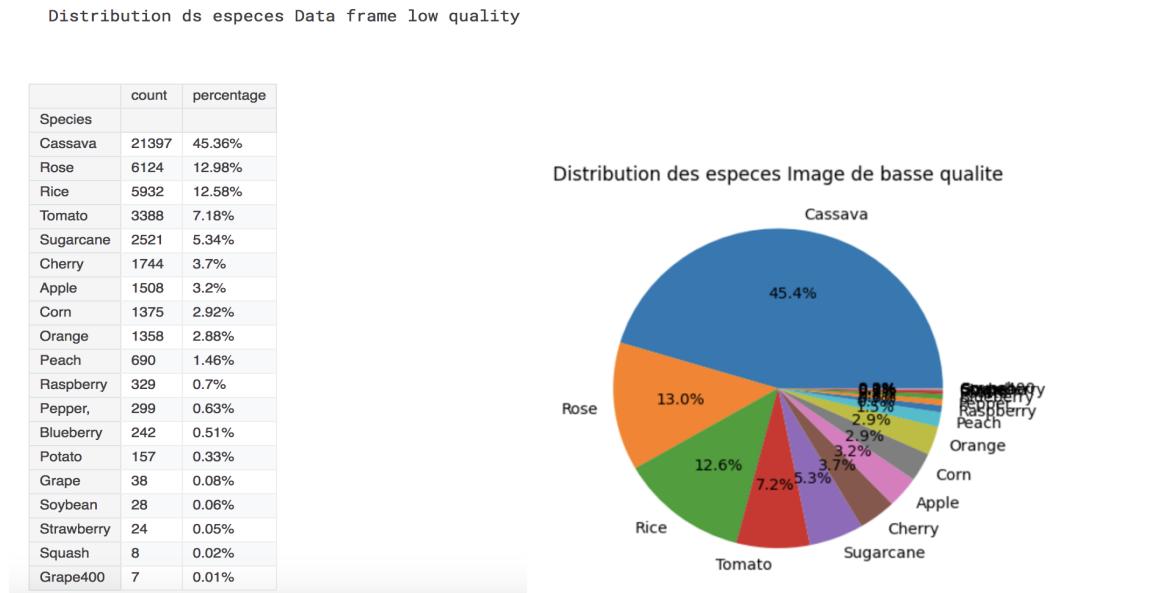
	count	percentage
Species		
Tomato	19542	24.58%
Grape	8989	11.31%
Apple	8206	10.32%
Corn	7770	9.77%
Potato	6971	8.77%
Pepper,	4577	5.76%
Strawberry	4474	5.63%
Peach	3767	4.74%
Cherry	2642	3.32%
Soybean	2499	3.14%
Squash	2162	2.72%
Blueberry	2028	2.55%
Raspberry	1897	2.39%
Grape400	1593	2.0%
Rose	1227	1.54%
Orange	1155	1.45%

Distribution des especes Image de bonne qualite



Dans le dataframe des images de bonnes qualités, il y a 79 599 images, dont les trois premières espèces représentent + 45% du total.

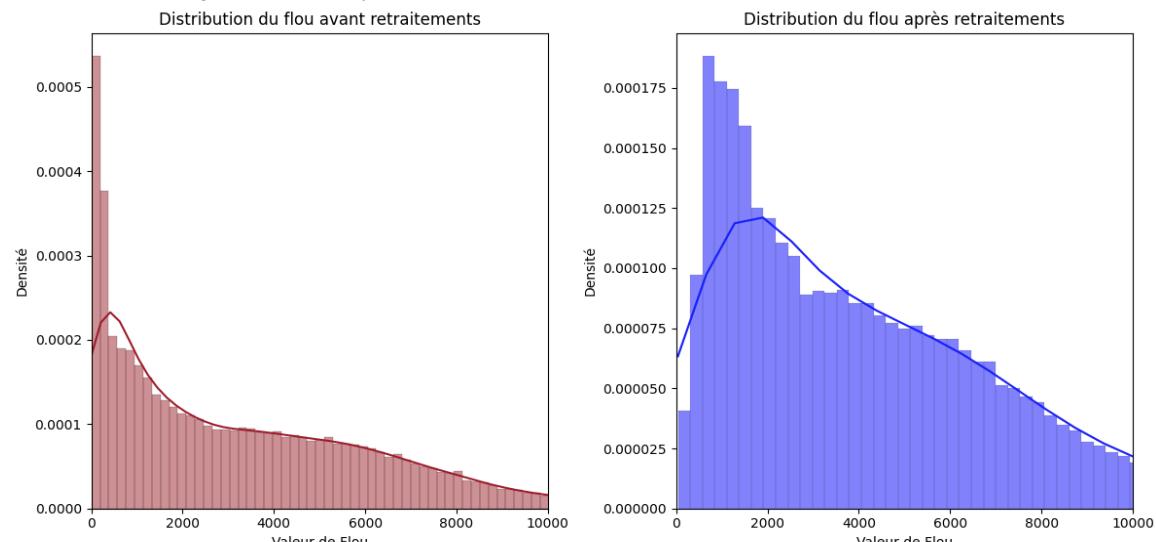
Distribution des espèces dans le dataframe “low quality”



Il y a 47 069 images, on peut remarquer que l'ensemble des images des espèces cassave, riz, et sucre de canne ne sont présentes que sur le dataframe de mauvaise qualité.

Mesure d'amélioration de la qualité des images

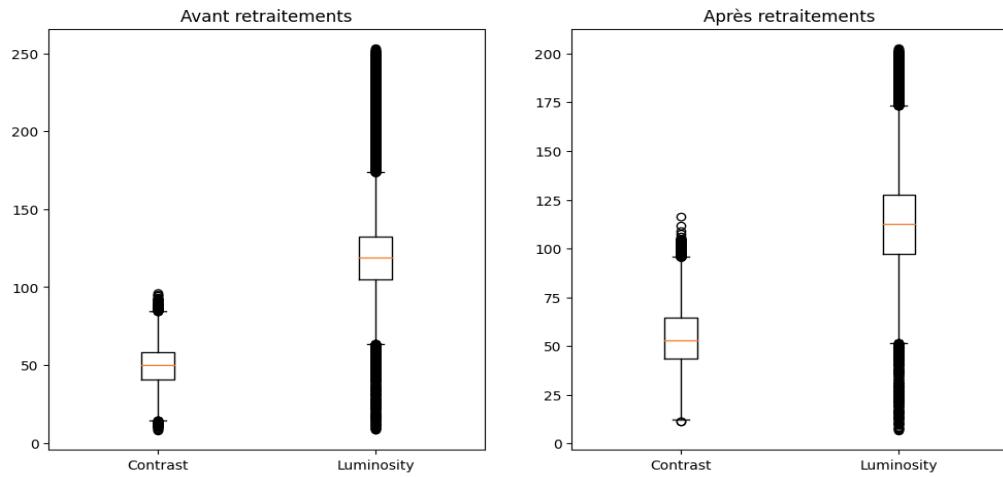
Distribution du flou avant/après retraitements



D'après la visualisation, la densité des valeurs de flou a augmenté après traitement, ce qui suggère une amélioration de la qualité des images. En effet, on peut observer que le pic de l'histogramme a migré de **500 avant le traitement à 2000 après le traitement**, ce qui indique

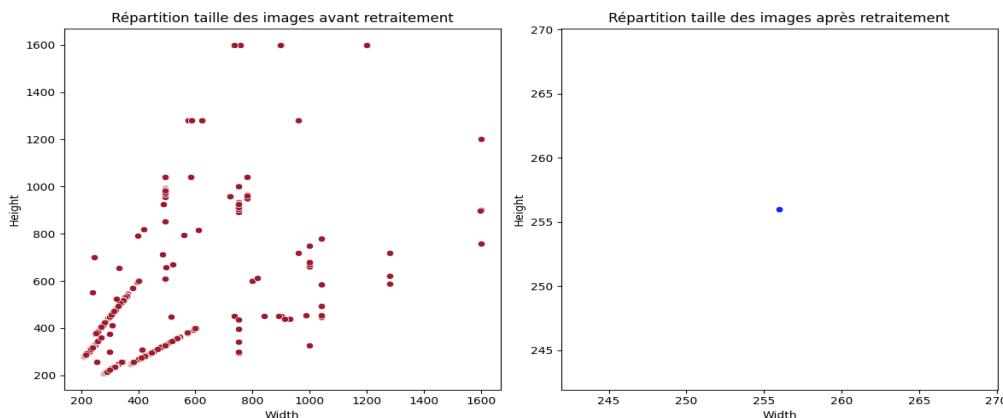
que les images sont devenues plus nettes et moins floues après avoir subi les ajustements sur la netteté.

Analyse du contraste et de la luminosité avant et après retraitement



Les boîtes à moustaches montrent que le retraitement a amélioré la qualité des images en réduisant les valeurs aberrantes. Pour le contraste, la médiane est restée stable, mais les valeurs très faibles de contraste ont été corrigées, et la gamme des valeurs s'est élargie. De même, pour la luminosité, la médiane est inchangée, mais les images trop lumineuses ont été ajustées, créant une distribution plus homogène. En résumé, les traitements ont réduit les extrêmes tout en maintenant une stabilité de la médiane.

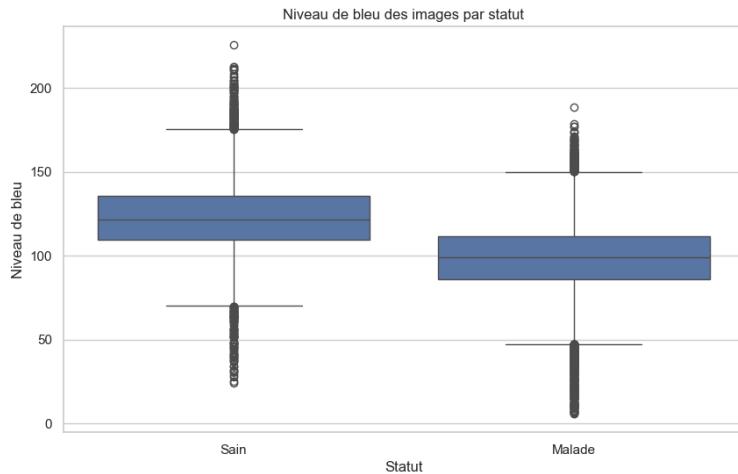
Analyse du redimensionnement



Avant retraitement, les images avaient des dimensions très variées, ce qui compliquait leur analyse et leur traitement uniforme. Cette hétérogénéité des tailles pouvait entraîner des biais lors de l'application de modèles d'apprentissage automatique ou d'autres analyses d'images, car les réseaux de neurones et les algorithmes de traitement d'images nécessitent des entrées de taille uniforme. Après retraitement, toutes les images ont été redimensionnées à une taille standard de 256x256 pixels, un format couramment utilisé dans la datascience. Cette normalisation permet non seulement de faciliter l'analyse et l'entraînement des modèles, mais elle garantit également que toutes les images sont traitées de manière cohérente, améliorant ainsi la performance des modèles d'analyse et de classification.

Analyses statistiques

Le niveau de bleu semble être le plus discriminant entre feuilles saines et feuilles malades, probablement lié à la présence de tâches de couleur jaune (R+V) sur la plupart de ces dernières.

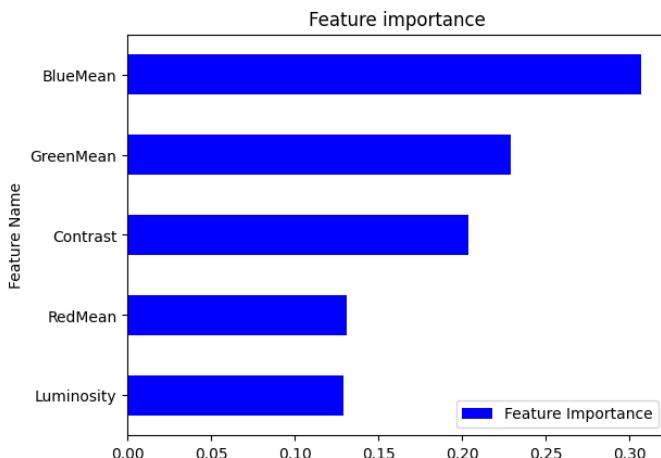


Si on réalise un test de student pour tester l'hypothèse que les niveaux de bleu sont significativement différents entre des feuilles saines et des feuilles malades, on trouve une statistique t de 128 et une valeur p très proche de 0, ce qui prouve que ce niveau de bleu est discriminant.

Ceci est confirmé par une première approche de classification via un premier modèle Random Forest mené en prenant pour Target le statut (Sain/Malade) et en Data, la moyenne de bleu, rouge, vert, le contraste et la luminosité donne un taux de bonne prédition de 88%.

Le graphique ci-contre montre l'importance des variables dans la contribution à la classification et confirme en contributeur principal le niveau de bleu.

Le taux de bonne prédition de la classe malade est de 93,5% mais le modèle ne fonctionne pas correctement pour le classement des images saines (taux de prédition de 74%)



Conclusions pré-modélisation

Nous disposons au terme de cette étape d'un fichier homogène en termes de qualité d'images, d'une quantité a priori suffisante pour appliquer un algorithme de reconnaissance, au déséquilibre entre classes moins marqué que sur les données d'origine, et qui affiche déjà quelques variables discriminantes sur l'une des deux variables cibles.

Partie 4 : Modélisation

Classification du problème

Type de problème de machine learning

La **cible du projet** est d'**offrir** à un individu, ne bénéficiant pas de l'expertise pour reconnaître une plante et sa maladie éventuelle, la **capacité** de l'identifier via l'appareil photo de son smartphone. Il s'agit donc d'un **problème de classification** des espèces ainsi que des maladies associées sur la base des images qui seront soumises par l'utilisateur.

Tâches de machine learning

La tâche associée est la **reconnaissance d'images**, spécifiquement de feuilles associées à 18 espèces végétales (fruits, légumineuses, céréales), dont les contours puis les détails sont progressivement analysés à travers les couches du réseau de neurones modélisé.

Métriques de performance utilisées pour comparer les modèles

A ce stade nous avons choisi 2 métriques.

Métrique principale

Accuracy : Compte tenu de la limitation du déséquilibre entre classes obtenu après la phase de pre-processing, nous avons choisi de retenir l'Accuracy comme métrique principale, définie comme le rapport entre le nombre de bonnes prédictions et le nombre total de prédictions.

Métriques secondaires

Loss : Nous avons également jaugé la rapidité de la convergence du modèle en suivant la notion de loss à chaque époque de l'apprentissage.

Selon les modèles et afin de nous assurer, malgré le faible déséquilibre de classes, que les classes les moins représentées soient correctement traitées, nous avons observé dans certains cas quelle était la k-ème Accuracy la plus faible parmi les classes proposées au modèle.

Choix du modèle et optimisation

Algorithmes testés

Nous avons retenu 6 algorithmes :

- 3 algorithmes Keras avec les modèles ResNet50V2, EfficientNetB0 et EfficientNetV2M;
- 1 algorithme FastAI avec le modèle efficientnet_b0
- 2 algorithmes Torch, l'un avec transfer learning (MobileNetV2, adapté pour une application mobile comme c'est l'objectif ici) et l'autre sans transfer learning
- 1 algorithme AlexNet

Résultats par modèle

1- KERAS

Keras est une bibliothèque open-source de haut niveau pour la création et l'entraînement de modèles d'apprentissage profond (deep learning). Elle a été développée par François Chollet en 2015 et est maintenant intégrée directement dans TensorFlow, où elle sert d'interface principale pour construire et entraîner des réseaux de neurones. Pour notre étude nous avons décidé de comparer 3 modèles de réseau de neurones : ResNet50V2, EfficientNetB0 et EfficientNetV2M en transfert learning et en fine tuning.

Description des modèles

ResNet50V2 :

- Architecture : ResNet50V2 (Residual Networks) utilise des blocs résiduels, ce qui permet de mieux entraîner des réseaux profonds en atténuant le problème de dégradation de la performance.
- Profundité : 50 couches.
- Performance : Bon compromis entre précision et temps de calcul, particulièrement efficace pour des tâches classiques de classification d'images (ImageNet).
- Avantages : C'est un modèle classique très utilisé avec des performances solides sur de nombreuses applications.
- Inconvénients : Moins efficace en termes de computation par rapport aux architectures récentes comme EfficientNet.

EfficientNetB0 :

- Architecture : EfficientNet utilise une méthode d'optimisation appelée "compound scaling", qui ajuste simultanément la profondeur, la largeur et la résolution de l'image, offrant ainsi une meilleure efficacité énergétique et une précision

supérieure.

- Profundité : Moins profond que ResNet50V2, mais beaucoup plus efficace pour les mêmes performances.
- Performance : Généralement plus rapide et plus efficace que ResNet50V2 pour la même précision.
- Avantages : Très efficace en termes de taille du modèle et d'utilisation des ressources, il donne de bons résultats même avec des réseaux moins profonds.
- Inconvénients : Moins éprouvé que ResNet50V2 dans certaines configurations.

EfficientNetV2M :

- Architecture : Une amélioration de l'EfficientNet original, qui intègre des techniques de fusion des convolutions et un meilleur équilibrage des ressources.
- Profundité : Plus profond que EfficientNetB0, ce qui le rend potentiellement plus puissant, mais aussi plus lourd en termes de calcul.
- Performance : Très performant sur des tâches complexes et des jeux de données volumineux, surpassant souvent EfficientNetB0 et ResNet50V2.
- Avantages : Très performant tout en restant optimisé pour des ressources limitées.
- Inconvénients : Modèle plus complexe et potentiellement plus long à entraîner comparé à EfficientNetB0.

Résultats

L'entraînement a été effectué 55 classes qui résultent du croisement espèce et maladie. Pour chaque modèle nous avons effectué un entraînement en transfert learning sur 10 époques avec les couches du modèle freezées. Nous avons dans une seconde étape nous avons effectué du fine tuning des modèles entraînés sur 30 époques avec les 30 dernières couches non freezées.

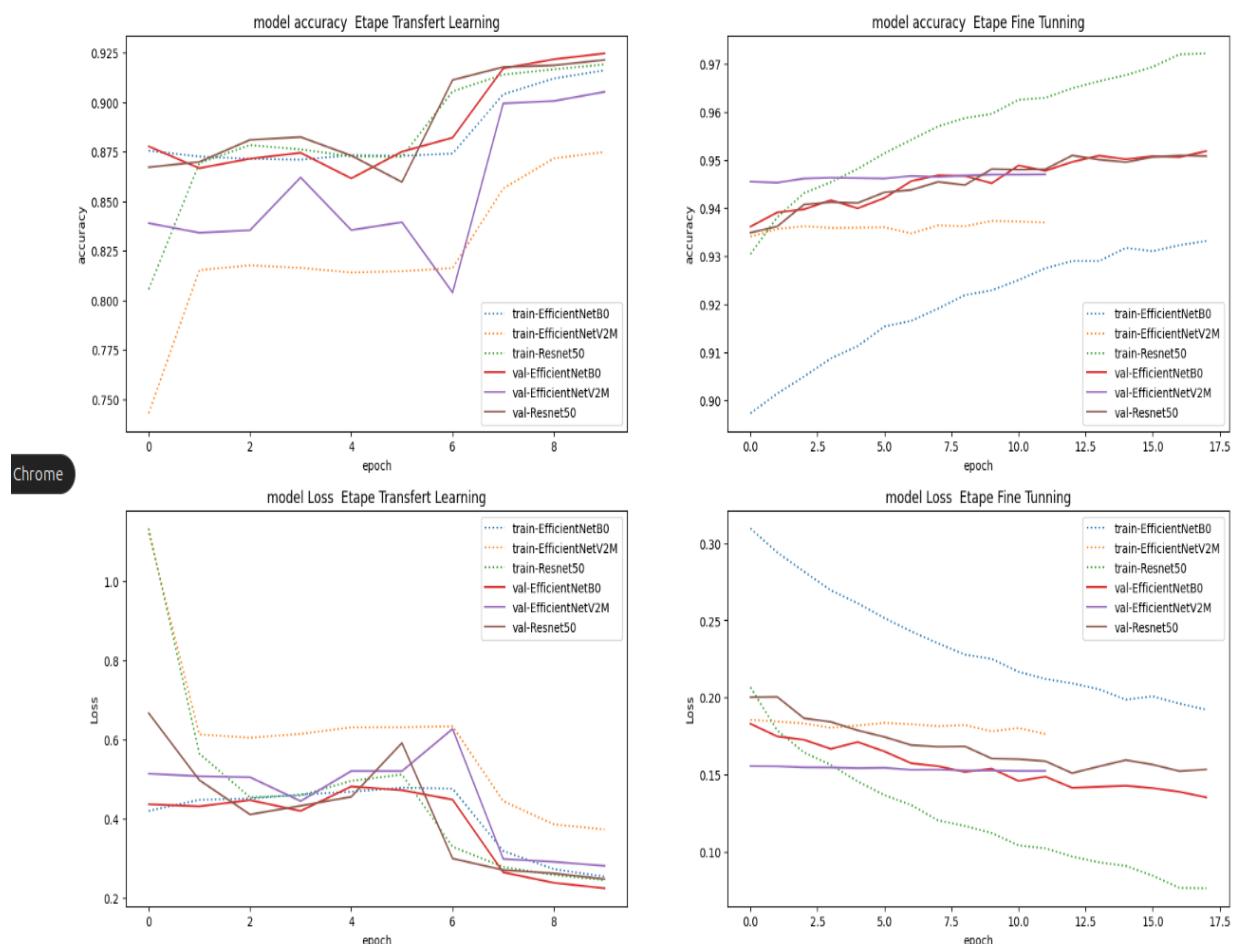
Modèle	Etape Transfer learning			Etape Fine tuning		
	ResNet50V2	EfficientNetB0	EfficientNetV2M	ResNet50V2	EfficientNetB0	EfficientNetV2M
Accuracy	0.92	0.92	0.91	0.95	0.95	0.95

Les 3 modèles plafonnent à 0.95 après le fine tuning. En effet, les résultats de la classification des images de l'espèce cassava sont très mauvais sur les trois modèles. Il faudra dans une seconde étape étudier en détail ces images.

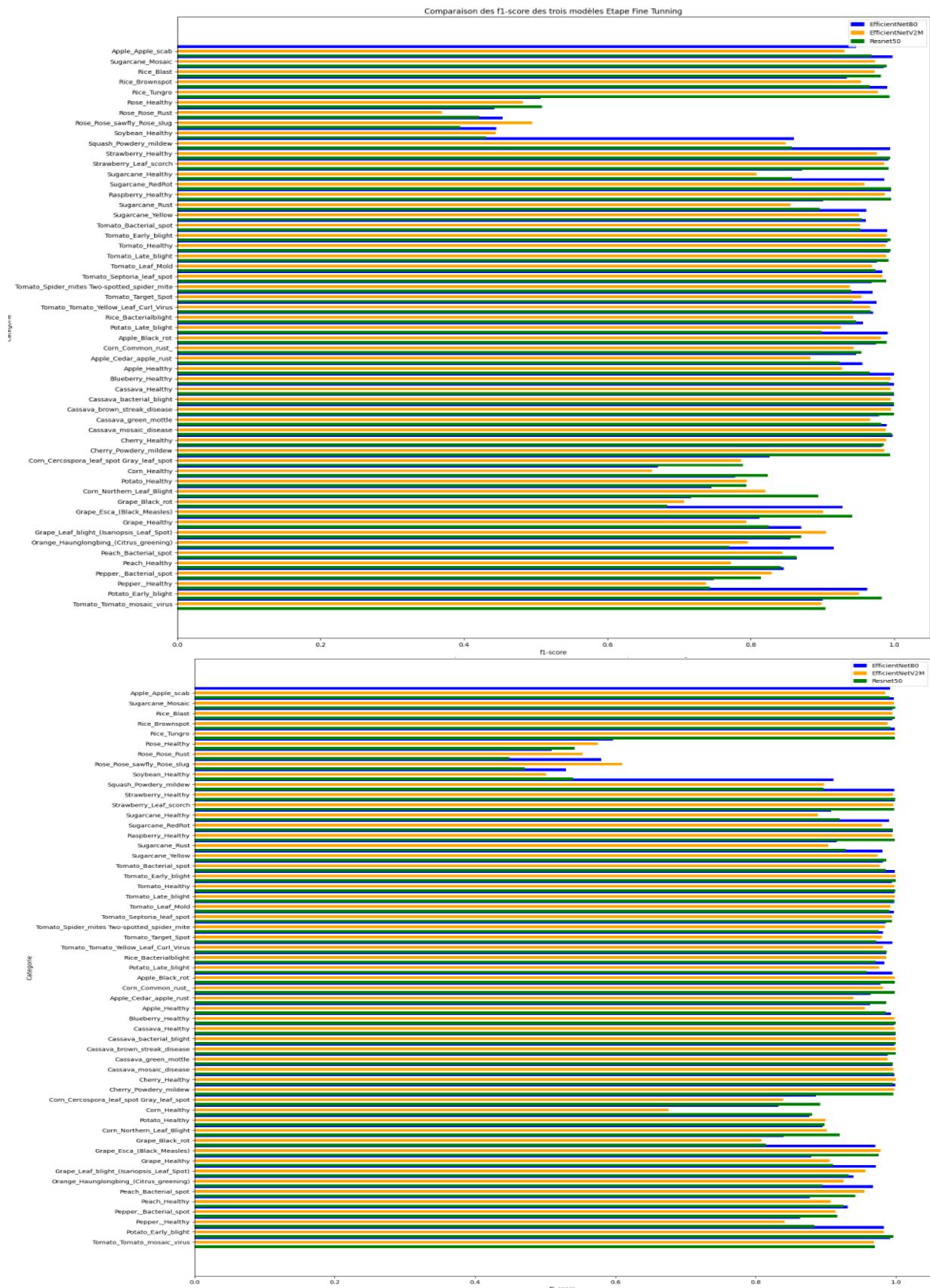
Exemple de classification report avec un zoom sur l'espèce cassava.

	precision	recall	f1-score	support
Apple_Apple_scab	1.00	0.99	0.99	502
Apple_Black_rot	1.00	1.00	1.00	497
Apple_Cedar_apple_rust	1.00	1.00	1.00	385
Apple_Healthy	0.99	1.00	0.99	502
Blueberry_Healthy	1.00	1.00	1.00	414
Cassava_Healthy	0.44	0.71	0.54	292
Cassava_bacterial_blight	0.52	0.40	0.45	128
Cassava_brown_streak_disease	0.65	0.37	0.47	262
Cassava_green_mottle	0.55	0.53	0.54	279
Cassava_mosaic_disease	0.91	0.89	0.90	1540
Cherry_Healthy	1.00	1.00	1.00	453
Cherry_Powdery_mildew	1.00	1.00	1.00	406
Corn_Cercospora_leaf_spot_Gray_leaf_spot	0.93	0.91	0.92	410
Corn_Common_rust_	0.99	1.00	1.00	477
Corn_Healthy	1.00	1.00	1.00	386
Corn_Northern_Leaf_Blight	0.93	0.93	0.93	477

Graphe comparatif de l'évolutions de la précision et de la perte des trois modèles



Comparaison du f1-score par classe et par modèles



2- FastAI

Fastai est une librairie open-source de haut niveau pour le deep learning.

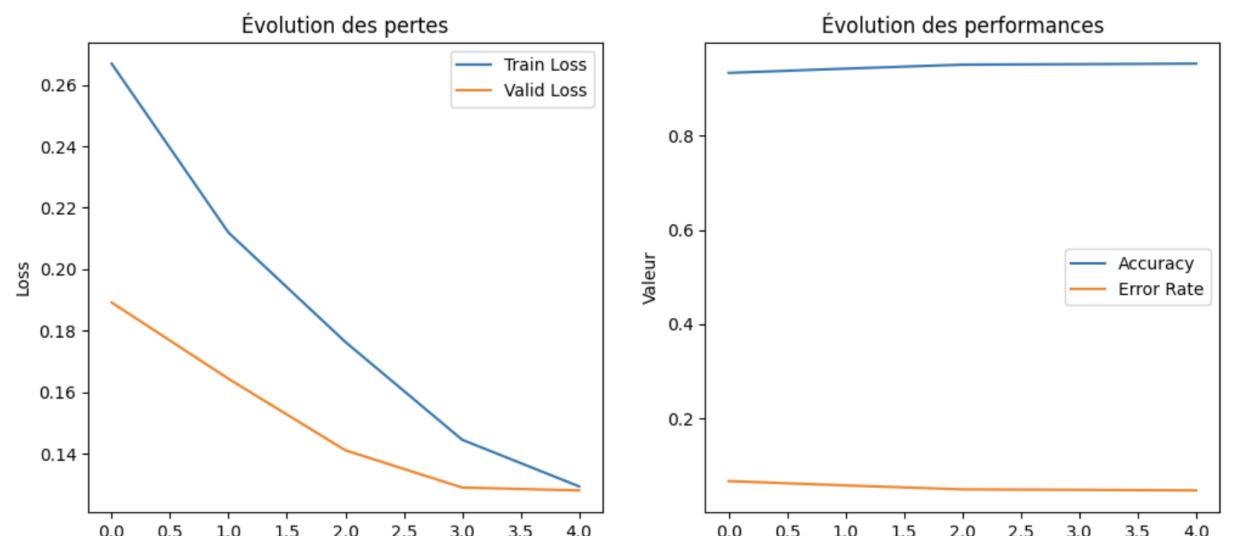
Construite sur PyTorch, elle a pour objectif de simplifier les entraînements de modèles, accélérer les déploiements de modèles tout en conservant la performance et les fonctionnalités avancées de Pytorch.

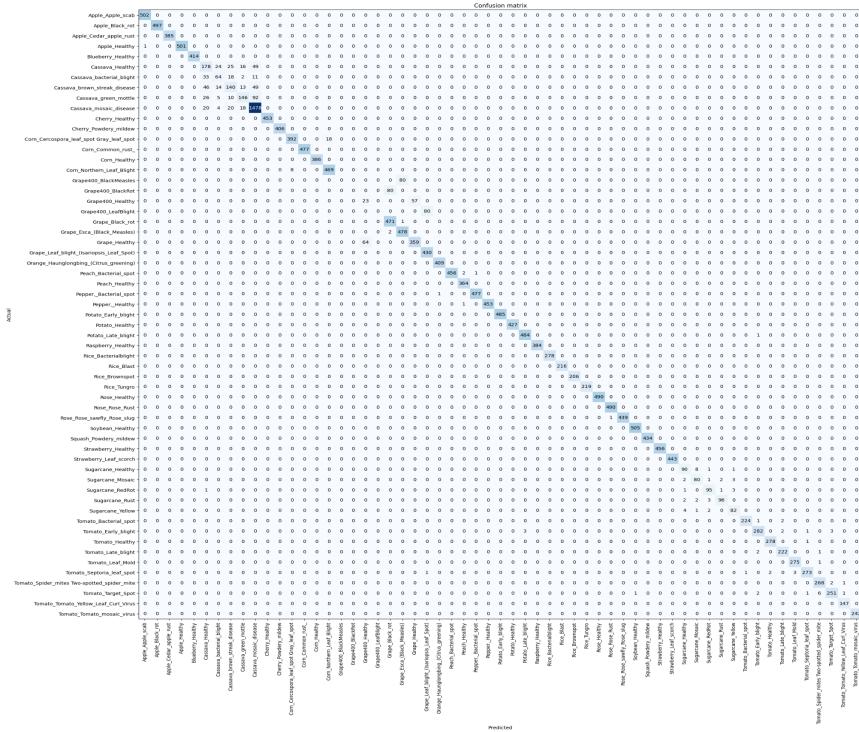
Le modèle sélectionné pour l'entraînement est **efficientnet_b0**. Proposé par Google, il est le plus petit des modèles de la famille EfficientNet, ce qui le rend pertinent pour son utilisation sur du matériel avec des ressources limitées (smartphone, tablette).

En analysant les résultats de performance à chaque itération, on constate que :

- 1) La perte de validation suit une tendance à la baisse, passant de **0.3093** à **0.1281**.
Rejoignant la perte d'entraînement, ce qui suppose une bonne généralisation du modèle
- 2) La précision croît jusqu'à atteindre **95%** sur 5 epoch seulement, avec un taux d'erreur à **4%**, ce qui implique une très bonne performance du modèle.

epoch	train_loss	valid_loss	accuracy	error_rate	time
0	0.266904	0.189192	0.933375	0.066625	08:03
1	0.211939	0.164384	0.942436	0.057564	08:02
2	0.176355	0.141109	0.950867	0.049133	07:55
3	0.144528	0.128985	0.951836	0.048164	08:02
4	0.129361	0.128059	0.953048	0.046952	08:02





3- Torch

Description des modèles

2 modèles ont été testés sous Torch :

- **1 modèle simple avec Transfer Learning**, basé sur le modèle pré-entraîné **MobileNetV2**, dont les couches intermédiaires sont neutralisées et seule la dernière couche est ajustée aux classes à prédire, puis ré-entraînée. Le choix s'est porté sur ce modèle car il est réputé être idéal pour des applications embarquées sur mobile, impliquant des contraintes de calcul (puissance limitée du matériel) ou nécessitant un fonctionnement en temps réel.
- **1 modèle simple sans Transfer Learning**, comprenant une première couche de convolution analysant le couleurs RGB (3 canaux) avec 32 filtres détectant les motifs simples et les contours, un deuxième bloc convolutif avec 64 filtres permettant de capter des motifs plus complexes (formes des feuilles, textures), une couche dense interprétant les motifs détectés et enfin la dernière couche de sortie donnant les probabilités par classe de plantes. La représentation du modèle est portée en Annexes.
- Les 2 modèles ont été testés successivement sur 18 classes (espèces) et 58 classes (croisements espèces x maladies)

Résultats

- S'agissant de la précision globale des modèles :

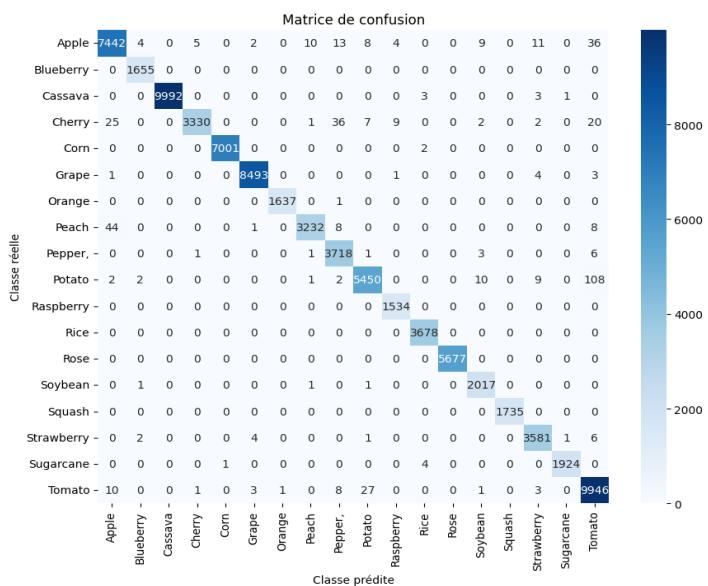
	18 classes		58 classes	
Modèle	transfer	sans transfer	transfer	sans transfer
Accuracy	99,38%	99,59%	94,79%	98,38%
3ème Accuracy la plus faible	98,15%	98,85%	40,44%	80,94%

Le modèle “simple” sans transfer learning apparaît plus précis avec des taux d’accuracy supérieur, a fortiori lorsque le nombre de classes augmentent.

La reconnaissance des espèces est bonne et aucune d'entre elles ne montre une accuracy inférieure à 97%.

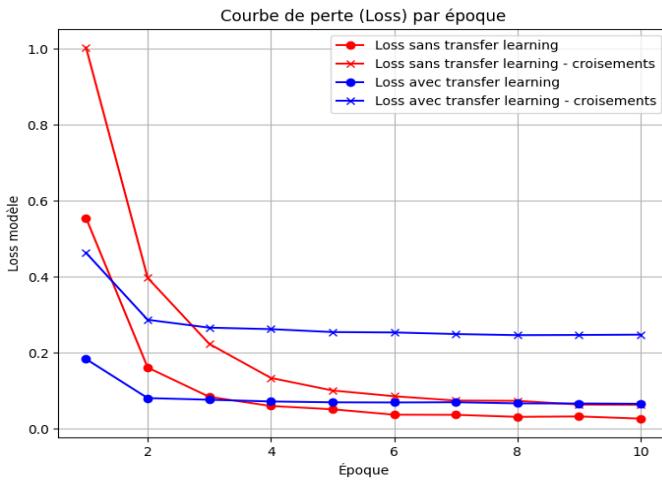
En revanche la reconnaissance des croisements espèces x maladies est plus problématique avec des précisions faibles sur les maladies touchant la vigne (avec des redondances de maladies ou des variantes de maladies qui semblent très proches qui seront traitées dans l'étape suivante) s'agissant du modèle sans transferlearning et de la cassava sur le modèle avec transferlearning.

- Les matrices de confusion pour les modèles à 18 classes sont présentées ci-dessous, en commençant par celle du modèle MobileNetV2.



		Matrice de confusion																	
		Classe réelle																	
		Apple	Blueberry	Cassava	Cherry	Corn	Grape	Orange	Peach	Potato	Raspberry	Rice	Rose	Soybean	Squash	Strawberry	Sugarcane	Tomato	
Apple	- 7491	9	1	0	4	8	0	8	2	1	2	0	0	2	0	9	0	7	
Blueberry	- 1	1654	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cassava	- 0	0	9997	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
Cherry	- 3	1	0	3343	0	6	0	9	17	4	10	1	0	8	0	2	0	28	
Corn	- 2	0	2	0	5992	0	1	1	0	0	0	3	0	0	0	0	0	2	0
Grape	- 0	0	0	0	0	8501	0	0	0	0	0	0	1	0	0	0	0	0	0
Orange	- 0	0	0	0	0	0	1635	0	1	0	0	0	0	0	0	0	0	0	2
Peach	- 5	1	0	0	1	2	0	3270	0	0	0	0	0	0	0	0	0	0	14
Pepper,	- 1	4	0	0	0	1	0	1	3687	5	1	0	0	10	0	1	0	19	
Potato	- 0	0	1	0	1	1	0	0	0	2	5554	7	0	0	3	0	4	0	11
Raspberry	- 0	0	0	0	0	0	0	0	0	0	1534	0	0	0	0	0	0	0	0
Rice	- 0	0	3	0	0	0	0	0	0	0	0	3671	0	3	0	1	0	0	0
Rose	- 0	0	0	0	0	0	0	0	0	0	0	0	5677	0	0	0	0	0	0
Soybean	- 0	0	0	1	0	0	0	0	0	0	0	0	2017	0	0	0	0	2	
Squash	- 0	0	0	0	30	0	0	0	0	0	0	0	1	0	0	1701	0	0	3
Strawberry	- 0	1	0	0	1	1	0	0	0	11	0	1	0	0	1	3570	0	9	
Sugarcane	- 0	0	0	0	1	0	0	0	0	0	0	3	0	0	0	0	1925	0	
Tomato	- 0	1	1	0	0	0	0	1	0	1	Raspberry	- 0	0	3	0	0	0	9992	
Apple		Blueberry	Cassava	Cherry	Corn	Grape	Orange	Peach	Potato	Raspberry	Rice	Rose	Soybean	Squash	Strawberry	Sugarcane	Tomato		

- **La convergence des modèles** lors de l'apprentissage est plus rapide pour le modèle pré-entraîné mais ce dernier est dépassé dès la 4ème époque, la loss convergeant vers 2,65% pour le modèle sans transfer learning 18 classes (resp. 6,27% avec 58 classes) et vers 6,55% pour le premier modèle 18 classes (resp. 24,75% avec 58 classes).



4- AlexNet

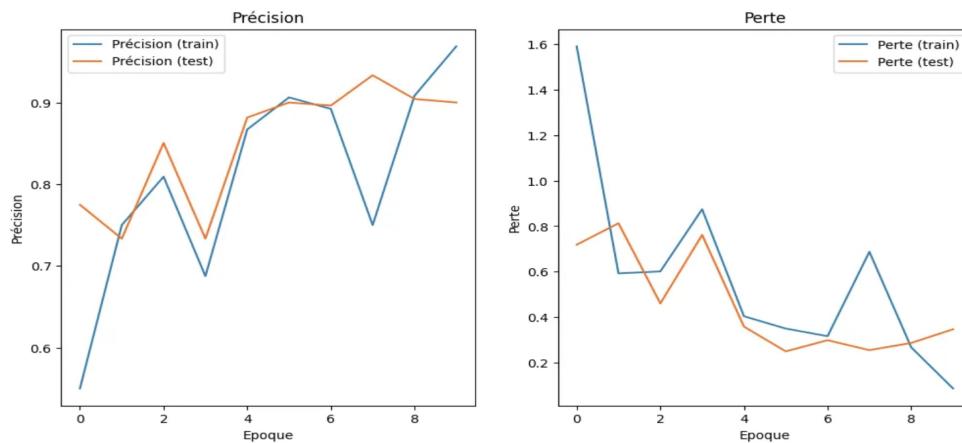
AlexNet est intéressant pour la classification d'images car il a introduit une architecture de réseau de neurones convolutifs profonds, utilisant des couches convolutives et des unités de rectification linéaire (ReLU), qui a significativement amélioré la précision de la reconnaissance d'images sur des jeux de données complexes comme ImageNet.

Le modèle a été initialisé en suivant l'architecture suivante :

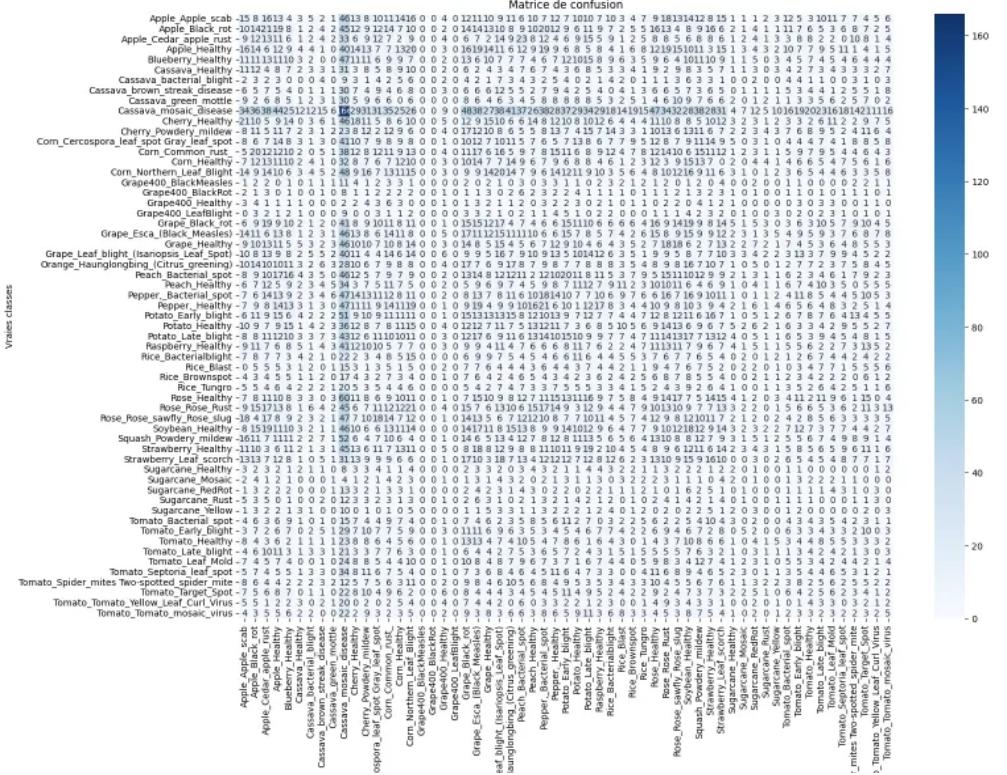
- **Couches convolutives et pooling** : L'architecture commence par deux couches convolutives suivies de couches de max-pooling pour extraire et réduire les caractéristiques spatiales des images. Les tailles des filtres et des strides varient pour capturer différents niveaux de détails.
- **Couches convolutives supplémentaires** : Trois couches convolutives supplémentaires sont ajoutées, suivies d'une couche de max-pooling, pour approfondir l'extraction des caractéristiques et capturer des motifs plus complexes dans les images.
- **Couches entièrement connectées et régularisation** : Après l'aplatissement des caractéristiques, deux couches entièrement connectées avec des unités de dropout sont utilisées pour la classification. La sortie finale est une couche dense avec une activation softmax pour la classification multi-classes.

La précision est de **80,40 %** sur les données d'entraînement et de **85,40 %** sur les données de test.

La perte (loss) semble globalement correcte car elle diminue au fil des époques, indiquant une amélioration du modèle, bien que des fluctuations soient observées sur les données d'entraînement.



L'analyse des métriques de classification indique que le modèle ne performe pas bien, avec des scores F1 très faibles pour toutes les classes, une précision globale de seulement 0,03, et des valeurs moyennes macro et pondérées également très basses, ce qui suggère une mauvaise capacité de généralisation et de classification des différentes catégories, probablement due à un **déséquilibre des classes dans les données**.



Optimisation

Nos modèles ont été optimisés en suivant 2 phases.

Phase 1

La première phase a reposé sur :

- **des corrections d'erreur** : mesures de précision sur des images test et non les images d'entraînement ;
- **des “élagages” rapides de modèles** jugés non pertinents ;
- **l'utilisation de techniques de fine-tuning** : choix des “optimizers”, gel/dégel de couches des modèles pour l'apprentissage, technique d'augmentation de data, ajustement des learning rates et introduction de scheduler, modification des fonctions de “loss”, techniques de callback, application de poids aux classes pour corriger les déséquilibres ;
- **des ajustements techniques pour un apprentissage plus rapide** : réorganisation du code ;
- **des choix réalisés pour le traitement de classes dont la prédiction est plus complexe** : il s'agit principalement de la classe “Cassava” avec la suppression pure et simple de ces classes ou bien focus sur la précision augmentée des modèles au fur et à mesure du fine-tuning.

Abandon de modèles non pertinents

2 modèles testés ont été immédiatement abandonnés :

- **Le modèle entraîné sous Pytorch sans transfer learning**, en raison de son taux de précision sur les images tests de 83%, très inférieur aux modèles pré-entraînés testés dans le chapitre précédent.
- **Le modèle AlexNet**, qui affiche à la fois un temps d'entraînement trop important et un taux de précision de 88%, également très en-deçà des autres modèles testés.

Ajustements techniques pour rapidité d'apprentissage

Les modèles testés sous FastAI dans l'étape précédente (EfficientNetB0 et VGG12) ont été migrés vers TensorFlow pour les raisons suivantes :

- TensorFlow est plus simple à intégrer sur Firebase ;
- Il est optimisé pour la performance et la taille des modèles ;
- Il intègre un large écosystème avec support pour la quantification (réduction de la taille des modèles) ;
- Il est compatible avec les modèles pré-entraînés comme les réseaux convolutifs (CNN) pour les images.

Choix réalisés pour le traitement de la Cassave

Le traitement de la Cassave dans les opération de “fine-tuning” du modèle ont pris deux formes:

- La suppression pure et simple de la cassave dans le cadre de l'apprentissage: c'est le cas dans ce qui suit sur les entraînements de VGG12 et EfficientNetB0 sous TensorFlow ;
- Le maintien de cette espèce dans la base d'apprentissage mais, avec une recherche spécifique d'amélioration de la prédiction: c'est ce qui a été fait sur MobileNetV2 sous Pytorch, et VGG16, Vit16, EfficientNetV2M et ResNet50V2 sous Keras.

Résultats du fine-tuning

MobileNetV2 sous Pytorch

Le modèle a été calibré et testé comme suit :

- # époques : 10
 - # époques sur le classifier : 5
 - époques de fine tuning sur les autres couches : 5
- Learning rate : 0.001 pour le classifier et 0.0001 pour le fine tuning
- Optimizer : Adam

L'accuracy globale est de 97,22% mais un F1-score de 65,21% seulement sur la Cassave Healthy

Le fine-tuning pour mieux prédire cette classe a consisté à:

- Augmentation sur les classes spécifiées : Random Horizontal Flip, Rotation aléatoire jusqu'à 30%
- Augmentation du nombre d'époques à : 20
- Entrainement de la dernière couche seulement
- Learning rate : 0.0001
- Optimizer : Adam

L'accuracy globale s'est avérée dégradée à 86,85% avec un F1-score de 18,80% sur la Cassave Healthy, ce qui a mené à l'abandon du modèle.

EfficientNetB0 et VGG12 sous TensorFlow

Le fine-tuning a consisté à:

- Organisation du code pour améliorer le temps d'apprentissage
- Augmentation de la data sur le modèle
- Ajouts des poids sur les classes pour lutter contre le déséquilibre
- Gel/Dégel du modèle de base
- Shuffle passé à False dans le dataset de test

Les métriques précision, recall et F1-score gagnent environ 1pt pour atteindre 95% et 98% pour respectivement EfficientNetB0 et VGG12. Ce dernier est donc plus performant, ce qui élimine le premier modèle.

Bien que dans l'apprentissage de ces deux modèles, les classes "Cassava" aient été exclues, on note des classes de tomates moins bien prédites telles que celles atteintes des maladies "Mosaic Virus" et "Spider Mites" (Précision : 73%, Recall : 98%, F1-Score : 84%). On note d'ailleurs qu'à l'oeil nu la résolution de certaines images ne permet pas de détecter ces maladies.

Mosaic Virus



Spider Mites



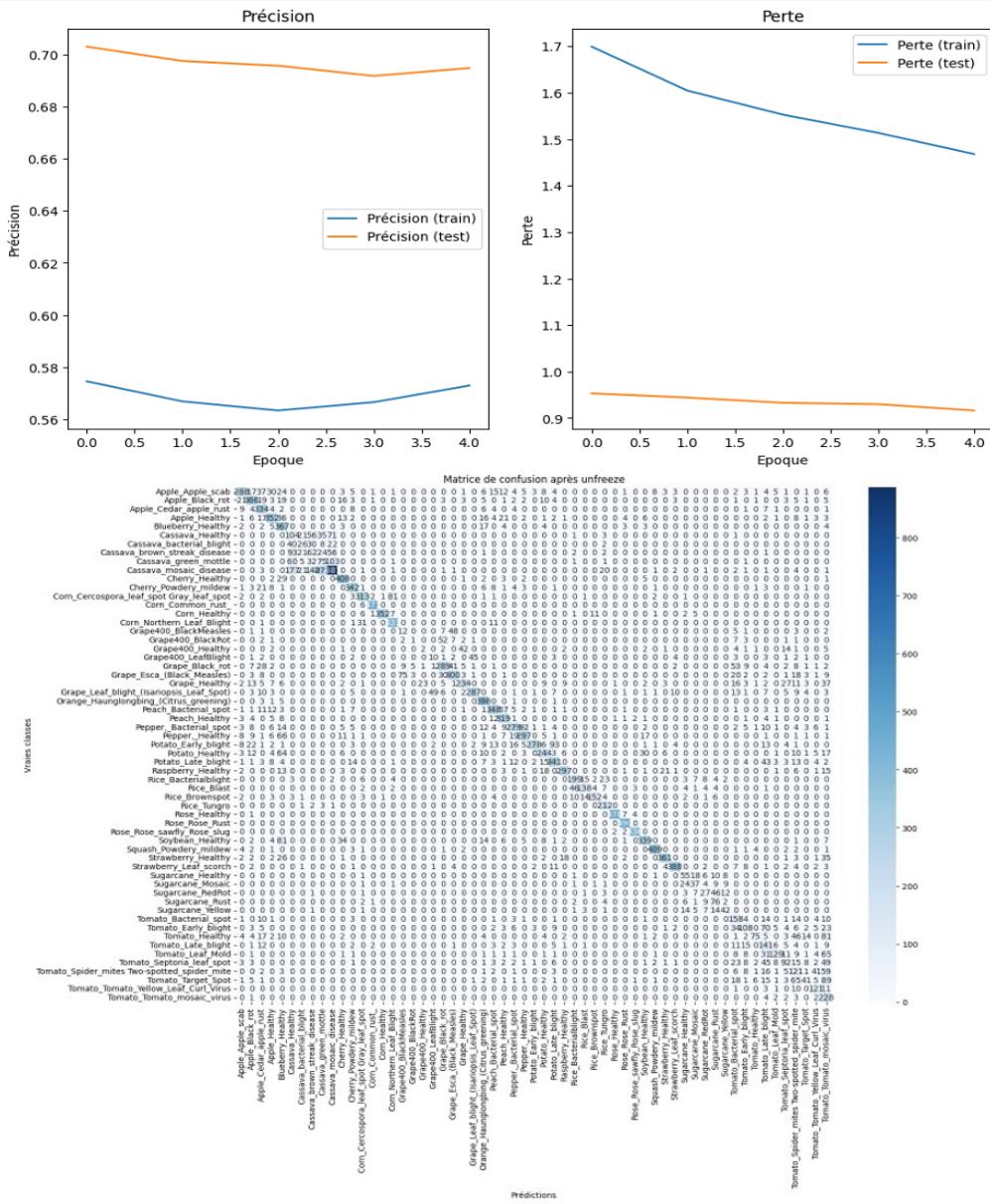
VGG16 sous Keras

L'entraînement de ce modèle a donné globalement des résultats décevants.

Pour ce modèle, les mesures de fine-tuning ont consisté à:

- Réduction de la taille des images pour améliorer la vitesse d'entraînement (128x128)
- Entraînement sur 10 époques avec freeze des paramètres
- Puis entraînement sur 10 epochs avec unfreeze des 5 dernières couches et ajout de poids pour gérer le déséquilibre des classes
- Learning rate (optimizer Adam) plus petit lors du dégel (0.00001 vs 0.0001 lors du freeze) pour améliorer l'apprentissage
- Callbacks :
 - EarlyStopping pour arrêter l'entraînement si la perte de validation ne s'améliore pas après 5 epochs
 - ReduceLROnPlateau pour réduire le taux d'apprentissage de 20% après 3 époques sans amélioration de la perte de validation
 - StopTrainingAtAccuracy pour arrêter l'entraînement dès que la précision atteint 90%

On constate une baisse de l'Accuracy de 75,02% à 70,02% au terme du fine-tuning. La classification s'améliore sur l'ensemble des classes sauf sur la Tomate, la Cassave et la Canne à sucre. La faible Accuracy et la longueur de l'entraînement nous font conclure à l'abandon du modèle.



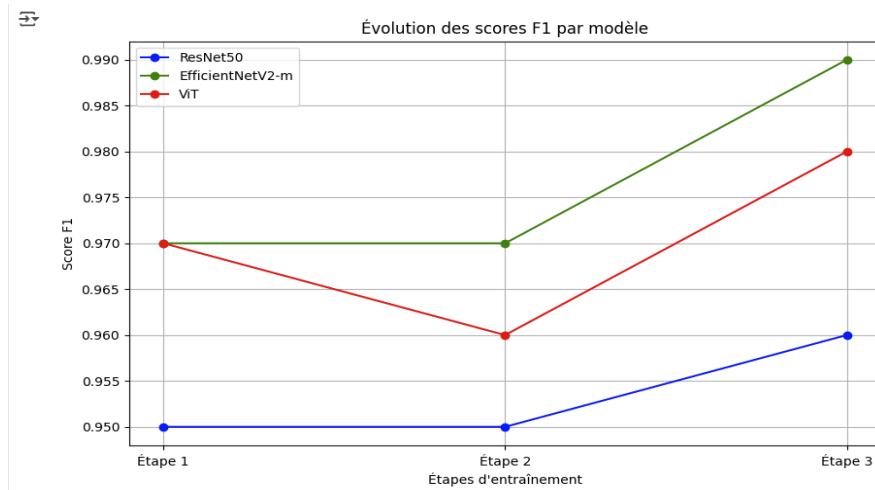
ResNet50V2, EfficientNetV2M et Vit16 sous Keras

Pour ces 3 modèles, nous avons suivi le même plan d'entraînement suivant.

Phase d'Entraînement	Optimiseur	Scheduler de Learning Rate	Perte	Callbacks	Détails supplémentaires
Premier Entraînement	Adam	CosineDecayRestarts en Fine tuning	SparseFocalLoss (sans poids)	earlystop, time_callback, printLR, model_checkpoint_callback	20 epochs max en transfert learning, 5 epochs max en finetuning avec toutes les couches dégélées
Deuxième Entraînement (Amélioration 1)	AdamW	CosineDecayRestarts	SparseFocalLoss (avec poids des classes)	earlystop, time_callback, printLR, model_checkpoint_callback	20 epochs max de finetuning

Troisième Entrainement (Amélioration 2)	AdamW	CosineDecayRestarts	SparseFocalLoss (avec poids des classes)	earlystop, time_callback, printLR, model_checkpoint_callback	20 epochs max de finetuning Augmentation manuelle des images des classes minoritaires sur Cassava + augmentation ciblée sur ces classes
---	-------	---------------------	--	--	---

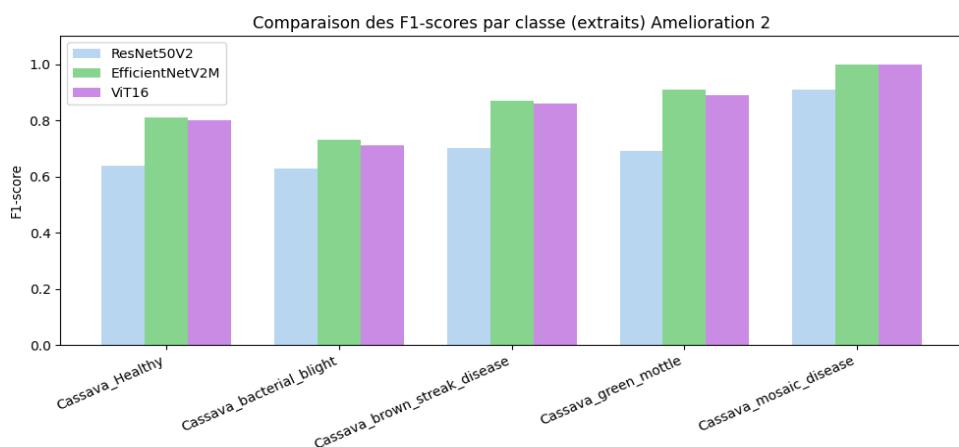
La troisième étape s'est avérée la plus significative en matière de gain de performance, avec un traitement spécifique apportée à la classe "Cassava" en augmentant manuellement les images des classes minoritaires et en appliquant une augmentation algorithmique ciblée.



Les résultats du fine-tuning par modèle sont résumés dans le tableau ci-dessous, les deux modèles EfficientNetV2M et ViT16 se distinguant sur les classes déséquilibrées comme la Cassave.

Modèle	Accuracy	Macro F1-score	Weighted F1	Observations clés
ResNet50V2	0.96	0.96	0.96	Bons résultats globaux, mais performance moyenne sur les classes Cassava
EfficientNetV2M	0.99	0.98	0.99	Meilleure performance globale, bonne gestion des classes déséquilibrées comme Cassava
ViT16	0.98	0.97	0.98	Meilleure performance globale, bonne gestion des classes déséquilibrées comme Cassava

Compte tenu du faible % de prédiction, la Cassave a fait l'objet d'un focus particulier. On constate que le finetuning réalisé a eu un effet spectaculaire sur la reconnaissance de ces classes (hausse du F1-score de 15pts environ chez EfficientNetV2M et ViT16 notamment sur la Cassave saine).



S'agissant des autres classes, on relève les principaux enseignements suivants:

- **Raisins, cerises, agrumes, pommes, pêches, etc. (classes équilibrées)** : les trois modèles ont des scores F1 proches de 1.00 sur toutes ces classes : aucune distinction significative.
- **Sugarcane (classes mineures)** : Bonne performance des trois modèles. EfficientNetV2M excellent suivi de ViT16
- **Tomato (13 sous-classes, très représenté mais varié)** : Bonne performance des trois modèles. EfficientNetV2M excellent suivi de ViT16

En conclusion, suivant les critères liant métriques de précision, de temps d'apprentissage et de taille du modèle, EfficientNetV2M semble devoir être privilégié.

Critère	Gagnant	Commentaire
Précision globale	EfficientNetV2M	Excellent suivi de Vit16
Robustesse sur classes rares	EfficientNetV2M	Meilleur équilibre F1
Performances Cassava	EfficientNetV2M	Résiste mieux au déséquilibre
Simplicité et rapidité entraînement	ResNet50V2	Moins lourd, mais moins performant
Durée d'inférence	ResNet50V2	4 fois plus rapide que EfficientNetV2M et 6 fois plus rapide que ViT16
Taille disque du modèle	ViT16	Le plus léger suivi de ResNet50V2 , EfficientNetV2M est deux fois plus lourd

Vision Transformer pure	ViT16	Très Bon
-------------------------	-------	----------

Conclusion au terme de la Phase 1

Nous tirons de cette première phase deux enseignements:

- **La qualité de la précision semble souffrir de la classe Cassave :** quel que soit le modèle employé, et malgré les efforts de fine-tuning, il semble qu'une difficulté de reconnaissance subsiste pour cette classe. Cette difficulté est selon intrinsèquement lié à la qualité des images et aux problèmes apparents de labellisation. Nous nous interrogeons donc sur l'abandon de cette classe dans le modèle final.
- **Les modèles affichent des tailles variables, certains sans doute incompatibles avec un usage mobile.** En conséquence, nous décidons de retenir à ce stade deux modèles, l'un léger, embarqué sur mobile (VGG12), et l'autre plus puissant hébergé dans un cloud (EfficientNetV2M).

Phase 2

La deuxième phase a été consacrée à l'interprétation des modèles en utilisant des outils comme GradCam ou SHAP ainsi qu'à l'apprentissage, par lequel nous commençons cette phase, d'un modèle alternatif aux CNN et Transformer Standard : Swintransformer.

Swintransformer

Le modèle utilisé est un Swin Transformer pré-entraîné, adapté aux 55 classes du jeu de données incluant la Cassave:

- **L'entraînement a été réalisé** avec un mécanisme d'early stopping qui interrompt l'apprentissage si la performance sur le jeu de validation ne s'améliore plus pendant trois époques consécutives.
- **L'optimiseur utilisé est AdamW**, qui semble reconnu pour sa robustesse et son efficacité pour ce modèle **avec un learning rate de 0,0001**.
- Des poids de classe sont calculés en fonction de la fréquence de chaque catégorie dans le jeu d'entraînement. **Le poids de la classe « Cassava_mosaic_disease » est volontairement réduit** afin de limiter son influence lors de l'apprentissage. Il y a en effet 5 fois plus d'images de Cassaves atteintes de cette maladie que dans les autres classes de Cassaves.
- À chaque époque, la fonction de coût utilisée est la CrossEntropyLoss pondérée par ces poids de classe.
- **Nous retenons le modèle ayant obtenu la meilleure performance sur le jeu de validation (les images du jeu de test).**

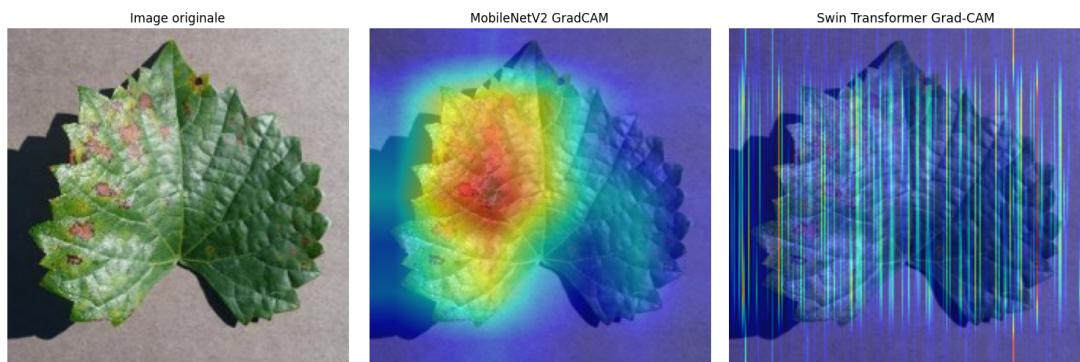
Dans les faits, nous avons eu besoin de 6 époques pour atteindre l'optimum avec un taux d'accuracy de 97,72%, un macro F1-score de 96,63% et un weighted F1-score de 97,73%.

La classe Cassave reste néanmoins encore très moyennement prédictive, rejoignant ainsi la conclusion générale que les images qui servent à l'apprentissage pour ces classes sont sans doute à l'origine de ce problème de classification (qualité, labellisation) : à titre d'exemple la classe "Cassava_healthy" affiche un F1-score de 66%.

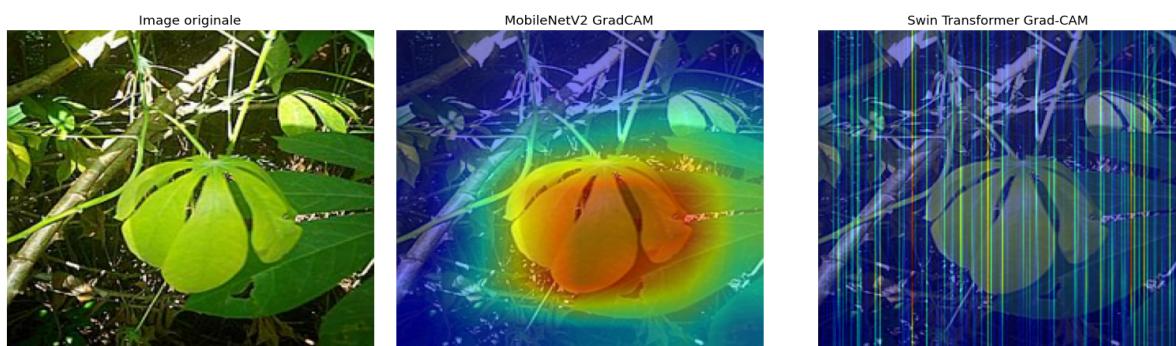
Interprétation comparée MobileNetV2 et Swintransformer

Afin d'interpréter les modèles, nous avons utilisé GradCam pour visualiser les régions de l'image ayant le plus influencé la prédiction du modèle sur la base du dernier bloc du modèle.

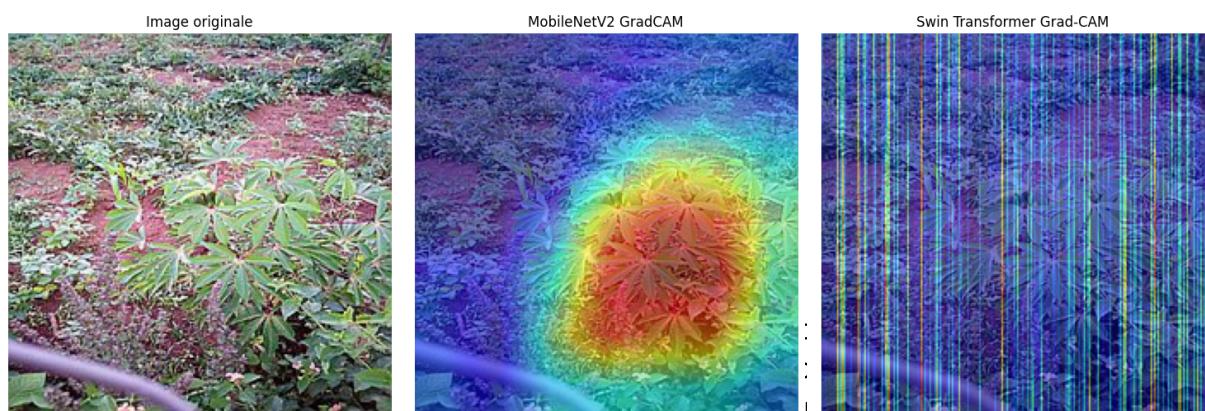
Pour une image de vigne atteinte de "Black Measles"



Pour une image de Cassave saine (environnement végétal)



Pour une deuxième image de Cassave (environnement terre)



En revanche pour SwinTransformer, GradCam affiche des zones de chaleurs sous formes de traits verticaux plus difficile à interpréter.

Interprétation EfficientNetV2M

Le modèle EfficientNetV2M nous avait donné une accuracy globale à 0.95 après le fine tuning. Cependant, les résultats de la classification des images de l'espèce cassava sont très mauvais :

	precision	recall	f1-score	support
Cassava_Healthy	0.44	0.71	0.54	292
Cassava_bacterial_blight	0.52	0.40	0.45	128
Cassava_brown_streak_disease	0.65	0.37	0.47	262
Cassava_green_mottle	0.55	0.53	0.54	279
Cassava_mosaic_disease	0.91	0.89	0.90	1540

Nous avons donc tâché de comprendre pourquoi le modèle classifie mal en regardant plus en détail les images de Cassave.

Nous avons choisi d'utiliser la méthode de saillance native à TensorFlow, car elle est pleinement compatible avec notre environnement et ne présente pas de problèmes d'incompatibilité, ce qui facilite son intégration.

1- Analyse d'un premier batch avec la carte de Saillance

Image 1



Carte de saillance



Image 2



Carte de saillance



- **Les cartes sont bien générées :**

Les zones d'attention (en rouge/orange) sont visibles sur les images et indiquent les régions jugées importantes par le modèle.

- **Alignment avec les feuilles :**

Les zones chaudes se trouvent souvent sur des parties pertinentes des feuilles (nervures, taches, contours), ce qui est encourageant.

- **Présence de bruit parasite :**

Dans certaines images, des zones d'attention sont visibles en dehors des feuilles, notamment sur l'arrière-plan ou sur d'autres objets. Cela suggère que le modèle peut être distrait par des éléments non pertinents.

- **Variabilité d'attention :**

Le modèle ne focalise pas systématiquement sur les mêmes zones d'une image à l'autre, ce qui reflète à la fois une adaptation aux différences d'images et un possible manque de stabilité dans la reconnaissance.

- **Zones de saillance parfois trop petites :**

Les régions influentes sont parfois très restreintes, ce qui peut refléter une faible confiance du modèle ou une difficulté à localiser les bonnes zones.

- **Présence de plusieurs feuilles :**

Lorsqu'il y a plusieurs feuilles dans l'image, le modèle semble confus et a du mal à identifier la feuille centrale ou la plus pertinente pour la classification.

- **Analyse ciblée des classes mal prédites :**

Une attention particulière sera portée sur les classes de cassave pour lesquelles le modèle obtient les moins bons résultats. Cela permettra d'identifier les causes des erreurs et d'améliorer la robustesse du modèle.

2- Analyse d'un second batch sur les espèces de Cassave mal classifiées avec la carte de Saillance

Image 0 - Vraie: 3 | Prédit: 0



Carte de saillance (mauvaise prédition)



Image 1 - Vraie: 4 | Prédit: 0



Carte de saillance (mauvaise prédition)



Confirmation de l'analyse :

L'observation globale sur les classes mal prédites confirme que le modèle ne distingue pas toujours correctement la feuille à analyser. Dans plusieurs cas, l'attention est partiellement portée sur l'arrière-plan ou sur d'autres feuilles, ce qui renforce l'idée d'une difficulté du modèle à isoler la feuille pertinente dans des scènes complexes.

3- Limites de l'interprétation avec SHAP et GradCam

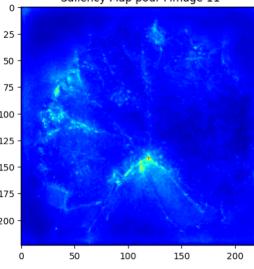
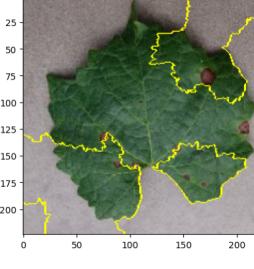
- L'utilisation de **SHAP DeepExplainer** a été écartée pour cette architecture, car cette méthode n'est pas encore compatible avec toutes les couches de TensorFlow 2, notamment celles présentes dans EfficientNet, comme DepthwiseConv2d et BiasAdd
- L'application de **Grad-CAM** a rencontré des problèmes lors de l'appel du modèle avec TensorFlow, qui refuse de traiter correctement les entrées. Nous suspectons une incompatibilité avec TensorFlow 2.19 ou un problème lié à la construction du modèle, rendant son utilisation impossible dans cette configuration.

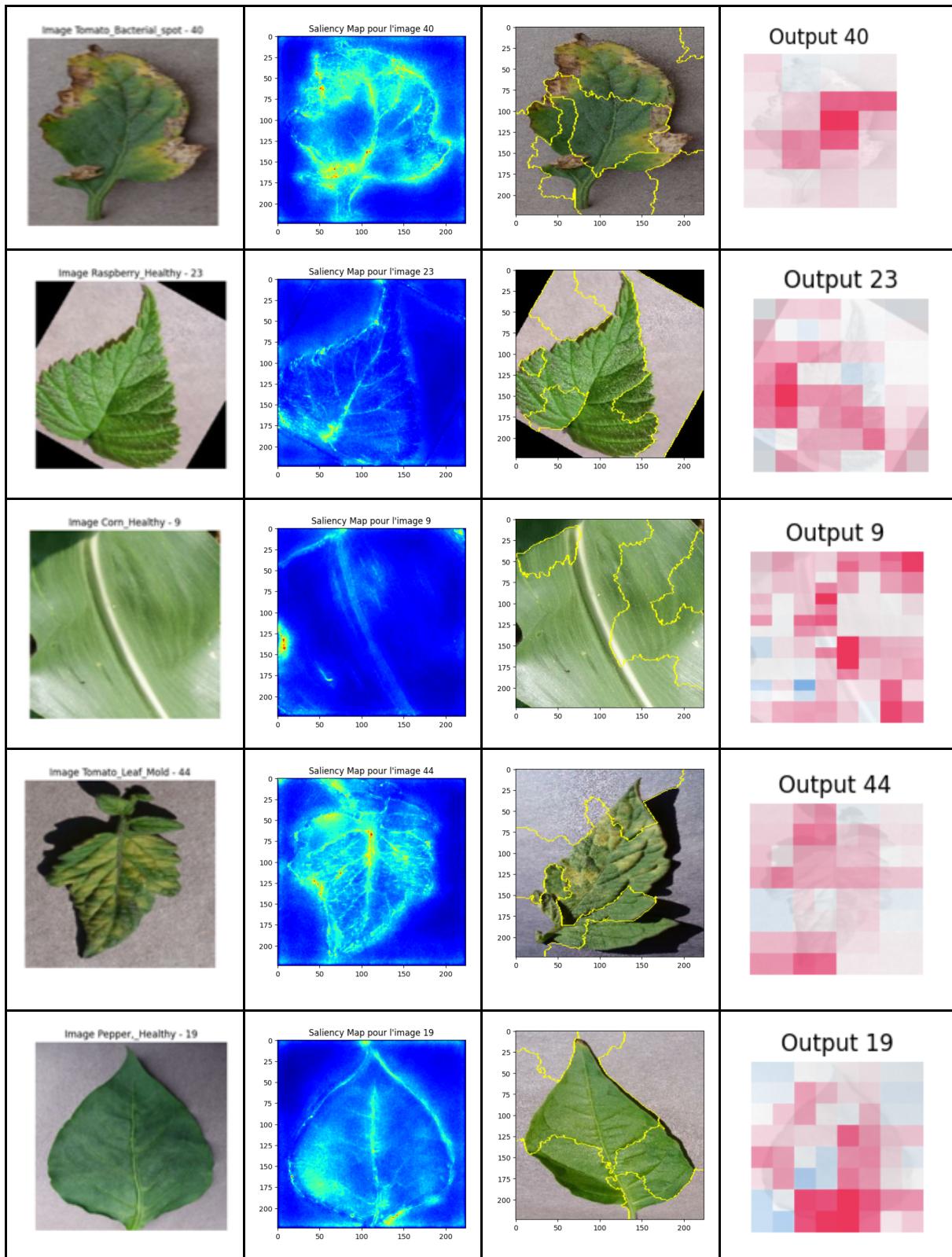
Ces difficultés techniques rencontrées avec SHAP et Grad-CAM sont aussi intrinsèquement liées à la complexité et à la profondeur de notre modèle EfficientNetV2M, qui comporte des couches spécifiques (DepthwiseConv2d, BiasAdd) rendant son interprétation plus difficile avec certains outils standards.

Interprétation VGG16

Pour comprendre les décisions de notre modèle VGG16, nous allons utiliser 3 outils d'interprétation :

- Saliency, qui permet de mettre en évidence les pixels les plus importants d'une image pour une prédiction donnée.
- SHAP (SHapley Additive exPlanations), pour visualiser les caractéristiques qui permettent la prédictions d'une classe sur une image.
- LIME (Local Interpretable Model-agnostic Explanations), qui donne les caractéristiques d'une image qui vont le plus influencer la décision du model.

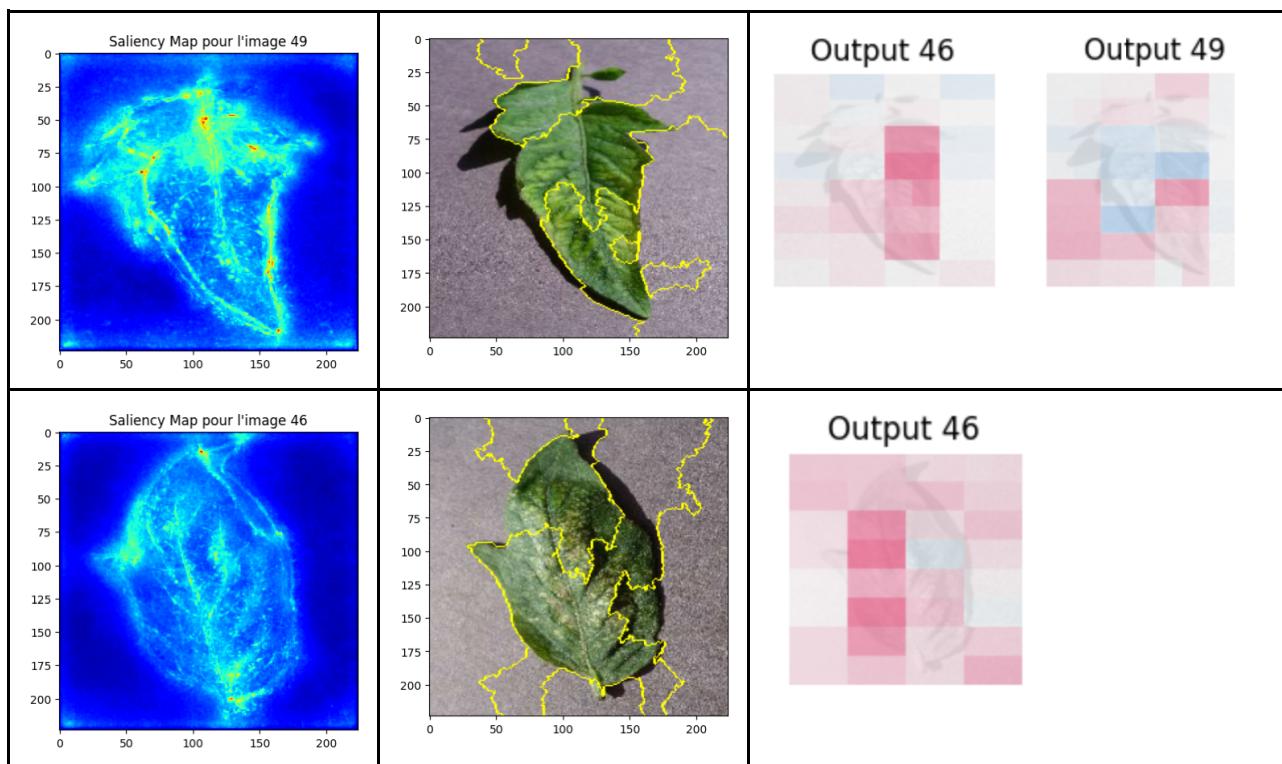
Image source	Saliency	LIME	SHAP
	<p>Saliency Map pour l'image 11</p> 		<p>Output 11</p> 



D'après nos outils d'interprétation, on remarque que les saillances désignent bien les parties des images importantes pour chacune des classes. Elle reconnaît les contours des plantes et les différences de couleurs issues des maladies.

Sur la classe "Corn_Healthy", on peut avoir un doute sur l'angle gauche de l'image qui est coloré : Est-ce que ce qui est mis en valeur est le fond noir ou le contour de la plante ? Cette surbrillance n'étant pas mise en valeur sur le coin bas à gauche, je pense qu'il s'agit bien du contour de la plante, permettant sa reconnaissance.

La colonne SHAP contient la classe pour laquelle l'image contient le plus de caractéristiques. Il confirme que notre modèle est capable de capturer les motifs et les éléments visuels pertinents pour effectuer des prédictions précises, étant donné qu'il s'agit à chaque fois de la bonne classe d'image sauf pour la classe 49 (Mosaic Virus). En analysant les graphiques sur cette classe, on confirme que le modèle a du mal à tirer ses caractéristiques spécifiques avec la classe 46.



Stratégies de traitement des images Cassava

Au vu des performances modestes obtenues par nos modèles sur le sous-ensemble Cassava, nous avons décidé de mettre en œuvre plusieurs stratégies d'amélioration. Ces approches visent à renforcer la qualité des données et à améliorer la capacité des modèles à discriminer les classes Cassava, souvent déséquilibrées et visuellement similaires.

1. Augmentation massive des classes minoritaires

Une première approche a consisté à rééquilibrer le dataset Cassava par une augmentation ciblée.

Principe

- Les classes minoritaires (Healthy, Bacterial blight, Brown streak disease, Green mottle) ont été massivement augmentées.
- L'objectif était d'atteindre un volume comparable à la classe majoritaire (Cassava Mosaic Disease).
- Les techniques utilisées incluent : rotations, flips, changements de luminosité, contrastes, translations, etc.

Objectifs

- Apporter de la diversité visuelle aux classes sous-représentées.
- Réduire le biais induit par la classe majoritaire.
- Améliorer la généralisation du modèle sur les classes rares.

2. Filtrage basé sur un score multi-critères

Une seconde approche a consisté à filtrer les images Cassava jugées peu fiables à l'aide d'un score multi-critères.

Définition du score

Le score est défini comme une combinaison pondérée de trois composantes :

- Précision locale du soft voting : probabilité que le soft voting des modèles prédit correctement cette image sur le dataset d'entraînement.
- Confiance du soft voting : probabilité moyenne attribuée à la classe majoritaire par le soft voting.
- Accord inter-modèles : mesuré par la divergence de Kullback-Leibler (KL) entre les distributions de sortie des différents modèles (plus la divergence est faible, plus les modèles sont en accord).

Formule du score :

$$\text{Score} = 0.5 \times \text{Précision Soft Voting} + 0.3 \times \text{Confiance Soft Voting} + 0.2 \times (1 - \text{Divergence KL})$$

Seuil de filtrage

Les images sont considérées comme peu fiables lorsque leur score se situe dans le premier décile (les 10 % les plus faibles). Ces images ont été exclues du dataset d'entraînement.

3. Relabellisation automatique basée sur le score

Enfin une troisième stratégie a consisté à ré-étiqueter automatiquement les images Cassava jugées peu fiables, plutôt que de les supprimer.

Processus de re-labellisation

- Les images du premier décile de score ont été re-labellisées automatiquement.
- Le nouveau label correspond à la classe prédite par le soft voting, considérée comme plus fiable que le label d'origine.
- Les autres images ont conservé leur annotation initiale.

Objectifs

- Corriger les labels potentiellement erronés.
- Maintenir la taille du dataset en remplaçant les étiquettes douteuses.
- Exploiter la confiance collective des modèles pour guider la correction.

[4. Fine-tuning des modèles sur les nouveaux datasets](#)

Les datasets générés (filtré, relabellisé, augmenté) ont été utilisés pour affiner les modèles préalablement entraînés EfficientNetV2M, ResNet50V2 et Swintransformer. Le modèle Convnext étant relativement plus long à entraîner, nous avons décidé de l'exclure de cette étude d'amélioration.

Méthodologie

- Une augmentation online (flip horizontal, variation de contraste, saturation, etc.) a été appliquée durant le fine-tuning.
- Le scheduler CosineDecayRestarts a permis d'ajuster dynamiquement le taux d'apprentissage.
- L'optimiseur utilisé est AdamW avec un weight decay de 1e-5.
- La fonction de perte choisie est une SparseFocalLoss ($\gamma = 2.0$), pondérée par des poids de classes.

Objectifs

- Exploiter les datasets nettoyés et enrichis.
- Adapter les modèles à une distribution de données plus fiable.
- Mesurer directement l'impact des stratégies sur les performances, notamment sur les classes Cassava.

[5. Résultats obtenus](#)

Les résultats suivants comparent les performances des modèles avant et après application des trois stratégies (filtrage, relabellisation, augmentation) sur l'ensemble des classes (global) et spécifiquement sur les classes Cassava.

Deux tableaux sont présentés :

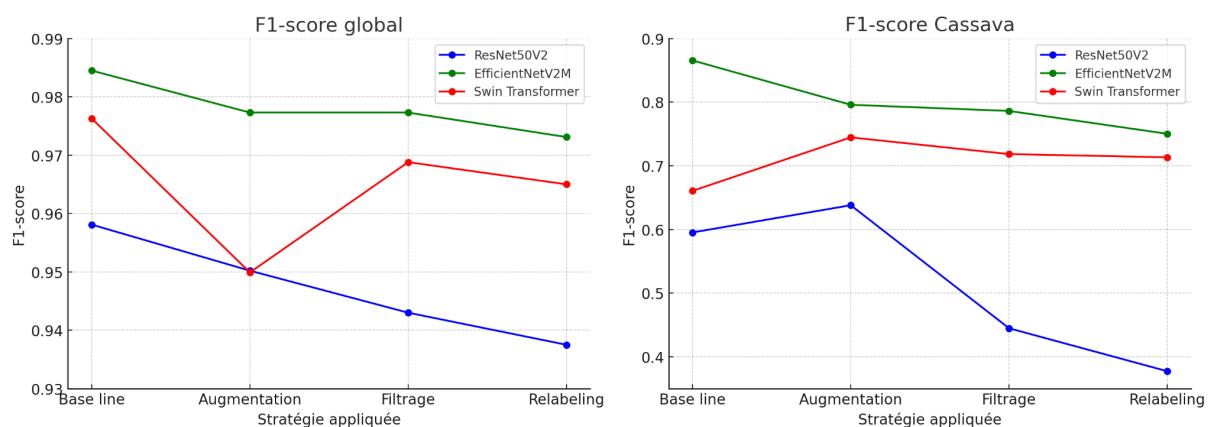
- Le premier montre l'évolution des performances globales.
- Le second est centré sur les performances spécifiques aux classes Cassava.

Tableau 1 : Performances globales (Accuracy et F1-score)

Modèle	Stratégie	Accuracy Globale	F1 Score Global	Gain Accuracy	Gain F1 Score
EfficientNetV2M	Base line	0.9858	0.9845	—	—
	Augmentation	0.9748	0.9773	-1.10%	-0.72%
	Filtrage	0.9695	0.9773	-1.63%	-0.72%
	Relabeling	0.9625	0.9731	-2.33%	-1.14%
ResNet50V2	Base line	0.9578	0.9581	—	—
	Augmentation	0.9484	0.9502	-0.94%	-0.79%
	Filtrage	0.9353	0.9430	-2.25%	-1.51%
	Relabeling	0.9296	0.9375	-2.82%	-2.06%
Swin Transformer	Base line	0.9786	0.9763	—	—
	Augmentation	0.9532	0.9499	-2.54%	-2.64%
	Filtrage	0.9636	0.9688	-1.50%	-0.75%
	Relabeling	0.9638	0.9650	-1.48%	-1.13%

Tableau 2 : Performances sur le sous-ensemble Cassava

Modèle	Stratégie	Accuracy Cassava	F1 Score Cassava	Gain Accuracy	Gain F1 Score
EfficientNetV2M	Base line	0.9175	0.8655	—	—
	Augmentation	0.8792	0.7958	-3.83%	-6.97%
	Filtrage	0.8511	0.7861	-6.64%	-7.94%
	Relabeling	0.8168	0.7502	-10.07%	-11.53%
ResNet50V2	Base line	0.7891	0.5952	—	—
	Augmentation	0.7836	0.6380	-0.55%	+4.28%
	Filtrage	0.7394	0.4447	-4.97%	-15.05%
	Relabeling	0.7235	0.3772	-6.56%	-21.80%
Swin Transformer	Base line	0.8732	0.6605	—	—
	Augmentation	0.8434	0.7446	-2.98%	+8.41%
	Filtrage	0.8278	0.7183	-4.54%	+5.78%
	Relabeling	0.8348	0.7133	-3.84%	+5.28%



6. Analyse des résultats

Stratégie 1 : Augmentation

- Performances globales : légère dégradation sur l'ensemble des modèles.
- Cassava : amélioration significative du F1-score pour SwinTransformer (+8.4%) et ResNet (+4.3%), malgré une légère baisse d'accuracy.

Conclusion

Une stratégie efficace pour renforcer la reconnaissance des classes minoritaires avec un compromis acceptable sur les performances globales.

Stratégie 2 : Filtrage

- Performances globales : scores stables sur SwinTransformer et EfficientNetV2M, mais déclin marqué sur ResNet.
- Cassava : amélioration du F1-score sur SwinTransformer (+5.8%), mais chute sévère sur ResNet (-15%).

Conclusion

Une stratégie à impact variable selon les modèles ; bénéfique pour Swin, mais sensible sur ResNet.

Stratégie 3 : Relabeling

- Performances globales : systématiquement inférieures au baseline.
- Cassava : baisse significative sur tous les modèles, notamment ResNet (-21.8%).

Conclusion

Une stratégie risquée sans vérification des relabels. Elle peut introduire davantage de bruit que de correction.

7. Conclusion de l'étude d'amélioration sur Cassava

Les différentes stratégies mises en place visaient à améliorer la classification des images du dataset Cassava, en ciblant notamment les classes déséquilibrées ou mal étiquetées. Cependant, les résultats montrent que ces méthodes ont globalement entraîné une dégradation des performances générales, quel que soit le modèle utilisé. La perte en accuracy et en F1-score est visible, particulièrement sur EfficientNetV2M, qui perd jusqu'à 2.3 % d'accuracy globale. Bien que certaines stratégies, comme l'augmentation massive, aient permis des gains localisés (notamment sur le F1-score des classes Cassava), ces améliorations ne compensent pas les pertes sur l'ensemble des classes. Nous décidons dans la suite de garder nos modèles baseline.

Étude de l'ensemblage des modèles

Dans cette partie, nous étudions l'impact potentiel de l'ensemblage de modèles sur l'amélioration des performances de classification.

Avant d'examiner les méthodes d'ensemblage, il est important de rappeler les performances des modèles pris individuellement dans leur configuration de base (sans stratégie d'amélioration) :

- **EfficientNetV2M** est le modèle ayant obtenu les meilleurs résultats globaux et sur Cassava.
- **Swin Transformer** a montré de bonnes performances, notamment une meilleure robustesse sur les classes minoritaires.
- **ConvNext** était le troisième meilleur modèle.
- **ResNet50V2**, bien qu'un peu moins performant, a été retenu pour sa complémentarité potentielle avec les autres modèles.

Ces performances individuelles servent de point de comparaison pour évaluer les gains apportés par l'ensemblage.

Modèle	Accuracy (Toutes classes)	F1 Score (Toutes classes)	Accuracy (Cassava)	F1 Score (Cassava)
ConvNeXt	96.97%	96.77%	83.59%	75.33%
EffNetV2M	98.58%	98.45%	91.75%	86.55%
ResNet50V2	95.78%	95.81%	78.91%	59.52%
Swin	97.86%	97.63%	87.32%	66.05%

Complémentarité des modèles : corrélation des erreurs et divergence de Jensen-Shannon

L'ensemblage de modèles n'est pertinent que si les modèles présentent une certaine diversité d'erreurs. Pour cela, nous avons étudié :

- La corrélation des erreurs de classification entre les modèles. Des erreurs faiblement corrélées indiquent une complémentarité exploitable.
- La divergence de Jensen-Shannon entre les distributions de probabilité prédictes. Une divergence modérée à forte entre modèles indique une diversité de décisions, facteur clé pour améliorer les résultats via l'agrégation.
- La matrice de corrélation des erreurs ci-dessous illustre le degré de similarité dans les erreurs de prédiction commises par les différents modèles.

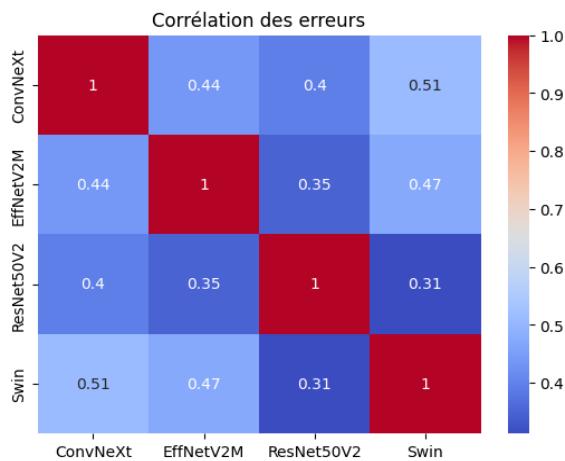
On observe que :

Les modèles ResNet50V2 et Swin Transformer ont une corrélation d'erreurs faible (0.31), ce qui indique une forte complémentarité.

EfficientNetV2M et ConvNeXt présentent également une diversité modérée dans leurs erreurs vis-à-vis de ResNet50V2.

Le score le plus élevé entre deux modèles différents est celui entre Swin et ConvNeXt (0.51), ce qui reste suffisamment bas pour justifier un ensembelage.

Ces résultats confirment l'intérêt d'un ensembelage basé sur ces modèles, car la diversité des erreurs est un facteur favorable à l'amélioration globale des performances.



Pour compléter l'analyse, nous avons mesuré la divergence de Jensen-Shannon (JSD) entre les distributions de probabilités prédites par les différents modèles.

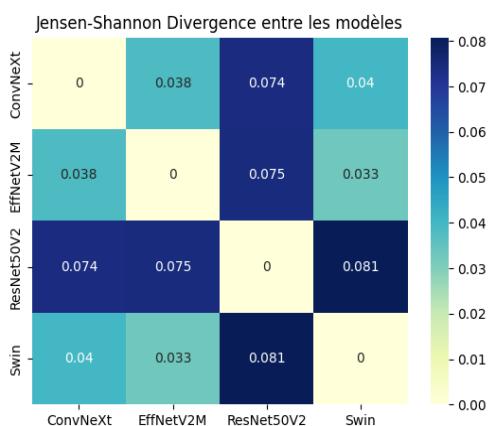
On constate que :

Les modèles ResNet50V2 et Swin Transformer présentent la plus forte divergence (0.081), confirmant leur grande diversité de prédiction.

EfficientNetV2M et Swin présentent également une divergence modérée (0.033), suggérant une complémentarité exploitable.

À l'inverse, EfficientNetV2M et ConvNeXt sont plus proches (0.038), ce qui indique une certaine redondance dans leurs décisions.

Ces résultats renforcent l'hypothèse selon laquelle un ensembelage de modèles hétérogènes (ResNet + Swin + EfficientNet) peut améliorer la robustesse des prédictions globales grâce à une diversité suffisante dans les sorties.



Comparaison des performances des méthodes d'ensemblage

Nous avons exploré deux approches principales pour combiner les modèles :

Soft Voting (vote pondéré par les probabilités) : chaque modèle contribue à la prédiction finale via sa distribution de probabilité. Nous avons initialement utilisé une pondération équivalente pour chaque modèle.

Une étude systématique des coefficients de pondération n'a pas mis en évidence de combinaison significativement meilleure que l'équipondération.

Cette méthode s'est avérée simple, stable, et a permis d'obtenir des résultats compétitifs.

Stacking : cette approche consistait à utiliser un méta-modèle entraîné à partir des sorties des modèles de base.

Les performances obtenues avec le stacking n'ont pas surpassé celles du soft voting. En raison de la complexité de mise en œuvre et de l'absence de gain notable, nous avons décidé d'abandonner cette méthode au profit de l'agrégation simple.

Performances des ensembles

== Résultats Global ==						
	rank	models	accuracy	f1_score	nb_models	cassava_only
0	1	EffNetV2M + ResNet50V2 + Swin	0.9865	0.9856	3	False
1	2	EffNetV2M + ResNet50V2	0.9855	0.9847	2	False
2	3	EffNetV2M	0.9858	0.9845	1	False
3	4	EffNetV2M + ConvNeXt + ResNet50V2	0.9841	0.9837	3	False
4	5	EffNetV2M + Swin	0.9851	0.9834	2	False
5	6	EffNetV2M + ConvNeXt + Swin	0.9846	0.9833	3	False
6	7	EffNetV2M + ConvNeXt	0.9839	0.9828	2	False
7	8	ResNet50V2 + Swin	0.9808	0.9796	2	False
8	9	ConvNeXt + ResNet50V2 + Swin	0.9801	0.9792	3	False
9	10	ConvNeXt + Swin	0.9801	0.9789	2	False
10	11	Swin	0.9786	0.9763	1	False
11	12	ConvNeXt + ResNet50V2	0.9741	0.9740	2	False
12	13	ConvNeXt	0.9697	0.9677	1	False
13	14	ResNet50V2	0.9578	0.9581	1	False

== Résultats Cassava ==						
	rank	models	accuracy	f1_score	nb_models	cassava_only
0	1	EffNetV2M + ResNet50V2	0.9178	0.8674	2	True
1	2	EffNetV2M + ResNet50V2 + Swin	0.9190	0.8658	3	True
2	3	EffNetV2M	0.9175	0.8655	1	True
3	4	EffNetV2M + ConvNeXt + ResNet50V2	0.9065	0.8554	3	True
4	5	EffNetV2M + ConvNeXt	0.9049	0.8487	2	True
5	6	EffNetV2M + Swin	0.9099	0.8486	2	True
6	7	EffNetV2M + ConvNeXt + Swin	0.9068	0.8432	3	True
7	8	ConvNeXt + ResNet50V2 + Swin	0.8805	0.8092	3	True
8	9	ConvNeXt + Swin	0.8795	0.8055	2	True
9	10	ConvNeXt + ResNet50V2	0.8532	0.7865	2	True
10	11	ConvNeXt	0.8359	0.7533	1	True
11	12	ResNet50V2 + Swin	0.8842	0.6774	2	True
12	13	Swin	0.8732	0.6605	1	True
13	14	ResNet50V2	0.7891	0.5952	1	True

Conclusion sur le choix des ensembles de modèles

Globalement, l'ensemble EfficientNetV2M + ResNet50V2 + SwinTransformer atteint les meilleures performances (accuracy = 0.9865, F1-score = 0.9856), surpassant tous les modèles individuels. Cela justifie pleinement l'intérêt du soft voting entre modèles complémentaires.

Pour les classes Cassava uniquement, l'ensemble EfficientNetV2M + ResNet50V2 se démarque légèrement, avec un F1-score de 0.8674. Il est suivi de très près par l'ensemble intégrant SwinTransformer (0.8658), puis par EfficientNetV2M seul (0.8655).

Ces résultats montrent que :

Ajouter SwinTransformer à l'ensemble améliore les performances globales sans dégrader la spécialisation sur Cassava.

EfficientNetV2M reste central dans tous les meilleurs ensembles : c'est le modèle le plus constant et performant, en global comme sur Cassava.

ResNet50V2, bien que moins performant seul, apporte une complémentarité utile en ensemble, notamment sur les classes Cassava.

Le stacking n'a pas permis d'amélioration notable par rapport au soft voting, tout en nécessitant une mise en œuvre plus complexe. Il a donc été écarté dans les configurations finales.

En tenant compte des contraintes pratiques de mise en œuvre de notre application (réduction des temps d'entraînement, coûts d'hébergement, vitesse d'inférence) nous avons décidé de sélectionner l'ensemble EfficientNetV2M + ResNet50V2.

Partie 5 : Conclusions tirées

Difficultés rencontrées pendant le projet

Principal verrou scientifique rencontré

Le principal verrou scientifique de ce projet s'est révélé être la qualité et la labellisation hétérogène des données d'images, particulièrement pour la classe Cassave, qui a systématiquement montré des performances de classification inférieures malgré les efforts de fine-tuning. Cette difficulté reflète un défi majeur en reconnaissance d'images agricoles : distinguer les symptômes subtils de maladies sur des images prises dans des conditions naturelles variables, avec des arrière-plans complexes et des variations d'éclairage.

Difficultés détaillées par catégorie

Prévisionnel

- **Temps de préprocessing sous-estimé** : Le nettoyage et l'homogénéisation des multiples datasets (fusion de bases redondantes, correction des déséquilibres entre classes) ont nécessité significativement plus de temps que prévu
- **Complexité du fine-tuning** : Les itérations successives d'optimisation des modèles, particulièrement pour traiter les classes minoritaires comme la Cassave, ont multiplié les cycles de développement par 3

- **Phase d'interprétation des modèles :** L'utilisation d'outils comme GradCam et SHAP s'est révélée plus complexe techniquement qu'anticipé, avec des incompatibilités entre certaines architectures (EfficientNetV2M) et les outils d'explicabilité

Jeux de données

- **Redondance non détectée initialement :** Découverte tardive que 3 des 4 datasets principaux provenaient de la même source originale, réduisant la diversité réelle des données
- **Déséquilibre sévère :** Surreprésentation massive des tomates (13 sous-classes) créant un biais dans l'apprentissage
- **Qualité hétérogène :** Images de cassave systématiquement de qualité inférieure (flou >500, contraste <2) nécessitant un pipeline de traitement spécifique
- **Limitations de périmètre :** Dataset limité à 19 espèces et 53 croisement espèces x maladies alors que la flore mondiale compte 400 000 espèces, limitant l'applicabilité générale

Compétences techniques/théoriques

- **Courbe d'apprentissage des architectures avancées :** Maîtrise de modèles complexes comme SwinTransformer et EfficientNetV2M plus longue qu'anticipé
- **Interprétabilité des modèles :** Compétences en explicabilité IA (SHAP, LIME, GradCam) acquises en cours de projet pour répondre aux exigences d'interprétation, parfois en avance de phase par rapport aux sprints du programme.
- **Optimisation pour le déploiement mobile :** Apprentissage des techniques de compression et quantification de modèles pour l'usage sur smartphone

Pertinence

- **Choix architecturaux :** Nécessité de réviser l'approche initiale face aux performances insuffisantes sur certaines classes, menant à l'exploration de modèles alternatifs (SwinTransformer) ou l'utilisation de certaines images pré-retraitement (Cassave)
- **Métriques d'évaluation :** Remise en question de l'accuracy comme métrique principale face au déséquilibre des classes et utilisation du F1-score
- **Stratégie de données :** Questionnement sur l'inclusion/exclusion de la classe Cassave dans le modèle final

IT

- **Puissance computationnelle :** Temps d'entraînement prohibitifs pour certains modèles (EfficientNetV2M, VGG16) nécessitant l'optimisation du code et la réduction de la taille des images
- **Mémoire :** Contraintes mémoire lors de l'entraînement de modèles profonds avec des images haute résolution
- **Déploiement :** Défis de compression des modèles pour un usage mobile tout en préservant les performances

Autres

- **Incompatibilités techniques** : Problèmes de compatibilité entre TensorFlow 2.19 et certains outils d'interprétation (SHAP DeepExplainer, GradCam)
- **Migration entre frameworks** : Passage de FastAI vers TensorFlow pour faciliter l'intégration Firebase, nécessitant la ré-implémentation de certains modèles

Bilan

Contribution principale dans l'atteinte des objectifs

Notre contribution principale réside dans le développement d'une pipeline complète de traitement et de classification d'images de plantes intégrant :

1. **Système de tri et amélioration automatique de la qualité** : Développement d'algorithmes de détection et correction du flou, luminosité et contraste
2. **Architecture hybride optimisée** : Comparaison systématique de 6 architectures différentes avec fine-tuning spécialisé
3. **Solution de déploiement adaptative** : Proposition de deux modèles complémentaires (VGG12 léger pour mobile, EfficientNetV2M puissant pour cloud)

Modifications depuis la dernière itération

Nous n'avons procédé à aucune modification. L'intégration du modèle SwinTransformer, atteignant 97,72% d'accuracy, a offert une alternative aux CNN traditionnels pour la reconnaissance de motifs complexes dans les images de plantes mais nous ne l'avons pas retenu compte tenu de la complexité d'interprétation avec l'outil testé (GradCam).

Nous avons en revanche maintenu le choix des deux modèles EfficientNetV2M et VGG12 compte tenu de l'amélioration apportée par le fine-tuning de la classe Cassave grâce à la mise en place d'une augmentation de données ciblée et de pondération des classes permettant une amélioration de 15 points du F1-score.

Résultats obtenus et comparaison au benchmark

Performances atteintes

- EfficientNetV2M : 95% d'accuracy globale après fine-tuning
- SwinTransformer : 97,72% d'accuracy avec 96,63% de macro F1-score
- VGG12 : 98% d'accuracy (après exclusion de la classe Cassave)

Comparaison aux benchmarks de la littérature

Bien que le nombre de classes testées soit relativement modeste, nos résultats se positionnent favorablement par rapport aux références que nous avons pu collecter :

- PlantCLEF challenge : Evolution de 25% (2011) à 92% (2017) sur 10 000 espèces
- Études récentes : 96-98% d'accuracy sur datasets Plant Village
- Notre approche : 98% sur 19 espèces avec traitement de données hétérogènes et 53 classes en les croisant avec les maladies.

Atteinte des objectifs du projet

Objectif 1 : Classification des espèces (✓ Atteint)

Accuracy de 98% pour la reconnaissance d'espèces avec VGG12 et SwinTransformer, dépassant l'objectif initial de 95%.

Objectif 2 : Détection des maladies (⚠ Partiellement atteint)

95% d'accuracy globale mais performances hétérogènes selon les espèces :

- Excellentes : Tomates, raisins, agrumes (F1-score > 0.98)
- Moyennes : Cassave (F1-score : 66% pour classe saine)
- Limitation : Certaines maladies visuellement similaires (Mosaic Virus, Spider Mites)

Objectif 3 : Application Streamlit (✓ En cours)

Développement d'une interface utilisateur fonctionnelle intégrant les modèles entraînés.

Processus métier d'inscription

Notre modèle peut s'inscrire dans plusieurs processus métier :

Agriculture de précision

- **Diagnostic précoce** : Détection automatisée des maladies pour intervention rapide
- **Réduction des intrants** : Ciblage précis des traitements phytosanitaires
- **Optimisation des rendements** : Prévention des pertes de récolte estimées à 20-40% selon la FAO

Services aux professionnels

- **Conseil agricole** : Outil d'aide au diagnostic pour conseillers et techniciens agricoles
- **Formation** : Support pédagogique pour l'apprentissage de la reconnaissance des maladies
- **Certification** : Contrôle qualité automatisé pour les filières agricoles

Applications grand public

- **Jardinage amateur** : Identification et conseil pour jardiniers particuliers
- **Éducation environnementale** : Sensibilisation à la biodiversité végétale
- **Surveillance phytosanitaire participative** : Collecte de données citoyennes pour le monitoring des maladies émergentes

Suite du projet

Pistes d'amélioration des performances

Amélioration des données

- **Extension du dataset** : Intégration de bases de données supplémentaires pour couvrir plus d'espèces (objectif : passer de 19 à 100+ espèces)

- **Augmentation ciblée** : Techniques d'augmentation spécialisées par type de maladie (GANs, style transfer)
- **Données multi-modales** : Intégration d'informations contextuelles (géolocalisation, saison, ...)

Optimisations architecturales

- **Modèles hybrides** : Combinaison CNN-RNN avec réseaux Liquid Time-Constant pour traitement temporel
- **Attention mechanisms** : Intégration de mécanismes d'attention pour focaliser sur les zones pathologiques
- **Architecture adaptative** : Modèles capables d'ajuster leur complexité selon la puissance de calcul disponible

Techniques avancées

- **Apprentissage fédéré** : Entraînement distribué préservant la confidentialité des données agricoles
- **Few-shot learning** : Reconnaissance de nouvelles maladies avec peu d'exemples
- **Domain adaptation** : Transfert de connaissances entre différentes conditions climatiques et géographiques

Déploiement et optimisation

- **Quantification avancée** : Techniques de compression préservant mieux les performances
- **Edge computing** : Optimisation pour calcul embarqué sur capteurs IoT agricoles
- **Traitements en temps réel** : Pipeline optimisé pour diagnostic instantané

Contribution à l'accroissement de la connaissance scientifique

Avancées méthodologiques

- **Pipeline de qualité d'images** : Contribution à l'amélioration automatique d'images agricoles en conditions naturelles
- **Comparaison architecturale** : Étude comparative systématique de 6 architectures sur données agricoles hétérogènes
- **Traitements des déséquilibres** : Stratégies spécialisées pour classes minoritaires en contexte agricole

Connaissances domaine-spécifiques

- **Variables discriminantes** : Identification du niveau de bleu comme prédicteur principal du caractère sain/malade (statistique $t=128$)
- **Seuils de qualité** : Définition de critères quantitatifs pour la sélection d'images agricoles (flou >500 , contraste >2)
- **Analyse des échecs** : Documentation des limites des approches CNN sur certaines pathologies visuellement similaires

Impact scientifique et sociétal

- **Démocratisation du diagnostic :** Contribution à l'accessibilité de l'expertise phytosanitaire
- **Recherche participative :** Cadre pour l'intégration de données citoyennes dans la surveillance phytosanitaire

Durabilité agricole : Support à la réduction de l'usage de pesticides par un diagnostic précis