

# Rapport technique d'évaluation

Reconnaissance de plantes



**Promotion: DS\_dec 22 Bootcamp**

Auteurs: Lucie Duhin, Nouha Taleb Salah, Dalvik Loger, Seidel Aimé Roch NDIKI

Mentors: Zakary\_Datascientest

# Table des Matières

- [Introduction au projet](#)
- [Compréhension et manipulation des données](#)
  - [Pertinence](#)
  - [Pre-processing et feature engineering](#)
- [Visualisations](#)
- [Exploration de couleur dans les données images de plantes](#)
  - [Etude de valeur moyenne de l'intensité de couleur vert pour les plantes malade et saines](#)
  - [Etude de variation de densité du noyau pour chacun des canaux de couleur](#)
  - [Conclusion](#)
- [Étapes de réalisation du projet](#)
  - [Classification du problème](#)
  - [Choix des modèles](#)
    - [1. Modèle de Machine Learning: utilisation du classifier Random Forest](#)
    - [2. Modèle de Deep Learning: classification en 2 étapes \(types de plantes puis état de la plante\)](#)
    - [3. Modèle de Deep Learning: modélisation en 1 seule étape \(classification des 38 classes\) \(Nouha\)](#)
- [Optimisation des modèles \(Dalvik\)](#)
- [Conclusion sur le projet](#)
- [Difficultés rencontrées lors du projet](#)
- [Bilan](#)
- [Suite du projet](#)
- [Bibliographie](#)
- [Annexes](#)

# Introduction au projet

## Contexte

L'accroissement fulgurant de performances dans les technologies de l'information et de la communication, le développement de l'intelligence artificielle, des techniques de machine learning et de deep learning ont touché l'ensemble de la société dans tous les domaines. C'est le cas pour l'agriculture: identifier, connaître et maîtriser les espèces, gérer et anticiper les cultures pour faire face à des risques climatiques, sanitaires et environnementaux de plus en plus fréquents sont les principales motivations des équipes de recherche.

La augmentation des techniques de diffusion de l'information, le partage des connaissances plus rapide à travers le monde, ont permis de satisfaire la curiosité du grand public via le développement des applications pour smartphones permettant la reconnaissance d'espèces et le diagnostic en ligne immédiat.

Les applications sont de plus en plus nombreuses, nous pouvons citer :

- pl@ntNet, PictureThis, PlantSnap... téléchargeables sur grands nombres de smartphone,
- e-phytia (inrae), agroscope (administration suisse), LABOCEA (laboratoire public breton)... , application visant la recherche, les exploitants agricoles et les particuliers: toutes ces applications sont utilisables en ligne, soit par diagnostic par image ou diagnostic guidé.

Le développement de ces outils possède de nombreux enjeux économiques parmi lesquels: faire face aux pénuries liées à la perte de récolte des exploitations malades, minimiser l'inflation donc meilleur encadrement du prix des fruits/légumes/céréales... .

Également des enjeux environnementaux comme anticiper les phytopathologies liées aux changements climatiques(sécheresse, pluies abondantes, augmentation des températures)... .

Enfin d'un point de vue scientifique, la détection précoce des maladies, leur identification et leur localisation va permettre le développement de nouvelles espèces plus robustes.

## Objectifs

Notre projet "Reconnaissance de plantes" est un projet de deep learning qui a pour objectif de classifier l'espèce d'une plante dans une image. Une fois la classification faite, une deuxième étape est de statuer sur l'état de cette plante (saine ou malade). L'objectif ultime étant de réussir à identifier la maladie (dans le cas d'une plante malade) pour en proposer un traitement efficace.

C'est un projet que nous allons donc réaliser en plusieurs étapes et qui va nous permettre de mettre en pratique nos connaissances en cours d'acquisition sur la classification d'image via les réseaux de neurones denses et de convolution.

# Compréhension et manipulation des données

## Exploration des données

Pour mener à bien notre projet, nous avons à notre disposition plusieurs sources de données que nous avons explorées. Le but de cette exploration étant de comprendre les données disponibles, être critique vis à vis du contenu et sélectionner la/les source(s) qui constitueront le point de départ de notre modélisation.

L'ensemble des sources exploitées est disponible sur le site de partage Kaggle aux liens suivants:

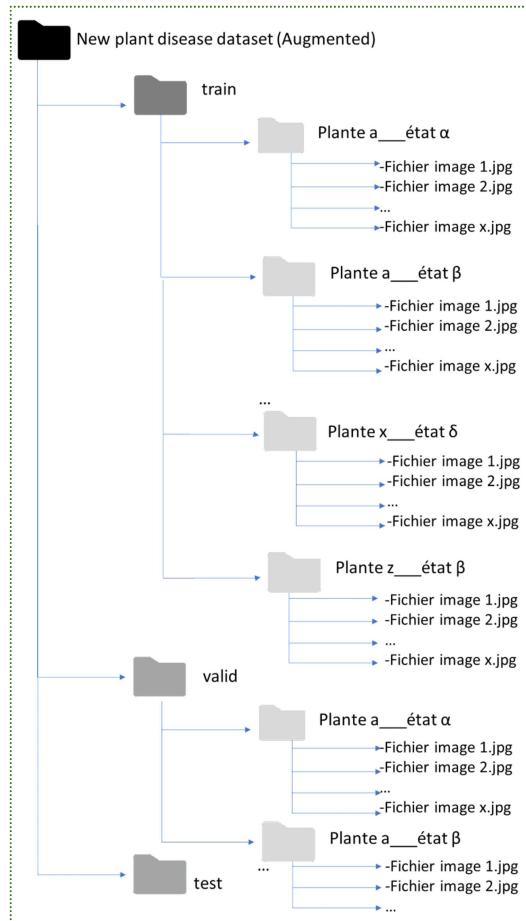
Maladie sur les plantes :

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

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

### A. new plant disease dataset

Cette première source de données est organisée sous forme de dossiers et sous-dossiers. sous le schéma suivant:



Ce dataset (New plant disease dataset (Augmented)) contient 3 répertoires principaux (train, valid et test), contenant chacun des sous-répertoires (ou sous dossiers) liés à l'espèce de plante et son état. Il existe un sous-dossier pour chaque maladie (ou état sain) identifiée. Chaque sous-dossiers de plante et son état contient une liste de fichiers image au format .jpg . nous en ferons une description visuelle dans la prochaine partie de ce rapport

L'ensemble des sous dossiers contient un total de 87867 fichiers image au format .jpg. Ce dataset ne contient pas de fichier .csv.

En voici la répartition dans le tableau ci-dessous:

| DATASET                   | RÉPERTOIRE PRINCIPAL | 38 SOUS DOSSIERS<br>classe de plantes__état: [nbre de fichiers image]   |
|---------------------------|----------------------|---|
|                           |                      |   |
| new plant disease dataset | train                | Raspberry__healthy': 1781, 'Apple__Apple_scab': 2016,<br>'Peach__Bacterial_spot': 1838, 'Tomato__Spider_mites<br>Two-spotted_spider_mite': 1741, 'Peach__healthy': 1728,<br>'Tomato__Early_blight': 1920, 'Potato__Late_blight': 1939,<br>'Corn_(maize)__Common_rust_': 1907, 'Strawberry__healthy':<br>1824, 'Apple__Black_rot': 1987, 'Soybean__healthy': 2022,<br>'Corn_(maize)__Northern_Leaf_Blight': 1908,<br>'Cherry_(including_sour)__Powdery_mildew': 1683,<br>'Apple__healthy': 2008, 'Pepper_bell__healthy': 1988,<br>'Corn_(maize)__Cercospora_leaf_spot_Gray_leaf_spot': 1642,<br>'Tomato__Septoria_leaf_spot': 1745,<br>'Tomato__Tomato_mosaic_virus': 1790,<br>'Tomato__Tomato_Yellow_Leaf_Curl_Virus': 1961,<br>'Corn_(maize)__healthy': 1859, 'Tomato__Target_Spot': 1827,<br>'Strawberry__Leaf_scorch': 1774,<br>'Grape__Leaf_blight_(Isariopsis_Leaf_Spot)': 1722,<br>'Grape__Esca_(Black_Measles)': 1920, 'Tomato__Late_blight':<br>1851, 'Orange__Haunglongbing_(Citrus_greening)': 2010,<br>'Squash__Powdery_mildew': 1736,<br>'Cherry_(including_sour)__healthy': 1826, 'Grape__Black_rot':<br>1888, 'Tomato__Bacterial_spot': 1702, 'Tomato__healthy':<br>1926, 'Potato__Early_blight': 1939, 'Potato__healthy': 1824,<br>'Pepper_bell__Bacterial_spot': 1913, 'Grape__healthy': 1692,<br>'Blueberry__healthy': 1816, 'Tomato__Leaf_Mold': 1882,<br>'Apple__Cedar_apple_rust': 1760 |
|                           | valid                | Tomato__Tomato_Yellow_Leaf_Curl_Virus': 490,<br>'Tomato__Tomato_mosaic_virus': 448,<br>'Apple__healthy': 502,<br>'Tomato__Spider_mites Two-spotted_spider_mite': 435,<br>'Grape__healthy': 423,<br>'Apple__Apple_scab': 504,<br>'Tomato__Early_blight': 480,<br>'Grape__Esca_(Black_Measles)': 480,<br>'Soybean__healthy': 505,<br>'Tomato__Septoria_leaf_spot': 436,<br>'Grape__Leaf_blight_(Isariopsis_Leaf_Spot)': 430,<br>'Orange__Haunglongbing_(Citrus_greening)': 503,<br>'Peach__Bacterial_spot': 459,<br>'Apple__Black_rot': 497,<br>'Strawberry__Leaf_scorch': 444,<br>'Strawberry__healthy': 456,<br>'Squash__Powdery_mildew': 434,<br>'Pepper_bell__healthy': 497,<br>'Cherry_(including_sour)__healthy': 456,<br>'Pepper_bell__Bacterial_spot': 478,<br>'Cherry_(including_sour)__Powdery_mildew': 421,<br>'Peach__healthy': 432,<br>'Corn_(maize)__Common_rust_': 477,<br>'Raspberry__healthy': 445,<br>'Corn_(maize)__healthy': 465,<br>'Blueberry__healthy': 454,<br>'Tomato__Leaf_Mold': 470,<br>'Tomato__Target_Spot': 457,<br>'Tomato__healthy': 481,<br>'Corn_(maize)__Northern_Leaf_Blight': 477,  |

|  |      |  |
|--|------|--|
|  |      | 'Tomato__Late_blight': 463,<br>'Tomato__Bacterial_spot': 425,<br>'Corn_(maize)__Cercospora_leaf_spot_Gray_leaf_spot': 410,<br>'Potato__Early_blight': 485,<br>'Potato__healthy': 456,<br>'Potato__Late_blight': 485,<br>'Apple__Cedar_apple_rust': 440,<br>'Grape__Black_rot': 472 |
|  | test |  |

Cette visualisation nous permet de comprendre que le dataset paraît équilibré en nombre de fichiers par sous-dossiers.

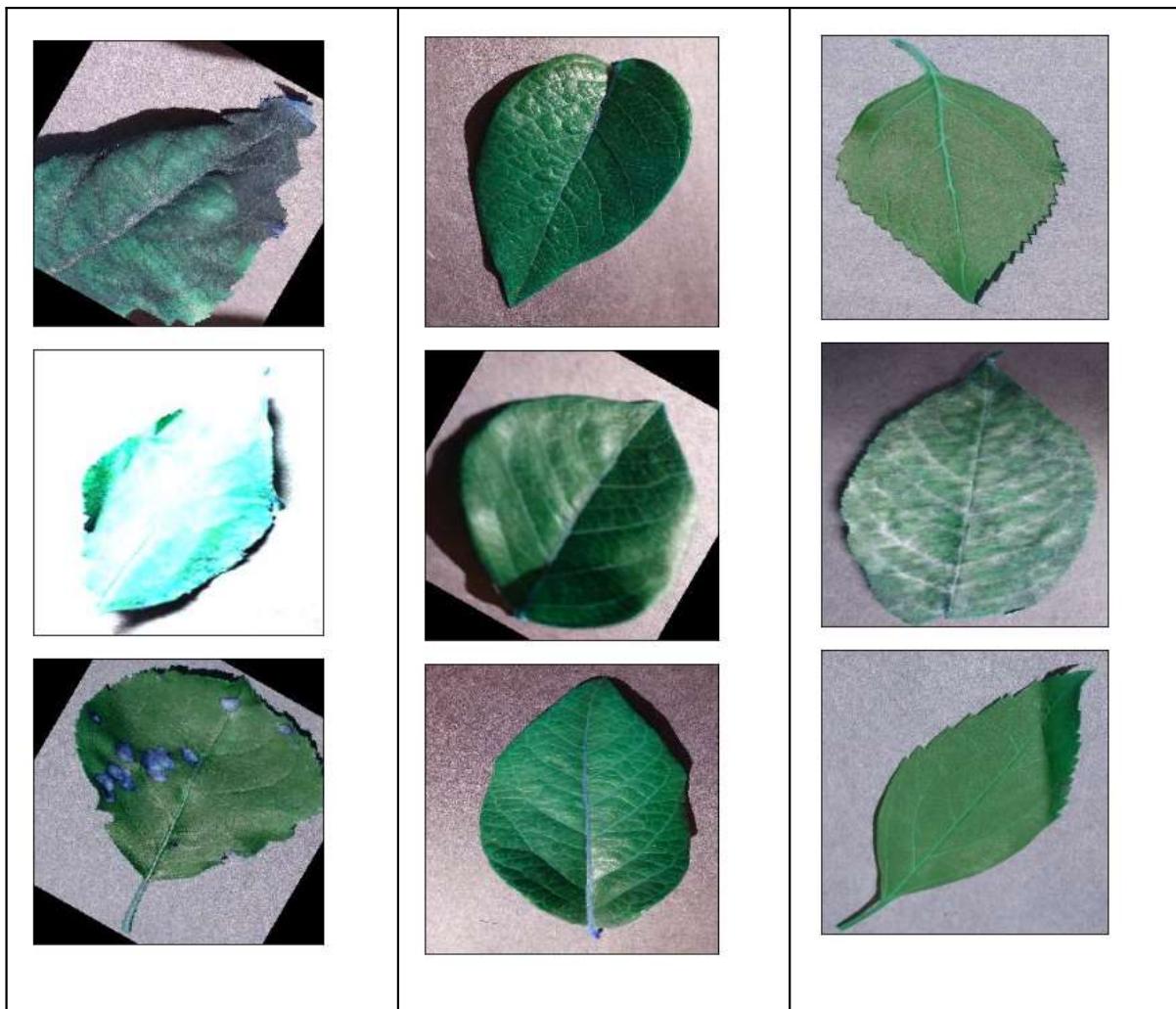
Nous avons généré la liste de sous-dossiers (38 sous-dossiers par répertoire train et valid) pour séparer les classes de plantes de leur état (sain= 'healthy' ou malade). Nous avons dénombré:

- 14 classes de plantes: ['Tomato', 'Apple', 'Grape', 'Soybean', 'Orange', 'Peach', 'Strawberry', 'Squash', 'Pepper\_bell', 'Cherry\_(including\_sour)', 'Corn\_(maize)', 'Raspberry', 'Blueberry', 'Potato']
- 26 sous-dossiers de plantes malades
- 12 sous-dossiers de plantes saines ('healthy')

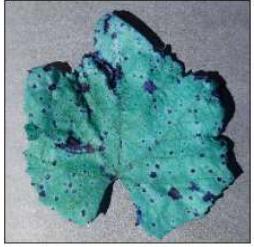
Cette répartition est identique pour le dossier valid.

Voici quelques exemples de fichiers images contenus dans le répertoire Train. Les fichiers ont été retenus par randomisation dans une boucle par catégorie de plantes.

| Classe Apple | Classe Blueberry | Classe Cherry |
|--------------|------------------|---------------|
|              |                  |               |



|             |              |               |
|-------------|--------------|---------------|
| Classe Corn | Classe Grape | Classe Orange |
|-------------|--------------|---------------|

|    |    |    |
|---|---|---|
|    |    |    |
|    |    |    |
| Classe Peach  | Classe Pepper   | Classe Potato   |
|  |  |  |
|  |  |  |
|  |  |  |
| Classe raspberry  | Classe Soybean  | Classe Squash   |

|   |   |   |
|---|---|---|
|    |    |  |
|    |    |  |
|    |    |  |
| <b>Classe Strawberry</b>  | <b>Classe Tomato</b>  |   |
|  |  |   |
|  |  |   |
|  |  |   |

Le dossier test:

Concernant le dossier test, celui-ci n'est pas organisé sous forme de sous dossier, puisque celui-ci ne doit pas contenir d'information risquant de biaiser le résultat de notre algorithme. Ce dossier contient 33 fichiers image sous le format .jpg.

Un autre point important concernant ce dataset est qu'il contient des informations augmentées. Nous avons dédié une partie de ce rapport à l'explication de ce terme.

### Affichage des dimensions des images

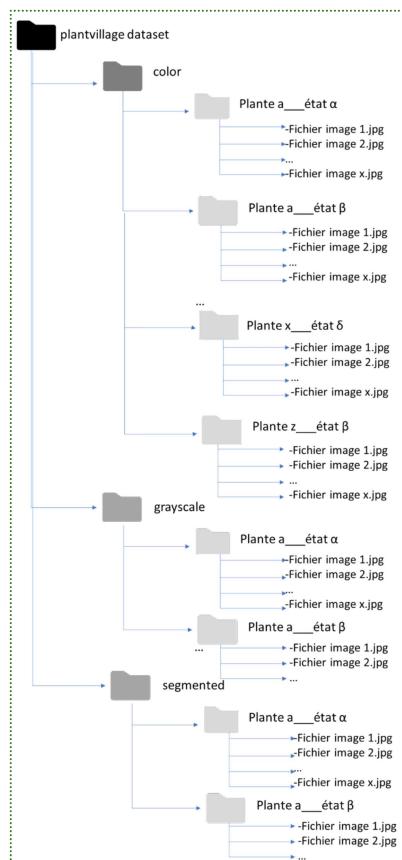
Considérons 2 types de plantes du dossier train (tomato et apple), en stockant les images de chaque catégorie de plantes dans une liste, nous pouvons extraire les dimensions de chaque image.

Celles-ci se trouvent au format (256,256,3), c'est-à-dire que chaque image du dataset est composée de 256 pixels en longueur et en hauteur sous 3 canaux différents correspondant aux couleurs Rouge, Vert et Bleu (RGB). Par la suite, nous devrons transformer ces informations de couleurs en array allant de 0 à 255 et traduisant le niveau de chaque couleur dans chaque pixel.

### B. plantvillage dataset

La source plantvillage dataset également disponible sur Kaggle, est aussi organisée sous forme de dossier et sous-dossiers mais contrairement à la première source, la répartition des images est faite d'une manière différente. Nous n'avons pas de sous-dossiers train, valid et test mais 3 sous-dossiers color, grayscale et segmented.

Voici le schéma:



Ce dataset (plantvillage) contient 3 répertoires principaux (color, grayscale, segmented), contenant chacun des sous-répertoires (ou sous dossiers) liés à l'espèce de plante et son état. Il existe un sous-dossier pour chaque maladie (ou état sain) identifiée. Chaque sous-dossiers de plante et son état contient une liste de fichiers image au format .jpg . nous en ferons une description visuelle dans la prochaine partie de ce rapport

Comme pour le dataset précédent, Nous avons effectué la même exploitation de la liste des sous dossiers et avons observé que le dataset plantvillage contient le même nombre de classe de plantes (14), le même nombre de sous-dossiers concernant les plantes malades (26) et le même nombre de sous-dossiers concernant les plantes saines (12) que le dataset new plant disease (Augmented).

Une différence a été observée, cela concerne le nombre total de fichiers image tous sous-dossiers confondus.

Ce dataset contient au total 162 916 fichiers images répartis de la façon suivante:

- \_ 54305 (color)
- \_ 54305 (grayscale)
- \_ 54306 (segmented)

Pour les 3 sous-dossiers, Il s'agit des mêmes images ayant subi des transformations:

- en niveau de gris pour grayscale,
- avec retrait de l'arrière-plan pour segmented.

Nous observons également, que le sous-dossier segmented contient un fichier image supplémentaire, l'analyse sous forme de dataframe nous a permis de montrer que ce fichier additionnel concerne la catégorie de plante "grape\_\_leaf\_blight\_(Isariopsis\_Leaf\_Spot)".

#### Affichage des dimensions des images

Nous avons vu précédemment avec le dataset New plant Disease que les dimensions de chaque image en couleurs sont au format (256,256,3). Nous avons répété cette opération de vérification pour le sous-dossier grayscale : le changement en niveau de gris ne change rien sur les dimensions d'images. nous obtenons toujours 3 canaux et des images contenant également 256 pixels en longueur et hauteur.

#### C. Comparaison des 2 datasets

L'analyse de ces 2 datasets, nous a permis de montrer que nous sommes en présence de 2 jeux de données similaires, les classes de plantes sont identiques, leurs états, le nombre de classe saines ainsi que les dimensions des fichiers images sont égaux. La différence réside dans le regroupement des images (train/Valid/test versus color/grayscale/Segmented). Par ailleurs, le dataset new plant disease contient des fichiers augmentés ce qui explique la quantité plus importante de fichier image dans le dossier train.

#### D. V2 plant seedling dataset

une troisième source de données est disponible sous le site kaggle et nommée V2 plant seedling Dataset: il s'agit de différencier les mauvaises herbes des semis qui donneront des plantes cultivées afin d'améliorer le rendement de la culture et éviter que les mauvaises herbes viennent en concurrence des cultures notamment au niveau de la ressource en nutriment et en eau. Ce dataset contient une série de photos en couleur réparties dans 12 classes.

Pour notre problématique, nous avons décidé d'écartez ce jeu de données et donc de ne pas le considérer pour le reste du projet.

## E. Définition de l'augmentation des données et quels impacts sur le dataset New plant disease (Augmented)

### a. définition (source invivoo.com)

L'augmentation des données est l'une des solutions les plus adaptées pour améliorer la puissance des algorithmes. Elle permet :

- De corriger le déséquilibre entre les différentes classes d'un jeu de données.
- Elle regroupe les techniques utilisées pour augmenter artificiellement la taille d'un groupe de données d'apprentissage en créant des versions modifiées d'images à partir des images d'apprentissage disponibles.

Les techniques d'augmentation peuvent créer des variations d'images qui peuvent améliorer la capacité des modèles d'entraînement pour généraliser ce qu'ils ont appris à de nouvelles images, ce qui améliore fortement la performance du modèle.

L'augmentation des données s'applique uniquement au jeu de données d'apprentissage et non pas au jeu de données de validation ou de test.

Les différentes augmentations des images :

- **Flip / retournement**

Un retournement (Flip) d'image signifie l'inversion des lignes ou des colonnes de pixels dans le cas d'un flip vertical ou horizontal respectivement. Les flips verticaux peuvent être obtenus grâce à une rotation d'une image de 180 degrés et à un flip horizontal.

- **Rotation :**

Une chose importante à noter à propos de cette opération est que les dimensions de l'image ne peuvent pas être conservées après la rotation: pour une image carrée, la rotation de 90° préservera la taille de l'image. Si c'est un rectangle, une rotation de 180 degrés en préservera la taille. La rotation de l'image selon des angles plus fins modifie également la taille finale de l'image.

- **Luminosité :**

La luminosité de l'image peut être augmentée soit par l'assombrissement, soit par l'éclaircissement des images, soit par les deux. L'objectif est de permettre à un modèle de généraliser à travers des images formées sur différents niveaux d'éclairage.

- **Crop (ZOOM):**

Contrairement à la mise à l'échelle, nous échantillonons aléatoirement une section de l'image d'origine. Nous dimensionnons ensuite cette section à la taille de l'image d'origine.

- **Translation (Shift):**

La translation consiste à déplacer l'image sur l'axe des X et/ou Y. Le shift horizontal translate l'image vers la gauche ou la droite tandis que le shift vertical translate l'image vers le haut ou le bas.

Cette méthode d'augmentation est très utile car la plupart des objets peuvent être localisés presque n'importe où dans l'image. Cela force le réseau de neurones à regarder partout.

- **Le bruit Gaussien :**

Le bruit gaussien, qui a une moyenne nulle, a essentiellement des points de données dans toutes les fréquences, ce qui déforme efficacement les caractéristiques de haute fréquence. Cela signifie également que les composants de fréquence inférieure (généralement, vos données prévues) sont également déformés, mais votre réseau de neurones peut apprendre à regarder au-delà de cela. L'ajout de la bonne quantité du bruit Gaussien peut améliorer la capacité d'apprentissage. Une version atténuée de ceci est le bruit du sel et du poivre, qui se

présente comme des pixels noirs et blancs répartis à travers l'image. Cela est similaire à l'effet produit par l'ajout de bruit gaussien à une image, mais peut avoir un niveau de distorsion de l'information inférieur.

#### b. impact sur le dataset New plant disease (Augmented)

Une comparaison de chaque dossier entre la source valid et la source train montre un ratio de 4 en nombre de fichiers et en poids.

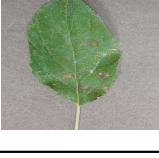
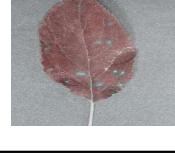
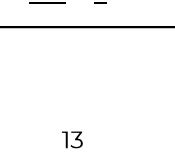
Le dossier train contient pour chaque fichier de la source valid, trois autres images augmentées. On peut observer une rotation de l'image d'angles différents.

|                | <b>Valid</b>  | <b>Train</b>                       |
|----------------|---------------|------------------------------------|
| <b>Fichier</b> | Fichier_valid | Fichier_valid+3 fichiers augmentés |
| <b>Poids</b>   | P             | 4P                                 |

D'ailleurs, un suffixe dans le nom du fichier permet de distinguer l'image source des autres images augmentées. Chaque suffixe contient le type de l'augmentation effectuée sur le fichier valid.

quelques exemples:

| classe           | fichier image initial                                | fichier image augmenté  | type d'augmentation    |
|------------------|--|---|------------------------|
| Grape__healthy   | 0ac4ff49-7fbf-4644-98a4-4dc596e2fa87__Mt.N.V_HL 9004 | 0ac4ff49-7fbf-4644-98a4-4dc596e2fa87__Mt.N.V_HL 9004_90deg          | <b>Rotation 90deg</b>  |
| Apple__Black_rot | 00e909aa-e3ae-4558-9961-336bb0f35db3__JR_FrgE.S 8593 | 00e909aa-e3ae-4558-9961-336bb0f35db3__JR_FrgE.S 8593_270deg         | <b>Rotation 270deg</b> |
| Peach__healthy   | 0a2ed402-5d23-4e8d-bc98-b264aeaa9c3fb__Rutg._HL 2471 | 0a2ed402-5d23-4e8d-bc98-b264aeaa9c3fb__Rutg._HL 2471_new30degFlipLR | <b>new30degFlipLR</b>  |

|                               |  |  |                        |
|-------------------------------|--|--|------------------------|
| <b>Strawberry_healthy</b>     | 0de0d575-5dd7-4fca-b003-d4a13354c84c89d_RS_HL 4629<br>      | 0de0d575-5dd7-4fca-b003-d4a13354c89d_RS_HL 4629_180deg<br>            | <b>Rotation 180deg</b> |
| <b>Blueberry_healthy</b>      | 0a8747da-e4f8-41bc-8a02-0e726696dfd4_RS_HL 0673<br>         | 0a8747da-e4f8-41bc-8a02-0e726696dfd4_RS_HL 0673_newPixel25<br>        | <b>newPixel25</b>      |
| <b>Tomato_healthy</b>         | 000bf685-b305-408b-91f4-37030f8e62db_GH_HL Leaf 308.1<br>  | 000bf685-b305-408b-91f4-37030f8e62db_GH_HL Leaf 308.1_flipTB<br>     | <b>flipTB</b>          |
| <b>Grape_Blk_rot</b>          | 00cab05d-e87b-4cf6-87d8-284f3ec99626_FAM_B.Rot 3244<br>   | 00cab05d-e87b-4cf6-87d8-284f3ec99626_FAM_B.Rot 3244_flipLR<br>      | <b>flipLR</b>          |
| <b>Apple_Cedar_apple_rust</b> | 0a41c25a-f9a6-4c34-8e5c-7f89a6ac4c40_FREC_C.Rust 9807<br> | 0a41c25a-f9a6-4c34-8e5c-7f89a6ac4c40_FREC_C.Rust 9807_newGRR<br>    | <b>newGRR</b>          |
| <b>Potato_healthy</b>         | e1b49d1b-165b-4f4e-8a30-735e62bc39c5_RS_HL 1839<br>       | e1b49d1b-165b-4f4e-8a30-735e62bc39c5_RS_HL 1839_new30degFlipTB<br>  | <b>new30degFlipTB</b>  |
| <b>Potato_healthy</b>         | e1b49d1b-165b-4f4e-8a30-735e62bc39c5_RS_HL 1839<br>       | e1b49d1b-165b-4f4e-8a30-735e62bc39c5_RS_HL 1839_new200degFlipLR<br> | <b>new200degFlipLR</b> |
| <b>Potato_healthy</b>         | e1b49d1b-165b-4f4e-8a30-735e62bc39c5_RS_HL 1839<br>       | e1b49d1b-165b-4f4e-8a30-735e62bc39c5_RS_HL 1839_new90degFlipLR<br>  | <b>new90degFlipLR</b>  |

|                |   |   |                       |
|----------------|---|---|-----------------------|
|                |  |  |                       |
| Potato_healthy | e1b49d1b-165b-4f4e-8a30-735e62bc39c5_RS_HL 1839                                   | e1b49d1b-165b-4f4e-8a30-735e62bc39c5_RS_HL 1839_new90degFlipTB                    | <b>new90degFlipTB</b> |

Dans le tableau ci-dessous, nous avons résumé les augmentations observées par classe de plantes (en bleu). Certaines classes n'ont subi aucune augmentation (en noir).

|    | DOSSIER  | 1-Rotation 90deg | 2-Rotation 270deg | 3-new30degFlipLR | 4-Rotation 180deg | 5-newPixel25 | 6-flipTB | 7-flipLR | 8-newGRR | 9-neww30degFlipTB | 10-neww200degFlipTB | 11-neww200degFlipTB |
|----|--|------------------|-------------------|------------------|-------------------|--------------|----------|----------|----------|-------------------|---------------------|---------------------|
| 1  | Apple Apple_scab                                 |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 2  | Apple Black_rot                                  |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 3  | Apple Cedar_apple_rust                           |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 4  | Apple healthy                                    |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 5  | Blueberry healthy                                |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 6  | Cherry_(including_sour)_healthy                  |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 7  | Cherry_(including_sour)_Powdery_mildew           |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 8  | Corn_(maize)_Cercospora_leaf_spot_Gray_leaf_spot |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 9  | Corn_(maize)_Common_rust                         |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 10 | Corn_(maize)_healthy                             |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 11 | Corn_(maize)_Northern_Leaf_Blight                |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 12 | Grape Black_rot                                  |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 13 | Grape_Esca_(Black_Measles)                       |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 14 | Grape healthy                                    |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 15 | Grape_Leaf_blight_(Isariopsis_Leaf_Spot)         |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 16 | Orange_Haunglongbing_(Citrus_greening)           |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |
| 17 | Peach Bacterial_spot                             |                  |                   |                  |                   |              |          |          |          |                   |                     |                     |

## Pertinence

L'observation des différentes sources de données nous a permis de définir le dataset "New plant disease dataset (Augmented)" comme étant la source de données à utiliser pour l'ensemble de notre projet. Ceci car les données sources des dossiers train sont augmentées et également car les données images sont réparties en dossier train/valid/test.

A ce stade du projet, l'ensemble des classes (dossiers et sous-dossiers) nous semble pertinent.

Il convient de rappeler que les fichiers images sont répartis en sous-dossiers nommés identiquement tel qu'apparaît le nom de la plante et son état (sain ou malade (ex: Apple\_Apple\_scab, Apple\_Healthy, Apple\_Black\_rot, Apple\_Cedar\_apple\_rust). Cette information est très importante car correspond à l'étiquette de donnée pour notre modélisation.

## Limite de données:

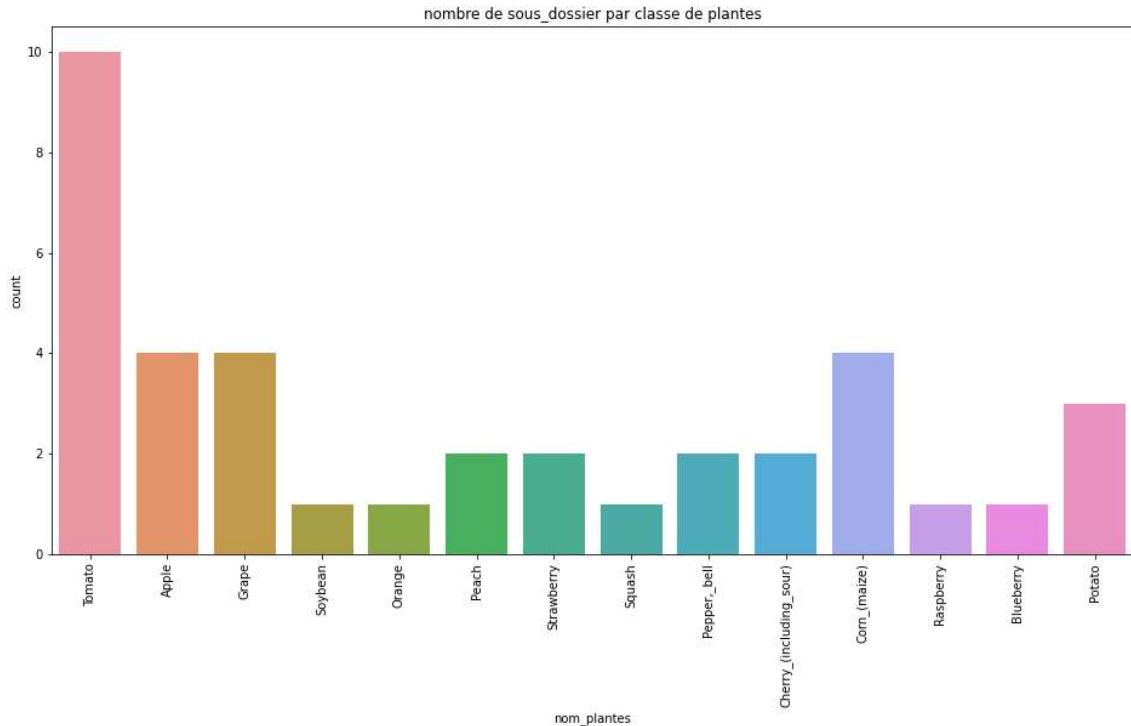
A cette étape du projet, nous n'avons pas observé de limite de données, le jeu de données semble équilibré entre les classes, présence d'un dossier test qui est séparé des données d'entraînement.

Cependant, nous avons l'habitude de travailler sur la base de fichier .csv, ce qui n'est pas le cas dans notre projet, puisqu'aucun fichier .csv n'est présent dans la database.

A l'inverse de la limite de données, le nombre total de fichiers image se situant aux alentours de 87 000, nous pouvons nous questionner sur la capacité de nos machines à pouvoir entraîner l'ensemble du dataset sur une problématique donnée.

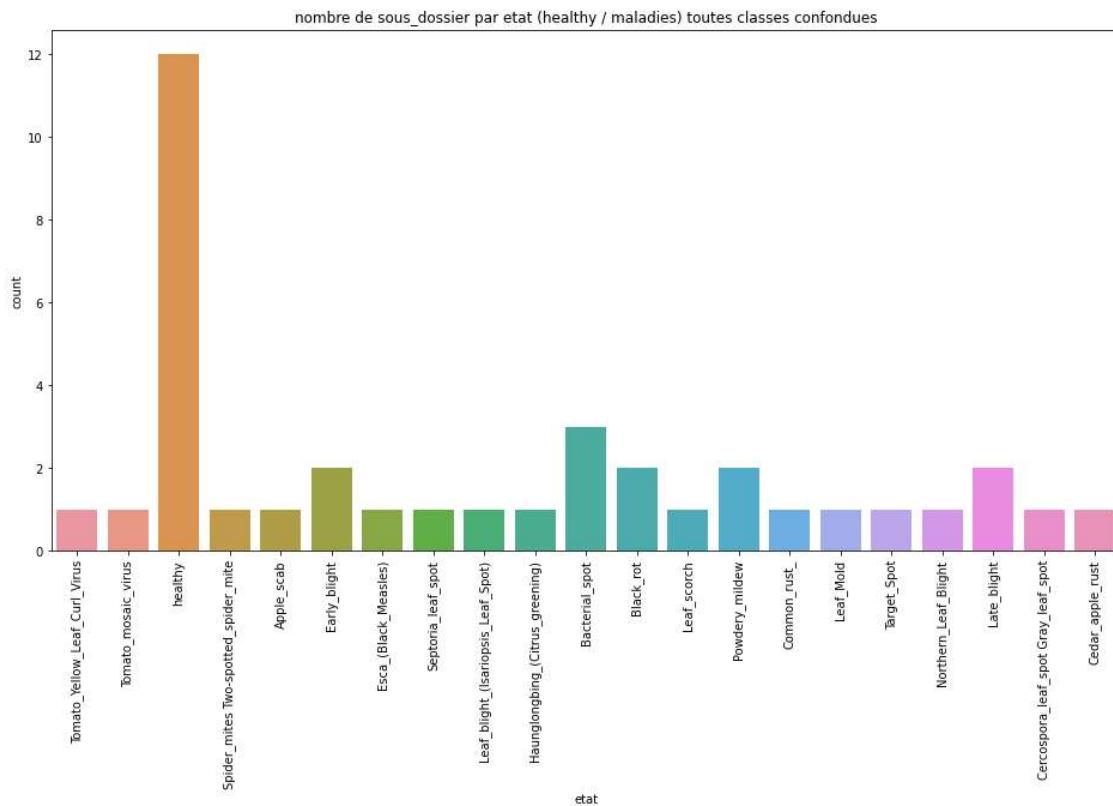
## Visualisations

Dans cette partie de visualisation des données, nous avons choisi de faire les représentations à travers la source New plant disease (Augmented) (nous avons vu précédemment que le dataset plantvillage est constitué des même classes de plantes). Une première étape consiste à visualiser le nombre de sous-dossiers par classe de plantes:



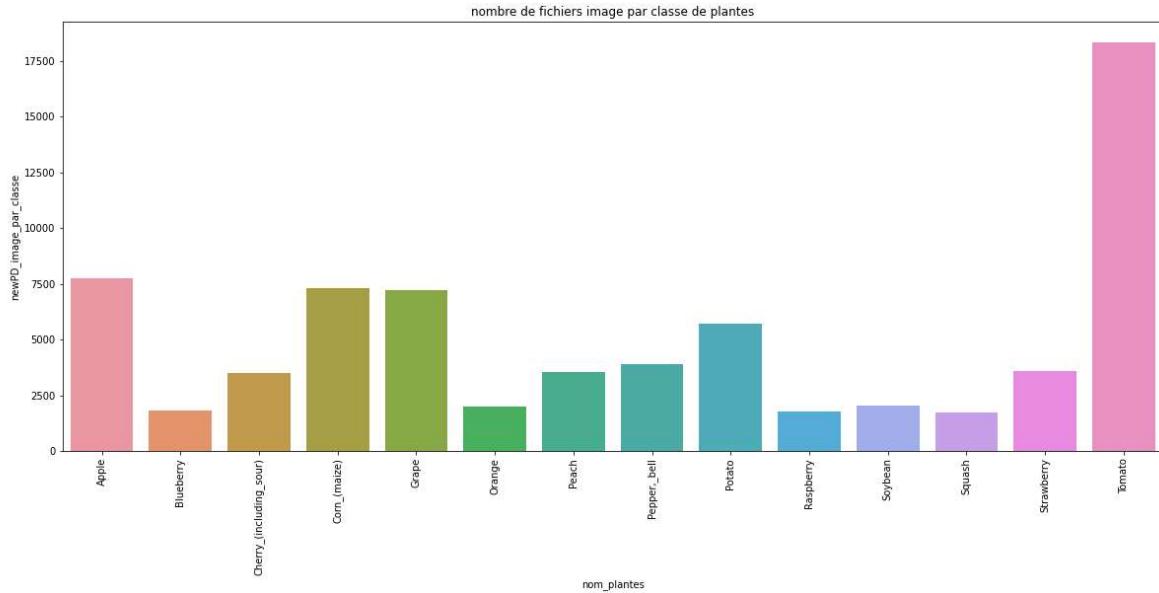
Le graphe, nous permet de constater que la classe des tomates est sur-représentée par rapport aux autres classes (10 sous dossiers), ce qui permet de conclure que cette classe doit avoir plus de maladies que les autres dans notre dataset.

Une deuxième visualisation, nous permet de comparer le nombre de sous dossiers par état (c'est à dire healthy versus les malades).



Grâce au graphique nous observons un total de 12 sous-dossiers pour les plantes catégorisées "healthy", cependant, nous savons que notre dataset contient 14 classes de plantes. En détaillant le tableau groupé de plantes par état nous constatons que l'orange et la courge n'ont pas de classe "healthy", c'est à considérer pour la suite du projet.

Une troisième visualisation concerne le nombre de fichier images réparti par classe de plantes:



Dans le dossier New plant disease train, nous constatons une prédominance du nombre de fichiers pour la classe "tomato" comparé aux autres classes du dossier train. Ceci montre que la classe tomato est victime de plus de maladies que les autres classes. Nous devons prendre en compte cette information pour la suite du projet. Cette observation est

également valable pour le dataset Plant village qui contient le même nombre de classe de plantes avec un nombre de fichier image moindre mais de même tendance.

## Exploration de couleur dans les données images de plantes

L'exploration des couleurs dans l'image vise à trouver des renseignements pertinents pour caractériser les plantes atteintes d'une maladie ou les plantes saines. En effet, l'exploration des couleurs dans l'image peut nous aider à décider de garder l'information de l'image ou non. En conséquence, cette information peut nous donner une indication de la réduction de l'information dans l'image. Par exemple, l'image grise contient nettement moins d'informations pour une image en couleur.

Nous commencerons par visualiser la valeur moyenne de couleur verte pour les plantes saines et pour les plantes malades.

Puis nous mesurerons les intensités moyennes pour chaque canal de couleur pour les plantes malades et pour les plantes en bonne santé.

Enfin, nous terminons avec une étude plus approfondie sur la variation des canaux de couleur dans une image pour une plante saine et l'autre malade.

### **Etude de valeur moyenne de l'intensité de couleur vert pour les plantes malade et saines**

La courbe et le tableau ci-dessous montre :

- l'intensité des valeurs vertes pour les plantes saines et les plantes malades.
- les valeurs moyennes de couleurs vertes, rouge et bleu.

### Intensité de couleur vert pour les plantes malades et les plantes saines

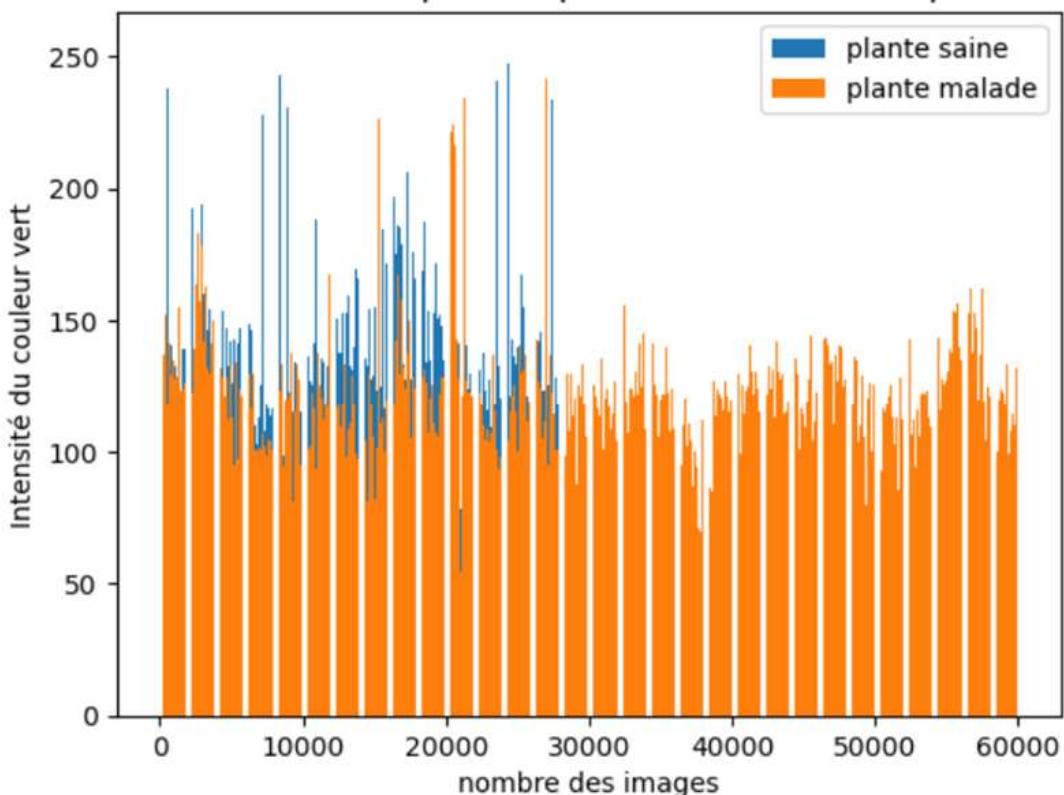


Figure 1 : Variations des couleurs des plantes malades et des plantes saines

Tableau 1: Moyenne de couleur par classe de plante malade/saine

| Plante | Moyenne de couleur | Moyenne de couleur verte | Moyenne couleur rouge | Moyenne couleur bleu |
|--------|--------------------|--------------------------|-----------------------|----------------------|
| Saine  | 129.78             | 139.38                   | 120.98                | 128.99               |
| Malade | 114.26             | 122.01                   | 103.04                | 117.74               |

D'après le tableau et graphe, nous pouvons constater que la différence entre la moyenne d'intensité de couleur verte pour une plante malade et une plante saine n'est pas significative.

En fait, l'écart en pourcentage entre la plante saine et la plante malade est d'environ 12%. De plus, d'après le graphique, nous pouvons avoir des plantes malades à une intensité de couleur verte plus que la plante saine.

Nous passons par la suite de l'étude de variations de canaux de couleur pour les plantes malades et saines.

## Etude de variation de densité du noyau pour chacun des canaux de couleur

Les canaux de couleur peuvent aider à fournir plus d'informations sur une image. Une image de l'océan sera plus bleue, tandis qu'une image d'un champ sera plus verte. Ce type d'informations peut être utile lors de la création de modèles ou de l'examen des différences entre les images.

Nous examinerons l'estimation de la densité du noyau pour chacun des canaux de couleur sur le même tracé afin de comprendre en quoi ils diffèrent.

Lorsque nous ferons ce tracé, nous verrons qu'une forme qui apparaît plus à droite signifie plus de cette couleur, tandis que plus à gauche signifie moins de cette couleur.

Les deux figures ci-dessous montrent la variation de la densité du noyau des canaux de couleur pour chaque image affichée à côté de sa courbe.

Nous avons testé deux types de plantes malade et saine. (Tomate/Pomme)

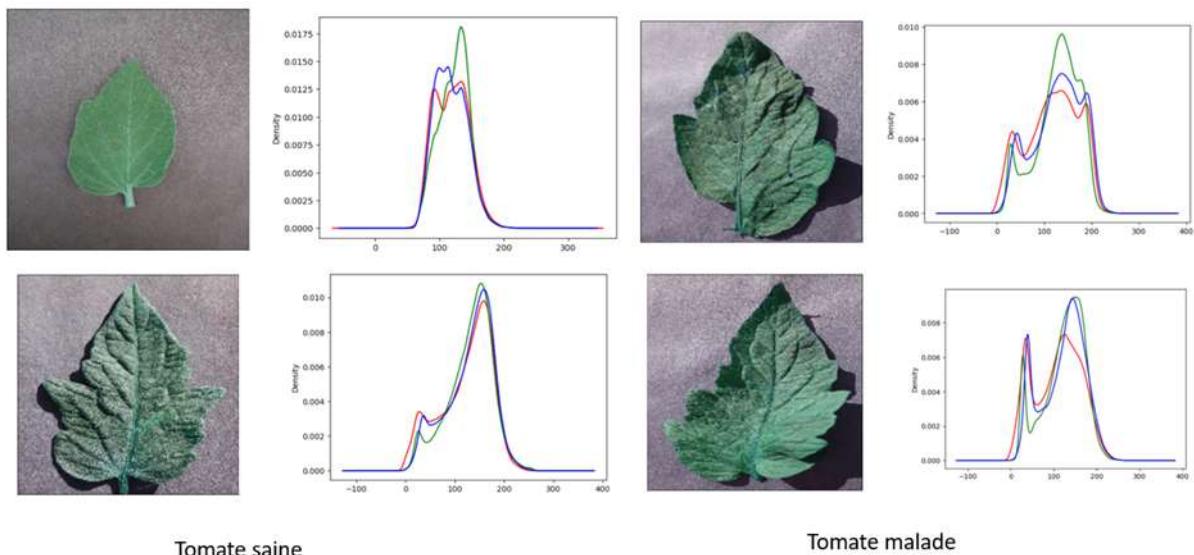


Figure 2: Variation de canaux de couleurs pour la plante de tomate saine /malade.

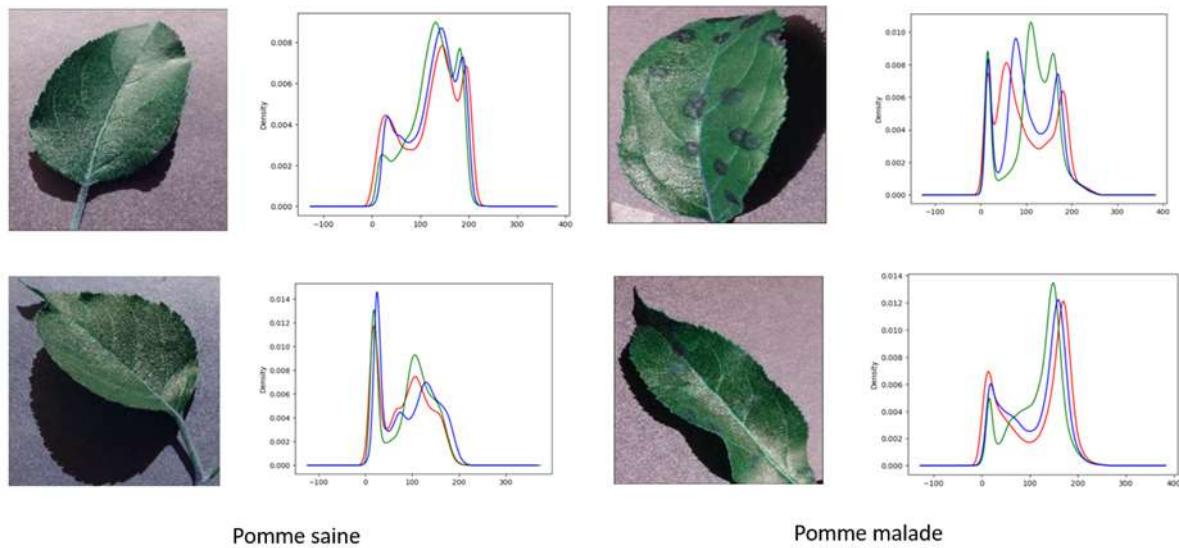


Figure 3 : Variation de canaux de couleurs pour la plante de pomme saine/malade.

Nous constatons qu'il n'y a qu'un seul sommet pour les tomates saines. En revanche, pour les tomates malades, il y a deux pics ou l'un de pic est aplati.

Toutefois, la variation de couleur n'est pas seulement visible lorsque la maladie est bien avancée (également pour les pommes)

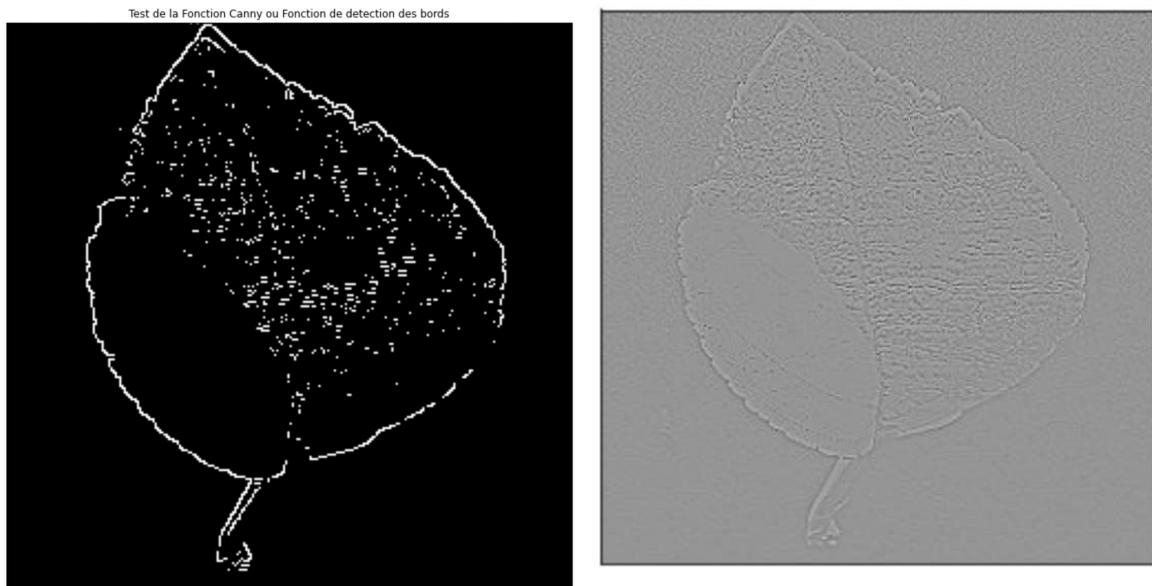
## Conclusion

Lors de l'étape suivante de la modélisation, dépendant de l'algorithme choisi, il est fort probable que nous devions travailler sur les photos grises pour réduire l'information des images.

# Etude sur la variation des canaux de couleurs

Enfin, nous terminons avec une étude plus approfondie sur la variation des canaux de couleur dans une image pour une plante saine et l'autre malade.

Celle-ci s'accompagne de la conception de différents calques de segmentation qui nous a permis

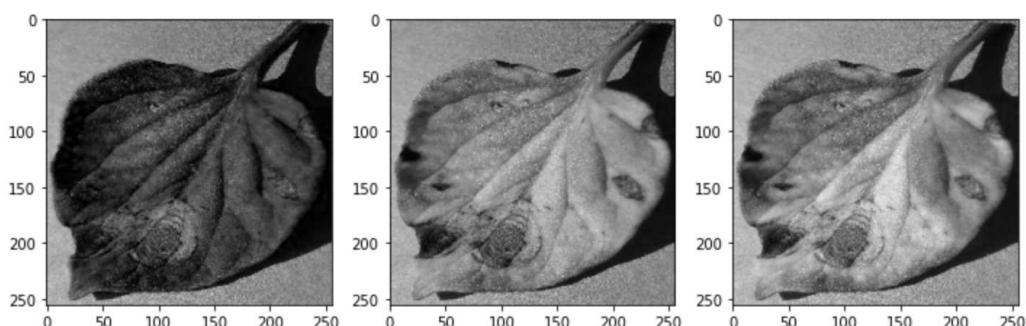


Calque de detection des formes

Calque de detection de la texture

entre autres d'extraire les contours de l'image, sa texture. Des caractéristiques qui ont été explorées dans le but d'identifier les différentes plantes..

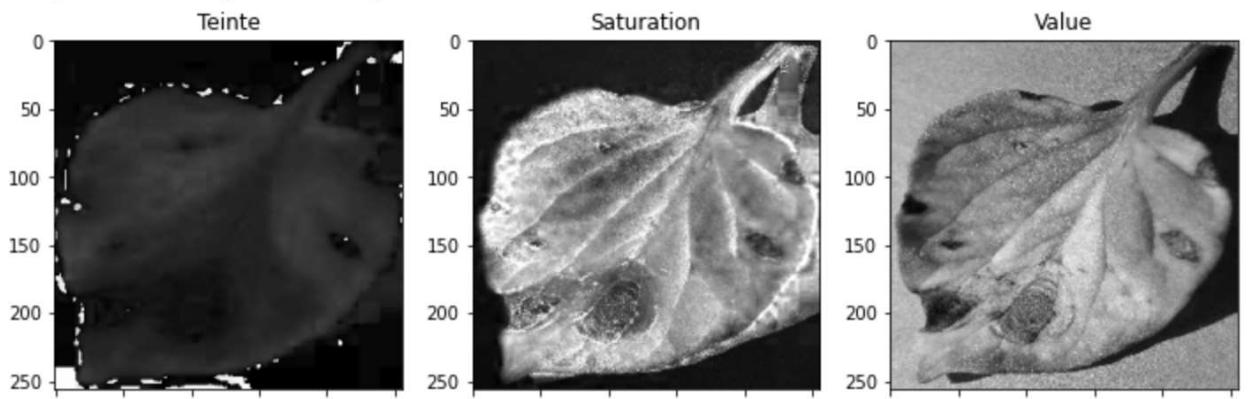
Nous avons remarqué antérieurement que les plantes atteintes de maladies avaient tendance à présenter des couleurs plus jaunes ou marron que les plantes saines. En utilisant la séparation des canaux RVB, nous avons pu mettre en évidence ces différences de teintes et utiliser cette information pour diagnostiquer les maladies.



Séparation des couleurs R, V ,B

Nous avons pu noter que les images surexposées ont une moyenne à 255 255 255, est ce une solution pour classer ces images.

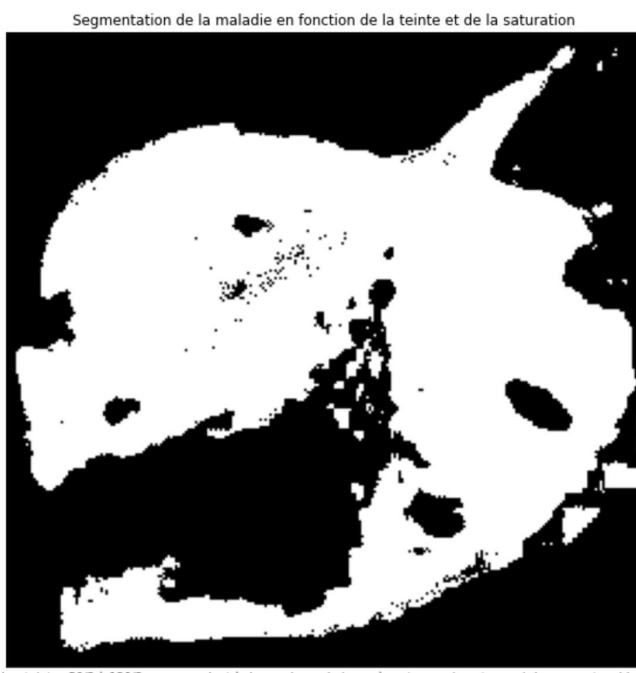
Nous avons exploré l'utilisation de calques pour extraire les valeurs, les teintes, la saturation ou



Légende

la balance des blancs des images dans le but d'augmenter notre jeu de données et d'équilibrer les classes pour améliorer les performances de prédiction. Cette démarche avait pour objectif d'améliorer les performances de prédiction en permettant à notre modèle d'analyser plus de cas et de disposer d'une représentation plus précise des différentes classes de données.

Voici un exemple avec l'utilisation d'un calque pour une image de plante malade, qui a été segmentée en différentes parties pour extraire les zones identifiées comme présentant des symptômes de maladies. Les zones présentant des couleurs anormales ont été isolées, ainsi que les parties affectées par des lésions ou des taches. Ces segments ont été séparés en différentes parties identifiables pour permettre une analyse plus précise de chaque zone et une meilleure évaluation de l'état de la plante.



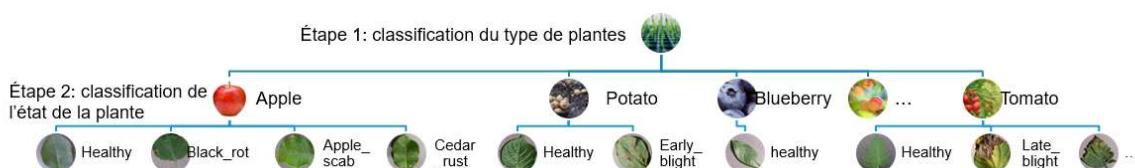
Légende

## Rendu 2 : rapport de modélisation

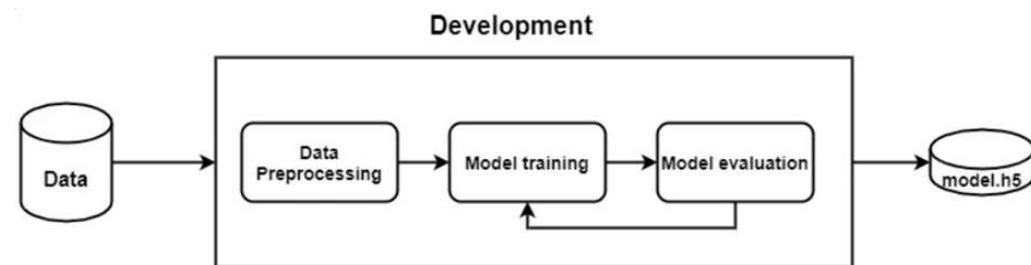
# Étapes de réalisation du projet

### Classification du problème

Considérant l'organisation de notre jeu de données, nous orientons notre projet sur une problématique de classification multi-classes par analyse d'images:



L'utilisation des réseaux de neurones convolutifs (librairie tensorflow et modules keras) va permettre de classifier ses images.



Pour cela, nous considérons plusieurs solutions de travail:

- La première étant de classifier par étapes comme dans le schéma précédent, c'est-à-dire d'abord le type de plante puis son état. Pour ce faire, nous devons considérer la réalisation de plusieurs modèles de deep learning reliés par des boucles.
- la seconde étant de réaliser une classification totale (plante et son état) en une seule étape.

Nous allons détailler ces 2 approches dans la suite de ce rapport

### Choix des modèles

1. Modèle de Deep Learning: classification en 2 étapes (types de plantes puis état de la plante)
  - a. Etape 1: Classification du type de plante

La problématique de cette étude étant de classifier les plantes, nous avons considéré dans ce modèle, l'ensemble des dossiers se terminant par “\_healthy” (correspondant au fichier des plantes saines).

- structuration des données

La structure de classification des fichiers images nous a permis de constituer un dataframe regroupant le chemin de chaque fichier image, son étiquette (correspondant au nom du dossier dans lequel est stocké l'image) et une clé numérique pour faciliter l'apprentissage des modèles.

Voici un extrait des 10 premières lignes de ce dataframe (df):

|           | filepath   | nameLabel      | label |
|-----------|--|----------------|-------|
| <b>0</b>  | ./Train healthy/Apple__healthy/0055dd26-23a7-... | Apple__healthy | 0     |
| <b>1</b>  | ./Train healthy/Apple__healthy/00907d8b-6ae6-... | Apple__healthy | 0     |
| <b>2</b>  | ./Train healthy/Apple__healthy/00907d8b-6ae6-... | Apple__healthy | 0     |
| <b>3</b>  | ./Train healthy/Apple__healthy/0098dbd9-286a-... | Apple__healthy | 0     |
| <b>4</b>  | ./Train healthy/Apple__healthy/00a6039c-e425-... | Apple__healthy | 0     |
| <b>5</b>  | ./Train healthy/Apple__healthy/00fca0da-2db3-... | Apple__healthy | 0     |
| <b>6</b>  | ./Train healthy/Apple__healthy/010125c0-e6f2-... | Apple__healthy | 0     |
| <b>7</b>  | ./Train healthy/Apple__healthy/010125c0-e6f2-... | Apple__healthy | 0     |
| <b>8</b>  | ./Train healthy/Apple__healthy/011d02f3-5c3c-... | Apple__healthy | 0     |
| <b>9</b>  | ./Train healthy/Apple__healthy/013b7c70-5e3b-... | Apple__healthy | 0     |
| <b>10</b> | ./Train healthy/Apple__healthy/017fd21b-142f-... | Apple__healthy | 0     |

En appliquant la méthode `value_counts()` sur la variable `nameLabel`, nous pouvons constater que le jeu de données est équilibré et ne nécessitera pas de traitement sur une classe particulière.

|                       |      |
|-----------------------|------|
| Soybean__healthy      | 2022 |
| Apple__healthy        | 2008 |
| Pepper,_bell__healthy | 1988 |

```

      Tomato__healthy      1926
      Corn_(maize)__healthy 1859
      Cherry_(including_sour)__healthy 1826
      Potato__healthy      1824
      Strawberry__healthy   1824
      Blueberry__healthy    1816
      Raspberry__healthy    1781
      Peach__healthy        1728
      Grape__healthy         1692
application du value_counts sur nameLabel

```

Le regroupement par valeur de la variable nameLabel, donne une vision d'ensemble du nom de la plante associé à son label numérique:

| nameLabel                        | label |
|----------------------------------|-------|
| Apple_healthy                    | 0     |
| Blueberry__healthy               | 1     |
| Cherry_(including_sour)__healthy | 2     |
| Corn_(maize)__healthy            | 3     |
| Grape__healthy                   | 4     |
| Peach__healthy                   | 5     |
| Pepper,_bell__healthy            | 6     |
| Potato__healthy                  | 7     |
| Raspberry__healthy               | 8     |
| Soybean__healthy                 | 9     |
| Strawberry__healthy              | 10    |
| Tomato__healthy                  | 11    |

- séparation du jeu de données

l'appel du sous module model\_selection de scikit learn et de la fonction train\_test\_split est l'outil communément utilisé pour séparer un jeu de données en set d'entraînement (80% du dataset) et en set de test.

Le set de test X\_test créé à partir du module tensorflow est un object tensorflow de type array ayant pour dimension (4459, 100,100,3) suite à un redimensionnement des images en 100,100 pour améliorer le temps de modélisation (par soucis de performance nous avons également testé 224,224 et 256,256).

- étude de différents modèles

Plusieurs modèles de deep learning ont pu être testés dans cette partie du projet. La différence se situant au niveau de la construction des couches du réseau de neurones de convolution.

#### \_ Modèle de transfer learning avec vgg16 / imagenet

Ce modèle d'apprentissage par transfert est un modèle de réseau de neurone convolutionnel dont les couches sont constituées de noyaux de convolution de petite taille (3x3). Imagenet correspond à la banque de données open source qui a permis l'entraînement de vgg.

| Layer (type)               | Output Shape         | Param # |
|----------------------------|----------------------|---------|
| <hr/>                      |                      |         |
| input_1 (InputLayer)       | [None, 100, 100, 3]  | 0       |
| block1_conv1 (Conv2D)      | (None, 100, 100, 64) | 1792    |
| block1_conv2 (Conv2D)      | (None, 100, 100, 64) | 36928   |
| block1_pool (MaxPooling2D) | (None, 50, 50, 64)   | 0       |
| block2_conv1 (Conv2D)      | (None, 50, 50, 128)  | 73856   |
| block2_conv2 (Conv2D)      | (None, 50, 50, 128)  | 147584  |
| block2_pool (MaxPooling2D) | (None, 25, 25, 128)  | 0       |
| block3_conv1 (Conv2D)      | (None, 25, 25, 256)  | 295168  |
| block3_conv2 (Conv2D)      | (None, 25, 25, 256)  | 590080  |
| block3_conv3 (Conv2D)      | (None, 25, 25, 256)  | 590080  |
| block3_pool (MaxPooling2D) | (None, 12, 12, 256)  | 0       |
| block4_conv1 (Conv2D)      | (None, 12, 12, 512)  | 1180160 |
| block4_conv2 (Conv2D)      | (None, 12, 12, 512)  | 2359808 |
| block4_conv3 (Conv2D)      | (None, 12, 12, 512)  | 2359808 |
| block4_pool (MaxPooling2D) | (None, 6, 6, 512)    | 0       |
| block5_conv1 (Conv2D)      | (None, 6, 6, 512)    | 2359808 |
| block5_conv2 (Conv2D)      | (None, 6, 6, 512)    | 2359808 |
| block5_conv3 (Conv2D)      | (None, 6, 6, 512)    | 2359808 |
| block5_pool (MaxPooling2D) | (None, 3, 3, 512)    | 0       |
| <hr/>                      |                      |         |
| Total params:              | 14,714,688           |         |
| Trainable params:          | 0                    |         |
| Non-trainable params:      | 14,714,688           |         |

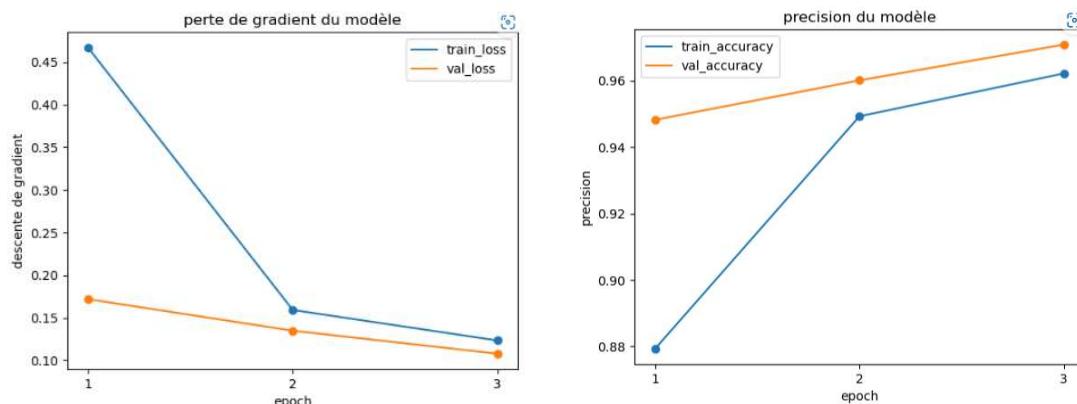
Ce modèle de base est incorporé à notre modèle séquentiel composé de plusieurs couches de convolution avec une fonction d'activation "relu" et une dernière couche dense comportant 12 unités de neurones correspondant aux 12 classes identifiées de notre problématique avec une couche d'activation "softmax".

| Model: "sequential"                               |                   |          |
|---|-------------------|----------|
| Layer (type)                                      | Output Shape      | Param #  |
| <hr/>   |                   |          |
| vgg16 (Functional)                                | (None, 3, 3, 512) | 14714688 |
| global_average_pooling2d (GlobalAveragePooling2D) | (None, 512)       | 0        |
| dense (Dense)                                     | (None, 1024)      | 525312   |
| dropout (Dropout)                                 | (None, 1024)      | 0        |
| dense_1 (Dense)                                   | (None, 512)       | 524800   |
| dropout_1 (Dropout)                               | (None, 512)       | 0        |
| dense_2 (Dense)                                   | (None, 12)        | 6156     |
| <hr/>   |                   |          |
| Total params:                                     | 15,770,956        |          |
| Trainable params:                                 | 1,056,268         |          |
| Non-trainable params:                             | 14,714,688        |          |

Afin d'éviter le surentraînement de notre modèle et d'optimiser le temps d'entraînement, des callbacks sont ajoutés à ce modèle: earlystopping et modelcheckpoint.

#### Résultat du modèle:

L'entraînement du modèle initialement prévu sur 5 epochs, s'est finalement arrêté après 3 epochs. Sur le graphique de descente de gradient en fonction des epochs, nous n'observons pas d'incident ou de phénomène de sur apprentissage. Concernant la précision du modèle, nous constatons une précision en augmentation supérieure à 0.96.



#### Interprétation des résultats:

Le rapport de classification nous indique que le modèle a une précision générale de 0.95. Il prédit avec une précision totale les classes 3 et 4 (correspondant à "Corn" et "Grape"). Il est extrêmement précis (0.97 à 0.99) pour les classes 0, 2, 7, 8 et 10 ("Apple", "Cherry", "Potato", "Raspberry" et "Strawberry").

En revanche, le modèle est moins précis (0.87 à 0.90) pour les catégories 11, 9 et 5 ("Tomato", "Soybean", "Peach").

L'analyse du rappel, nous permet de constater que le modèle confond certaines classes (0.87 à 0.89), c'est le cas des classes 0, 1 et 7. Cette analyse est d'autant plus visible sur la matrice de confusion.

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.98      | 0.89   | 0.93     | 396     |
| 1            | 0.92      | 0.87   | 0.90     | 345     |
| 2            | 0.99      | 0.92   | 0.95     | 398     |
| 3            | 1.00      | 1.00   | 1.00     | 386     |
| 4            | 1.00      | 0.96   | 0.98     | 319     |
| 5            | 0.90      | 0.99   | 0.94     | 354     |
| 6            | 0.94      | 0.98   | 0.96     | 401     |
| 7            | 0.98      | 0.87   | 0.92     | 370     |
| 8            | 0.97      | 0.95   | 0.96     | 317     |
| 9            | 0.89      | 0.97   | 0.93     | 402     |
| 10           | 0.98      | 0.97   | 0.98     | 374     |
| 11           | 0.87      | 0.99   | 0.93     | 397     |
| accuracy     |           |        | 0.95     | 4459    |
| macro avg    |           | 0.95   | 0.95     | 4459    |
| weighted avg |           | 0.95   | 0.95     | 4459    |

Rapport de classification des 12 classes de plantes

#### Analyse de la classe 0 (Apple):

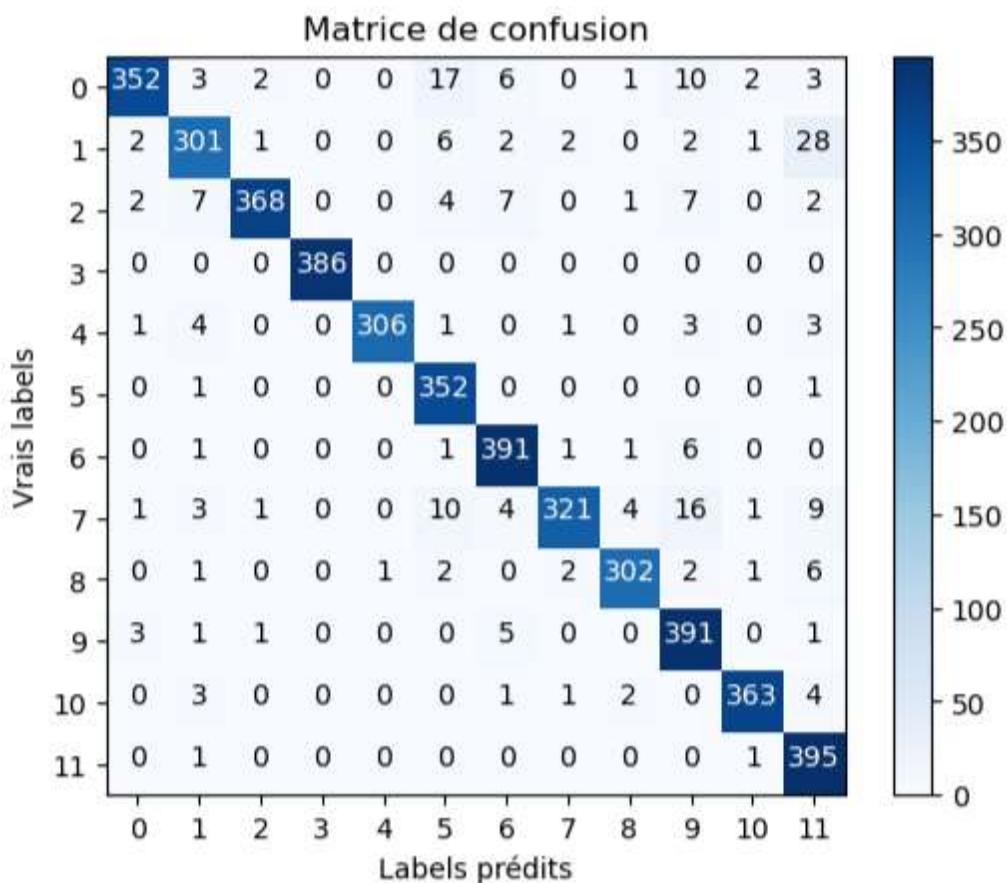
La matrice de confusion nous permet aisément de voir que le modèle confond la classe des Apple avec les classes 5 et 9 (Peach et Soybean) et dans certains cas moins nombreux avec la classe 6 (Pepper).

Analyse de la classe 1 (Blueberry):

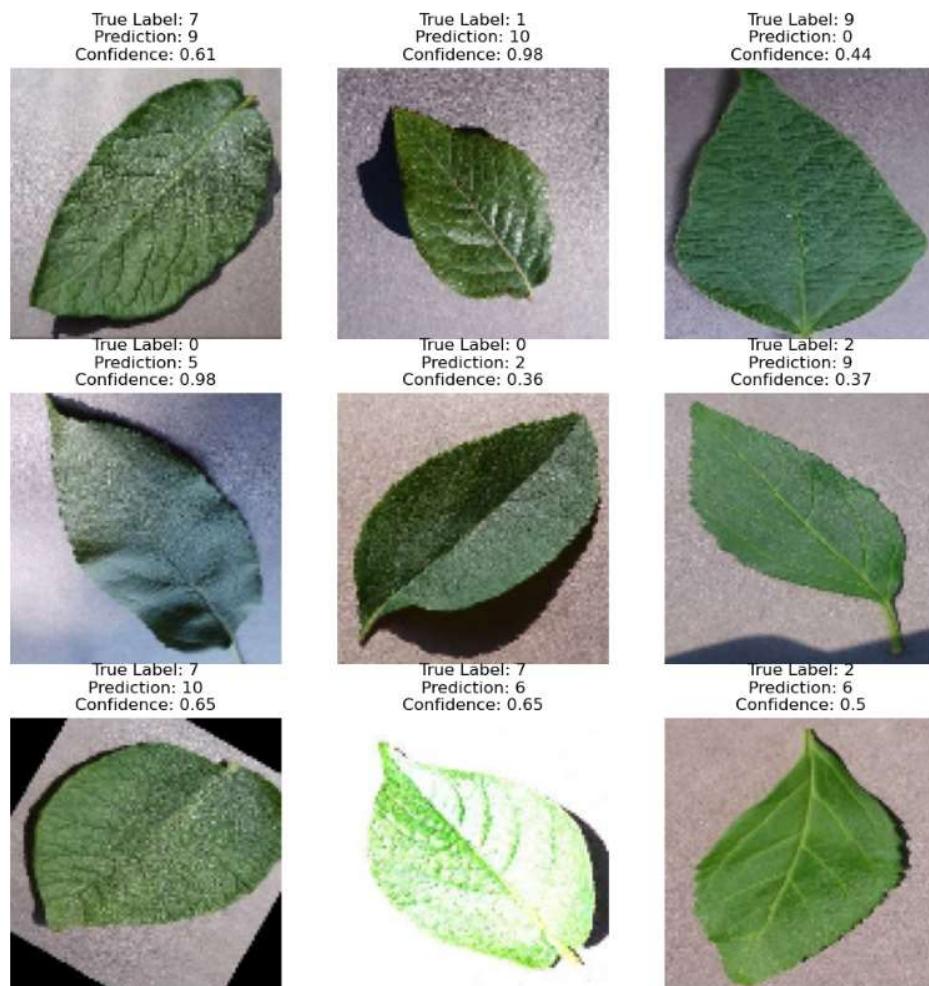
Le type de plante 1 (Blueberry) est confondu par le modèle avec la classe 11 (Tomate) et sur quelques cas avec la classe 5 (Peach).

Analyse de la classe 7 (Potato):

La classe 7 (Potato) est confondue avec la classe 5 et la classe 9 (Peach et Soybean) tout comme la classe Apple.



Quelques illustrations des erreurs du modèles:

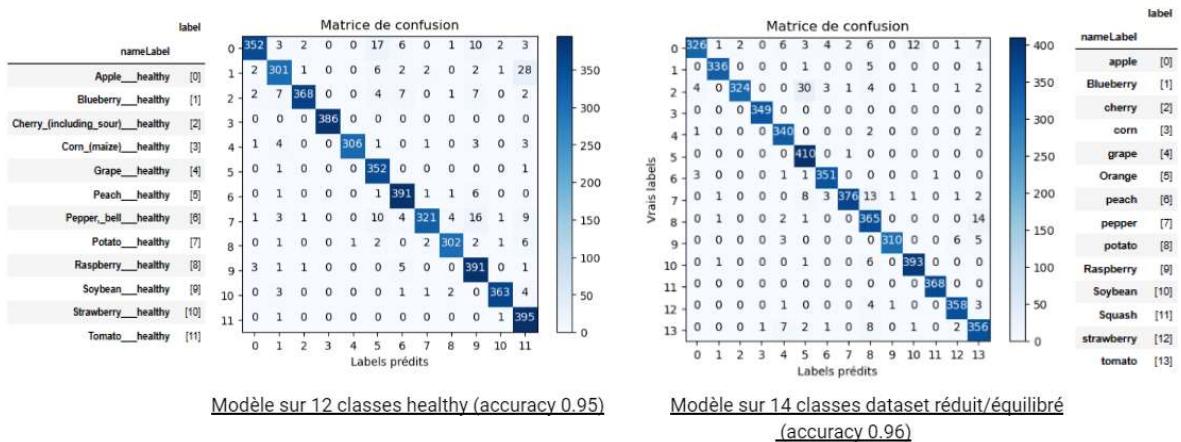


Conclusion sur le modèle vgg 16/imagenet:

Le modèle vgg16/imagenet entraîné sur les sous dossiers de plante healthy est très performant et permet d'obtenir des résultats de prédictions excellents sur certaines classes de plantes. Cependant, il serait intéressant de l'améliorer sur certaines classes en ajoutant des masques supplémentaires ou en nuançant certaines balances de couleurs / expositions.

Optimisation du modèle:

Il est aussi fort probable qu'un entraînement sur des classes de plante \_healthy uniquement puisse desservir lors de l'utilisation du modèle pour classifier une plante (saine ou malade) à partir d'une photo. C'est pour cela que nous avons également réaliser l'entraînement du modèle sur la base vgg16/imagenet sur un dataset contenant les 38 classes (plantes + état) avec un nombre de fichiers réduits mais équilibré.



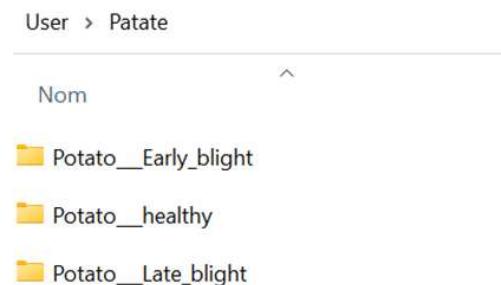
Même si la précision générale est équivalente, nous constatons une amélioration concernant les classes apple, blueberry, potato. Par contre, il semblerait que la classe cherry soit moins reconnue. L'avantage de la modélisation avec un dataset réduit permet également d'obtenir 2 classes supplémentaires (orange et Squash) qui ne sont pas représentées dans le dataset \_healthy.

### b. étape 2: classification de l'état de plante (healthy vs malade)

Dans cette problématique de classification de l'état de la plante, nous avons développé un modèle par type de plante. Les modèles sont construits à partir de réseau de convolution créé à partir de zéro.

- structure et récupération des données

Le jeu de données est installé en local sur chacun des postes de travail de l'équipe. La structure des données est très importante pour pointer les fichiers images correspondant à chaque type de plante. La structure de nos données est organisée de la même façon pour toutes les classes, en voici la structure:



L'utilisation du sous module `preprocessing.image` de keras et de la méthode “`from_directory`” (ou “`flow_from_directory`”) permet de pointer le dossier plante, de compter le nombre de fichiers image et de repérer le nombre de classes (correspondant au sous dossiers). L’attribut `class_names` va permettre d’afficher le nom des classes repérées.

exemple de Apple, 4 class\_names:

```
['Apple__Apple_scab',
 'Apple__Black_rot',
 'Apple__Cedar_apple_rust',
 'Apple__healthy']
```

Nous avons donc appliqué cette méthode de récupération pour chaque type de plante.

Nous avons vu dans la première partie du rapport d'exploitation de données, que le nombre de fichiers image par plante est important (9728 fichiers image pour Apple par ex), nous avons décidé de réduire le dataset de chaque plante, redimensionner les images puis normaliser les données (/255) pour optimiser le temps d'entraînement de nos modèles.

- étude de la modélisation

### **Architecture du modèle**

Nous allons construire notre modèle à partir de zéro. Après avoir expérimenté différents modèles, nous avons trouvé l'architecture ci-dessous plus performante.

Le modèle de CNN créé est basé sur un modèle séquentiel dans lequel nous avons ajouté:

\_ 6 couches de convolution (les données d'entrée concernant la dimension des images étant de 256,256,3), la quantité de neurone fixée à 32, la taille du noyau fixée à (3,3) .

\_ 6 couches de maxPooling permettant de réduire la quantité de donnée en utilisant une matrice (2,2)

\_ 1 couche permettant d'aplatir les données

\_ 1 couche dense de 64 neurones avec une fonction d'activation 'relu' \_1 couche dense à n\_classes (correspondant au nombre de classe par plante) avec une fonction d'activation "softmax" (car contexte de classification cette fonction va sortir les probabilités de chaque classe.)

Nous n'avons pas redimensionné les images lors du chargement, car nous avons ajouté une couche de redimensionnement dans notre modèle.

```
models.Sequential([
    layers.Conv2D(32, (3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, kernel_size=(3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, kernel_size=(3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Conv2D(64, (3,3), activation='relu'),
    layers.MaxPooling2D((2,2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])
```

### **Compilation et entraînement du modèle**

La compilation des modèles s'effectue avec l'optimizer 'Adam', et la métrique surveillée est l'accuracy. Nous avons défini un callback EarlyStopping pour éviter un ajustement excessif.

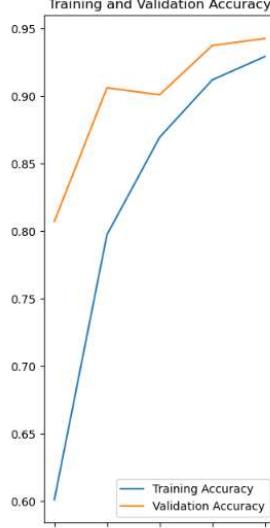
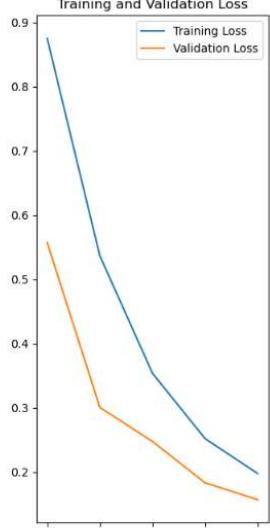
L'entraînement se fait sur 5 époques et l'historique d'entraînement est stocké dans la variable d'historique

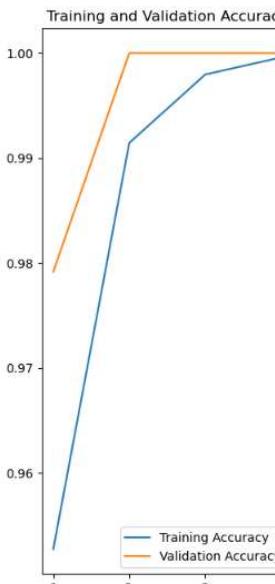
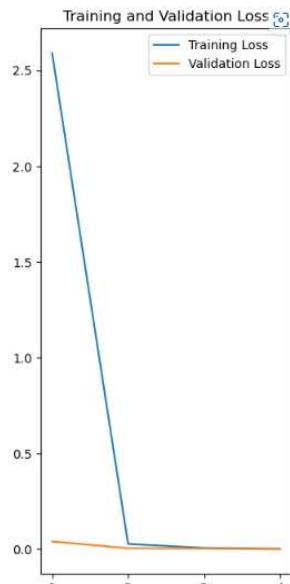
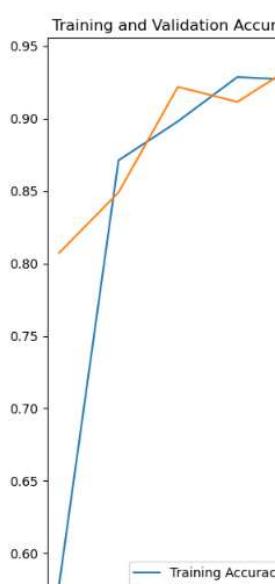
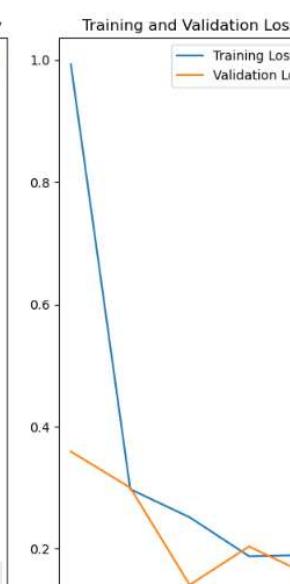
```
Epoch 3/5
156/156 [=====] - 271s 2s/step - loss: 0.0908 - accuracy: 0.9659 - val_loss: 0.1092
9740
Epoch 4/5
156/156 [=====] - 231s 1s/step - loss: 0.0872 - accuracy: 0.9669 - val_loss: 0.0581
9740
Epoch 5/5
156/156 [=====] - 259s 2s/step - loss: 0.0632 - accuracy: 0.9766 - val_loss: 0.0538
9844
```

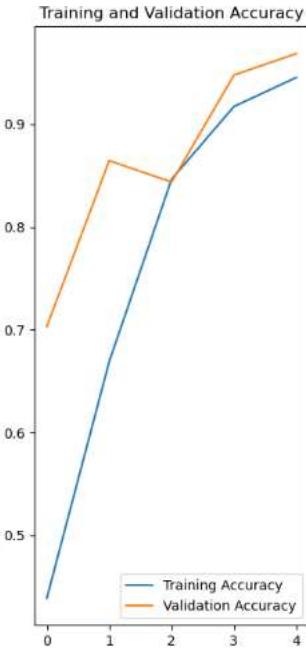
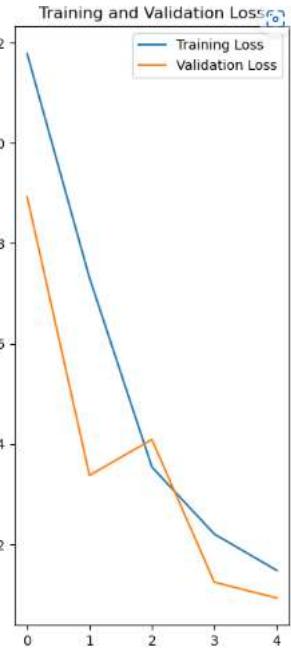
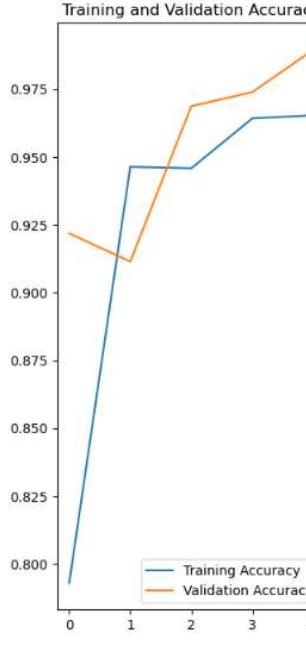
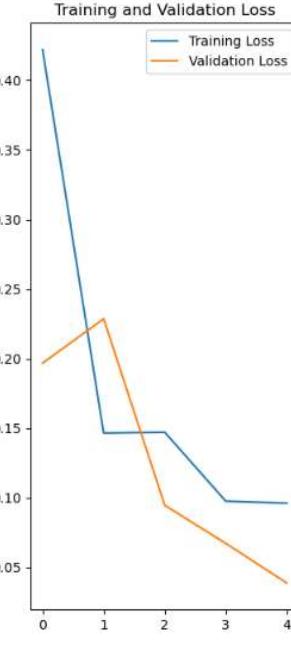
- Evaluation du modèle

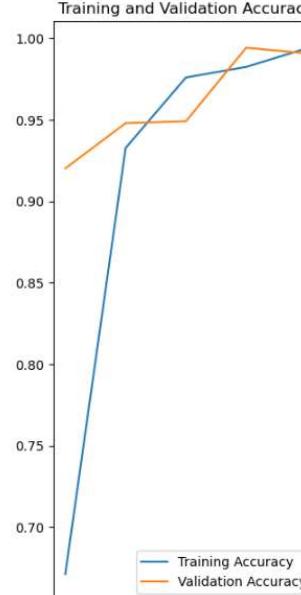
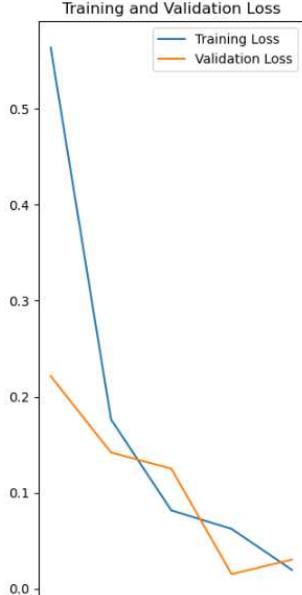
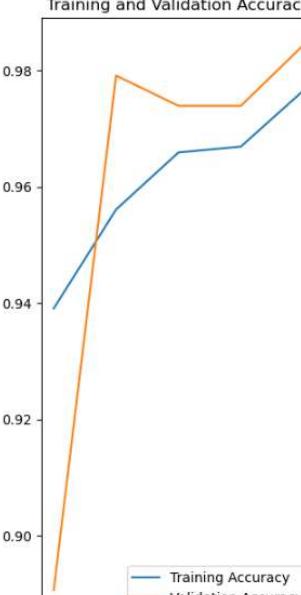
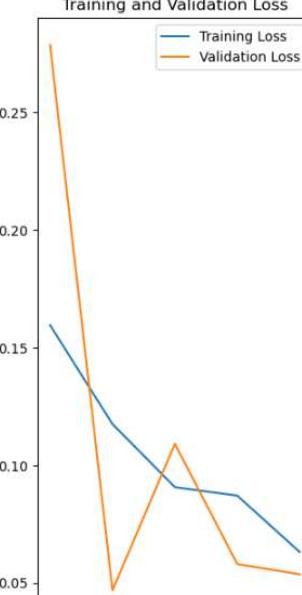
TensorFlow a mis de côté les informations de précision et perte lors de la phase d'entraînement et pour chaque epochs. Il nous suffit de les récupérer.

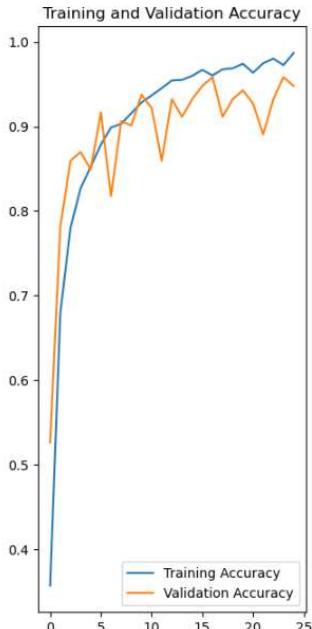
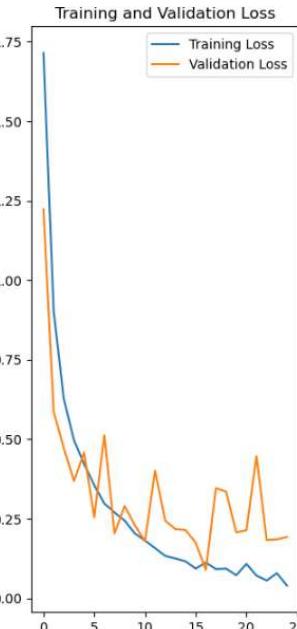
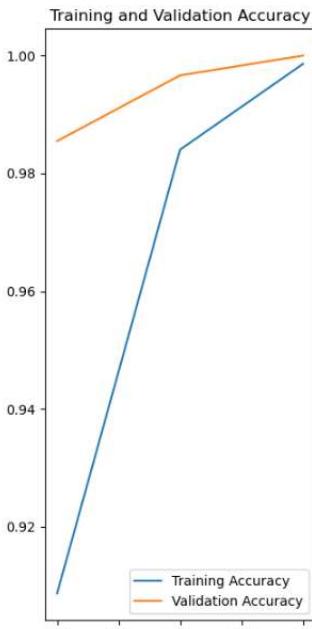
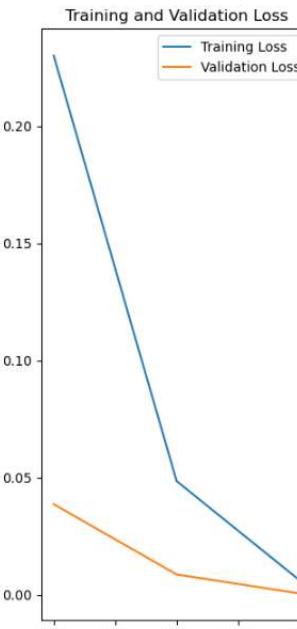
Tableau résumé des scores de modèles, courbes de précision et de perte de gradient des principales plantes du dataset\*:

| Classe de plante | Score (loss, accuracy) du modèle          | courbe de précision  | courbe de perte de gradient   |
|------------------|---|--|---|
| Apple            | [0.12362731248140335, 0.9622641801834106] |  |  |

| Cherry | [0.00162700901273638, 1.0]                |  <p>Training and Validation Accuracy</p> <table border="1"> <thead> <tr> <th>Epoch</th> <th>Training Accuracy</th> <th>Validation Accuracy</th> </tr> </thead> <tbody> <tr><td>1</td><td>0.96</td><td>0.98</td></tr> <tr><td>2</td><td>0.99</td><td>1.00</td></tr> <tr><td>3</td><td>0.995</td><td>1.00</td></tr> <tr><td>4</td><td>1.00</td><td>1.00</td></tr> </tbody> </table>   | Epoch | Training Accuracy | Validation Accuracy | 1 | 0.96 | 0.98 | 2 | 0.99 | 1.00 | 3 | 0.995 | 1.00 | 4 | 1.00 | 1.00 |  <p>Training and Validation Loss</p> <table border="1"> <thead> <tr> <th>Epoch</th> <th>Training Loss</th> <th>Validation Loss</th> </tr> </thead> <tbody> <tr><td>1</td><td>2.5</td><td>0.0</td></tr> <tr><td>2</td><td>0.0</td><td>0.0</td></tr> <tr><td>3</td><td>0.0</td><td>0.0</td></tr> <tr><td>4</td><td>0.0</td><td>0.0</td></tr> </tbody> </table> | Epoch | Training Loss | Validation Loss   | 1     | 2.5           | 0.0             | 2 | 0.0 | 0.0  | 3 | 0.0  | 0.0  | 4 | 0.0  | 0.0  |   |      |      |   |      |      |
|--------|---|---|-------|-------------------|---------------------|---|------|------|---|------|------|---|-------|------|---|------|------|---|-------|---------------|---|-------|---------------|-----------------|---|-----|------|---|------|------|---|------|------|---|------|------|---|------|------|
| Epoch  | Training Accuracy                         | Validation Accuracy   |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 1      | 0.96                                      | 0.98  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 2      | 0.99                                      | 1.00  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 3      | 0.995                                     | 1.00  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 4      | 1.00                                      | 1.00  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| Epoch  | Training Loss                             | Validation Loss   |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 1      | 2.5                                       | 0.0   |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 2      | 0.0                                       | 0.0   |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 3      | 0.0                                       | 0.0   |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 4      | 0.0                                       | 0.0   |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| Corn   | [0.17177796363830566, 0.9341736435890198] |  <p>Training and Validation Accuracy</p> <table border="1"> <thead> <tr> <th>Epoch</th> <th>Training Accuracy</th> <th>Validation Accuracy</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.60</td><td>0.81</td></tr> <tr><td>1</td><td>0.87</td><td>0.85</td></tr> <tr><td>2</td><td>0.90</td><td>0.92</td></tr> <tr><td>3</td><td>0.92</td><td>0.91</td></tr> <tr><td>4</td><td>0.93</td><td>0.94</td></tr> </tbody> </table> | Epoch | Training Accuracy | Validation Accuracy | 0 | 0.60 | 0.81 | 1 | 0.87 | 0.85 | 2 | 0.90  | 0.92 | 3 | 0.92 | 0.91 | 4   | 0.93  | 0.94          |  <p>Training and Validation Loss</p> <table border="1"> <thead> <tr> <th>Epoch</th> <th>Training Loss</th> <th>Validation Loss</th> </tr> </thead> <tbody> <tr><td>0</td><td>1.0</td><td>0.35</td></tr> <tr><td>1</td><td>0.65</td><td>0.30</td></tr> <tr><td>2</td><td>0.25</td><td>0.15</td></tr> <tr><td>3</td><td>0.18</td><td>0.20</td></tr> <tr><td>4</td><td>0.18</td><td>0.18</td></tr> </tbody> </table> | Epoch | Training Loss | Validation Loss | 0 | 1.0 | 0.35 | 1 | 0.65 | 0.30 | 2 | 0.25 | 0.15 | 3 | 0.18 | 0.20 | 4 | 0.18 | 0.18 |
| Epoch  | Training Accuracy                         | Validation Accuracy   |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 0      | 0.60                                      | 0.81  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 1      | 0.87                                      | 0.85  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 2      | 0.90                                      | 0.92  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 3      | 0.92                                      | 0.91  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 4      | 0.93                                      | 0.94  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| Epoch  | Training Loss                             | Validation Loss   |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 0      | 1.0                                       | 0.35  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 1      | 0.65                                      | 0.30  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 2      | 0.25                                      | 0.15  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 3      | 0.18                                      | 0.20  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 4      | 0.18                                      | 0.18  |       |                   |                     |   |      |      |   |      |      |   |       |      |   |      |      |   |       |               |   |       |               |                 |   |     |      |   |      |      |   |      |      |   |      |      |   |      |      |

| Grape | [0.09914905577898026, 0.9603794813156128] |  <p>Training and Validation Accuracy</p> <table border="1"> <thead> <tr> <th>Epoch</th> <th>Training Accuracy</th> <th>Validation Accuracy</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.48</td><td>0.70</td></tr> <tr><td>1</td><td>0.85</td><td>0.85</td></tr> <tr><td>2</td><td>0.85</td><td>0.85</td></tr> <tr><td>3</td><td>0.92</td><td>0.95</td></tr> <tr><td>4</td><td>0.94</td><td>0.96</td></tr> </tbody> </table>  | Epoch | Training Accuracy | Validation Accuracy | 0 | 0.48 | 0.70 | 1 | 0.85 | 0.85 | 2 | 0.85 | 0.85 | 3 | 0.92 | 0.95 | 4 | 0.94 | 0.96 |  <p>Training and Validation Loss</p> <table border="1"> <thead> <tr> <th>Epoch</th> <th>Training Loss</th> <th>Validation Loss</th> </tr> </thead> <tbody> <tr><td>0</td><td>1.15</td><td>0.90</td></tr> <tr><td>1</td><td>0.55</td><td>0.35</td></tr> <tr><td>2</td><td>0.40</td><td>0.40</td></tr> <tr><td>3</td><td>0.25</td><td>0.15</td></tr> <tr><td>4</td><td>0.20</td><td>0.10</td></tr> </tbody> </table>  | Epoch | Training Loss | Validation Loss | 0 | 1.15 | 0.90 | 1 | 0.55 | 0.35 | 2 | 0.40 | 0.40 | 3 | 0.25 | 0.15 | 4 | 0.20 | 0.10 |
|-------|---|---|-------|-------------------|---------------------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|--|-------|---------------|-----------------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|
| Epoch | Training Accuracy                         | Validation Accuracy   |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 0     | 0.48                                      | 0.70  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 1     | 0.85                                      | 0.85  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 2     | 0.85                                      | 0.85  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 3     | 0.92                                      | 0.95  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 4     | 0.94                                      | 0.96  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| Epoch | Training Loss                             | Validation Loss   |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 0     | 1.15                                      | 0.90  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 1     | 0.55                                      | 0.35  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 2     | 0.40                                      | 0.40  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 3     | 0.25                                      | 0.15  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 4     | 0.20                                      | 0.10  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| Peach | [0.09079906344413757, 0.9635535478591919] |  <p>Training and Validation Accuracy</p> <table border="1"> <thead> <tr> <th>Epoch</th> <th>Training Accuracy</th> <th>Validation Accuracy</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.80</td><td>0.93</td></tr> <tr><td>1</td><td>0.95</td><td>0.91</td></tr> <tr><td>2</td><td>0.95</td><td>0.96</td></tr> <tr><td>3</td><td>0.96</td><td>0.97</td></tr> <tr><td>4</td><td>0.96</td><td>0.98</td></tr> </tbody> </table> | Epoch | Training Accuracy | Validation Accuracy | 0 | 0.80 | 0.93 | 1 | 0.95 | 0.91 | 2 | 0.95 | 0.96 | 3 | 0.96 | 0.97 | 4 | 0.96 | 0.98 |  <p>Training and Validation Loss</p> <table border="1"> <thead> <tr> <th>Epoch</th> <th>Training Loss</th> <th>Validation Loss</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.40</td><td>0.20</td></tr> <tr><td>1</td><td>0.15</td><td>0.22</td></tr> <tr><td>2</td><td>0.12</td><td>0.10</td></tr> <tr><td>3</td><td>0.10</td><td>0.08</td></tr> <tr><td>4</td><td>0.10</td><td>0.05</td></tr> </tbody> </table> | Epoch | Training Loss | Validation Loss | 0 | 0.40 | 0.20 | 1 | 0.15 | 0.22 | 2 | 0.12 | 0.10 | 3 | 0.10 | 0.08 | 4 | 0.10 | 0.05 |
| Epoch | Training Accuracy                         | Validation Accuracy   |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 0     | 0.80                                      | 0.93  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 1     | 0.95                                      | 0.91  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 2     | 0.95                                      | 0.96  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 3     | 0.96                                      | 0.97  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 4     | 0.96                                      | 0.98  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| Epoch | Training Loss                             | Validation Loss   |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 0     | 0.40                                      | 0.20  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 1     | 0.15                                      | 0.22  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 2     | 0.12                                      | 0.10  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 3     | 0.10                                      | 0.08  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 4     | 0.10                                      | 0.05  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |

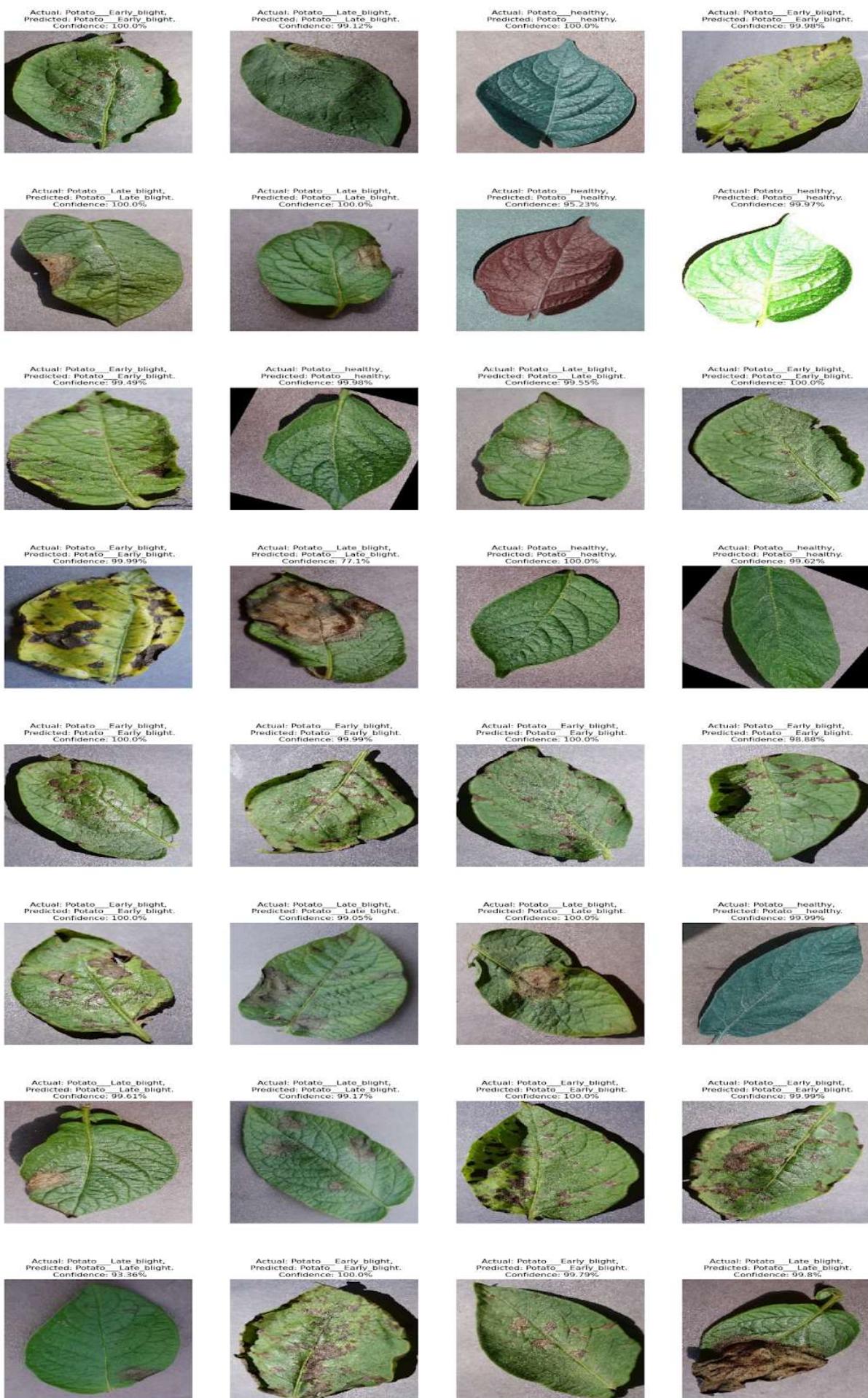
| Pepper | $[0.021686404943466187, 0.9948717951774597]$ |  <p>Training and Validation Accuracy</p> <table border="1"> <thead> <tr> <th>Epoch</th> <th>Training Accuracy</th> <th>Validation Accuracy</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.65</td><td>0.92</td></tr> <tr><td>1</td><td>0.94</td><td>0.95</td></tr> <tr><td>2</td><td>0.98</td><td>0.95</td></tr> <tr><td>3</td><td>0.99</td><td>0.99</td></tr> <tr><td>4</td><td>0.99</td><td>0.99</td></tr> </tbody> </table>  | Epoch | Training Accuracy | Validation Accuracy | 0 | 0.65 | 0.92 | 1 | 0.94 | 0.95 | 2 | 0.98 | 0.95 | 3 | 0.99 | 0.99 | 4 | 0.99 | 0.99 |  <p>Training and Validation Loss</p> <table border="1"> <thead> <tr> <th>Epoch</th> <th>Training Loss</th> <th>Validation Loss</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.60</td><td>0.22</td></tr> <tr><td>1</td><td>0.18</td><td>0.05</td></tr> <tr><td>2</td><td>0.08</td><td>0.08</td></tr> <tr><td>3</td><td>0.05</td><td>0.02</td></tr> <tr><td>4</td><td>0.03</td><td>0.01</td></tr> </tbody> </table>  | Epoch | Training Loss | Validation Loss | 0 | 0.60 | 0.22 | 1 | 0.18 | 0.05 | 2 | 0.08 | 0.08 | 3 | 0.05 | 0.02 | 4 | 0.03 | 0.01 |
|--------|--|---|-------|-------------------|---------------------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|--|-------|---------------|-----------------|---|------|------|---|------|------|---|------|------|---|------|------|---|------|------|
| Epoch  | Training Accuracy                            | Validation Accuracy   |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 0      | 0.65   | 0.92  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 1      | 0.94   | 0.95  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 2      | 0.98   | 0.95  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 3      | 0.99   | 0.99  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 4      | 0.99   | 0.99  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| Epoch  | Training Loss                                | Validation Loss   |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 0      | 0.60   | 0.22  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 1      | 0.18   | 0.05  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 2      | 0.08   | 0.08  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 3      | 0.05   | 0.02  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 4      | 0.03   | 0.01  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| Potato | $[0.030832966789603233, 0.9945651888847351]$ |  <p>Training and Validation Accuracy</p> <table border="1"> <thead> <tr> <th>Epoch</th> <th>Training Accuracy</th> <th>Validation Accuracy</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.94</td><td>0.88</td></tr> <tr><td>1</td><td>0.96</td><td>0.98</td></tr> <tr><td>2</td><td>0.97</td><td>0.97</td></tr> <tr><td>3</td><td>0.97</td><td>0.97</td></tr> <tr><td>4</td><td>0.98</td><td>0.99</td></tr> </tbody> </table> | Epoch | Training Accuracy | Validation Accuracy | 0 | 0.94 | 0.88 | 1 | 0.96 | 0.98 | 2 | 0.97 | 0.97 | 3 | 0.97 | 0.97 | 4 | 0.98 | 0.99 |  <p>Training and Validation Loss</p> <table border="1"> <thead> <tr> <th>Epoch</th> <th>Training Loss</th> <th>Validation Loss</th> </tr> </thead> <tbody> <tr><td>0</td><td>0.15</td><td>0.28</td></tr> <tr><td>1</td><td>0.10</td><td>0.05</td></tr> <tr><td>2</td><td>0.09</td><td>0.10</td></tr> <tr><td>3</td><td>0.08</td><td>0.05</td></tr> <tr><td>4</td><td>0.07</td><td>0.05</td></tr> </tbody> </table> | Epoch | Training Loss | Validation Loss | 0 | 0.15 | 0.28 | 1 | 0.10 | 0.05 | 2 | 0.09 | 0.10 | 3 | 0.08 | 0.05 | 4 | 0.07 | 0.05 |
| Epoch  | Training Accuracy                            | Validation Accuracy   |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 0      | 0.94   | 0.88  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 1      | 0.96   | 0.98  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 2      | 0.97   | 0.97  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 3      | 0.97   | 0.97  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 4      | 0.98   | 0.99  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| Epoch  | Training Loss                                | Validation Loss   |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 0      | 0.15   | 0.28  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 1      | 0.10   | 0.05  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 2      | 0.09   | 0.10  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 3      | 0.08   | 0.05  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |
| 4      | 0.07   | 0.05  |       |                   |                     |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |  |       |               |                 |   |      |      |   |      |      |   |      |      |   |      |      |   |      |      |

|  |  |  |
|--|--|--|
| Tomato   | <p>[0.33856672048568726,<br/>0.9072453379631042]</p>           |     |
| Strawberry   | <p>[0.0017257792642340064<br/>,</p> <p>0.9995553493499756]</p> |   |
| * types de plantes ayant un nombre de classe > ou= 2 |  |  |

### Interprétation des résultats

Le score de précision de chaque modèle par type de plante est supérieur à 0.93 excepté pour les tomates dont le score de précision est de 0.90. Il serait nécessaire d'améliorer ce score en utilisant du transfer learning ou en modifiant les couches de convolution du modèle actuel afin d'obtenir une modélisation finale plus performante.

### Affichage de quelques prédictions (modèle potato)



### c. Concaténation des modèles et prédiction finale

L'étape finale dans notre modèle à 2 étapes est de rassembler les 2 modèles pour prédire la plante et son état. Pour cela, nous avons importé, dans un nouveau notebook, les modèles préalablement sauvegardés en .h5.

Ensuite, notre source de données contenant un dossier "test", nous nous sommes servi de ce dossier pour appliquer notre modèle à 2 étapes.

Le dossier test est constitué de 33 fichiers image en .jpg; leur nom va nous permettre de vérifier la précision de notre modèle.

La première étape de cette concaténation de modèles est la création d'un dataframe contenant en première colonne le chemin d'accès des images, en deuxième colonne le nom de l'image. A cela nous avons ajouté 2 colonnes vides ['prediction\_plante', 'prediction\_etaf'] qui seront complétées lors de la prédiction.

|   | filepath                   | test_image_names    | prediction_plante | prediction_etaf |
|---|----------------------------|---------------------|-------------------|-----------------|
| 0 | ./test/AppleCedarRust1.JPG | AppleCedarRust1.JPG |                   |                 |
| 1 | ./test/AppleCedarRust2.JPG | AppleCedarRust2.JPG |                   |                 |
| 2 | ./test/AppleCedarRust3.JPG | AppleCedarRust3.JPG |                   |                 |
| 3 | ./test/AppleCedarRust4.JPG | AppleCedarRust4.JPG |                   |                 |
| 4 | ./test/AppleScab1.JPG      | AppleScab1.JPG      |                   |                 |

aperçu de df\_test

En première étape, nous appliquons la méthode predict sur l'array constitué par df\_test.filepath en appelant le modèle « model\_entier\_vgg16\_inetV3.h5 » que nous avons vu précédemment.

Le calcul de la meilleure probabilité (méthode argmax) va nous donner le rang de celle-ci dans chaque array (y\_prob) et nous permettra donc de déterminer la classe de chaque plante du dataset test par l'application d'une boucle sur la colonne df\_test['prediction\_plante'].

|    | filepath                          | test_image_names           | prediction_plante | prediction_etat |
|----|-----------------------------------|----------------------------|-------------------|-----------------|
| 0  | ./test/AppleCedarRust1.JPG        | AppleCedarRust1.JPG        | 1                 | 1               |
| 1  | ./test/AppleCedarRust2.JPG        | AppleCedarRust2.JPG        | 0                 | 0               |
| 2  | ./test/AppleCedarRust3.JPG        | AppleCedarRust3.JPG        | 0                 | 0               |
| 3  | ./test/AppleCedarRust4.JPG        | AppleCedarRust4.JPG        | 0                 | 0               |
| 4  | ./test/AppleScab1.JPG             | AppleScab1.JPG             | 0                 | 0               |
| 5  | ./test/AppleScab2.JPG             | AppleScab2.JPG             | 0                 | 0               |
| 6  | ./test/AppleScab3.JPG             | AppleScab3.JPG             | 12                | 12              |
| 7  | ./test/CornCommonRust1.JPG        | CornCommonRust1.JPG        | 3                 | 3               |
| 8  | ./test/CornCommonRust2.JPG        | CornCommonRust2.JPG        | 3                 | 3               |
| 9  | ./test/CornCommonRust3.JPG        | CornCommonRust3.JPG        | 3                 | 3               |
| 10 | ./test/PotatoEarlyBlight1.JPG     | PotatoEarlyBlight1.JPG     | 8                 | 8               |
| 11 | ./test/PotatoEarlyBlight2.JPG     | PotatoEarlyBlight2.JPG     | 8                 | 8               |
| 12 | ./test/PotatoEarlyBlight3.JPG     | PotatoEarlyBlight3.JPG     | 8                 | 8               |
| 13 | ./test/PotatoEarlyBlight4.JPG     | PotatoEarlyBlight4.JPG     | 8                 | 8               |
| 14 | ./test/PotatoEarlyBlight5.JPG     | PotatoEarlyBlight5.JPG     | 8                 | 8               |
| 15 | ./test/PotatoHealthy1.JPG         | PotatoHealthy1.JPG         | 8                 | 8               |
| 16 | ./test/PotatoHealthy2.JPG         | PotatoHealthy2.JPG         | 8                 | 8               |
| 17 | ./test/TomatoEarlyBlight1.JPG     | TomatoEarlyBlight1.JPG     | 13                | 13              |
| 18 | ./test/TomatoEarlyBlight2.JPG     | TomatoEarlyBlight2.JPG     | 13                | 13              |
| 19 | ./test/TomatoEarlyBlight3.JPG     | TomatoEarlyBlight3.JPG     | 13                | 13              |
| 20 | ./test/TomatoEarlyBlight4.JPG     | TomatoEarlyBlight4.JPG     | 13                | 13              |
| 21 | ./test/TomatoEarlyBlight5.JPG     | TomatoEarlyBlight5.JPG     | 13                | 13              |
| 22 | ./test/TomatoEarlyBlight6.JPG     | TomatoEarlyBlight6.JPG     | 13                | 13              |
| 23 | ./test/TomatoHealthy1.JPG         | TomatoHealthy1.JPG         | 13                | 13              |
| 24 | ./test/TomatoHealthy2.JPG         | TomatoHealthy2.JPG         | 13                | 13              |
| 25 | ./test/TomatoHealthy3.JPG         | TomatoHealthy3.JPG         | 13                | 13              |
| 26 | ./test/TomatoHealthy4.JPG         | TomatoHealthy4.JPG         | 13                | 13              |
| 27 | ./test/TomatoYellowCurlVirus1.JPG | TomatoYellowCurlVirus1.JPG | 13                | 13              |
| 28 | ./test/TomatoYellowCurlVirus2.JPG | TomatoYellowCurlVirus2.JPG | 13                | 13              |
| 29 | ./test/TomatoYellowCurlVirus3.JPG | TomatoYellowCurlVirus3.JPG | 13                | 13              |
| 30 | ./test/TomatoYellowCurlVirus4.JPG | TomatoYellowCurlVirus4.JPG | 13                | 13              |
| 31 | ./test/TomatoYellowCurlVirus5.JPG | TomatoYellowCurlVirus5.JPG | 13                | 13              |
| 32 | ./test/TomatoYellowCurlVirus6.JPG | TomatoYellowCurlVirus6.JPG | 13                | 13              |

| nameLabel /label |      |
|------------------|------|
| apple            | [0]  |
| Blueberry        | [1]  |
| cherry           | [2]  |
| corn             | [3]  |
| grape            | [4]  |
| Orange           | [5]  |
| peach            | [6]  |
| pepper           | [7]  |
| potato           | [8]  |
| Raspberry        | [9]  |
| Soybean          | [10] |
| Squash           | [11] |
| strawberry       | [12] |
| tomato           | [13] |

df\_test avec prediction\_plante

Interprétation de la première étape :

Le modèle semble très bien fonctionner et prédit 31 classes de plantes sur 33 (soit 94% de précision). Les erreurs observées se situent toutes les 2 pour la catégorie des pommes, le fichier ligne 0 est attribué à la catégorie Blueberry, le fichier ligne 6 est quant à lui attribué à la catégorie strawberry.

La deuxième étape du modèle consiste à utiliser la prédiction de la plante pour déclencher le choix du modèle de plante à appeler et ainsi prédire l'état de la plante.

De la même façon, la recherche de la probabilité la plus forte va déterminer le rang de cet élément et ainsi déterminer la classe correspondant à l'état de la plante.

Cette information est reportée dans la dernière colonne du dataframe appelée ['prediction\_etat'].

|    | filepath                         | test_image_names           | prediction_plante | prediction_etaat | Label 'etat'   | Label  |
|----|----------------------------------|----------------------------|-------------------|------------------|--|--|
| 0  | /test/AppleCedarRust1.JPG        | AppleCedarRust1.JPG        | 1                 | 2                | 'Apple__Apple_scab',<br>'Apple__Black_rot',<br>'Apple__Cedar_apple_rust',<br>'Apple__healthy'  | [0]<br>[1]<br>[2]<br>[3]   |
| 1  | /test/AppleCedarRust2.JPG        | AppleCedarRust2.JPG        | 0                 | 2                |  |  |
| 2  | /test/AppleCedarRust3.JPG        | AppleCedarRust3.JPG        | 0                 | 2                |  |  |
| 3  | /test/AppleCedarRust4.JPG        | AppleCedarRust4.JPG        | 0                 | 2                |  |  |
| 4  | /test/AppleScab1.JPG             | AppleScab1.JPG             | 0                 | 0                | 'Corn_(maize)_Cercospora_leaf_spot;',<br>'Corn_(maize)_Common_rust',<br>'Corn_(maize)_Northern_Leaf_Blight',<br>'Corn_(maize)_healthy'   | [0]<br>[1]<br>[2]<br>[3]   |
| 5  | /test/AppleScab2.JPG             | AppleScab2.JPG             | 0                 | 0                |  |  |
| 6  | /test/AppleScab3.JPG             | AppleScab3.JPG             | 12                | 3                |  |  |
| 7  | /test/CornCommonRust1.JPG        | CornCommonRust1.JPG        | 3                 | 0                | 'Potato_Early_blight',<br>'Potato_Late_blight',<br>'Potato_healthy'  | [0]<br>[1]<br>[2]  |
| 8  | /test/CornCommonRust2.JPG        | CornCommonRust2.JPG        | 3                 | 1                |  |  |
| 9  | /test/CornCommonRust3.JPG        | CornCommonRust3.JPG        | 3                 | 2                |  |  |
| 10 | /test/PotatoEarlyBlight1.JPG     | PotatoEarlyBlight1.JPG     | 8                 | 1                | 'Grape__Black_rot',<br>'Grape__Esca_(Black_Measles)',<br>'Grape__Leaf_blight_(Isariopsis_Leaf_Spot)',<br>'Grape__healthy'  | [0]<br>[1]<br>[2]<br>[3]   |
| 11 | /test/PotatoEarlyBlight2.JPG     | PotatoEarlyBlight2.JPG     | 8                 | 0                |  |  |
| 12 | /test/PotatoEarlyBlight3.JPG     | PotatoEarlyBlight3.JPG     | 8                 | 3                |  |  |
| 13 | /test/PotatoEarlyBlight4.JPG     | PotatoEarlyBlight4.JPG     | 8                 | 2                |  |  |
| 14 | /test/PotatoEarlyBlight5.JPG     | PotatoEarlyBlight5.JPG     | 8                 | 3                |  |  |
| 15 | /test/PotatoHealthy1.JPG         | PotatoHealthy1.JPG         | 8                 | 0                | 'Tomato_Bacterial_spot',<br>'Tomato_Early_blight',<br>'Tomato_Late_blight',<br>'Tomato_Leaf_Mold',<br>'Tomato_Septoria_leaf_spot',<br>'Tomato_Spider_mites_Two-spotted:',<br>'Tomato_Target_Spot',<br>'Tomato_Tomato_Yellow_Leaf_Curl_Virus',<br>'Tomato_Tomato_mosaic_virus',<br>'Tomato_healthy' | [0]<br>[1]<br>[2]<br>[3]<br>[4]<br>[5]<br>[6]<br>[7]<br>[8]<br>[9] |
| 16 | /test/PotatoHealthy2.JPG         | PotatoHealthy2.JPG         | 8                 | 0                |  |  |
| 17 | /test/TomatoEarlyBlight1.JPG     | TomatoEarlyBlight1.JPG     | 13                | 2                |  |  |
| 18 | /test/TomatoEarlyBlight2.JPG     | TomatoEarlyBlight2.JPG     | 13                | 0                |  |  |
| 19 | /test/TomatoEarlyBlight3.JPG     | TomatoEarlyBlight3.JPG     | 13                | 3                |  |  |
| 20 | /test/TomatoEarlyBlight4.JPG     | TomatoEarlyBlight4.JPG     | 13                | 3                |  |  |
| 21 | /test/TomatoEarlyBlight5.JPG     | TomatoEarlyBlight5.JPG     | 13                | 3                |  |  |
| 22 | /test/TomatoEarlyBlight6.JPG     | TomatoEarlyBlight6.JPG     | 13                | 3                |  |  |
| 23 | /test/TomatoHealthy1.JPG         | TomatoHealthy1.JPG         | 13                | 3                |  |  |
| 24 | /test/TomatoHealthy2.JPG         | TomatoHealthy2.JPG         | 13                | 0                |  |  |
| 25 | /test/TomatoHealthy3.JPG         | TomatoHealthy3.JPG         | 13                | 3                |  |  |
| 26 | /test/TomatoHealthy4.JPG         | TomatoHealthy4.JPG         | 13                | 3                |  |  |
| 27 | /test/TomatoYellowCurlVirus1.JPG | TomatoYellowCurlVirus1.JPG | 13                | 3                |  |  |
| 28 | /test/TomatoYellowCurlVirus2.JPG | TomatoYellowCurlVirus2.JPG | 13                | 3                |  |  |
| 29 | /test/TomatoYellowCurlVirus3.JPG | TomatoYellowCurlVirus3.JPG | 13                | 3                |  |  |
| 30 | /test/TomatoYellowCurlVirus4.JPG | TomatoYellowCurlVirus4.JPG | 13                | 0                |  |  |
| 31 | /test/TomatoYellowCurlVirus5.JPG | TomatoYellowCurlVirus5.JPG | 13                | 2                |  |  |
| 32 | /test/TomatoYellowCurlVirus6.JPG | TomatoYellowCurlVirus6.JPG | 13                | 0                |  |  |

df\_test['prediction\_plante','prediction\_etaat']

Tableau des labels des états

### Interprétation de la deuxième étape du modèle:

Il semble que cette deuxième prédiction concernant l'état de la plante n'est pas aussi performante que la première étape. En effet, en comparant avec le tableau des labels des état de plante (à droite), la capacité du modèle à prédire correctement la classe réelle de la photo n'est que de 7 sur 33 soit une précision de 0.21.

### Conclusion sur le modèle à 2 étapes :

Comme nous avons pu le constater, le modèle fonctionne très bien pour déterminer la catégorie de plante mais ne sait pas catégoriser son état.

Il serait intéressant de parcourir à nouveau le jeu de données et voir si une augmentation du nombre de données à entraîner serait bénéfique pour la seconde étape du modèle. Il faudrait nécessairement faire un compromis entre performance et temps d'apprentissage.

Nous pouvons également nous interroger sur l'application d'un masque (type segmentation ou changement d'exposition de lumière) afin d'isoler certaines parties de la feuille et rendre l'interprétation du modèle plus facile. Nous pouvons également nous interroger sur la construction du modèle CNN utilisé dans l'étape 2, en ajoutant des couches de convolution ou en utilisant un modèle de transfer learning type vgg16, inception ou dense121.

Enfin, pour cette problématique nous avons choisi 2 types de modélisations; celle-ci s'est effectuée en 2 étapes, nous allons aborder la deuxième option de ce projet qui est la modélisation en 1 seule étape en catégorisant l'ensemble des 38 classes dans un modèle unique.

## 2. modèle de deep learning: modélisation en 1 seule étape (classification des 38 classes) avec vgg16

Cette classification a été créée avec les 38 classes, celle-ci sur un modèle VGG16 pour visualiser des premiers résultats. Ce modèle préconfiguré repose sur le réseau VGG mais avec une normalisation par lots, ce qui signifie que chaque couche du réseau est normalisée. Le jeu de données ImageNet contient plus de dix millions d'URL d'images avec libellés. Un million d'images sont également associées à des cadres de délimitation spécifiant un emplacement plus précis pour les objets étiquetés.

| Model: "sequential_1"                               |                         |          |
|---|-------------------------|----------|
| Layer (type)  | Output Shape            | Param #  |
| vgg16 (Functional)                                  | (None, None, None, 512) | 14714688 |
| global_average_pooling2d_1 (GlobalAveragePooling2D) | (None, 512)             | 0        |
| dense_3 (Dense)                                     | (None, 1024)            | 525312   |
| dropout_2 (Dropout)                                 | (None, 1024)            | 0        |
| dense_4 (Dense)                                     | (None, 512)             | 524800   |
| dropout_3 (Dropout)                                 | (None, 512)             | 0        |
| dense_5 (Dense)                                     | (None, 1)               | 513      |

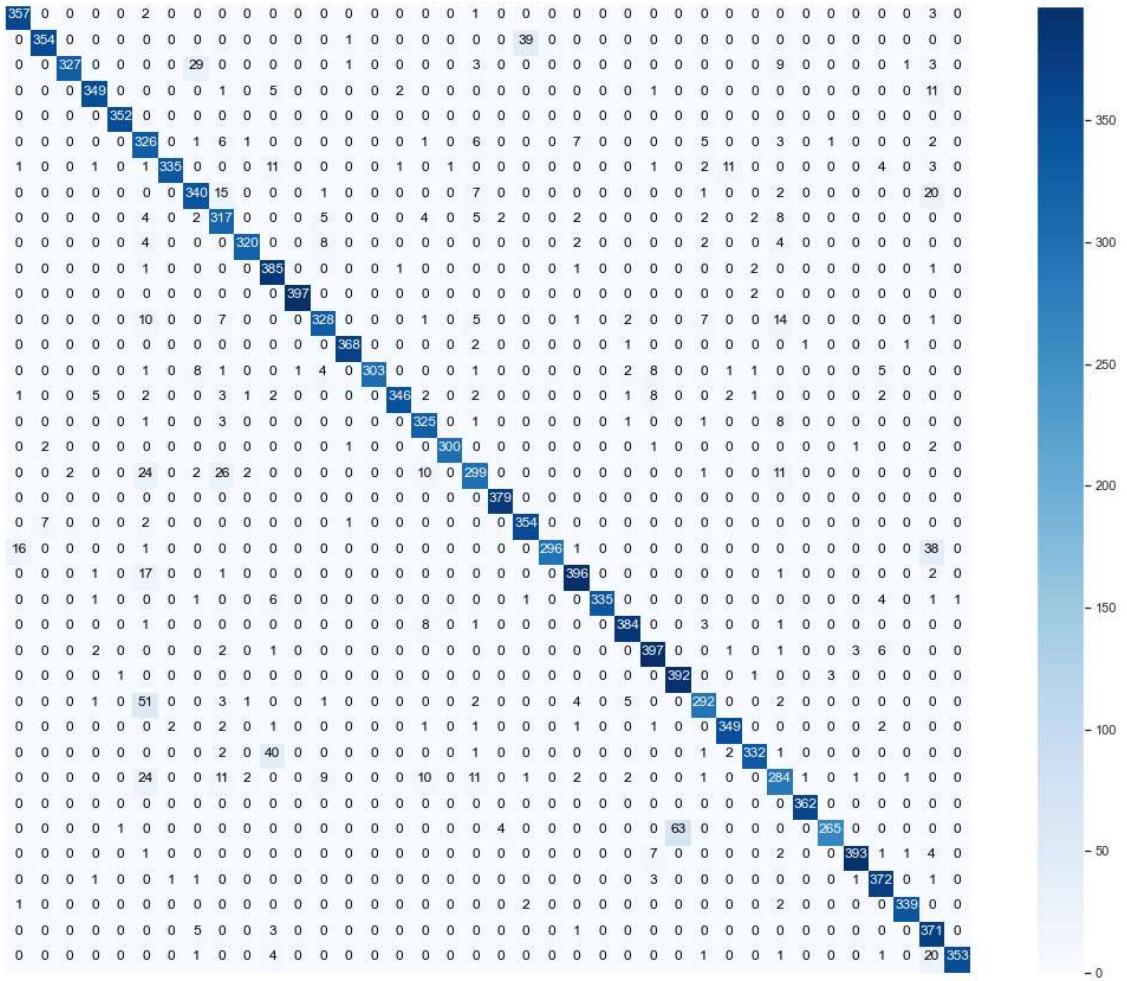
Total params: 15,765,313  
Trainable params: 1,050,625  
Non-trainable params: 14,714,688

Modèle VGG16

Après avoir utilisé le modèle préconfiguré VGG16, nous avons ajouté des couches Denses pour l'adapter à notre problématique. Après quelques epochs, nous avons observé que le modèle atteignait un plateau avec une précision d'environ 93%.

|              |      |      |      |       |
|--------------|------|------|------|-------|
| 35           | 0.99 | 0.99 | 0.99 | 344   |
| 36           | 0.77 | 0.98 | 0.86 | 380   |
| 37           | 1.00 | 0.93 | 0.96 | 381   |
| accuracy     |      |      | 0.93 | 14059 |
| macro avg    | 0.94 | 0.93 | 0.93 | 14059 |
| weighted avg | 0.94 | 0.93 | 0.93 | 14059 |

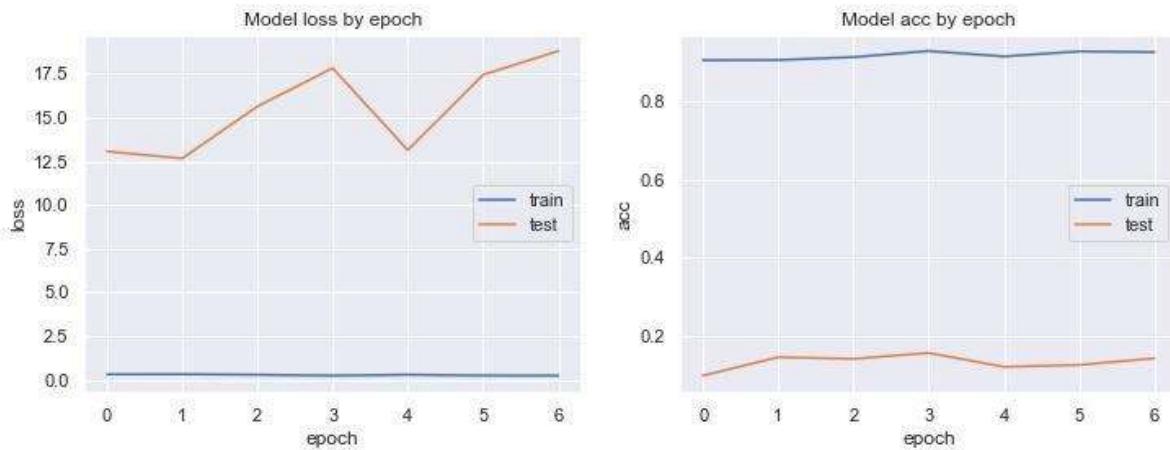
rapport de classification



## Matrice de confusion 38 classes

Nous avons testé différents hyper-paramètres pour optimiser notre modèle, comme l'optimizer 'Adam', qui a finalement été retenu car les autres options n'étaient pas convaincantes. Nous avons obtenu une matrice de confusion à la suite de cette prédiction.

l'appel de ce modèle pour prédire les résultats sur l'ensemble de fichier est nous donne les prédictions regroupées dans le tableau dessous:



| Unnamed: 0 | filepath  | nameLabel                  | label | Classe                                |
|------------|---|----------------------------|-------|---------------------------------------|
| 0          | archive/New Plant Diseases Dataset(Augmented)/... | AppleScab3.JPG             | 25    | Apple__Apple_scab                     |
| 1          | archive/New Plant Diseases Dataset(Augmented)/... | TomatoEarlyBlight2.JPG     | 8     | Tomato__Late_blight                   |
| 2          | archive/New Plant Diseases Dataset(Augmented)/... | TomatoEarlyBlight3.JPG     | 5     | Tomato__Target_Spot                   |
| 3          | archive/New Plant Diseases Dataset(Augmented)/... | PotatoHealthy1.JPG         | 36    | Potato__healthy                       |
| 4          | archive/New Plant Diseases Dataset(Augmented)/... | AppleScab2.JPG             | 25    | Apple__Apple_scab                     |
| 5          | archive/New Plant Diseases Dataset(Augmented)/... | TomatoEarlyBlight1.JPG     | 8     | Tomato__Late_blight                   |
| 6          | archive/New Plant Diseases Dataset(Augmented)/... | PotatoHealthy2.JPG         | 36    | Potato__healthy                       |
| 7          | archive/New Plant Diseases Dataset(Augmented)/... | AppleScab1.JPG             | 25    | Apple__Apple_scab                     |
| 8          | archive/New Plant Diseases Dataset(Augmented)/... | TomatoEarlyBlight4.JPG     | 18    | Tomato__Early_blight                  |
| 9          | archive/New Plant Diseases Dataset(Augmented)/... | TomatoEarlyBlight5.JPG     | 18    | Tomato__Early_blight                  |
| 10         | archive/New Plant Diseases Dataset(Augmented)/... | TomatoEarlyBlight6.JPG     | 18    | Tomato__Early_blight                  |
| 11         | archive/New Plant Diseases Dataset(Augmented)/... | PotatoEarlyBlight4.JPG     | 2     | Potato__Early_blight                  |
| 12         | archive/New Plant Diseases Dataset(Augmented)/... | PotatoEarlyBlight5.JPG     | 2     | Potato__Early_blight                  |
| 13         | archive/New Plant Diseases Dataset(Augmented)/... | PotatoEarlyBlight2.JPG     | 2     | Potato__Early_blight                  |
| 14         | archive/New Plant Diseases Dataset(Augmented)/... | PotatoEarlyBlight3.JPG     | 7     | Potato__Late_blight                   |
| 15         | archive/New Plant Diseases Dataset(Augmented)/... | PotatoEarlyBlight1.JPG     | 2     | Potato__Early_blight                  |
| 16         | archive/New Plant Diseases Dataset(Augmented)/... | TomatoYellowCurlVirus2.JPG | 24    | Tomato__Tomato_Yellow_Leaf_Curl_Virus |
| 17         | archive/New Plant Diseases Dataset(Augmented)/... | TomatoYellowCurlVirus3.JPG | 24    | Tomato__Tomato_Yellow_Leaf_Curl_Virus |
| 18         | archive/New Plant Diseases Dataset(Augmented)/... | TomatoYellowCurlVirus1.JPG | 24    | Tomato__Tomato_Yellow_Leaf_Curl_Virus |
| 19         | archive/New Plant Diseases Dataset(Augmented)/... | TomatoHealthy4.JPG         | 8     | Tomato__Late_blight                   |
| 20         | archive/New Plant Diseases Dataset(Augmented)/... | TomatoYellowCurlVirus4.JPG | 24    | Tomato__Tomato_Yellow_Leaf_Curl_Virus |
| 21         | archive/New Plant Diseases Dataset(Augmented)/... | TomatoHealthy1.JPG         | 22    | Tomato__healthy                       |
| 22         | archive/New Plant Diseases Dataset(Augmented)/... | TomatoYellowCurlVirus5.JPG | 24    | Tomato__Tomato_Yellow_Leaf_Curl_Virus |
| 23         | archive/New Plant Diseases Dataset(Augmented)/... | TomatoHealthy3.JPG         | 22    | Tomato__healthy                       |
| 24         | archive/New Plant Diseases Dataset(Augmented)/... | TomatoHealthy2.JPG         | 22    | Tomato__healthy                       |

Nous constatons 28 prédictions correctes sur 33 soit une précision de 0.84 pour la classification à 1 étape avec le modèle vgg16.

Ce résultat est bien supérieur à celui de la classification en 2 étapes.

### 3. Modèle de Deep Learning: modélisation en 1 seule étape (classification des 38 classes)

Dans ce modèle d'apprentissage supervisé, nous proposerons un modèle d'apprentissage profond qui classe simultanément l'espèce de la plante et son état.

Nous représentons dans ce paragraphe les pas accomplis dans la réalisation de la reconnaissance de plante.

Nous commençons par le processus de prétraitement des données d'image.

Ensuite, nous décrivons l'architecture du modèle que nous avons employée.

Nous concluons en visualisant les résultats et en analysant les avantages et les inconvénients du modèle.

#### a) Pré-traitement des données

Tout projet d'apprentissage automatique supervisé commence par la collecte de données que nous utiliserons comme ensemble de données d'entraînement.

Nous avons utilisé le package Tensorflow pour collecter les données et les étiquettes en fonction de son nom de dossiers.

Premièrement nous avons transformé les images en type TensorFlow contenant les données en array numpy et les étiquettes. Ensuite nous avons divisé les données en cinq parties pour les entraîner cinq fois en utilisant la méthode de Transfer Learning.

Chaque partie de données est sous divisée en 0.8 pour les données d'entraînement, 0.1 pour les données de validation et 0.1 pour les données de test.

Il faut noter aussi que nous avons appliqué la méthode rééchantillonnage.

Afin d'optimiser les performances du pipeline d'entraînement nous avons appliqué la technique de « prefetching & Caching » de package TensorFlow.

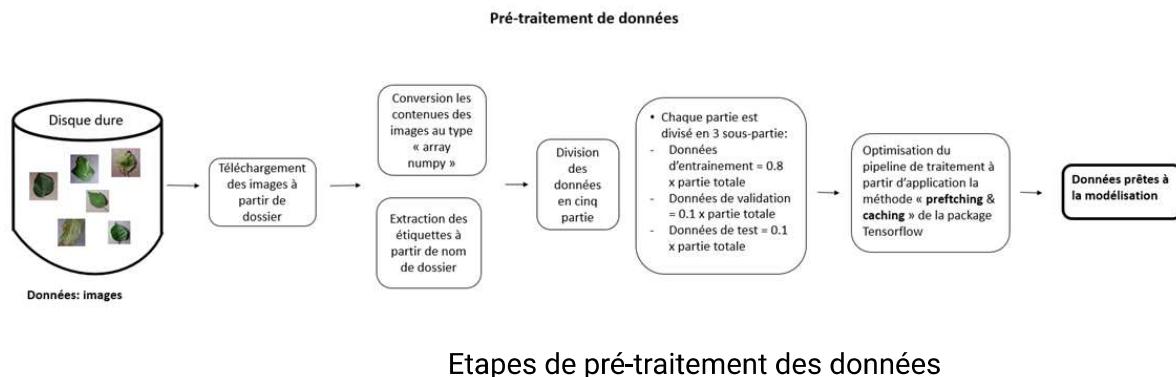
#### **prefetching:**

La technique de prefetching de TensorFlow charge et prépare le prochain lot de données à l'avance pendant que le lot actuel est traité par le modèle. Cela permet de réduire considérablement le temps d'attente du modèle pour le chargement du prochain lot de données, ce qui améliore les performances du pipeline.

#### **Caching :**

La technique Caching de TensorFlow stocke les données prétraitées en mémoire ou sur disque après qu'elles ont été chargées et traitées, de sorte que les mêmes données peuvent être utilisées dans les itérations suivantes sans avoir à les recharger et à les prétraiter à

nouveau. Cela permet de réduire le temps nécessaire au prétraitement des données et d'améliorer les performances globales du pipeline. (voir la figure ci-dessous qui explique les étapes de prétraitement des données)



**Note:** Nous n'avons appliqué d'augmentation de données puisque nous travaillons sur des données déjà augmenté.

### b) Architecture du modèle et l'algorithme d'optimisation

Nous avons utilisé la méthode Transférer l'apprentissage pour générer toutes les données dont nous disposons.

À cause de la capacité limitée de l'ordinateur dans le traitement, nous avons divisé la donnée brute en cinq parties. Ensuite, nous entraînons la première partie. Après cela, nous utilisons son poids de modélisation comme paramètre d'initialisation pour entraîner le modèle sur la deuxième partie des données et ainsi de suite jusqu'à ce que nous parvenions à entraîner toutes les données sur le même modèle.

Nous présentons alors l'architecture de modèle employée et l'algorithme d'optimisation.

### Architecture de modèle

les données utilisées correspondent au type d'image. Par conséquent l'architecture choisie est le « réseau de convolution neuronale » CNN car cette technique est la plus efficace dans la classification des données de type image.

La figure ci-dessous présente l'architecture de modèle choisi:

```

Model: "sequential_1"
=====
Layer (type)          Output Shape       Param #
=====
sequential (Sequential)    (32, 256, 256, 3)        0
=====
conv2d (Conv2D)         (32, 254, 254, 32)      896
max_pooling2d (MaxPooling2D) (32, 127, 127, 32)    0
conv2d_1 (Conv2D)        (32, 125, 125, 64)     18496
max_pooling2d_1 (MaxPooling2D) (32, 62, 62, 64)    0
conv2d_2 (Conv2D)        (32, 60, 60, 64)     36928
max_pooling2d_2 (MaxPooling2D) (32, 30, 30, 64)    0
conv2d_3 (Conv2D)        (32, 28, 28, 64)     36928
max_pooling2d_3 (MaxPooling2D) (32, 14, 14, 64)    0
conv2d_4 (Conv2D)        (32, 12, 12, 64)     36928
max_pooling2d_4 (MaxPooling2D) (32, 6, 6, 64)     0
conv2d_5 (Conv2D)        (32, 4, 4, 64)      36928
max_pooling2d_5 (MaxPooling2D) (32, 2, 2, 64)     0
flatten (Flatten)        (32, 256)           0
dense (Dense)            (32, 64)            16448
dense_1 (Dense)          (32, 38)            2470
=====
Total params: 186,022
Trainable params: 186,022
Non-trainable params: 0

```

Architecture de modèle de classification 38 classe.

La première couche nommée « séquentiel » sert à normaliser l'image et ajuster sa dimension.

L'objectif de normalisation est d'améliorer la convergence de modèle, de réduire le surajustement et de généraliser le modèle autrement dit de rendre le modèle plus performant sur de nouvelles données.

La deuxième couche du CNN est une couche convolutive. Elle applique un ensemble de filtres à l'image d'entrée, qui convolue l'image avec le filtre et produit une carte de caractéristiques. Chaque filtre détecte une caractéristique ou un motif spécifique dans l'image, comme les bords, les courbes ou les textures. Les filtres sont appris au cours de la formation et le nombre de filtres dans cette couche détermine la complexité des caractéristiques apprises.

La couche de max\_pooling sert à extraire les features de l'image et réduire la dimension spatiale de ces features.

Le processus de convolution, d'activation et de mise en commun est répété plusieurs fois, formant ainsi plusieurs couches convolutives. Chaque couche apprend des caractéristiques de plus en plus complexes qui sont des combinaisons des caractéristiques apprises dans les couches précédentes.

Après la dernière couche convolutive, la sortie est aplatie en un vecteur 1D et passe par une ou plusieurs couches entièrement connectées. Ces couches utilisent les caractéristiques apprises pour classer l'image d'entrée dans l'une des classes possibles. La dernière couche

est la couche softmax, qui convertit la sortie de la couche précédente en une distribution de probabilité sur les classes.

**Note :** nous avons utilisé la fonction d'activation ReLU dans les couches cachées. En effet, cette fonction est préférée dans les CNN pour les projets d'apprentissage en profondeur en raison de sa « sparsity », de sa non-linéarité, de sa convergence plus rapide, de sa capacité à éviter le problème du vanishing gradient et de ses performances empiriquement prouvées.

### Algorithme d'optimisation

En ce qui concerne l'algorithme d'optimisation dans le processus d'entraînement, nous avons choisi l'algorithme ADAM. Cet algorithme est souvent utilisé dans ce type de projet et cela est dû principalement à son taux d'apprentissage adaptatif et à sa robustesse.

Et pour finir, nous avons choisi la métrique Accuracy pour évaluer le modèle. En effet, nous sommes dans la problématique de classification et cet outil de mesure est courant et simple pour évaluer les performances des modèles de classifications.

#### c) Evaluation de modèle

Le tableau ci-dessous montre l'amélioration de la performance de modèle en fonction du nombre de données entraînées.

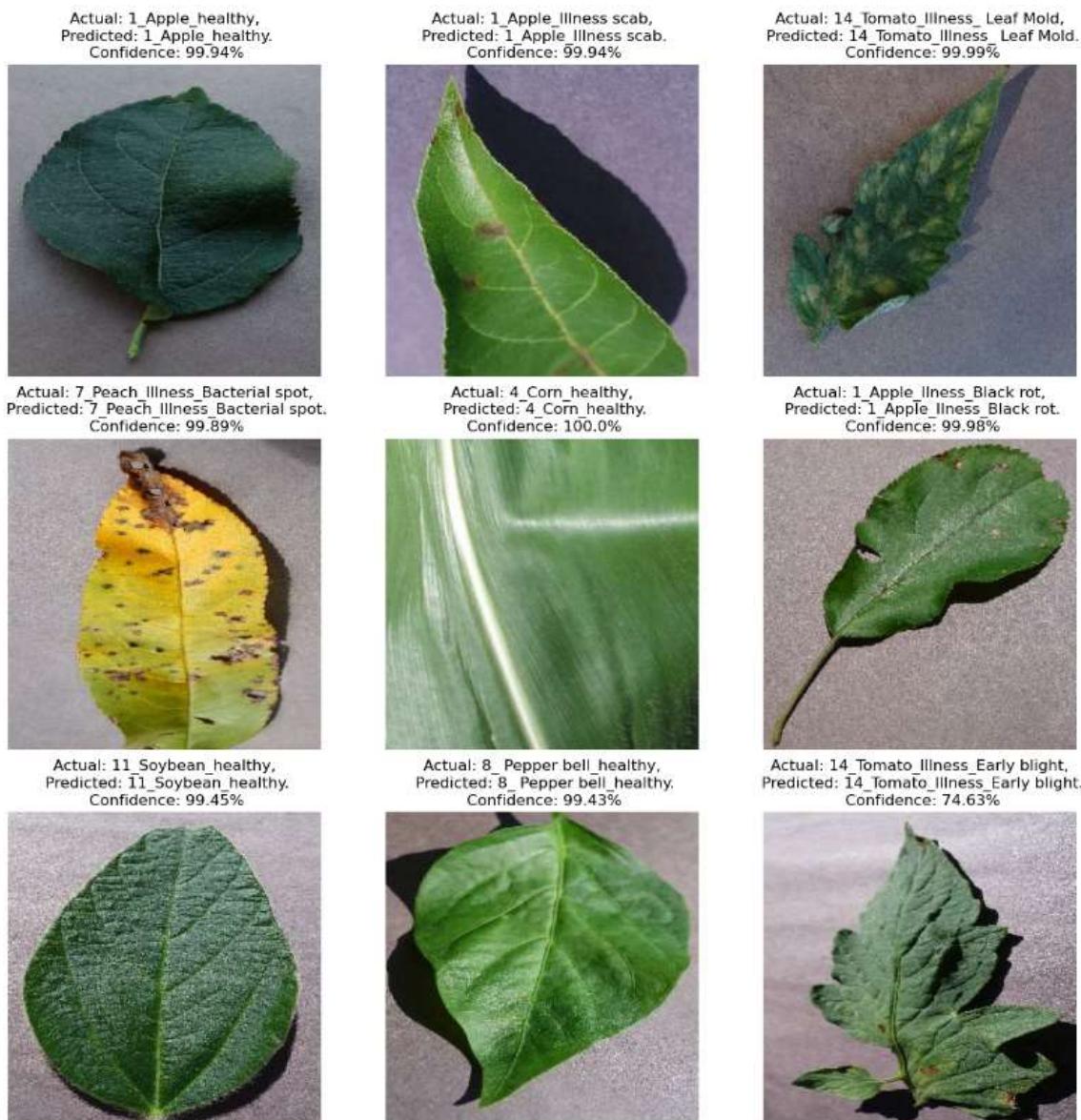
Tableau d'évolution de valeur d' "Accuracy"

|                          | Valeur de « Accuracy » |
|--------------------------|------------------------|
| Entrainement de partie 1 | 0.9129                 |
| Entrainement de partie 2 | 0.9628                 |
| Entrainement de partie 3 | 0.9646                 |
| Entrainement de partie 4 | 0.9771                 |
| Entrainement de partie 5 | 0.9802                 |

Après l'entraînement de toutes les données, nous arrivons à une valeur d'accuracy de 0.98.

Pour le score de modèle, nous avons testé et nous avons obtenu 0.92 de valeur de score.

La figure ci-dessous montre une comparaison entre la prédiction de modèle entraîné et la classe réel de la plante.



Visualisation de résultat de prédiction.

Il est clair que le modèle est bien performant au niveau de détection du type de plante, de son état et de son type de maladie. Cependant, ce modèle reste très spécifique à ce type de données autrement dit nous ne pouvons pas les généralisées pour les autres types de plantes. De plus, le temps de l'entraînement est long. En effet, pour 87000 images, l'entraînement est dure 8h pour un ordinateur de RAM 16GB et core i7.

### Application du modèle au dossier test

l'appel de ce modèle de prédiction sur 38 classes appliquée aux 33 fichiers images stockés dans le dossier test, nous donne les prédictions regroupées dans le tableau suivant:

|    | filepath                          | test_image_names           | prediction_plante | prediction_etat                                |
|----|-----------------------------------|----------------------------|-------------------|--|
| 0  | ./test/AppleCedarRust1.JPG        | AppleCedarRust1.JPG        | 15                |  |
| 1  | ./test/AppleCedarRust2.JPG        | AppleCedarRust2.JPG        | 15                | ['10_Raspberry_healthy', 0                     |
| 2  | ./test/AppleCedarRust3.JPG        | AppleCedarRust3.JPG        | 15                | '11_Soybean_healthy', 1                        |
| 3  | ./test/AppleCedarRust4.JPG        | AppleCedarRust4.JPG        | 15                | '12_Squash_Illness_Powdery Mildew', 2          |
| 4  | ./test/AppleScab1.JPG             | AppleScab1.JPG             | 30                | '13_Strawberry_Illness_Leaf scorch', 3         |
| 5  | ./test/AppleScab2.JPG             | AppleScab2.JPG             | 16                | '13_Strawberry_healthy', 4                     |
| 6  | ./test/AppleScab3.JPG             | AppleScab3.JPG             | 24                | '14_Tomato_Illness_ Leaf Mold', 5              |
| 7  | ./test/CornCommonRust1.JPG        | CornCommonRust1.JPG        | 22                | '14_Tomato_Illness_Bacterial spot', 6          |
| 8  | ./test/CornCommonRust2.JPG        | CornCommonRust2.JPG        | 22                | '14_Tomato_Illness_Early blight', 7            |
| 9  | ./test/CornCommonRust3.JPG        | CornCommonRust3.JPG        | 22                | '14_Tomato_Illness_Late blight', 8             |
| 10 | ./test/PotatoEarlyBlight1.JPG     | PotatoEarlyBlight1.JPG     | 35                | '14_Tomato_Illness_Septoria leaf spot', 9      |
| 11 | ./test/PotatoEarlyBlight2.JPG     | PotatoEarlyBlight2.JPG     | 35                | '14_Tomato_Illness_Spider mites', 10           |
| 12 | ./test/PotatoEarlyBlight3.JPG     | PotatoEarlyBlight3.JPG     | 19                | '14_Tomato_Illness_Target Spot', 11            |
| 13 | ./test/PotatoEarlyBlight4.JPG     | PotatoEarlyBlight4.JPG     | 35                | '14_Tomato_Illness_Yellow Leaf Curl Virus', 12 |
| 14 | ./test/PotatoEarlyBlight5.JPG     | PotatoEarlyBlight5.JPG     | 35                | '14_Tomato_healthy', 13                        |
| 15 | ./test/PotatoHealthy1.JPG         | PotatoHealthy1.JPG         | 37                | '1_Apple_Illness_rust', 14                     |
| 16 | ./test/PotatoHealthy2.JPG         | PotatoHealthy2.JPG         | 37                | '1_Apple_Illness_scab', 15                     |
| 17 | ./test/TomatoEarlyBlight1.JPG     | TomatoEarlyBlight1.JPG     | 7                 | '1_Apple_Illness_Black rot', 16                |
| 18 | ./test/TomatoEarlyBlight2.JPG     | TomatoEarlyBlight2.JPG     | 8                 | '1_Apple_healthy', 17                          |
| 19 | ./test/TomatoEarlyBlight3.JPG     | TomatoEarlyBlight3.JPG     | 12                | '2_Blueberry_healthy', 18                      |
| 20 | ./test/TomatoEarlyBlight4.JPG     | TomatoEarlyBlight4.JPG     | 7                 | '2_Cherry_Illness_Powdery Mildew', 19          |
| 21 | ./test/TomatoEarlyBlight5.JPG     | TomatoEarlyBlight5.JPG     | 7                 | '2_Cherry_healthy', 20                         |
| 22 | ./test/TomatoEarlyBlight6.JPG     | TomatoEarlyBlight6.JPG     | 38                | '3_Corn_Illness_Common rust', 21               |
| 23 | ./test/TomatoHealthy1.JPG         | TomatoHealthy1.JPG         | 14                | '3_Corn_Illness_Northern Leaf Blight', 22      |
| 24 | ./test/TomatoHealthy2.JPG         | TomatoHealthy2.JPG         | 20                | '3_Corn_Illness_Cercospora', 23                |
| 25 | ./test/TomatoHealthy3.JPG         | TomatoHealthy3.JPG         | 11                | '4_Grape_Esca', 24                             |
| 26 | ./test/TomatoHealthy4.JPG         | TomatoHealthy4.JPG         | 14                | '4_Grape_Illness_Black rot', 25                |
| 27 | ./test/TomatoYellowCurlVirus1.JPG | TomatoYellowCurlVirus1.JPG | 12                | '4_Grape_Illness_Leaf blight', 26              |
| 28 | ./test/TomatoYellowCurlVirus2.JPG | TomatoYellowCurlVirus2.JPG | 12                | '5_Grape_healthy', 27                          |
| 29 | ./test/TomatoYellowCurlVirus3.JPG | TomatoYellowCurlVirus3.JPG | 12                | '5_Grape_Illness_Haunglongbing', 28            |
| 30 | ./test/TomatoYellowCurlVirus4.JPG | TomatoYellowCurlVirus4.JPG | 12                | '6_Orange_Illness_Haunglongbing', 29           |
| 31 | ./test/TomatoYellowCurlVirus5.JPG | TomatoYellowCurlVirus5.JPG | 12                | '6_Orange_Illness_Bacterial spot', 30          |
| 32 | ./test/TomatoYellowCurlVirus6.JPG | TomatoYellowCurlVirus6.JPG | 12                | '7_Peach_Illness_Bacterial spot', 31           |

L'analyse des prédictions, nous donne 25 prédictions correctes sur les 33 fichiers, ce qui nous donne une précision du modèle de 0.75. Ce modèle est toujours plus performant que le modèle à 2 étapes.

## Optimisation des modèles

Nous avons conçu différents types de modèles pour chacune des 14 classes, notamment en utilisant le transfert learning avec DenseNet121. Ce modèle préconfiguré est un réseau dense qui a été entraîné sur le jeu de données Imagenet et a une profondeur de 121 couches. Contrairement à RESNET, qui combine les couches à l'aide de l'addition, DenseNet combine les couches à l'aide de la concaténation.

Nous avons principalement utilisé les fonctions de perte 'sparse\_categorical\_crossentropy' et 'categorical\_crossentropy', ainsi que les métriques 'accuracy' et 'acc'. Nous avons également testé différentes fonctions d'activation telles que 'relu' et 'sigmoid', avec une

dernière couche en 'softmax'. Cependant, nous avons observé de l'overfitting, et avons donc exploré différents callbacks tels que 'learning-rate' ou 'checkpoint', ainsi que des layers de normalisation, de régularisation et de pooling, notamment ceux inclus dans les modèles préconfigurés.

| Model: "sequential_2"                        |              |          |
|--|--------------|----------|
| Layer (type)                                 | Output Shape | Param #  |
| inception_resnet_v2 (Functional)             | (None, 1000) | 55873736 |
| batch_normalization_203 (BatchNormalization) | (None, 1000) | 4000     |
| dense_12 (Dense)                             | (None, 64)   | 64064    |
| dropout_16 (Dropout)                         | (None, 64)   | 0        |
| dense_13 (Dense)                             | (None, 128)  | 8320     |
| dense_14 (Dense)                             | (None, 256)  | 33024    |
| dropout_17 (Dropout)                         | (None, 256)  | 0        |
| dense_15 (Dense)                             | (None, 512)  | 131584   |
| dense_16 (Dense)                             | (None, 1024) | 525312   |
| dropout_18 (Dropout)                         | (None, 1024) | 0        |
| dense_17 (Dense)                             | (None, 144)  | 147600   |
| dense_18 (Dense)                             | (None, 64)   | 9280     |
| dense_19 (Dense)                             | (None, 32)   | 2080     |
| dropout_19 (Dropout)                         | (None, 32)   | 0        |
| dense_20 (Dense)                             | (None, 38)   | 1254     |

Total params: 56,800,254  
Trainable params: 924,518  
Non-trainable params: 55,875,736

---

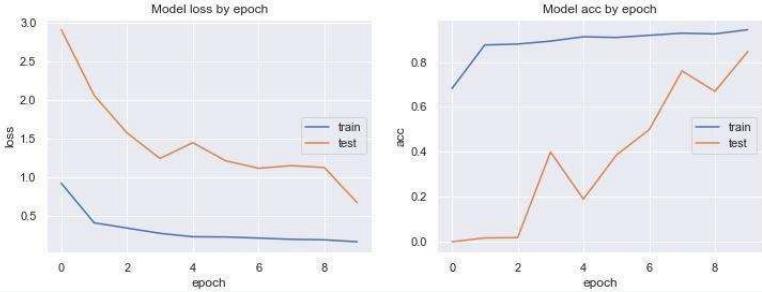
| Model: "sequential_1"                    |              |         |
|--|--------------|---------|
| Layer (type)                             | Output Shape | Param # |
| densenet121 (Functional)                 | (None, 1000) | 8062504 |
| batch_normalization (BatchNormalization) | (None, 1000) | 4000    |
| dense (Dense)                            | (None, 1024) | 1025024 |
| dense_1 (Dense)                          | (None, 512)  | 524800  |
| dropout (Dropout)                        | (None, 512)  | 0       |
| dense_2 (Dense)                          | (None, 382)  | 195966  |
| dense_3 (Dense)                          | (None, 144)  | 55152   |
| dense_4 (Dense)                          | (None, 88)   | 12760   |
| dense_5 (Dense)                          | (None, 32)   | 2848    |
| dropout_1 (Dropout)                      | (None, 32)   | 0       |
| dense_6 (Dense)                          | (None, 12)   | 396     |

Total params: 9,883,450  
Trainable params: 1,818,946  
Non-trainable params: 8,064,504

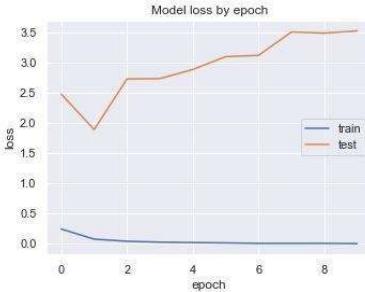
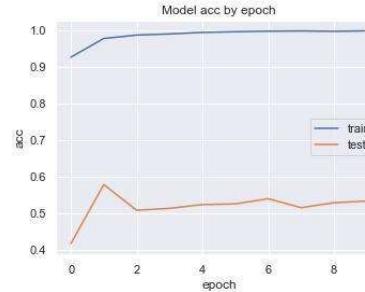
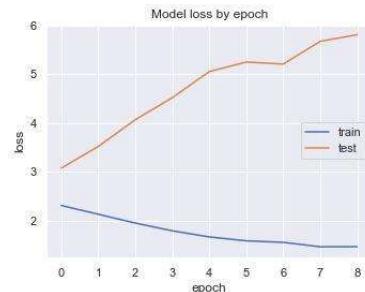
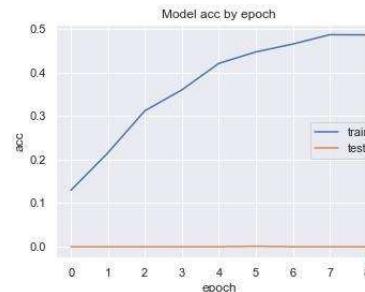
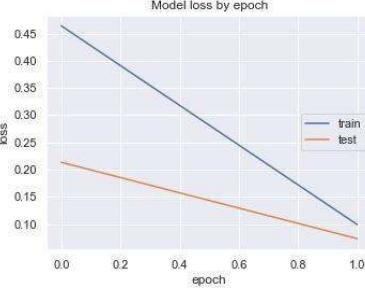
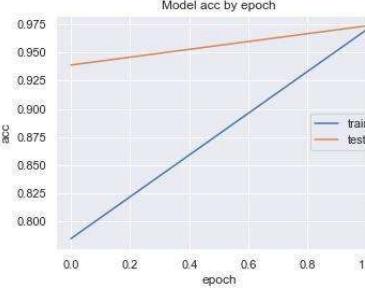
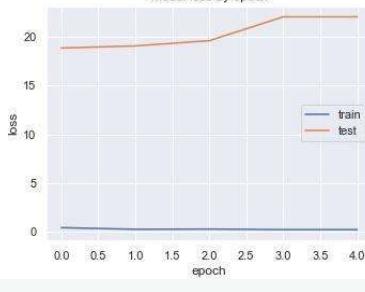
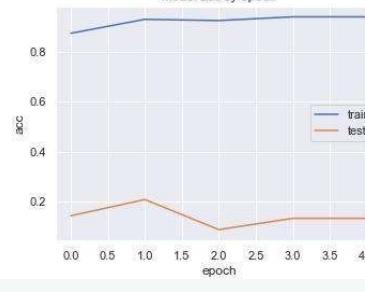
Légende

Les couches de convolution ont principalement été en Conv2D ou Dense. Bien que d'autres types de couches telles que SeparableConv2D, Conv3D ou DepthwiseConv2D auraient pu être étudiées, leurs dimensions et paramètres sont peu intuitifs avec des modèles de transfert learning, nous avons donc choisi de les écarter.

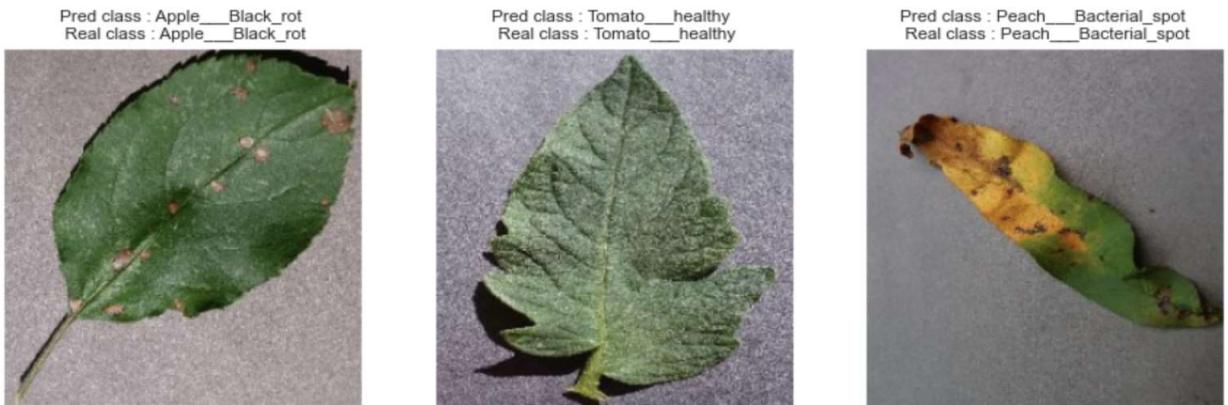
### Différents résultats selon par type de classification

| Classe de Plante | Modèle utilisé | Graphique de précision et de perte   |
|------------------|----------------|--|
| Pomme            | InceptionV3    |  |

## Différents résultats selon par type de classification

|                |                   |   |
|----------------|-------------------|---|
| Pomme          | Conv2D            |       |
| Tomato         | Conv2D            |       |
| Fraise         | DenseNet121       |   |
| Type de plante | InceptionResnetV2 |   |

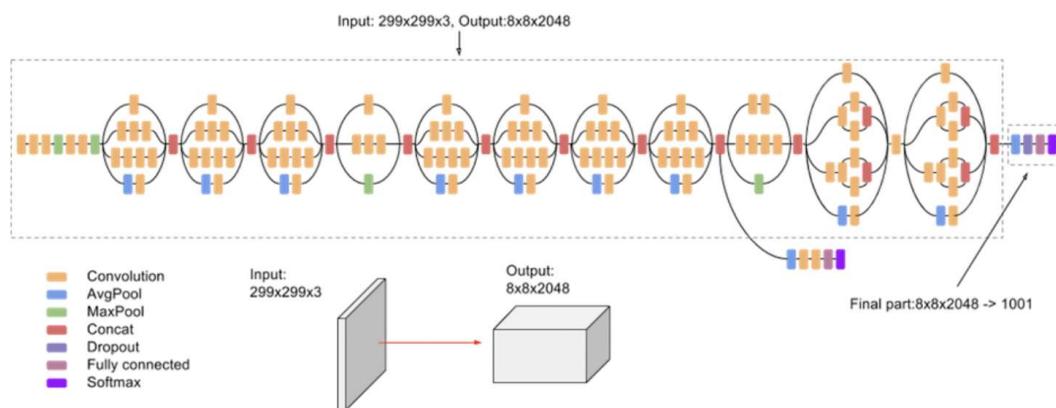
A l'aide de fonctions de visualisation quelques observations de classes peuvent être observées.



Des modèles plus complexes ont commencé à développer avec une architecture plus profonde et complexe que nos modèles initiaux. Ils prennent également plus de temps à entraîner en raison de leur taille et de leur quantité de paramètres. Malgré cela, nous sommes convaincus qu'ils pourraient fournir des prédictions plus précises et plus fiables pour notre classification à long terme. Pour le moment, ces modèles ne sont que des prototypes et nécessitent encore beaucoup de travail et d'optimisation avant d'être déployés en production.

Quelques détails sur les modèles :

Inception v3 est un modèle de reconnaissance d'image qui a atteint une précision supérieure à



78,1% sur l'ensemble de données ImageNet. Ce modèle est l'aboutissement de nombreuses idées développées par plusieurs chercheurs au fil des ans.

En voici un diagramme :

Le modèle lui-même est composé de composants symétriques et asymétriques, dont les convolutions, le pool moyen, le pool maximal, les concaténations, les abandons et les couches entièrement connectées. La normalisation des lots est utilisée de manière approfondie dans le modèle et appliquée aux entrées d'activation. La perte est calculée à l'aide de Softmax.

```
Model: "sequential_3"
Layer (type)          Output Shape       Param #
=====
inception_v3 (Functional)    (None, 1000)      23851784
batch_normalization_298 (BatchNormalization) (None, 1000)      4000
dense_21 (Dense)           (None, 1024)       1025024
batch_normalization_299 (BatchNormalization) (None, 1024)      4096
dense_22 (Dense)           (None, 512)        524800
dropout_20 (Dropout)        (None, 512)        0
dense_23 (Dense)           (None, 382)        195966
dense_24 (Dense)           (None, 144)         55152
dense_25 (Dense)           (None, 88)          12760
dense_26 (Dense)           (None, 32)          2848
dropout_21 (Dropout)        (None, 32)          0
dense_27 (Dense)           (None, 38)          1254
=====
Total params: 25,677,684
Trainable params: 1,821,852
Non-trainable params: 23,855,832
```

## Légende

# Conclusion sur le projet

Pour conclure sur le projet, nous pouvons nous attarder sur les résultats de précision de chaque modèle (modèle de classification en 1 étape versus modèle de classification à 2 étapes). Nous avons pu observer, lors des entraînements des modèles que les précisions sont équivalentes et excellentes (au-delà de 0.90). Cependant lors du passage à l'étape de test, les modèles à 1 étape (38 classes) obtiennent de meilleurs résultats de prédiction.

Il est nécessaire de prendre en considération le temps d'entraînement qui peut être avantageux dans le modèle à 2 étapes puisqu'avec une machine puissante, l'entraînement des modèles pour chaque plante pourrait se faire en parallèle. A l'inverse, les modèles plus complexes sont moins rapides à entraîner.

## Difficultés rencontrées lors du projet

La richesse du dataset avec des données augmentées nous a amené à une gestion du volume de données nécessaire pour pouvoir entraîner nos modèles: obligation de réduire le dataset pour pouvoir entraîner un modèle de 38 classes.

Lors des premiers essais de modélisation, l' entraînement de l'ensemble du dataset nous a vite amené des problèmes quant à la limite dans la puissance de stockage et la mémoire ram pour exécuter des programmes en ligne (colab par ex). Nous avons dû travailler en local.

Le jeu de données est composé exclusivement de fichiers images organisés en dossier et sous dossier, sans support .csv. Il nous a été nécessaire de trouver une façon de créer nos propres bases de données (type dataframe) ce qui nous a ralenti dans la mise en route et l'exécution du projet (essai de création d'un dataframe avec toutes les données pixel, découverte du module os pour naviguer dans les dossiers/sous dossiers)

Le sujet de classification de plantes à partir d'images étant un sujet de deep learning, nous avons découvert une multitude de possibilités en termes de création de CNN + combinaison de transfer learning. Il a été impossible de tester l'ensemble de ces solutions, nous avons dû faire des choix dans les sélections de modèles au vu des temps d'entraînement

L'acquisition des compétences de deep learning en cours de projet n'a pas été simple et a ralenti notre progression, les derniers modules de la formation sont de solides supports pour compléter le projet.

## Suite du projet

Bien que nos premiers algorithmes décrits dans ce rapport nous donnent des résultats très encourageants, il peut toujours être amélioré pour atteindre des niveaux de précision d'excellence. Pour cela, nous avons plusieurs pistes de travail en cours ou à explorer telles que:

- l'utilisation d'autres méthodes de réduction de dimension type PCA
- l'exploration des données d'entraînement et effectuer une réduction de données différentes de celle utilisée à ce stade du projet
- l'observation de maladies communes pour certaines plantes: , il serait intéressant d'explorer cette caractéristique pour le modèle à 2 étapes (et notamment l'étape de prédiction de l'état) Elle peut être efficace lorsque certaines plantes qui ne sont pas entraînées sur le modèle de reconnaissance, partagent les mêmes caractéristiques de maladie.
- une mise en relation avec des professionnels dans la recherche des maladies de plantes pourrait permettre d'aller plus loin dans l'analyse et l'intégration de variables (telles que des dimensions de feuilles, des caractéristiques typiques de maladies etc...) et ainsi ne pas se restreindre uniquement aux données pixels d'une image
- L'intégration du modèle à dans une application web pour être mis à la disposition des utilisateurs (recherche agronome, public), cependant un enrichissement des classes entraînées est nécessaire.



# Bibliographie

recherche sur le sujet:

- [La gestion des risques en agriculture | Ministère de l'Agriculture et de la Souveraineté alimentaire](#)
- [2000-2016 / Le diagnostic des maladies des plantes en mode geek | INRAE INSTIT](#)

l'augmentation des données:

- [Augmentation d'images pour améliorer les modèles Machine Learning P1 \(invivoo.com\)](#)

bibliothèques:

- <https://keras.io/api/>
- <https://pythonforge.com/module-os-systeme-dexploitation/>
- [https://www.tensorflow.org/?gclid=Cj0KCQiApKagBhC1ARIsAFc7Mc4x5wUn08-8gBu8FroXqrnYILA4N-YMpMpRkT3Bfs3WZnyUSTuzrkaAocDEALw\\_wcB&hl=fr](https://www.tensorflow.org/?gclid=Cj0KCQiApKagBhC1ARIsAFc7Mc4x5wUn08-8gBu8FroXqrnYILA4N-YMpMpRkT3Bfs3WZnyUSTuzrkaAocDEALw_wcB&hl=fr)
- [https://www.tensorflow.org/tutorials/keras/save\\_and\\_load?hl=fr](https://www.tensorflow.org/tutorials/keras/save_and_load?hl=fr)

modèles:

- <https://datascientest.com/quest-ce-que-le-modele-vgg>

Résolution de problèmes:

- <https://stackoverflow.com/questions/tagged/python>

## Annexes

- [Diagramme de Gantt.](#)
- Essai de classification de l'état de la plante par Machine Learning
- Description des fichiers de code:
  - partie exploration des données:
    - [notebook 1 exploration des sources de données](#)
    - [rapport d'exploration des données](#)
    - [notebook 4h: Exploration des couleurs des plantes](#)
    - [exploration du projet \(calque\)](#)
  - partie modélisation 2 étapes::
    - [notebook modélisation 2 étapes: étape 1 type de plantes\\_dataset healthy](#)
    - [notebook modélisation 2 étapes: étape 1 type de plante\\_dataset réduit](#)
    - [notebook boucle de prédiction modèle 2 étapes](#)
  - partie modélisation 1 étape (38 classes):
    - [notebook 4h: Modélisation 38 classes](#)
    - [Approche technique finale](#)

# Modèle de Machine Learning: utilisation du classifier Random Forest

La première étape que nous avons menée a été de tester une classification simple par machine learning en utilisant le classificateur Random Forest (sans fixation de paramètres) sur la catégorie des pommes (Apple). La problématique pour ce cas étant de classifier correctement si la plante est saine ("Healthy") ou si elle est malade ("Black\_rot", "Apple\_scab", "Cedar\_apple\_rust").

Une image étant composée d'un array avec 3 dimensions (RGB), il nous a fallu au préalable, transformer les données 3d en un vecteur plat regroupant les données de tous les pixels de l'image et reconnaissable par le modèle de Machine Learning.

La lecture et la transformation des fichiers images de chaque sous-dossier de Apple a été faite en utilisant le module CV2 (méthodes imread et resize pour redimensionner les images en 100,100).

Puis nous avons réalisé une boucle permettant de compléter un dataframe initialement vide. Une ligne de dataframe correspondant aux variables d'une image.

Le dataframe contient au final n\_lignes (correspondant au nombre de fichiers image dans le dossier) et 30 001 colonnes (30 000 correspondant au données pixel (100\*100\*3) et 1 colonne représentant l'état de la plante ("healthy", "Black\_rot", "Apple\_scab", "Cedar\_apple\_rust"), dans notre cas cela est également la variable cible.

En utilisant la méthode value\_counts de pandas nous obtenons la répartition des différentes classes de notre dataframe (df\_Apple\_RF):

| classe/variable cible | count :nombre de lignes(=images) par classe |
|-----------------------|---|
| Healthy               | 2008  |
| Black_rot             | 1987  |
| Apple_scab            | 2016  |
| Cedar_apple_rust      | 1760  |

Après avoir séparé nos variables explicatives, de notre variable cible ("etat"), nous avons appliqué la méthode train\_test\_split en conservant 20% du dataset pour la taille de test. L'entraînement de ce modèle est extrêmement rapide sur les 4 classes de Apple. Nous constatons un précision générale du modèle de 0.89, le modèle étant précis pour la classe 3 (Cedar\_apple\_rust) à 0.95. Cependant le modèle fait des erreurs dans la prédiction et confond la classe 1 (black\_rot) avec la classe 2 (apple\_scab), de même pour la classe 2 souvent confondue avec la classe 0 et la 1.

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0            | 0.88      | 0.93   | 0.90     | 396     |
| 1            | 0.86      | 0.82   | 0.84     | 408     |
| 2            | 0.87      | 0.89   | 0.88     | 395     |
| 3            | 0.95      | 0.92   | 0.94     | 356     |
| accuracy     |           |        | 0.89     | 1555    |
| macro avg    | 0.89      | 0.89   | 0.89     | 1555    |
| weighted avg | 0.89      | 0.89   | 0.89     | 1555    |

[rapport de classification RF sur Apple](#)

```
array([[367,  20,   8,   1],
       [ 22, 336,  38,  12],
       [ 18,  24, 350,   3],
       [ 10,  10,   8, 328]], dtype=int64)
```

[Matrice de confusion RF sur Apple](#)

### Conclusion sur le modèle de Random Forest:

Le modèle de machine learning Random forest est très rapide à entraîner, il obtient une précision générale de 0.89 qui reste correcte mais peut certainement être amélioré en ajoutant des caractéristiques (variables) telles que des moyennes de couleurs, des longueurs / largeurs de feuilles, ... . Cependant, la structure des données en dataframe et donc vecteurs plats est très volumineuse en termes de stockage et l'entraînement de plusieurs classes à la fois sur ce type de modèle est très compliqué car nous nous confrontons à la limite de mémoire ram de nos machines.