Classification des maladies des vignes en utilisant les CNN et le Transfer Learning

Classification des maladies des vignes en utilisant les CNN et le Transfer Learning 1

1. Introduction 2
2. Fondements théoriques 2

Réseaux de neurones convolutifs (CNN) 2

Transfer Learning 4

3. Mise en œuvre 5

Présentation du jeu de données 5

Prétraitement des données 5

Choix du modèle CNN pré-entraîné 6

Mise en place du modèle CNN 8

4. Évaluation des performances 9
5. Conclusion 10

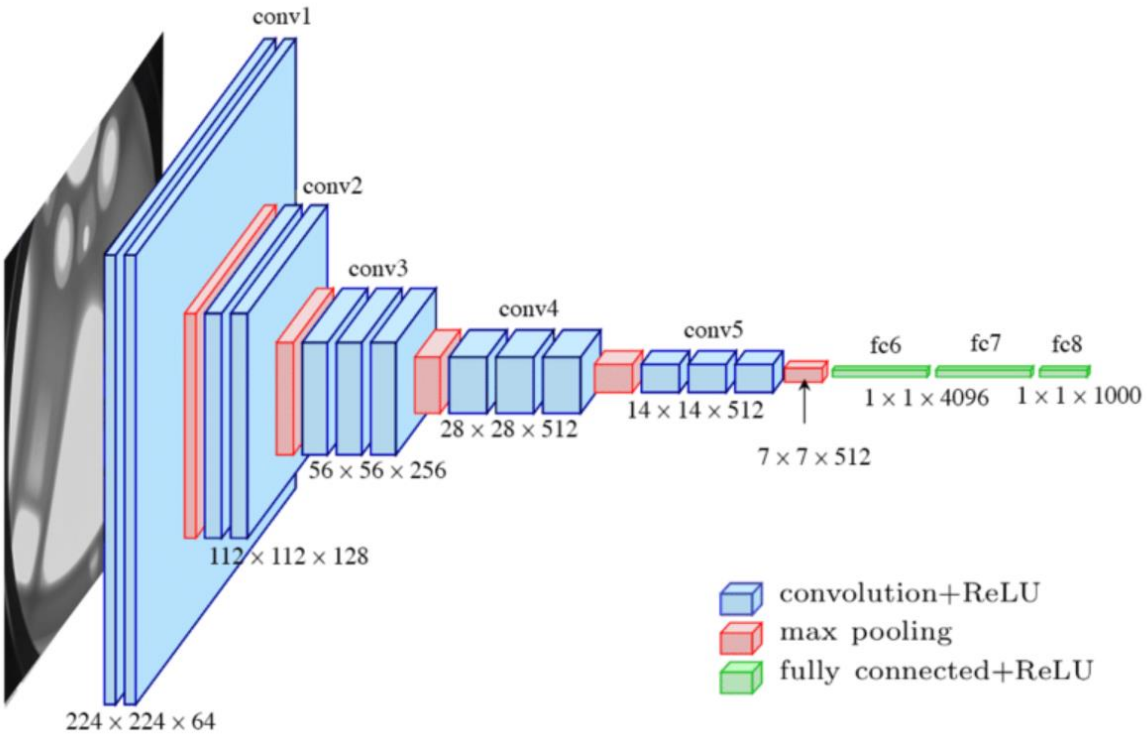
1. Introduction

Dans cette documentation technique, nous allons explorer l'utilisation des réseaux de neurones convolutifs (CNN) pour la classification automatique des maladies des feuilles de vigne. Le transfer learning sera également utilisé pour tirer parti des connaissances préalables d'un modèle CNN pré-entraîné, ce qui permettra de bénéficier de la puissance de ces modèles même avec des ensembles de données de petite taille.

2. Fondements théoriques

Réseaux de neurones convolutifs (CNN)

Les réseaux de neurones convolutifs sont une classe de modèles d'apprentissage profond largement utilisés pour l'analyse et la classification d'images. Inspirés par le fonctionnement du cortex visuel du cerveau humain, les CNN sont spécifiquement conçus pour extraire des caractéristiques significatives des images, permettant ainsi de résoudre des problèmes de vision par ordinateur.



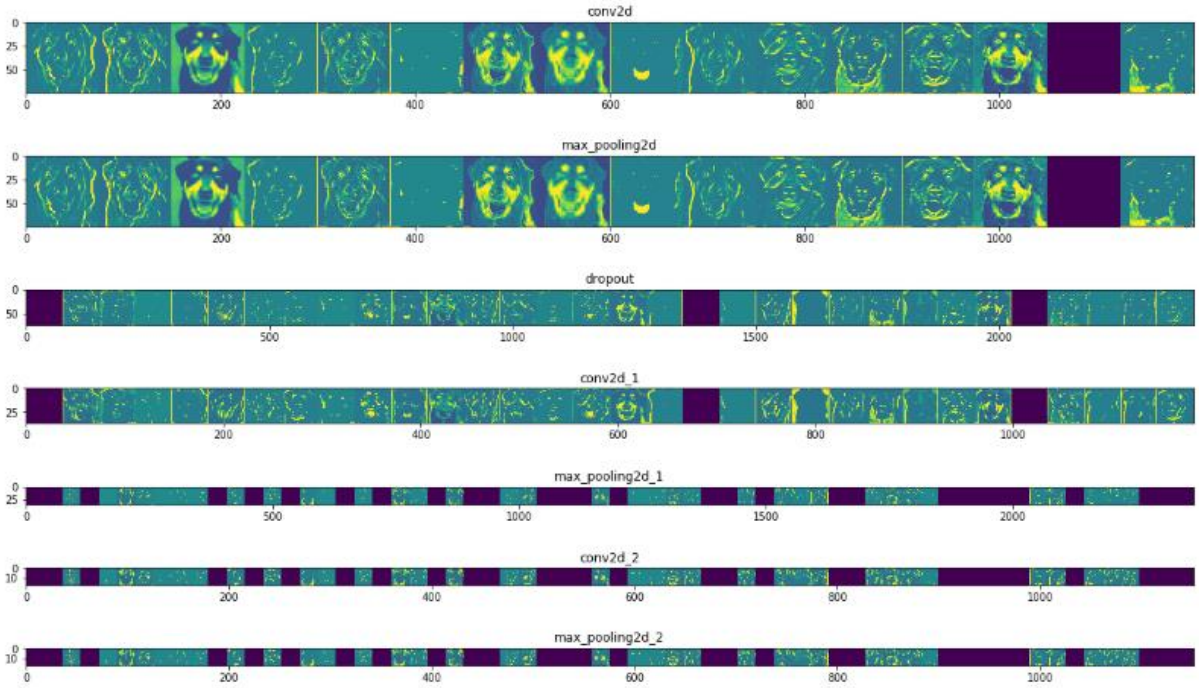
Un CNN est composé de plusieurs couches empilées, chacune ayant un rôle spécifique dans le processus d'analyse et de classification des images. Les principales couches d'un CNN sont les suivantes :

- **Couches convolutives** : Les couches convolutives sont responsables de la détection des motifs et des caractéristiques dans les images. Chaque couche convolutive est constituée de plusieurs filtres (ou noyaux) qui glissent sur l'image en effectuant des opérations de convolution. Ces filtres permettent de détecter des motifs tels que des bords, des textures ou des formes dans différentes parties de l'image.

- **Couches de pooling** : Les couches de pooling sont utilisées pour réduire la taille spatiale des représentations obtenues après les couches convolutives. La méthode de pooling la plus courante est le max-pooling, qui consiste à prendre la valeur maximale d'une région donnée dans la carte d'activation. Cela permet de réduire le nombre de paramètres du modèle, tout en maintenant les informations essentielles sur les caractéristiques détectées.

- **Couches complètement connectées** : Après plusieurs couches convolutives et de pooling, les informations sont aplatie (flattened) pour être passées à des couches complètement connectées, également appelées couches denses. Ces couches sont similaires aux couches d'un réseau de neurones traditionnel et sont utilisées pour combiner les caractéristiques extraites afin de produire les prédictions finales.

Voici un exemple de ce que ça donne avec une image de chien :



Lorsque nous appliquons l'image d'un chien à travers les différentes couches du réseau de neurones convolutifs (CNN), le modèle parvient à extraire des caractéristiques précises, telles que les oreilles, les yeux, etc. Au fur et à mesure que l'information progresse plus en profondeur dans les couches du CNN, le modèle peut identifier des motifs plus complexes et spécifiques, ce qui permet une meilleure reconnaissance de l'image du chien dans son ensemble. Le CNN est capable de capturer les traits distinctifs du chien et de le classifier de manière précise en tant qu'objet spécifique dans l'image.

De la même façon qu'avec l'exemple du chien ce modèle nous permettra d'extraire les caractéristiques des images de feuilles de vignes pour les classifier avec une précision maximale. Il produira en sortie un vecteur de valeurs comprises entre 0 et 1, reflétant la probabilité que l'image d'entrée appartienne à une classe spécifique.

Transfer Learning

Le transfer learning consiste à utiliser les connaissances acquises par un modèle pré-entraîné sur une tâche source pour améliorer les performances d'un modèle sur une tâche cible similaire. Cette approche est particulièrement utile lorsque nous avons une quantité limitée de données disponibles pour la tâche cible, ce qui est notre cas avec les images de feuilles de vignes.

Le processus de transfer learning implique généralement deux étapes :

- **Pré-entraînement du modèle source** : Dans cette étape, un modèle de réseau de neurones est entraîné sur une tâche source à l'aide d'un ensemble de données volumineux et représentatif. Ce modèle apprend des représentations visuelles riches et généralisables des caractéristiques des images. Nous utiliserons un modèle déjà entrainé dans notre cas, modèle dont on détaillera l'architecture plus tard

- **Fine-tuning (Adaptation) sur la tâche cible** : Une fois le modèle source pré-entraîné, nous pouvons l'utiliser comme point de départ pour la tâche cible spécifique, dans notre cas, la classification des maladies des feuilles de vigne. Au lieu de partir de zéro, nous gardons généralement les couches convolutives du modèle source intactes car elles contiennent déjà des caractéristiques utiles pour la vision. Cependant, nous remplaceront les couches de classification par de nouvelles couches adaptées au nombre de maladies plus non malade. Ces nouvelles couches seront ensuite entraînées uniquement sur l'ensemble de données de la tâche cible.

Le transfert de connaissances du modèle source vers la tâche cible présente plusieurs avantages significatifs :

- **Meilleure généralisation** : Les caractéristiques apprises par le modèle source sur des données diverses et massives sont souvent plus généralisables. En transférant ces caractéristiques vers la tâche cible, nous pouvons mieux généraliser à partir d'un ensemble de données plus restreint, ce qui est particulièrement bénéfique lorsque les données de la tâche cible sont limitées.

- **Réduction du risque de surapprentissage (overfitting)** : En utilisant un modèle pré-entraîné, nous réduisons la probabilité de surapprentissage car le modèle source a déjà été régularisé par l'ensemble de données de la tâche source.

En utilisant cette technique, nous pouvons construire un modèle performant pour la classification des maladies de feuilles de vignes même avec des ensembles de données plus petits et tirer pleinement parti de l'expertise acquise par des modèles pré-entraînés sur de vastes ensembles de données.

3. Mise en œuvre

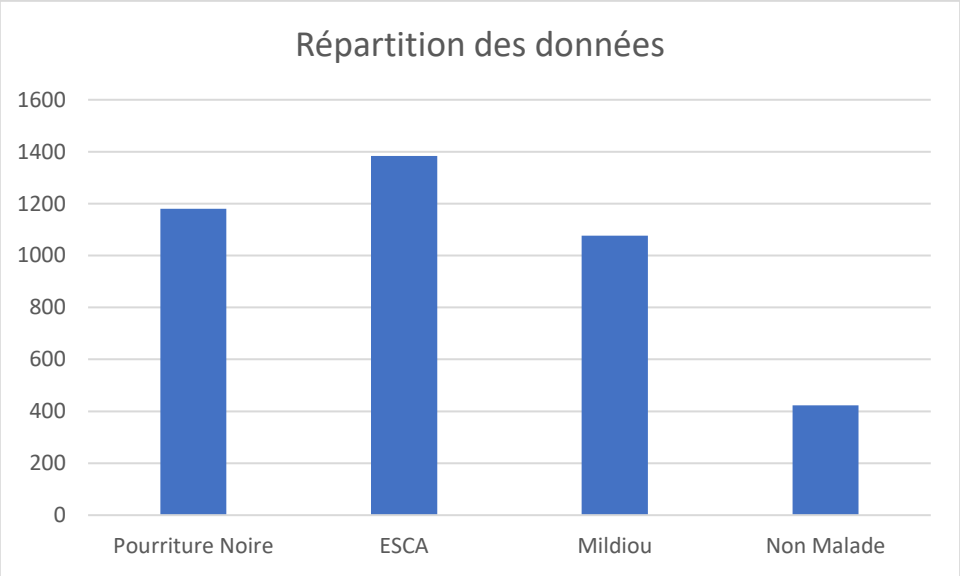
Présentation du jeu de données

Pour entrainer notre modèle nous allons utiliser un jeu de données récupérer contenant actuellement 3 maladies :

- Mildiou
- Maladie de l'esca
- Pourriture noire

Et des images de feuilles de vignes saines.

Voici la représentions graphique de notre un jeu de données à ce jour il pourra être enrichi avec d'autres images à l'avenir :



Prétraitement des données

Avant d'entrainer notre modèle nous devons venir traiter nos images, afin d'optimiser au mieux l'apprentissage du CNN :

- **Normalisation des données** : Les valeurs des pixels dans les images peuvent varier sur une large échelle, ce qui peut entraîner des problèmes de convergence lors de l'entraînement. Pour résoudre cela, les pixels des images sont convertis en échelle de valeurs entre 0 et 1 en divisant chaque valeur de pixel par 255. Cette mise à l'échelle assure que toutes les valeurs de pixel sont dans la même plage, ce qui facilite l'optimisation du modèle.

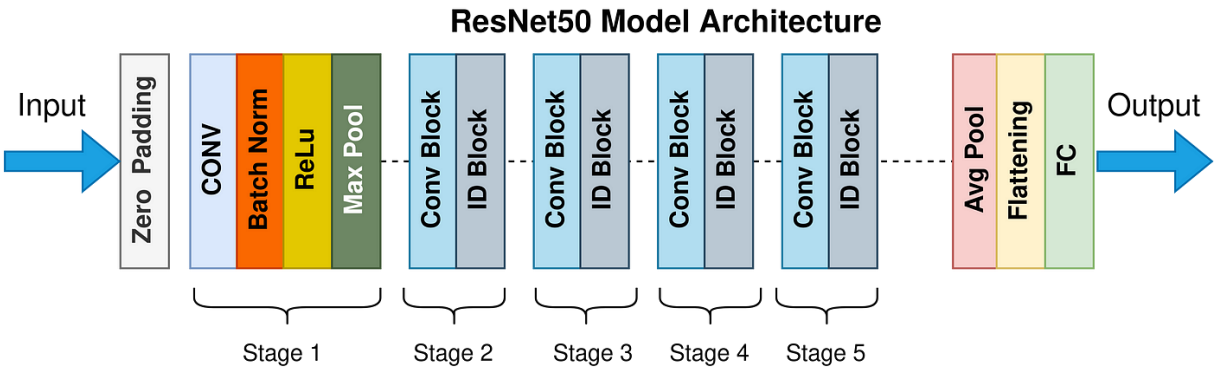
- **Augmentation des données (Data Augmentation)** : Appliquer des transformations aléatoires sur les images existantes pour créer de nouvelles variations. Cela aide notre modèle à généraliser mieux et à être plus robuste face à des variations mineures dans les images réelles. Les transformations typiques incluent des rotations, des retournements, des changements d'échelle, des zooms, des translations.

- **Séparation des ensembles d'entraînement, de validation et de test** : Une fois les données prétraitées, l'ensemble de données est divisé en trois parties distinctes : l'ensemble d'entraînement, l'ensemble de validation et l'ensemble de test. L'ensemble d'entraînement est utilisé pour entraîner le modèle, l'ensemble de validation est utilisé pour ajuster les hyperparamètres du modèle et éviter le surapprentissage, tandis que l'ensemble de test est utilisé pour évaluer les performances finales du modèle de manière impartiale.

Ces étapes sont cruciales pour que notre modèle ne fasse pas du surapprentissage (overfitting).

Choix du modèle CNN pré-entraîné

Pour effectuer notre transfer learning discuté plus haut nous avont besoin d'un modèle pré-entrainé, pour cela nous allons utilisé ResNet50 :



layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Comme nous pouvons le voir dans le tableau, l'architecture de ResNet 50 contient les éléments suivants :

1. Une convolution avec une taille de noyau de 7 * 7 et 64 noyaux différents, tous avec une taille de pas (stride) de 2, ce qui nous donne 1 couche.
2. Ensuite, nous voyons un max pooling avec une taille de pas de 2.
3. Dans la convolution suivante, il y a un noyau 1 * 1 de 64, suivi d'un noyau 3 * 3 de 64, et enfin d'un noyau 1 * 1 de 256. Ces trois couches sont répétées au total 3 fois, ce qui nous donne 9 couches à cette étape.
4. Ensuite, nous voyons un noyau de 1 * 1 de 128, suivi d'un noyau de 3 * 3 de 128, et enfin d'un noyau de 1 * 1 de 512. Cette étape est répétée 4 fois, ce qui nous donne 12 couches à cette étape.
5. Après cela, il y a un noyau de 1 * 1 de 256, puis deux autres noyaux de 3 * 3 de 256 et 1 * 1 de 1024. Cette étape est répétée 6 fois, donnant un total de 18 couches.
6. Ensuite, nous avons à nouveau un noyau de 1 * 1 de 512, suivi de deux autres de 3 * 3 de 512 et 1 * 1 de 2048, répétés 3 fois, ce qui donne un total de 9 couches.
7. Après cela, nous effectuons un pooling moyen (average pooling) et terminons avec une couche complètement connectée contenant 1000 nœuds, suivie à la fin d'une fonction softmax, ce qui nous donne 1 couche.

Nous ne comptons pas réellement les fonctions d'activation et les couches de pooling max / average.

Ainsi, en totalisant cela, nous obtenons un réseau convolutif profond de 50 couches, d’où ResNet50.

Utiliser ResNet50 pour la classification des maladies des vignes est pertinent pour plusieurs raisons :

- **Architecture profonde et performante** : ResNet50 est un réseau de neurones convolutifs (CNN) profond, composé de 50 couches, qui a été éprouvé et largement utilisé dans de nombreuses tâches de classification d'images. Cette architecture profonde permet au modèle de capturer des caractéristiques complexes et hiérarchiques à partir des images, ce qui est essentiel pour distinguer efficacement les différentes maladies des feuilles de vigne.

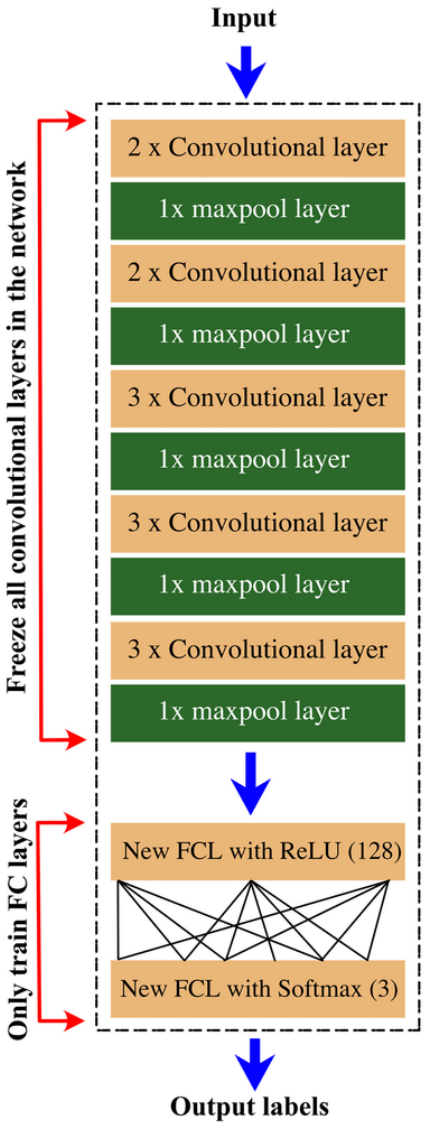
- **Transfer learning** : Comme mentionné précédemment, ResNet50 est pré-entraîné sur des ensembles de données massifs, tels que ImageNet, contenant des millions d'images appartenant à différentes classes. Cela signifie que le modèle a déjà acquis une connaissance générale des caractéristiques visuelles à partir de cet entraînement préalable.

- **Disponibilité et facilité d'utilisation** : ResNet50 est largement utilisé et bien documenté dans la communauté. De nombreuses implémentations sont disponibles dans des bibliothèques d'apprentissage automatique populaires telles que TensorFlow et PyTorch, ce qui facilite grandement son utilisation et son intégration dans notre projet de classification de maladies de feuilles de vigne.

Mise en place du modèle CNN

Nous devons remplacer les dernières couches de notre modèle ResNet50 pour s'adapter à notre tâche de classification spécifique des maladies des feuilles de vigne. Nous remplaçons les couches de classification d'origine du modèle, qui comportent 1000 nœuds pour les classes ImageNet, par de nouvelles couches de classification avec le nombre de classes spécifique à notre problème, dans notre cas actuel 4 (Non Malade, Mildiou, Pourriture noire et Maladie de l'esca). Ces nouvelles couches seront initialisées aléatoirement.

Notre jeu de données étant petit, il sera aussi judicieux de congeler les couches convolutives pré-entraînées du modèle ResNet50. Cela signifie que les poids de ces couches ne seront pas mis à jour lors de l'entraînement, empêchant ainsi la perte des connaissances acquises lors de l'entraînement initial sur ImageNet.



Le choix des hyperparamètres du modèle n'est pas encore défini, il le sera lors de test sur notre modèle via une Recherche par grille (Grid Search) qui consiste à spécifier une grille de valeurs pour chaque hyperparamètre que l'on souhaite optimiser. Le processus essaie toutes les combinaisons possibles d'hyperparamètres dans la grille et évalue les performances du modèle pour chacune de ces combinaisons. Cela permet de trouver les meilleures valeurs d'hyperparamètres parmi celles spécifiées dans la grille.

4. Évaluation des performances

Afin d'évaluer notre modèle final nous utiliserons ces métriques ci-dessous pour chaque classe (mildiou, pourriture noire, esca et non malade) :

Matrice de confusion : La matrice de confusion permet d'analyser les performances du modèle pour chaque classe individuelle. Cela permettra de voir les vrais positifs, les vrais négatifs, les faux positifs et les faux négatifs pour chaque classe.

		Réel			
		Mildiou	Pourriture Noire	Esca	Saine
Prédiction	Mildiou	TP	FP	FP	FP
	Pourriture Noire	FN	TP	FP	FP
	Esca	FN	FN	TP	FP
	Saine	FN	FN	FN	TP

TP (True Positive) : Le nombre d'exemples correctement classés comme la classe correspondante (par exemple, feuilles "mildiou" prédites comme "mildiou").

FP (False Positive) : Le nombre d'exemples incorrectement classés comme la classe correspondante (par exemple, feuilles "mildiou" prédites comme "pourriture noire", "esca" ou "saine").

FN (False Negative) : Le nombre d'exemples de la classe correspondante qui ont été mal classés (par exemple, feuilles "mildiou" prédites comme "pourriture noire", "esca" ou "saine").

Précision : La précision mesure la proportion des exemples classés comme positifs pour une classe spécifique, par exemple "mildiou" qui sont réellement positifs parmi toutes les prédictions positives effectuées pour cette classe. Une haute précision indique que le modèle a une faible tendance à donner des faux positifs pour cette classe. Dans le contexte des maladies de feuilles de vigne, une haute précision signifie que lorsque le modèle prédit qu'une feuille appartient à une classe spécifique par exemple, "mildiou", il est généralement correct.

Recall : Le recall mesure la capacité du modèle à identifier correctement les exemples positifs d'une classe spécifique parmi tous les exemples réels de cette classe. Une haute valeur de rappel indique que le modèle ne manque pas de détecter les vrais positifs pour cette classe. Dans le contexte des maladies de feuilles de vigne, un rappel élevé signifie que le modèle a une faible tendance à donner des faux négatifs, c'est-à-dire qu'il est capable de bien détecter les feuilles réellement atteintes par une maladie donnée.

F1-score : Le F1-score est une métrique qui combine la précision et le rappel en une seule valeur. Le F1-score est particulièrement important dans notre cas car les classes sont déséquilibrées en taille (images non malade faible comparé aux malades), il donne une meilleure compréhension de la performance globale du modèle en tenant compte des faux positifs et des faux négatifs.

5. Conclusion

En conclusion, nous avons utilisé les réseaux de neurones convolutifs (CNN) en combinant le transfer learning pour classifier automatiquement les maladies des feuilles de vigne.

Le modèle pré-entraîné ResNet50 a été utilisé comme point de départ, permettant de bénéficier de ses connaissances préalables.

Cette approche nous a permis d'obtenir un modèle performant malgré un petit ensemble de données, grâce à la capacité du modèle à extraire des caractéristiques complexes des images.

Les avantages de cette méthode incluent une meilleure généralisation et une réduction du risque de surapprentissage, à l'avenir nous pourront venir ajuster les hyperparamètres pour améliorer encore les performances du modèle.