FACULTÉ DES SCIENCES ET TECHNOLOGIES

Université de Lille

Projet de février Master 2
Mathématiques et Applications
Ingénierie Statistique et Numérique

Transfer learning et classification d'images en écologie appliqués aux données peu nombreuses

Réalisé par :
Martin DESMAZIERES
Martin FASQUELLE
El Moustapha MALICK

Encadré par :
Charlotte BAEY

Année universitaire 2022 - 2023

Table des matières

Table des figures

Liste des tableaux

1 Introduction

De nos jours les méthodes de reconnaissance d'images ont des applications dans de nombreux secteurs et domaines. En écologie par exemple, la détection d'une espèce à partir d'une photographie permet d'automatiser le comptage et l'estimation statistique de l'abondance de la faune et de la flore qui est essentielle dans les efforts de conservation des écosystèmes face aux changements environnementaux (Torney et al., 2019 [1]). Ces images peuvent être obtenues aux moyens de pièges photographiques, de vues aériennes ou grâces à des participations citoyennes comme par exemple le projet Pl@ntNet. Malgré ces nombreux moyens, les données sont souvent trop peu nombreuses pour entraîner des algorithmes performants.

Or, l'essor de l'intelligence artificielle rend accessible de plus en plus d'algorithmes développés par diverses entités et structures. Des réseaux de neurones convolutifs (CNN) entraînés durant de nombreuses heures, spécifiquement pour la reconnaissance d'images, sur des jeux de données de plus en plus gros sont maintenant disponibles pour qui veut les utiliser. Ainsi, la solution à la taille limitée d'un jeu de données pourrait être le transfert learning qui consiste à utiliser un modèle déjà entraîné pour construire un nouveau modèle sur la base des données disponibles.

Dans ce projet on tente de construire un classifieur capable de reconnaître l'espèce d'une fleur parmi 17 espèces différentes à partir de sa photographie. La difficulté de ce projet repose dans le fait qu'on ne dispose que de 80 fleurs par espèces soit 1360 données au total ce qui est un peu juste pour développer un algorithme performant à partir de rien. Aussi nous nous intéressons aux méthodes existantes permettant de compenser la faible taille d'un jeu de données et plus particulièrement aux méthodes du transfert learning (TL). Un exemple des différentes espèces sont représentées dans la figure 5.

Dans un premier temps nous faisons un résumé du fonctionnement des réseaux de neurones convolutifs et du principe du transfert learning. Dans un second temps nous opérons une classification au moyen d'un réseau CNN, le réseau ResNet18 dont la structure est présentée en annexe, sans intervention de poids ou de données extérieures.

Dans un troisième temps nous nous intéressons à l'application des méthodes de TL. Enfin on compare les différentes méthodes en terme de performance et de temps d'apprentissage.

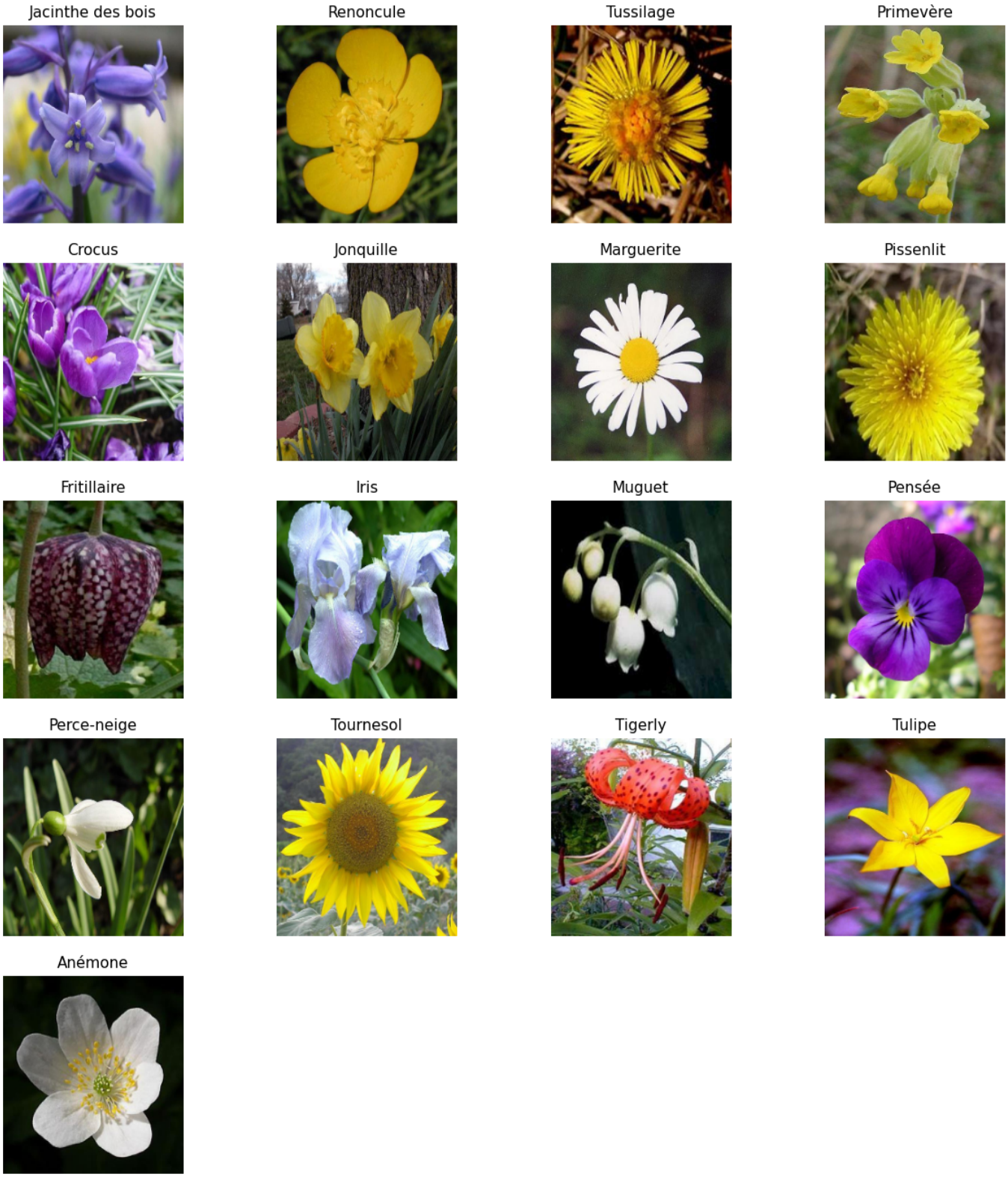


FIGURE 1 – Les dix-sept espèces de fleurs

2 Base théorique

L'apprentissage par transfert est surtout très exploité en *deep learning*. Les exemples d'applications les plus courants sont dans les domaines du traitement du langage naturel (*NLP*) et de la vision par ordinateur (*computer vision*). Les modèles de reconnaissance d'image s'appuient sur des réseaux de neurones assez particuliers : les réseaux de neurones convolutifs. Avant de rentrer en détail sur la méthode du transfert learning, nous exposons brièvement le fonctionnement de ce type de réseau.

2.1 Réseaux de neurones convolutifs (CNN)

Les réseaux de neurones à convolution sont une sous-catégorie de réseaux de neurones réputés pour être les plus performants dans la reconnaissance visuelle. Les images sont préalablement transformées en matrice de pixels en deux ou trois dimensions en fonction de si les images sont en niveau de gris ou en couleurs. Comme ces réseaux sont spécialement conçus pour traiter des images en entrée, ils disposent d'une architecture plus spécifique qu'un réseau de neurones complètement connecté. On distingue alors deux parties : une liée à l'extraction des caractéristiques (features) et l'autre à la reconnaissance (classification) des images.

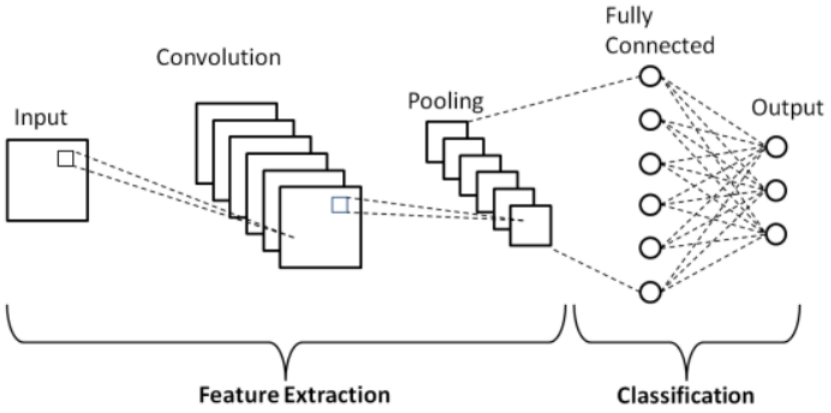


FIGURE 2 – Schéma de l'architecture d'un réseau de neurones convolutif

La seconde partie est commune à tout type de réseaux de neurones, mais le bloc d'extraction des caractéristiques est spécifique aux réseaux de neurones à convolution.

Cette partie du réseau est marquée par une succession de couches dites de convolution et de couches dites de pooling.

Les couches de convolution sont celles permettant l'extraction des caractéristiques. Elles utilisent des filtres (aussi appelés noyaux) pour détecter les motifs spécifiques des images. Ces filtres sont des matrices de nombres réels et sont appliqués à l'image d'entrée en effectuant une opération de convolution. Cela consiste à calculer la somme pondérée des valeurs de pixel recouvertes par le filtre à chaque position.

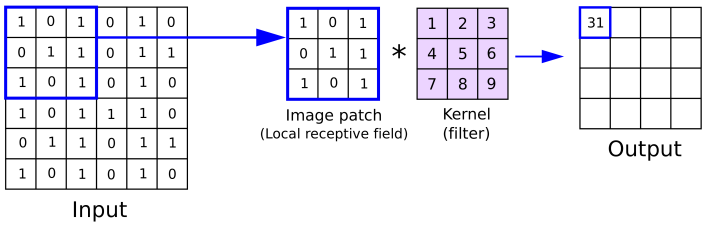


FIGURE 3 – Schéma explicatif d'une couche de convolution pour un filtre 3x3

Une couche de convolution est souvent suivie d'une couche de pooling. Ces couches sont utilisées pour réduire la taille spatiale des sorties des couches de convolution (les *feature maps*) en sous-échantillonnant les caractéristiques les plus importantes. L'intérêt est de réduire le nombre de paramètres du réseau, ce qui diminue les coûts de calcul et de stockage. Leurs présences peuvent également aider à prévenir du sur-apprentissage.

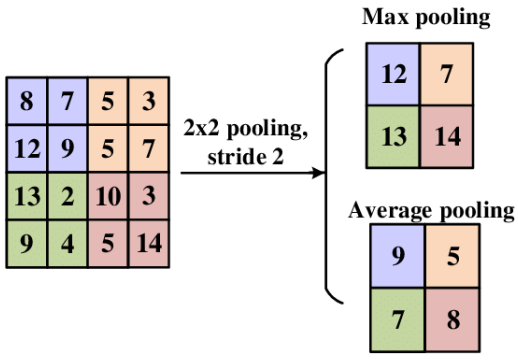


FIGURE 4 – Schéma explicatif d'une couche de pooling pour un découpage de taille 2x2

Il existe plusieurs types de pooling. L'exemple ci-dessus illustre les deux types les plus courants. Le max pooling conserve uniquement la valeur maximale de chaque sous-matrice tandis que l'average pooling sauvegarde la moyenne.

2.2 Transfer Learning

L'apprentissage par transfert est une méthode d'apprentissage grâce à laquelle on ré-exploite les connaissances acquises d'un modèle pré-entraîné sur une tâche afin d'en résoudre une autre. Dans le cadre des réseaux de neurones convolutifs, il consiste à réutiliser les couches d'extraction pré-entraînées d'un modèle sans ses couches de classification. Ces dernières sont alors personnalisées en fonction de la tâche que l'on souhaite réaliser.

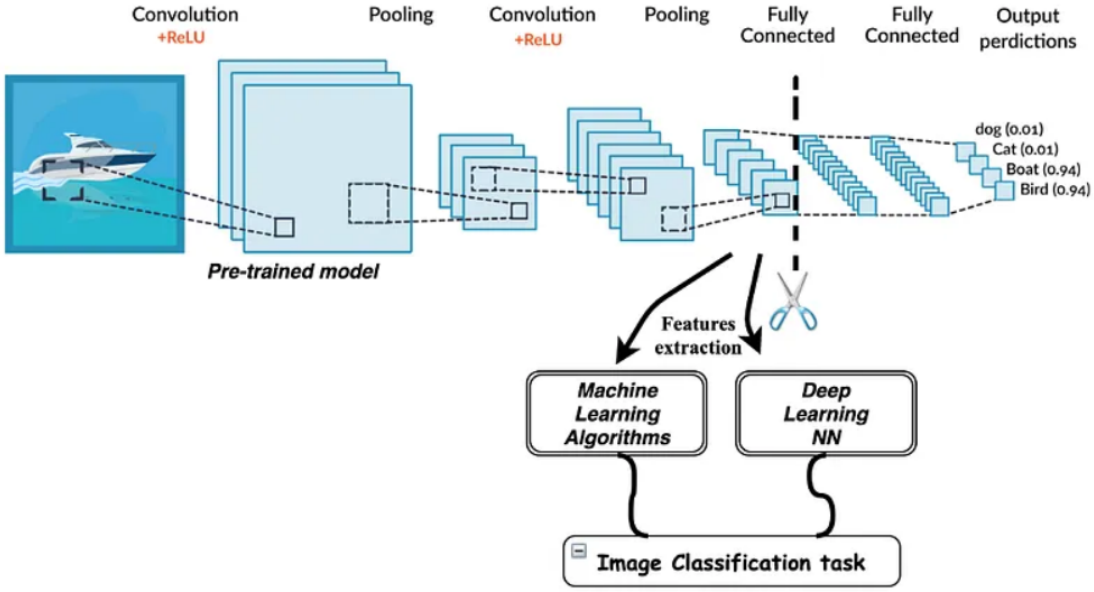


FIGURE 5 – Schéma d'un CNN dans le cas d'un transfert learning

Le transfert learning est essentiellement utilisé pour deux raisons : Premièrement, il permet de réduire considérablement les coûts de calcul ; nous n'avons pas à ré-entraîner l'intégralité du réseau de neurones, mais uniquement les couches de classification (ou tout au plus quelques couches d'extractions de caractéristiques). Deuxièmement, il permet de remédier à un faible nombre de données et d'obtenir, à partir d'un jeu de données de taille modeste, des résultats satisfaisants et exploitables.

Dans la littérature on distingue plusieurs types de transfert learning, inductif et transductif. Les différences sont difficiles à cerner et il n'est pas nécessaire de les comprendre pour pouvoir l'utiliser. Pour approfondir sur la différence entre les deux méthodes, se référer à la page datascientest sur le transfert learning.

3 Entraînement complet avec ResNet18

Dans cette section on expose les résultats issus de l'entraînement complet du réseau ResNet18. Dans un premier temps on entraîne le réseau sur les données brutes et puis sur des données ré-échantillonnées par pivotage des images. On détail la méthode plus bas.

Pour toutes les méthodes qui suivent, on sépare les données en un échantillon d'entraînement (80% de la taille de l'échantillon) et un échantillon de test identique pour chaque application de façon à pouvoir comparer les résultats. On utilise l'accuracy comme métrique (bonne classification divisée par la taille de l'échantillon). L'entraînement d'un modèle de machine learning prend du temps et il est courant de faire tourner les algorithmes plusieurs fois d'affiliée sur les données (sur plusieurs epochs) pour améliorer les résultats. Dans chacune de nos applications on fait tourner les algorithmes sur 10 epochs. On utilise la fonction de perte dite de l'entropie croisée et la méthode de descente stochastique du gradient (SGD) pour optimiser nos poids.

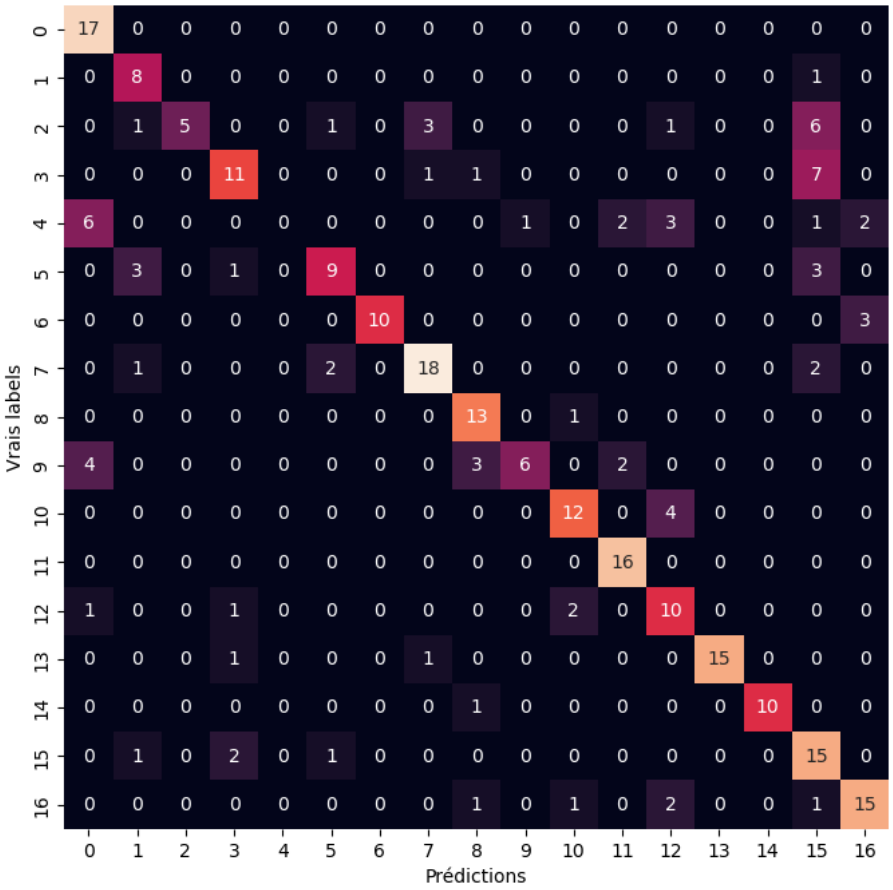
3.1 ResNet18 sur données brutes

On dispose de 80% de l'échantillon pour entraîner nos données soit 1088 images. On lance l'algorithme pour 10 epochs. L'entraînement dure 24 minutes à l'issue desquelles on obtient une accuracy de 69.85%. C'est un résultat satisfaisant au vu du nombre de données dont on disposait, mais un peu faible pour un usage industriel. La matrice de confusion du modèle est présentée dans la figure 6.

Les espèces les moins bien classées sont les Iris (9) avec 40% de bonnes classifications, les tussilages (2) avec 29% et les Crocus (4) dont aucune image n'a été reconnue. Toutes les autres ont plus de 50% de bonnes classifications. Deux colonnes de la matrice ressortent : D'abord, les Crocus (label 4) et les Iris (9) sont classifiées en tant que Jacinthe des bois (0) soit trois fleurs de couleur violette ou à tendance violette. Deuxièmement, les tussilages (2) et les primevères (3) sont vues comme des tulipes, trois fleurs jaunes. L'algorithme fait donc des confusions assez naïves de couleurs.

FIGURE 6 – Matrice de confusion ResNet18 sur données brutes

(a) Matrice de confusion



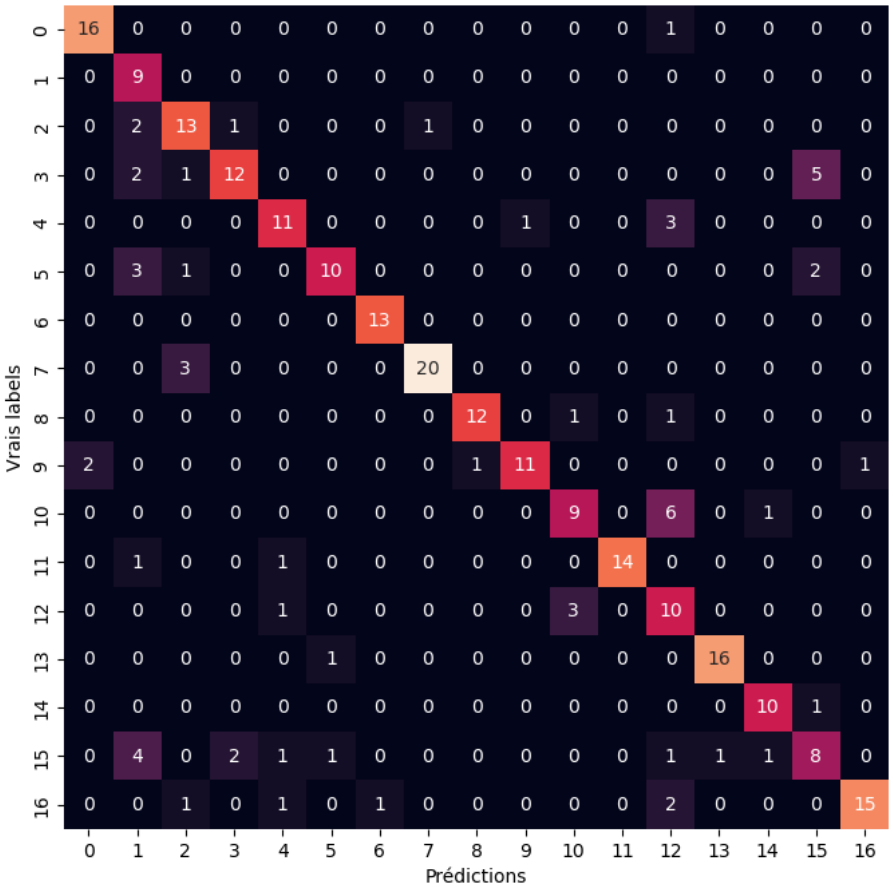
(b) Labels

3.2 ResNet18 et pivotage des images (data augmentation)

Un moyen simple pour potentiellement augmenter les performances de l'algorithme est d'augmenter la taille du jeu de données en faisant par exemple pivoter les images de l'échantillon d'entraînement de plusieurs angles différents. On fait pivoter chaque images du jeu d'entraînement de quatre angles différents (0°-90°-180°-270°) qui comporte donc désormais 4 352 images. Comme on pouvait s'y attendre, le temps de compilation augmente en proportion, 1h30 d'entraînement au bout duquel on atteint une accuracy de 76.84%. On présente la matrice de confusion dans la figure 7.

FIGURE 7 – Matrice de confusion ResNet18 sur données brutes

(a) Matrice de confusion



(b) Labels

Seules les tulipes ont moins de 50% de bonnes classifications, seulement 42%. Les valeurs de la diagonale sont à peu près partout plus élevées, spécialement pour les crocus (4) qui sont désormais majoritairement reconnues par l'algorithme (60%). On note la persistance de la confusion entre les primevères et les tulipes ainsi que l'aggravation de la confusion des perce-neige (12) vus comme du muguet (10). Avec une petite manipulation et au prix d'un temps de calcul bien plus élevé nous obtenons des résultats bien plus satisfaisants même s'ils restent insuffisants dans beaucoup d'usages industriels.

4 Application du transfert learning

Dans cette section on tente de classifier nos données à l'aide du ResNet18, préalablement entraîné sur des images plus diverses et bien plus nombreuses. Dans un premier temps on utilise les poids issus de l'entraînement sur le recueil d'images ImageNet, dans un second temps on utilise des poids issus de l'entraînement sur le recueil d'images de plantes, Pl@ntNet.

4.1 ResNet18 et ImageNet

ImageNet est une base de données d'images annotées produite par l'organisation du même nom, à destination des travaux de recherche en vision par ordinateur. Elle utilise la production participative dans son processus d'annotation. ResNet18 a été entraîné sur environ 1.2 millions d'images et de plus 1000 classes différentes. Pour appliquer le transfert learning on fige les poids préalablement enregistrés et on modifie la dernière couche du réseau en prenant soin de changer la taille de l'output pour qu'il corresponde à nos labels. Sur PyTorch le code s'écrit comme ci-dessous :

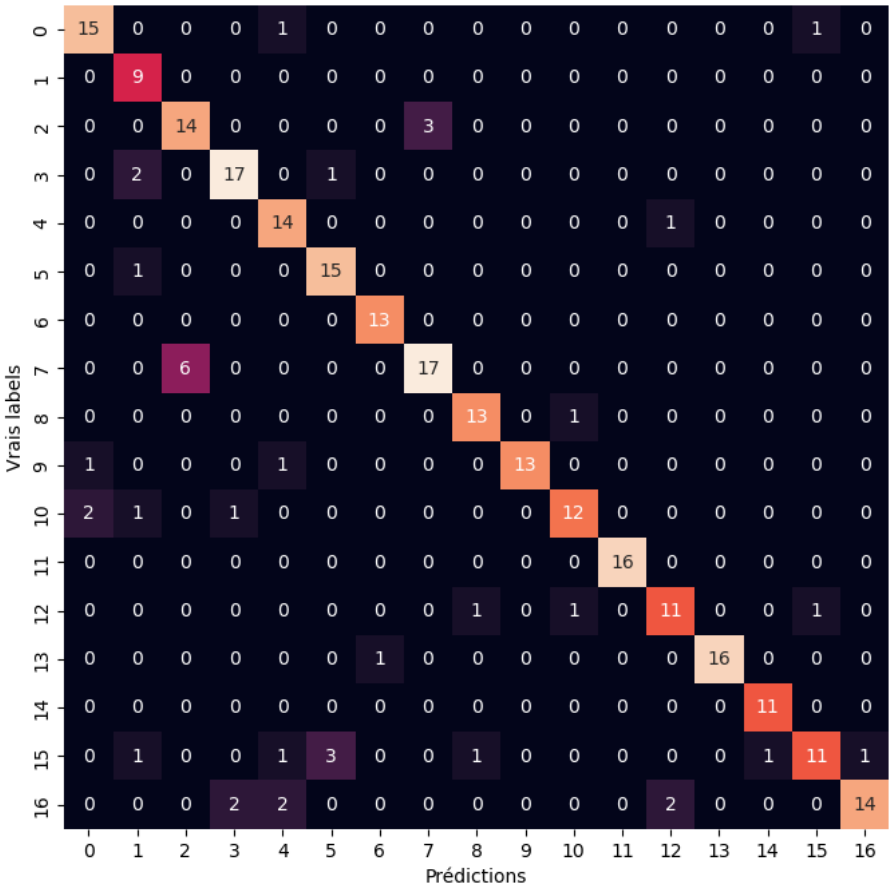
```
model = resnet18(weights=True)

    # on fige les poids
for param in model.parameters():
    param.requires_grad = False

    # récupération de la taille de l'input de la dernière couche
num_ftrs = model.fc.in_features
    # modification de la dernière couche
model.fc = nn.Linear(num_ftrs, 17)

```

Ainsi on entraîne uniquement la dernière couche du réseau, celle que nous venons de redéfinir. Comme on pouvait s'y attendre, le temps de compilation est très inférieur à celui des deux modèles précédents avec seulement 8 minutes et 30 secondes pour nos 1 088 données d'entraînement. On obtient également une accuracy bien meilleure de 84.93%. La figure 8 nous montre la matrice de confusion obtenue.

Les tulipes (15) restent l'espèce la moins bien reconnues par le classifieur, mais le taux de bonne classification de l'espèce augmente et passe de 42 à 58%.

FIGURE 8 – Matrice de confusion ResNet18 avec poids ImageNet

(a) Matrice de confusion



(b) Labels

Les images de la base de données ImageNet sont très diverses et on souhaite savoir s'il est possible d'améliorer le résultat en utilisant les poids issus d'un entraînement sur des images plus ressemblantes aux nôtre.

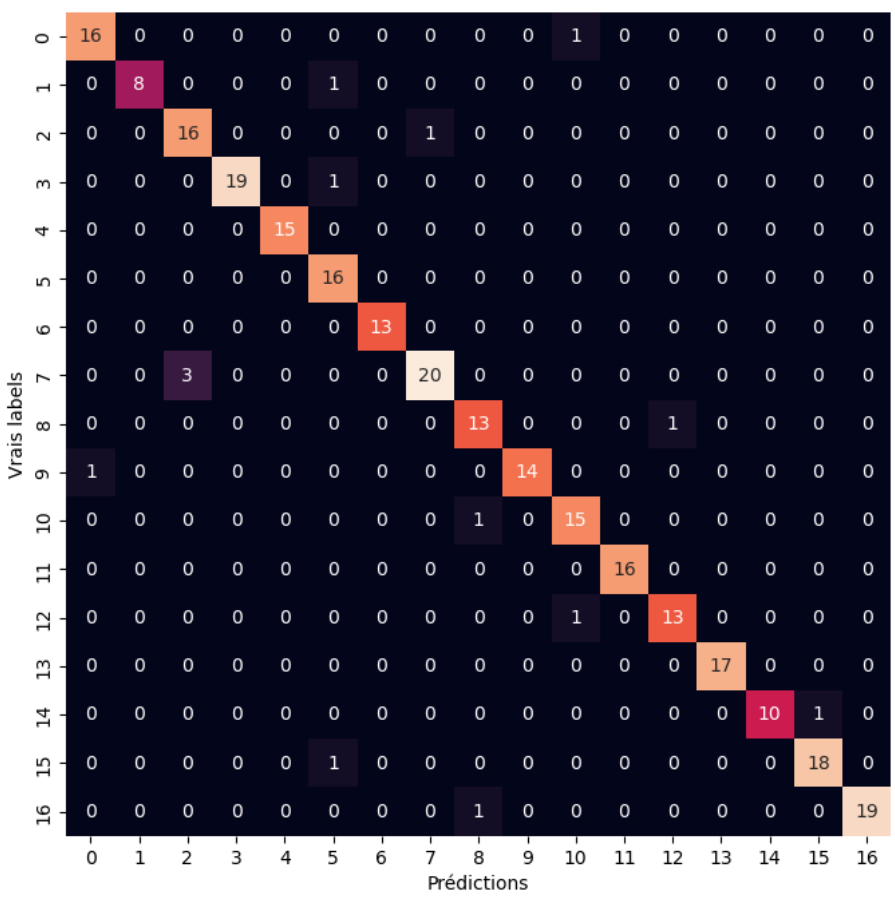
4.2 ResNet18 et pl@ntNet

Pl@ntNet est un projet informatique d'identification des plantes à partir de photographies. Il est l'œuvre de scientifiques (informaticiens et botanistes) d'un consortium regroupant plusieurs instituts de recherche français. Les tenants du projet ont construit un jeu de données regroupant 306 293 images de 1081 espèces différentes et mettent à disposition des poids issus de l'entraînement sur ce jeu de données de différents réseaux dont le ResNet18. Là encore le temps d'entraînement n'est que de 8 minutes et 30

secondes. On aboutit une accuracy de 94.85%. On présente la matrice de confusion dans la figure 9.

FIGURE 9 – Matrice de confusion ResNet18 avec poids Pl@ntNet

(a) Matrice de confusion (b) Labels



Pour chaque espèce la classification est très correcte. Le pire taux de bonne classification est celui des Pissenlits (7) avec 87%. Hormis cette espèce et les renoncules (1), les taux de bonnes classification sont supérieurs à 90% pour toutes les espèces. 3 des 23 pissenlits ont été classifiés tussilages mais à part cette erreur aucune valeur hors diagonale ne ressort vraiment.

5 Conclusion

La classification d'images a de nombreuses applications notamment en écologie. Son utilisation au moyen de réseaux de neurones convolutionnels requière l'intervention d'un grand nombre de données qui fait souvent défaut aux usagers. Cependant, le nombre grandissant de modèles pré-entraînés en libre service et les techniques de machine learning telles que le transfert learning permettent bien souvent de compenser cette insuffisance. Après avoir présenté le fonctionnement des réseaux convolutionnels et le principe du transfert learning, nous les avons appliqués afin de construire un classifieur capable de reconnaître l'espèce d'une fleur parmi 17 espèces à l'aide de 1360 images. Dans un premier temps sans utiliser le transfert learning, d'abord sur les données brutes puis sur une augmentation artificielle par pivotage des images. Dans un second temps en se servant du transfert learning, à l'aide des poids de modèle entraîné sur les images diverses du jeu ImageNet puis des images de la base Pl@ntNet plus proche de nos données. Le tableau 10 récapitule nos résultats en terme de précision des algorithmes et de temps de compilation.

Méthode	Base d'apprentissage	Base de test	Temps de calcul (jupyter)
Resnet18	80.70%	69.85%	24min
Resnet18 et data augmentation (x4)	90.90%	76.84%	1h30
Resnet18 et ImageNet	93.20%	84.93%	8min30
Resnet18 et Pl@ntNet	95.22%	94.85%	8min30

TABLE 1 – Récapitulatifs des résultats

Il apparaît clairement qu'en utilisant le tranfert learning pour classifier nos images on parvient à implémenter des algorithmes bien plus efficaces que ce soit en terme de résultat ou de temps d'apprentissage. Avant d'implémenter un algorithme de classification on devrait vérifier s'il existe un algorithme classifiant des données du même type ou proche des données dont on dispose.

Nous avons mis en évidence la supériorité du transfert learning pour une mission

bien définie. Notre méthode n'est cependant par parfait : nous avons lancés tous les algorithmes pour un nombre d'epochs fixe et arbitraire, 10. Le nombre d'epochs a une grande influence sur les résultats : s'il est trop faible on ne tire pas le maximum de l'algorithme. S'il est trop élevé l'entraînement dure inutilement longtemps et dans le pire des cas l'algorithme sur-apprend. Même si nous nous sommes préalablement assuré qu'aucun de nos modèles ne sur-apprenaient sur 10 epochs, l'idéal aurait été de relever le nombre d'epochs optimal pour chaque configuration, mais le long temps de compilation que cela aurait nécessité nous a empêché de le faire. Pour la même raison nous avons peu fait varier les paramètres des modèles ainsi que la fonction de perte et les méthodes d'optimisation (SGD, Adam, etc.) pour observer leur influence sur les résultats.

Durant ce travail nous nous sommes intéressé uniquement à l'entraînement de la dernière couche du réseau. En théorie il est également possible d'entraîner les couches plus profondes du réseau (le *fine tuning*), celles qui dans un réseau convolutionnel s'occupent de l'extraction des features. De même, il existe de nombreux autres réseaux facile d'accès pré-entraînés sur les données d'ImageNet et Pl@ntNet. Une suite à ce travail pourrait être d'exploiter ces voies et d'observer les retombées sur l'accuracy et le temps de calcul.

Références

[1] Colin J. Torney, David J. Lloyd-Jones, Mark Chevallier, David C. Moyer, Honori T. Maliti, Machoke Mwita, Edward M. Kohi, and Grant C. Hopcraft. A comparison of deep learning and citizen science techniques for counting wildlife in aerial survey images. *Methods in Ecology and Evolution*, 10(6) :779–787, 2019.

6 Annexe : Structure du réseau Resnet18

		3×3 max pool, stride 2
conv2_x	$56 \times 56 \times 64$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

FIGURE 10 – Structure du réseaux Resnet18