UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHÉMATIQUES ET INFORMATIQUE
APPLIQUÉES

PAR
SEYDOU BA

DÉTECTION DES MAUVAISES HERBES DANS LES CULTURES DE BLEUETS NAINS

Juillet 2023

Remerciements

Je tiens tout d'abord à exprimer ma profonde gratitude envers les personnes qui ont contribué de manière significative à la réalisation de ce mémoire. Le parcours de rédaction de ce document n'aurait pas été possible sans le soutien inestimable de plusieurs personnes qui ont joué un rôle crucial dans mon cheminement académique et personnel.

Tout d'abord, je voudrais exprimer ma reconnaissance sincère envers mes professeurs encadreurs, Usef Faghihi et François Meunier. Leur expertise, leur patience et leur encouragement ont été inestimables pour moi. Leur engagement à m'orienter dans mes recherches et à me guider dans ma démarche scientifique ont été essentiels à la réalisation de ce mémoire.

J'adresse également mes remerciements les plus chaleureux à ma famille, en particulier à ma mère et à mes sœurs. Leur amour inconditionnel, leur soutien indéfectible et leurs encouragements constants m'ont donné la force de poursuivre mes études avec détermination.

Je tiens à exprimer une reconnaissance spéciale à ma mère, dont le dévouement et les sacrifices ont été la clé de ma réussite. Son soutien sans faille et ses encouragements continus ont été une source d'inspiration pour moi tout au long de ce parcours académique.

Enfin, mes remerciements spéciaux à mes chers cousins, Abdoulaye Ly et Mouhamadou Ly ainsi qu'à sa femme Mariame Kane. Vous avez été comme une seconde famille pour moi, toujours présents pour me soutenir et me pousser à donner le meilleur de moi-même. Votre encouragement constant et vos précieux conseils ont été des moteurs pour persévérer dans mes études et dans la réalisation de ce mémoire. Je suis comblé d'avoir des membres de famille aussi exceptionnels que vous.

En somme, ce mémoire n'aurait pas pu voir le jour sans le soutien et l'apport de ces personnes exceptionnelles dans ma vie. Leurs encouragements, leurs conseils et leur affection ont été des moteurs essentiels dans la réussite de ce mémoire. Je leur suis infiniment reconnaissant.

Résumé

La gestion des mauvaises herbes est un pilier essentiel dans le domaine agricole, notamment en agriculture de précision. Ce type d'agriculture permet de mieux répondre aux besoins des consommateurs sur le marché en ayant recours à de nouvelles technologies pour augmenter les rendements des cultures, en permettant d'optimiser le travail des agriculteurs tout en favorisant la réduction de la consommation d'énergie, d'eau et d'intrants comme les engrais chimiques, les pesticides et herbicides. Le principal objectif de l'agriculteur est d'optimiser le désherbage de ses champs avec le recours à de nombreuses techniques d'analyse et de contrôle des mauvaises herbes basée entre autres sur l'utilisation d'images haute résolution. Ces techniques permettent de cartographier les champs en faisant une reconnaissance automatique géoréférencée des mauvaises herbes. Les images prises par des drones équipés de caméras multispectrales seront ainsi dans la présente étude, traitées de façon à en extraire les zones infestées par des mauvaises herbes. Des méthodes utilisant des approches de segmentation des images multispectrales basées sur différents indices de végétations comme le NDVI, le ExG, le ExR ou le ExGR afin de d'abord différencier le sol de la végétation et d'autre part les mauvaises herbes du bleuet nain seront aussi étudiées. Dans ce contexte, notre recherche a pour principal objectif de faciliter la détection des mauvaises herbes les plus répandues dans les champs de bleuets nains. Grâce aux expérimentations qui ont été menées dans cette recherche, nous avons aussi implémenté un algorithme de segmentation et de reconnaissance basé sur une approche neuronale, celle des réseaux de neurones convolutifs comme ResNet et UNET. Ces réseaux convolutifs ont été retenus dans la présente recherche puisqu'ils sont capables de détecter avec une bonne précision des mauvaises herbes comme la Comptonie voyageuse (fougère douce) et le Kalmia à feuilles étroites (crevard de mouton), dans les champs de bleuets nains.

Mots-clés : agriculture de précision, mauvaises herbes, bleuets nains, segmentation d'images, indices de végétation, Réseaux de neurones convolutifs.

Abstract

Weed management is an essential pillar in the agricultural sector, particularly in precision agriculture. This type of farming allows for better responsiveness to consumer demands in the market by leveraging new technologies to increase crop yields while optimizing the work of farmers and promoting reduced energy, water, and input consumption such as chemical fertilizers, pesticides, and herbicides. The primary goal of the farmer is to optimize weed control in their fields by employing various techniques for weed analysis and control, including the use of high-resolution imagery.

These techniques enable field mapping through georeferenced automatic recognition of weeds. Images captured by drones equipped with multispectral cameras are processed in this study to extract weed-infested areas. Methods utilizing multispectral image segmentation approaches based on various vegetation indices like NDVI, ExG, ExR, or ExGR are also investigated. These methods are employed to differentiate between soil and vegetation, as well as between weeds and lowbush blueberries.

In this context, our research primarily aims to facilitate the detection of the most common weeds in lowbush blueberry fields. Through the experiments conducted in this research, we have also implemented a segmentation and recognition algorithm based on a neural approach, specifically Convolutional Neural Networks (CNNs) like ResNet and UNET. CNNs were chosen for this research because they can accurately detect weeds such as sweet fern and sheep laurel in lowbush blueberry fields.

Keywords : precision agriculture, weeds, lowbush blueberries, image segmentation, vegetation indices, Convolutional Neural Networks (CNNs).

Table des matières

Table des figures

Chapitre 1 Introduction

Le traitement électronique des données joue un rôle primordial ayant un impact dans plusieurs filières de la recherche scientifique ou le monde professionnel notamment dans le domaine agricole plus précisément avec l'agriculture de précision devenue un pilier de la société d'aujourd'hui. Le désir d'une forte production pousse les agriculteurs à recourir à de nouvelles technologies d'optimisation permettant ainsi une importante rentabilité répondant ainsi à la demande croissante des consommateurs, mais pouvant avoir un impact néfaste à l'environnement. La quête de productivité accrue est confrontée à de nombreux facteurs, dont la gestion des mauvaises herbes. La connaissance des interactions entre le sol, les cultures et les mauvaises herbes constitue un atout important chez les agriculteurs et celle-ci permet à ces derniers de se tourner vers l'utilisation des méthodes comme l'usage des pesticides, d'herbicides et le désherbage manuel dont l'utilisation affecte en aucun cas l'écosystème. Par contre, pour une gestion des cultures conventionnelles, les entreprises productrices à grandes échelles optent vers l'utilisation d'herbicides leur permettant ainsi de produire avec un rendement important tout en évitant un niveau de coût de production trop élevée. La localisation préalable géoréférencée des zones infestées par les mauvaises herbes d'un champ est un enjeu important dans le processus de gestion moins coûteux de ces mauvaises herbes. Ainsi cela permet aux agriculteurs de mieux cibler avant toutes interventions phytosanitaires les zones infestées d'un champ. Une telle détection préalable des zones infestées permettrait entre autres de réduire la quantité d'herbicide utilisée en culture conventionnelle puisque les arrosages pourraient être limitées à ces zones préalablement localisées. D'autre part une localisation préalable des mauvaise permettrait aux producteurs biologiques de mieux orienter les opérations de sarclage. La détection préalable des mauvaises herbes devient incontournable dans les cultures de bleuets nains biologiques étant donné que les opérations de sarclage sont onéreuses à cause des besoins importants en main-d'oeuvre et machinerie spécialisée.

Avec l'avancée considérable des technologies en matière de traitement et d'analyse d'images,

de l'optimisation des approches de discrimination des mauvaises herbes et des cultures, il devient possible de faire une classification ciblée de ces mauvaises herbes. L'ajout d'un système de vision artificielle automatisé pouvant faciliter la détection préalable des mauvaises herbes permettrait alors d'optimiser le processus d'éradication des mauvaises herbes.

1.1 Objectifs du mémoire

Le principal objectif de ce mémoire consiste à développer une méthode de détection automatique de mauvaises herbes dans les champs de bleuets nains tant conventionnels que biologiques. La méthode automatique implémentée fait d'abord l'acquisition d'images obtenus par un drone, effectue ensuite la détection des zones infestées par les mauvaises herbes et génère ultimement une carte géoréférencée de ces zones infestées. Ainsi, avec ces cartes géoréférencées l'agriculteur pourrait orienter plus efficacement les opérations phytosanitaires. Cette approche originale aura des répercussions positives tant sur le plan environnemental que sur le plan économique visant une réduction de l'effort du désherbage manuel et de l'utilisation des herbicides. Par conséquent, cela permettra de moins nuire le sol avec les produits chimiques tels que les herbicides en limitant leur application qu'aux zones infestées réduisant ainsi les efforts et les dépenses affectés.

La présente mémoire est structurée de la façon suivante :
Dans le chapitre 2, nous allons présenter les articles desquels est inspirée la présente recherche similaires. Cette littérature portant sur les approches de détection et de classification automatiques des mauvaises herbes. De plus, et certaines librairies utilisées dans le cadre de cette recherche sont aussi répertoriées dans ce chapitre. Ensuite, au chapitre 3, la méthodologie utilisée pour résoudre la problématique de cette recherche est exposée. La méthodologie suggérée est subdivisées en deux parties. La première partie explique les transformations diverses qui sont effectuées sur les images pour permettre leur utilisation subséquente par les algorithmes de classification automatique. La deuxième partie montre les techniques utilisées pour permettre la différenciation des mauvaises herbes et des plants de bleuets nains.

Finalement, au chapitre 4, les résultats des expérimentations effectuées dans le but de tester

les approches de classification automatique des mauvaises herbes implémentées sont présentés puis en guise de conclusion au chapitre 5 nous allons discuter des différentes perspectives qui pourraient permettre de faire évoluer la recherche.

1.2 Conclusion

En résumé, les nouvelles technologies d'optimisation en agriculture de précision permettent aux cultivateurs d'optimiser le rendement de leur culture. Cependant, une quête de grand rendement ne devrait pas signifier nuire à l'environnement, mais plutôt tenter de trouver des façons innovantes d'optimiser entre autres la gestion des opérations phytosanitaires tout en réduisant les coûts et les impacts sur l'écosystème. Ainsi, proposer une solution permettant de produire une carte géoréférencée des zones infestées par les mauvaises herbes favoriserait ainsi les objectifs de rentabilité et de respect de l'environnement à la fois. Ce mémoire permet de présenter une telle solution aux problèmes de localisation préalable des mauvaises herbes. Ce mémoire décrit cette solution constituée d'étapes qui doivent être mises en oeuvre pour permettre la détection automatique de ces mauvaises herbes.

Dans le chapitre suivant, une revue de l'état de l'art portant sur les recherches similaires sera étudiée afin d'y trouver des techniques et des solutions pour arriver à résoudre la présente problématique de cette recherche. Par conséquent, des revues scientifiques, des librairies de traitement d'images seront alors présentées.

Chapitre 2 Revue de littérature

2.1 Introduction

Dans ce chapitre, nous allons introduire les différentes techniques utilisées en traitement d'images et plus particulièrement la détection et la classification dans le domaine de l'agriculture de précision avec l'utilisation d'algorithmes spécialisés. En premier lieu, nous allons d'abord porter notre attention sur la culture du bleuet nain et aussi de la gestion des mauvaises herbes dans ce genre de culture. Ensuite, nous exposerons les différents types de mauvaises herbes, plus particulièrement le Kalmia à feuilles étroites et la Comptonie voyageuse. Ensuite, nous ferons référence aux recherches portant sur la détection et la discrimination des mauvaises herbes dans différentes cultures tout en abordant les diverses approches utilisées dans le domaine du traitement d'images et classification automatiques, appliquées en particulier au domaine de l'agriculture de précision.

2.2 Culture du bleuet nain et la gestion des mauvaises herbes

Le bleuet est une plante vivace indigène dont la culture s'effectue sur des sols acides. L'article de Gagnon (2009) porte sur la production du bleuet nain au Saguenay-Lac Saint-Jean. Cet article vise à donner des solutions pratiques aux agriculteurs pour lutter contre les mauvaises herbes selon le type de culture du bleuet nain. Il existe deux types de culture : conventionnelle et biologique suivant des cycles bisannuels et dans certains cas trisannuels. Durant chaque cycle, les agriculteurs tentent de maximiser la rentabilité de leurs cultures, mais la gestion des mauvaises herbes reste une grande problématique. La croissance du bleuet fait face à la concurrence de plusieurs plantes qualifiées de mauvaises herbes dont deux principales : le Kalmia à feuilles étroites et la Comptonie voyageuse. Ces mauvaises herbes aiment les sols acides à l'instar du bleuet, tout en masquant les bleuets du soleil ainsi réduisant la production de fruits, diminuant la qualité des produits et interférant lors de la cueillette. La variabilité de la croissance et de l'occupation du sol de la culture du bleuet nain ne permet

pas de facilement lutter contre les plantes nuisibles. Afin de mieux optimiser le contrôle des mauvaises herbes, il existe plusieurs herbicides, dont l'hexazinone et le glyphosate pour les cultures conventionnelles. Pour mieux contrôler les plants nuisibles, le sarclage des champs peut aussi être préconisé conjointement aux herbicides. L'utilisation d'herbicides et du sarclage sont des activités très coûteuses pour les deux types de cultures : conventionnelle et biologique. Pour réduire ces coûts exorbitants, la mise en place d'un système automatisé pour la géolocalisation préalable des mauvaises herbes est une solution envisageable et meilleure pour n'importe quels types de cultures. En localisant plus précisément les zones infestées, les quantités d'herbicides ou d'opérations de désherbages manuelles peuvent ainsi être réduites puisque les surfaces traitées sont strictement celles infestées.

2.3 Les mauvaises herbes

2.3.1 Le kalmia à feuille étroite

FIGURE 1 – Le Kalmia à feuilles étroites

Appartenant à la famille des Ericaceae, le Kalmia à feuilles étroites est connu avec plusieurs surnoms le Crevard de mouton, Laurel, sheep laurel ou lambkill (Voir Fig. 1). Ces derniers prennent leur origine sur le fait de leur toxicité envers les espèces animales dont le mouton. Cette plante fait partie des sept espèces de Kalmia que l'on retrouve en Amérique du Nord, celle-ci étant plutôt située au nord-est du continent. Le kalmia est décrit par Gagnon (2009) comme un arbuste pouvant atteindre 80 cm de hauteur et ses feuilles, persistantes même en hiver, ont une forme elliptique d'une longueur de 2 à 9 cm.

2.3.2 La comptonie voyageuse

La comptonie voyageuse ou comptonia peregrina, aussi communément appelée fougère douce, est un arbuste de petite taille, indigène en Amérique du Nord (Voir Fig. 2). La tige a un port dressé et mesure de 0,5 à 1 m de hauteur. Sa racine est composée de rhizomes robustes, grisâtres ou rougeâtres produisant de nombreuses pousses végétatives. Expliquée par Gagnon (2010a), cette plante appartenant à la famille des myricacées est une unique espèce de comptonie d'Amérique du Nord.

FIGURE 2 – La comptonie voyageuse

2.4 Prétraitement des images

Le prétraitement des images est un processus très important dans le traitement et l'analyse d'image. Après la prise des images, celles-ci doivent subir un prétraitement avant leur utilisation. Ce processus diffère selon leur utilisation, le processus d'acquisition des images, le type d'améliorations à apporter à ces images et les résultats souhaités.

2.4.1 Approches basées sur le spectre visible (couleur)

La transformation d'image basée sur le spectre visible (couleur) consiste à faire des opérations sur les composants de couleurs dans le visible. Ainsi la méthode de prétraitement la plus utilisée est la transformation d'une image du modèle RVB (Rouge, Vert et Bleue) en une image en nuance de gris (image en tons de gris) afin ensuite de réduire le bruit de ces images, améliorer le contraste des images et pouvoir appliquer divers filtres. Pour appliquer

cette première méthode de transformation, une formule simple est appliquée consistant à faire la moyenne des trois couleurs (rouge,vert,bleue) de chaque pixel de l'image comme décrite ci-dessous :

$$Image = \frac{R + G + B}{3}$$

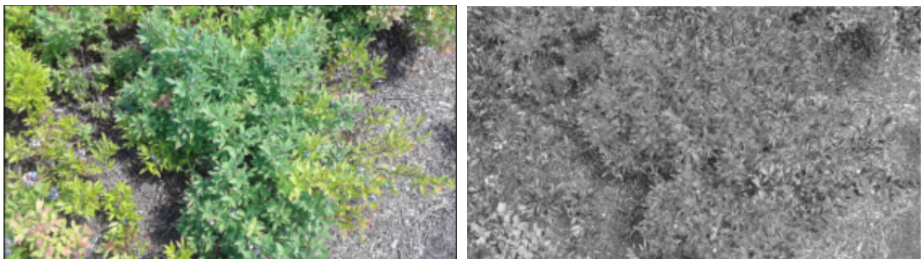

FIGURE 3 – Transformation RGB (image gauche) à niveau de gris (image droite)

Mais cette formule ne donne pas un bon contraste facilitant la discrimination de la végétation par rapport au sol (voir figure 3). Afin d'avoir un meilleur contraste dans l'image, les recommandations de l'Union International des Télécommunications (UIT-R, 2011) suggèrent une normalisation des trois couleurs du modèle RGB comme décrite par la formule suivante (voir une image normalisée fig. 4 image à droite) :

$$Image = 0.299 * R + 0.587 * B + 0.114 * G$$

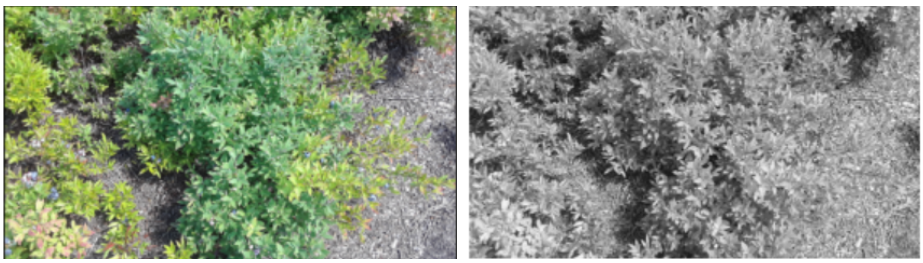

FIGURE 4 – Transformation RGB (image gauche) à niveau de gris (image droite) suivant les recommandations UIT

Dans ming Li et al. (2009), la transformation du modèle RVB (Rouge, Vert et Bleue) en modèle TSL (Teinte, Saturation et Luminosité) est clairement expliquée où seulement la composante teinte moins sensible aux variations d'illumination est utilisée dans la segmentation.

Dans Hamuda et al. (2016), le raisonnement derrière le calcul d'une dizaine d'indices de végétation est brièvement expliqué où chacun des indices donne un résultat entre 0 et 1, où les valeurs calculées proche de 0 correspondent au sol et celles proches de 1 à la végétation. Ces opérations sont visualisées sur une image en niveau gris comme présentée sur la figure 5 où l'indice ExG (excess of Green) permet de faire ressortir les surfaces de végétation.

2.5 Les indices de végétation

Un indice de végétation est une valeur unique calculée à partir d'une combinaison de bandes spectrales. Cet indice est utilisé pour rehausser la présence d'éléments végétaux verts et ainsi aider à les distinguer des autres objets présents dans l'image. On note une liste exhaustive d'indices de végétation mais nous allons en citer quelques-uns plus populaires et utilisés.

2.5.1 L'indice d'excès de vert ExG

L'indice de végétation ExG (EXcesss of Green) est un indice intuitif dans sa définition, qui est calculé par la manipulation du modèle colorimétrique RVB, cherchant ainsi à rehausser la composante verte de l'image d'un facteur deux. Cet indice produit une image de niveaux de gris où les zones de plantes vertes sont parfaitement discriminées (Meyer et al. (1999)) (Voir fig. 5). Cet indice se calcule par la formule suivante :

$$ExG = 2g - r - b$$

où r, g et b sont respectivement la composante vert, rouge et bleu (les composants chromatiques) d'un pixel de l'image :

$$r = \frac{R^*}{R^* + G^* + B^*}, \quad g = \frac{G^*}{R^* + G^* + B^*} \quad , \quad b = \frac{B^*}{R^* + G^* + B^*}$$

R^*, G^* et B^* sont les valeurs normalisées du modèle colorimétrique RGB dans un intervalle de 0 à 1 :

$$R^* = \frac{R}{R_m}, \quad G^* = \frac{G}{G_m} \quad , \quad B^* = \frac{B}{B_m}$$

R, G et B sont les valeurs de chaque composante rouge, vert et bleu respectivement d'un pixel dans une image et $R_m = G_m = B_m = 255$ pour une image de 24 bits comme celle de la figure 5 (Image gauche).

FIGURE 5 – Exemple d'application de l'indice ExG (Image droite) sur une image RGB (Image gauche)

2.5.2 L'indice d'excès de rouge ExR

Partant du même principe que l'indice d'excès de vert (ExG), mais s'inspirant du fait que l'œil humain perçoit plus la couleur rouge que le vert ou le bleu (Meyer et al. (1999)). L'indice ExR amplifie la composante rouge dans une proportion de 1.3, et ce selon le ratio

de bâtonnets rouges et verts de l'oeil humain. Cet indice est très utile pendant la saison d'automne ou un grand nombre de végétaux prennent la couleur rouge et aussi pour séparer la végétation (hors de la saison d'automne) du sol si ce dernier possède une couleur rouge. La formule de calcul de l'indice ExR est :

$$ExR = 1.3R - G$$

FIGURE 6 – Exemple d'application de l'indice ExR (Image droite) sur une image RGB (Image gauche)

2.5.3 L'indice de végétation par différence normalisée NDVI

L'indice de végétation par différence normalisée ou NDVI est déduit à partir des composants rouge et proche infrarouge. Le NDVI permet de mettre en évidence la différence entre la bande

visible du rouge et celle du proche infrarouge. Cet indice exploite une des conséquences du phénomène de photosynthèse puisque les feuilles des plantes absorbent donc réfléchissent peu le rayonnement rouge et fortement le rayonnement proche infrarouge. Cet indice est ainsi sensible à la vigueur et la densité du couvert de végétation. Les valeurs du NDVI sont comprises entre -1 et 1, dont les valeurs négatives correspondent aux surfaces autres que les couverts végétaux où la réflectance dans le proche infrarouge est très supérieure à celle du rouge. Dans le cas des sols nus, le NDVI présente des valeurs proches de 0 car les réflectances sont du même ordre de grandeur dans le rouge et le proche infrarouge. Le NDVI est défini comme suit :

$$NDVI = \frac{PIR - R}{PIR + R}$$

où les termes PIR et R sont les composants du proche infrarouge et du rouge de chaque pixel dans une image.

2.6 Étapes du processus de classification

2.6.1 Segmentation

2.6.1.1 Les méthodes de seuillage

Sachant que certains indices de végétation permettent visuellement de facilement différencier la végétation par rapport au sol. Des techniques de segmentation automatique peuvent ainsi être appliquées sur les images d'indices de végétation pour permettre justement l'automatisation de la détection et la discrimination de la végétation par rapport au sol. À priori le but d'une méthode de seuillage est de séparer deux classes d'objets grâce à un seuil prédéterminé ou à l'aide d'une méthode automatisant le processus de sélection optimal du seuil. Cependant, il existe plusieurs techniques de segmentation utilisées dans la littérature permettant une séparation plus précise entre le sol et la végétation. Nous allons exposer quelques-unes de ces méthodes de seuillage dans les sections suivantes.

2.6.1.1.2 La méthode d'Otsu

La méthode Otsu comme son nom l'indique a été développée par Otsu (1979). Elle est une technique de seuillage d'image qui permet de séparer une image en deux classes (avant-plan et arrière-plan) en fonction de l'intensité des pixels. Le seuillage est un processus de conversion d'une image en niveaux de gris en une image binaire, où les pixels avec des intensités supérieures à un certain seuil sont affectés à une classe (premier plan) et les pixels avec des intensités inférieures au seuil sont affectés à une autre classe (arrière-plan). La méthode Otsu détermine la valeur seuil optimale en maximisant la variance entre les deux classes. Ceci est réalisé en minimisant la somme de la variance au carré dans chaque classe, connue sous le nom de variance intra-classe. La valeur de seuil optimale est ensuite utilisée pour créer l'image binaire. Elle est aussi une technique simple et efficace largement utilisée dans les applications de traitement d'images et de vision par ordinateur. La méthode de Otsu, est notamment utile pour séparer des images avec des distributions d'intensité bimodales, où les pixels peuvent être facilement séparés en deux classes distinctes en fonction de leurs valeurs d'intensité.

Soient C1 et C2 deux classes de pixels. C1 est définie comme étant la classe 1 composée des pixels ayant une valeur comprise entre 0 et la valeur seuil k avec k<255, ce groupe de pixels représente l'arrière-plan. C2 est définie comme la classe 2 composée des pixels compris entre k et 255 (inclus), ce groupe de pixels représente quant à lui les objets recherchés. Les étapes de l'implémentation de l'algorithme de Otsu sont les suivantes :

1. Traiter l'image d'entrée
2. Obtenir l'histogramme de l'image (distribution des pixels)
3. Normaliser l'histogramme obtenu à l'étape 2 afin d'avoir un histogramme dont les valeurs sont comprises entre 0 et 1, valeurs qui représentent la probabilité d'occurence des intensités de chaque pixel
4. Estimer la probabilité qu'un pixel de l'image appartienne à C1 en faisant la somme de toutes les colonnes de l'histogramme comprise entre 0 et k.
5. Estimer la probabilité qu'un pixel de l'image appartienne à C2 en faisant cette fois-ci 1- proba(C1) calculée dans l'étape précédente.

6. Estimer la variance interclasse entre C1 ET C2 grâce à la formule suivante :

$$var_{C1interC2} = \frac{(Moy * Proba_{C1}(k) - Moy_{C1}(k)))^2}{Proba_{C1}(k) * Proba_{C2}(k)}$$

   Avec Moy étant la moyenne de l'image d'entrée, Proba$_{C1}(k)$ étant la probabilité qu'un pixel appartienne à C1, Moy(k) la moyenne des pixels de la classe C1.

7. Les étapes 3, 4, 5, 6 sont répétées pour toutes les valeurs de k possible.
8. Choisir k avec une variance interclasse maximale (MAX(varC1interC2))

2.6.1.1.3 La méthode Isodata

Isodata est un algorithme de classification non supervisé nommé "Iterative Self-Organizing Data Analysis Technics" (Merzougui et al. (2016)). Cet algorithme est une version améliorée de l'algorithme de K-means. Cette amélioration réside dans sa capacité à adapter dynamiquement le nombre de clusters, à gérer des clusters de tailles inégales et à prendre en compte les covariances entre les caractéristiques des données et cela en fait un choix plus flexible et puissant pour la classification non supervisée dans de nombreuses situations . Au cours du processus itératif, la méthode autorise le fractionnement, la fusion entre les nuages de points proches et la suppression de nuages de petites tailles en fonction des paramètres de seuil d'entrée. Cependant, tous les pixels sont classés dans la classe la plus proche sauf si l'utilisateur ne spécifie pas un écart type ou un seuil de distance auxquels certains pixels pourraient ne pas être classés s'ils ne répondent pas aux critères de sélection. Ce processus se poursuit jusqu'à ce que le nombre de pixels de chaque classe ne varie pas plus que le seuil de changement du nombre de pixels sélectionné ou que le nombre maximal d'itérations soit atteint. L'algorithme de la méthode Isodata est le suivant :

1. **Fixer** les paramètres (nombre de classes, pourcentage de seuil, nombre d'itérations...)
2. **Initialiser** les centroïdes à des valeurs aléatoires dans l'espace d'observations
3. **Affecter** les observations aux classes des centroïdes qui leur sont les plus proches
4. **Tester** les conditions pour fusionner ou diviser des classes

5. **Actualiser** les centroïdes de chaque classe

6. **Stopper** l'algorithme lorsque l'un des critères d'arrêt est satisfait, **sinon aller à 3**.

2.6.1.1.4 La méthode de seuillage de Li

Li and Lee (1993) ont introduit une méthode de seuillage d'images fondée sur l'algorithme de seuil d'entropie croisée minimale. Le problème de sélection du seuil est résolu en minimisant l'entropie croisée entre l'image originale et sa version segmentée. L'entropie croisée est calculée pour chaque pixel correspondant des deux images à l'aide d'un algorithme informatique basé sur leur histogramme respectif. Sans faire d'hypothèses à priori sur la distribution de la population, cette méthode fournit une estimation non biaisée d'une version binarisée de l'image au sens théorique de l'information. La méthode de Li a établi que le seuil optimal est calculé de façon à minimiser $\eta(t) = \eta_1(t) + \eta_2(t)$ où :

$$\eta_1(t) = \sum_{i=1}^{t-1} i * h_i * log\left(\frac{i}{\mu_1(t)}\right) \quad et \quad \eta_2(t) = \sum_{i=t}^{255} i * h_i * log\left(\frac{i}{\mu_2(t)}\right)$$

$$\mu_1(t) = \frac{\sum_{i=1}^{t-1} i * h_i}{\sum_{i=1}^{t-1} h_i} \quad et \quad \mu_2(t) = \frac{\sum_{i=t}^{255} i * h_i}{\sum_{i=t}^{255} h_i}$$

$\mu_1(t)$ et $\mu_2(t)$ sont les moyennes des classes 1 et 2 en fonction du niveau de gris t et h_i le nombre de pixels de niveau de gris i.

2.6.1.1.5 La méthode de seuillage de Yen

Yen et al. (1995) décrivent une technique de seuillage automatique à plusieurs niveaux consistant à trouver automatiquement le nombre de classes d'objets tout en minimisant une fonction de coût dont la valeur repose sur deux critères. Le premier, Dis(k), est la différence entre l'image originale et l'image segmentée et le second le nombre de bits composant la valeur k du nombre de classes. Ces critères sont exprimés selon l'équation suivante :

$$C(k) = \rho\left(Dis\left(k\right)\right)^{1/2} + \left(log_2(k)\right)^2$$

où ρ est une constante de pondération positive, k le nombre de classes et Dis(k) est le coefficient de dissimilarité et log2 (k) le nombre de bits pour représenter k. On note que quand le nombre de classes augmente, le facteur Dis(k), réduit la dissimilarité et augmente le nombre de bits représentant k donc dans cette équation, on recherche un compromis entre les deux facteurs qui la compose.

Les approches de seuillage évoquées précédemment sont généralement utilisées afin de segmenter une image en 2 régions distinctes, cependant il arrive souvent que plus de 2 régions se trouvent dans une image, donc dans ce cas une approche multiseuillage devrait être envisagée.

2.6.1.1.6 La méthode de seuillage multi Otsu

La méthode de seuillage Multi Otsu est une variante de la méthode d'Otsu, initialement proposée par Otsu (1979). L'idée d'utiliser plusieurs valeurs de seuil pour segmenter une image en plusieurs classes ou régions a ensuite été introduite comme une variante de la méthode d'Otsu par Liao et al. (2001). Cela implique de diviser l'image en plusieurs niveaux ou régions en fonction des valeurs d'intensité des pixels, en utilisant plusieurs valeurs de seuil plutôt qu'une seule valeur. Pour l'implémenter, l'image originale est d'abord transformée en un histogramme, qui représente la distribution des intensités de pixels dans l'image. L'histogramme est ensuite divisé en un nombre spécifié de classes ou de bacs, et les valeurs de seuil optimales qui sont trouvées minimise la somme des variances intra-classes résultantes. Ce processus est répété pour chacune des classes jusqu'à ce que le nombre de niveaux ou de régions souhaité soit obtenu. Le seuillage multi-Otsu peut être utile pour les tâches de segmentation et d'analyse d'images qui nécessitent la séparation de plusieurs objets ou régions avec des valeurs d'intensité différente telle que la reconnaissance d'objets et l'analyse d'images médicales. Il peut également être utilisé pour améliorer le contraste des images en séparant les pixels en plusieurs niveaux avec des valeurs d'intensité plus distinctives.

2.6.1.2 Clustering

Le clustering est une technique d'exploration de données et d'apprentissage automatique qui est utilisée pour regrouper des points de données (observations) en clusters en fonction de la similarité de leurs caractéristiques propres. L'objectif du clustering est de partitionner les données en clusters de sorte que les points d'un même cluster soient plus similaires entre eux qu'avec les points d'autres clusters. Cette technique est utile pour un large éventail d'applications, notamment l'analyse de données, la reconnaissance de formes et la segmentation d'images. Il s'agit également d'une étape fondamentale dans de nombreux pipelines d'apprentissage automatique, car elle peut aider à identifier des modèles dans les données et faciliter une analyse plus approfondie. Il existe plusieurs algorithmes de clustering différents dont nous allons exposer les plus utilisés.

2.6.1.2.1 DBSCAN Clustering

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) est un algorithme de clustering proposé par Ester et al. (1996) utilisé pour partitionner un ensemble de points de données en clusters en fonction de leur densité. Il est particulièrement utile pour trouver des clusters de formes arbitraires et identifier les valeurs aberrantes dans les données. L'algorithme DBSCAN a deux paramètres principaux : Eps et MinPts. Eps est la distance maximale entre un point de données et son voisin le plus proche pour être considéré comme faisant partie du même cluster. MinPts est le nombre minimum de points de données requis pour former un cluster. L'algorithme fonctionne en commençant avec un point de données et en développant un cluster autour de celui-ci si la densité des points de données dans le cluster répond aux critères spécifiés par Eps et MinPts. Si un point de données ne répond pas à ces critères, il est considéré comme une valeur aberrante et n'est inclus dans aucun cluster. L'un des avantages de l'algorithme DBSCAN est qu'il n'oblige pas l'utilisateur à spécifier à l'avance le nombre de clusters. Il est également résistant au bruit et peut gérer efficacement de grands ensembles de données. Cependant, il peut être sensible au choix de Eps et MinPts et peut ne pas fonctionner correctement sur des données avec des densités non uniformes. Pour effectuer le clustering DBSCAN, les étapes suivantes sont généralement suivies :

1. Choisir une valeur pour Eps, qui correspond à la distance maximale entre un point de données et son voisin le plus proche pour être considéré comme faisant partie du même cluster.
2. Choisir une valeur pour MinPts, qui correspond au nombre minimum de points de données requis pour former un cluster.
3. Pour chaque point de données, calculer la distance à ses voisins les plus proches.
4. Si un point de données a au moins un nombre MinPts de voisins à une distance Eps, il est considéré comme un point central.
5. Développer les clusters à partir de chaque point central en ajoutant tous les points à une distance Eps du point central, jusqu'à ce qu'aucun autre point ne puisse être ajouté au cluster.
6. Attribuer les points restants à un groupe de bruit (valeurs aberrantes).

2.6.1.2.2 K-moyennes

K-moyenne est un algorithme de clustering proposé à l'origine par Stuart Lloyd (1957), mais il a ensuite été popularisé par Edward W. Forgy (1965) puis par MacQueen (1967). Il est utilisé pour partitionner un ensemble de points de données en K clusters distincts. L'objectif de l'algorithme K-means est de minimiser la somme des distances au carré entre chaque point de données et la moyenne de son cluster attribué. Elle est une méthode de clustering populaire et largement utilisée en raison de sa simplicité et de son efficacité. Il est particulièrement utile pour les grands ensembles de données et peut être appliqué à divers types de données, y compris les données numériques, catégorielles et textuelles. Cependant, il présente certaines limites, notamment la nécessité de spécifier à l'avance le nombre de clusters et la sensibilité aux affectations initiales du centroïde. L'algorithme K-moyennes comprend les étapes suivantes :

1. **Initialiser** les K centroïdes, qui représentent les clusters initiaux.
2. **Attribuer** chaque point de données au centroïde le plus proche, formant les K clusters.

3. **Calculer** la moyenne de chaque cluster et mettre à jour les K centroïdes avec les nouvelles moyennes.
4. **Réaffecter** chaque point de données au centroïde le plus proche.
5. **Répéter** les étapes 3 et 4 jusqu'à ce que les centres de gravité convergent ou jusqu'à ce qu'un nombre prédéterminé d'itérations ait été atteint.

2.6.1.3 Approche de segmentation basée sur les CNN

Pour l'approche de segmentation basée sur les réseaux de neurones convolutifs (CNN), nous allons exposer l'architecture UNET qui après un entraînement peut segmenter une image en région déjà étiquetée donc classée.

UNET est une architecture de réseau neuronal convolutionnel (CNN) qui a été développée par Ronneberger et al. (2015) pour la segmentation d'images biomédicales à l'Université de Fribourg, en Allemagne pour les tâches de segmentation d'images. C'est aujourd'hui l'une des approches les plus couramment utilisées dans toute tâche de segmentation sémantique. Il s'agit d'un réseau neuronal entièrement convolutif composé d'un encodeur et d'un décodeur conçu pour apprendre à partir de moins d'échantillons d'entraînement.

L'architecture UNET (Voir fig. 7) est un réseau encodeur-décodeur en forme de U, qui se compose de quatre blocs de codage et quatre blocs de décodage connectés via un pont. Le réseau d'encodeurs (chemin de contraction) permet d'abord de réduire les dimensions spatiales de l'image originale de moitié tout en générant en sortie un vecteurs de caractéristiques représentatives des objets dans l'image originale. De même, le réseau de décodeurs agit en sens inverse en augmentant du double les dimensions spatiales du vecteur en entrées du réseau de décodeurs permettant de générer en sortie une version segmentée et étiquetée de l'image originale. **Le réseau d'encodeurs** agit donc comme un extracteur de caractéristiques et apprend une représentation abstraite de l'image d'entrée à travers une séquence des blocs d'encodeurs. Chaque bloc codeur se compose de deux convolutions 3x3, où chaque convolution est suivie d'une fonction d'activation ReLU (Rectified Linear Unit). La fonction d'activation ReLU introduit la non-linéarité dans le réseau, ce qui contribue à une meilleure

généralisation des données d'entraînement. La sortie du ReLU agit comme une connexion de saut pour le bloc de décodeurs correspondant. Suit ensuite une opération de pooling MAX de 2x2, où les dimensions spatiales (hauteur et largeur) des cartes d'entités sont réduites de moitié. Cela réduit le coût de calcul en diminuant le nombre de paramètres devant être entraînés.

Le réseau de décodeurs est utilisé pour prendre la représentation abstraite sortant des blocs d'encodage et de générer un masque de segmentation sémantique. Le bloc décodeur commence par une convolution de transposition 2x2. Ensuite, cette sortie est concaténée avec le mappage d'entités de saut de connexions correspondant à partir du bloc d'encodeurs. Ces connexions de saut fournissent des fonctionnalités des couches antérieures qui sont parfois perdues en raison de la profondeur du réseau. Ensuite, deux convolutions 3x3 sont utilisées, où chaque convolution est suivie d'une fonction d'activation ReLU. La sortie du dernier décodeur passe par une convolution 1x1 avec activation sigmoïde. La fonction d'activation sigmoïde donne le masque de segmentation représentant la classification au pixel près.

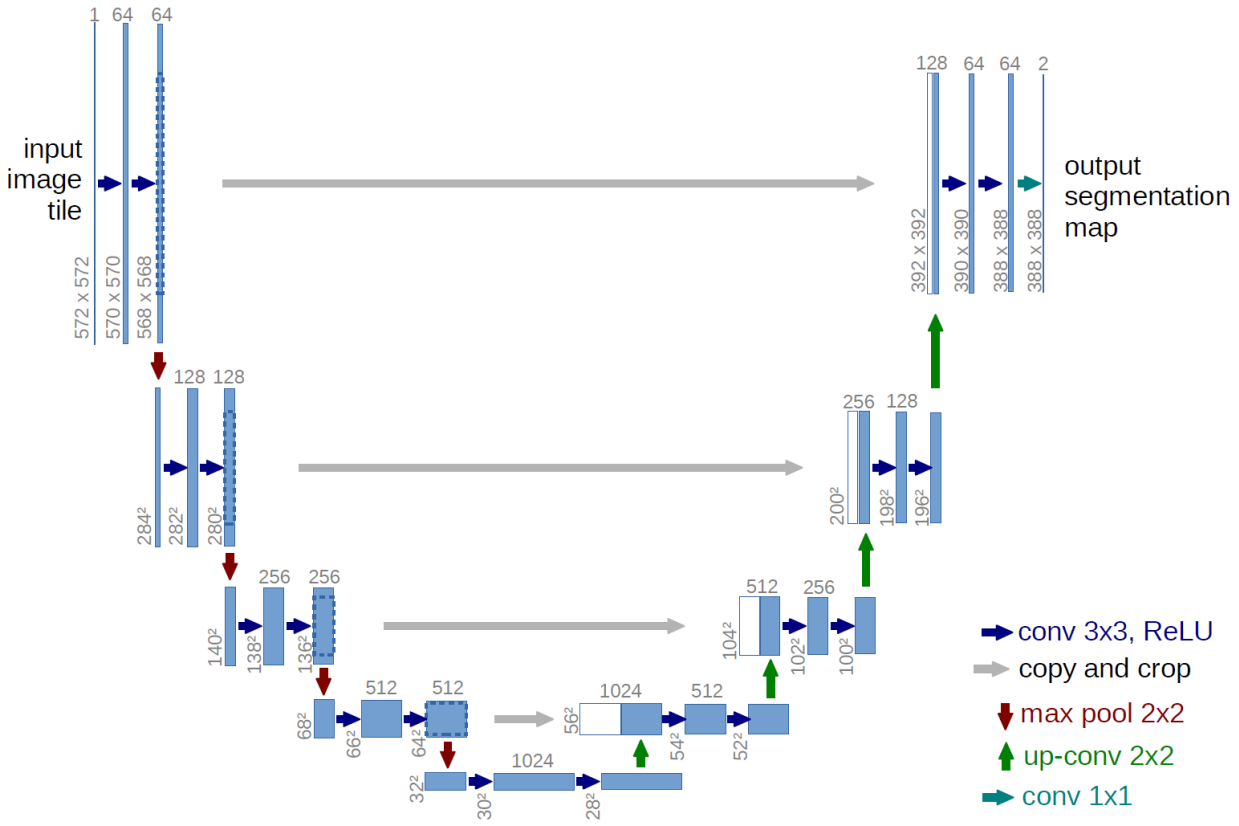

FIGURE 7 – Architecture U-NET
Source : (Ronneberger et al. (2015))

2.7 Techniques de classification

Avec les régions extraites de l'étape de segmentation, il est aussi possible d'extraire d'autres caractéristiques, comme des paramètres de formes de ces régions. L'ensemble des caractéristiques extraites peuvent ensuite être utilisées dans la phase de classification. Plusieurs algorithmes de classification peuvent être utilisés. Des approches comme :

KNN (K-Nearest Neighbors)

KNN est un algorithme d'apprentissage non paramétrique d'apprentissage automatique[Cover and Hart (1967), Fix and Hodges (1989)] utilisé pour les problèmes de classification et de régression. Cet algorithme fonctionne simplement en stockant tous les points d'observations (exemples) donnés lors de l'entraînement et leur étiquette. Au moment de l'utilisation d'un classificateur, les k points dans l'espace des données d'entraînement les plus près sont sé-

lectionnés et leur étiquette est consultée. La classification se fait en prenant le mode des étiquettes de ces k points les plus près. Dans la classification, la sortie est une appartenance à une classe (prédiction) et dans la régression, la sortie est la valeur d'une variable continue. Cependant, la méthode KNN peut être coûteuse en calcul, car elle nécessite de stocker et de comparer tous les points de données d'apprentissage pour chaque prédiction. Elle est également sensible au choix de la mesure de distance, la valeur de K choisie, et à l'échelle des données. Pour classer un nouveau point de données à l'aide de KNN, les étapes suivantes sont généralement suivies :

1. **Calculer** la distance entre le nouveau point de données et tous les points de données dans l'espace d'entraînement.
2. **Sélectionner** les K points de données dans l'espace d'entraînement les plus proches du nouveau point de données à classer.
3. **Attribuer** l'étiquette de classe au nouveau point de données en fonction de la classe majoritaire des K voisins les plus proches.

La valeur de K est un hyperparamètre choisi par l'utilisateur. Une valeur plus élevée de K signifie que l'algorithme est moins sensible aux valeurs aberrantes, mais peut produire un modèle moins précis. Une valeur plus petite de K signifie que le modèle est plus sensible aux valeurs aberrantes, mais peut donner un modèle plus précis.

SVM (Support Vector Machine)

Les machines à vecteurs de support (SVM) sont un type d'algorithme d'apprentissage supervisé puissant et largement utilisé pour résoudre des problèmes de classification (linéaire et non linéaire), de régression et même de détection des valeurs aberrantes, ce qui en fait l'un des algorithmes d'apprentissage automatique les plus populaires [M. Mohammed and Bashier (2017) , Géron (2017)]. Son utilisation est également populaire dans le diagnostic des défauts des machines tournantes (Widodo and Yang (2007)). Ils sont basés sur l'idée de trouver un hyperplan capable de séparer des ensembles de données en espaces d'entités de grande dimension. La séparation entre les jeux de données est appelée marge, et la SVM maximise cette marge (M. Mohammed and Bashier (2017)).

Un jeu de données séparable linéairement permet au SVM de définir des hyperplans capables de séparer les données en catégories, quel que soit le nombre de dimensions présentes dans l'espace d'entités. Cependant, dans la plupart des applications, les informations ne sont pas toujours séparables linéairement dans des espaces d'entités avec une dimensionnalité donnée. Ainsi, il est nécessaire de mapper l'ensemble de données dans un espace d'entités avec un plus grand nombre de dimensions, dans lequel les données seront linéairement séparables. Ce processus de mappage est effectué à l'aide de noyaux, par exemple des noyaux de fonctions de base polynomiales et radiales (RBF) [M. Mohammed and Bashier (2017) , Géron (2017)].

Dans les tâches de classification, un algorithme SVM prend un ensemble de données d'apprentissage étiquetées et trouve l'hyperplan dans l'espace des caractéristiques qui sépare au maximum les différentes classes. Les nouveaux points de données à être classés sont ensuite classés en fonction du côté de l'hyperplan sur lequel ils se trouvent.

Dans les tâches de régression, un algorithme SVM trouve l'hyperplan qui correspond le mieux aux données, avec la marge maximale entre les points de données et cet hyperplan.

Les SVM présentent un certain nombre d'avantages, notamment leur capacité à gérer des données de grande dimension, leur capacité à gérer des données avec plusieurs classes et leur capacité à gérer des données avec du bruit ou des valeurs aberrantes. Ils sont également relativement simples à mettre en œuvre et peuvent être efficaces pour les grands ensembles de données. Cependant, ils peuvent être sensibles au choix de la fonction du noyau et peuvent nécessiter beaucoup de calculs pour de très grands ensembles de données.

Réseaux de neurones probabilistes (PNN)

Les réseaux de neurones de probabilités ou Probabilistic neural networks (PNN) en anglais sont introduits d'abord par Specht (1990). L'architecture relativement simple donnée par Specht introduit une partie probabiliste issue des réseaux bayésiens contenant quatre couches majeures : la couche d'entrée, la couche cachée (couche de motifs), la couche de sommation

et la couche de sortie illustrée dans la figure 8 :

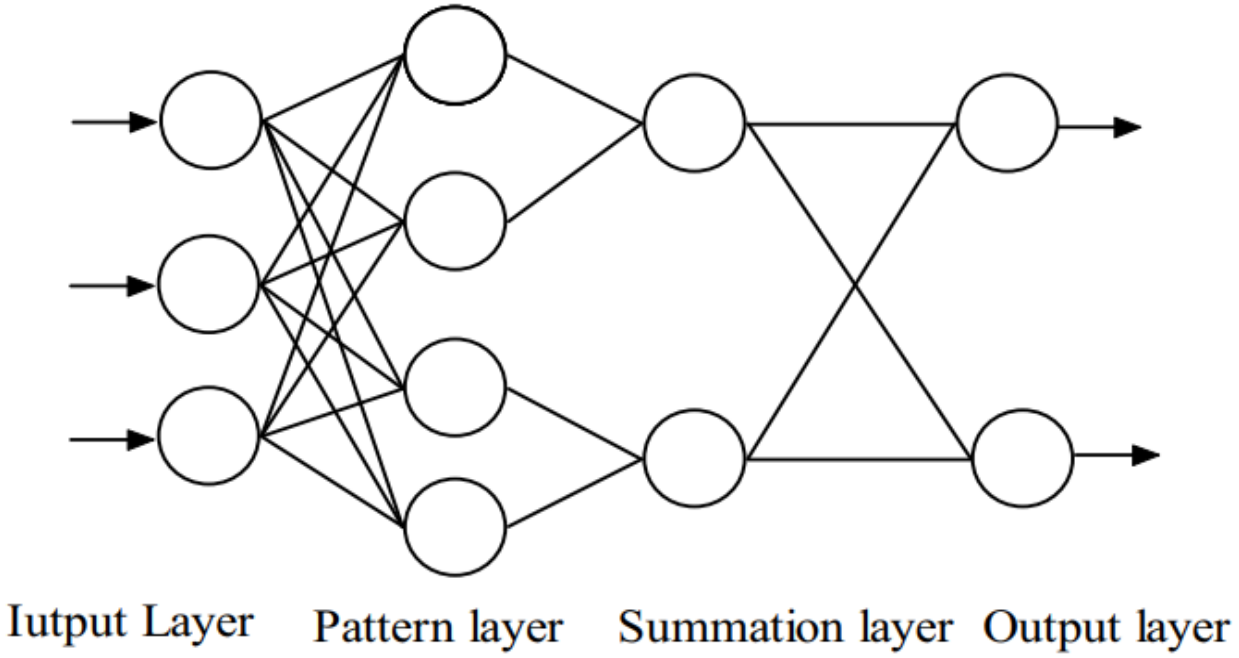

FIGURE 8 – Réseaux de neurones probabilistes
Source : (Chen et al. (2010))

Seuls les neurones de deux couches adjacentes sont interconnectés. L'information transite dans un seul sens, d'une couche n à une couche n+1. Chaque neurone d'une couche est dédié à une même tâche. La couche d'entrée transmet les informations de chacun des neurones en entrée à la couche de motifs suivante avec laquelle ce neurone est totalement connecté. Dans la couche de motifs, chaque classe est assignée à un neurone. Chaque neurone effectue le calcul de la distance euclidienne entre l'observation nouvelle et l'observation apprise. La couche de sommation récupère ces informations en effectuant une somme des sorties des neurones cachés et transmet le résultat de cette sommation à la couche de sortie qui effectue une prédiction de la classe.

Chen et al. (2010) ont effectué une étude comparative pour identifier les mauvaises herbes dans une culture de maïs avec l'utilisation des réseaux de neurones probabilistes et la rétro-propagation donnant un résultat de 95% de bonnes classifications. Ainsi, cette étude semble vouloir donner avantage aux réseaux de neurones probabilistes en ce qui concerne la classi-

fication automatique des mauvaises herbes dans les cultures en rangées comme celle du maïs.

Réseaux de neurones convolutif (CNN)

Les architectures de réseaux de neurones convolutifs obtiennent souvent de grandes performances dans le domaine de l'apprentissage profond. Leur facilité d'utilisation les rendent populaires et très utilisées. Il en existe plusieurs, nous allons en exposer quelques-unes d'entre elles :

LeNet-5

LeNet-5 est une architecture de réseau neuronal convolutif (CNN) développée par LeCun et al. (1989). Il s'agit d'un travail de pionnier dans le domaine de l'apprentissage profond et est largement considéré comme la première mise en œuvre réussie d'un CONVNET pour la classification d'images. Il consiste en une séquence de couches convolutives, de mise en commun et entièrement connectées. Il a été conçu pour classer les chiffres manuscrits de l'ensemble de données MNIST (Lecun et al. (1998)), mais a également été appliqué à d'autres tâches de classification d'images. Cette architecture possède un certain nombre de caractéristiques clés qui ont contribué à son succès, notamment sa capacité à apprendre automatiquement des fonctionnalités à partir des données, sa capacité à généraliser avec de nouvelles données et son nombre relativement faible de paramètres. Malgré son âge, LeNet-5 reste un modèle populaire et influent dans le domaine de l'apprentissage en profondeur, et son architecture de base a inspiré de nombreux CNNs ultérieurs..

AlexNet

AlexNet est une architecture de réseau neuronal convolutif (CNN) développée par Krizhevsky et al. (2012), auteur principale ayant comme prénom Alex, d'où son nom AlexNet, ainsi que les chercheurs Ilya Sutskever et Geoffrey Hinton(Krizhevsky et al. (2012)). AlexNet a été développé pour classer les images de l'ensemble de données ImageNet, qui contient plus d'un million d'images dans 1000 classes. Il a obtenu des performances nettement meilleures que les architectures CNN précédentes et a remporté le défi de reconnaissance visuelle à grande

échelle ImageNet (ILSVRC) en 2012 avec un résultat top-5 avec 15.3% d'erreurs, devançant le second au classement de 10%. AlexNet consiste en une séquence de couches convolutives, de mise en commun et entièrement connectées, et a introduit plusieurs innovations qui sont devenues la norme dans les CNN modernes, notamment l'utilisation d'unités linéaires rectifiées (ReLU) comme fonction d'activation et l'utilisation de l'abandon (dropout) pour éviter le surajustement. Dans l'ensemble, AlexNet est une contribution significative au domaine de l'apprentissage en profondeur, et son succès sur l'ensemble de données ImageNet a contribué à faire des CNNs un outil puissant pour les tâches de classification d'images.

GoogLeNet (Inception V1)

GoogLeNet (également connu sous le nom d'Inception V1) est une architecture de réseau neuronal convolutif (CNN) développée par Christian Szegedy et al. chez Google en 2014 (Szegedy et al. (2014)). Cette architecture a aussi été développé pour classer les images de l'ensemble de données ImageNet, qui rappelons le, contient plus d'un million d'images dans 1000 classes. GoogleLeNet a obtenu des performances nettement meilleures que les architectures CNN précédentes et a remporté le ImageNet Large Scale Visual Recognition Challenge (ILSVRC) en 2014. Les résultats obtenus diminuent considérablement la marge d'erreur comparée aux résultats obtenus dans les précédentes versions du concours ainsi qu'avec son second concurrent. Ceci en utilisant des sous-réseaux nommés «module Inception» qui sous-entend le fait d'aller plus profondément dans le réseau. GoogLeNet est connu pour son utilisation de modules de démarrage, qui sont conçus pour réduire le nombre de paramètres et le coût de calcul du réseau. Il a également introduit l'utilisation de classificateurs auxiliaires, qui sont des classificateurs supplémentaires formés avec le classificateur principal afin d'améliorer les performances.

VGGNet

VGGNet est une architecture de réseau neuronal convolutif (CNN) qui a été développée par Karen Simonyan et Andrew Zisserman à l'Université d'Oxford en 2014 (Simonyan and Zisserman (2014)). VGGNet a aussi été développé pour classer les images de l'ensemble de données ImageNet. Il a atteint des performances de pointe sur l'ensemble de données

ImageNet et a été finaliste du ImageNet Large Scale Visual Recognition Challenge (ILSVRC) en 2014 en obtenant la seconde place. Connu pour son utilisation d'une architecture profonde et étroite, avec un grand nombre de couches convolutives et un nombre relativement faible de filtres dans ces couches. Il a également introduit l'utilisation de petits filtres convolutifs de 3x3, qui sont devenus un standard dans les CNNs modernes.

Convnet

Un réseau neuronal convolutif (Convnet ou CNN) est un type de réseau neuronal profond spécialement conçu pour traiter des données avec une topologie en forme de grille, telle qu'une image. Les Convnets sont composés de plusieurs couches de nœuds interconnectés, chaque couche apprenant à extraire des caractéristiques de plus en plus complexes des données.

Particulièrement utiles pour des tâches telles que la classification d'images, la détection d'objets et la génération d'images, ils sont capables d'apprendre automatiquement des représentations hiérarchiques des données, ce qui leur permet d'atteindre des performances de pointe sur de nombreuses tâches.

Les Convnets sont composés de plusieurs couches, y compris des couches convolutives, des couches de regroupement et des couches entièrement connectées. Les couches convolutives appliquent des opérations de convolution aux données d'entrée, qui permet l'extraction des caractéristiques locales des données. Les couches de regroupement réduisent la dimensionnalité des données en les sous-échantillonnant, ce qui contribue à réduire la complexité de calcul du réseau. Les couches entièrement connectées connectent chaque neurone d'une couche à chaque neurone de la couche suivante et sont utilisées pour effectuer la prédiction ou la classification finale.

Resnet

ResNet (Residual Network) est une architecture de réseau neuronal convolutif (CNN) développée par Kaiming He et al. en 2015 (He et al. (2015)). Il s'agit aussi à l'instar de LeCun, d'un travail de pionnier dans le domaine de l'apprentissage en profondeur et est largement considéré comme une étape majeure dans le développement des CNN pour la classification

des images. Développé pour classer les images du jeu de données ImageNet. Il a obtenu des performances nettement meilleures que les architectures CNN précédentes et a remporté le ImageNet Large Scale Visual Recognition Challenge (ILSVRC) en 2015 avec un taux d'erreurs de 3,57% sur l'ensemble de test ImageNet. ResNet est connu pour son utilisation de connexions résiduelles, qui sont des connexions raccourcies qui sautent une ou plusieurs couches et permettent au réseau d'apprendre le mappage résiduel entre l'entrée et la sortie d'une couche. Cela permet à ResNet de former des réseaux très profonds, avec jusqu'à 152 couches, sans souffrir du problème de gradient de fuite qui peut survenir dans les réseaux très profonds.

Dans ce contexte, plusieurs architectures de réseaux de neurones convolutifs sont utilisées pour la classification de plantes permettant ainsi de différentier les bonnes plantes comme le bleuet nain des mauvaises herbes comme le Kalmia à feuilles étroites. Par exemple, Sa et al. (2016) , dos Santos Ferreira et al. (2017) et Bodhwani et al. (2019) utilisent des réseaux neuronaux convolutifs pour identifier les plantes. Les images hautes résolutions présentées dans ces articles sont prises avec une profondeur de champ très proches des plantes, permettant donc de capturer des détails sur la couleur et la forme de ces plantes. C'est pourquoi l'utilisation de réseaux neuronaux est justifiée dans ces articles, car ceux-ci performent bien quand beaucoup de caractéristiques doivent être extraites de ces images pour permettre une classification efficace. Sachant qu'il est possible d'utiliser les réseaux de neurones convolutifs pour classifier les plantes, le réseau de neurone Resnet sera alors l'approche utilisée dans cette présente recherche pour discerner les mauvaises herbes des bleuets.

2.8 Conclusion

Dans ce chapitre, le cycle de culture du bleuet a d'abord été abordé, suivi d'une description des mauvaises herbes nuisibles à cette plante. Ensuite, nous avons divisé le processus de détection automatique des mauvaises herbes en deux approches : une première approche faisant l'ébauche des étapes du processus de segmentation basées sur les méthodes de seuillage, de clustering et une méthode de segmentation basée sur les réseaux de neurones convolutifs. Pour la deuxième approche, nous avons exposer les différentes techniques de classification existantes dans la littérature. Dans le chapitre suivant, nous détaillerons les concepts et les techniques explorés dans le présent chapitre qui seront mis en pratique pour résoudre la problématique de détection automatique des mauvaises herbes dans les cultures de bleuets qui fait l'objet du présent mémoire. La méthodologie préconisée sera alors présentée concrètement afin de permettre d'atteindre les objectifs fixés au début du mémoire.

Chapitre 3 Méthodologie

3.1 Introduction

Dans ce présent chapitre nous allons implémenter un algorithme de segmentation et de reconnaissance automatique de mauvaises herbes dans les cultures de bleuets nains basée sur une approche neuronale. Ce genre d'approches a été présentées au chapitre 2, dans la revue de littérature. De ces approches présentées, celles des réseaux de neurones convolutifs comme la combisaison de ResNet et UNET ont été retenues dans la présente recherche puisque celles-ci offrent les meilleures perspectives en termes d'efficacité de détection et de classification.

3.2 Prétraitements des données

Avant d'entamer toutes tâches de reconnaissance et de classification il faut tout d'abord effectuer un prétraitement des données. Le prétraitement des données est une technique d'exploration de données qui consiste à transformer des données brutes dans un format compréhensible pour un réseau de neurones. Les données réelles sont souvent incomplètes, incohérentes et/ou absentes de certains comportements ou tendances, et sont susceptibles de contenir de nombreuses erreurs. Le prétraitement des données est une méthode éprouvée pour résoudre ces problèmes.

3.2.1 Annotation des données

L'annotation des données pour la vision par ordinateur est à priori une opération difficile, car il existe plusieurs types de techniques utilisées pour configurer les algorithmes dédiés à l'apprentissage des ensembles de données et à la prédiction des résultats. L'annotation d'image est une méthode d'annotation des images contenant les objets d'intérêt pour ainsi les rendre reconnaissables aux machines. Bien qu'il existe différents types de techniques d'annotation d'images, la segmentation sémantique est l'une d'entre elles. La segmentation sémantique permet de fournir une image correctement annotée aux algorithmes de vision par

ordinateur. Il s'agit purement d'une technique à forte intensité de main-d'œuvre nécessitant une précision au niveau des pixels. Et cette technique est non seulement cruciale mais aussi coûteuse, par conséquent, l'utilisation du bon outil est très important pour maintenir une précision élevée. C'est dans cette logique que nous avons utilisé l'outil Napari pour annoter les données pour la segmentation sémantique. Dans cette présente recherche, le processus d'annotation des images a permis de distinguer 4 classes : l'arrière-plan, les plants de bleuet, les plants de Comptonie voyageuse et les plants de Kalmia à feuilles étroites.

3.3 Organisation des données

3.3.1 Séparation des données

Une fois l'annotation des images effectuée et avant l'entraînement proprement dit du réseau neuronal qui permet l'automatisation de la segmentation et la reconnaissance des images de plantes comme les mauvaises herbes, il faut construire la banque de données d'images. Cette banque d'images est construite comme suit : on doit d'abord créer les dossiers suivants : Train et Validation contenant chacun des sous dossiers images et labels ainsi qu'un dossier Test contenant qu'un sous-dossier images utilisées pour tester le modèle préalablement entraîné. Il est bon de noter ici, que nos données comportent des images regroupant des plants de bleuet et des plants de mauvaises herbes (ex : Kalmia et/ou Comptonie), des images contenant que des mauvaises herbes (Comptonie et/ou Kalmia) et des images contenant que des plants de bleuet (Voir Figure 9).

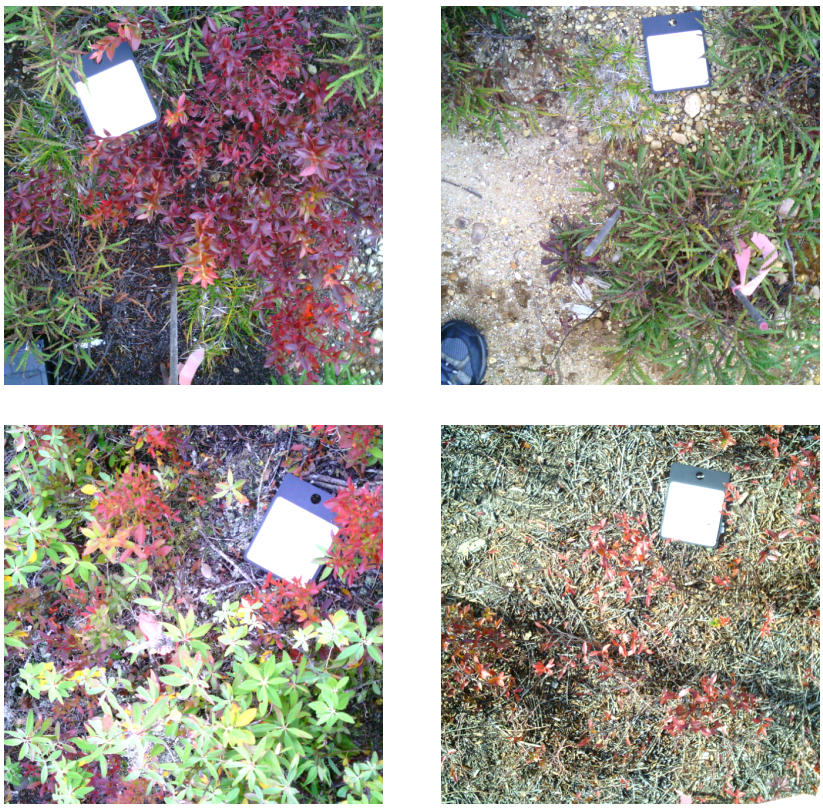

FIGURE 9 – À gauche des images regroupant des plants de bleuet et des plants de mauvaises herbes (Kalmia et/ou Comptonie) et à droite une image contenant que des mauvaises herbes (Comptonie) et une image contenant que des plants de bleuet

Pour l'organisation des données, on a trois dossiers : Train, Validation et Test dont les deux premiers dossiers contiennent les sous dossiers images et labels alors que le dossier Test contient que le sous dossier images. Dans le sous dossier images de chacun des deux premiers dossiers Train et Validation, on trouve des images mixées (image regroupant des plants de bleuet et une mauvaise herbe : Kalmia ou Comptonie) et des images non mixées (regroupant seulement une mauvaise herbe ou le bleuet). De plus dans chaque dossier Train et Validation nous retrouvons un sous-dossier labels dans lequel les images labellisées sont situées. Pour le dossier Test, on a seulement le sous dossier images contenant aussi des images mixées et des images non mixées. L'organisation des données peut être observée sur la Figure 10.

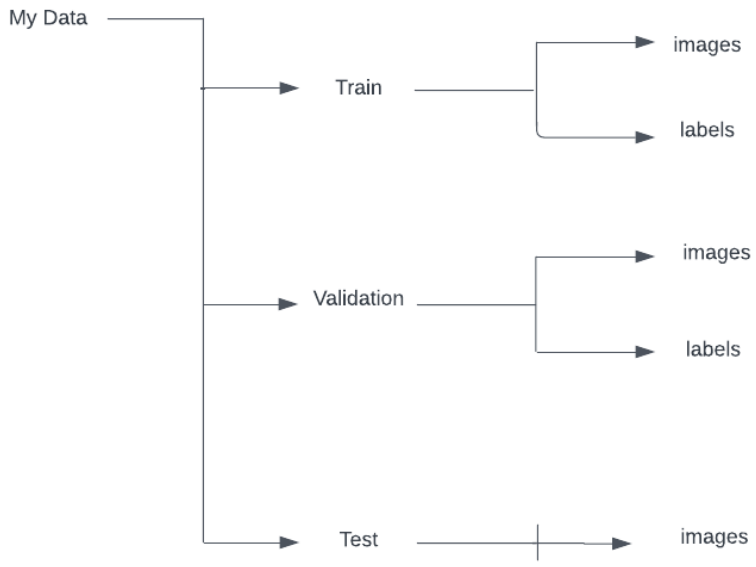

FIGURE 10 – Organisation des données

La distribution exacte des données est décrite dans le tableau 1. Notons que le nombre d'images contenu dans le dossier Test n'a aucun impact sur l'apprentissage étant donné qu'on a besoin que de quelques images pour tester l'efficacité de la prédiction du modèle.

	Train	Validation	Test
Images	17	7	9

TABLE 1 – Nombre d'images dans le corpus de données

Les images contenues dans les dossiers d'entraînement (Train) représentent un peu plus de 60% du corpus d'images initial alors que celles dans les dossiers de validation (Validation) presque 40%.

Cependant, n'ayant pas assez d'images dans ce corpus de données initial, nous avons alors été forcé de faire un traitement supplémentaire qui consistait à augmenter ce corpus d'images afin d'obtenir des résultats de classification suffisamment bon pour atteindre nos objectifs de performance.

3.3.2 Augmentation des données

Avant de pouvoir séparer nos données, et sachant que pour notre recherche nous avions besoin d'un nombre assez important d'images exploitables pour espérer avoir de meilleurs résultats de classification, nous devions d'abord penser augmenter notre corpus d'images. Afin de résoudre cette problématique nous avons appliqué sur le corpus d'images original, une technique d'augmentation déjà utilisée par la communauté scientifique. Nous avons dans cet optique, utilisé la bibliothèque albumentations qui est une bibliothèque d'augmentation d'images rapide et flexible permettant entre autres de faire des manipulations et des augmentations sur les images en temps réel. Parmi ces manipulations nous utilisons le retournement horizontal, le retournement vertical, la rotation, le changement d'échelle, la luminosité et le contraste.

La répartition des données après l'augmentation faite sur nos images Train et Validation est décrite dans le tableau 2. Notons que le nombre d'images contenu dans le dossier Test n'a subi aucune augmentation sachant qu'on aura besoin de ces images lors de la prédiction une fois le traitement terminé (entraînement et validation).

	Train	Validation	Test
Images	8500	3500	9

TABLE 2 – Nombre d'images dans le corpus de données après augmentation

3.4 Extraction et visualisation des données

3.4.1 Extraction des données (.zip)

Le dossier contenant les images annotées et augmentées est par la suite archivé en format Zip. Ces données sont ensuite chargées dans un dossier Drive pour faciliter l'accès et l'exploitation. Pour les expérimentations (entraînement, validation et test), la plateforme Google Colab va être utilisée pour octroyer le droit d'accès du script python au dossier zippé contenant les données. Une fois que le dossier zippé est accessible et téléchargé dans Google Colab, la prochaine phase consiste à décompresser les données contenues dans le dossier des données

afin de les manipuler et les exploiter pour la suite des processus d'entraînement, de validation et de test du modèle de classification automatique de mauvaises herbes.

3.4.2 Visualisation des données

Après le processus d'extraction des données, nous avons créé un chemin d'accès pour permettre l'accès aux dossiers des étiquettes (mask) et des images :

```
path = '/content/'
path_lbl = path + 'train/masks'
path_img = path + 'train/images'
```

Ensuite, on peut sélectionner et visualiser la liste des identifiants de fichier images du dossier Train :

```
[203] imageFileNames = get_image_files(path_img)
      imageFileNames.sort()
      imageFileNames[:1]

      (#1) [Path('/content/train/images/RGBBACKGROUND_2020-09-03_001-Comptonie_0.png')]
```

Puis ensuite, sélectionner et visualiser la liste des identifiants de fichier d'étiquettes associés aux images.

```
label_names = get_image_files(path_lbl)
label_names.sort()
label_names[:1]

(#1) [Path('/content/train/masks/RGBBACKGROUND_2020-09-03_001-Comptonie_0_mask.png')]
```

Les fichiers d'étiquettes de chaque image sont facilement identifiables par le libellé _mask ajouté à leurs noms. C'est de cette façon qu'il devient possible d'associer les étiquettes aux images. Par conséquent, nous pouvons créer une fonction lambda (voir le code de la fonction label_func()) définissant la relation entre le dossier des étiquettes et celui des images.

```
[162] def label_func(x):
        x = str(x).replace('images','masks').replace('.png','_mask.png')
        return Path(x)
```

Dans la figure 11, il est possible de visualiser une image source comportant du bleuet (partie haute de l'image) et de la Comptonie (centre de l'image) et la figure 12 une image résultante de l'étiquetage. Cette image permet de constater que les pixels correspondant au bleuet sont étiquetés en brun et la Comptonie en bleu pâle. Ainsi, pour visualiser l'image, nous utiliserons **PILIMAGE** de la bibliothèque Fastai.

```
img_fn = imageFileNames[1465]
img = PILImage.create(img_fn)
img.show(figsize=(5,5))
```

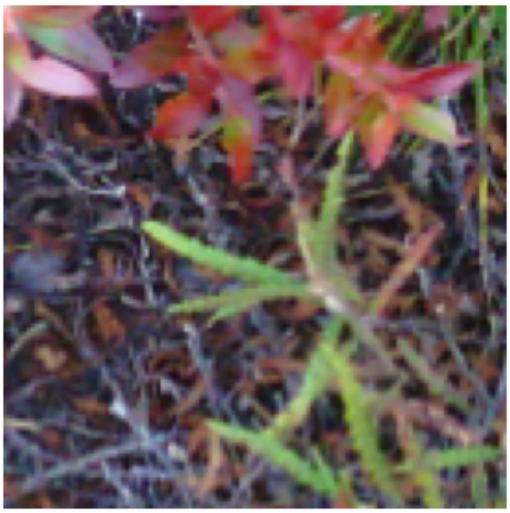

FIGURE 11 – Image source dans le spectre visible. Les feuilles des plants de bleuets sont rouges et celles de la Comptonie sont vertes.

Les images standards (voir figure 11) sont constituées de pixels numérisés en format point flottant représentant l'intensité du rayonnement visible réfléchit par le couvert végétal. Cependant, les images d'étiquettes (voir figure 12) ne sont pas des images typiques. Elles peuvent

être constituées de valeurs numériques entières qui sont les étiquettes représentant les différents objets classifiés dans l'image. Ainsi, pour pouvoir visualiser les étiquettes d'une image, nous utiliserons **PILMASK** de la bibliothèque Fastai.

```
msk = PILMask.create(label_func(img_fn))
msk.show(figsize=(5,5), alpha=1)
```

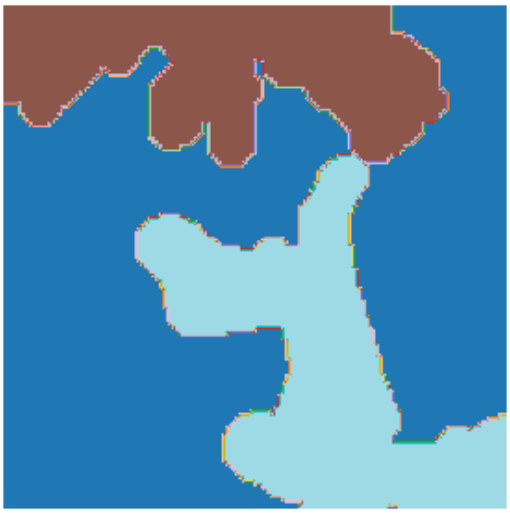

FIGURE 12 – Image d'étiquettes découlant du processus de labellisation d'une image panchromatique (spectre visible) (voir figure 11)

Maintenant, nous présentons un exemple du contenu d'un fichier d'étiquettes que nous appelons masque :

```
[ ]  tensor(msk)

     tensor([[1, 1, 1,  ..., 0, 0, 0],
             [1, 1, 1,  ..., 0, 0, 0],
             [1, 1, 1,  ..., 0, 0, 0],
             ...,
             [0, 0, 0,  ..., 2, 2, 2],
             [0, 0, 0,  ..., 2, 2, 2],
             [0, 0, 0,  ..., 2, 2, 2]], dtype=torch.uint8)
```

FIGURE 13 – Tableau des étiquettes de chaque pixel d'une image. Correspondant à l'image précédente découlant du processus labellisation. Les premiers pixels sont étiquetés 1 (Bleuets) et les derniers 2 (Comptonie).

Nous pouvons voir que le fichier d'étiquettes présenté à la figure 13 est en fait une liste de valeurs numériques entières identifiant l'étiquette (classe d'appartenance : 0 : Background, 1 : Bleuet, 2 : Comptonie, 3 : Kalmia) de chaque pixel d'une image, sauvegardé dans un fichier .txt

```
[19] codes = np.loadtxt('codes.txt', dtype=str); codes

     array(['Background', 'Bleuet', 'Comptonie', 'Kalmia'], dtype='<U10')
```

3.5 Entraînement et validation

Après avoir effectué les phases de prétraitement et d'organisation des données, il est maintenant possible de débuter l'entraînement du réseau de neurones convolutifs qui se compose quant à lui de plusieurs parties successives.

3.5.1 Importation des librairies utiles

L'industrie informatique accélère le développement de machines intelligentes, capables de présenter un comportement humain en matière d'apprentissage. Cette simulation de l'intelligence humaine s'appuie sur diverses bibliothèques Python spécialement conçues pour dynamiser et optimiser cette branche de l'informatique. Dans les sections précédentes, nous avons déjà cité certaines bibliothèques Python sans explicitement les présenter. Ces bibliothèques sont : la couche d'application Vision de la bibliothèque Fastai (Learner, Optimizer, Datablock), le package Torchvision de la bibliothèque Pytorch, la bibliothèque Numpy de TensorFlow pour la manipulation de matrices et de tableaux multidimensionnels et la bibliothèque Matplotlib.pyplot pour créer des visualisions statiques et interactive.

3.5.2 Création d'un groupe de données (Lot de données, batch) à l'aide de l'API de bloc de données Fastai

Nous pouvons créer un groupe de données (lot de données, batch) à l'aide de l'API de bloc de données Fastai.

```
dls1 = SegmentationDataLoaders.from_label_func(
            path,
            bs=8,
            fnames = imageFileNames,
            splitter = GrandparentSplitter(train_name='train', valid_name='valid'),
            label_func = label_func,
            codes = ['Background', 'Bleuet', 'Comptonie', 'Kalmia'],
            )
```

Pour ce faire, nous utilisons la fonctionnalité **SegmentationDataLoaders.from_label_func** en fonction du type d'entrées que nous avons besoin. Les arguments de cette méthode sont :

- **fnames** : les noms des fichiers de données d'image associés au chemin spécifié.
- **splitter** : facteur de découpage permettant de séparer notre jeu de données en deux dossiers : Train et Validation à l'aide de la fonction **GrandparentSplitter**. Nous devons fournir un chemin d'accès valide aux dossiers contenant ces fichiers images, Train et Validation.
- **label_func** : permet de spécifier la fonction lambda label_func définissant les étiquettes associées aux images.
- **codes** : ce fichier contient la liste de noms de chaque classe. Ces fichiers sont alors chacun marqué d'un seul numéro. Il devient donc possible de spécifier à l'API que chaque bloc de données est associé à une étiquette numérique.

3.5.3 Taille du lot de traitement (batch size)

Ici, la taille du lot (batch size) est le nombre d'images à traiter à la fois. Le traitement nécessite une quantité énorme de ressources. Nous avons géré cette problématique en réduisant le nombre d'images traitées par lot. La valeur du nombre d'images peut ainsi varier et prendre des valeurs de 8, 16, ... ou 64 en fonction de la configuration du nombre d'unités de traitement points flottants (GPUs) disponibles. Ainsi, nous avons arrêté notre choix sur une taille de lot de 8 images par époque (epoch).

3.5.4 Nombre d'itérations

Dans le contexte de l'apprentissage automatique, le terme utilisé epoch (époque) est synonyme d'itération et fait référence à un passage sur l'ensemble de données d'apprentissage

pendant la phase d'apprentissage d'un modèle d'apprentissage automatique. Au cours d'une époque, l'algorithme d'apprentissage prend l'intégralité de l'ensemble de données d'apprentissage et le divise en plusieurs lots (batch) plus petits. Le modèle est ensuite entraîné successivement sur chaque lot, les paramètres (poids des connexions entre neurones de couches successives) du modèle étant mis à jour après le passage de chaque lot dans le réseau de la couche d'entrée jusqu'à la couche de sortie. Dans cette présente recherche, nous avons entraîné la couche entièrement connectée (proche de la sortie du réseau), initialement non figée, pendant quelques époques (5) en utilisant la méthode .fit_one_cycle() afin de configurer les paramètres de cette couche entièrement connectée que nous voulons adapter pour la tâche spécifique de détection/classification de mauvaises herbes. Ensuite, nous avons défigé les calques figés au préalable en utilisant la méthode .unfreeze() en poursuivant l'entraînement du modèle pendant 10 epochs. Le processus itératif peut être interrompu par un script (voir la section 3.5.5) si aucune amélioration des métriques d'évaluation de la qualité du modèle n'est notée. Au terme d'un certain nombre d'itérations, lorsque le modèle optimal est obtenu, le modèle d'apprentissage est sauvegardé.

3.5.5 Les sauvegardes et arrêts précoces

Au fur et à mesure que le traitement progresse, notre modèle peut éprouver des phases de non-apprentissage, ce qui signifie qu'il atteint une itération où il n'apprend pas plus. Pour résoudre ce problème, nous utilisons la fonction EarlyStoppingCallback pour implémenter un arrêt anticipé intégré dans le module Callback de la bibliothèque Fastai. Cela nous permet de passer à l'étape suivante, évitant ainsi un traitement prolongé inutile. Dans le même module, nous faisons appel à la fonction SaveModelCallback pour enregistrer le modèle et la fonction ShowGraphCallback pour tracer les graphes de nos métriques de la qualité du modèle entraîné.

3.6 Le modèle

La segmentation d'image est la méthode permettant de partitionner une image en différentes régions, chaque région ayant une étiquette différente. Les réseaux de neurones convolutifs

standards réussissent pour les images plus simples, mais ne donnent pas toujours de bons résultats pour les images plus complexes. C'est pour ces raisons que d'autres algorithmes de segmentation comme UNet et ResNet ont été retenus dans cette recherche.

3.6.1 U-Net Architecture

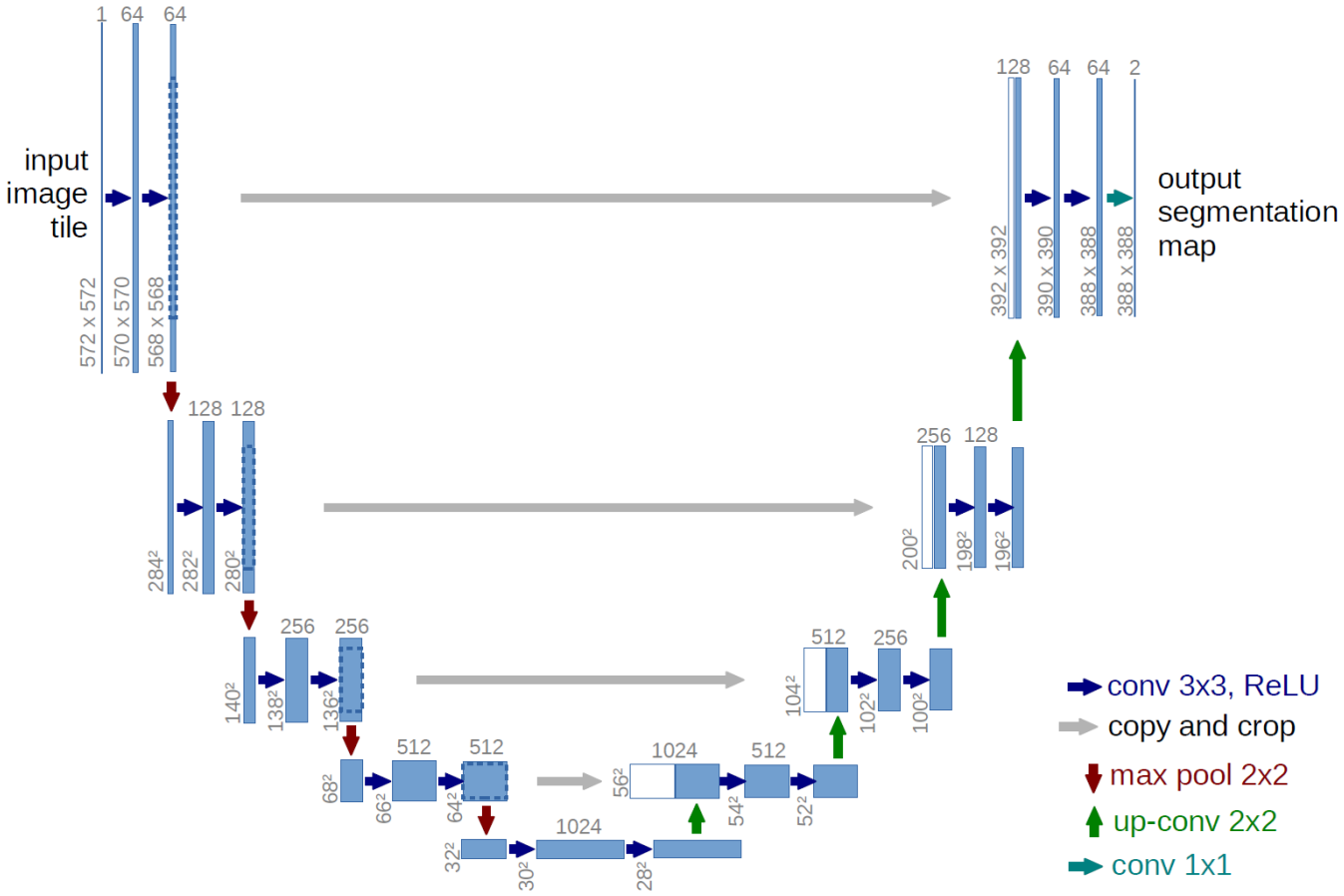

FIGURE 14 – Architecture U-NET
Source : (Ronneberger et al. (2015))

L'architecture UNet (voir Figure 14) se compose de couches d'opérations de convolution (couches convolutives), de regroupement maximal (pooling max), d'activation ReLU, de concaténation et d'échantillonnage Up, organisées en trois sections : contraction (branche gauche) aussi appelé encodage, goulot d'étranglement (en bas) et section d'expansion aussi appelée décodage (branche droite). La section des contractions comporte 4 blocs de contraction. Chaque bloc de contraction reçoit une entrée, applique deux couches ReLu de convolution 3X3, puis un pooling 2X2 max (regroupement MAX). Le nombre de cartes d'entités

est doublé à chaque couche de pooling max. La couche de goulot d'étranglement utilise deux couches Conv 3X3 et une couche de convolution 2X2 up. La section d'expansion se compose de plusieurs blocs d'expansion, chaque bloc passant l'entrée à deux couches Conv 3X3 et à une couche de suréchantillonnage 2X2 qui réduit de moitié le nombre de canaux d'entité. Il inclut également une concaténation avec la carte d'entités recadrée correspondante à partir du chemin de contraction. En fin de compte, la couche Conv 1X1 est utilisée pour rendre le nombre de cartes d'entités identique au nombre de segments (classes) souhaités dans la couche de sortie avec activation sigmoïde. U-net utilise une fonction de perte pour chaque pixel de l'image. Cela facilite l'identification des cellules individuelles dans la carte de segmentation. La fonction d'activation sigmoïde donne le masque de segmentation représentant la classification au pixel près.

3.6.2 Réseaux résiduels (ResNet)

Dans les réseaux neuronaux traditionnels, avoir plus de couches signifie un meilleur réseau, mais en raison du problème de gradient de fuite, les poids des premières couches du réseau ne seront pas mis à jour correctement par la rétropropagation. Comme le gradient d'erreur est rétropropagé aux couches précédentes à partir de la couche de sortie, les ajustements des poids répétés par ces facteurs différentiels décroissants rendent le gradient de plus en plus petit. Ainsi, avec plus de couches dans les réseaux, les performances du processus d'entraînement deviennent saturées et diminuent rapidement. Un réseau Res-Net résout ce problème en ajoutant des blocs résiduels qui permettent de transmettre les entrées d'une couche directement vers la sortie de cette même couche. Les liens transmis directement de l'entrée vers la sortie de tels blocs résiduels se verront attribués des poids unitaire. Dans ce type d'architecture, lorsque la rétropropagation est effectuée, le gradient sera pondéré par les poids des liens dits résiduels qui sont de 1. Cette approche préserve alors les entrées et évite toute perte d'information.

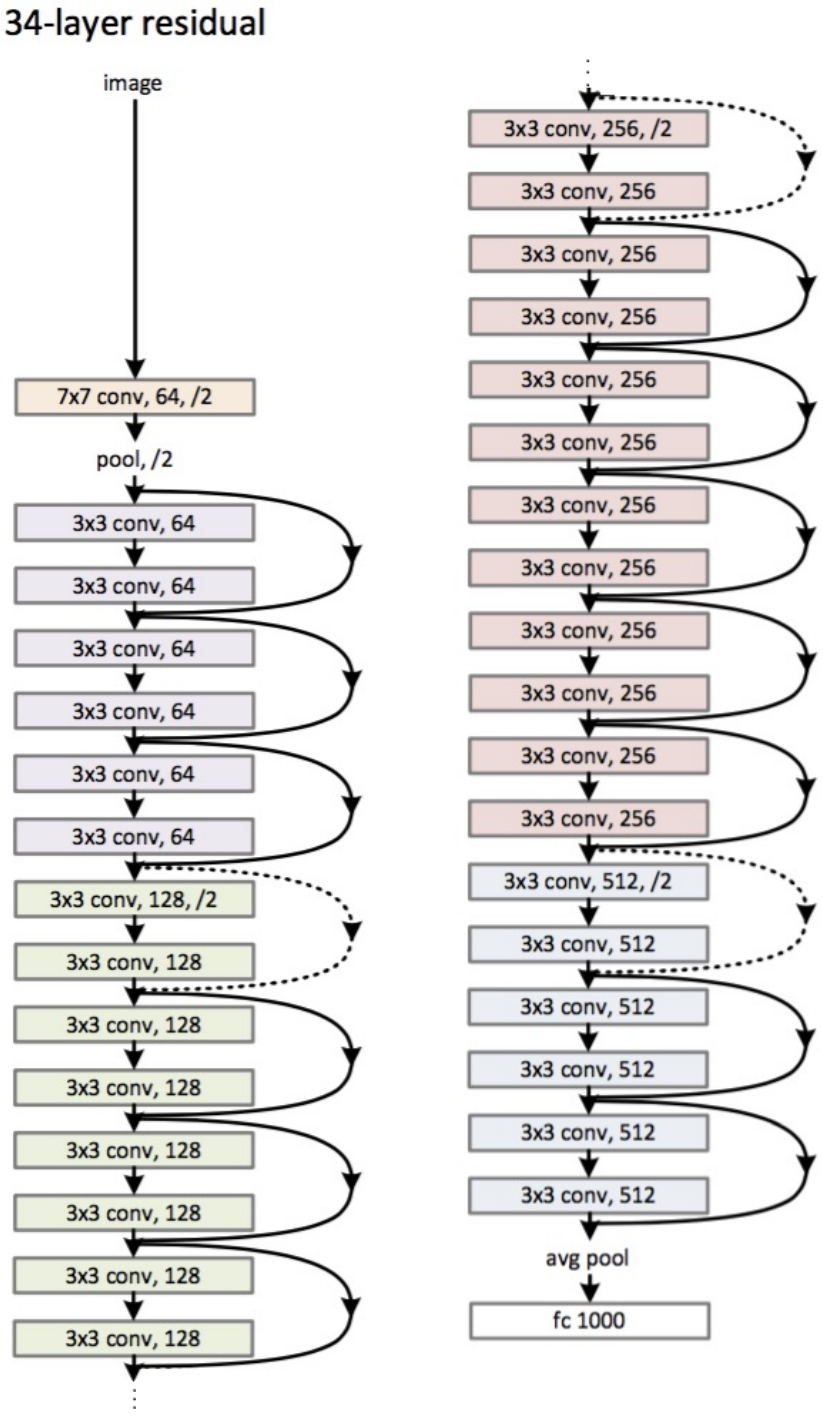

FIGURE 15 – Res-Net34 Architecture
Source : (`https://www.analyticsvidhya.com/`)

L'architecture ResNet-34 se compose de 34 couches, qui sont organisées en plusieurs étapes. La première étape effectue une série d'opérations de convolution et de regroupement maximum pour réduire les dimensions spatiales de l'entrée. Les étapes suivantes sont constituées de blocs résiduels, chacun contenant plusieurs couches convolutives avec des fonctions de normalisation par lots et d'activation ReLU. L'architecture ResNet-34 (voir Figure 15) se résume ainsi :

1. Couche d'entrée : l'entrée du réseau est une image RVB de taille fixe.
2. Couche convolutive : La première couche convolutive possède 64 filtres de taille 7x7 avec un saut de 2 et un remplissage des bordures valide.
3. Couche de mise en commun maximale : une couche de mise en commun maximale de taille 3x3 avec un saut de 2, est appliquée après la première couche convolutive.
4. Blocs résiduels : les couches restantes du réseau sont organisées en plusieurs étapes, chacune contenant plusieurs blocs résiduels. Chaque bloc résiduel se compose de deux couches convolutives avec des filtres 3x3, une normalisation par lots et des fonctions d'activation ReLU. La sortie de la deuxième couche convolutive est ajoutée à l'entrée du bloc, et le résultat est passé par une autre fonction d'activation ReLU. Dans les réseaux neuronaux traditionnels, chaque couche alimente la couche suivante. Mais dans un réseau avec des blocs résiduels, chaque couche alimente la couche suivante avec à des intervalles de couches données des entrées connectées directement avec des couches de niveaux plus profonds (voir Figure 16)

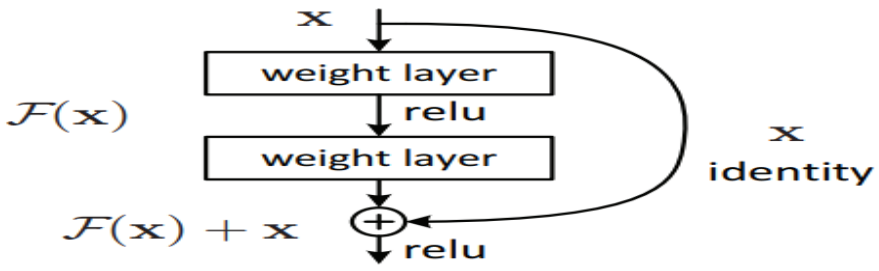

FIGURE 16 – Bloc résiduel

5. Couche de regroupement de moyenne globale : une couche de regroupement de moyenne globale est appliquée à la sortie du dernier bloc résiduel pour générer un vecteur de caractéristiques.
6. Couche entièrement connectée : le vecteur de caractéristiques est ensuite passé à travers une couche entièrement connectée avec une fonction d'activation Softmax pour générer la sortie finale.

Maintenant, nous pouvons modéliser notre propre modèle de réseau Res-Net pour segmenter et classifier des images de cultures de bleuets pour ainsi détecter la présence de mauvaises herbes. Ce modèle est créé avec la librairie Fastai en utilisant ce qu'on appelle un « apprenant » (learner). Nous pouvons alors concevoir un apprenant de style convnet en utilisant le module unet_learner définit dans le module Vision.learner pour obtenir rapidement un modèle adapté à l'apprentissage par transfert. En bref, c'est un concept dans lequel nous entraînons notre modèle à l'aide d'un modèle prédéfini / préentraîné. Dans cette recherche, nous avons utilisé le modèle ResNet34 dans le cadre de l'apprentissage par transfert.

```
learn1 = unet_learner(dls1, resnet34, metrics=[partial(accuracy, axis=1)])
```

De plus, nous devons spécifier certaines métriques qui permettent d'évaluer les performances du modèle entraîné. Ces métriques de performances sont imprimées pendant l'entraînement permettant ainsi de suivre les performances du réseau entraîné. Les métriques utilisées dans cette étude sont : l'exactitude et la perte d'entropie croisée.

3.7 Choix du taux d'apprentissage

Avec notre modèle défini, nous pouvons maintenant spécifier le taux d'apprentissage à l'aide d'un outil de recherche de taux d'apprentissage réalisé en utilisant la méthode .lr_find() afin de sélectionner optimalement un taux d'apprentissage permettant au modèle d'apprendre efficacement.

3.8 Évaluation des performances

L'évaluation des performances de la classification du réseau entraîné au préalable sera basée sur les métriques : exactitude et la fonction de perte (loss). Nous discuterons de ces mesures plus en détails dans le chapitre 4 consacré aux résultats et discussion.

3.9 Les outils de développement utilisés

3.9.1 Outils et technologies

Il existe différentes techniques pour effectuer du machine learning, mais elles nécessitent toutes des ressources matérielles relativement importantes, en fonction des données à traiter et du contexte de recherche. Dans notre cas, les images constituent la base d'un traitement gourmand en ressources matérielles, ce qui signifie qu'une machine suffisamment puissante est nécessaire pour les expérimentations.

3.9.1.1 Google Colaboratory

Google Colaboratory abrégé sous Google Colab, est un environnement de bloc-notes Jupyter basé sur le cloud qui permet aux utilisateurs d'exécuter du code Python et d'effectuer des tâches d'analyse de données dans un environnement collaboratif en ligne. Colab offre un accès gratuit aux ressources informatiques, y compris le CPU, le GPU et le TPU, ce qui en fait un choix populaire pour les tâches d'apprentissage automatique. L'un des principaux avantages de Colab est qu'il fournit un environnement préconfiguré avec de nombreuses bibliothèques et frameworks de science des données populaires, notamment TensorFlow, PyTorch et scikit-learn. Cela signifie que les utilisateurs peuvent démarrer rapidement avec des tâches d'apprentissage automatique, sans avoir à passer du temps à configurer leur propre environnement de développement. Les blocs-notes Colab peuvent être facilement partagés avec d'autres et peuvent être utilisés pour l'analyse et la recherche collaborative de données. Plusieurs utilisateurs peuvent travailler simultanément sur le même bloc-notes, ce qui facilite le partage d'idées, la collaboration sur le code et la reproduction d'expériences. Une autre caractéristique clé de Colab est son intégration à Google Drive, qui permet aux utilisateurs

de stocker et d'accéder aux données et au code directement depuis leur compte Google Drive. Cela facilite la gestion des données et la collaboration avec les autres, en particulier pour les équipes travaillant sur des projets partagés. Colab prend également en charge diverses autres fonctionnalités, telles que le contrôle de version, les extraits de code et la possibilité d'exécuter des commandes shell, ce qui en fait un outil puissant pour les tâches de science des données et d'apprentissage automatique.

3.9.2 Le choix du langage de programmation

Le choix du langage python pour nos expérimentations repose sur plusieurs raisons. Ces raisons justifiant ce choix sont nombreuses telles que ce langage offre d'abord une grande flexibilité, une communauté riche fournissant une source de documentations non négligeable, et s'intégrant bien avec un large éventail de bibliothèques de recherche et de traitement d'images, dont nous ferons la présentation dans la section suivante.

3.9.3 Librairies et Frameworks

Il existe de nombreuses bibliothèques et frameworks dans les domaines de l'apprentissage automatique et de l'imagerie. Nous décrivons quelques-unes des principales librairies que nous avons utilisés dans notre travail de recherche.

3.9.3.1 Numpy

NumPy est une bibliothèque Python pour le calcul numérique qui prend en charge les grands tableaux et matrices multidimensionnels, ainsi qu'un large éventail de fonctions mathématiques pour opérer sur ces tableaux et matrices. Celle-ci est construite autour d'une puissante structure de données appelée ndarray (tableau à N dimensions), qui est une collection de valeurs du même type de données, organisées en une grille de dimensions spécifiées par un tuple d'entiers. NumPy fournit aussi un large éventail de fonctions pour créer, manipuler et effectuer des opérations sur des ndarrays, y compris des fonctions mathématiques de base, des opérations d'algèbre linéaire, de transformées de Fourier, etc. L'un des principaux avantages de NumPy est sa rapidité et son efficacité, qui proviennent de sa capacité à effectuer des opérations mathématiques complexes sur de grands tableaux de manière vectorisée, sans avoir

besoin de boucles explicites. NumPy est également hautement interopérable avec d'autres bibliothèques de calcul scientifique, telles que SciPy, Pandas et matplotlib, ce qui en fait une partie intégrante de l'écosystème de calcul scientifique en Python (voir Figure 17)

```
>>> import numpy as np
>>> a = np.array([2, 3, 4])
>>> a
array([2, 3, 4])
>>> a.dtype
dtype('int64')
>>> b = np.array([1.2, 3.5, 5.1])
>>> b.dtype
dtype('float64')
```

FIGURE 17 – Exemple d'utilisation de Numpy
Source : (`https:/Numpy.org/`)

3.9.3.2 Matplotlib

Matplotlib est une bibliothèque Python permettant de créer des visualisations statiques, animées et interactives en Python. C'est un outil puissant et flexible pour créer des visualisations de haute qualité, et il est largement utilisé dans les communautés scientifiques et d'ingénierie pour l'exploration de données, l'analyse de données et la présentation de données. Matplotlib fournit une gamme d'outils pour créer des graphiques, des histogrammes, des nuages de points, des graphiques à barres et d'autres types de visualisations. Il offre également une gamme d'options de personnalisation pour contrôler l'apparence des tracés, y compris les styles de ligne, les marqueurs, les couleurs et les étiquettes. Matplotlib peut aussi créer des figures de qualité publication dans un large éventail de formats, notamment PDF, SVG, PNG, etc. (voir Figure 18).

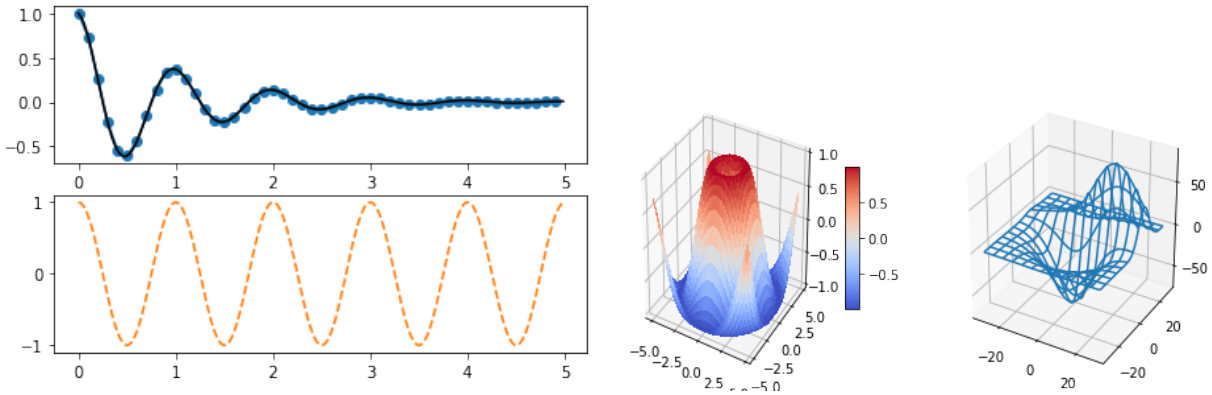

FIGURE 18 – Exemples de schémas avec matplotlib
Source : (`https:/matplotlib.org/`)

3.9.3.3 Pytorch

PyTorch est un framework d'apprentissage automatique open source largement utilisé pour modéliser, entraîner et expérimenter des réseaux de neurones. Développé par le groupe de recherche sur l'IA de Facebook, ce framework est connu pour son graphe de calcul dynamique, qui permet un calcul flexible et efficace des réseaux de neurones. Il fournit une API simple et intuitive pour la modélisation et l'entraînement de réseaux de neurones, et est devenu très populaire dans la communauté de l'apprentissage en profondeur en raison de sa facilité d'utilisation, de sa flexibilité et de sa rapidité. Il prend en charge un large éventail d'opérations et d'architectures et fournit une vaste collection de modèles préentraînés qui peuvent être facilement personnalisés pour diverses applications. PyTorch est écrit en Python et fournit des interfaces pour d'autres langages populaires tels que C++, Java et MATLAB. Il dispose également d'une communauté de développeurs importante et active, qui fournit une assistance, de la documentation et une multitude de bibliothèques et d'outils tiers. PyTorch est très souvent comparé à d'autres frameworks d'apprentissage en profondeur populaires comme TensorFlow et Keras, chacun de ces frameworks ayant ses propres forces et faiblesses. Cependant, le graphe de calcul dynamique et la flexibilité de PyTorch en ont fait un choix populaire parmi les chercheurs et les développeurs pour la modélisation et l'entraînement de modèles d'apprentissage en profondeur de pointe.

3.9.3.4 Torchvision

Torchvision est une bibliothèque Python open source qui fait partie du framework PyTorch. Cette bibliothèque fournit un ensemble d'outils et d'utilitaires pour les tâches de vision par ordinateur. Elle est conçue pour fournir une intégration transparente entre PyTorch et les ensembles de données, modèles et transformations de vision par ordinateur courants. L'une des principales caractéristiques de Torchvision est sa prise en charge de nombreux ensembles de données de vision par ordinateur populaires, tels que CIFAR-10, CIFAR-100, MNIST et ImageNet, qui sont couramment utilisés comme références par les modèles d'apprentissage automatique. Torchvision fournit une API simple et efficace pour charger et prétraiter ces ensembles de données, ce qui facilite leur intégration dans les flux de travail d'apprentissage automatique. Torchvision fournit également une gamme de modèles préentraînés pour une variété de tâches de vision par ordinateur, telles que la détection d'objets, la segmentation sémantique et la classification d'images. Ces modèles sont entraînés sur de grands ensembles de données et peuvent être affinés ou utilisés comme extracteurs de caractéristiques pour l'apprentissage par transfert. En plus des ensembles de données et des modèles, Torchvision fournit également une gamme de fonctions de transformation d'image, telles que le recadrage, le redimensionnement et la normalisation, qui peuvent être utilisées pour prétraiter les données d'image à utiliser avec des modèles d'apprentissage automatique. Elle prend également en charge les techniques d'augmentation des données telles que le recadrage aléatoire, le retournement et la rotation, qui peuvent être utilisées pour augmenter la diversité des données d'apprentissage et ainsi améliorer les performances d'un modèle. Cette bibliothèque est conçue pour être flexible et extensible, et fournit une gamme d'outils et d'utilitaires pour personnaliser et étendre ses fonctionnalités.

3.9.3.5 Fastai

Fastai est une bibliothèque Python open source pour l'apprentissage en profondeur qui repose sur pyTorch. Elle est conçue pour rendre l'apprentissage en profondeur plus accessible aux praticiens ayant différents niveaux d'expertise, en fournissant des abstractions de haut niveau, des API intuitives et des modèles et ensembles de données prédéfinis. Cette bibliothèque

fournit une large gamme de fonctionnalités pour travailler avec des modèles d'apprentissage en profondeur, y compris des modules pour la classification d'images, la détection d'objets, le traitement du langage naturel et l'analyse de données tabulaires. Il fournit également des outils pour concevoir des modèles, visualiser les résultats et déployer des modèles en production. L'une des principales caractéristiques de Fastai est son approche de l'apprentissage par transfert, qui consiste à prendre un modèle préentraîné et à l'affiner pour une tâche spécifique. Fastai fournit une gamme de modèles préentraînés qui ont été modélisés sur de grands ensembles de données et peuvent être facilement adaptés à de nouvelles tâches avec quelques lignes de code. Cela permet aux praticiens de se lancer facilement dans l'apprentissage en profondeur, même s'ils ont une expérience limitée dans le domaine. Un autre aspect important de Fastai est son accent sur la recherche. La bibliothèque est conçue pour être hautement extensible, avec une architecture modulaire qui permet aux développeurs d'ajouter de nouvelles fonctionnalités et de personnaliser les modules existants. Fastai est également activement utilisée dans la recherche, avec une large communauté de développeurs contribuant à la bibliothèque et publiant des articles basés sur leur travail avec le framework. Plusieurs raisons pour lesquelles nous avons utilisé cette bibliothèque sont :

1. **API de haut niveau facile à utiliser** : la bibliothèque Fastai fournit une API de haut niveau qui simplifie les tâches courantes d'apprentissage en profondeur, telles que la classification d'images, le traitement du langage naturel et l'analyse des données tabulaires. La bibliothèque automatise de nombreuses tâches fastidieuses impliquées dans la mise en place et la formation de modèles d'apprentissage en profondeur, tels que la préparation des données, la sélection des modèles et le réglage des hyperparamètres.
2. **Modèles de pointe :** la bibliothèque Fastai comprend des modèles pré-formés de pointe, tels que Resnet utilisé dans cette recherche, EfficientNet et Bert, qui peuvent être facilement affinés pour des tâches spécifiques comme le traitement du langage naturel.
3. **Recherche de pointe :** La bibliothèque Fastai est développée par une équipe de chercheurs et d'ingénieurs qui travaillent activement à faire progresser le domaine de

l'apprentissage en profondeur. La bibliothèque comprend bon nombre des dernières percées de recherche dans le domaine, telles que l'architecture du transformateur pour le traitement du langage naturel et l'architecture U-Net pour la segmentation des images.

4. **Expérimentation rapide :** la bibliothèque Fastai fournit un cadre flexible et personnalisable qui permet une expérimentation rapide et un prototypage. La bibliothèque prend en charge un large éventail de techniques d'apprentissage en profondeur, telles que l'apprentissage par transfert, l'augmentation des données et la formation de précision mixte, qui peut être facilement intégrée dans les modèles.
5. **Support communautaire :** La bibliothèque Fastai dispose d'une grande communauté active d'utilisateurs et de développeurs qui contribuent au développement de la bibliothèque et fournissent un soutien et des conseils aux nouveaux utilisateurs. La communauté comprend des forums, des blogs et des réseaux sociaux où les utilisateurs peuvent poser des questions, partager des connaissances et collaborer sur des projets.

La figure 19 montre le code de chargement du modèle entraîné ResNet34 avec Fastai avec l'utilisation de la bibliothèque **Fastai** :

```
from fastai.vision.models import resnet34
from fastai.vision import *
import matplotlib.pyplot as plt
from fastai.vision.models import resnet34

# Load the ResNet34 architecture
model = resnet34()

# Display the architecture summary
print(model)
```

FIGURE 19 – Code de chargement du modèle entraîné ResNet34 avec Fastai

3.9.3.6 Napari

Napari est une bibliothèque Python open source permettant de visualiser et d'explorer de grandes données d'image multidimensionnelles en 2D et 3D. Cette bibliothèque fournit une interface simple et intuitive pour la visualisation interactive des données et est conçue pour

être facile à utiliser par les experts et les non-experts en analyse d'images. L'une des principales caractéristiques de Napari est sa capacité à gérer des ensembles de données d'images volumineux et complexes, tels que des images de microscopie multicanaux, des données médicales volumétriques et des données de séries chronologiques. Elle fournit une gamme d'outils pour explorer et manipuler ces types de données, y compris le zoom, le panoramique, le découpage et le seuillage. Napari fournit également une gamme de plugins et d'extensions qui permettent aux utilisateurs d'étendre les fonctionnalités de la bibliothèque. Ces plugins incluent des outils d'analyse de données, d'apprentissage automatique et d'annotation, ainsi que l'intégration avec d'autres bibliothèques de science des données populaires telles que NumPy, SciPy et scikit-image. Cette bibliothèque prend également en charge une gamme de formats de données, notamment TIFF, HDF5 et DICOM, ce qui facilite l'importation et l'utilisation de différents types de données. Elle fournit également une gamme d'options pour exporter des visualisations et des données, y compris l'enregistrement d'images, de films et de modèles 3D.

Plusieurs raisons nous a poussé à utiliser cette bibliothèque dans notre recherche :

1. **Interactif et convivial :** Napari Annotation fournit une interface interactive et conviviale pour l'anotation des images. Il permet aux utilisateurs de créer, d'éditer et de supprimer des annotations à l'aide d'une variété d'outils, tels que des pinceaux, des polygones, des rectangles, etc. (Voir Figure 20)
2. **Flexibilité :** l'annotation Napari prend en charge une variété de types d'annotation, y compris des points, des lignes, des polygones, des rectangles et des étiquettes. Il permet également aux utilisateurs de personnaliser l'apparence visuelle des annotations, telles que la couleur, l'épaisseur de la ligne et la transparence.
3. **L'intégration avec l'écosystème scientifique Python :** l'annotation Napari est conçue pour s'intégrer de manière transparente à l'écosystème scientifique du Python, notamment Numpy, Scikit-Image, Dask, etc. Cela facilite le chargement, la manipulation et l'analyse des ensembles de données importants et complexes.
4. **Exportation d'annotations :** l'annotation Napari permet aux utilisateurs d'expor-

ter des annotations dans une variété de formats, tels que JSON, CSV et TIFF. Cela permet de partager facilement des annotations avec d'autres et de les utiliser dans d'autres outils et applications.

5. **Open-source et axée sur la communauté :** Napari Annotation est un projet open source, ce qui signifie qu'il est disponible gratuitement et peut être modifié et étendu par la communauté. Il a une communauté active de développeurs et d'utilisateurs qui contribuent à son développement et fournissent un soutien aux autres.

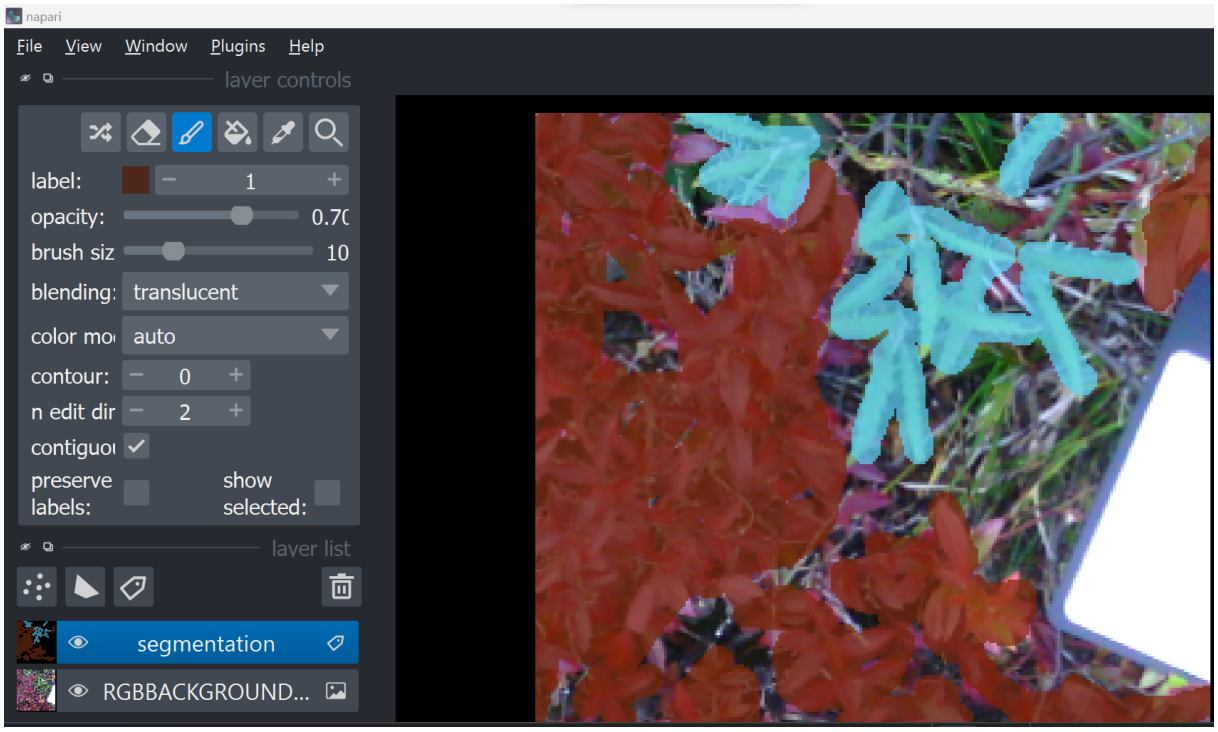

FIGURE 20 – Interface de l'outil Napari utilisée dans le contexte de cette recherche pour l'annotation d'images : Annotation du bleuet nain en rouge et la Comptonie voyageuse en bleu

3.9.3.7 Albumentations

Albumentations est une bibliothèque Python open source pour l'augmentation d'image, qui est une technique couramment utilisée dans les tâches d'apprentissage automatique et de vision par ordinateur pour générer des données de formation supplémentaires en appliquant des transformations aux images existantes. L'augmentation d'image peut améliorer les performances d'un modèle en augmentant la diversité des données d'apprentissage, ce qui rend

le modèle plus robuste à différents types d'entrées. Sa principale caractéristique est sa rapidité et son efficacité. Cette bibliothèque est construite au-dessus de la bibliothèque OpenCV, qui est optimisée pour le traitement d'images hautes performances, et peut être utilisée pour augmenter rapidement et efficacement de grands ensembles de données et fournit une large gamme de fonctions de transformation d'image, y compris la rotation, le recadrage, la mise à l'échelle (scaling), le retournement, la distorsion, l'instabilité des couleurs, etc. Albumentations est conçue pour être flexible et personnalisable, avec une API simple et intuitive qui permet aux utilisateurs d'appliquer facilement une variété d'augmentations d'image à leurs données. Cette bibliothèque prend aussi en charge une large gamme de formats d'entrée et de sortie, y compris les tableaux NumPy, les images PIL et les images OpenCV, et peut être facilement intégrée dans les workflows d'apprentissage automatique existants. Une autre caractéristique importante d'Albumentations est sa prise en charge de l'accélération CPU et GPU, ce qui permet d'augmenter de grands ensembles de données en parallèle sur des CPU ou GPU multicœurs. Cela peut réduire considérablement le temps nécessaire pour générer des données augmentées et permet d'expérimenter plus rapidement différentes stratégies d'augmentation et d'hyperparamétrages. Plusieurs raisons pour lesquelles nous avons utilisé cette bibliothèque pour l'augmentation de l'image sont :

1. **Variété de transformations :** Albumentations fournit une large gamme de fonctions de transformation d'image, y compris des rotations aléatoires, des flips, des translations, des changements d'échelle et des manipulations de couleurs. Ces fonctions peuvent être combinées pour créer des augmentations plus complexes qui aident à créer diverses données de formation qui peuvent ensuite permettre d'entraîner un réseau de neurones.
2. **Performance :** Albumentations est conçue pour être rapide et efficace, ce qui rend cette bibliothèque bien adaptée aux grands ensembles de données et aux applications en temps réel.
3. **Flexibilité :** Albumentations est hautement configurable et peut être utilisée avec une variété de cadres d'apprentissage en profondeur, notamment Pytorch, Tensorflow et Keras. Elle permet également aux utilisateurs de définir des fonctions d'augmentation

personnalisées.

4. **Open-source :** Albumentations est une bibliothèque open source, ce qui signifie qu'elle est disponible librement pour une utilisation et peut être modifiée et étendue par la communauté.

5. **À la pointe de la technologie :** Albumentations est basée sur des techniques de traitement d'image de pointe et est constamment mise à jour avec les derniers résultats de recherche. Cela garantit qu'elle fournit des augmentations de haute qualité qui sont efficaces pour améliorer les performances d'un modèle d'apprentissage en profondeur.

La figure 21 montre le code pour faire les augmentations dans le cadre de cette recherche. Ces augmentations sont faites image par image vu que notre banque de données initiale ne contenait que 33 images.

```
from PIL import Image
import os
import shutil
import cv2
import matplotlib.pyplot as plt
import albumentations as A
transform = A.Compose([
    A.RandomCrop(width=128, height=128),
    A.HorizontalFlip(p=0.5),
    A.RandomBrightnessContrast(p=0.2),
    A.VerticalFlip(p=0.5) ,
    A.ShiftScaleRotate(shift_limit=0.0625,scale_limit=0.1,rotate_limit=45,p=0.5),
    ])
image_path = "C:\\Users\\basey\\Pictures\\data\\train\\images\\"
mask_path = "C:\\Users\\basey\\Pictures\\data\\train\\masks\\"


im =  cv2.imread(r"C:\Users\basey\Pictures\data\dataset\train\RGBBACKGROUND_2020-09-03_005-BleuetRouge-Comptonie.png")
im =  cv2.cvtColor(im, cv2.COLOR_BGR2RGB)
mas = cv2.imread(r"C:\Users\basey\Pictures\data\dataset\train_masks\RGBBACKGROUND_2020-09-03_005-BleuetRouge-Comptonie.png")


for i in range(2):
    transformed = transform(image=im ,mask= mas)
    transformed_image = transformed['image']
    transformed_mask = transformed['mask']
    fig, ax = plt.subplots(figsize=(8, 8),  nrows=1, ncols=2)
    ax[0].imshow(transformed_image)
    ax[1].imshow(transformed_mask[:,:,0])

for j,image in enumerate(image_list):
         Image.fromarray(image).save(image_path +  f"RGBBACKGROUND_2020-09-03_005-BleuetRouge-Comptonie_{j}.png")

for k,image in enumerate(mask_list):
        Image.fromarray(image).save(mask_path +  f"RGBBACKGROUND_2020-09-03_005-BleuetRouge-Comptonie_{k}_mask.png")
```

FIGURE 21 – Code des augmentations de deux images prises comme exemple avec la bibliothèque avec Albumentation

Et la figure 22 montre les 2 images augmentées obtenues avec cette bibliothèque en utilisant le code présenté à la figure 21 :

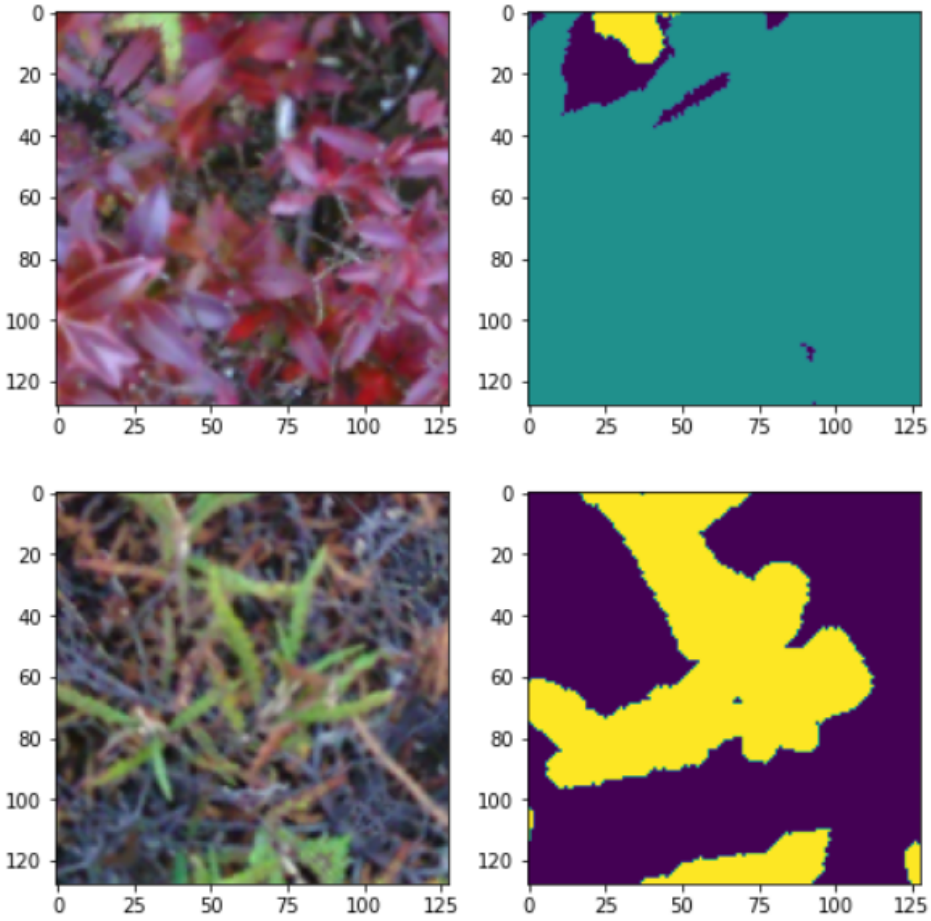

FIGURE 22 – Images augmentées obtenues (images gauches) avec la bibliothèque Albumentation

3.10 Conclusion

Dans ce chapitre, nous avons présenté la méthodologie utilisée pour permettre la détection automatique de mauvaises herbes comme le Kalmia à feuilles étroites dans les cultures de bleuet nain. Nous avons d'abord introduit les méthodes de prétraitement du corpus de données, l'organisation de ces données et les traitements nécessaires pour les exploiter à des fins expérimentales. Ensuite, nous avons décrit l'algorithme de segmentation et de reconnaissance basé sur une approche neuronale comme ResNet et UNET. Dans le chapitre suivant, nous présentons nos résultats en nous concentrant sur les différents outils de mesures qui permettent de savoir à quel point le modèle est performant pour segmenter/détecter les mauvaises herbes dans des images.

Chapitre 4 Résultats

Dans ce présent chapitre, nous allons présenter les résultats obtenus après de multiples expérimentations. Les résultats diffèrent selon le modèle utilisé, le prétraitement effectué et aussi toutes autres manipulations ayant précédé l'apprentissage du réseau de neurones dédié à la détection automatique de mauvaises herbes dans les champs de bleuets sauvages. Dans ce qui suit, nous verrons les résultats de détection/classification obtenus, leurs interprétations ainsi que les moyens mis en place permettant de les améliorer. Ensuite, nous discuterons des potentielles améliorations et des perspectives qui peuvent être envisagées.

4.1. Les métriques de performances

Maintenant, nous pouvons définir notre modèle. Cette opération se fait à l'aide du module wrapper unet_learner de Fastai, qui spécifie de façon très générale que nous visons à utiliser un réseau neuronal convolutif. Cette fonctionnalité reçoit ensuite une référence sur les données devant servir à l'entraînement et la validation du modèle de réseau de neurones convolutif choisi, le type de modèle neuronal utilisé et les métriques de performance. Ces informations qui sont spécifiées en entrée comme paramètres. Comme présenté à la figure 23, nous passons comme référence notre Data Loader (chargeur de données) en paramètre à la méthode unet_learner. Nous spécifions également qu'une architecture resnet34 pré-entraînée sera chargée et utilisée pour faire l'apprentissage pas transfert et ensuite, après la phase d'entraînement, la classification des mauvaises herbes. La métrique de précision est également transmise. Il convient ici de noter qu'il s'agit d'une métrique de performance et qu'il ne faut pas la confondre avec la fonction de perte.

FIGURE 23 – Chargement du modèle pré-entraîné ResNet34

4.1.1 La fonction perte

Avec la classe unet_learner définie et instanciée en l'objet learn1, nous pouvons maintenant commencer le processus de modélisation du réseau de neurones convolutif. La première étape de ce processus est de déterminer le taux d'apprentissage avec l'aide d'un outil de recherche de taux d'apprentissage optimal réalisé en utilisant la méthode .lr_find() afin de sélectionner un taux d'apprentissage permettant au modèle neuronal d'apprendre efficacement (voir Figure 24).

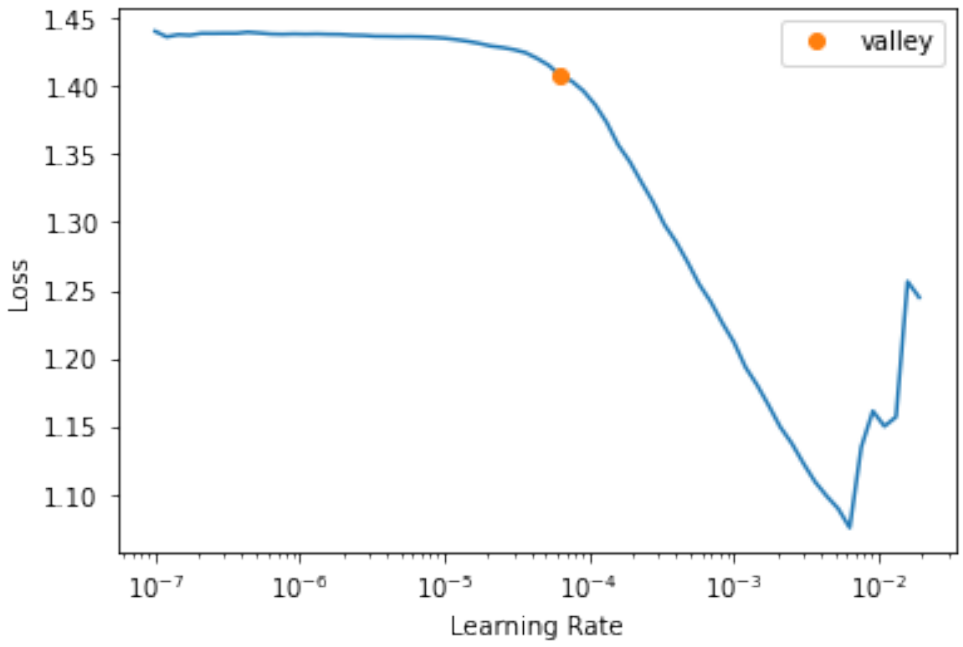

FIGURE 24 – Graphique de recherche du taux d'apprentissage

L'apparence de ce graphique de recherche du taux d'apprentissage pourrait être différente à chaque nouvelle exécution. Cependant, la valeur de perte (loss) semble augmenter rapidement quand le taux d'apprentissage est plus grand que 1e-2, valeur de perte qui est minimale proche de ce même taux d'apprentissage. Notre objectif est de sélectionner un taux d'apprentissage juste avant ce point minimal. Dans notre étude le taux d'apprentissage a été fixé à 1e-4.

Cependant, nous ne nous attendons pas à ce que chaque couche de notre réseau nécessite le même taux d'apprentissage. D'abord, le taux d'apprentissage optimal pour l'apprentissage des couches d'un réseau pré-entraîné devrait pouvoir être inférieur à celui requis pour l'en-

traînement d'un réseau sans pré-entraînement. De plus, tant pour les réseau pré-entraînés que non pré-entraînés, les couches du réseau plus en aval (vers la sortie) devraient nécessiter un taux d'apprentissage plus élevé que celles au début du réseau.

Dans cet esprit, nous pouvons utiliser des taux d'apprentissage adaptés aux couches d'un réseau de neurones selon la profondeur de chaque couche. Ce qui revient à sélectionner un taux d'apprentissage plus faible pour les couches antérieures du réseau (au début). Le taux d'apprentissage augmentant au fur et à mesure que nous nous déplaçons vers les couches en aval (sorties) du réseau neuronal. Cette approche adaptative permet finalement d'avoir un taux d'apprentissage global du réseau neuronal plus proche de son optimum absolu. Cette approche de sélection adaptative peut être réalisée en spécifiant la méthode .fit_one_cycle() slice object qui produit des valeurs équidistantes entre 1e-4 et 1e-3 (donc proche de la valeur de 1e-4 déterminée par le chercheur de taux d'apprentissage) et ajuste les taux d'apprentissage de chaque couche de manière appropriée.

En utilisant ce taux d'apprentissage, nous pouvons ensuite former la couche non figée pendant quelques époques (5 epochs) en utilisant la méthode .fit_one_cycle() afin de configurer les paramètres de la couche entièrement connectée pour notre tâche spécifique de détection/classification de mauvaises herbes. Les résultats optimaux obtenus dans notre études nous ont permis d'avoir une valeur de perte (Loss) minimale égale à 0.196 en apprentissage et de 0.268 en validation (voir Figure 25).

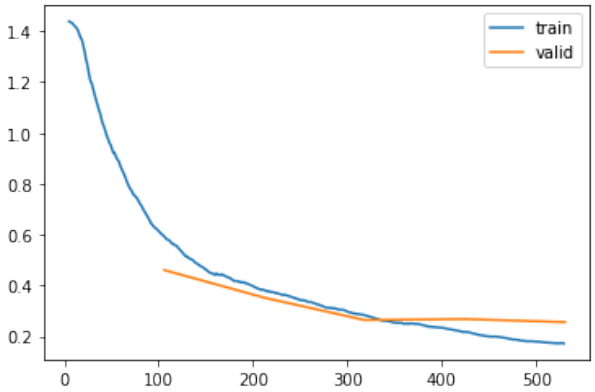

FIGURE 25 – Graphique de perte (Loss) en fonction des époques (epochs) pour la couche non figée

Sur la figure 25, nous pouvons voir que la valeur de la fonction de perte LOSS a une tendance décroissante ce qui signifie que les résultats de détection/classification tendent à s'améliorer.

Ensuite, nous avons figé les calques en utilisant la méthode .unfreeze() pendant un certain nombre époques (10) en utilisant la méthode .fit_one_cycle(). Les résultats optimaux nous ont permis d'avoir une valeur de perte (Loss) minimale égale à 0.136 en apprentissage et de 0.244 en validation (voir Figure 26).

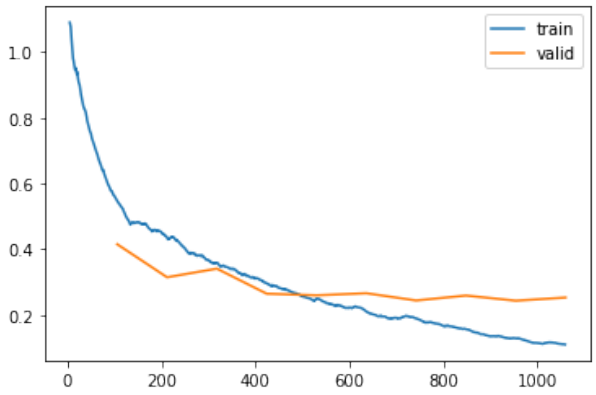

FIGURE 26 – Graphique de perte (Loss) en fonction des époques (epochs) pour la couche figée

Sur la figure 26, nous pouvons aussi voir que la valeur de la fonction de perte LOSS a une tendance décroissante, ce qui signifie que les résultats de détection/classification s'améliorent.

En somme, nous pouvons dire qu'après avoir figée notre modèle de réseau de neurones convolutif, nous constatons que la perte (Loss) en apprentissage et en validation a considérablement diminuée par rapport à celles mesurées pour le modèle non figé.

4.1.2 L'exactitude de prédiction

Cette mesure détermine à quel point les prédictions du modèle neuronal correspondent aux étiquettes représentant la vérité terrain (ground truth). D'après la documentation de Fastai, lors de la phase d'apprentissage, cette métrique structure d'abord toutes les valeurs de sortie (prédictions) sous forme d'une matrice permettant de mettre en correspondance ces prédictions avec leur masque de segmentation découlant des opérations de labellisation. En fait, cette métrique permet de faire correspondre la sortie argmax de la prédiction de chaque

pixel de l'image en entrée avec le masque de la cible de chacun de ces pixels. En comparant ces valeurs de prédiction et étiquette cible il devient facile de calculer le nombre d'équivalences pour chaque classe (types de mauvaises herbes) et de déduire ensuite les moyennes de bonnes prédictions pour chaque classe. Étant donné que nos données sont bien organisées, cette métrique est bien adaptée pour mesurer l'efficacité de la classification des prédictions. Dans ce contexte, nous avons pu atteindre une précision en validation égale à 92.33% et ce après avoir formé la couche non figée pendant quelques époques (5) en utilisant la méthode .fit_one_cycle() et 92.63% après avoir figer les calques en utilisant la méthode .unfreeze() pendant (10) époques (voir Figure 27 contenant les valeurs de perte et de précision).

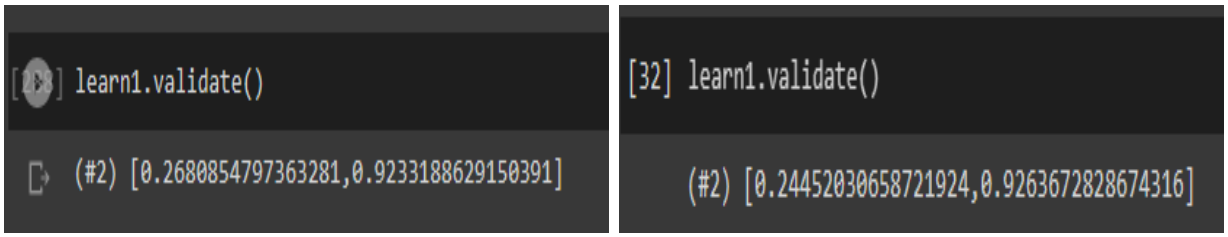

FIGURE 27 – L'exactitude des prédictions avec une couche non figée (à gauche) et l'exactitude avec une couche figée (à droite) pour la validation.

4.1.3 Inférence

Une fois la formation du modèle neuronal terminée et que notre modèle est chargé, les chargeurs de données ayant été soit recréés ou restaurés pour tenir compte des données de test à utiliser pour évaluer les performances d'inférence de notre réseau, nous sommes prêts à commencer le processus d'inférence. L'inférence est effectuée avec la méthode test_dl en utilisant les 9 images contenues dans le dossier test (voir Figure 28).

FIGURE 28 – Inférence sur les 9 images du dossier test

Maintenant, nous pouvons visualiser les résultats découlant du processus d'inférence et voir comment se comporte notre modèle sur ces données de test.

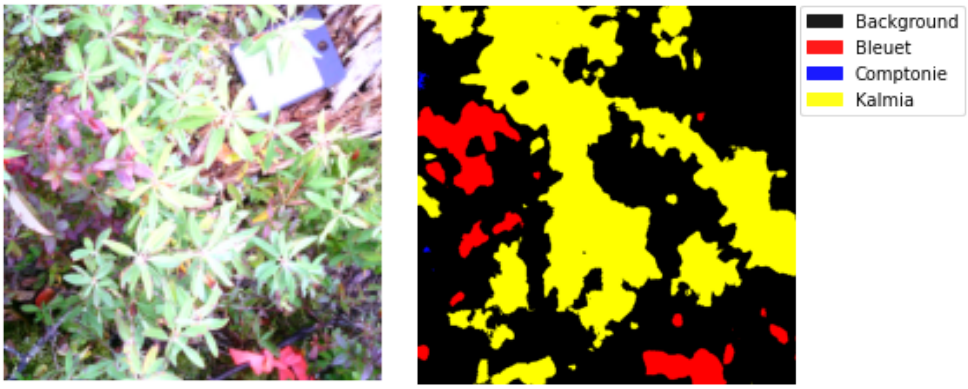

FIGURE 29 – Images contenant du bleuet et du kalmia à feuilles étroites (mauvaises herbes)

L'image segmentée dans la figure 29 montre une bonne correspondance globale avec les résultats attendus en termes d'exactitude de la segmentation. Cependant, il y a quelques éléments à prendre en compte pour expliquer ces observations.

Tout d'abord, la présence de petites taches bleues (en haut à gauche de l'image segmentée (image droite)) correspondant à la classe de la comptonie peut être expliquée par les variations introduites lors de l'augmentation des images. L'ajustement de la luminosité et du contraste peut avoir provoqué des variations visuelles subtiles qui ont été interprétées par le modèle comme étant de la comptonie. Cela démontre que le modèle est sensible à de telles variations et qu'il est important de prendre en compte la diversité des conditions d'éclairage lors des processus d'entraînement et de validation du modèle.

En ce qui concerne la confusion entre la banderole rouge (en bas à droite de l'image originale (image gauche)) et du bleuet (segmentée en rouge dans l'image segmentée (image droite)), cette confusion peut être attribuée à plusieurs facteurs. Premièrement, si la base de données d'entraînement contient un nombre limité d'exemples de bleuets avec une banderole rouge, le modèle pourrait avoir eu du mal à généraliser cette combinaison spécifique. Cela souligne l'importance d'une base de données d'entraînement bien équilibrée et représentative de toutes les variations possibles dans les images réelles.

Deuxièmement, les augmentations telles que le zoom et le contraste peuvent avoir renforcées les similitudes visuelles entre la banderole rouge (en bas de l'image originale (image gauche)) et le bleuet, ce qui a rendu difficile la distinction pour le modèle. Une augmentation plus prudente et équilibrée pourrait aider à mieux préserver les caractéristiques distinctives de chaque classe.

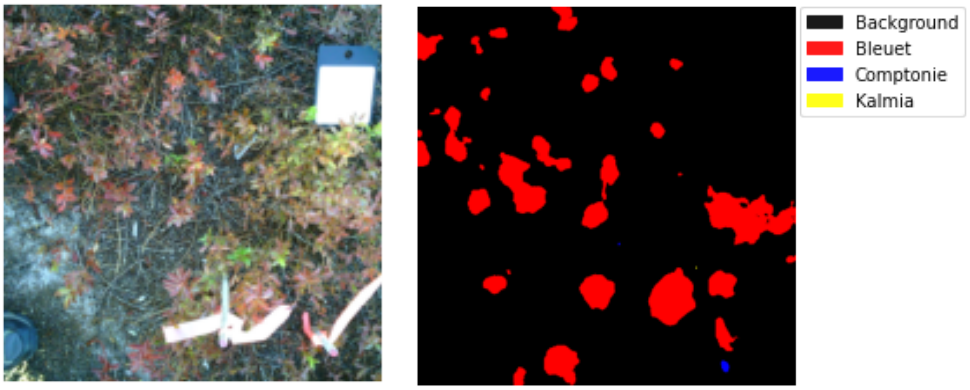

FIGURE 30 – Images contenant du bleuet rouge (feuilles rouges) et du bleuet vert (feuilles vertes)

Dans l'image segmentée (image droite) présentée dans la figure 30, les résultats de la segmentation sont cohérents avec ceux de l'exactitude globale. La présence de la tache bleue indique la présence de la classe de la comptonie (coin inférieur droit image droite), une mauvaise herbe en réalité absente dans l'image originale (image gauche). Cependant, nous constatons que cette tache bleue est le résultat d'une confusion entre la comptonie et le bleuet vert (avec feuilles vertes).

Cette confusion entre la comptonie et le bleuet vert peut s'expliquer par plusieurs facteurs. Tout d'abord, il est possible que le classificateur ait considéré certaines caractéristiques visuelles du bleuet vert comme étant similaires à celles de la comptonie. Ces similitudes peuvent inclure des aspects tels que la couleur, la texture, la forme ou d'autres attributs visuels qui se recoupent entre ces deux plantes.

Un autre facteur qui peut contribuer à l'absence de détection du bleuet vert est le manque de données d'entraînement et de validation spécifiques pour cette classe (bleuet vert). Si le modèle n'a pas été suffisamment exposé à des exemples de bleuets verts lors de l'entraînement,

il peut avoir du mal à les reconnaître et à les segmenter correctement. Par conséquent, le bleuet vert peut ne pas être détecté ou être confondu avec une autre classe, en l'occurrence la comptonie dans ce cas.

Les résultats de segmentation que nous pouvons observer dans la figure 31 sont en parfait accord avec les mesures d'exactitude globale. Les prédictions réalisées par le modèle ont permis une segmentation précise des images, notamment en ce qui concerne la distinction entre le bleuet et la comptonie. Ce succès peut être attribué à plusieurs facteurs, mais l'un des principaux est la qualité de la base d'entraînement et de validation utilisée, qui a permis au modèle d'apprendre de manière efficace les caractéristiques distinctives de chaque classe, dans ce cas précis le bleuet et la comptonie. La comptonie ayant d'ailleurs des feuilles de formes distinctives permettant au réseau de bien les identifier.

La présence d'un grand nombre d'images représentant le bleuet et la comptonie dans la base d'entraînement a joué un rôle crucial dans l'obtention de ces résultats précis. Le modèle a pu se familiariser avec les caractéristiques visuelles spécifiques de chaque classe et les utiliser pour effectuer une segmentation précise lors de la phase de test. Cette distinction claire entre les images des différentes classes s'est maintenue lors du processus de validation, ce qui a également contribué à la précision des résultats.

Il est important de souligner que l'exactitude des prédictions dépend de la qualité et de la diversité des données d'entraînement. Dans ce cas, la base d'entraînement a été soigneusement conçue pour capturer les variations et les caractéristiques distinctives du bleuet et de la comptonie, ce qui a permis au modèle d'apprendre à les différencier avec précision.

Ces bons résultats de segmentation pour le bleuet à terre (feuilles de bleuet tombées) et la comptonie dans l'image segmentée témoignent de l'efficacité du modèle entraîné. Ils démontrent également l'importance de constituer une base de données d'entraînement représentative et diversifiée pour permettre au modèle d'apprendre les caractéristiques spécifiques de chaque classe.

En conclusion, les résultats de segmentation obtenus sont le fruit d'un modèle bien entraîné

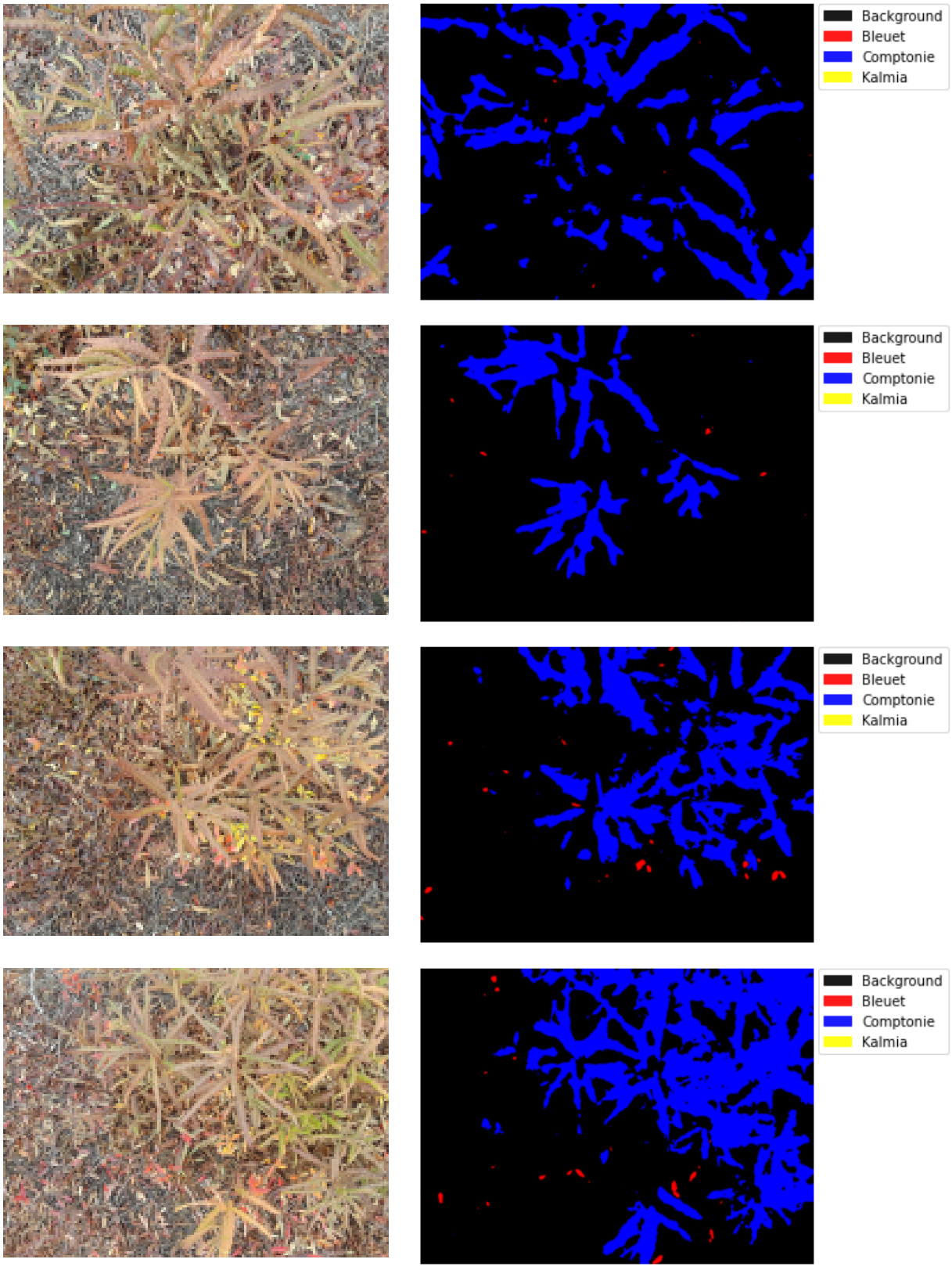

FIGURE 31 – Images contenant de la comptonie (mauvaises herbes) et du bleuet

sur une base de données de qualité. La distinction claire entre le bleuet à terre et la comptonie dans les images segmentées reflète la capacité du modèle à apprendre les caractéristiques visuelles distinctives de chaque classe.

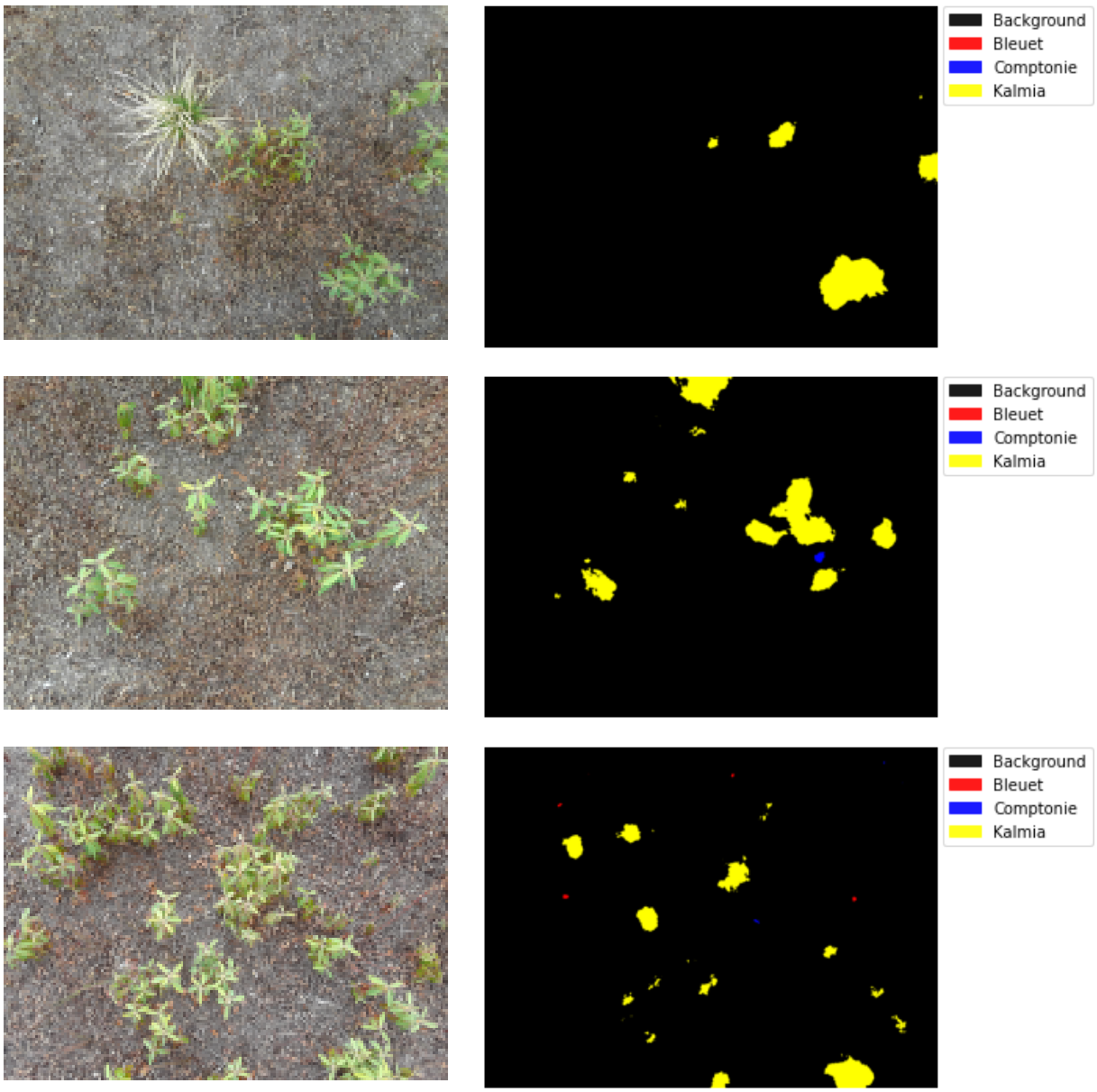

FIGURE 32 – Images contenant que du kalmia (mauvaises herbes)

Les résultats de segmentation présentés dans la figure 32 démontrent une segmentation globalement satisfaisante de nos images. Cependant, il est important de noter la présence de quelques zones où l'on peut observer à la fois de la comptonie et du bleuet (image segmentée en bas). Cette observation peut être expliquée par plusieurs facteurs. Tout d'abord, les ajustements de luminosité et de contraste effectués lors des augmentations appliquées à nos images peuvent avoir introduit des variations visuelles qui ont pu induire le classificateur en erreur. Ces variations peuvent rendre difficile la distinction précise entre la comptonie et le bleuet, conduisant à des zones où les deux classes semblent coexister dans l'image segmentée.

En ce qui concerne la non-segmentation complète du kalmia, cela peut être attribué à la difficulté d'annoter avec précision les plantes de cette taille à l'aide de l'outil Napari. Les images de kalmia peuvent contenir des détails fins ou des variations subtiles qui rendent leur annotation plus complexe. Par conséquent, cela peut avoir limité la capacité du réseau à détecter et à segmenter le kalmia de manière exhaustive.

Pour améliorer cette situation, il serait bénéfique d'explorer différentes approches. Cela pourrait inclure l'optimisation des paramètres d'augmentation afin de mieux reproduire les variations réelles présentes dans les images. Il serait également utile d'envisager d'acquérir davantage de données d'entraînement représentatives du kalmia, en mettant l'accent sur la diversité des exemples pour mieux capturer les variations visuelles de cette classe. De plus, l'utilisation d'outils d'annotation plus adaptés à la taille des plantes, ainsi que l'exploration de techniques de segmentation plus avancées, pourraient contribuer à améliorer la précision et l'exhaustivité de la segmentation du kalmia.

En résumé, bien que les résultats de segmentation dans la figure 32 montrent une segmentation globalement assez satisfaisante, la présence de comptonie et de bleuet dans certaines zones de l'image segmentée soulève certaines interrogations. Les ajustements de luminosité et de contraste ainsi que la taille des plantes de kalmia peuvent influencer ces résultats. En tenant compte de ces facteurs et en explorant des approches d'amélioration spécifiques, il serait alors possible d'améliorer la précision et la fiabilité de la segmentation des différentes classes d'objets.

En conclusion, le modèle ResNet34 que nous avons utilisé à montrer des performances très encourageantes sur notre ensemble de données. Avec une exactitude de 92,63% sur les données de validation et des valeurs de perte de 0,244 en validation et 0,136 en apprentissage, le modèle a atteint une précision élevée et a réussi à minimiser l'erreur lors de l'inférence (phase de validation).

Les résultats de l'inférence étaient cohérents avec l'exactitude du modèle, ce qui indique une capacité de généralisation solide. De plus, la diminution de la perte au fil des itérations d'apprentissage témoigne de l'amélioration progressive des performances du modèle.

Ces résultats témoignent de la capacité du modèle à apprendre et à capturer les caractéristiques distinctives des différentes classes présentes dans notre ensemble de données. Le haut niveau d'exactitude sur les données de validation indique que le modèle est capable de prendre des décisions précises sur de nouvelles données. Il est important de noter que l'exactitude et la perte ne sont que des mesures globales de performance et qu'il est essentiel de considérer d'autres métriques et d'évaluer le modèle sur différents ensembles de données pour obtenir une évaluation plus complète.

4.2 Discussion

Les résultats de notre étude ont montré une variabilité considérable, mettant en évidence les défis et les possibilités d'optimisation dans le domaine de l'apprentissage automatique. Nous avons réalisé des progrès significatifs en améliorant les performances de notre modèle, mais nous reconnaissons qu'il existe encore des limites à prendre en compte.

L'une des principales limites de notre étude réside dans les données que nous avons utilisées. Le nombre limité d'images disponibles et leur qualité relativement basse ont présenté des défis pour la classification précise. Obtenir un ensemble de données de grande taille et de haute qualité est essentiel pour obtenir des performances optimales. Il est important d'investir du temps et des efforts dans l'organisation et le prétraitement des données afin de garantir une base de données solide pour l'apprentissage.

Nous avons également rencontré des difficultés dans la tâche de segmentation. La présence d'autres classes dans certaines images, comme du kalmia ou de la comptonie dans une image de bleuet, a posé des défis supplémentaires. De plus, les augmentations appliquées aux images, telles que les ajustements de luminosité et de contraste, ont parfois introduit des biais et ont affecté les performances de segmentation. Il est crucial de trouver un équilibre entre l'augmentation des données pour améliorer la généralisation du modèle et la préservation des caractéristiques essentielles pour une segmentation précise.

Pour surmonter ces limites, nous estimons qu'il est nécessaire de donner un temps précieux pour l'acquisition de données de meilleure qualité, en s'assurant d'inclure une diversité d'exemples pour chaque classe d'objets. L'amélioration des techniques de prétraitement des données, y compris l'élimination du bruit et la normalisation appropriée, peut également contribuer à améliorer les performances du modèle pour obtenir des résultats plus précis.

Chapitre 5 Conclusion et perspectives

L'objectif de ce mémoire était d'explorer des solutions afin d'optimiser certaines opérations phytosanitaires permettant la détection automatique de mauvaises herbes. Cette recherche permet d'offrir des outils logiciels aux agriculteurs afin de les assister dans leurs tâches de gestion des mauvaises herbes dans leurs cultures de bleuets nains. Ces outils permettent la détection automatique précoce des mauvaises herbes, ainsi de limiter la prolifération de mauvaises herbes comme la comptonie voyageuse et le kalmia à feuilles étroites dans les champs du bleuet. Par conséquent, les cultures conventionnelles et les cultures biologiques peuvent être plus rentables et plus respectueuses de l'environnement. En effet, cette solution permet de réduire l'utilisation d'herbicide dans les cultures conventionnelles puisque la détection précoce et précise des mauvaises herbes permet des interventions (arrosage aux herbicides) mieux ciblées. Pour les cultures biologiques ce sont les coûts de main-d'œuvre pour le désherbage manuel qui sont réduits pour les mêmes raisons que dans les cultures conventionnelles.

Le principal défi auquel nous devions faire face dans cette recherche, était de pouvoir discriminer la comptonie et le kalmia, mauvaises herbes qui se répandent dans les champs de bleuets. Pour résoudre ce problème, nous avons premièrement acquis une base de données comportant des images des différentes espèces (bleuets, kalmia, comptonie). Ensuite, ces images ont subi différentes étapes de traitement, d'organisation et par la suite ces images ont été utilisées pour faire des expérimentations ultérieures nécessaires permettant la détection automatique des mauvaises herbes.

Deuxièmement, notre recherche s'est basée sur un algorithme de segmentation et de reconnaissance basée sur une approche neuronale. Cette approche expérimentée est celle des réseaux de neurones convolutifs comme Unet et ResNet. Notre traitement est basé sur des moyens techniques accessibles et disponibles à n'importe quel endroit où il y a un accès à Internet. Différentes technologies utilisées dans cette recherche telles que les librairies et le

langage de programmation python sont de libres accès et gratuites. Le traitement de ces réseaux de neurones à convolutions nécessite cependant une configuration avancée ainsi que la plateforme Google Colaboratory. Cette plateforme collaborative facilite grandement notre recherche la rendant par le fait même plus accessible. L'algorithme de segmentation et de reconnaissance présentée dans cette recherche est basé sur le modèle Unet utilisant un modèle préentraîné ResNet34 qui a prouvé son efficacité dans d'autres recherches. Par la suite, nous avons effectué des ajustements comme une augmentation sur nos images afin d'améliorer notre modèle et par conséquent obtenir des résultats très satisfaisants avec plus de 90% de précision au niveau de la détection/segmentation. des mauvaises herbes. Niveau de précision permettant d'optimiser les coûts des opérations phytosanitaires en permettant de réduire les quantités d'herbicide utilisées dans les culture conventionnelles et les coûts d'éradication manuelle dans les cultures biologiques.

Finalement, notre algorithme a prouvé avec ce niveau de précision (plus de 90%) qu'il est bien possible de segmenter nos images avec une bonne précision. Les résultats obtenus concordent bien et prouvent l'efficacité de notre démarche.

Les résultats prometteurs que nous avons obtenus dans cette recherche ouvrent des perspectives pour d'autres recherches. En effet, ils peuvent servir de bases pour d'autres recherches. Avec la possibilité d'utiliser la base de données déjà constituée pour faire d'autres expérimentations et utiliser les résultats que nous avons comme base de comparaison, ou alors en construisant d'autres bases de données avec plus d'images.

Notre étude peut également être reprise dans un système de détection et de traitement de mauvaises herbes, en implémentant par exemple la partie que nous avons expérimentée, c'est à dire un algorithme de segmentation et de reconnaissance automatique dans un système intelligent de détection et d'éradication des mauvaises herbes. Ainsi avec l'évolution de la vision par ordinateur, d'autres algorithmes de segmentation d'objets pourraient être expérimentés pour mieux détecter les différentes espèces de plantes présentes dans les images. Par exemple, l'utilisation de la plateforme Detectron2 de FaceBook et aussi le modèle SegFormer un framework de segmentation sémantique simple, efficace, mais puissant qui unifie Transformers

avec décodeurs légers de perceptrons multicouches pourraient être explorés.

Références

Vinit Bodhwani, D.P. Acharjya, and Umesh Bodhwani. Deep residual networks for plant identification. *Procedia Computer Science*, 152 :186–194, 2019. ISSN 1877-0509. doi : https://doi.org/10.1016/j.procs.2019.05.042. URL `https://www.sciencedirect.com/science/article/pii/S1877050919306945`. International Conference on Pervasive Computing Advances and Applications- PerCAA 2019.

Li Chen, Jin-Guo Zhang, Hai-Feng Su, and Wei Guo. Weed identification method based on probabilistic neural network in the corn seedlings field. In *2010 International Conference on Machine Learning and Cybernetics*, volume 3, pages 1528–1531, 2010. doi : 10.1109/ICMLC.2010.5580822.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1) :21–27, 1967. doi : 10.1109/TIT.1967.1053964. URL `https://doi.org/10.1109/TIT.1967.1053964`.

Alessandro dos Santos Ferreira, Daniel Matte Freitas, Gercina Gonçalves da Silva, Hemerson Pistori, and Marcelo Theophilo Folhes. Weed detection in soybean crops using convnets. *Computers and Electronics in Agriculture*, 143(nil) :314–324, 2017. doi : 10.1016/j.compag.2017.10.027. URL `https://doi.org/10.1016/j.compag.2017.10.027`.

Drones Imaging. Indice de végétation ndvi. `https://www.dronesimaging.com/conseils-en-photogrammetrie/`.

E Edward W. Forgy. Cluster analysis of multivariate data : efficiency versus interpretability of classifications. *Biometrics*, 21 :768, 1965.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm

for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

Evelyn Fix and Joseph L. Hodges. Discriminatory analysis - nonparametric discrimination : Consistency properties. *International Statistical Review*, 57 :238, 1989.

Sophie Gagnon. Essai et expérimentation sur la pollinisation et la réduction des herbicides dans la production du bleuet semi-cultivé au saguenay-lac-saint-jean. *Agrinova | Recherche et innovation en agriculture*, 2009.

Sophie Gagnon. La comptonie voyageuse. *Agrinova | Recherche et innovation en agriculture*, 2010a.

Sophie Gagnon. Le kalmia à feuilles étroites. *Agrinova | Recherche et innovation en agriculture*, 2010b.

A. Géron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly, Sebastopol, CA, USA, 2017.

Esmael Hamuda, Martin Glavin, and Edward Jones. A survey of image processing techniques for plant extraction and segmentation in the field. *Computers and Electronics in Agriculture*, 125(nil) :184–199, 2016. doi : 10.1016/j.compag.2016.04.024. URL `https://doi.org/10.1016/j.compag.2016.04.024`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL `https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf`.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D.

Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4) :541–551, 1989. doi : 10.1162/neco.1989.1.4.541.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998. doi : 10.1109/5.726791.

C.H. Li and C.K. Lee. Minimum cross entropy thresholding. *Pattern Recognition*, 26(4) : 617–625, 1993. ISSN 0031-3203. doi : https://doi.org/10.1016/0031-3203(93)90115-D. URL `https://www.sciencedirect.com/science/article/pii/003132039390115D`.

Ping-Sung Liao, Tse-Sheng Chen, and P. C. Chung. A fast algorithm for multilevel thresholding. *J. Inf. Sci. Eng.*, 17 :713–727, 2001.

M. B. Khan M. Mohammed and E. B. M. Bashier. *Machine Learning : Algorithms and Applications*. CRC Press, Boca Raton, FL, USA, 2017.

J MacQueen. Classification and analysis of multivariate observations. In *5th Berkeley Symp. Math. Statist. Probability*, pages 281–297. University of California Los Angeles LA USA, 1967.

M. Merzougui, M. Nasri, and Ahmad El allaoui. Isodata et les algorithmes génétiques pour une classification non supervisée. 05 2016.

George E. Meyer, Timothy W. Hindman, and Koppolu Laksmi. Machine vision detection parameters for plant species identification. In *Other Conferences*, 1999.

Dong ming Li, Yuan zhi Wang, and Bo Du. Research on segmentation methods of weed and soil background under hsi color model. In *2009 Second International Workshop on Knowledge Discovery and Data Mining*, page nil, 1 2009. doi : 10.1109/wkdd.2009.113. URL `https://doi.org/10.1109/wkdd.2009.113`.

Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1) :62–66, 1979. doi : 10.1109/TSMC.1979.4310076.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for

biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. URL http://arxiv.org/abs/1505.04597.

Inkyu Sa, Zongyuan Ge, Feras Dayoub, Ben Upcroft, Tristan Perez, and Chris Mccool. Deepfruits : A fruit detection system using deep neural networks. *Sensors*, 16 :1222, 08 2016. doi : 10.3390/s16081222.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.

Donald F. Specht. Probabilistic neural networks. *Neural Networks*, 3(1) :109–118, 1990. ISSN 0893-6080. doi : https://doi.org/10.1016/0893-6080(90)90049-Q. URL https://www.sciencedirect.com/science/article/pii/089360809090049Q.

S. P. Stuart Lloyd. Binary block coding. *Bell System Technical Journal*, 36(2) :517–535, 1957. doi : https://doi.org/10.1002/j.1538-7305.1957.tb02410.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/j.1538-7305.1957.tb02410.x.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

Achmad Widodo and Bo-Suk Yang. Support vector machine in machine condition monitoring and fault diagnosis. *Mechanical Systems and Signal Processing*, 21(6) :2560–2574, 2007. ISSN 0888-3270. doi : https://doi.org/10.1016/j.ymssp.2006.12.007. URL https://www.sciencedirect.com/science/article/pii/S0888327007000027.

Jui-Cheng Yen, Fu-Juay Chang, and Shyang Chang. A new criterion for automatic multilevel thresholding. *IEEE Transactions on Image Processing*, 4(3) :370–378, 1995. doi : 10.1109/83.366472. URL https://doi.org/10.1109/83.366472.