UNIVERSITÉ DU QUÉBEC À MONTRÉAL

IDENTIFICATION AUTOMATIQUE D'ARBRES À PARTIR DE PHOTOS

MÉMOIRE

PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE EN INFORMATIQUE

PAR

MAXIME FAUBERT LAURIN

JANVIER 2024

REMERCIEMENTS

Merci à ma direction de recherche, Marie-Jean Meurs et Joël Lefebvre, de m'avoir guidé et supporté durant cette période de ma vie. Merci aussi à tous ceux qui m'ont soutenus ; particulièrement Rose Latendresse, Virginie Sorel et Marie Michèle Uyisenga qui m'ont permis de persévérer jusqu'à la fin.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ALGORITHMES

RÉSUMÉ

Le monde vit de multiples crises environnementales : changements climatiques, perte de biodiversité, dégradation des sols, pollution, ... Les villes peuvent parfois être vues comme une source de ces problèmes. On y couvre le sol d'asphalte et de béton, on détruit les écosystèmes qui étaient présents, on y produit déchets et gaz polluants, ... Pourtant, elles peuvent aussi contribuer significativement aux solutions dont on a grand besoin.

Parmi les moyens à la disposition des municipalités se trouve leur forêt urbaine. Ces arbres sont présents dans les parcs, mais aussi sur les bords des rues ainsi que les terrains des résidents, entreprises et bâtiments publics. Ils peuvent jouer un rôle fondamental pour combattre les crises environnementales et améliorer la qualité de vie des citoyens. Pour maximiser leurs bienfaits, ces forêts urbaines doivent être gérées adéquatement, et cette gestion nécessite de bien les connaître.

Ce travail visera l'amélioration de la quantité et la qualité des informations que les municipalités possèdent sur leur forêt urbaine. Dans ce but, nous avons créé quatre ensembles de données afin d'entraîner des modèles d'identification automatique des arbres. Nous proposons aussi une méthodologie pour gérer et conserver ses données de façon à pouvoir les utiliser et les améliorer en collaboration avec d'autres acteurs partageant des objectifs similaires.

Nous suggérons ensuite une méthode permettant de détecter des prédictions non fiables afin d'augmenter la proportion d'identifications exactes par nos modèles. Ce type de détecteur est particulièrement utile dans notre contexte où on peut avertir une personne utilisant un système qu'il vaudrait mieux reprendre une photo différente ou procéder à une identification manuelle.

Finalement, nous démontrons la capacité de l'apprentissage profond de discriminer différents types d'arbres, au moins pour des types grossiers, à partir de photos aériennes.

Mots-clefs Apprentissage profond ; Intelligence artificielle ; Climat ; Vision par ordinateur ; Identification de végétaux ; Environnement ; Villes ; Forêts urbaines ; Géomatique ; Système d'information géographique

INTRODUCTION

La population est de plus en plus sensible à l'importance de l'environnement et de sa préservation. Les signes de sa dégradation étant de plus en plus visibles (changements climatiques, perte de biodiversité, pollution, ...) et sa protection est un enjeu croissant. Parmi les tâches servant la protection de l'environnement, se trouve la gestion des forêts urbaines. Ces arbres présents sur le territoire des villes, que ce soit dans les parcs, le long des rues ou sur les terrains privés, contribuent au bien-être de la population et de la biodiversité locale par les services écosystémiques qu'ils offrent. Ces services incluent de nombreux bienfaits tels que la régulation du climat, de la qualité de l'air (Salmond *et al.*, 2016; Jim et Chen, 2008; Livesley *et al.*, 2016), des sols et de l'eau (Livesley *et al.*, 2016) ainsi que la mitigation des inondations (Livesley *et al.*, 2016), sans oublier l'amélioration de la santé et du bien-être des humains partageant ces espaces (Livesley *et al.*, 2016).

Une bonne gestion de celle-ci est toutefois nécessaire afin d'assurer sa bonne santé ainsi que d'obtenir les services écosystémiques souhaités. Les forêts urbaines ont tendance à devenir des monocultures choisies purement selon le courant esthétique du moment en l'absence d'une gestion éclairée.

La première étape d'une bonne gestion consiste à avoir un inventaire de bonne qualité. Avoir un bon inventaire est un processus long et coûteux qu'il faut répéter régulièrement afin de le maintenir à jour. C'est sans compter les difficultés supplémentaires d'inclure les arbres des propriétés privées. Ce travail, pourtant crucial, est donc souvent de mauvaise qualité, que ce soit parce qu'il n'est pas à jour, incomplet, contient des erreurs ou est sous une forme peu utile. Au début de ce projet, certain arrondissements de la ville de Montréal tenaient encore leurs inventaires sur papier ; pire, certains ne se donnaient tout simplement pas la peine de tenir des inventaires. L'objectif de ce travail sera donc de contribuer à baisser les coûts et le temps nécessaires à la prise de ces inventaires en contribuant à la création d'un outil permettant l'automation d'une partie de ce processus.

Cet outil est un projet plus large de gestion des forêts urbaines appelé SylvCiT (SylvCiT, 2023) en collaboration entre des étudiants et professeurs d'informatique et de biologie de l'Université du Québec à Montréal. Ce projet est une application en ligne, libre et scientifique, de gestion des forêts urbaines ; offrant des recommandations d'espèces à planter selon

1. les espèces déjà présentes,

2. les conditions locales,
3. les objectifs visés.

SylvCiT étant accessible à tous, il serait bénéfique de donner la possibilité aux citoyens de contribuer à sa croissance en leur permettant d'y ajouter des arbres. Ce serait particulièrement pratique pour les arbres privés auxquels les employés municipaux n'ont pas toujours accès, mais aussi pour permettre des mises à jour ponctuelles des inventaires, même lorsque la ville n'a pas les ressources à y consacrer.

Question de recherche

Les travaux présentés dans ce mémoire s'inscrivent donc tous dans l'objectif d'*utiliser l'intelligence artificielle et la vision par ordinateur afin d'accélérer et faciliter l'identification automatique d'arbres en milieu urbain*. Les solutions envisagées tiendront compte de la volonté de permettre à la communauté de les utiliser pour contribuer à leur région.

Plusieurs défis se présentent dans un tel projet. D'abord, la difficulté d'un monde ouvert dans lequel il est presque impossible de couvrir toutes les classes possibles d'arbres à identifier. De plus il existe des différences entre les sujets d'une même classe dû à l'environnement dans lequel ils se développent, même pour une même zone climatique, les villes forment des microclimats. Le moment de l'observation est aussi en cause, non seulement dû à l'âge du sujet, mais surtout la saison. Nous combinons aussi la nécessité de discriminer nos classes à la fois selon des critères grossiers (p. ex. les différences entre les feuillus et conifères) et des critères très fins (p. ex. la rugosité des feuilles pour différencier certaines espèces d'ormes). Finalement, la disponibilité des données est souvent un problème en apprentissage automatique, mais les derniers défis mentionnés décuplent ce problème. Même si plusieurs ont travaillé sur des objectifs similaires, ces travaux semblent inexistants dans notre région.

Une contribution majeure de nos travaux consiste donc en la création d'ensembles de données. Un ensemble se concentrant sur des images de feuilles d'espèces présentes dans la région de Montréal. Un autre contient des images de plain pied (l'arbre complet ou presque) prises au niveau de la rue (à partir du sol) des mêmes espèces. Les deux derniers ensembles contiennent des images avec une vue du haut : des images tirées de services de cartographie en ligne pour l'un, et des images de plus haute résolution prises par une entreprise de géomatique pour l'autre. Il sera éventuellement nécessaire de gérer non seulement ces données, mais également les données qui seront, nous l'espérons, générées

par l'utilisation de SylvCiT ainsi que les apports de travaux futurs. Nous proposons donc aussi une méthode qui permettra de gérer toutes ces données efficacement et d'une façon qui rendra leur utilisation aisée pour de futurs travaux.

Nous proposons et analysons une méthode permettant de détecter des prédictions jugées peu fiables afin de pouvoir les rejeter. Ce type de méthode permet en général d'améliorer les performances des prédictions restantes d'un modèle. Dans notre contexte, il présente l'avantage supplémentaire de pouvoir prévenir l'utilisateur, lui permettant de prendre une photo différente du sujet, procéder à une identification manuelle ou autre solution pertinente. Finalement, nous étudions la possibilité de non seulement identifier la présence d'arbres à partir des airs, mais de discriminer entre différents types.

Nous commencerons par présenter dans le chapitre 1, une revue de littérature sur les sujets qui seront traités dans ce document. Le chapitre 2 présentera ensuite les ensembles de données qui ont été construits dans le cadre de nos travaux. Il exposera également la difficulté de trouver des données déjà existantes pouvant être utilisées et les tâches nécessaires pour pouvoir les utiliser. Le chapitre 3 décrit une méthode pour gérer les données, les modèles utilisés dans le cadre des expériences ainsi que la méthodologie de ces expériences et les métriques utilisées pour les évaluer. Finalement, le chapitre 4 montre les résultats des expériences et analyse leurs performances.

CHAPITRE 1
ÉTAT DE L'ART

Ce chapitre débute par une revue de concepts fondamentaux sur les réseaux de neurones qui seront utilisés dans nos recherches. La section 1.2 présente les techniques d'apprentissage profond utilisées spécifiquement avec des images. La section 1.3 contient certains défis de la classification dans le monde réel et des solutions pour y faire face. Finalement, un aperçu des méthodes pour détecter les arbres sur des images aériennes est présenté dans la section 1.4.

1.1 Les réseaux de neurones

Concepts généraux

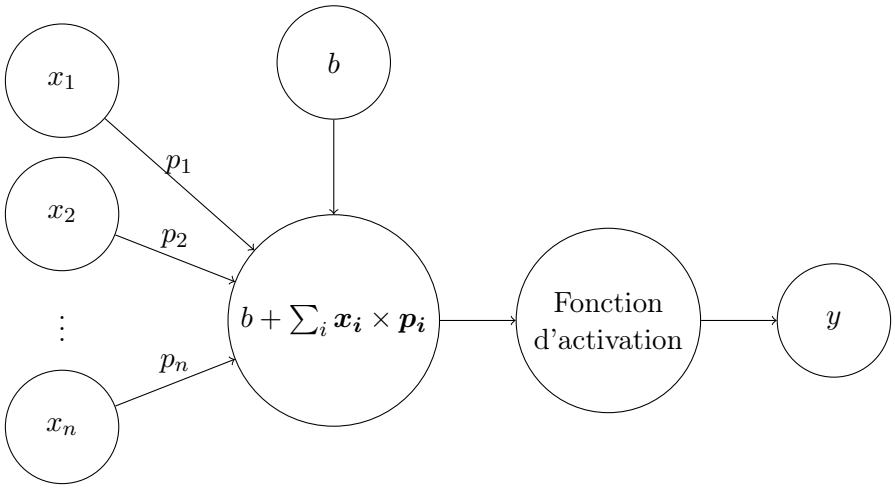


FIGURE 1.1 : Structure d'un neurone artificiel

Un neurone artificiel (figure 1.1) reçoit en entrée des valeurs x_i. Chacune de ces valeurs est multipliée par un facteur p_i appelé poids. Il additionne ensuite toutes les valeurs obtenues ainsi qu'un biais b qui ne dépend par des entrées. On réfère collectivement aux poids et biais sous le nom de paramètres. Ce résultat passe ensuite par une fonction d'activation pour obtenir la sortie finale du neurone (y) (Russell et Norvig, 2009, ch. 18.7).

Un neurone seul ne peut calculer que des fonctions très simples. Intervient alors le théorème de Taylor, stipulant qu'une fonction répondant à certains critères peut être approximée dans le voisinage d'un point par la somme de fonctions polynômiales dont les éléments sont les dérivées de la fonction

en ce point (Taylor, 1715, pp. 21–23). Suivant ce principe, des neurones peuvent être combinés pour approximer des fonctions plus complexes. Ce concept est éventuellement validé formellement par le théorème d'approximation universelle (Cybenko, 1989). Cette utilisation de plusieurs neurones sur un même ensemble d'entrées permet d'obtenir les premiers réseaux de neurones à une couche cachée.

Bien qu'il soit théoriquement possible de calculer n'importe quelle fonction avec une seule couche de neurones, une fonction complexe peut en nécessiter un nombre vertigineux et, donc, un temps de calcul peu ou pas du tout réaliste en pratique. L'ajout de couches intermédiaires (voir figure 1.2) a rendu possible l'entraînement de réseaux pour des fonctions de plus en plus complexes dans des temps plus raisonnables (Baral *et al.*, 2018). La recherche dans ce domaine, l'apprentissage profond, a également permis la découverte d'architectures spécialisées pour certaines tâches telles que les réseaux à convolutions (voir sous-section 1.2), particulièrement adaptées au travail avec des images.

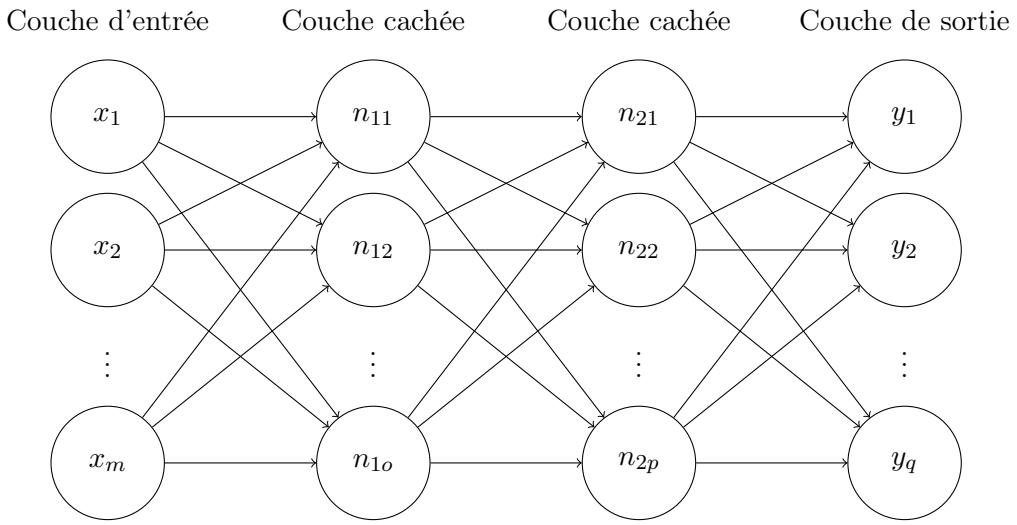


FIGURE 1.2 : Structure d'un réseau de neurones

Initialisation des paramètres

Une bonne stratégie d'initialisation des paramètres est importante pour obtenir une convergence rapide. Une mauvaise initialisation peut même aller jusqu'à l'empêcher complètement. Malheureusement, il est difficile de concevoir de bonnes stratégies puisque les mécanismes d'optimisation d'un réseau ne sont pas encore bien compris.

La seule propriété certaine est que des paramètres s'appliquant sur les mêmes entrées doivent avoir des valeurs différentes (Goodfellow *et al.*, 2016, ch. 8.4). L'algorithme d'optimisation étant déterministe, il produirait les mêmes résultats s'il utilisait des paramètres identiques. Même si certaines stratégies que nous verrons plus tard peuvent apporter des éléments stochastiques, il reste préférable d'initialiser les paramètres avec des valeurs différentes. Il est possible, par exemple, que cela permette d'éviter de perdre des informations lors de l'entraînement.

L'idéal pourrait être d'utiliser des fonctions différentes pour chaque paramètre, mais, en raison de leur abondance, cela entraînerait un temps de calcul significatif. On préfère donc utiliser une fonction unique tirant ses valeurs d'une distribution aléatoire à grande entropie. Cette méthode est simple et efficace en plus d'avoir une probabilité presque nulle d'assigner la même valeur à deux paramètres.

Les techniques d'initialisation essaient de trouver un juste milieu entre deux facteurs. Premièrement, des valeurs plus grandes semblent plus facilement briser la symétrie (les valeurs similaires obtiennent des modifications similaires à l'entraînement) en plus de réduire la perte de signal aux étapes linéaires. Ces étapes linéaires étant calculées via des produits de matrices, de plus grandes valeurs donnent de plus grands résultats. Deuxièmement, de trop grandes valeurs peuvent causer une explosion des résultats, ceux-ci devenant de plus en plus grands de façon incontrôlable. Des valeurs trop grandes risquent aussi de saturer les fonctions d'activation (Goodfellow *et al.*, 2016).

Les valeurs choisies pour les poids viennent, aujourd'hui, presque toujours de distributions uniformes ou normales. Glorot et Bengio (2010) déterminent que la grandeur idéale des valeurs est relative à la taille des couches. Ils calculent aussi que la magnitude des poids devrait rester dans une certaine zone afin d'obtenir des résultats optimaux. Pour une i^{e} couche de taille n_i, la distribution des poids P entre la couche i et la couche $i+1$ devrait avoir la propriété suivante :

$$n_i Var(P) = 1 \tag{1.1}$$

Ils proposent donc l'utilisation de la distribution uniforme 1.2, ainsi que sa version normale (1.3).

Cette approche est connue sous le nom de méthode de Xavier.

$$P \sim U\left(-\sqrt{\frac{6}{n_i + n_{i+1}}}, \sqrt{\frac{6}{n_i + n_{i+1}}}\right) \tag{1.2}$$

$$P \sim N\left(0, \frac{2}{n_i + n_{i+1}}\right) \tag{1.3}$$

Une autre équipe (He *et al.*, 2015b) tente d'améliorer la vitesse de convergence avec les modèles profonds en proposant une autre méthode. Selon eux, leur méthode obtient également de meilleurs résultats lorsque des fonctions d'activation non linéaires sont utilisées, ce que Glorot and Bengio n'avaient pas étudiées dans leurs travaux. Ils calculent que les distributions utilisées devraient avoir la propriété suivante :

$$\frac{n_i}{2} Var(P) = 1 \tag{1.4}$$

Ils arrivent ainsi à des distributions similaires (méthode de Kaiming) :

$$P \sim U\left(-\sqrt{\frac{6}{n_i}}, \sqrt{\frac{6}{n_i}}\right) \tag{1.5}$$

$$P \sim N\left(0, \frac{2}{n_i}\right) \tag{1.6}$$

Ces valeurs améliorent la vitesse de convergence, particulièrement sur les modèles très profonds (avec beaucoup de couches).

Les distributions 1.5 et 1.6 sont particulièrement efficaces pour préserver la magnitude de la variance lors de la propagation avant. Pour prioriser la préservation de la magnitude de la variance lors de la propagation arrière, il suffit de remplacer n_i par n_{i+1}. La méthode de Xavier crée une forme de compromis en intégrant ces deux valeurs dans la même distribution.

Les méthodes ajustant la magnitude des poids selon leur quantité a le désavantage de donner des valeurs très petites sur des modèles très larges. Pour pallier ce problème, on peut utiliser une méthode d'initialisation creuse (Martens, 2010). On établit alors un nombre k de poids qui seront initialisés avec des valeurs différentes de zéro. On utilisera alors cette valeur k pour déterminer la distribution d'où seront tirées les valeurs des poids. Cette méthode permet d'obtenir une plus grande variance dans les valeurs retenues, mais pose de forts a priori sur les poids choisis. Il faut également être

prudent avec les modèles utilisant certaines techniques ou filtres tel que le *max pooling* (dont nous parlerons dans la section 1.2).

Les biais choisis sont habituellement constants et déterminés selon des heuristiques. Simplement les mettre à zéro fonctionne dans la plupart des situations (Goodfellow *et al.*, 2016). Il faut toutefois choisir une stratégie adaptée à l'initialisation des poids et, parfois, d'autres facteurs.

On choisira ainsi parfois les biais de façon à éviter la saturation à l'initialisation (Goodfellow *et al.*, 2016). La fonction d'activation ReLu, par exemple, donnera une sortie de zéro pour toute entrée égale ou inférieure à zéro. Il peut alors être utile d'initialiser les biais à 0.1 plutôt que 0 afin de réduire le nombres de sorties nulles.

Dans le cas des biais de la sortie d'un modèle, on peut initialiser les biais de façon à influencer les statistiques marginales de façon appropriée Goodfellow *et al.* (2016). S'il s'agit d'une tâche de classification avec une représentation inégale de chaque classe, initialiser les biais avec des valeurs qui représentent la distribution des classes.

1.2 Classification d'images

Pour travailler avec des images, les réseaux de neurones à convolutions se sont imposés dans la dernière décennie suite aux performances extraordinaires du modèle *AlexNet* (Krizhevsky *et al.*, 2012; Szegedy *et al.*, 2016) dans le concours *ImageNet Large Scale Visual Recognition Challenge* [1] de 2012 (voir la figure 1.3). Ils sont particulièrement adaptés aux données organisées sur une ou plusieurs dimensions et peuvent être tolérants à des transformations géométriques telles que la translation en plus de permettre de travailler avec des entrées de tailles variables (Goodfellow *et al.*, 2016, ch. 9).

Une couche de convolution reçoit en entrée un tenseur (généralisation du concept de matrice à plus de 2 dimensions) (LeCun *et al.*, 1989). Quand on travaille avec des images, ce tenseur possède au moins quatre dimensions : la taille du lot (pour traiter plusieurs images à la fois), la hauteur et largeur ainsi que le nombre de canaux (les couleurs). Une matrice de convolution est alors appliquée au tenseur afin de créer une carte d'attributs qui abstrait l'information de l'entrée sous la forme

1. `https://image-net.org/challenges/LSVRC/2012/`

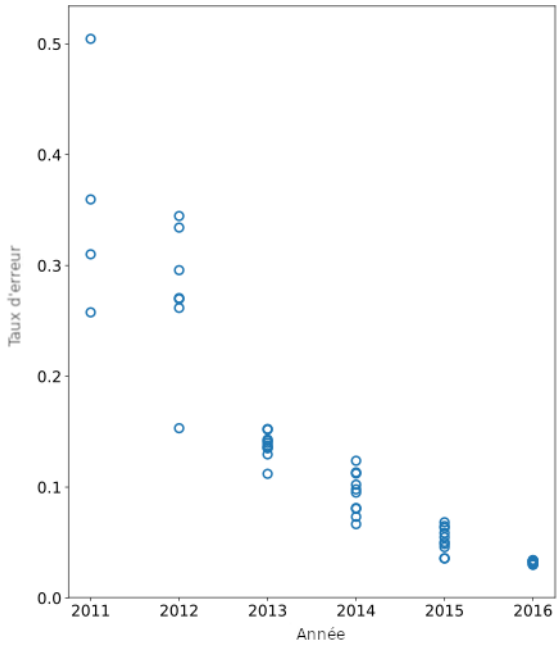


FIGURE 1.3 : Résultats des compétitions ImageNet de 2011 à 2016 (Gkrusze, 2018)

d'un nouveau tenseur (Goodfellow *et al.*, 2016, ch. 9.1).

Afin d'améliorer les prédictions, une fonction d'activation est habituellement appliquée à la sortie de chaque neurone artificiel (Russell et Norvig, 2009, ch. 18.7). Il s'agit d'une fonction non-linéaire qui permettra au modèle de mieux s'adapter au problème. Une fonction très utilisée depuis *AlexNet*, ReLU, est donnée par $f(x) = max(0, x)$. Cette étape est appelée couche d'activation.

Après une ou plusieurs séquences de convolution et activation, vient une couche de *pooling*. Cette couche réduit la taille d'un tenseur en échantillonnant son contenu et introduit une tolérance au changement d'échelle de l'image (Szegedy *et al.*, 2016) ainsi qu'une tolérance locale à la translation (Mouton *et al.*, 2020). Pour chaque ensemble de cellules d'une taille choisie, on sélectionne une seule valeur pour les représenter toutes. On peut choisir cette valeur en faisant la moyenne, de façon stochastique ou autres fonctions plus complexe. De nos jours, on choisit le plus souvent le maximum puisque cette méthode s'avère être la plus performante dans la plupart des cas (Scherer *et al.*, 2010). La figure 1.4 présente un exemple de *max pooling* avec un filtre 2×2 et un pas de 2.

Un bloc convolutif est habituellement composé d'une ou plusieurs couches convolutives (et couches

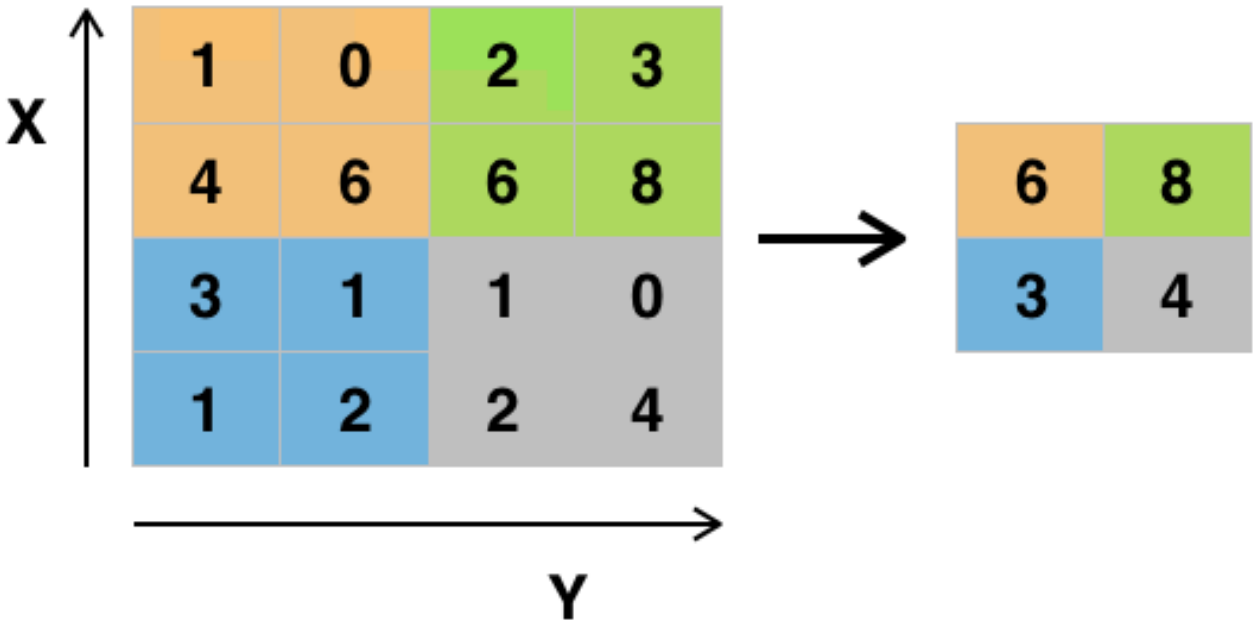


FIGURE 1.4 : Max pooling avec un filtre 2 × 2 et un pas de 2 (Aphex34, 2015)

d'activation) suivies d'une couche de *pooling*. Comme l'illustre la section rouge de la figure 1.6, plusieurs blocs convolutifs se succèdent afin d'extraire des attributs de plus en plus abstraits de l'entrée initiale (voir figure 1.5) (Albawi *et al.*, 2017). La sortie du dernier bloc convolutif sert finalement d'entrée au classifieur, composé d'une ou plusieurs couches de neurones entièrement connectées dans lesquelles, comme le nom l'indique, chaque neurone d'une couche est connecté à tous les neurones de la couche suivante. Le vecteur finalement obtenu à la sortie de ces couches correspond aux prédictions pour chacune des classes du problème.

1.3 Détection d'anomalies dans les prédictions

Il existe plusieurs types d'anomalies qu'il peut être pertinent d'identifier. Dans leur méta analyse, Geng *et al.* (2021) classent ces différentes anomalies en fonction de trois critères : les contextes à l'entraînement et dans les tests ainsi que l'objectif de la détection. Les contextes sont déterminés en fonction de la connaissance que l'on a des classes :

Les classes que l'on sait connues Il s'agit de données bien étiquetées et groupées en classes

Les classes que l'on sait inconnues Ce sont les données avec un certain étiquetage, mais non groupées en classe pertinentes pour la classification

Les classes que l'on ne sait pas connues Il s'agit de classes que l'on connaît, mais pour

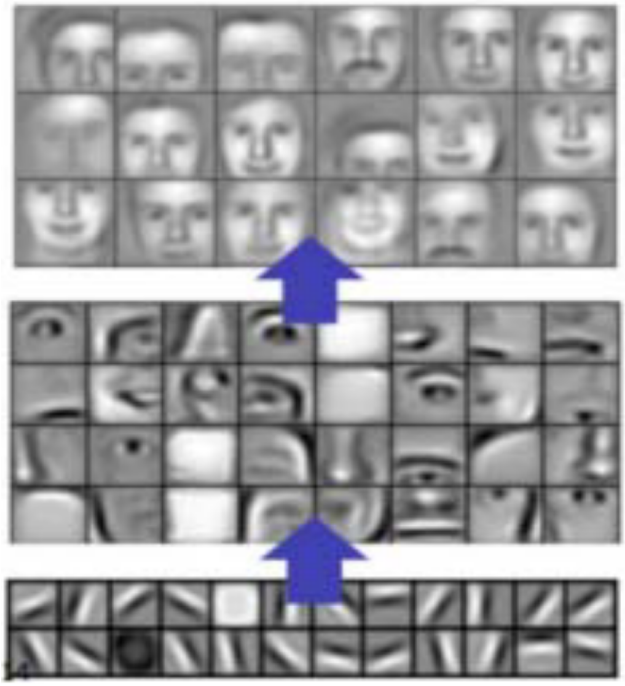


FIGURE 1.5 : Attributs appris par un réseau de neurones à convolutions (Albawi *et al.*, 2017)

lesquelles aucunes données ne sont disponibles pour l'entraînement

Les classes que l'on ne sait pas inconnues On ne connaît pas ces classes et aucunes données ne sont disponibles pour l'entraînement

Deux des types d'anomalies identifiées par Geng et al. (2021) semblent intéressantes dans le cadre de notre projet. La reconnaissance d'ensembles ouverts où l'on entraîne les modèles sur des classes que l'on sait connues, mais qui contiendront aussi des classes que l'on ne sait pas inconnues pendant les tests. Dans ce cas, l'objectif est de détecter les exemples inconnus et les rejeter ou être capable de découvrir les nouvelles classes auxquelles ces exemples appartiennent. Le deuxième type est la classification avec option de rejet. On sait les classes connues autant à l'entraînement qu'aux tests, l'objectif étant de détecter les prédictions peu fiables et de les rejeter. Ces deux types d'anomalies correspondent également aux deux motifs de rejet identifiés par Hendrickx *et al.* (2021), soit le rejet de nouveauté et le rejet d'ambiguïté.

1.3.1 Les ensembles ouverts

Une limitation importante des classifieurs est la nature ouverte du monde auquel certains seront confrontés. Il n'est possible de les entraîner que sur un nombre fini de catégories qui ne sera souvent

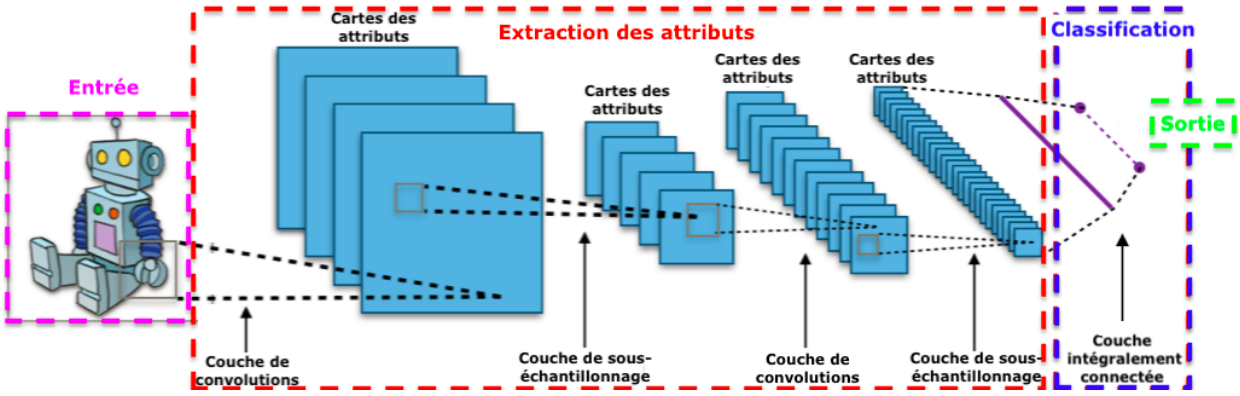


FIGURE 1.6 : Architecture de réseau neuronal convolutif classique (Aphex34 et ClaudeCoulombe, 2021)

qu'un sous-ensemble des catégories possibles. Concernant la classification d'arbres à Montréal, les catégories connues sont les espèces figurant sur la liste fournie par la ville et qui devrait contenir toutes les espèces plantées dans les espaces publiques. Des citoyens pourraient toutefois avoir planté d'autres espèces sur leur terrain, espèces qui ne seraient donc pas présentes dans les données d'entraînement de nos modèles.

Formellement, travailler avec des ensembles ouverts nécessite une solution en trois étapes (Bendale et Boult, 2015) :

1. Une fonction permettant de reconnaître les données appartenant à de nouvelles classes,
2. Un processus d'étiquetage créant les nouvelles classes et les appliquant aux données appropriées,
3. Une fonction d'apprentissage incrémentale ajoutant les nouvelles classes ainsi que les fonctions permettant de les classifier au modèle.

Dans le cadre de ce document, l'intérêt est porté particulièrement à la première étape : les fonctions permettant de reconnaître les données représentant de nouvelles classes. Une solution proposée à ce problème consiste à ajouter une classe « autre » dans laquelle ces cas pourront être classés. Des données variées sélectionnées en dehors des classes connues sont alors étiquetées « autre » pour entraîner le modèle. La création d'un ensemble de données contenant suffisamment d'exemples « autre » pertinents devient alors le point d'intérêt de cette technique (Karmakar et Pal, 2018).

1.3.2 Option de rejet

Puisque notre projet se prêtera bien à la possibilité de fournir de nouvelles données au besoin (reprendre des photos sous des angles ou distances différentes), l'option de rejeter des prédictions peu fiables et en prévenir l'utilisateur est particulièrement intéressante.

Un réseau de neurones donne ses prédictions sous la forme de scores de confiance numériques, il est donc possible de rejeter des prédictions en établissant expérimentalement un score minimal nécessaire pour chaque classe. Lors de l'entraînement, et en ne gardant que les bonnes prédictions, on détermine le score minimum pour considérer qu'une prédiction est fiable pour chacune des classes (Hang et Aono, 2016). Si une classe est prédite avec un score en dessous de ce minimum, la prédiction est rejetée et on considère que le modèle ne sait pas dans quelle classe cette entrée devrait être. Les auteurs ont également testé cette technique sur des ensembles ouverts et ont également obtenu une légère amélioration de leurs prédictions, rejetant environ 2,5% des images appartenant à des classes inconnues. Une fois l'entraînement terminé, il est possible de continuer à mettre à jour ce score minimum en fonction des résultats si une rétroaction est possible (les auteurs mentionnent les données de validations puisque leur valeur de vérité est disponible). Ces mises à jour permettraient d'obtenir des scores minimums plus représentatifs que ceux obtenus sur les données d'entraînement.

1.3.3 Évaluation de modèles avec option de rejet

En général, on veut évaluer la justesse des prédictions du modèle. Lorsqu'une option de rejet est introduite, on ne peut se contenter d'évaluer normalement les prédictions qui sont conservées. En effet, il suffirait d'utiliser une fonction qui rejette énormément de prédictions et, tant qu'elle rejette surtout les prédictions erronées, on obtiendrait d'excellentes évaluations. Il faut donc évaluer les modèles de façon à tenir compte du fait que des prédictions sont écartées (Hendrickx *et al.*, 2021).

Condessa *et al.* (2017) soulignent qu'évaluer uniquement la qualité du rejet présente un problème similaire puisqu'une fonction de rejet qui exclut très peu de prédictions serait favorisée. Par conséquent, il est nécessaire d'évaluer la probabilité qu'une prédiction soit juste et conservée ou qu'elle soit mauvaise et rejetée. Ils suggèrent donc de considérer la fonction de rejet comme un classifieur

binaire dont la matrice de confusion est donnée par

$$\begin{bmatrix} |A \cap \bar{R}| & |\bar{A} \cap \bar{R}| \\ |A \cap R| & |\bar{A} \cap R| \end{bmatrix}$$

où A est l'ensemble des prédictions exactes du premier classifieur et R est l'ensemble des prédictions rejetées par la fonction de rejet. La notation $|A|$ indique la cardinalité de A et $\bar{A}$ est le complément de A.

L'autre façon d'évaluer ce type de modèle consiste à évaluer les prédictions non rejetées en fonction du taux de rejet (Hendrickx *et al.*, 2021). Pour ce faire, Nadeem *et al.* (2009) introduisent la courbe Exactitude-Rejet (ARC). Cette courbe décrit l'exactitude du modèle en fonction du taux de rejet, l'objectif étant que l'aire sous la courbe soit la plus élevée possible. Il est, bien sûr, possible d'utiliser des courbes comparant d'autres métriques avec le taux de rejet telle que la F-mesure (Pillai *et al.*, 2011).

Il est toutefois nécessaire de pouvoir faire varier le taux de rejet du modèle afin de dessiner cette courbe et ce n'est donc pas toujours une méthode applicable. Si ce n'est pas possible, on peut adapter cette méthode pour dessiner une courbe en fonction d'un autre hyper-paramètre approprié (Abbas *et al.*, 2019; Hanczar, 2019).

1.4 Détection d'arbres à partir des airs

Les techniques de détection d'arbres cherchent à déterminer quelles surfaces sont couvertes par ceux-ci. Le résultat attendu est un masque indiquant quels pixels, puisque l'on travaille avec des images numériques, font partie d'un arbre. Idéalement, on voudrait que ce masque soit capable de différencier les individus, indiquant ainsi l'appartenance d'un pixel à un arbre particulier.

Ces techniques nécessitent habituellement des données spécialisées nécessitant l'utilisation de satellites ou autre matériel spécialisé. Le plus souvent, il est nécessaire d'avoir des bandes spectrales hors du spectre de la lumière visible. Parfois, on a besoin d'information sur l'altitude. On parlera alors souvent de modèle numérique d'élévation (MNE) qui modélise la surface d'un terrain et l'élévation des structures qui s'y trouvent.

1.4.1 Détection de la végétation

Pour trouver des arbres dans une image aérienne, on commence par la prétraiter en trouvant la végétation. Cela se fait en comparant les bandes spectrales de l'image à l'aide d'algorithmes de traitement d'images tels que *normalized difference vegetation index* (NDVI) (Peng Gong *et al.*, 2003). Cet algorithme mesure la différence entre la lumière infrarouge proche (que la végétation reflète) et rouge (que la végétation absorbe) pour quantifier la présence de végétation. La nécessité d'avoir de l'information sur l'intensité de bandes spectrales en dehors du spectre visible (infrarouge proche) a pour conséquence que NDVI ne peut généralement être utilisé qu'avec des images satellites puisque les drones et avions permettant la prise d'images aériennes ne sont généralement pas équipés d'appareils permettant la capture de ces bandes spectrales. Suite à l'utilisation de NDVI, il est possible de faire une première segmentation sémantique de la végétation en fusionnant les régions dont l'indice obtenu grâce à cet algorithme est faible (signifiant qu'elles sont similaires).

1.4.2 Extraction des arbres

Si la méthode de prise d'images permet d'avoir de l'information sur l'altitude, il est possible de séparer les arbres du reste de la végétation avec des approches basées sur les différences de hauteur. En marquant manuellement quelques points que l'on sait être le sol, on ne garde que les régions dont l'altitude dépasse celle de ces points par une certaine valeur (Huang *et al.*, 2018).

En l'absence d'information explicite sur l'altitude, il est possible d'estimer les différences de hauteurs entre différents pixels à l'aide d'approches basées sur l'analyse des textures. Ce type d'analyse quantifie la variation de l'intensité des pixels d'une région donnée. Une intensité de zéro dénote un pixel noir alors qu'un pixel blanc aura la valeur la plus élevée de l'échelle utilisée. Les valeurs de textures changeant selon l'altitude, on peut alors extraire les arbres en gardant les pixels hauts et rejetant les pixels bas, qui correspondent plutôt à de la végétation basse telle que de l'herbe (B.L. *et al.*, 2018).

1.4.3 Délimitation des arbres

Une méthode similaire utilise plutôt un algorithme basé sur les niveaux de gris et une transformation de l'image, *grey-level morphological top-hat* avec un élément structurant ayant la forme d'un

disque. Ce traitement permet d'obtenir un masque révélant les zones ayant vaguement la forme d'un disque. En utilisant les données d'altitudes on peut trouver les maximums locaux, correspondant aux sommets des arbres (Xiao *et al.*, 2018). Les maximums près du niveau du sol sont éliminés ainsi que ceux qui correspondent à des branches. Il reste ensuite à délimiter la couronne des arbres trouvés.

Il est possible de délimiter les couronnes individuelles avec une analyse de contraste. Sur le masque obtenu par l'application de l'algorithme NDVI, le sommet des arbres ont des valeurs élevées alors que les contours ont des valeurs plus basses (Huang *et al.*, 2018).

L'approche des régions croissantes s'avère aussi utile. En partant de valeurs de départ déterminées préalablement (p. ex. en trouvant le sommet des arbres), on détermine à chaque itération quels pixels adjacents appartiennent à la même région. On prend cette décision en fonction de la similitude de certains critères. Les critères choisis par B. L. et al. (2018) permettent d'analyser les différences d'altitudes entre les différentes régions créées et c'est en analysant ces gradients de hauteur qu'il est possible de progressivement agrandir ces régions jusqu'à ce que chacune d'entre elles corresponde à une couronne individuelle. Xiao et al. (2018) utilisent plusieurs données incluant les couleurs, l'altitude ainsi que la distance avec le sommet de l'arbre, celui-ci devant être à peu près au centre. Ils éliminent également les couronnes anormalement larges comparativement à leurs voisins.

Un traitement des contours trouvés améliore ensuite les contours trouvés en les lissant et éliminant les trous. Ce traitement fait usage de filtres morphologiques utilisant les opérations d'érosion et de dilatation (Huang *et al.*, 2018).

Ce chapitre a présenté les concepts utiles à la classification d'images et la détection d'arbres. Il est maintenant temps de présenter les données qui ont été rassemblées pour rendre nos projets possibles.

CHAPITRE 2
ENSEMBLES DE DONNÉES

La qualité des prédictions des modèles étant dépendante des données avec lesquelles ils sont entraînés, une attention particulière doit être portée à la création des ensembles de données servant à l'entraînement. Dans ce chapitre, nous identifierons d'abord les différentes contraintes guidant nos choix dans la section 2.1. La section 2.2 présentera ensuite les ensembles de données *Macro*, *Horizon*, *Sylphe* et *Map* qui ont été construits ainsi que leurs caractéristiques [1].

2.1 Contraintes

Les contraintes guidant les choix pour les ensembles de données découlent de deux facteurs principaux : nos objectifs ainsi que les limites de nos moyens et de ceux de nos partenaires.

L'objectif étant d'automatiser l'identification d'arbres en milieu urbain, il est généralement préférable que nos données proviennent également de milieux urbains puisque les arbres croissent différemment dans ces milieux qu'en milieu naturel. Au delà de l'apparence de l'arbre, le contexte peut aussi avoir de l'importance : une image de la rue ou près d'une résidence sera bien différente d'une image prise dans un boisé.

Il est également souhaité que les résultats de ces recherches permettent la création d'outils utilisables par la population en général. Le but étant de favoriser la participation citoyenne, par exemple, en permettant aux citoyens d'identifier les arbres présents sur leur terrain privé pour les inclure dans les inventaires des outils ou villes. Les données doivent donc être comparables à ce qu'une personne utilisatrice pourrait obtenir avec un appareil commun tel qu'un téléphone cellulaire ou un drone disponible au grand public. Cette contrainte élimine donc la possibilité d'utiliser du matériel spécialisé tel que des appareils de *laser imaging, detection, and ranging* (LIDAR), qui utilisent des lasers pour mesurer les distances avec précision, ou des images en multiples bandes spectrales fournies par des satellites tels que Lansat-7 qui prend des images en 7 canaux de couleurs dont 4 sont dans les infrarouges et les 3 autres sont dans la bande du spectre visible. Seules des images en

1. Les ensembles de données sont disponibles à l'adresse suivante : `https://cloud.labikb.ca/index.php/s/Dncs4wEn8Z2biyM`

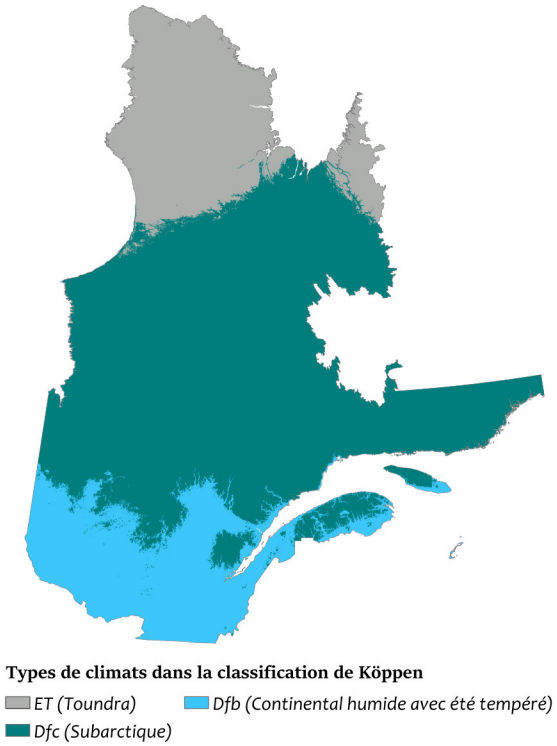


FIGURE 2.1 : Carte des climats de Köppen au Québec (Peterson, 2016)

trois bandes « classiques » (rouge, vert, bleu) seront donc utilisées.

Le projet devant être utile pour nos partenaires des villes de Montréal et des environs, les images devront représenter adéquatement la végétation présente sur leur territoire. Le milieu urbain a déjà été évoqué, il s'y ajoute la nécessité que les données proviennent de zones avec un climat similaire. Nous tenterons donc de nous limiter aux zones de climat continental humide avec été tempéré qui, selon la méthode de classification des climats de Köppen, est le climat couvrant le sud du Québec (zone Dfb dans la Figure 2.1).

Finalement, les partenaires fournissent également une liste d'espèces d'arbres présentes sur leur territoire qui devront être présentes dans les données. Cette liste est présentée dans l'annexe A. Ils fournissent aussi des inventaires des arbres présents sur leur territoire. Malheureusement, les inventaires ne contiennent pas tous les arbres présents, certains arrondissements n'ont pas d'inventaires numériques (qu'ils soient sur papier ou juste inexistants) et ne contiennent habituellement pas les arbres des propriétés privées. Étant construits manuellement par des êtres humains, ils contiennent aussi des erreurs plus ou moins problématiques telles que des erreurs dans les coordonnées géographiques

ou des écritures non conformes des noms d'espèces. Ces problèmes font partie des motivations pour les travaux présentés dans ce documents, mais limitent également l'utilité de ces inventaires. Malgré tout, ils permettent la création d'ensembles de données tel que la prochaine section le montrera.

2.2 Les ensembles de données

2.2.1 Ensemble Macro

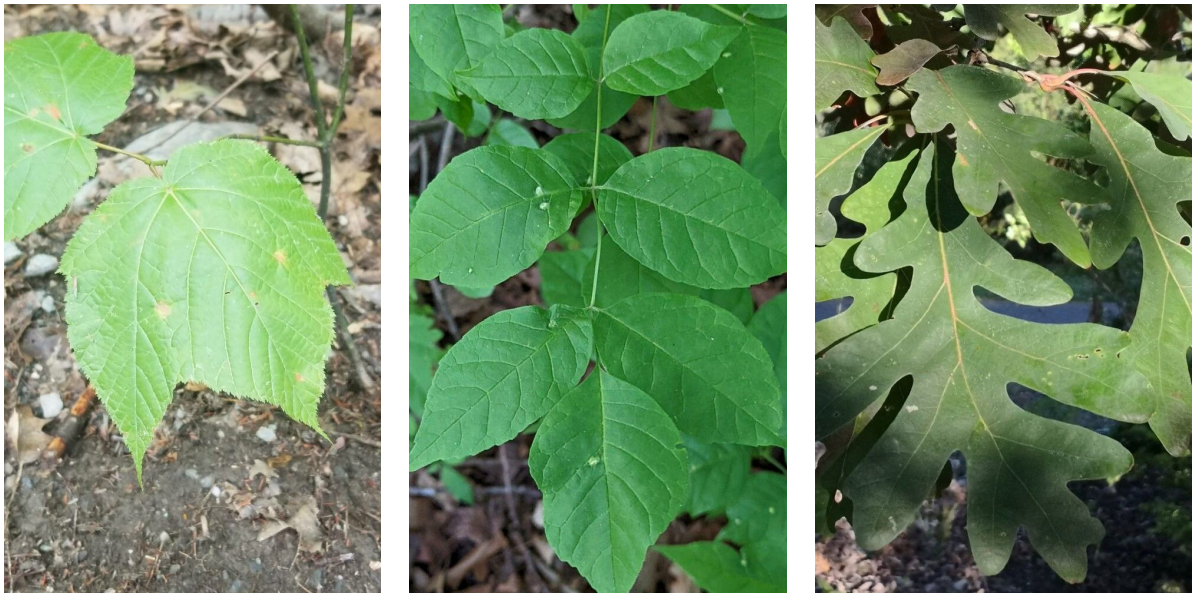
Cet ensemble contient 57 914 photos d'arbres centrées sur leurs feuilles tel qu'on peut le voir à la Figure 2.2. Les images ont différentes tailles, échelles et résolutions puisqu'elles proviennent de personnes utilisatrices d'une plate-forme en ligne (Pl@ntNet team, 2022). Cette plate-forme assure une certaine qualité en utilisant un système de vote. Les membres de la communauté s'exprime donc sur différents critères tels que la qualité de l'image et l'exactitude de l'identification.

Puisque l'emphase est mise sur les feuilles, il n'a pas été jugé utile de filtrer les images selon leur provenance. L'hypothèse étant que les caractéristiques d'une feuille ne devraient pas changer significativement selon le milieu. L'emphase des images étant sur les feuilles, cela limite également la présence d'un arrière plan pouvant fournir des informations différentes selon le milieu. Les images ne provenant pas de milieux urbains n'ont donc pas été exclues, mais beaucoup de ces images étant géolocalisées, il serait possible de les filtrer grossièrement selon leur provenance géographique si désiré.

Étant donné que ces images sont prises par des membres variés de la communauté de *Pl@ntNet*, leurs caractéristiques (orientation, résolution, ...) peuvent variées. Elles sont toutefois toutes en trois canaux de couleurs (RGB). Dans l'archive rendu disponible, un fichier texte associe chaque fichier image à l'espèce du sujet.

2.2.2 Ensemble Horizon

Cet ensemble contient 58 521 photographies d'arbres de 259 espèces différentes. Toutes sont prises au niveau de la rue et la plupart montrent l'arbre au complet, le reste se concentrant sur certaines parties de l'arbre. Elles ont été prises par des personnes variées pour une plate-forme communautaire en ligne (iNaturalist, 2022). Elles ont donc de bonnes chances de bien représenter le type de photographies prises par des personnes utilisatrices d'un outil en ligne comme celui auquel ces travaux



(a) Acer pensylvanicum (b) Fraxinus americana (c) Quercus alba

FIGURE 2.2 : Exemples de photos de feuilles Affouard *et al.* (2020)

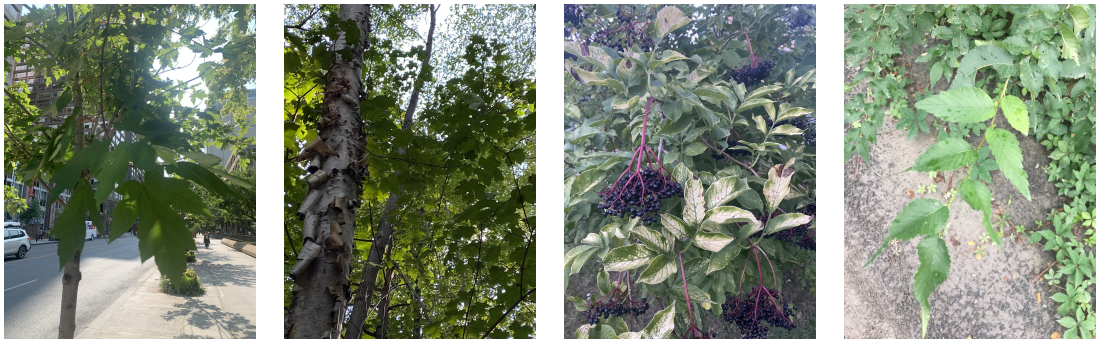
contribueront.

Il aurait été possible de concevoir un filtre rudimentaire pour séparer les photos prises en milieu urbain de celles prises en milieu rural ou forestier en utilisant les positions géographiques. Il a toutefois été jugé qu'éliminer un nombre satisfaisant d'images indésirables sans perdre trop d'images de milieu urbain aurait demandé plus de temps que cela en valait la peine à cause du grand nombre de zones boisées à recenser. Il a donc été jugé préférable de ne pas séparer les photos selon le milieu. Toutefois, un découpage grossier a été effectué afin de s'assurer que toutes les images proviennent de spécimens localisés au Québec dans des zones au climat similaire à celui de Montréal. La figure 2.3 montre quelques exemples d'images de cet ensemble.

Afin d'assurer la qualité de cet ensemble de données, le système de niveau d'iNaturalist a été utilisé. Il s'agit d'étiquettes qui sont associées à chaque observation (nom donné à l'ensemble des données fournies par une personne utilisatrice à propos d'un sujet) indiquant sa qualité. Seules des observations de niveau recherche ont été utilisées pour bâtir cet ensemble de données. Ce niveau indique que l'observation a les caractéristiques suivantes :

— est datée

— est géoréférencée



(a) Acer rubrum (b) Betula papyrifera (c) Sambucus nigra (d) Ulmus pumila

FIGURE 2.3 : Exemples de photos prises au niveau de la rue (iNaturalist Contributors, 2022)

— contient au moins une photo

— il y a un consensus par les utilisateurs de la plateforme sur l'identification

Afin de se procurer les photos, un script a été écrit afin d'utiliser l'API d'iNaturalist (Kueda, 2020) pour automatiquement télécharger les images répondant aux critères désirés (espèce et emplacement).

Cet ensemble de données doit devenir la base d'un outil intégré dans un logiciel (SylvCiT, 2023) utilisé par les villes. Il est donc appelé à grandir avec l'utilisation de cet outil et une attention particulière a été portée à son organisation et stockage. Plutôt que d'utiliser le système de fichiers pour classer les images selon leur espèce, le logiciel d'indexation Solr (Apache, 2023d) a été utilisé.

Chaque observation indexée avec Solr est accompagnée d'informations pouvant être utiles pour entraîner des modèles selon différents objectifs. La nomenclature taxonomique complète [2] (tous les rangs) est incluse afin de pouvoir créer des classes selon diverses granularités. On y trouve aussi l'origine de l'observation ; seulement iNaturalist pour l'instant, mais les observations de pl@ntNet pourrons éventuellement être ajoutées ainsi que les observations créées par les utilisateurs d'outils utilisant cet ensemble de données. S'il y a lieu, l'identifiant unique sur la plate-forme d'origine est aussi inclus. Sont ajoutés un identifiant unique pour notre propre ensemble, la date de l'observation, ses coordonnées géographiques.

2. Pour un érable rouge, on aurait par exemple : Règne : Plantae, Sous-règne : Tracheobionta, Division : Magnoliophyta, Classe : Magnoliopsida, Sous-classe : Rosidae, Ordre : Sapindales, Famille : Sapindaceae, Genre : Acer, Espèce : Acer rubrum

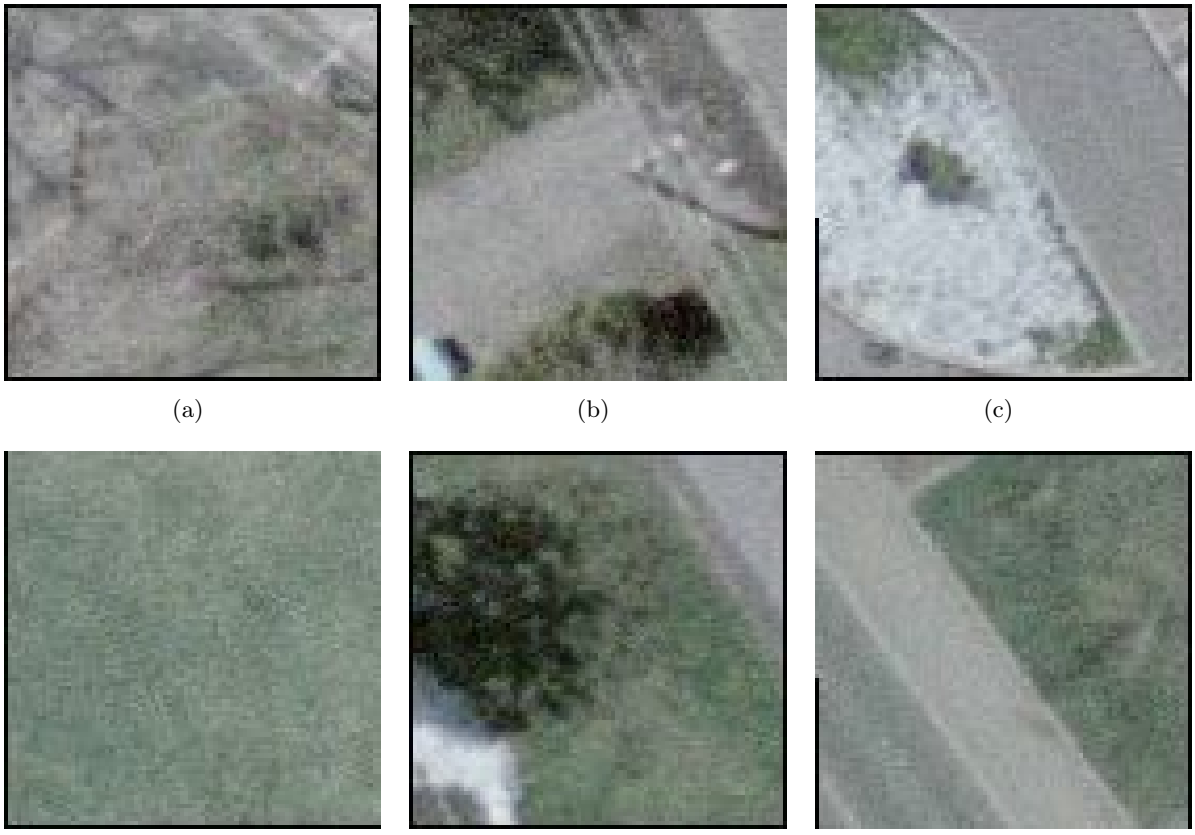
Chaque image de chaque observation contient également son identifiant unique de la plate-forme d'origine et une url où elle est disponible s'il y a lieu. On y ajoute un identifiant unique pour notre ensemble, sa licence, son auteur, format (jpg, png, ...), ses dimensions (en pixels) ainsi que le nom du fichier. Si l'information est disponible, on précise également quelle partie de la plante est montrée.

De plus, les images sont enregistrées sur le *Hadoop Distributed File System* (HDFS), un système de fichiers destiné aux données massives (voir subsection 3.1.1), afin de pouvoir facilement migrer vers un environnement de production dans l'avenir et pouvoir gérer la quantité de données résultant de l'utilisation du logiciel.

2.2.3 Ensemble Sylphe

Cet ensemble contient 738 987 images d'arbres vus des airs. Des exemples d'images de cet ensemble de données sont présentés dans la figure 2.4. Tous ces arbres sont sur le territoire de l'île de Montréal au printemps 2002. L'échelle est la même pour chaque image à 1 : 8000 avec une résolution de 1814 dpi. Chaque image couvre une surface d'environ 10×10 mètres. Malheureusement, les seules images trouvées pouvant être utilisées ont été prises avant l'apparition des feuilles au printemps.

Cet ensemble de données a été créé à partir de 1518 images aériennes de l'île de Montréal. Ces images ont été prises par la compagnie Alta Photo en 2002 (Alta Photo, 2002) en utilisant un appareil spécialisé en photographies aériennes *RMK TOP* (Zeiss, 1994). Elles sont rendues disponibles par la division de la géomatique de la ville de Montréal (Montréal, 2016). Chacune de ces images couvre approximativement 3,25 km^2 et est associée à un polygone de coordonnées permettant de la géoréférencer. Une fois les images géoréférencées, il était possible de faire des recoupements avec l'inventaire de la ville de Montréal afin de trouver la position d'arbres sur les images. Pour chaque position d'arbre ainsi identifiée, une vignette correspondant à 10×10 mètres centrée sur cette position est découpée, enregistrée et les informations liées à l'arbre (trouvés dans l'inventaire) sont associées à cette image dans un index via *Solr* (des fichiers *json* contenant cette information sont aussi fournis dans l'archive de l'ensemble de données).



(a) (b) (c)

(d) (e) (f)

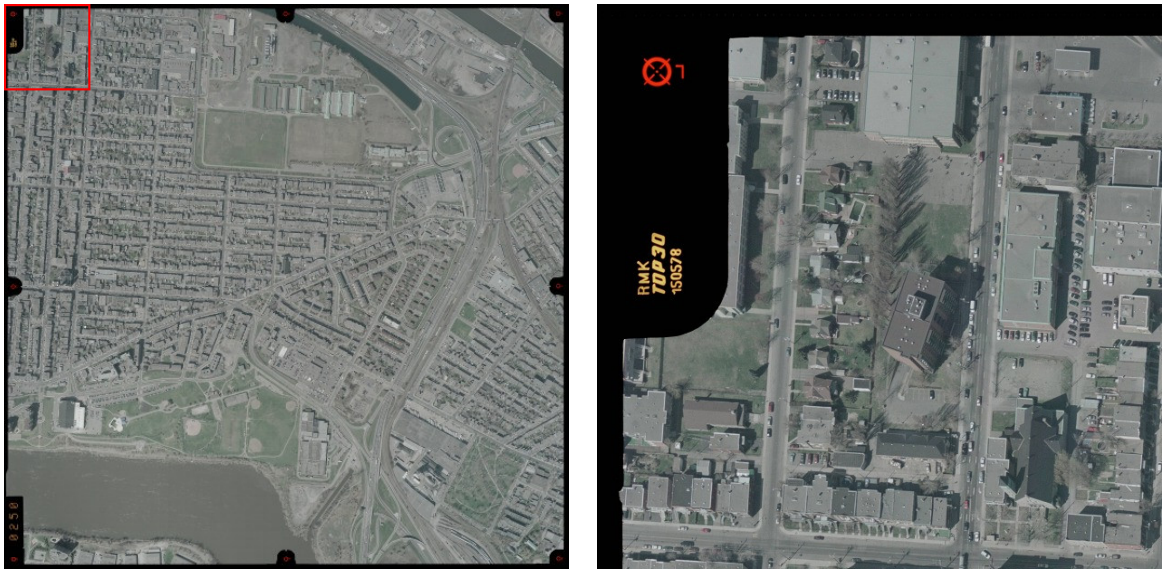
FIGURE 2.4 : Exemples d'images d'arbres vus des airs

Géoréférencement

Les images téléchargées ne contiennent pas toutes les données de géoréférencement nécessaires. Trois informations sont nécessaires : des points de contrôle donnant les coordonnées de certains points de l'image, une matrice de transformation pour interpoler le reste des coordonnées et, finalement, le système de coordonnées utilisé.

Les images contiennent leur matrice de transformation dans leurs méta-données et l'ensemble de données inclut le système de coordonnées utilisé. L'ensemble de données contient également un fichier de polygones donnant les coordonnées des points de contrôle. Les points de contrôle sur les images ne sont pas inclus d'une façon permettant d'automatiser simplement le géoréférencement des images. Ces points correspondent plutôt à huit marqueurs par image prenant la forme de petites cibles dans les coins et à la moitié de chaque côté (voir Figure 2.5).

Pour trouver les marqueurs, une image de référence a été créée ne contenant qu'un marqueur. Sur chaque image, les positions des marqueurs ont été automatiquement découvertes en comparant l'image de référence aux pixels de l'image avec une technique de *template matching*. Une fois les marqueurs ainsi découverts, il a été possible de géoréférencer les images automatiquement. La qualité de ce géoréférencement a été constatée visuellement en sélectionnant aléatoirement des images. Ces images ont été superposées à une carte de référence dans un système d'information géographique (SIG) (plus connu sous son appellation anglophone GIS pour *geographic information system*). L'alignement de points de référence, rues et bâtiments, ont ensuite été confirmé visuellement. L'alignement n'est pas parfait, mais la marge d'erreur a été jugée satisfaisante pour les besoins de l'ensemble de données.



(a) Photo de la région autour de la station de métro LaSalle

(b) Marqueur de coin

FIGURE 2.5 : Exemple de photo originale de la photothèque de Montréal (Alta Photo, 2002)

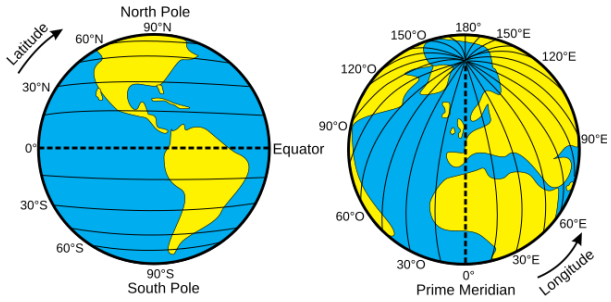
Systèmes de coordonnées

Pour représenter adéquatement le monde, deux familles de systèmes de coordonnées sont utilisées selon les situations : *geographic coordinate system* (GCS) et *projected coordinate system* (PCS) (Chang, 2019).

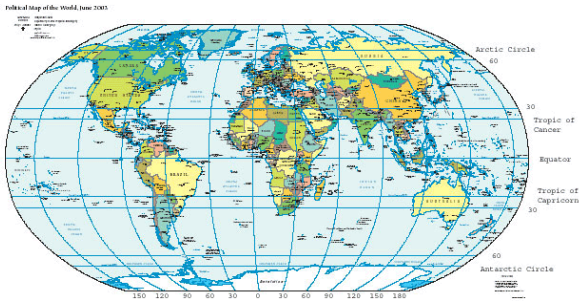
— Les GCS permettent de situer un point sur terre (ou un autre corps céleste) dans un modèle en trois dimensions, le géo-positionnement par satellite (GPS) est probablement le plus connu de ces systèmes.

— Les PCS servent à positionner un point sur une projection bidimensionnelle.

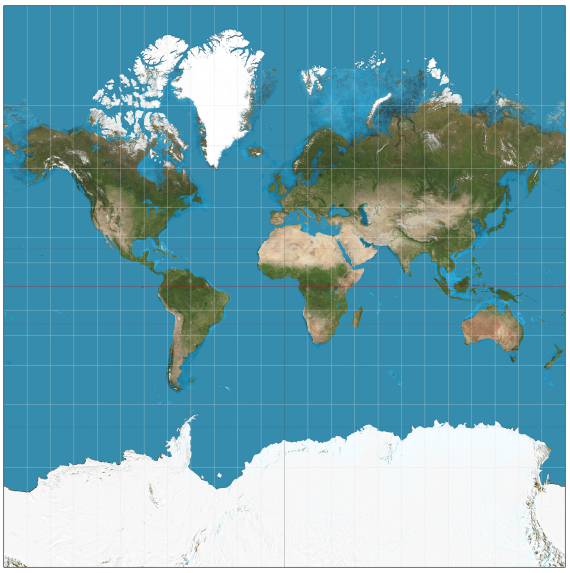
Vue l'impossibilité d'aplatir une surface courbe sans la déformer, les PCS sont nécessaires pour dessiner des cartes. Les particularités locales du relief nécessitent également une multitude de systèmes de projection différents pour bien représenter différentes régions.



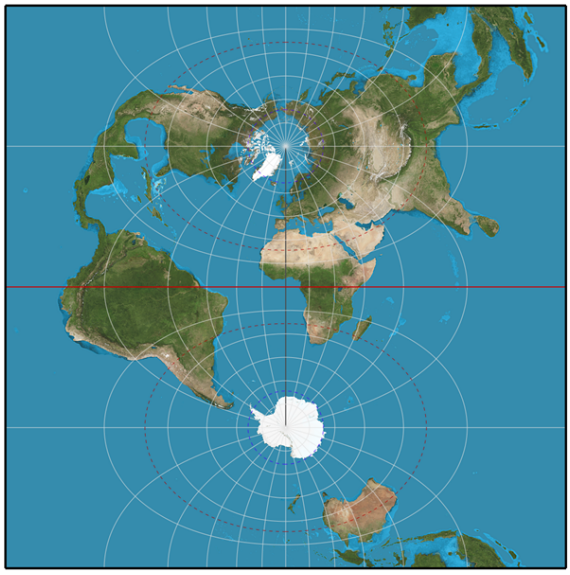
(a) Système géographique (Djexplo, 2019)



(b) Projection type Robinson (CIA et Niteowlneils, 2006)



(c) Projection type Mercator (Strebe, 2011)



(d) Projection type Mercator Transverse (Mercator, 2010)

FIGURE 2.6 : Visualisation de systèmes de coordonnées

La province du Québec, à elle seule, est couverte par 652 PCS qui lui sont spécifiques, sans compter les systèmes désuets (Klokan Technologies GmbH, 2022a). Le fichier de polygones (contient les ensembles de coordonnées correspondant aux sommets de chaque polygone représentant un objet) de l'ensemble de données indique que le PCS utilisé est EPSG : 3857. Il s'agit d'un système de type pseudo-Mercator utilisé par la plupart des services de cartes en ligne comme *Google Maps*, *OpenStreetMap* et *Bing* pour sa capacité à représenter toute la terre sauf près des pôles (Klokan

Technologies GmbH, 2022b). Une projection de type Mercator est une projection cylindrique, comme si on déroulait le globe en étirant le haut et le bas pour en faire un rectangle (voir sous-figure 2.6(c)).

L'inventaire des arbres de Montréal utilise le GPS, il est donc nécessaire de faire une conversion des coordonnées pour trouver ses arbres dans les images. Pour faire cette conversion, on doit appliquer une opération linéaire aux coordonnées de base qui dépendent des systèmes de coordonnées d'entrée et de sortie. Heureusement, il existe déjà des bibliothèques pour faire ce genre de tâches. En python, on peut utiliser le module *Transformer* de la bibliothèques *pyproj* (voir l'algorithme 2.1).

```python
from pyproj import Transformer

# Retourne un couple (latitude, longitude) dans le systeme de coordonnees out_crs
#  a partir des coordonnees fournies dans le systeme in_crs
# CRS est un objet representant un systeme de coordonnees
def conversion_coordonnees(latitude, longitude, in_crs, out_crs):
    transformer = Transformer.from_crs(in_crs, out_crs)
    return transformer.(transform(latitude, longitude))
```

Algorithme 2.1 : Conversion de coordonnées en Python

Nous avons manqué de temps pour le faire, mais, à l'avenir, il serait possible d'améliorer cet ensemble de données en passant d'abord les photos dans un outil de segmentation capable de repérer les arbres. Cela permettrait de sélectionner plus exactement les sujets et d'éliminer les autres arbres. Cela permettrait également d'éliminer les images qui ne contiennent pas d'arbres comme de la sous-figure 2.4(d), probablement causées par des erreurs dues à des différences entre la réalité et les données prises dans l'inventaire ou la mort prématurée d'un arbre. En ajoutant ces informations aux inventaires et le géoréférencement, on obtiendrait certainement un ensemble de données de meilleure qualité.

2.2.4 Ensemble Map

Les sources d'images aériennes les plus accessibles au public sont probablement les services de cartographie en ligne tels que *Google Map* (Google, 2023) ou *OpenStreetMap* (OpenStreetMap, 2023). Un ensemble de données devait donc être créé pour tester si de telles images permettent une classification des arbres.

Pour ce faire, il a été décidé d'utiliser l'API de *Mapbox* (mapbox, 2023) afin d'obtenir des images

aériennes. Les images fournies par *Mapbox* viennent de sources variées, certaines sont ouvertes, d'autres sont commerciales. Ces images sont disponibles sous le format de tuiles *raster*, des images de forme carrée très utilisées pour générer des cartes en ligne. Pour une échelle donnée z, le monde sera découpé selon une mosaïque de 2^z tuiles horizontalement et verticalement, pour un total de 2^{2z} tuiles. Chaque tuile est identifiée par sa position dans la mosaïque ainsi que son échelle. La figure 2.7 illustre ce système pour une échelle de $z = 2$.

FIGURE 2.7 : Schéma de numérotation pour carte en ligne en mosaïque (OpenStreetMap contributors, 2020)

Ce système régulier de découpage permet d'identifier quelle tuile contient une coordonnées géographique de façon fiable en utilisant les équations 2.1 et 2.2 pour obtenir sa position (x, y) selon l'échelle z utilisée.

$$x = \left\lfloor \frac{longitude + 180}{360} \cdot 2^z \right\rfloor \tag{2.1}$$

$$y = \left\lfloor \left(1 - \frac{\ln\left(\tan\left(latitude \cdot \frac{\pi}{180} \right) + \frac{1}{\cos\left(latitude \cdot \frac{\pi}{180} \right)} \right)}{\pi} \right) \cdot 2^{z-1} \right\rfloor \tag{2.2}$$

Il a été décidé d'utiliser deux échelles pour construire cet ensemble de données : 17 et 18. À ces échelles, une tuile *raster* à la latitude de Montréal couvre une surface carrée dont les côtés mesurent approximativement 86 et 21 mètres respectivement. La figure 2.8 montre une tuile *raster* à l'échelle 17 qui contient, entre autres, des sorties de la station de métro Berri-UQAM à Montréal.

FIGURE 2.8 : Exemple de tuile *raster* de *Mapbox* à l'échelle 17

Puisque l'on veut extraire des vignettes centrées sur des arbres à identifier, il est nécessaire de géoréférencer les tuiles obtenues. Pour cet ensemble, la tâche est bien plus simple que pour l'ensemble Sylphe puisque les coins des tuiles sont des points de contrôle exacts. On obtient les coordonnées du coin supérieur gauche en isolant la longitude dans l'équation 2.1 et la latitude dans l'équation 2.2. On obtient ainsi les équations 2.3 et 2.4. On peut calculer les coordonnées des autres coins en utilisant ces équations sur les tuiles adjacentes : $(x+1, y)$, $(x, y+1)$ et $(x+1, y+1)$.

$$longitude = \frac{x}{2^z} \cdot 360 - 180 \tag{2.3}$$

$$latitude = \arctan\left(\sinh\left(\pi - \frac{y}{2^z} \cdot 2\pi\right)\right) \cdot \frac{180}{\pi} \tag{2.4}$$

Une fois les images géoréférencées, il est possible de découper les vignettes centrées sur les arbres

à identifier. On peut voir des exemples du résultat final de ces opérations dans les figures 2.9 et 2.10 pour les échelles 17 et 19 respectivement. Puisqu'il s'agit d'un ensemble de données destiné à des expériences préliminaires, il a été divisé selon le rang taxonomique pertinent au domaine le plus grossier : la classe. Ce qui donne trois classes : Ginkgoopsida, Magnoliopsida et Pinopsida ; qui correspondent, respectivement, aux Ginko biloba, aux feuillus et aux conifères.

(a) Conifères (b) Feuillus

(c) Ginko biloba

FIGURE 2.9 : Exemples d'images d'arbres extraites de *Mapbox* à l'échelle 17

(a) Conifères (b) Feuillus

(c) Ginko biloba

FIGURE 2.10 : Exemples d'images d'arbres extraites de *Mapbox* à l'échelle 19

Cet ensemble de données a l'avantage d'être facile à étendre si on le désire. Comme on peut le voir dans les figures 2.9 et 2.10, la résolution est toutefois basse. Il est aussi important de noter que les tuiles *raster* sont des images composites créées pour les besoins des cartes en ligne qui ne correspondront peut-être pas aux besoins de la classification d'images. C'est pour ces raisons que

ce petit ensemble de données a été créé, pour étudier si ce type d'images peut être utilisé dans ce cadre.

2.3 Conclusion

Finalement, avec l'augmentation du nombre de villes partenaires et le partage de leurs inventaires, il serait maintenant pertinent d'élargir la recherche d'images aériennes à de nouveaux territoires. Il a également été possible de constater qu'il n'y a pas de données de bonne qualité disponibles pour les objectifs d'identification d'arbres par des photos aériennes. Il a été nécessaire de faire beaucoup de travail pour créer l'ensemble Détection d'arbres à partir des airs et sa qualité n'est pas au niveau espéré. Il sera donc pertinent d'acquérir ces données dans l'avenir en faisant l'aquisition d'un drone par exemple. Cela permettra d'améliorer les données disponibles de deux façons : choisir les conditions d'acquisition (saison, résolution, altitude, ...) et d'avoir toutes les métadonnées dans les fichiers d'images (le géoréférencement principalement).

Nous pouvons donc maintenant passer à la suite qui expliquera comment nous comptons nous y prendre pour intégrer ces nouvelles données pour des travaux futurs. Il y sera aussi présenté les travaux qui utiliseront les données de ce chapitre.

	Macro	Horizon	Sylphe	Map (17/19)
Nombre d'éléments	57 914	58 521	738 987	2037 / 822
Nombre de classes	170	259	152	3
Source	Pl@ntNet (Affouard *et al.*, 2020)	iNaturalist (iNaturalist Contributors, 2022)	Photothèque de Montréal (Alta Photo, 2002)	*Mapbox* (mapbox, 2023)
Licence	CC BY-SA 4.0	CC BY-SA 4.0	CC BY-SA 4.0	ODbL v1.0

Tableau 2.1 : Propriétés des ensembles de données

CHAPITRE 3
MÉTHODOLOGIE

Dans ce chapitre, nous commencerons par présenter de la gestion des données à la section 3.1. Il s'agit d'un sujet important compte tenu du nombre d'ensembles de données déjà présents, mais surtout de la quantité de données qui ont besoin d'être gérées dans un service en ligne accessible à tous. La section 3.2 abordera ensuite les modèles d'apprentissage profond *VGG* et *ResNet* qui seront utilisés pour nos travaux. Nous y aborderons aussi les transformations appliquées aux images pour augmenter nos données ainsi que les détails d'implémentation des modèles et d'exécution de notre code. Les sections 3.3 et 3.4 présenterons ensuite des concepts motivant deux expériences et la méthodologie suivie pour les mener. Finalement, la section 3.5 détaillera les métriques utilisées pour l'analyse des résultats de nos expériences.

3.1 Gestion des données

Quelques centaines de Giga octets de fichiers peuvent être hébergées sur une seule machine moderne sans problèmes. Un ensemble de données dans lequel chaque donnée ne peut appartenir qu'à une seule classe se gère facilement en utilisant le système de fichier. Ces caractéristiques ne s'appliquent toutefois pas à ce que nos données sont appelées à devenir. Puisque les travaux présentés dans ce document s'inscrivent dans un projet de gestion des forêts urbaines qui se poursuivra bien après sa rédaction, il est nécessaire de prévoir pour ce futur afin qu'il soit possible de continuer à le bâtir.

Comme mentionné dans le chapitre précédent, il sera nécessaire d'ajouter de nombreuses données. Acquérir des images aériennes de tout le territoire de Montréal et des autres villes partenaires créera une grande quantité de données. Cette quantité de données sera d'autant plus grande que c'est une tâche qui devra être répétée à différents moments durant l'année pour tenir compte des différences saisonnières. En utilisant nos outils, les images soumises par les personnes utilisatrices seront aussi ajoutées aux données. Tous cela exige que la solution d'hébergement ait la possibilité de grandir en fonction du volume de données qu'elle doit contenir. Il faut aussi qu'elle soit capable de fournir les données dans un délai raisonnable, possiblement à une multitude de requêtes simultanées.

Toutes ces données doivent aussi être accessibles selon de nombreux critères. Il pourrait être demandé

d'accéder à des données pour certains mois de l'année par exemple, ou seulement des images de feuilles par exemple. Il devrait aussi être possible d'obtenir les données selon des classes générées automatiquement selon le rang taxonomique (p. ex. la classe pour différencier des conifères de feuillus, plutôt que l'espèce).

Pour répondre à ces besoins, deux outils avec lesquels nous avons pu nous familiariser dans le cadre de nos études ont été choisis. Le premier, *Hadoop* (Apache, 2023a), résout le problème du volume des données et de leur accès par plusieurs utilisateurs en parallèle. Le second, *Solr* (Apache, 2023d), nous permettra d'organiser nos données efficacement.

3.1.1 *Hadoop*

Hadoop est une suite de logiciels libres écrite en java et faite pour travailler avec des données massives. Elle permet de gérer et traiter des données distribuées sur différents noeuds, chaque noeud regroupant plusieurs machines standard. Il est possible d'échelonner facilement une installation, sur un seul noeud ou des milliers sans effort supplémentaire significatif. Deux logiciels sont au coeur du fonctionnement d'*Hadoop* : *Hadoop Distributed File System* (HDFS) (Apache, 2023b) et *MapReduce* (Apache, 2023c).

HDFS gère le stockage des données. Les fichiers sont fractionnés en blocs et distribués sur les différents noeuds. HDFS assure également une robustesse des données en créant une redondance. Il gère également automatiquement l'accès aux données durant des situations problématiques comme les pannes, de façon à permettre une expérience sans tracas à l'utilisateur.

MapReduce est le module original de traitement des données. Il fonctionne en deux étapes principales : le *Map* et le *Reduce*. Durant l'étape de *Map*, le problème est divisé en sous-problèmes, puis envoyé à différents noeuds. Les noeuds recevant un sous-problème peuvent, eux-mêmes, le diviser en problèmes encore plus petits. Éventuellement, les sous-problèmes arrivent à un point où il n'est plus pratique de les diviser et sont résolus par les noeuds qui les reçoivent. Le système peut également redémarrer des noeuds ou transférer la tâche à un autre noeud en cas d'erreur. L'étape de *Reduce* fait alors remonter les résultats, calculant un résultat partiel, à chaque étape, jusqu'à ce que le noeud d'origine reçoive les résultats partiels et calcule le résultat final du problème d'origine.

La suite *Hadoop* comprend de nombreux autres logiciels servant des besoins spécialisés ou destinés à améliorer les performances dans certaines situations. L'un d'eux mérite d'être mentionné, *Spark* (Apache, 2018), qui est de plus en plus utilisé aux dépends de *MapReduce*. Il sert, lui aussi, à faire des calculs distribués, mais est beaucoup plus rapide, jusqu'à 300 fois plus dans certains tests.

Dans le cadre de ce mémoire, seul HDFS est d'intérêt. Les ensembles de données seront enregistrés sur ce système de fichiers afin de pouvoir plus facilement gérer leur croissance future. L'objectif de ces recherches étant de fournir des outils utiles au public, il sera alors possible d'augmenter les ensembles de données avec les images fournies par les utilisateurs. Il est alors probable que les capacités d'un seul serveur seront insuffisantes autant pour le stockage des données que pour le calcul nécessaire pour gérer les opération sur le système de fichier, une solution distribuée pourrait alors être nécessaire.

3.1.2 *Solr*

Solr est un autre logiciel libre en java. Il s'agit d'un logiciel de recherche utilisant *Lucene* (Apache, 2023e) comme moteur de recherche ; une bibliothèque libre, également écrite en Java, mais aussi disponible en d'autre langages. *Solr* permet l'indexation de documents contenant des données structurées ou du texte. Il est possible de fournir un schéma d'indexation à *Solr* afin d'améliorer ses performances de recherche si l'on connaît à l'avance le type de requêtes qui seront utiles. Il est ensuite possible de faire des recherches dans les documents en faisant des requêtes à *Solr* en utilisant des couples clef-valeur si l'on cherche dans des données structurées, ou en plein texte. Il est aussi possible de faire des recherches incluant plusieurs critères et *Solr* supporte les suggestions asynchrones. Finalement, *Solr* peut retourner les résultats en les ordonnant par pertinence.

Tout comme les logiciels de la suite *Hadoop*, *Solr* peut être distribué. Il est également facile d'augmenter le nombres de machines sur lesquelles une instance de *Solr* est exécutée grâce au concept de *shards*. Chaque *shard* est une partition horizontale du moteur de recherche lui permettant de distribuer ses données et ses calculs. Une partition horizontale se fait en séparant les données en fonction des entrées (correspondant à des rangées - horizontales donc - dans la représentation classique d'une base de données relationnelle), ce qui permet de réduire le nombre d'index dans chaque *shard*. Des partitions verticales se sépareraient plutôt les documents entre elles ; contenant ainsi, chacune, tous

les index, mais seulement les résultats associés aux documents qu'elles connaissent.

L'utilisation de *Solr* pour indexer les ensembles de données utilisés permettra de créer automatiquement des sous-ensembles pertinents à certaines expériences en fournissant des critères. Il sera ainsi possible, sans opérations manuelles, de changer les classes (selon le rang taxonomique par exemple) ou d'inclure des images selon les besoins (vues aériennes ou au niveau de la rue, emphase sur une partie de l'arbre, résolution, ...). L'utilisation d'une solution comme *Solr* plutôt qu'une base de données est justifiée de façon similaire HDFS, la quantité de données à gérer est grande et il est attendu qu'elle croisse et devienne massive.

3.2 Modèles utilisés

3.2.1 Transformations

Avant de servir à l'entraînement du modèle, les images subissent quelques transformations. Elles sont toutes redimensionnées et leurs couleurs normalisées. Lors de l'entraînement, elles sont en plus légèrement rognées aléatoirement et, la moitié du temps, renversées horizontalement. Les images aériennes sont également renversées verticalement une fois sur deux.

3.2.2 Les modèles choisis

VGG

VGG (Simonyan et Zisserman, 2014) est le résultat d'efforts pour approfondir les réseaux convolutifs puisque plus de couches améliorent habituellement les performance d'un réseau de neurones. Alors que les modèles avant VGG étaient composés d'une couche convolutive pour chaque couche de *pooling*, VGG en utilise plusieurs. Chaque couche convolutive utilise un petit champs réceptif (3×3) plutôt que les champs plus grands (7×7 à 11×11) des modèles précédents.

Une succession de couches convolutives avec ces petits champs réceptifs obtiennent une couverture similaire (avec des champs « virtuels » plus grands) à une couche unique qui aurait un plus grand champs, mais utilisent un nombres de paramètres réduits. Par exemple, avec une image possédant c canaux de couleurs, 3 couches convolutives avec un champs réceptifs de 3×3 utilisent $3(3^2 \cdot c^2) = 27c^2$ paramètres alors qu'une seule couche avec un champ réceptif équivalent de 7×7 utilise $7^2 \cdot c^2 = 49c^2$

paramètres. L'utilisation d'une couche d'activation après chacune des couches convolutives améliore également les résultats en augmentant le pouvoir discriminant du modèle.

Les performances de modèles plus profonds atteignent toutefois un plateau autour de 19 couches, au delà de quoi les performances ont même tendance à se dégrader (He *et al.*, 2015a). Le prochain modèle utilisé a été créé spécifiquement dans le but de répondre à ce problème.

Resnet

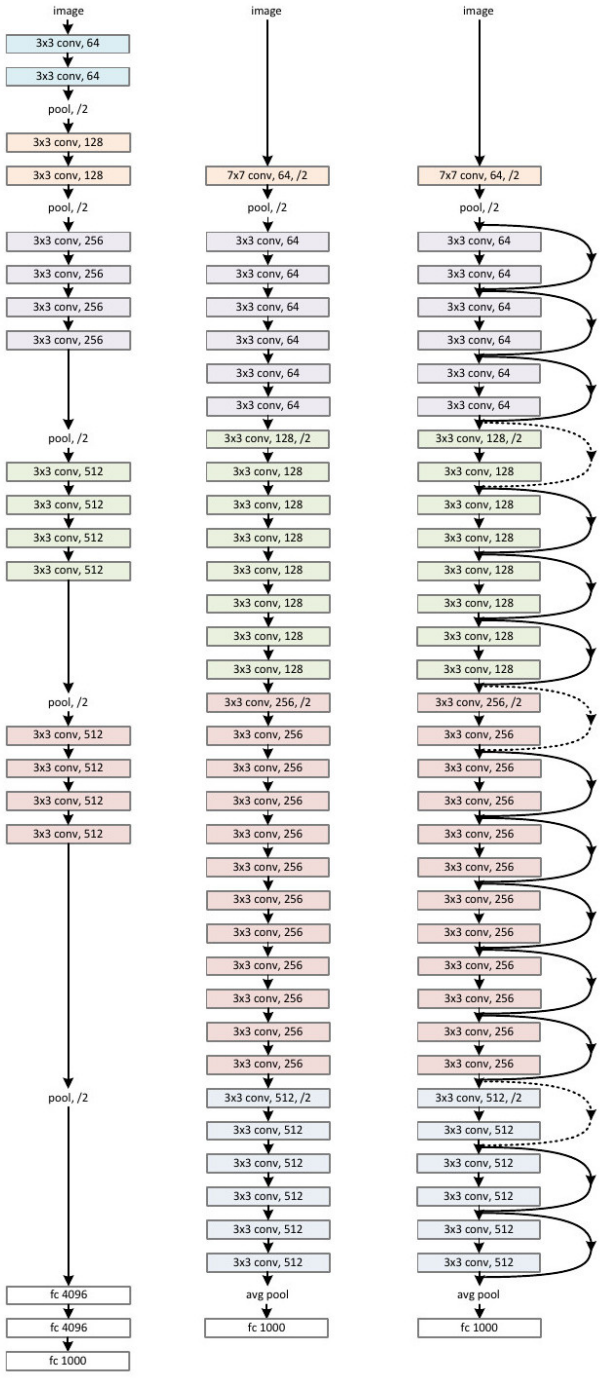
Avant la création de *ResNet* (He *et al.*, 2015a), les réseaux convolutifs les plus profonds pouvaient atteindre de 16 à 30 couches avant de voir leurs performances se dégrader (He *et al.*, 2015a). Cette dégradation est liée au fait que les sorties des couches profondes ont tendances à saturer : prendre des valeurs similaires correspondant aux valeurs asymptotiques des fonctions d'activation (fonctions non linéaires) ou nulles (ReLU). Pour permettre d'approfondir encore plus les réseaux de neurones, *ResNet* introduit l'utilisation de réseaux résiduels (*residual network*).

Comme illustré dans la Figure 3.2, un bloc résiduel reçoit une entrée x et applique une fonction $F(x)$ comme d'habitude, mais garde aussi une version de ces entrées inchangées en appliquant la fonction identité. Ces résultats sont ensuite fusionnés en les additionnant. La sortie y du bloc est donc donnée par $y = F(x) + x$.

L'utilisation de ces connexions résiduelles permet ainsi de combattre la dégradation causée par la saturation introduite par un grand nombres de couches. Les résultats sont des modèles pouvant dépasser les 100 couches, *ResNet152*, en particulier, en a 152 et bat les modèles VGG sur plusieurs ensembles de données (He *et al.*, 2015a).

3.2.3 Détails d'implémentation et exécution

Les programmes ont été écrit en Python en utilisant les bibliothèques de pyTorch (Paszke *et al.*, 2019). Les couches convolutives téléchargées ont été pré-entraînées sur ImageNet (Deng *et al.*, 2009) et les poids du classifieur ont été initialisés selon une loi uniforme $P_{i,j} \sim U(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}})$ où n est le nombre de paramètres (poids) de la couche. Les deux premiers blocs convolutifs sont fixés de façon à ce que leurs paramètres ne soient pas modifiés lors de l'entraînement puisque ces blocs sont déjà



(a) VGG-19 (b) Resnet-34 sans réseaux résiduels (c) Resnet-34 avec réseaux résiduels

FIGURE 3.1 : Structure des modèles VGG et Resnet (He *et al.*, 2015a)

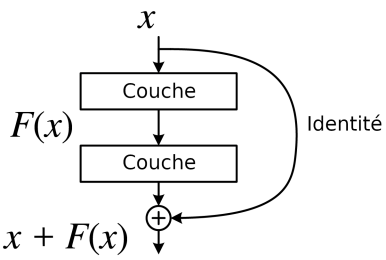


FIGURE 3.2 : Visualisation d'un bloc résiduel sautant 2 couches (LunarLullaby, 2015)

entraînés à détecter des attributs de bas niveau communs à toutes les images. Les blocs suivants subissent l'entraînement normalement afin de permettre leur spécialisation à nos domaines.

Pour entraîner les modèles, l'entropie croisée a été choisie comme fonction objectif à minimiser. Cette fonction permet de mesurer la différence moyenne entre deux distributions de probabilités. Dans un problème de classification, les vecteurs de prédictions constituent les distributions de données et l'entropie croisée d'un vecteur sera plus basse si ses prédictions sont plus exactes.

Afin de minimiser cette fonction, l'algorithme du gradient stochastique (*stochastic gradient descent*) a été retenu. Cet algorithme calcule un gradient unitaire indiquant la direction dans laquelle modifier les paramètres pour maximiser la fonction objectif, le gradient négatif permettant de la minimiser (Goodfellow *et al.*, 2016, chap 4.3). Avant de modifier les paramètres en fonction de ce gradient, il est ajusté. Une valeur trop grande risque d'empêcher la convergence en tournant autour du minimum sans pouvoir s'en approcher suffisamment. Une valeur trop petite ralentit la convergence et risque de piéger le modèle dans un minimum local loin du minimum global, car il devient impossible de sauter les maximums locaux. Un rythme d'apprentissage initial de 0,001 multiplie donc ce gradient. Afin de s'approcher plus rapidement du minimum et augmenter les chances de sauter les maximums locaux lorsque le gradient pointe la même direction, on lui additionne une partie de la dernière modification déterminée en la multipliant par une valeur d'inertie de 0,9. Finalement, toutes les sept époques d'entraînement, ces valeurs sont réduites de 90% (valeur gamma de 0,1) afin d'affiner la recherche à mesure que l'on approche d'une solution optimale.

Régularisation

Un problème commun lorsque l'on entraîne de grands modèles est qu'il leur est possible de s'ajuster fortement aux données d'entraînement sans améliorer leur capacité de généralisation à de nouvelles données. Afin de limiter ces risques de sur-ajustement, de nombreuses techniques de régularisations ont été développées. Ces techniques ont pour objectif de réduire le taux d'erreurs pendant la phase de tests, au risque de l'augmenter à l'entraînement (Goodfellow *et al.*, 2016, chap 7). Nos modèles ne font pas exception et utilisent plusieurs de ces techniques qui sont décrites ici.

Une technique particulièrement simple que nos modèles utilisent, le *early stopping*, vient de l'observation de l'évolution des taux d'erreurs lors de l'entraînement. Au début, les taux d'erreurs des phases d'entraînement et de validation diminuent tous les deux, mais après un certain temps, le taux d'erreurs des tests se met à augmenter alors que celui de l'entraînement continue à diminuer (Goodfellow *et al.*, 2016, chap 7.8). Arrêter prématurément l'entraînement lorsque l'on détecte ce phénomène permet de sélectionner les paramètres lorsque leur capacité de généralisation est maximale.

La prochaine technique, introduite par Hinton *et al.* (2012), le décrochage (*dropout*), consiste à éliminer temporairement des neurones en mettant leur valeur à zéro. L'effet est d'entraîner un réseau de neurones réduit, et différent lors de chaque passe. Le choix des neurones à éliminer est un processus stochastique dans lequel chaque neurone a une probabilité p d'être conservé. Outre la régularisation obtenue par l'ajout de bruit, le décrochage prévient la co-adaptation ; situation dans laquelle le comportement de groupes de neurones devient fortement corrélé. Le décrochage régularise également les modèles en faisant un processus similaire à l'agrégation de nombreux modèles plus petits (Srivastava *et al.*, 2014). Tel que conseillé par Srivastava *et al.* (2014), nos modèles utilisent une probabilité p de 0,5 pour les neurones des couches cachées et de 0,8 pour les neurones de la couche d'entrée.

Finalement, les modèles sont entraînés sur des noeuds de calcul GPU nVidia K80 via les serveurs rendus accessibles par Calcul Québec. Chacun de ces noeuds est composé de 12 GPU avec 12 Go de mémoire par GPU. Avec de tels GPU, il a été possible d'utiliser des lots de 256 images à chaque passe d'entraînement.

3.3 Détection d'anomalies

Il existe de nombreux objectifs pour lesquels on pourrait vouloir détecter des anomalies dans les prédictions d'un modèle. Plusieurs de ces objectifs et leurs solutions ont été explorées dans la section 1.3. Dans le contexte où un utilisateur peut reprendre une photo du même sujet, il est intéressant de pouvoir détecter si la prédiction semble peu fiable et d'en avertir l'utilisateur. Il peut alors prendre une nouvelle photo sous un angle différent ou avec un différent centre d'intérêt (une feuille ou une fleur par exemple). Il s'agit d'une situation classique de classification avec option de rejet dans laquelle les données sont entraînées et testées sur des classes connues. Le seul objectif de l'option de rejet est donc d'identifier et rejeter les prévisions peu fiables.

Il existe déjà plusieurs solutions à ce problème, mais elles nécessitent des informations acquises lors de l'entraînement ou des mécanismes inclus dans le modèle. La solution envisagée ici s'inspire de Hang et Aono (2016), mais prendra une forme lui permettant d'être ajoutée facilement à n'importe quel classifieur. À l'instar de Hang et Aono (2016), un score minimal sera déterminé pour décider si une prévision sera considérée fiable. Plutôt que de déterminer ce score à l'entraînement, un test statistique sera utilisé directement sur les vecteurs de prédictions afin de déterminer le seuil minimal pour ce vecteur. Soit le vecteur $\mathbf{r}$ contenant les prédictions du classifieur pour chacune des n classes. Supposant que le modèle n'a pas encore été entraîné, ses paramètres sont des variables aléatoires. Chaque valeur du vecteur $\mathbf{r}$ est donc la somme de variables aléatoires dont la distribution est similaire. Ces conditions permettent d'appliquer le théorème centrale limite et d'affirmer que la distribution des prédictions suivra une loi normale dont la moyenne et l'écart-type peuvent être calculés à partir des distributions des paramètres du classifieur. Ce théorème stipulant que, sous certaines conditions, la somme d'un grand nombre de variables aléatoires indépendantes suit approximativement une loi aléatoire normale (Alalouf *et al.*, 2002).

L'entraînement du modèle est sensé retirer l'aspect aléatoire des paramètres du modèle et donner une fonction permettant de classifier nos données. L'idée d'un test statistique est de tester que l'entropie a été réduite au point où la prévision est bien le résultat de cette fonction. On fera donc l'hypothèse que les prédictions suivent toujours une distribution aléatoire de loi normale et un test sera appliqué afin de vérifier si cette hypothèse peut être rejetée. La distribution des paramètres après entraînement étant inconnue, la moyenne et l'écart-type de la loi normale hypothétique seront estimées empiriquement en calculant la moyenne et l'écart-type du vecteur $\mathbf{r}$.

Pour décider si la classe (i) ayant reçu le score le plus élevé doit être gardée ou rejetée, on fait donc un test d'hypothèse. Puisque l'on veut tester si $\mathbf{r}_i$ est suffisamment élevée pour être significative, on opte pour un test unilatéral à droite. L'hypothèse nulle H_0 est donc que la valeur de $\mathbf{r}_i$ n'est pas plus élevée que la moyenne de toutes les valeurs de $\mathbf{r}$. Si l'hypothèse nulle ne peut être rejetée, on considère que la prédiction n'est pas fiable et on la rejette.

Si l'hypothèse nulle est rejetée, on fait le même test sur les autres valeurs. Si une ou plusieurs autres valeurs permettent de rejeter l'hypothèse nulle, il y a ambiguïté dans les résultats et on rejette la prédiction. Si la prédiction n'a toujours pas été rejetée à ce point, on la considère alors suffisamment fiable pour la garder.

Puisque que la prédiction du modèle suit une loi normale lorsque aléatoire, sa fonction de densité est donnée par l'équation 3.1.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{3.1}$$

$$S(x) = P(X > x) = \frac{1}{\sigma\sqrt{2\pi}} \int_x^\infty e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \tag{3.2}$$

$$S^{-1}(p) = c \tag{3.3}$$

Les fonctions f, S et S^{-1} sont, respectivement, fonction de densité, fonction de survie et l'inverse de S. X représente la variable aléatoire, σ et μ sont l'écart-type et la moyenne, c est une valeur critique et p est la valeur-p.

Puisque le test est unilatéral à droite, on rejettera H_0 si la probabilité qu'une prédiction aléatoire X soit plus grande que la prédiction $\mathbf{s}_i$ est plus petite ou égale à une certaine probabilité critique p telle que donnée par la fonction de survie 3.2. En prenant $\mathbf{r}_i$ pour l'équation de survie, on rejettera donc H_0 si $S(\mathbf{r}_i) < p$.

Cette expérience devant être répétée pour chaque valeur du vecteur $\mathbf{r}$, on calculera plutôt une seule fois une valeur critique (c), pour une valeur déterminée de p, au delà de laquelle une valeur de $\mathbf{r}$ devrait être rejetée. Cette valeur est obtenut par l'inverse de la fonction de survie (équation 3.3). L'algorithme 3.1 permet d'obtenir les index des prédictions, contenues dans le vecteur *confidences* $(\mathbf{r})$, pour lesquelles H_0 doit être rejetée pour un p donné.

```python
from math import sqrt
from scipy import stats

def get_significant_indexes(confidences, p=0.05):
  mean = sum(confidences)/len(confidences)
  var = sum([ (x-mean)**2 for x in confidences ]) / (len(confidences)-1)
  # 'C' est une valeur critique au dela de laquelle
  # on doit doit rejeter l'hypothese nulle
  C = stats.norm.isf(p, loc=mean, scale=var)
  return [ i for i,x in enumerate(confidences) if x>C ]
```

Algorithme 3.1 : Trouver les prédictions pour lesquelles l'hypothèse nulle doit être rejetée

3.4 Identification depuis les airs

À ce point, nos partenaires du département des sciences biologiques nous ont informés qu'ils aimeraient pouvoir estimer la couverture végétale offerte par différents types d'arbres. Il faudrait donc pouvoir identifier les arbres à partir d'images aériennes plutôt que de photos prises à partir du sol afin d'éventuellement être capable de segmenter une image en fonction du couvert végétal offert par ces arbres. On a pu voir à la section 1.4 qu'il existe déjà des méthodes efficaces pour détecter la couverture des arbres à partir de photos aériennes. Puisque nous voulons aller plus loin et identifier ces arbres, il sera nécessaire de tester d'autres approches. L'expérience présentée ici a pour objectif de déterminer si un réseau de neurones recevant des images en trois bandes de couleurs (rouge, vert et bleu) peut discriminer entre différents types d'arbres.

Pour ces expériences, des images d'arbres ont été prélevées de service de cartographie en ligne à deux échelles différentes (voir section 2.2.4 : Ensemble Map). Ces images sont déjà centrées sur l'arbre à identifier dans chaque cas et couvrent une superficie devant couvrir l'arbre au complet et minimisant autant que possible la présence d'autres arbres. Pour chacune de ces expériences, le modèle *VGG19* aura pour tâche de déterminer si les images appartiennent à l'une de ces trois grandes classes du vivant : Ginkgoopsida (dont la seule espèce encore existante est le Ginkgo biloba), Magnoliopsida (feuillus), Pinopsida (connifères).

En tout, six expériences binaires auront lieu. Pour chaque expérience, un des trois types d'arbres constituera la classe à identifier (positive), les deux autres seront les images à rejeter (négatives). Les deux classes auront un nombre d'images égal, on choisira donc aléatoirement la moitié des images de chaque type négatif pour constituer la classe négative. On répétera ces expériences pour chacune des deux échelles.

3.5 Métriques utilisées

Pour pouvoir analyser le résultat d'expériences, on doit avoir des métriques objectives permettant de les comparer. On doit d'abord pouvoir qualifier les prédictions de façon à pouvoir ensuite les utiliser pour pouvoir calculer des métriques représentant la performance du modèle les ayant générées.

On évalue la qualité d'une prédiction selon deux axes. D'abord, on dira que la prédiction est positive si elle détermine que l'image appartient bien à la classe évaluée, et négative sinon. Ensuite, la prédiction sera dite vraie si elle est exacte et fausse sinon. Cela donne quatre catégories possibles résumées dans le tableau 3.1 : vrais positifs, faux positifs, vrais négatifs et faux négatifs. Si l'expérience a de multiples classes, chaque prédiction sera évaluée en fonction de chaque classe comme s'il s'agissait de plusieurs expériences binaires.

		Classe prédite	
		Classe testée	Autre
Classe réelle	Classe testée	Vrai Positif (VP)	Faux Négatif (FN)
	Autre	Faux Positif (FP)	Vrai Négatif (VN)

Tableau 3.1 : Qualité d'une prédiction

Les expériences binaires sont celles où on teste une seule classe et on cherche à savoir si l'image en fait partie ou non. Pour ces expériences, la précision et le rappel seront utilisés, car elles permettent d'évaluer facilement et significativement la pertinence des résultats. La F-mesure sera également utilisée, donnant une valeur unique résumant la précision et le rappel en donnant à chacune un poids égal.

Pour rappel :

$$precision = \frac{VP}{VP + FP} \tag{3.4}$$

$$rappel = \frac{VP}{VP + FN} \tag{3.5}$$

$$F - mesure = 2\frac{precision \cdot rappel}{precision + rappel} = \frac{2VP}{2VP + FP + FN} \tag{3.6}$$

Pour les expériences avec plusieurs classes, la proportion de prédictions exactes est ajoutée sous le

nom Top 1. Pour les usages visés par ces expériences, il n'est pas intéressant de savoir que la bonne classe était une des n premières prédictions des modèles, c'est pourquoi les scores tels que Top 5 ou autres Top n ne seront pas utilisés.

En revanche, la précision, le rappel et la f-mesure seront utilisés sous la forme de moyennes. Deux moyennes peuvent être calculées pour chacune de ces métriques : micro et macro. La micro valeur se calcule en prenant les totaux de tous les vrais positifs, faux positifs, vrais négatifs, faux négatifs. On fait ensuite le calcul habituel pour la métrique concernée avec ces valeurs. Pour la macro valeur, on calcule d'abord la valeur pour chaque classe comme si l'expérience avait été binaire, puis on fait la moyenne de ces valeurs. Les micro et macro f-mesures sont calculées normalement à partir de la précision et du rappel appropriés.

$$macro - precision = \frac{\Sigma_{i=1}^{n} precision_i}{n} \tag{3.7}$$

$$macro - rappel = \frac{\Sigma_{i=1}^{n} rappel_i}{n} \tag{3.8}$$

$$macro - f - mesure = 2\frac{macro - precision \cdot macro - rappel}{macro - precision + macro - rappel} \tag{3.9}$$

$$micro - precision = \frac{\Sigma_{i=1}^{n} VP_i}{\Sigma_{i=1}^{n} VP_i + \Sigma_{i=1}^{n} FP_i} \tag{3.10}$$

$$micro - rappel = \frac{\Sigma_{i=1}^{n} VP_i}{\Sigma_{i=1}^{n} VP_i + \Sigma_{i=1}^{n} FN_i} \tag{3.11}$$

$$micro - f - mesure = 2\frac{micro - precision \cdot micro - rappel}{micro - precision + micro - rappel} \tag{3.12}$$

Ce chapitre a montré comment nos expériences pourront être réalisées. On y a vu comment les données seront gérées et les modèles qui les utiliseront. Les expériences à mener ont également été expliquées ainsi que les métriques qui permettront d'évaluer leurs performances. Nous sommes maintenant prêts à prendre connaissance des résultats.

CHAPITRE 4
EXPÉRIENCES ET DISCUSSIONS

Les travaux présentés dans les chapitres précédents ont permis la réalisation de deux expériences. Ce chapitre en présentera les résultats et les conclusions que l'on peut en tirer. La première expérience sur les options de rejet présentée à la section 4.1 concerne nos efforts pour détecter automatiquement des prédictions peu fiables probablement fausses. La seconde, à la section 4.2, est une preuve de concept dont l'objectif est de déterminer s'il semble possible d'entraîner un modèle capable de discriminer différents types d'arbres à partir d'images prises depuis les airs.

4.1 Expérience sur les options de rejet

Pour cette expérience les variantes à 18, 50 et 152 couches du model *ResNet* ont été utilisées ainsi que *VGG* en configuration D et E (16 et 19 couches). Une couche *softmax* a également été appliquée à la sortie du classifieur.

Quatre ensembles de données ont été utilisés pour cette expérience. Deux ensembles publics, CIFAR10 et CIFAR100 (Krizhevsky *et al.*, 2009) (10 et 100 classes respectivement), ont été utilisés afin d'avoir des résultats potentiellement comparables à d'autres travaux. Pour tester si la technique utilisée est utile dans le cadre du projet, deux ensembles ont été créés à partir de l'ensemble Macro. Le premier comprend 7 classes qui sont des espèces d'érable (argenté, de Pennsylvanie, rouge, etc). Le deuxième a 21 classes choisies parmi les genres disponibles dans l'ensemble (chêne, érable, frêne, etc). Une seule espèce par genre a été utilisée pour cet ensemble. Toutes les classes de ces deux ensembles sont constituées de 500 images chacune.

Comme mentionné précédemment, un avantage de la méthode présentée est qu'elle est indépendante du modèle et il est donc inutile de refaire l'expérience pour chaque valeur-p. Une nouvelle valeur critique est simplement recalculée pour chaque valeur-p afin de rejeter les prédictions jugées non fiables. Pour chaque valeur p choisie, on calcule alors une matrice de confusion avec les prédictions qui n'ont pas été rejetées. Les données extraites à ce point consistent donc en une matrice de confusion pour chaque combinaison ensemble de données, modèle et valeur-p, ainsi qu'un compte des prédictions rejetées (divisées selon que les prédictions étaient exactes ou fausses). Les différentes métriques ont

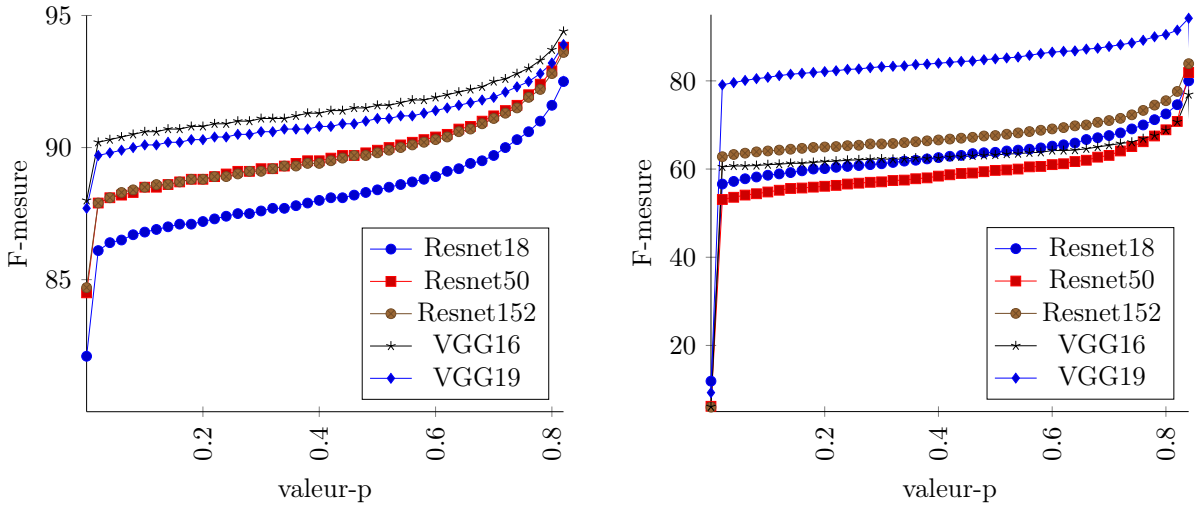
alors été calculées pour chacune de ces combinaisons. À noter que les données correspondant à une valeur p de 0 (lorsque présentes) sont en fait les valeurs brutes où aucune prédiction n'a été rejetée.

Résultats

À première vue, la méthode proposée semble prometteuse. Les graphes des figures 4.1 et 4.2 sur la mesure F montrent en effet une amélioration significative des résultats dès les valeurs p les plus faibles. Les résultats s'améliorent progressivement avec l'augmentation de la valeur p et on obtient un point d'inflexion peu surprenant lorsque près de 100% des prédictions sont rejetées quand cette valeur atteint environ 0,8. Les taux d'erreurs des prédictions présentés dans la figure 4.3 montrent la même tendance.

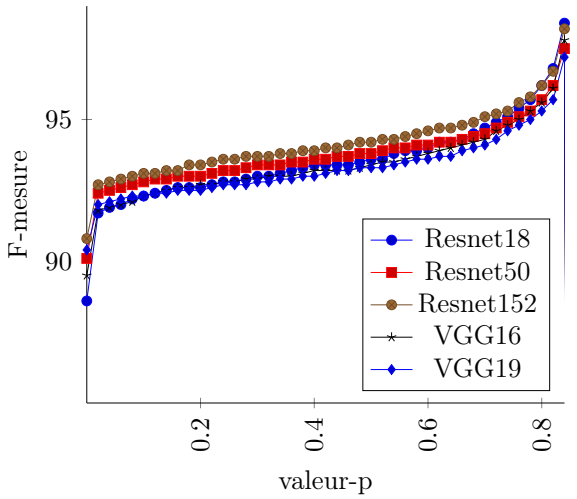
Comme vu dans la sous-section 1.3.3, l'évaluation de ce type de modèle doit tenir compte des prédictions rejetées. Les graphes des figures 4.4 et 4.5 montrent la proportion de bonnes et mauvaises prédictions qui sont rejetées afin d'obtenir les améliorations observées. Une méthode très stricte peut donc sembler extrêmement performante si on utilise ce genre de métriques.

C'est d'ailleurs ce qu'on observe dans les graphes des figures 4.1, 4.2 et 4.3 lorsque la valeur p dépasse 0,8, les prédictions deviennent particulièrement excellentes, mais c'est parce qu'on rejette la majorité des prédictions, près de 100% lorsque l'expérience est médiocre à la base comme c'est le cas sur l'ensemble de données CIFAR100. Une bonne métrique doit donc tenir compte de ces prédictions qui sont rejetées.

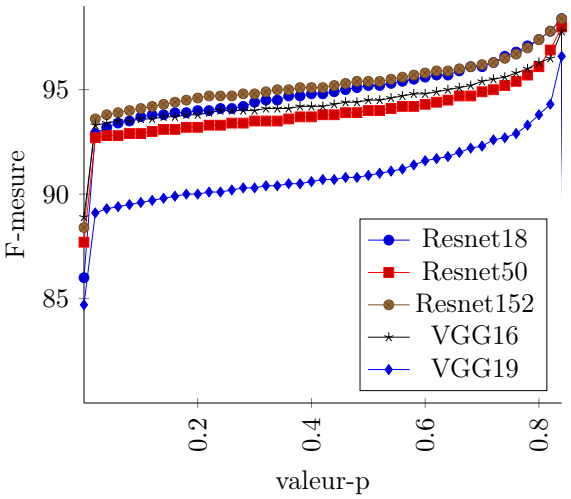


(a) CIFAR10

(b) CIFAR100

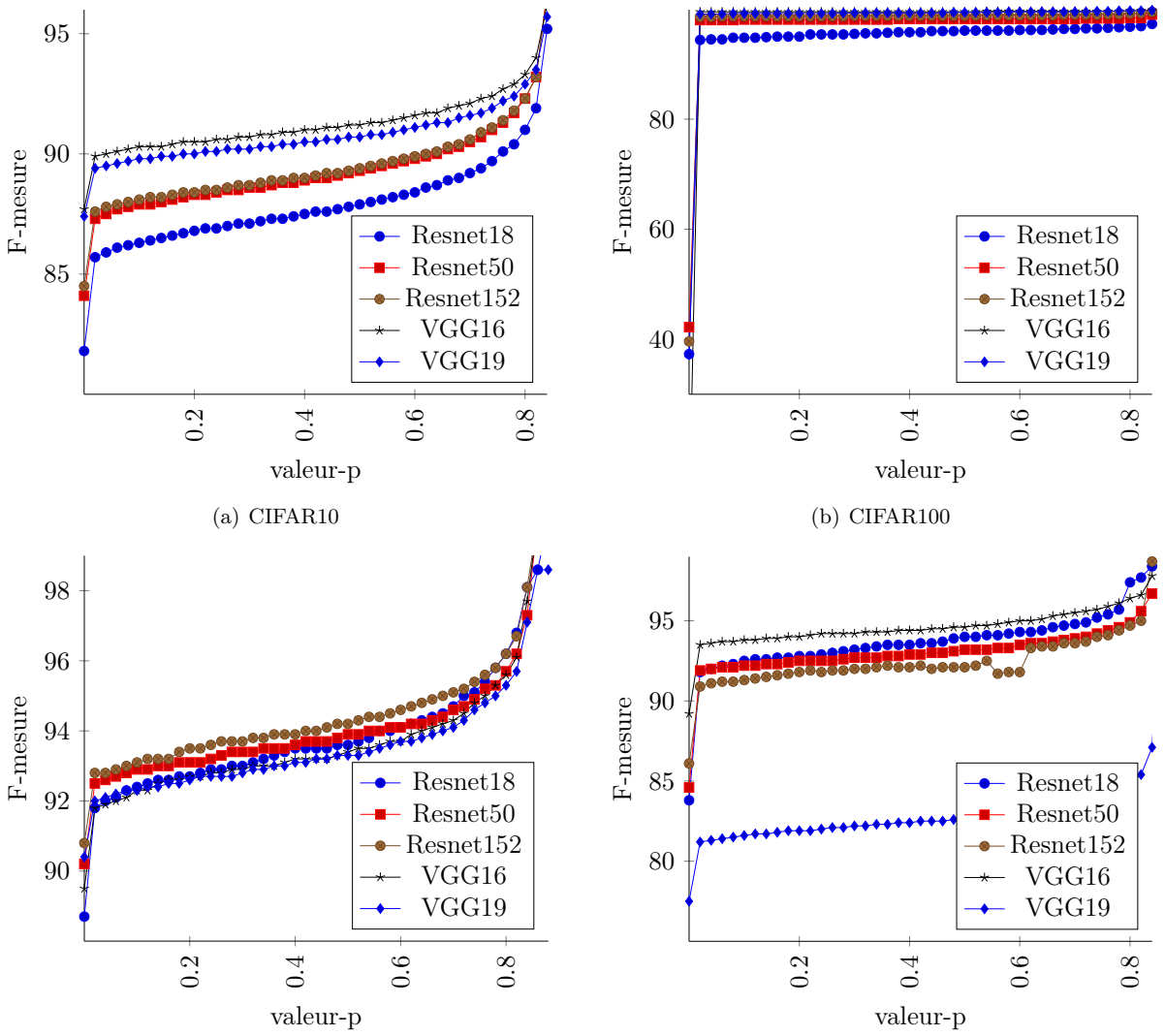


(c) Feuilles : Érables



(d) Feuilles : Genres

FIGURE 4.1 : Micro f-mesures pour différentes valeurs-p



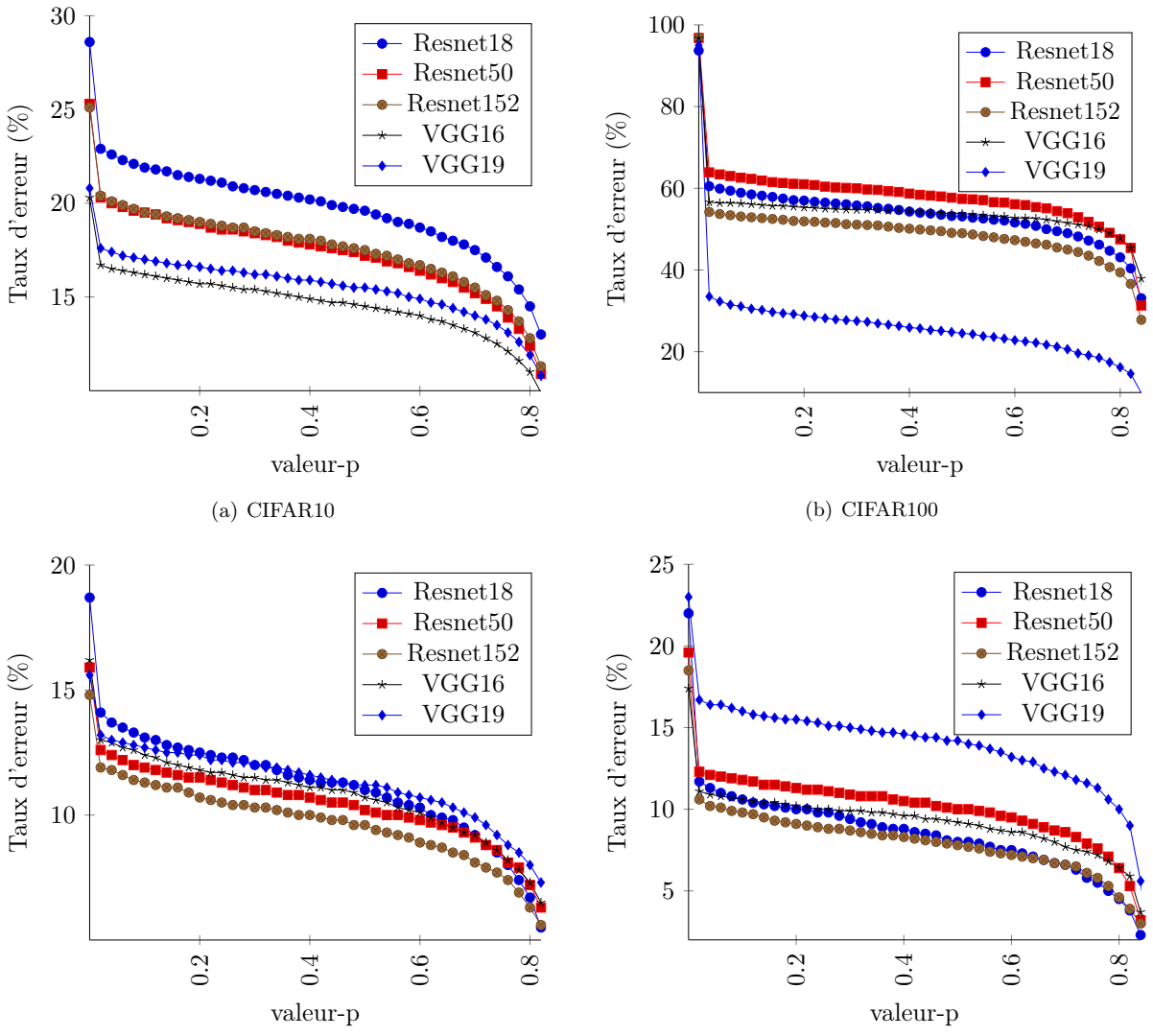
(a) CIFAR10

(b) CIFAR100

(c) Feuilles : Érables

(d) Feuilles : Genres

FIGURE 4.2 : Macro f-mesures pour différentes valeurs-p



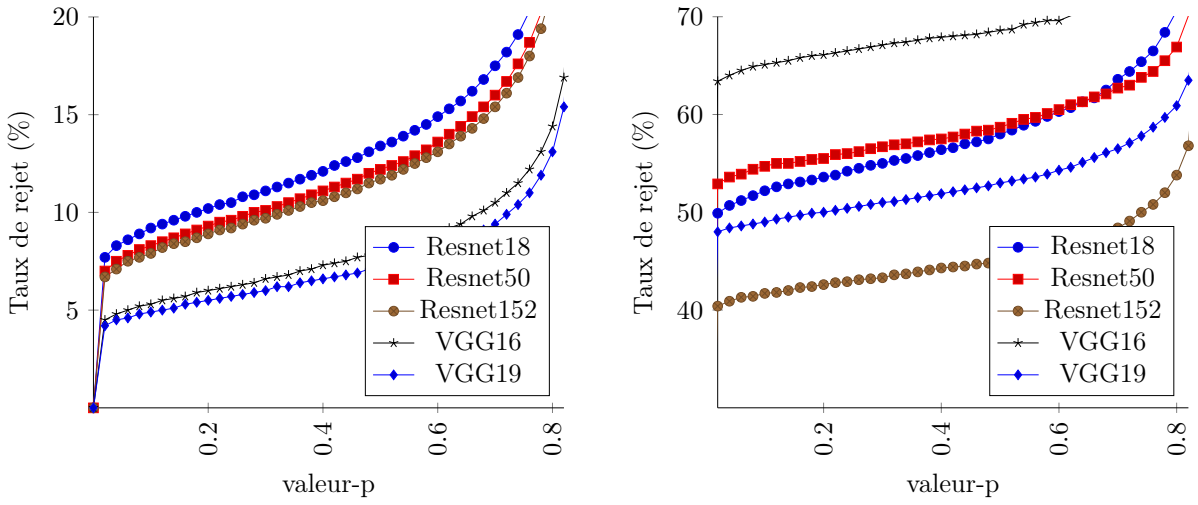
(a) CIFAR10

(b) CIFAR100

(c) Feuilles : Érables

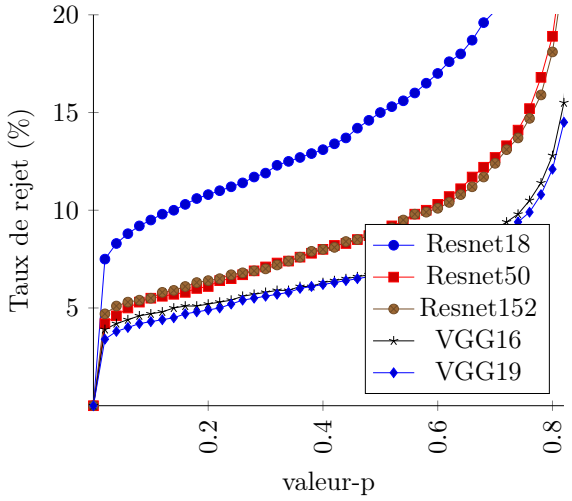
(d) Feuilles : Genres

FIGURE 4.3 : Taux d'erreur des prédictions pour différentes valeurs-p

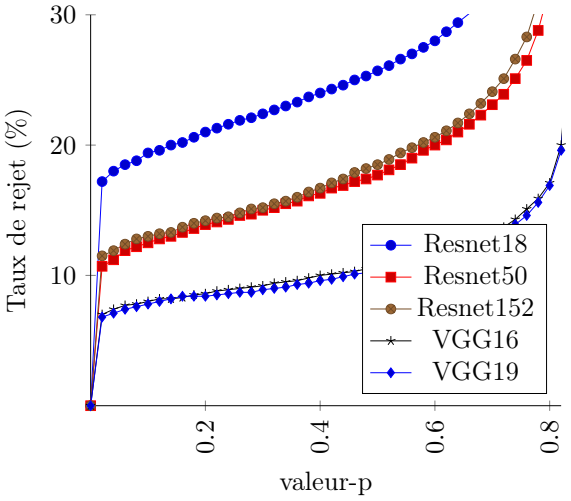


(a) CIFAR10

(b) CIFAR100

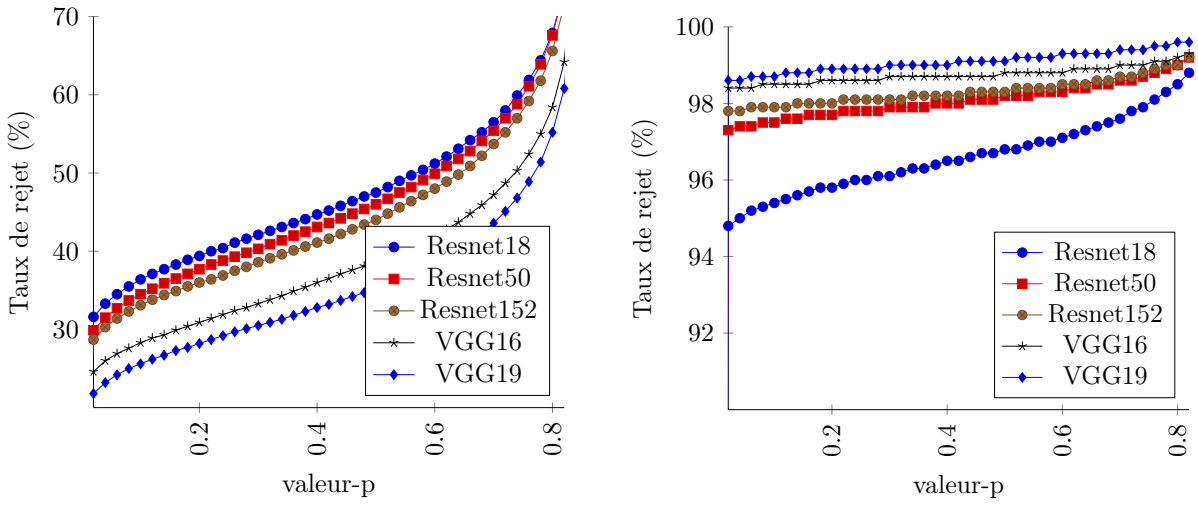


(c) Feuilles : Érables



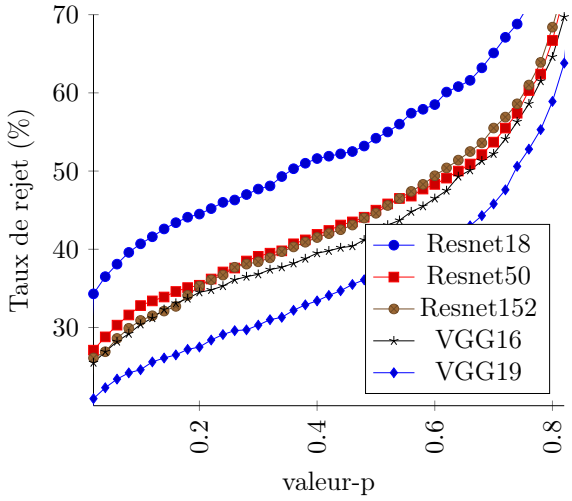
(d) Feuilles : Genres

FIGURE 4.4 : Taux de rejet des bonnes prédictions pour différentes valeurs-p

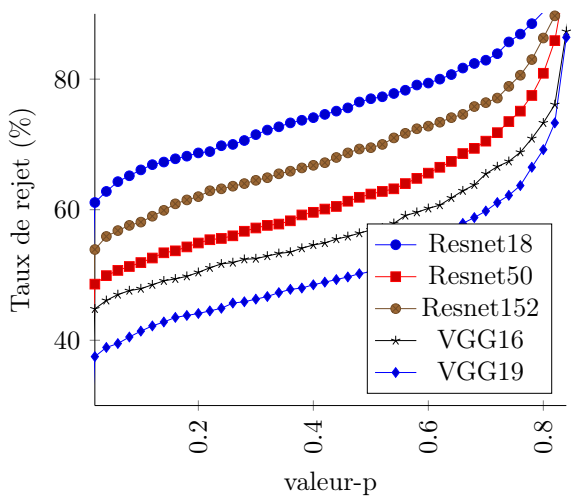


(a) CIFAR10

(b) CIFAR100



(c) Feuilles : Érables



(d) Feuilles : Genres

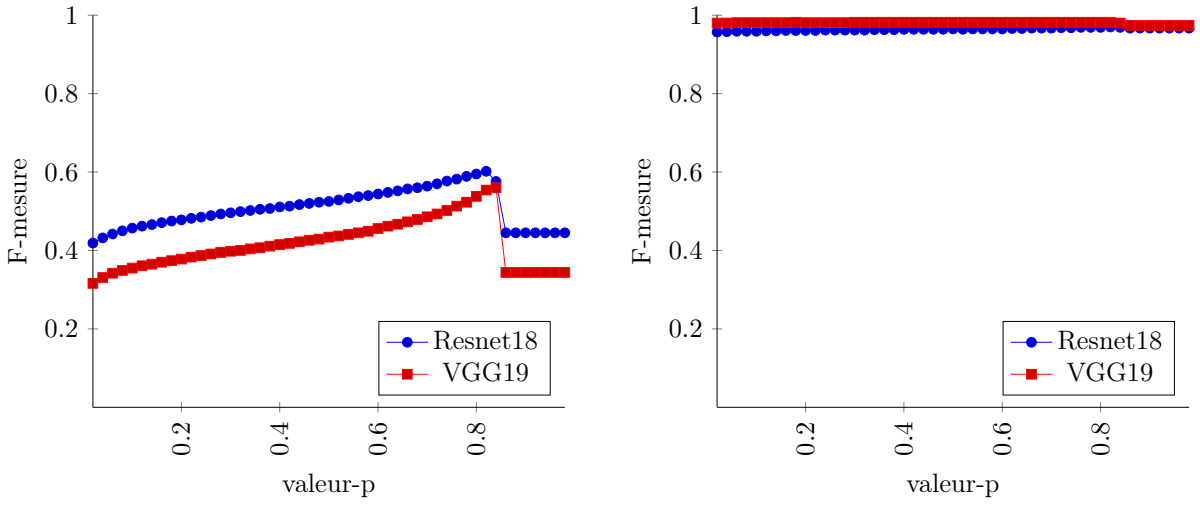
FIGURE 4.5 : Taux de rejet des mauvaises prédictions pour différentes valeurs-p

Il a été décidé de considérer la fonction de rejet comme étant elle-même un modèle de prédiction. Il a ainsi été possible de dresser une matrice de confusion comme s'il s'agissait d'un classifieur binaire tentant de déterminer si les prédictions du précédent modèle sont exactes. À partir de cette matrice de confusion, il a été possible de déterminer les F-mesures de cette fonction pour les différents trios d'ensembles de données, modèles et valeurs p. Ces valeurs sont présentées dans les graphes de la figure 4.6. Afin d'alléger les graphes, seuls les modèles *Resnet18* et *VGG19* sont présentés puisqu'ils bornent les résultats des autres modèles.

En l'absence de travaux précédents auxquels nous comparer (voir plus bas), on peut évaluer grossièrement les performances de notre fonction en utilisant l'échelle suivante :

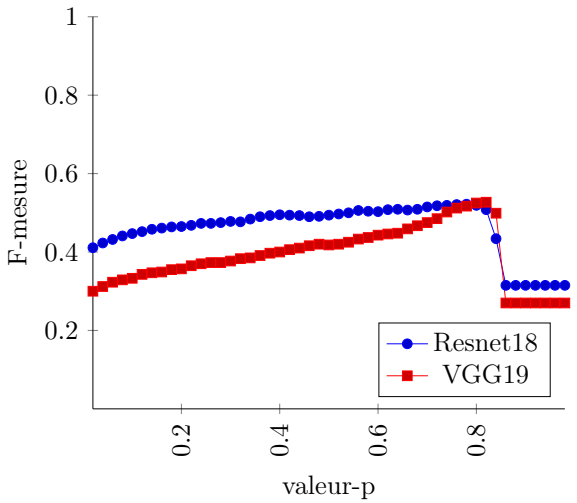
F-mesure	Interprétation
[0,9; 1]	Très bon
[0,8; 0,9[	Bon
[0,5; 0,8[	Acceptable
[0; 0,5[	Mauvais

En suivant cette échelle, on voit que cette fonction de rejet est acceptable à partir d'une certaine valeur p pour tous les modèles. Il est aussi possible, en comparant avec les graphes de la Figure 4.3, de voir que la fonction obtient de meilleurs scores lorsque le modèle sur lequel on l'applique est moins performant. Il semble donc pertinent de pousser plus loin nos travaux afin de l'améliorer et de mieux voir comment elle se compare à d'autres travaux sur le sujet.

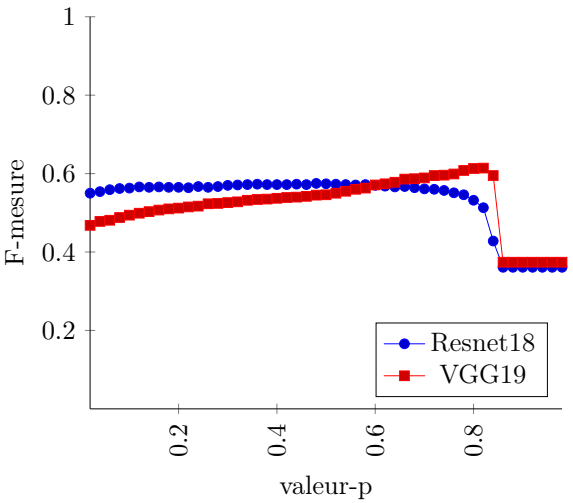


(a) CIFAR10

(b) CIfar100



(c) Feuilles : Érables



(d) Feuilles : Genres

FIGURE 4.6 : F-mesures de la détection d'erreurs pour différentes valeurs p

Une difficulté majeure se présente toutefois au moment de comparer les résultats obtenus avec ceux de travaux similaires. Cette expérience a été faite il y a plusieurs années, à un moment où je n'avais trouvé presque aucune information sur la détection d'anomalie ou les options de rejet. Beaucoup des travaux sur le sujets ont été publiés plus tard, mais aussi, le vocabulaire qui m'a finalement permis de trouver les publications pertinentes m'était alors inconnu (la section 1.3 réfère ces travaux). Par conséquent, je n'ai pas pu définir mon expérience de façon à la rendre comparable à d'autres travaux.

Il reste que, à ma connaissance, la méthode présentée n'a toujours pas été étudiée en tant que fonction de rejet pour des classifieurs et que ses performances semblent dignes d'intérêt.

Il sera pertinent de refaire cette expérience de façon à pouvoir comparer convenablement ses performances à d'autres solutions. Pour avoir des résultats comparables, il faudra utiliser les mêmes ensembles de données que les articles de référence ainsi qu'un modèle similaire. Le test d'hypothèse utilisé a l'avantage de pouvoir exercer un contrôle, bien qu'indirect, sur le taux de rejet en faisant varier la valeur p. Il sera donc possible de tracer sa courbe Exactitude-Rejet en plus des métriques obtenues en considérant la fonction de rejet comme un classifieur (voir la section 1.3.3 pour plus d'information sur ces concepts).

4.2 Expériences sur l'identification depuis les airs

Des expériences ont été effectuées afin d'explorer la possibilités de différencier différents types d'arbres à partir d'images aériennes. Le tableau 4.1 résume les métriques calculées pour chacune de ces expériences. L'état de l'art n'ayant pas d'expérience similaires, nous utiliserons à nouveau l'échelle présentée à la section précédente pour évaluer ces résultats. Selon cette échelle, toutes les f-mesures se trouvent dans le haut de la catégorie acceptable ($[0{,}5; 0{,}8]$). Considérant qu'il ne s'agit que d'une expérience préliminaire, ces résultats sont encourageants, démontrant clairement qu'il est possible pour un réseau de neurones de discriminer entre différents types d'arbres à partir d'images aériennes.

Une de nos hypothèses était que les expériences sur les Ginkgoopsida et les Magnoliopsida obtiendraient des scores inférieurs aux expériences sur les Pinopsida. Cette hypothèse vient de la facilité attendue de différencier les feuilles des feuillus et les épines des conifères. Les biologistes avec lesquels nous travaillons sont d'ailleurs plus intéressés à la distinction entre les feuillus et les conifères que

Classe positive	Échelle	Matrice de confusion	Précision	Rappel	F-mesure	Époques (max : 25)
Ginkgoopsida	17	$\begin{bmatrix} 36 & 16 \\ 18 & 38 \end{bmatrix}$	0,6923	0,6667	0,6792	4
Ginkgoopsida	19	$\begin{bmatrix} 49 & 13 \\ 5 & 41 \end{bmatrix}$	0,7903	0,9074	0,8448	16
Magnoliopsida	17	$\begin{bmatrix} 44 & 26 \\ 10 & 28 \end{bmatrix}$	0,6286	0,5185	0,7097	5
Magnoliopsida	19	$\begin{bmatrix} 36 & 16 \\ 18 & 38 \end{bmatrix}$	0,6923	0,6667	0,6792	25
Pinopsida	17	$\begin{bmatrix} 43 & 17 \\ 11 & 37 \end{bmatrix}$	0,7167	0,7963	0,7544	5
Pinopsida	19	$\begin{bmatrix} 44 & 10 \\ 10 & 40 \end{bmatrix}$	0,7586	0,8148	0,7857	20

TABLEAU 4.1 : Performances de *VGG19* pour identifier des arbres à partir d'images de services de cartographie en ligne

celles entre les Ginkgoopsida et les Magnoliopsida. Cette hypothèse est généralement validée, mais le score de l'expérience Ginkgoopsida à l'échelle 19 est intrigant.

Sa valeur de rappel nettement plus élevée que celle de sa précision suggère que les erreurs du modèle viennent surtout de l'inclusion d'images négatives plutôt que de l'omission d'images positives. Ce constat semble également aller dans le sens de notre hypothèse. Ultimement, ce score vient probablement de la faible taille de l'ensemble de données à l'échelle 19.

Le tableau 4.1 inclut aussi l'époque qui a été retenue. Cette donnée nous semble particulièrement intéressante dans ce cas. On voit qu'à l'échelle 17, cette valeur est très faible et indique un sur ajustement très rapide aux données d'entraînement. Ce problème est probablement dû à la faible résolution des images. Considérant le peu de pixels de chaque image (3600 pour cette échelle), le nombre de variations possibles est plutôt faible. Ce constat n'est pas présent à l'échelle 19, mais, encore une fois, ce n'est dû qu'à la faible taille de l'ensemble de données, chaque époque étant donc très courte.

Ces expériences nous convainquent donc qu'il est pertinent de poursuivre l'objectif de classifier des arbres à partir d'images aériennes. En regardant les images des figures 2.9 et 2.10 nous sommes

persuadés qu'un humain n'obtiendrait pas d'aussi bons résultats. Les expériences indiquent aussi que les images de services de cartographie en ligne pourraient suffire pour différencier les feuillus des conifères. Toutefois, elles ne semblent pas d'assez bonne qualité pour discriminer selon des classes plus fines. Dans cette optique, un autre ensemble de données a déjà été préparé afin de poursuivre nos expériences en ce sens (voir l'ensemble *Sylphe* à la section 2.2.3).

Bien d'autres expériences restent à faire, malheureusement, la construction d'ensembles de données a laissé peu de temps pour les utiliser. Les expériences qui ont été menées à bien ont toutefois pu nous montrer que certaines avenues de recherche envisagées ont du potentiel et qu'il vaudra la peine de continuer à les explorer.

CONCLUSION

L'accès à des inventaires de qualité est une nécessité dans la gestion des forêts urbaines pour les villes voulant faire face aux défis environnementaux actuels. Ce mémoire a présenté plusieurs travaux en lien avec l'intelligence artificielle dont l'objectif est d'aider ces villes à maintenir de tels inventaires en baissant leurs coûts et le temps nécessaire à cette tâche.

Dans un premier temps, nous avons créé quatre ensembles de données qui pourront servir à entraîner des modèles d'apprentissage automatique pour identifier des arbres à partir de différents types d'images : leurs feuilles, de plein pied et à partir des airs. La qualité et la quantité de données sont essentielles en apprentissage automatique et nos recherches ont identifié un manque criant de données utilisables pour ce type de travaux. Nous sommes donc d'autant plus satisfaits que nos travaux s'inscrivent dans un projet plus large qui permettra de continuellement augmenter la quantité de données et les rendre disponibles pour continuer à faire avancer l'état de l'art dans ce domaine.

Dans cette optique, nous avons ensuite présenté un cadre pour héberger et gérer ces données, une méthode pouvant suivre l'évolution de ces données et pouvant les rendre accessibles de façon fiable étant nécessaire pour répondre à nos attentes.

Nous avons également présenté une nouvelle fonction de rejet. Celle-ci a l'avantage de pouvoir être intégrée facilement à n'importe quel classifieur sans avoir besoin de l'entraîner à nouveau. Avoir une telle option de rejet permet de grandement améliorer la qualité des inventaires d'arbres puisqu'il est toujours possible de tenter une nouvelle identification lorsque celle obtenue précédemment ne semble pas assez fiable.

Nous avons finalement démontré la capacité de l'apprentissage profond de discriminer certains types d'arbres à partir de photos aériennes. Cette démonstration a même révélée que des types grossiers tels que les conifères et les feuillus peuvent être différenciés à partir d'images aussi accessibles que celles des services de cartographie en ligne.

Nous croyons que l'intégration de nos travaux dans un outil disponible au public est une occasion de

le sensibiliser à la diversité et l'importance des forêts urbaines. L'ajout d'un module éducatif, peut-être sous la forme d'un wiki, contribuerait aux objectifs environnementaux de cet outil. En lien avec nos travaux sur l'identification d'arbres, nous recommandons d'y expliquer comment les identifier en tant qu'humain et comment fonctionne une clef d'identification. Il serait ensuite pertinent d'y inclure une clef d'identification pour notre climat. Plusieurs organismes, tels que la réserve de biosphère du mont Saint-Hilaire (mont Saint-Hilaire, 2023) et le jardin botanique de Montréal (Montréal, 2010), ont déjà fait des clefs d'identification en ligne. Rendre disponible des liens vers ces ressources permettrait à la communauté de les utiliser pour leur propre enrichissement et celui des données de SylvCiT.

Il y a encore beaucoup de place à l'amélioration pour remplir nos objectifs et il y a déjà plusieurs expériences que nous envisageons. Nous voulons tester si une identification itérative selon la granularité permettrait une amélioration des résultats. Un modèle commencerait par prédire l'appartenance à une classe grossière comme la classe taxonomique (Ginkgoopsida, Magnoliopsida et Pinopsida), un modèle spécialisé pour cette classe prédirait ensuite une classe plus fine comme l'ordre. On procéderait ainsi jusqu'à la classe la plus fine nous intéressant, l'espèce ou même la variété. L'hypothèse est qu'une telle approche permettrait à chacun des modèles de spécialiser ses couches à convolutions pour détecter des motifs pertinents aux classes pertinentes. Nous utiliserions ainsi notre connaissance du domaine pour obtenir de multiples modèles plus rapides à entraîner que l'unique modèle, plus grand, qui serait nécessaire pour obtenir les mêmes performances.

La difficulté de trouver des données adaptées nous poussent également à considérer la prise de nos propres images à l'aide de drones. Cela nous donnerait un meilleurs contrôle sur la qualité des images ainsi que sur les paramètres de leurs acquisition : saison, zone géographique, altitude, etc.

Nous aimerions profiter d'une autre possibilité offerte par notre domaine pour essayer une technique inspirée du *bagging*. Serait-il possible d'utiliser plusieurs images du même sujet pour obtenir de meilleures prédictions ?

Finalement, nous aimerions travailler à l'acquisition automatique d'autres informations nécessaires à la gestion des forêts urbaines. Deux informations nous intéressent particulièrement. Nous voulons premièrement être capables d'obtenir le diamètre à hauteur de poitrine, une mesure standard du

diamètre d'un tronc d'arbre. Les caméras modernes contiennent toutes sortes de méta données et nous croyons possible d'en utiliser certaines, comme la distance focale, pour calculer cette valeur à partir de la largeur du tronc en pixels. Le second projet, ambitieux, serait de détecter et identifier automatiquement la présence de maladies ou de ravageurs dans un sujet. Obtenir cette information plus rapidement et systématiquement a le potentiel de permettre de prévenir ou d'enrayer plus rapidement toute épidémie menaçant les forêts urbaines.

ANNEXE A
LISTE D'ESPÈCES

Les espèces suivantes sont les espèces d'arbres qui sont présentes sur les territoires d'intérêts pour ce document :

- Abies balsamea
- Abies concolor
- Abies koreana
- Abies sibirica
- Acer campestre
- Acer freemanii
- Acer ginnala
- Acer grandidentatum
- Acer griseum
- Acer miyabei
- Acer negundo
- Acer nigrum
- Acer palmatum
- Acer pensylvanicum
- Acer platanoides
- Acer pseudoplatanus
- Acer rubrum
- Acer saccharinum
- Acer saccharum
- Acer spicatum
- Acer tataricum
- Acer triflorum
- Acer truncatum
- Aesculus arguta
- Aesculus carnea
- Aesculus flava
- Aesculus glabra
- Aesculus hippocastanum
- Aesculus indica
- Aesculus octandra
- Aesculus parviflora
- Aesculus x bushii
- Ailanthus altissima
- Alnus crispa
- Alnus glutinosa
- Alnus incana
- Alnus rugosa
- Amelanchier alnifolia
- Amelanchier arborea
- Amelanchier canadensis
- Amelanchier grandiflora
- Amelanchier laevis
- Amorpha fruticosa
- Aralia elata
- Aralia spinosa
- Betula alba
- Betula alleghaniensis
- Betula glandulosa
- Betula lenta
- Betula nigra
- Betula papyrifera
- Betula pendula
- Betula platyphylla
- Betula populifolia
- Betula x 'Crimson frost'
- Buddleia alternifolia
- Callicarpa dichotoma
- Caragana arborescens
- Carpinus betulus
- Carpinus caroliniana
- Carya cordiformis
- Carya illinoinensis
- Carya ovata
- Catalpa bignonioides
- Catalpa ovata
- Catalpa speciosa
- Celtis occidentalis
- Cephalanthus occidentalis
- Cercidiphyllum japonicum
- Cercis canadensis
- Chamaecyparis nootkatensis
- Chionanthus virginicus
- Cladrastis kentukea
- Cladrastis lutea
- Cladrastis tinctoria
- Cornus alternifolia
- Cornus stolonifera
- Corylus americana
- Corylus avellana
- Corylus colurna
- Crataegus canadensis
- Crataegus crus-galli
- Crataegus laevigata
- Crataegus mordenensis
- Crataegus phaenopyrum
- Crataegus rotundifolia
- Crataegus viridis
- Elaeagnus angustifolia
- Eucommia ulmoides
- Euonymus atropurpureus
- Euonymus bungeanus
- Euonymus europaeus
- Fagus grandifolia
- Fagus sylvatica
- Fraxinus americana
- Fraxinus angustifolia
- Fraxinus bungeana
- Fraxinus chinensis
- Fraxinus excelsior
- Fraxinus griffithii
- Fraxinus insularis
- Fraxinus longicuspis
- Fraxinus mandshurica
- Fraxinus nigra
- Fraxinus ornus
- Fraxinus pallisae
- Fraxinus pennsylvanica
- Fraxinus profunda
- Fraxinus quadrangulata
- Fraxinus sieboldiana

— Fraxinus sogdiana
— Fraxinus stylosa
— Fraxinus velutina
— Fraxinus x 'Northern Gem'
— Ginkgo biloba
— Gleditsia triacanthos
— Gymnocladus dioicus
— Halesia carolina
— Hamamelis mollis
— Hamamelis virginiana
— Heptacodium miconioides
— Hippophae rhamnoides
— Juglans ailantifolia
— Juglans cinerea
— Juglans mandshurica
— Juglans nigra
— Juglans regia
— Juniperus communis
— Juniperus virginiana
— Kolkwitzia amabilis
— Larix decidua
— Larix kaempferi
— Larix laricina
— Larix leptolepis
— Ligustrum vulgare
— Lindera benzoin
— Liquidambar styraciflua
— Liriodendron tulipifera
— Maackia amurensis
— Magnolia acuminata
— Magnolia fraseri
— Magnolia kobus
— Magnolia loebneri
— Magnolia rustica
— Magnolia spp
— Magnolia stellata
— Magnolia x soulangeana
— Malus antonovka
— Malus baccata
— Malus coronaria
— Malus domestica
— Malus hupehensis
— Malus ioensis
— Malus prunifolia
— Malus sargentii
— Malus spp
— Metasequoia glyptostroboides
— Morus alba
— Morus rubra
— Nyssa sylvatica
— Ostrya virginiana
— Phellodendron amurense
— Phellodendron lavallei
— Picea abies
— Picea asperata
— Picea engelmannii
— Picea glauca
— Picea mariana
— Picea omorika
— Picea pungens
— Picea rubens
— Pinus aristata
— Pinus banksiana
— Pinus cembra
— Pinus contorta
— Pinus densiflora
— Pinus flexilis
— Pinus koraiensis
— Pinus mugo
— Pinus nigra
— Pinus parviflora
— Pinus peuce
— Pinus ponderosa
— Pinus resinosa
— Pinus rigida
— Pinus strobus
— Pinus sylvestris
— Platanus acerifolia
— Platanus occidentalis
— Populus alba
— Populus balsamifera
— Populus canadensis
— Populus canescens
— Populus deltoides
— Populus grandidentata
— Populus nigra
— Populus simonii
— Populus sp.(hybrides)
— Populus tremula
— Populus tremuloides
— Prunus americana
— Prunus armeniaca
— Prunus cerasifera
— Prunus cerasus
— Prunus cistena
— Prunus domestica
— Prunus maackii
— Prunus nigra
— Prunus padus
— Prunus pensylvanica
— Prunus persica
— Prunus sargentii
— Prunus serotina
— Prunus serrulata
— Prunus subhirtella
— Prunus triloba
— Prunus virginiana
— Prunus yedoensis
— Pseudotsuga menziesii
— Ptelea trifoliata
— Pterocarya stenoptera
— Pyrus calleryana
— Pyrus communis
— Pyrus fauriei
— Pyrus michauxii
— Pyrus pyrifolia
— Pyrus ussuriensis
— Quercus acutissima
— Quercus alba
— Quercus bicolor
— Quercus coccinea
— Quercus ellipsoidalis
— Quercus imbricaria
— Quercus macrocarpa
— Quercus muehlenbergii
— Quercus palustris
— Quercus petraea
— Quercus robur
— Quercus rubra
— Rhamnus cathartica
— Rhamnus frangula
— Rhus glabra
— Rhus typhina
— Robinia pseudoacacia
— Robinia slavinii
— Salix alba
— Salix babylonica
— Salix bebbiana
— Salix caprea
— Salix eleagnos
— Salix fragilis

- Salix matsudana
- Salix miyabeana
- Salix nigra
- Salix pentandra
- Salix sepulcralis
- Sambucus canadensis
- Sambucus nigra
- Sambucus pubens
- Sciadopitys verticillata
- Shepherdia argentea
- Sophora japonica
- Sorbus alnifolia
- Sorbus americana
- Sorbus aria
- Sorbus aucuparia
- Sorbus decora
- Sorbus folgneri
- Sorbus intermedia
- Sorbus latifolia
- Sorbus thuringiaca
- Sorbus xanthocarpa
- Staphylea trifolia
- Syringa josikaea
- Syringa pekinensis
- Syringa prestoniae
- Syringa reticulata
- Syringa villosa
- Syringa vulgaris
- Tamarix parviflora
- Taxus cuspidata
- Thuja occidentalis
- Thuja standishii
- Tilia americana
- Tilia cordata
- Tilia europaea
- Tilia flavescens
- Tilia mongolica
- Tilia platyphyllos
- Tilia tomentosa
- Tilia vulgaris
- Tsuga canadensis
- Ulmus americana
- Ulmus carpinifolia
- Ulmus davidiana
- Ulmus glabra
- Ulmus parvifolia
- Ulmus propinqua
- Ulmus pumila
- Ulmus rubra
- Ulmus thomasii
- Ulmus wilsoniana
- Ulmus x
- Ulmus x 'Groeneveld'
- Viburnum lantana
- Viburnum lentago
- Viburnum opulus
- Viburnum prunifolium
- Viburnum trilobum
- Vitex negundo
- Zanthoxylum americanum
- Zelkova serrata

GLOSSAIRE

geographic coordinate system Système de coordonnées permettant de situer un point sur terre. 24, 62

projected coordinate system Système de coordonnées permettant de représenter des données géographiques sur une projection de la surface de la terre (une carte par exemple). De nombreux systèmes existent selon la région à représenter et le type de projection utilisé. 24, 62

ARC courbe Exactitude-Rejet. 14, 53

DHP diamètre à hauteur de poitrine. 57

GCS *geographic coordinate system*. 24

GPS géo-positionnement par satellite. 24, 26

HDFS *Hadoop Distributed File System*. 22, 32–34

LIDAR *laser imaging, detection, and ranging*. 17

MNE modèle numérique d'élévation. 14

NDVI *normalized difference vegetation index*. 15, 16

niveau de la rue Caractérise une prise de vue à partir du sol avec un angle à peu près parallèle au sol. 2

observation Une observation correspond à une entrée d'un sujet. Elle peut être associée à plusieurs images du même sujet et un sujet peut avoir plusieurs observations si elles ont été entrées séparément (divers moments, utilisateurs, ...). 20–22

paramètres Appellation faisant référence aux poids des neurones artificiels ainsi qu'à leurs biais. 4

PCS *projected coordinate system*. 24, 25

SIG système d'information géographique. 24

SylvCiT Application en ligne de gestion de la forêt urbaine. Il s'agit du projet dans lequel s'inscrit les travaux présentés dans ce mémoire.. 1–3, 57

UQAM Université du Québec à Montréal. 1

BIBLIOGRAPHIE

Abbas, M. R., Nadeem, M. S. A., Shaheen, A., Alshdadi, A. A., Alharbey, R., Shim, S.-O. et Aziz, W. (2019). Accuracy rejection normalized-cost curves (arnccs) : A novel 3-dimensional framework for robust classification. *IEEE Access*, *7*, 160125–160143. http://dx.doi.org/10.1109/ACCESS.2019.2950244

Affouard, A., Joly, A., Lombardo, J.-C., Champ, J., Goeau, H. et Bonnet, P. (2020). Pl@ntNet observations. version 1.2. http://dx.doi.org/10.15468/gtebaa

Alalouf, S., Labelle, D. et Ménard, J. (2002). *Introduction à la statistique appliquée*. Longueuil, Québec : Loze-Dion.

Albawi, S., Mohammed, T. A. et Al-Zawi, S. (2017). Understanding of a convolutional neural network. Dans *2017 International Conference on Engineering and Technology (ICET)*, 1–6. http://dx.doi.org/10.1109/ICEngTechnol.2017.8308186

Alta Photo (2002). Photos aériennes de la Ville de Montréal. http://depot.ville.montreal.qc.ca/geomatique/photo_aerienne/2002/photo_2002.zip.

Apache, S. F. (2006–2023a). Apache Hadoop. https://hadoop.apache.org/. Licence : Apache License 2.0.

Apache, S. F. (2006–2023b). Apache Hadoop 3.3.6 - hdfs. https://hadoop.apache.org/docs/r3.3.6/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html. Licence : Apache License 2.0.

Apache, S. F. (2006–2023c). Apache Hadoop 3.3.6 - mapreduce. https://hadoop.apache.org/docs/r3.3.6/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html. Licence : Apache License 2.0.

Apache, S. F. (2006–2023d). Solr. https ://solr.apache.org/downloads.html. Licence : Apache License 2.0.

Apache, S. F. (2011 – 2023e). Welcome to Apache Lucene. https://lucene.apache.org/index.html.

Apache, S. F. (2018). Apache Spark. https://spark.apache.org/. Licence : Apache License 2.0.

Aphex34 (2015). max_pooling with 2×2 filter and stride = 2. https://upload.wikimedia.org/wikipedia/commons/e/e9/Max_pooling.png. Licence : CC BY-SA 4.0, Modifiée par Maxime Faubert Laurin (2023).

Aphex34 et ClaudeCoulombe (2021). French version of a classic convolutional neural network architecture with some additional visual enhancement. https://upload.wikimedia.org/wikipedia/commons/5/5b/Typical_cnn_fr.png?uselang=fr. Licence : CC BY-SA 4.0.

Baral, C., Fuentes, O. et Kreinovich, V. (2018). *Why Deep Neural Networks : A Possible Theoretical Explanation*, Dans *Constraint Programming and Decision Making : Theory and Applications*, (p. 1–5). Springer International Publishing : Cham

Bendale, A. et Boult, T. (2015). Towards Open World Recognition. Dans *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1893–1902. http://dx.doi.org/10.1109/CVPR.2015.7298799

B.L., B., N., S. et Hebbar, R. (2018). Tree Crown Detection and Extraction from High Resolution Satellite Images in an Urban Area. Dans *2018 International Conference on Design Innovations for 3Cs Compute Communicate Control (ICDI3C)*, 8–11. http://dx.doi.org/10.1109/ICDI3C.2018.00011

Chang, K.-T. (2019). *Introduction to Geographic Information Systems* (9 éd.). New-York : McGraw-Hill Education.

CIA et Niteowlneils (2006). Great circles of latitude. https://upload.wikimedia.org/wikipedia/commons/a/ab/WorldMapLongLat-eq-circles-tropics-non.png?uselang=fr. Licence : Domaine Public.

Condessa, F., Bioucas-Dias, J. et Kovačević, J. (2017). Performance measures for classification systems with rejection. *Pattern Recognition*, *63*, 437–450. http://dx.doi.org/10.1016/j.patcog.2016.10.011

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, *2*(4), 303–314. http://dx.doi.org/10.1007/BF02551274

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K. et Fei-Fei, L. (2009). Imagenet : A large-scale hierarchical image database. Dans *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–255. http://dx.doi.org/10.1109/CVPR.2009.5206848

Djexplo (2019). Latitude and longitude of the earth fr. https://upload.wikimedia.org/wikipedia/commons/8/86/Latitude_and_Longitude_of_the_Earth_fr.svg?uselang=fr. Licence : Domaine Public.

Geng, C., Huang, S.-j. et Chen, S. (2021). Recent Advances in Open Set Recognition : A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *43*(10), 3614–3631. http://dx.doi.org/10.1109/TPAMI.2020.2981604

Gkrusze (2018). Imagenet error rate history (just systems). https://upload.wikimedia.org/wikipedia/commons/4/4f/ImageNet_error_rate_history_%28just_systems%29.svg. Licence : CC BY-SA 4.0, Traduite par Maxime Faubert Laurin (2023).

Glorot, X. et Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. Dans Y. W. Teh et M. Titterington (dir.). *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 de *Proceedings of Machine Learning Research*, 249–256., Chia Laguna Resort, Sardinia, Italy. PMLR.

Goodfellow, I., Bengio, Y. et Courville, A. (2016). *Deep Learning*. MIT Press.

Google (2023). Google maps. https://www.google.ca/maps.

Hanczar, B. (2019). Performance visualization spaces for classification with rejection option. *Pattern Recognition*, *96*, 106984. http://dx.doi.org/10.1016/j.patcog.2019.106984

Hang, S. T. et Aono, M. (2016). Open world plant image identification based on convolutional neural network. Dans *2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 1–4. http://dx.doi.org/10.1109/APSIPA.2016.7820676

He, K., Zhang, X., Ren, S. et Sun, J. (2015a). Deep residual learning for image recognition. http://dx.doi.org/10.48550/ARXIV.1512.03385

He, K., Zhang, X., Ren, S. et Sun, J. (2015b). Delving deep into rectifiers : Surpassing human-level performance on imagenet classification. Dans *2015 IEEE International Conference on Computer Vision (ICCV)*, 1026–1034. http://dx.doi.org/10.1109/ICCV.2015.123

Hendrickx, K., Perini, L., Van der Plas, D., Meert, W. et Davis, J. (2021). Machine Learning with a Reject Option : A survey. http://dx.doi.org/arXiv:2107.11277

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I. et Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. http://dx.doi.org/10.48550/arXiv.1207.0580

Huang, X., Shi, C. et Liew, S. C. (2018). Tree Crown Detection and Delineation Using Optical Satellite Imagery. Dans *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 2944–2947. http://dx.doi.org/10.1109/IGARSS.2018.8517962

iNaturalist (2022). A community for naturalists : iNaturalist. https://www.inaturalist.org.

iNaturalist Contributors (2022). iNaturalist Research-grade Observations. http://dx.doi.org/10.15468/ab3s5x

Jim, C. Y. et Chen, W. Y. (2008). Assessing the ecosystem service of air pollutant removal by urban trees in Guangzhou (China). *Journal of Environmental Management*, *88*(4), 665–676. http://dx.doi.org/10.1016/j.jenvman.2007.03.035

Karmakar, B. et Pal, N. R. (2018). How to make a neural network say "Don't know". *Information Sciences*, *430-431*, 444–466. http://dx.doi.org/10.1016/j.ins.2017.11.061

Klokan Technologies GmbH (2022a). Coordinate reference systems for "Quebec". https://epsg.io/?q=Quebec.

Klokan Technologies GmbH (2022b). Wgs 84 / pseudo-mercator - spherical mercator, google maps, openstreetmap, bing, arcgis, esri - epsg :3857. https://epsg.io/3857.

Krizhevsky, A., Hinton, G. et Nair, V. (2009). *Learning multiple layers of features from tiny images*. Rapport technique, University of Toronto.

Krizhevsky, A., Sutskever, I. et Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Dans *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Kueda (2020). Api reference · inaturalist. https://www.inaturalist.org/pages/api+reference.

LeCun, Y. *et al.* (1989). Generalization and network design strategies. *Connectionism in perspective*, *19*(143-155), 18.

Livesley, S. J., McPherson, E. G. et Calfapietra, C. (2016). The Urban Forest and Ecosystem Services : Impacts on Urban Water, Heat, and Pollution Cycles at the Tree, Street, and City Scale. *Journal of Environmental Quality*, *45*(1), 119–124. http://dx.doi.org/10.2134/jeq2015.11.0567

LunarLullaby (2015). Resblock. https://upload.wikimedia.org/wikipedia/commons/b/ba/ResBlock.png. Licence : CC BY-SA 4.0, Traduite par Maxime Faubert Laurin (2023).

mapbox (2023). Maps, geocoding, and navigation apis & sdks | mapbox. https://www.mapbox.com/.

Martens, J. (2010). Deep learning via hessian-free optimization. Dans *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, p. 735–742., Madison, WI, USA. Omnipress.

Mercator, P. (2010). Merctransph. https://upload.wikimedia.org/wikipedia/commons/1/15/MercTranSph.png. Licence : Domaine Public.

mont Saint-Hilaire, R. d. b. d. (2023). Outil d'identification des arbres. https ://centrenature.qc.ca/arbres/.

Montréal, J. b. d. (2010). Clé d'identification des arbres. https ://www.aucoeurdelarbre.ca/fr/hors-sentier/identifier-arbres-cle-html.php.

Montréal, V. d. (2016). Photographies aériennes - Photothèque - Site web des données ouvertes de la Ville de Montréal. https://donnees.montreal.ca/dataset/phototheque.

Mouton, C., Myburgh, J. C. et Davel, M. H. (2020). Stride and Translation Invariance in CNNs. Dans A. Gerber (dir.). *Artificial Intelligence Research*, Communications in Computer and Information Science, 267–281., Cham. Springer International Publishing. http://dx.doi.org/10.1007/978-3-030-66151-9_17

Nadeem, M. S. A., Zucker, J.-D. et Hanczar, B. (2009). Accuracy-Rejection Curves (ARCs) for Comparing Classification Methods with a Reject Option. Dans *Proceedings of the third International Workshop on Machine Learning in Systems Biology*, 65–81. PMLR. ISSN : 1938-7228. Récupéré le 2023-08-01 de https://proceedings.mlr.press/v8/nadeem10a.html

OpenStreetMap (2023). Openstreetmap. https://www.openstreetmap.org/.

OpenStreetMap contributors (2020). Numbering scheme for tiled web map. https://upload.wikimedia.org/wikipedia/commons/f/f2/Tiled_web_map_numbering.png. Licence : CC BY-SA 2.0.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J. et Chintala, S. (2019). PyTorch : An Imperative Style, High-Performance Deep Learning Library. Dans *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Peng Gong, Ruiliang Pu, Biging, G. S. et Larrieu, M. R. (2003). Estimation of forest leaf area index using vegetation indices derived from Hyperion hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, *41*(6), 1355–1362. http://dx.doi.org/10.1109/TGRS.2003.812910

Peterson, A. (2016). Types de classification du climat Köppen du Québec. https://commons.wikimedia.org/wiki/File:Québec_fr_Köppen.svg#file. Licence : CC BY-SA 4.0.

Pillai, I., Fumera, G. et Roli, F. (2011). A Classification Approach with a Reject Option for Multi-label Problems. Dans G. Maino et G. L. Foresti (dir.). *Image Analysis and Processing – ICIAP 2011*, Lecture Notes in Computer Science, 98–107., Berlin, Heidelberg. Springer. http://dx.doi.org/10.1007/978-3-642-24085-0_11

Pl@ntNet team (2022). Pl@ntnet. https://plantnet.org/.

Russell, S. et Norvig, P. (2009). *Artificial intelligence* (3 éd.). Upper Saddle River, NJ : Pearson.

Salmond, J. A., Tadaki, M., Vardoulakis, S., Arbuthnott, K., Coutts, A., Demuzere, M., Dirks, K. N., Heaviside, C., Lim, S., Macintyre, H., McInnes, R. N. et Wheeler, B. W. (2016). Health and climate related ecosystem services provided by street trees in the urban environment. *Environmental Health*, *15*(1), S36. http://dx.doi.org/10.1186/s12940-016-0103-6

Scherer, D., Müller, A. et Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. Dans *International conference on artificial neural networks*, 92–101. Springer.

Simonyan, K. et Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. http://dx.doi.org/10.48550/ARXIV.1409.1556

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. et Salakhutdinov, R. (2014). Dropout : a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929–1958. http://dx.doi.org/10.5555/2627435.2670313

Strebe (2011). Mercator projection square. https://upload.wikimedia.org/wikipedia/commons/7/73/Mercator_projection_Square.JPG. Licence : CC BY-SA 3.0.

SylvCiT (2023). SylvCiT. https://sylvcit.ca/.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. et Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. Dans *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2818–2826. ISSN : 1063-6919, http://dx.doi.org/10.1109/CVPR.2016.308

Taylor, B. (1715). *Methodus incrementorum directa & inversa. Auctore Brook Taylor ...* Londini : Typis Pearsonianis prostant apud Gul. Innys ad Insignia Principis in Coemeterio Paulino, 1715.

Xiao, C., Qin, R., Huang, X. et Li, J. (2018). Individual Tree Detection from Multi-View Satellite Images. Dans *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*, 3967–3970. http://dx.doi.org/10.1109/IGARSS.2018.8518040

Zeiss (1994). RMK TOP aerial survey camera sytem. https://aerial-survey-base.com/wp-content/uploads/2017/01/RMK-TOP-Brochure_1994_LR.pdf.