UCLouvain

École polytechnique de Louvain

Automatisation de la classification des spécimens d'insectes provenant de boîtes entomologiques de collections muséales

Auteur: **Nicolas Delcommune**
Promoteur: **Sébastien Jodogne**
Lecteurs: **Edouard Chatzopoulos, Grégoire Noël**
Année académique 2024–2025
Master [60] en sciences informatiques

Remerciements

Je tiens tout d'abord à exprimer ma plus profonde gratitude au Dr. Grégoire Noël, mon superviseur au laboratoire entomologique de Gembloux, pour son accompagnement tout au long de ce projet. Son expertise, sa disponibilité et son engagement ont été essentiels à la réalisation de ce mémoire, et je lui en suis profondément reconnaissant.

Je remercie également le Pr. Sébastien Jodogne pour la confiance qu'il m'a accordée, malgré la situation particulière dans laquelle je me trouvais.

Je remercie aussi le Pr. Frédéric Francis pour son accueil au sein du laboratoire de Gembloux.

Enfin, je tiens à exprimer toute ma reconnaissance à ma famille et à mes amis, dont le soutien moral a été une source constante de motivation. Leurs encouragements m'ont permis de rester concentré et persévérant.

À toutes ces personnes qui ont contribué, de près ou de loin, à l'accomplissement de ce mémoire, je vous adresse mes plus sincères remerciements.

Résumé

L'identification et la classification des insectes, essentielles pour l'entomologie et la préservation de la biodiversité, demeurent des tâches laborieuses lorsqu'elles reposent sur des méthodes manuelles. Ce projet vise à automatiser la détection et la classification des spécimens d'insectes à partir d'images numériques de boîtes entomologiques issues de collections muséales. En intégrant des techniques avancées d'apprentissage profond et de vision par ordinateur, deux approches complémentaires ont été développées : la détection des insectes via les modèles YOLO et Faster R-CNN, suivie d'une classification précise grâce aux réseaux de neurones ResNet et EfficientNet.

Les résultats démontrent que les modèles développés offrent une précision élevée dans la reconnaissance des espèces, tout en optimisant considérablement le temps d'analyse par rapport aux méthodes traditionnelles. Toutefois, des défis subsistent, notamment pour les espèces morphologiquement proches et les spécimens de petite taille. Ce travail met en lumière les bénéfices d'une telle automatisation, en facilitant le traitement à grande échelle des collections entomologiques et en ouvrant la voie à de nouvelles applications pour le suivi de la biodiversité et la recherche écologique.

Table des matières

1 Introduction **1**
1.1 Contexte 1

2 Étude des méthodes existantes **3**
2.1 Méthodes traditionnelles de classification d'insectes 3
2.2 Apprentissage profond et vision par ordinateur 4
2.2.1 Concepts de Base 4
2.2.2 Les Réseaux de Neurones Convolutifs 5
2.3 Modèles de détection d'objets 8
2.3.1 YOLO 8
2.3.2 Faster R-CNN 11
2.4 Modèles de classification d'images 12
2.4.1 ResNet 12
2.4.2 EfficientNet 13

3 Objectifs et matériel **15**
3.1 Objectifs 15
3.2 Matériel utilisé 15
3.2.1 Données d'entrées 16
3.2.2 Langages de programmation et bibliothèques 16
3.2.3 LUCIA 17

4 Design expérimental **18**
4.1 Description du programme existant 18
4.1.1 Annotation des boîtes 18
4.1.2 Implémentation de YOLO 20
4.1.3 Implémentation de Faster R-CNN 22
4.2 Améliorations de la détection des insectes 23
4.2.1 Annotations négatives 23
4.2.2 Pseudo-labelling 24
4.2.3 Autres modifications 25

4.3 Classification des insectes 25
4.3.1 Préparation du jeu de données 25
4.3.2 Implémentation des modèles de classification 26

5 Résultats **29**
5.1 Résultats de la détection d'insectes 29
5.1.1 Coleoptera 29
5.1.2 Lepidoptera 33
5.2 Résultats de la classification d'insectes 36
5.2.1 Comparaison des modèles de classification 36
5.2.2 Analyse détaillée des résultats 37
5.2.3 Facteurs influant sur la précision des modèles 39

6 Discussion **41**
6.1 Perspectives d'amélioration 41
6.1.1 Détection 41
6.1.2 Classification 41
6.1.3 Gestion des ressources 42
6.2 Contribution et bénéfices du projet 42
6.3 Limites et difficultés rencontrées 43

7 Conclusion **44**

Bibliographie

Annexes
A Code . . . . .
B Espèces de Coleoptera utilisées lors de la phase de classification . .
C Espèces de Lepidoptera utilisées lors de la phase de classification . .

Table des figures

1.1 Photographie d'une boîte entomologique provenant de l'AfricaMuseum. 2

2.1 Ilustration de la couche de convulation dans un CNN, montrant l'application d'un motif sur une image d'entrée pour générer plusieurs cartes de caractéristiques (feature-map) à l'aide de l'opération Conv2D. [1] 6
2.2 Illustration du processus de max pooling avec un filtre 2x2 et un pas de 2. [2] 7
2.3 Illustration du quadrillage d'image par YOLO. [3] 9
2.4 Illustration de la détection et de la localisation des objets par YOLO. [3] 9
2.5 Carte de probabilité et exemple de détection par YOLO. [3] 10
2.6 Exemple d'image après suppression des détections superflues par YOLO. [3] 10
2.7 Illustration du fonctionnement de Faster R-CNN. [4] 11

3.1 Exemple de lancement d'un script sur LUCIA. 17

4.1 Exemple de photographie de boîte entomologique. 19
4.2 Exemple de boîte partiellement et complètement annotée. 19
4.3 Exemple de fichier d'annotation de boîte entomologique. 19
4.4 Commande de lancement de l'entraînement de YOLO. 20
4.5 Exemple de résultat pour une prédiction par YOLO. 21
4.6 Exemples d'erreurs faites par le modèle de détection 23
4.7 Exemple d'annotations négatives sur une boîte entomologique. . . . 24
4.8 Exemple de fichier résultat d'une détection. 25
4.9 Format du dataset d'entraînement pour la classification. 26
4.10 Image utilisé pour le dataset d'entraînement de l'insecte Goliathus_goliathus. 26
4.11 Exemple de paramétrage du ImageDataGenerator. 27
4.12 Fichier résultat de la classification. 28

5.1 Comparaison entre le comptage manuel et les prédictions des différents modèles pour l'ordre des Coleoptera. 30
5.2 Comparaison entre le comptage manuel et les prédictions des différents modèles (avec améliorations) pour l'ordre des Coleoptera . . . 31
5.3 Evolution des tendances d'erreur selon les modèles de détection pour l'ordre des Coleoptera. 32
5.4 Comparaison des erreurs relatives et absolues des différents modèles de détection pour l'ordre des Coleoptera 32
5.5 Comparaison entre le comptage manuel et les prédictions des différents modèles pour l'ordre des Lepidoptera. 34
5.6 Comparaison entre le comptage manuel et les prédictions des différents modèles (avec améliorations) pour l'ordre des Lepidoptera . . 34
5.7 Evolution des tendances d'erreur selon les modèles de détection pour l'ordre des Lepidoptera. 35
5.8 Comparaison des erreurs relatives et absolues des différents modèles de détection pour l'ordre des Lepidoptera 35
5.9 Exemple de boîte entomologique montrant des insectes qui se superposent. 36
5.10 Tableau récapitulatif des résultats des deux modèles de classification. 37
5.11 Matrice de confusion d'une classification par nom d'espèce pour l'ordre des Coleoptera par le modèle ResNet. 37
5.12 Illustration d'une erreur courante du modèle de détection pour l'ordre des Coleoptera 38
5.13 Matrice de confusion d'une classification par nom d'espèce pour l'ordre des Lepidoptera par le modèle ResNet. 38
5.14 Illustration d'une erreur courante du modèle de détection pour l'ordre des Lepidoptera 39
5.15 Graphique montrant l'influence de la résolution sur la précision. . . 40
5.16 Graphique montrant l'influence de la taille des insectes sur la précision des modèles. 40

Chapitre 1

Introduction

1.1 Contexte

Les insectes sont les organismes les plus diversifiés sur Terre, représentant plus de 4 millions d'espèces dans le monde. Ils jouent un rôle essentiel dans le maintien des écosystèmes naturels en assurant des fonctions écologiques et des services écosystémiques cruciaux. [5] [6] [7] Ces derniers incluent leur importance pour la production agricole, leur contribution au cycle des nutriments et leur influence sur la physiologie, l'activité, et la dynamique des populations végétales. Environ 60% des oiseaux dépendent des insectes pour se nourrir et environ un tiers des cultures destiné à notre alimentation est pollinisé par ceux-ci [8]. Cependant, ces dernières décennies, les insectes ont été gravement affectés par la crise mondiale de la biodiversité, exacerbée par la dégradation des paysages due à l'intensification agricole, l'urbanisation, et d'autres activités humaines.

L'identification spécifique des insectes repose sur une classification bien établie connue sous le nom de systématique. Ce processus utilise traditionnellement des traits morphologiques, tels que la couleur des antennes, la taille du proboscis, ou encore la nervation alaire, pour distinguer les différentes espèces à l'aide de clés dichotomiques de détermination [9]. Cependant, la caractérisation morphologique des espèces est sujette à certaines limitations telles que des espèces indissociables, des complexes spécifiques, etc. Cette classification manuelle des insectes reste également un processus complexe et répétitif, qui mobilise des experts spécialisés pour identifier chaque spécimen. Cette approche, bien qu'efficace, limite considérablement la capacité des chercheurs à analyser rapidement un grand nombre de spécimens. Face à la croissance des collections entomologiques et au besoin de surveiller la biodiversité, cette méthode peut se révéler peu adaptée aux besoins actuels de rapidité et d'efficacité dans le suivi des espèces d'insectes. Les boîtes entomologiques

jouent un rôle très important dans l'étude des insectes. Ces boîtes, conçues pour protéger les échantillons des dégradations, sont des outils incontournables pour les entomologistes. Elles permettent une conservation de longue durée et offrent un accès visuel direct pour les analyses ou comparaisons.

FIGURE 1.1 – Photographie d'une boîte entomologique provenant de l'AfricaMuseum.

Toutefois, ces boîtes physiques présentent des limites, elles ont toujours un risque de dégradation avec le temps et il est très difficile pour les chercheurs de se partager des échantillons à travers le monde.

Dans ce contexte, la numérisation prend une importance croissante. L'évolution des technologies numériques permet aujourd'hui de constituer d'importantes bases de données d'images, facilitant ainsi la distinction de ces espèces. Cette transition vers des collections numériques offre plusieurs avantages. Elle permet une conservation qui restera intacte. Ensuite, elle permet de partager les informations sur les spécimens à une échelle mondiale et ce, rapidement et facilement. Cette transition permet de rendre les données accessibles à un large public et donc d'accélérer les recherches.

Face aux limites des approches traditionnelles et grâce aux avancées en numérisation, l'intelligence artificielle offre des solutions prometteuses. En exploitant des modèles d'apprentissage automatique capables d'analyser des images d'insectes, l'IA peut automatiser la classification en s'appuyant sur des bases taxonomiques telles que l'ordre, la famille, le genre ou l'espèce. Ces algorithmes modernes permettent d'accélérer le processus de d'identification des insectes mais aussi d'en améliorer la précision.

Ainsi, la numérisation et l'automatisation par intelligence artificielle peuvent aider grandement la recherche en entomologie, en rendant possible l'analyse de collections à très grandes échelle, dans un contexte où la surveillance de la biodiversité devient de plus en plus importante.

Chapitre 2

Étude des méthodes existantes

2.1 Méthodes traditionnelles de classification d'insectes

La classification des insectes repose historiquement sur des méthodes traditionnelles qui impliquent l'examen minutieux des spécimens physiques. Ces méthodes, largement basées sur des caractéristiques morphologiques, requièrent une expertise spécialisée et une formation approfondie en entomologie. Il existe différentes techniques de classification, en voici une liste non-exhaustive :

— *Identification morphologique :* Il s'agit d'une méthode qui repose sur l'analyse de caractéristiques physiques des insectes [10]. Les chercheurs examinent des traits spécifiques tels que la morphologie externe ou interne. Des traits tels que la forme et la taille des ailes, la structure des antennes, le nombre et la disposition des pattes, etc. Dans certains cas, l'examen des structures internes, telles que les organes reproducteurs ou les systèmes digestifs, est nécessaire pour des identifications précises. Cela peut donc impliquer la dissection des spécimens pour avoir une analyse approfondie.

— *Clés dichotomiques :* Les clés dichotomiques [11] sont un outil présentant une série de questions binaires qui permettent aux utilisateurs de prendre des décisions basées sur des caractéristiques observables. Chaque option conduit vers une autre question jusqu'à l'identification finale de l'espèce. Ce processus peut être long et nécessite une connaissance préalable des traits caractéristiques des groupes d'insectes. Cette méthode peut devenir complexe lorsqu'il s'agit d'espèces très similaires.

— *Études morphométriques :* Les études morphométriques consistent à mesurer quantitativement les dimensions et les proportions des structures corporelles des insectes. [12] Cette approche permet d'analyser les variations morpho-

logiques au sein et entre les espèces. Les chercheurs utilisent des techniques statistiques pour établir des relations entre les mesures et identifier des modèles qui peuvent indiquer des différences spécifiques à des espèces.

— *Analyse de la variation génétique :* La classification traditionnelle s'est également enrichie de l'analyse de la variation génétique. [13] Les chercheurs utilisent des techniques moléculaires, comme le séquençage de l'ADN, pour évaluer les différences génétiques entre les populations d'insectes. Cela permet de clarifier des ambiguïtés taxonomiques lorsque les traits morphologiques sont peu fiables.

Bien que ces approches soient efficaces, elles présentent des limites. La diversité morphologique au sein des espèces rend parfois difficile l'identification précise des spécimens. De plus, la manipulation physique peut compliquer le traitement rapide d'importantes collections. L'expertise humaine est essentielle dans ces méthodes mais elle peut également introduire des erreurs, notamment lorsque les spécialistes sont confrontés à des échantillons mal conservés.

2.2 Apprentissage profond et vision par ordinateur

L'apprentissage profond (ou communément appelé *deep learning*) est une sous-catégorie de l'intelligence artificielle, qui permet à un ordinateur d'apprendre et d'interpréter des données complexes comme des images. Il repose principalement sur des réseaux de neurones artificiels, qui simulent en quelque sorte la façon dont le cerveau humain traite et reconnaît des motifs dans les données. Voici une présentation générale du fonctionnement de l'apprentissage profond et de la vision par ordinateur, suivie d'une explication plus détaillée des réseaux de neurones convolutifs (CNN), qui sont nécessaires pour analyser des images et faire des prédictions.

2.2.1 Concepts de Base

L'apprentissage profond utilise des réseaux de neurones multicouches, organisés en plusieurs couches interconnectées d'unités de calcul appelées *neurones artificiels*. [14] Ces réseaux sont entraînés pour reconnaître et interpréter des modèles complexes en passant par trois phases principales :

I. *Phase d'Entraînement :* Le réseau de neurones est exposé à de grandes quantités de données étiquetées, c'est-à-dire des données pour lesquelles on connaît déjà les résultats souhaités. Par exemple, si l'objectif est de classer des images d'insectes, chaque image d'entraînement est labelisée avec le nom

de l'espèce. Le réseau utilise ces données pour apprendre des relations et des caractéristiques spécifiques aux images, en ajustant ses paramètres internes au fil du temps. Ceux-ci sont principalement les poids et les biais des connexions entre les neurones. Cette phase vise donc à apprendre un modèle initial en identifiant des patterns dans les données étiquetées.

II. *Propagation avant et arrière :* Dans chaque phase d'entraînement, le réseau applique un processus de propagation avant, passant l'image de couche en couche. À chaque étape, le réseau identifie et analyse des caractéristiques de plus en plus fines et abstraites (les bords, les motifs, puis des formes plus complexes). Après cela, il utilise une rétropropagation des erreurs, qui permet de corriger les erreurs en ajustant les poids et les biais des neurones, ce qui affine ses prédictions. Cette étape correspond au cœur de l'apprentissage : la propagation avant génère une prédiction, et la propagation arrière ajuste les paramètres pour corriger les erreurs.

III. *Apprentissage par itération :* Le réseau de neurones procède par itérations (appelées epochs) pour ajuster progressivement ses paramètres et optimiser sa capacité à reconnaître les images avec précision. Chaque itération rapproche le modèle du résultat souhaité, et plus le nombre de données d'entraînement est grand, plus le modèle devient performant et précis. Cette phase consiste donc à améliorer le modèle par des passages successifs sur les données pour minimiser l'erreur globale.

2.2.2 Les Réseaux de Neurones Convolutifs

Les réseaux de neurones convolutifs (en anglais, CNN pour *Convolutional Neural Networks*) sont des types de réseaux de neurones spécifiquement conçus pour le traitement et l'analyse d'images. Un réseau de neurones est un modèle d'apprentissage automatique inspiré du fonctionnement du cerveau humain. Il est constitué d'unités de calcul organisées en couches. Ces couches sont interconnectées et permettent au réseau d'apprendre des relations complexes entre les données d'entrée et les résultats souhaités.
Un CNN décompose une image en petits morceaux pour en extraire des caractéristiques essentielles, permettant ainsi de reconnaître des objets, des formes, et des motifs complexes dans une image. Un CNN est formé par un empilement de couches distinctes, chacune ayant une fonction spécifique pour traiter et transformer les données. [2]

I. *Couche de convolution (CONV) :* La première étape d'un CNN consiste en l'application d'une opération mathématique appelée *convolution* sur l'image. Cette opération consiste à faire passer des filtres sur des portions de l'image, appelées champs récepteurs, pour extraire des caractéristiques spécifiques,

comme des bords ou des motifs. Trois hyperparamètres contrôlent la sortie de cette couche :

— **Profondeur :** nombre de filtres appliqués simultanément. Chaque filtre détecte une caractéristique précise.

— **Pas :** il contrôle le chevauchement des champs récepteurs. Plus il est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.

— **Marge :** ajout de zéros autour de l'image pour contrôler la taille de la sortie.

Dans l'apprentissage automatique, un hyperparamètre est un paramètre dont la valeur est utilisée pour contrôler le processus d'apprentissage.

La sortie de cette couche, appelée *volume de sortie* est tridimensionnelle, elle inclut les valeurs de hauteur, largeur et profondeur des filtres. Le partage des paramètres de filtrage entre les neurones d'un noyau de convolution (ou *patch*) permet de rendre le modèle invariant par translation. L'idée est que si un filtre fonctionne bien en haut à gauche de l'image, il sera tout aussi efficace en bas à droite. Ainsi, tous les neurones d'une couche de convolution partagent les mêmes poids et biais. Chaque tranche (ou noyau) de cette couche effectue une convolution sur l'image d'entrée, produisant une image intermédiaire. Plusieurs noyaux permettent de créer une couche de traitement, et l'empilement de couches génère l'image finale.

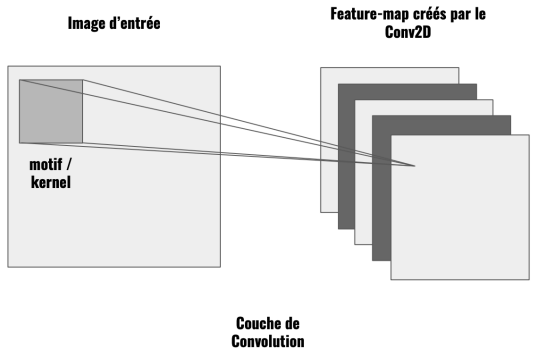


FIGURE 2.1 – Ilustration de la couche de convulation dans un CNN, montrant l'application d'un motif sur une image d'entrée pour générer plusieurs cartes de caractéristiques (feature-map) à l'aide de l'opération Conv2D. [1]

En effectuant plusieurs convolutions successives, le CNN détecte des caractéristiques de plus en plus sophistiquées, telles que les formes et les textures propres aux différents objets dans l'image. Il va d'abord détecter des caractéristiques simples, comme les bords de l'image, et ensuite, il identifiera des

formes plus complexes comme des antennes ou des ailes d'insectes dans le cas de reconnaissance d'image d'insectes.

II. *Couche de pooling (POOL) :* La couche de pooling permet de réduire la taille des images traitées tout en conservant les informations essentielles. [2] Concrètement, elle divise l'image en petits carrés et applique une opération sur chaque carré pour en extraire une seule valeur, souvent la valeur maximale (appelée *max-pooling*). Cela réduit la quantité de paramètres, aide à prévenir le sur-apprentissage et rend le modèle plus robuste aux variations de position de l'objet dans l'image.
Le max-pooling 2x2, par exemple, utilise des tuiles de 2x2 pixels et garde la valeur maximale de chaque tuile. Cela divise la taille de l'image par quatre, ce qui est assez courant dans les CNN. D'autres types de pooling existent, comme le average pooling, qui prend la moyenne des valeurs de chaque tuile. Cependant, le max-pooling est généralement plus efficace, car il accentue les "activations fortes" ou les éléments importants dans l'image.

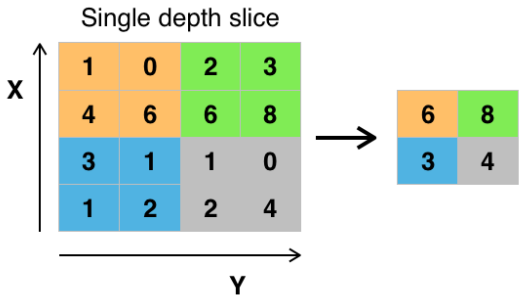


FIGURE 2.2 – Illustration du processus de max pooling avec un filtre 2x2 et un pas de 2. [2]

Le pooling peut donc apporter de gros gains en calcul, mais il réduit aussi l'information, ce qui peut limiter la précision.

III. *Couche de correction :* Une fonction mathématique peut être opérée entre les différentes couches afin d'améliorer l'efficacité du traitement. [15] Il en existe plusieurs :

- La correction ReLU (Unité Linéaire Rectifiée) : $f(x) = max(0, x)$. Cette fonction améliore l'efficacité du réseau en amplifiant les caractéristiques essentielles. Cette fonction est souvent privilégiée pour accélérer l'apprentissage sans compromettre la précision.
- La correction par tangente hyperbolique : $f(x) = tanh(x)$
- La correction par la angente hyperbolique saturante : $f(x) = |tanh(x)|$
- La correction par la fonction sigmoïde : $f(x) = (1 + e - x) - 1$

IV. *Couche entièrement connectée (FC) :* Les caractéristiques extraites sont finalement envoyées à une ou plusieurs couches entièrement connectées, où chaque neurone est connecté à tous les neurones de la couche précédente. [15] Cette étape est cruciale pour associer les caractéristiques obtenues à des classes spécifiques d'objets (par exemple, les différentes espèces d'insectes). Ces couches agissent comme des classificateurs, permettant au CNN de prédire des classes précises pour les objets dans une image.

V. *Couche de parte (LOSS) :* La dernière couche du CNN mesure l'écart entre les prédictions du modèle et les vraies valeurs. [2] La perte Softmax est souvent utilisée pour les problèmes de classification. En minimisant cette perte lors de l'entraînement, le CNN ajuste ses paramètres pour améliorer sa précision.

Avec ces différentes couches, le CNN apprend à identifier et à classifier des motifs dans des images.

Régularisation et Dropout

Afin d'éviter le sur-apprentissage, une technique de régularisation, appelée *Dropout*, consiste à désactiver aléatoirement un pourcentage de neurones (unités de calcul) pendant l'entraînement. [2] Cela empêche le modèle de trop dépendre de certaines connexions ou caractéristiques spécifiques et favorise ainsi une meilleure généralisation sur de nouvelles données. Pendant l'entraînement, les neurones désactivés ne participent pas à la propagation des informations, ce qui force le réseau à apprendre des représentations plus robustes.

Applications des CNN

Aujourd'hui, les CNN sont utilisés dans une multitude de domaines comme la reconnaissance faciale, la classification des images médicales ou encore la conduite autonome. Leur capacité à identifier des objets avec précision dans des images en fait un outil essentiel pour la vision par ordinateur.

2.3 Modèles de détection d'objets

Dans le cadre de la détection d'objets, deux modèles de détection se démarquent par leur efficacité et leur utilisation : YOLO et Faster R-CNN. Ces modèles diffèrent par leur approche, chacun offrant des avantages de vitesse ou de précision.

2.3.1 YOLO

Le modèle YOLO (*You Only Look Once*, traduit « Tu ne regardes qu'une fois »), est un algorithme d'apprentissage profond introduit en 2016 par Joseph Redmon. [16] Conçu pour la détection d'objets en temps réel, il a permis une avancée majeure dans des applications nécessitant une vitesse et une précision

élevées. Sa structure repose sur une approche unique où une seule passe sur l'image est nécessaire pour détecter et classifier les objets, contrairement à de nombreux autres algorithmes qui fonctionnent en deux étapes.

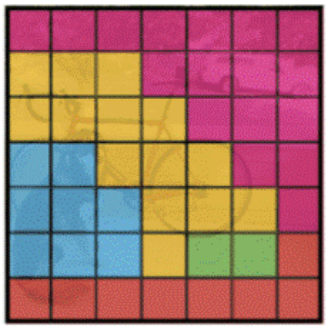
L'algorithme fonctionne comme ceci : il divise l'image d'entrée en une grille de cellules. Chaque cellule de cette grille est responsable de prédire la présence ou non d'objets dans sa zone et génère des boîtes englobantes avec une probabilité associée. [16] Cela signifie que chaque cellule de la grille contribue à localiser les éventuels objets dans son périmètre.

FIGURE 2.3 – Illustration du quadrillage d'image par YOLO. [3]

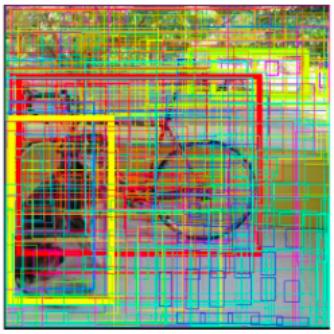
Dans chacune de ces cellules, YOLO utilise une combinaison de réseaux de convolution et de couches entièrement connectées pour détecter les objets et déterminer leur position avec des boîtes de cadrage. Une cellule peut prédire plusieurs boîtes, mais une boîte n'est conservée que si la probabilité de la présence d'un objet est suffisamment élevée. Cette probabilité (ou score de confiance) correspond donc à la certitude du modèle qu'un objet est présent dans la boîte détectée.

FIGURE 2.4 – Illustration de la détection et de la localisation des objets par YOLO. [3]

YOLO associe ensuite une probabilité de classe à chaque boîte retenue, définissant quel type d'objet il a détecté. [3] Cette étape est réalisée via une carte de probabilité qui mesure la probabilité d'appartenance de chaque boîte à une classe spécifique, par exemple, un chien, un vélo ou une voiture comme dans la Figure 2.4.



(a) Exemple de carte de probabilité générée par YOLO.



(b) Exemple de détection des types d'objets par YOLO.

FIGURE 2.5 – Carte de probabilité et exemple de détection par YOLO. [3]

Enfin, YOLO utilise un procédé de *Non-Maxima Suppression* (NMS) pour éliminer les détections superflues. [3] Cette méthode va conserver la boîte avec la plus haute probabilité et supprimer toutes les boîtes ayant un indice de confiance trop faible ou une trop grande superposition (mesurée par l'IoU).

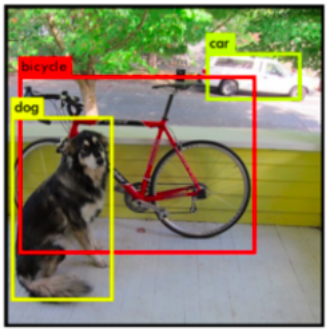


FIGURE 2.6 – Exemple d'image après suppression des détections superflues par YOLO. [3]

L'IoU (*Intersection over Union*) est une métrique qui permet de mesurer la précision de la localisation de l'objet en comparant l'aire d'intersection de la boîte détectée avec la boîte réelle (étiquetée manuellement). Un IoU de 1 indique une superposition parfaite et un IoU de 0 indique une absence totale de chevauchement.

2.3.2 Faster R-CNN

Le modèle Faster R-CNN (*Region-based Convolutional Neurol Network*, traduit Réseau neuronal convolutif basé sur les régions) est une méthode de détection d'objets introduite par Shaoqing Ren en 2015. Conçu pour améliorer l'efficacité des versions précédentes (comme Fast R-CNN), il intègre un réseau de propositions de régions (*Region Proposal Network*, RPN), qui génère dynamiquement des régions d'intérêt (*Region of Interest*, ROI) contenant potentiellement des objets. Ce modèle est largement utilisé dans les applications où la précision de la localisation prime sur la rapidité.
La méthode Faster R-CNN s'appuie sur une approche en deux étapes : premièrement, elle génère des propositions de régions susceptibles de contenir des objets, puis elle affine et classe ces régions pour identifier précisément chaque objet.

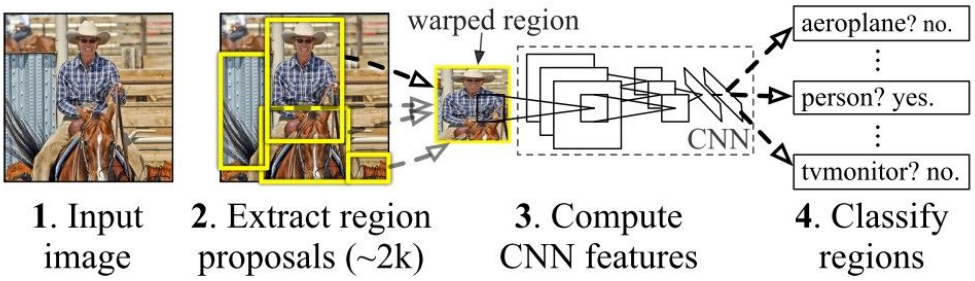


FIGURE 2.7 – Illustration du fonctionnement de Faster R-CNN. [4]

Dans la première étape, Faster R-CNN utilise le RPN pour parcourir l'image et produire des boîtes candidates appelées propositions de régions. [17] Ce réseau extrait des caractéristiques de l'image à l'aide de couches convolutionnelles et évalue la probabilité de présence d'un objet dans chaque proposition, ainsi que ses coordonnées. Contrairement à YOLO qui utilise une grille fixe, Faster R-CNN est flexible et génère dynamiquement les régions en fonction du contenu de l'image.

Ensuite, dans la seconde étape, les propositions de régions sont traitées par une couche de détection qui applique un réseau neuronal convolutif pour analyser en détail chaque région, affiner les prédictions et attribuer une classe aux objets détectés. Pour chaque région, Faster R-CNN produit une boîte englobante et une probabilité de classification. [17]

Enfin, le modèle utilise également la méthode de NMS pour éliminer les doublons de détection [17], comme dans YOLO, en conservant la boîte avec le plus haut score de confiance pour chaque objet détecté et en supprimant les autres boîtes ayant une grande superposition. La précision de Faster R-CNN est mesu-

rée par l'Intersection over Union (IoU). les régions en fonction du contenu de l'image.

Faster R-CNN est donc un modèle adapté aux applications nécessitant une détection précise et fiable, bien qu'il soit généralement plus lent que YOLO en raison de son processus en deux étapes [4]

2.4 Modèles de classification d'images

Les modèles de classification d'images sont des réseaux de neurones conçus pour analyser des images et leur attribuer une étiquette, permettant ainsi d'identifier le contenu de celles-ci. Ils fonctionnent en extrayant des caractéristiques spécifiques à l'image et en les analysant à travers différentes couches de neurones afin de prédire la catégorie qui correspond le mieux à l'image observée. Les deux modèles présentés ci-dessous sont également des CNN mais mieux adaptés pour la classification.

2.4.1 ResNet

ResNet (*Residual Network*), introduit en 2015 par Microsoft, est une architecture de réseau de neurones profonds conçue pour résoudre le problème du *Vanishing gradient*, qui rend l'entraînement difficile dans les réseaux très profonds. [18] Le Vanishing gradient est un phénomène où, lors de la propagation arrière, les gradients deviennent de plus en plus petits à mesure qu'ils se propagent à travers les couches du réseau. Cela empêche les couches profondes d'apprendre efficacement, car les poids ne sont pas ajustés de manière significative. Dans un réseau classique, l'ajout de couches augmente les risques que l'information soit diluée à chaque couche, diminuant ainsi la précision. ResNet utilise des connexions résiduelles pour contourner ce problème.

Habituellement, chaque couche transforme complètement ses données d'entrée en sortie. ResNet introduit une nouvelle approche : au lieu de rendre dépendante chaque couche de la sortie à la couche précédente, il introduit des connexions résiduelles (ou *skip connections*). Ces connexions permettent de « sauter » certaines couches, ajoutant l'entrée initiale aux sorties des couches ultérieures. Ce mécanisme va créer des blocs résiduels, où le réseau apprend la différence entre l'entrée et la sortie attendue, au lieu de transformer entièrement l'entrée.

Les connexions résiduelles permettent d'entraîner des réseaux plus profonds sans dégrader les performances, car elles facilitent la propagation de l'information et des gradients, ce qui permet d'éviter des problèmes de Vanishing gradient. Cela signifie que ResNet peut comporter de nombreuses couches et rester très performant.

L'efficacité de ResNet se démarque pour des tâches où la reconnaissance de détails complexes est cruciale, comme la classification d'images haute résolution. [18] Grâce à sa capacité d'apprentissage d'une représentation très détaillée, il peut capturer les caractéristiques subtiles et sous-jacentes d'un objet dans une image. Par conséquent, non seulement ResNet a amélioré la précision dans de nombreux standards de classification d'images, mais il a également inspiré des architectures encore plus avancées pour des tâches complexes de vision par ordinateur, telles que la segmentation d'images et la détection d'objets. ResNet reste un modèle fondamental pour les réseaux de neurones profonds, car il améliore la stabilité et la précision du modèle même dans les architectures très profondes, tout en les rendant plus efficaces grâce aux connexions résiduelles.

2.4.2 EfficientNet

EfficientNet, introduit en 2019 par des chercheurs de Google, est une architecture de réseau de neurones conçue pour optimiser l'équilibre entre efficacité et performance. [19] Contrairement aux modèles traditionnels qui nécessitent souvent des compromis entre précision et ressources, EfficientNet utilise une approche novatrice appelée *compound scaling*, traduit scalabilité composée [20].

Ce concept repose sur le fait de modifier les trois dimensions fondamentales d'un réseau [19] : la largeur, la profondeur et la résolution.

— La largeur détermine le nombre de canaux dans chaque couche. En augmentant celle-ci, le modèle peut apprendre des caractéristiques plus complexes. A l'inverse, la réduire conduit à un modèle plus léger, adapté aux environnements à plus faibles ressources.

— La profondeur fait référence au nombre de couches. Des modèles plus profonds peuvent capturer des représentations plus riches, mais nécessitent plus de ressources. Des modèles avec une profondeur moins élevée sacrifient de la précision pour de la performance.

— La résolution concerne la taille des images d'entrée. Des images de plus haute résolution fournissent plus d'informations, mais encore une fois, demandent plus de puissance de calcul.

EfficientNet utilise également un coefficient de composition (noté φ) pour adapter ces dimensions de manière équilibrée. [19] En ajustant ce coefficient, il est possible des créer des variantes en fonction des besoins en matière de performances et de ressources.

Au niveau de son architecture, EfficientNet utilise des couches de types *Mobile Inverted Bottleneck* (MBConv) [19], qui sont une combinaison de convolutions séparables en profondeur et des blocs résiduels inversés. Les convolutions séparables en profondeur décomposent une convolution classique en deux étapes, réduisant ainsi le nombre de paramètres et les ressources utilisées tout en maintenant la capacité d'apprentissage tandis que les blocs résiduels inversés permettent au modèle d'apprendre des représentations plus complexes en facilitant le passage d'informations à travers le réseau. Cette architecture intègre également un mécanisme appelé *Squeeze-and-Excitation* (SE) qui aide le modèle à se concentrer sur des caractéristiques pertinentes en apprenant des dépendances entre les canaux de caractéristiques.

EfficientNet permet d'avoir des performances impressionnantes dans de nombreuses applications de vision par ordinateur, surpassant les modèles précédents tout en utilisant moins de paramètres. Cette efficacité en fait un choix privilégié pour diverses applications.

De nombreuses applications de ce modèle existent dans divers domaines. Par exemple, dans l'analyse biomédicale [21], en utilisant la détection et la classification de cellules anormales ou de tumeurs sur des images médicales, où des détails subtils doivent être analysés avec précision. Ce modèle est également utilisé dans l'industrie alimentaire, pour la reconnaissance faciale, dans l'agriculture, dans la sécurité, etc.

Chapitre 3

Objectifs et matériel

3.1 Objectifs

L'objectif de ce projet est de développer un système semi-automatisé capable de détecter et de classifier des insectes à partir d'images provenant d'insectes se situant dans des collections entomologiques. Cet outil permettra de quantifier le nombre d'insectes présents dans chaque boîte et d'effectuer ensuite un tri des spécimens par classe, ordre, famille, genre et espèce. En intégrant des algorithmes avancés de détection et de classification, le système visera à atteindre une précision élevée tout en traitant efficacement de grandes quantités de données. Les données résultantes seront analysées afin d'établir des statistiques sur la précision et la fiabilité de l'outil développé. Cette évaluation est essentielle pour valider les performances du projet et pour identifier les axes d'amélioration. Ce travail vise ainsi à contribuer à l'évolution de l'entomologie contemporaine, en facilitant la gestion des collections muséales et en enrichissant notre compréhension de la biodiversité, un élément essentiel à la préservation de l'équilibre de notre planète et à la lutte contre les menaces qui pèsent sur les écosystèmes.

3.2 Matériel utilisé

Cette section décrit les ressources et outils utilisés pour la détection et la classification d'insectes. La méthodologie repose sur la préparation de photos de boîtes, ainsi que l'utilisation de langages et bibliothèques spécialisées en vision par ordinateur et en apprentissage profond.

3.2.1 Données d'entrées

Les données utilisées dans ce projet sont des photographies de boîtes entomologiques conservées à l'AfricaMuseum et à l'Institut royal des sciences naturelles de Belgique (IRSNB).
Ces boîtes ont été numérisées à l'aide d'un système photographique professionnel comprenant un appareil photo Canon 5DSR associé à un objectif Canon EF 40mm. Les boîtes étaient photographiées individuellement dans une armoire blanche, équipée de panneaux LED assurant un éclairage uniforme. L'ensemble du système a permis de capturer des images haute résolution (8688 x 5792 pixels) de chaque boîte.

La collection des boîtes utilisées comprend :

— 3365 photos de boîtes entomologiques contenant des spécimens d'espèces de Coleoptera.

— 4812 photos de boîtes entomologiques contenant des spécimens d'espèces de Lepidoptera.

Pour l'étape de détection, l'ensemble de ces boîtes est utilisé. Pour l'étape de classification un échantillon spécifique a été retenu. Cet échantillon comprend 14 espèces de Coleoptera et 33 espèces de Lepidoptera (voir annexe B et C).

3.2.2 Langages de programmation et bibliothèques

Dans ce projet, le langage de programmation Python a été choisi pour son efficacité, sa simplicité d'utilisation et ses nombreuses bibliothèques spécialisées en apprentissage autonome. Python est largement utilisé dans le domaine de l'intelligence artificielle pour sa flexibilité et son vaste écosystème, offrant de nombreux outils pratiques pour la manipulation de données et le développement de modèles. De nombreuses bibliothèques ont été utilisées dans ce projet, voici un aperçu de quelques-unes d'entre elles :

— *PyTorch :* Il s'agit d'un framework de Deep Learning utilisé pour la création et l'entraînement de modèles. Dans ce projet, il permet d'utiliser et d'entraîner le modèle YOLO.

— *TorchVision :* Elle fournit des modèles pré-entraînés. Dans le cadre de ce projet, elle a été employée pour la détection d'insectes avec le modèle Faster R-CNN. Cette bibliothèque inclut également des transformations pour le traitement de l'augmentation des images, permettant ainsi de renforcer la diversité des données d'entraînement.

— *Ultralytics :* Cette implémentation de YOLO, basée sur PyTorch, a facilité l'entraînement des modèles.

— *TensorFlow :* Cette bibliothèque et son API Keras ont été utilisés pour la classification d'insectes.

3.2.3 LUCIA

LUCIA, un super-ordinateur du CÉCI (Consortium des Équipements de Calcul Intensif), a joué un rôle clé dans ce projet. Hébergé par le Cenaero et financé par le Fonds de la Recherche Scientifique ainsi que par la Région Wallonne, LUCIA est un cluster de calcul intensif conçu pour exécuter des programmes complexes et gourmands en termes de ressources. [22] Il permet de gérer de grandes quantités de données et d'effectuer de nombreux calculs en parallèle.

Dans le cadre de ce projet, LUCIA a permis d'entraîner et exécuter les modèles de Deep Learning pour la détection et la classification d'insectes. Grâce à sa puissance de calcul distribuée, LUCIA a permis de gérer efficacement les ressources nécessaires pour entraîner les modèles complexes. Des opérations qui auraient pu durer des jours ont été exécutées en quelques minutes grâce à cet outil.

En outre, son utilisation est relativement simple, il suffit de spécifier les ressources nécessaires (comme le nombre de processeurs, la mémoire) à l'aide des options disponibles, puis d'exécuter le programme normalement. Par exemple, la commande suivante :

```
srun --account=****** --cpus-per-task=32 --time=40:00 --pty --mem=200G --gpus-per-node=4 --partition=gpu python exemple.py
```

FIGURE 3.1 – Exemple de lancement d'un script sur LUCIA.

Cette commande indique que l'exécution du script « exemple.py » sera effectué avec 32 CPU, 200 Go de mémoire, 4 GPU, et un temps d'exécution maximal de 40 minutes.

Chapitre 4

Design expérimental

4.1 Description du programme existant

Ce travail intervient dans la suite d'un programme fait antérieurement qui se concentrait uniquement sur la détection de spécimens d'insectes provenant de boîtes entomologiques. Ce chapitre décrit l'état du programme tel qu'il l'était.

Ce programme est divisé en plusieurs parties. La première partie permet l'annotation des boîtes afin d'avoir une base de données d'entraînement. Les deux autres parties utilisent les deux algorithmes de détection vus précédemment (YOLO et Faster R-CNN).

4.1.1 Annotation des boîtes

L'objectif de cet outil est de faciliter l'annotation des insectes dans des images en permettant à l'utilisateur de sélectionner des paramètres de détection (couleur, cadrage, etc.) et de visualiser les résultats en temps réel.

Ce programme utilise l'interface graphique de PyQt5 et la bibliothèque OpenCV pour permettre à l'utilisateur de sélectionner des images et de créer des annotations basées sur des contours dessinés autour d'insectes présents sur l'image.

L'utilisateur commence par sélectionner une image de boîte entomologique.

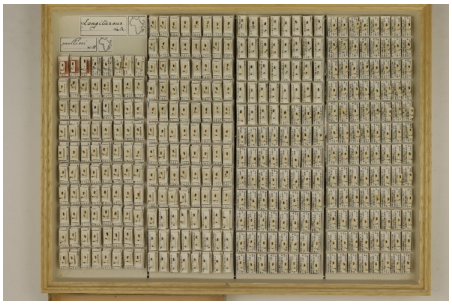
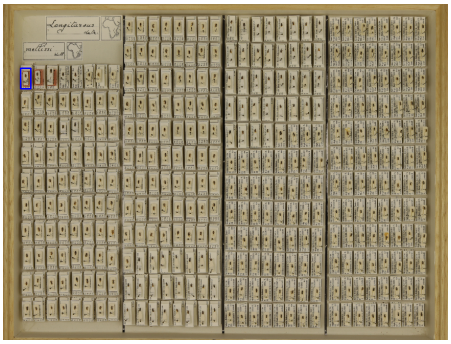
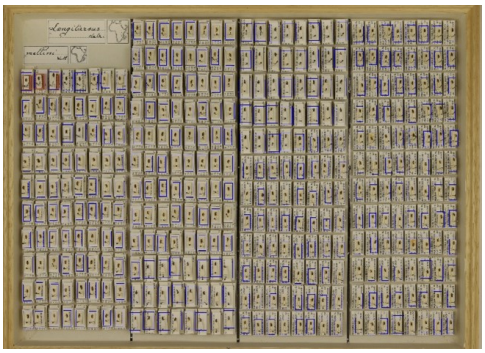


FIGURE 4.1 – Exemple de photographie de boîte entomologique.

Il peut ensuite décider d'ajuster des paramètres afin d'avoir une vision plus claire de l'image.
Après cette étape, qui peut être facultative, l'utilisateur annote chaque insecte présent dans la boîte en l'encadrant simplement à l'aide de sa souris.



(a) Exemple de boîte avec une annotation.



(b) Exemple de boîte complètement annotée.

FIGURE 4.2 – Exemple de boîte partiellement et complètement annotée.

Le résultat de cette annotation est sous la forme d'un fichier texte comprenant l'ensemble des coordonnées des encadrements d'insectes.

```
0 0.377 0.7695 0.086 0.059
0 0.7116666666666667 0.7295 0.08066666666666666 0.053
0 0.379 0.729 0.09 0.058
0 0.6343333333333333 0.7235 0.082 0.053
0 0.46166666666666667 0.7215 0.082 0.053
0 0.28633333333333333 0.6945 0.09266666666666666 0.057
0 0.375 0.689 0.094 0.06
0 0.46166666666666667 0.6765 0.08333333333333333 0.057
0 0.549 0.669 0.07933333333333334 0.054
0 0.13566666666666666 0.6665 0.07933333333333334 0.055
0 0.2896666666666667 0.6565 0.08866666666666667 0.057
0 0.37933333333333336 0.645 0.09333333333333334 0.06
0 0.4663333333333333 0.631 0.086 0.056
0 0.206 0.631 0.07466666666666667 0.056
0 0.132 0.629 0.08133333333333333 0.06
0 0.8066666666666666 0.6215 0.088 0.055
```

FIGURE 4.3 – Exemple de fichier d'annotation de boîte entomologique.

4.1.2 Implémentation de YOLO

Dans ce projet, le premier algorithme utilisé pour la détection d'insectes est YOLO, le modèle YOLOv8 plus précisément. Bien que ce modèle soit pré-entrainé, il ne prend pas en compte spécifiquement les insectes, qui ne sont pas inclus dans les classes d'objets sur lesquelles il a été initialement entraîné. C'est pourquoi il est nécessaire de personnaliser celui-ci.

L'étape la plus importante de la personnalisation est la création d'un jeu de données adapté aux insectes. Pour ce faire, un dossier comportant les images de boîtes entomologiques ainsi qu'un dossier comportant les annotations de ces images (fichiers texte avec les coordonnées précises des insectes) ont été créés. Il y a un total de 40 boîtes entomologiques annotées. Ces dossiers sont ensuite divisés en deux ensembles : un ensemble d'entraînement (80%) et un ensemble de validation (20%), afin de permettre au modèle de s'adapter aux données et d'évaluer sa performance sur des données non vues pendant l'entraînement.

L'entraînement du modèle est effectué en utilisant cette commande :

```
yolo task=detect mode=train model=yolov8x.pt data=custom_data.yaml epochs=100
imgsz=2000 plots=True device=0,1,2,3 close_mosaic=100
```

FIGURE 4.4 – Commande de lancement de l'entraînement de YOLO.

Voici une explication détaillée de chaque élément de la commande :

— *task=detect* : Ce paramètre indique que le modèle est utilisé pour une tâche de détection d'objets.

— *mode=train* : Il définit le mode de fonctionnement sur entraînement.

— *model=yolov8x.pt* : Il indique le modèle pré-entrainé utilisé.

— *data=custom_data.yaml* : Il s'agit du fichier de configuration contenant les informations sur le jeu de données. Ce fichier spécifie où sont situés les dossiers d'entraînement et de validation.

— *epochs=100* : Nombre total d'epoch d'entraînement. Une epoch est un passage complet de tout le dataset d'entraînement à travers le modèle.

— *imgsz=2000* : Taille des images utilisées pour l'entraînement. Plus la résolution est élevée, plus la précision sera élevée mais ce sera également plus gourmand en ressources.

— *plots=True* : Active la génération de graphiques pendant et après l'entraînement.

— *device=0,1,2,3* : Spécifie les GPU utilisés pour l'entraînement.

— *close_mosaic=100* : Désactive l'utilisation de la mosaïque pour toutes les epochs. La mosaïque est une technique d'augmentation de données utilisée pour améliorer la généralisation.

Pendant cet entraînement, le modèle utilise donc les images d'entraînement pour apprendre à détecter les insectes. À chaque epoch, le modèle ajuste ses poids pour réduire l'écart entre ses prédictions et les annotations réelles, ce qui minimise la fonction de perte.

Après chaque epoch, le modèle est évalué pour calculer des métriques telles que la précision (proportion de prédictions correctes parmi toutes celles faites), le rappel (proportion des objets correctement détectés parmi tous les objets présents dans les annotations) et le mAP (*Mean Average Precision*, mesure globale de performance, calculée en tenant compte des différentes valeurs de seuils de confiance). La précision et le rappel permettent de mesurer la performance du modèle tandis que le mAP offre une vue d'ensemble de sa capacité à détecter les insectes à différents seuils de confiance.

Le modèle produisant les meilleures performances est enregistré sous le nom « *best.pt* » et est ensuite utilisé pour effectuer des prédictions sur de nouvelles images.

L'étape de prédiction consiste à détecter les insectes dans des images non vues et à enregistrer les résultats dans un fichier Excel. Le script, permettant la prédiction, utilise le modèle entraîné (best.pt) et applique les prédictions sur chaque image du dossier de test. Les paramètres de prédictions incluent un seuil de confiance, un seuil IoU et une taille d'image pour garantir une précision optimale. Dans ce cas-ci, le seuil de confiance est à 0.5, le seuil IoU à 0 et l'imgsz à 2000.

Pour une image prédite, la ligne correspondante dans le fichier Excel ressemble à ceci :

method	model	dir	conf	iou	imgsz	max_det	path	pixx	pixy	nb	area_tot	w_mean	h_mean
YOLO	runs/detect/train	/gpfs/home/acad/ucl-ingi/nde	0.5	0	2000	1000	Alticidae_Drawer_0063.JPG	6000	4000	368	5758098	8.676.902.173.913.040	17.714.945.652.173.900

FIGURE 4.5 – Exemple de résultat pour une prédiction par YOLO.

Les sorties comprennent la méthode utilisée (*method*), le modèle utilisé (*model*), le dossier comportant l'image (*dir*), le seuil de confiance (*conf*), le seuil d'IoU (*iou*), la taille d'image (*imgz*), le nombre maximum de détections par image (*max_det*), le chemin de l'image (*path*), les dimensions de l'image (*pixx, pixy*), le nombre d'insectes détecté (*nb*), la surface totale détectée (*area_tot*) et la largeur et hauteur moyennes (*w_mean et h_mean*). Ces résultats constituent une base pour évaluer la performance du modèle et effectuer des comparaisons avec d'autres méthodes.

4.1.3 Implémentation de Faster R-CNN

Le second algorithme utilisé pour la détection d'insectes est Faster R-CNN. Tout comme YOLO, ce modèle pré-entraîné nécessite une phase de personnalisation pour être utilisé dans le cadre de la détection d'insectes.

De la même manière qu'avec le modèle YOLO, deux dossiers comportant des images et leurs annotations ont été créés. La répartition est également la même, 80% pour l'entraînement et 20% pour la validation.

Pour l'entraînement du modèle, Faster R-CNN utilise une architecture basée sur ResNet50, un réseau de neurones convolutif profond utilisé pour extraire des caractéristiques pertinentes des images. De plus, pour gérer les variations de taille des insectes, une FPN (Feature Pyramid Network) est intégrée. [4] Cette composante permet au modèle d'extraire des informations à plusieurs échelles en combinant des caractéristiques issues de différentes couches du réseau.
Les principales étapes de l'entraînement sont :

- Le chargement du modèle pré-entrainé : le modèle est initialisé avec des poids pré-entraînés. Dans ce cas-ci, la tête de classification (la dernière couche) est adaptée pour ne détecter que deux classes : les insectes ou l'arrière-plan.
- La préparation des données : Les images et les annotations sont converties en tenseurs (format standard utilisé pour entraîner des modèles de Deep Learning) et un DataLoader personnalisé est créé pour charger les données par lots (batch size). 16 images par lot pour l'entraînement et 8 images par lot pour la validation.
- L'entraînement personnalisé : le modèle est entraîné sur 100 epochs avec l'optimiseur SGD (Stochastic Gradient Descent), qui met à jour les poids du modèle à chaque itération afin de minimiser la fonction de perte. Le taux d'apprentissage (learning rate) est fixé à 0.0001 pour garantir une convergence progressive. À chaque itération, les images et annotations sont passées au modèle. Une fonction de perte est alors calculée en fonction des prédictions par rapport aux annotations réelles.
- La sauvegarde du modèle : le modèle ayant les meilleures performances est sauvegardé sous le nom spécifié par l'utilisateur.

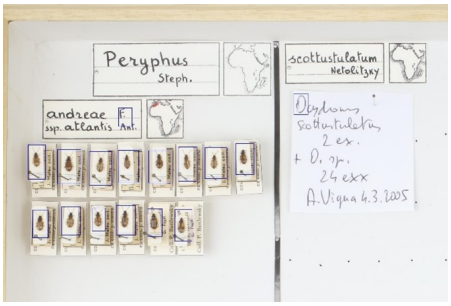
Une fois le modèle entraîné, il est utilisé pour détecter des insectes dans des images non vues. Celui-ci est restauré à partir des poids sauvegardés et configuré en mode évaluation. Chaque image à prédire est ensuite convertie en tenseur et passée au modèle. Le modèle génère des prédictions et un seuil de confiance est appliqué afin de filtrer les détections fiables. Enfin, les résultats de ses prédictions sont enregistrés dans un fichier Excel, dans le même format que pour YOLO.

4.2 Améliorations de la détection des insectes

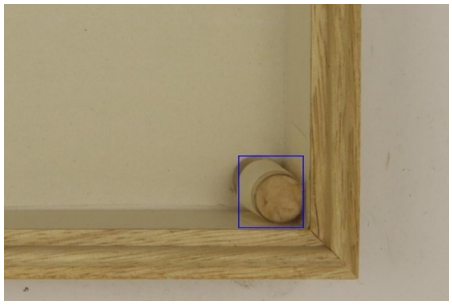
L'un des objectifs de ce projet est d'améliorer la précision des algorithmes de détection en affinant leur capacité à correctement identifier les insectes présents dans les boîtes. En analysant les performances des modèles utilisés, plusieurs pistes d'amélioration ont été explorées pour traiter les problèmes rencontrés, tels que la détection d'objets non pertinents ou la limitation de la taille du jeu de données d'entraînement. Deux approches ont été mises en œuvre : les annotations négatives et le pseudo-labelling. Ces deux techniques visent à renforcer l'efficacité des modèles.

4.2.1 Annotations négatives

Après analyse des résultats des prédictions faites par les deux algorithmes précédemment mentionnés, il est apparu que des objets non pertinents étaient détectés comme des insectes. Ces objets incluent des défauts sur les boîtes, des étiquettes, des éléments du fond ou autres. Ces erreurs sont fréquentes car ces éléments peuvent parfois être confondus avec des insectes en raison de leurs formes ou textures similaires.



(a) Exemple 1 de mauvaise annotation.



(b) Exemple 2 de mauvaise annotation.

FIGURE 4.6 – Exemples d'erreurs faites par le modèle de détection

Afin de résoudre ce problème et d'améliorer la précision des prédictions, des *annotations négatives* ont été ajoutées au jeu de données d'entraînement. Ces annotations permettent d'apprendre au modèle à ne pas sélectionner certains objets, qui peuvent partager des caractéristiques visuelles similaires à celles des insectes.

Ces annotations désignent des objets ressemblant à des insectes, mais qui ne le sont pas et sont spécifiquement étiquetés comme « non insectes ». En ajoutant ces annotations, le modèle apprend à éviter de classer ces objets comme des insectes, ce qui permet de réduire le taux de faux positifs.

Pour intégrer ces annotations négatives dans le jeu de données, le travail se fait de la même manière que pour les annotations d'insectes. Mais dans ce cas, la sélection se porte uniquement sur les objets qui ne sont pas des insectes.

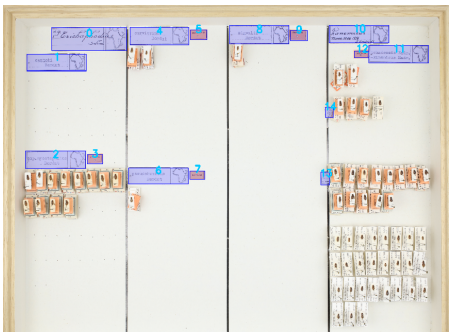


FIGURE 4.7 – Exemple d'annotations négatives sur une boîte entomologique.

4.2.2 Pseudo-labelling

Un des défis majeurs rencontrés dans ce projet est la taille relativement réduite des données d'entraînement par rapport à la taille totale des boîtes entomologiques à classifier. En effet, un jeu de données limité peut entraîner des problèmes de sur-apprentissage (ou *overfitting*), ce qui peut nuire à la précision du modèle. Le sur-apprentissage survient lorsque le modèle apprend trop bien les détails spécifiques et les anomalies du jeu de données d'entraînement, au point qu'il devient trop spécialisé sur ces données et perd la capacité à bien performer sur de nouvelles données. Le modèle devient excellent pour prédire les objets présents dans les images d'entraînement mais échoue à détecter correctement des objets similaires dans des images inconnues.

De plus, la création du jeu de données d'entraînement nécessite des annotations manuelles, une tâche à la fois longue et coûteuse en ressources, surtout lorsque le nombre d'images à annoter est élevé. Afin d'améliorer la qualité de l'entraînement et d'augmenter la diversité du dataset d'entraînement, une approche appelée le pseudo-labelling est utilisée.

Le pseudo-labelling consiste à ajouter des données supplémentaires au jeu de données d'entraînement en utilisant les prédictions faites par le modèle sur des images non étiquetées manuellement. Les images ajoutées sont des images parfaitement prédites, c'est-à-dire que chaque insecte est correctement détecté dans la boîte. L'idée est donc de faire en sorte que le modèle apprenne non seulement à partir

des annotations humaines initiales, mais aussi à partir de ses propres prédictions fiables. Cette technique permet d'enrichir le dataset sans nécessiter de nouvelles annotations manuelles, ce qui est particulièrement utile lorsque le jeu de données d'origine est assez petit. [23]

Ainsi, pour augmenter la taille et la diversité du dataset d'entraînement, des boîtes d'insectes parfaitement prédites ont été ajoutées. Cela permet donc théoriquement au modèle d'améliorer sa capacité de détection.

4.2.3 Autres modifications

Afin de pouvoir améliorer l'efficacité du modèle, 30 boîtes sont annotées en plus. Ce qui permet d'avoir un plus grand jeu de données d'entraînement.
De plus, les coordonnées des insectes sont ajoutées au fichier résultat. Ce qui donne un résultat comme celui-ci :

method	model	dir	conf	iou	imsz	max_det	path	pixx	pixy	nb	area_tot	w_mean	h_mean	boxes
FasterRCNN	model_coleo	/gpfs/home/a	0	0	0	1000	/gpfs/home/a	6000	4000	191	5399389	0	0	[(2682, 2812, 3068, 3309), (3862, 3989, 2837, 3081), (3721, 3855, 1061, 1320)

FIGURE 4.8 – Exemple de fichier résultat d'une détection.

Où la colonne « boxes » est ajoutée et contient l'ensemble des coordonnées des insectes détectés dans l'image de boîte entomologique. Ces coordonnées permettront de découper ces images pour récupérer des images individuelles d'insectes qui permettront de former le dataset d'entraînement de la classification.

4.3 Classification des insectes

La détection des insectes constitue une étape préalable cruciale. La phase suivante consiste à classifier ces insectes détectés en fonction de leur espèce. Cette classification est essentielle pour organiser efficacement les données collectées. Cette section traite des étapes mises en place pour la classification des insectes, de la préparation du jeu de données jusqu'à l'implémentation des modèles.

4.3.1 Préparation du jeu de données

Afin de garantir la qualité de l'entraînement des modèles de classification, un ensemble d'images est constitué en sélectionnant des échantillons variés et représentatifs des insectes détectés lors de la phase de détection. Cette approche vise à s'assurer que le jeu de données couvre un éventail assez large de formes, tailles et espèces.

Une annotation spécifique est réalisée pour chaque image sélectionnée. Ces annotations incluent des informations taxonomiques détaillées, telles que l'ordre, la famille, le genre, l'espèce et le nom scientifique.

Ces informations se trouvent dans un fichier structuré sous la forme suivante :

Image_Name	Order	Family	Genus	Species	Scientific_name
cetoniinae_drawer_0001_crop_1.jpg	Coleoptera	Scarabaeidae	Goliathus	goliathus	Goliathus_goliathus

FIGURE 4.9 – Format du dataset d'entraînement pour la classification.

La diversité des images contribue à améliorer la capacité du modèle à reconnaître des insectes dans de nouvelles données qu'il n'a pas encore vues.
En lien avec ce fichier, il y a également un dossier comprenant les images individuelles des insectes. Ces images sont générées par un script python reprenant les coordonnées des insectes détectés lors de la phase précédente.

FIGURE 4.10 – Image utilisé pour le dataset d'entraînement de l'insecte Goliathus_goliathus.

4.3.2 Implémentation des modèles de classification

L'implémentation des modèles de classification, qu'il s'agisse de ResNet ou d'EfficientNet, repose sur une structure de code similaire. La description qui suit est donc valable pour ces deux modèles mais ne va détailler précisément que le modèle le plus efficace trouvé, ResNet.

La première étape consiste à charger les données. Les images et les annotations (nom scientifique de l'espèce) sont chargées à partir d'un fichier Excel et d'un dossier contenant les images individuelles d'insectes. Les étiquettes des espèces sont encodées via un *LabelEncoder*, puis transformées en format *one-hot encoding* pour une classification multi-classes. Les images sont redimensionnées afin de s'adapter aux exigences du modèle (224x224 pour le modèle le plus optimal). Une analyse des impacts de la résolution sur les performances du modèle est présente dans le chapitre 4 : « Résultats ».

Afin d'améliorer la généralisation du modèle, une méthode d'augmentation d'images est utilisée. Cette méthode est réalisée à l'aide de la classe *ImageDataGenerator*, qui permet de faire des transformations sur le dataset afin d'augmenter celui-ci et donc de potentiellement améliorer la précision du modèle. [24] Ces transformations incluent un paramètre de rotation (où le degré est choisi), une valeur de décalage horizontal ou vertical, une valeur de zoom, une valeur de cisaillement et un choix sur une inversion horizontale. Ces paramètres d'augmentation sont ajustés de manière expérimentale pour maximiser les performances du modèle. Les valeurs retenues sont celles-ci :

```
datagen = ImageDataGenerator(
    rotation_range=40,
    width_shift_range=0.3,
    height_shift_range=0.3,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

FIGURE 4.11 – Exemple de paramétrage du ImageDataGenerator.

Le modèle utilisé, ResNet, est chargé avec des poids pré-entraînés sur ImageNet afin de bénéficier des images apprises sur un large éventail générique. Ces poids sont utilisés comme base pour un entraînement supplémentaire adapté à la classification d'insecte. Cette méthode s'appelle le *fine-tuning*. L'architecture du modèle est adaptée de cette manière :

- L'extraction des caractéristiques : la partie inférieure du modèle, appelée *backbone* et constituée de plusieurs couches convolutives, est conservée pour extraire des caractéristiques visuelles pertinentes à partir des images.
- Les caractéristiques extraites passent par une couche de pooling global (*GlobalAveragePooling2D*) afin de réduire leur dimensionnalité tout en préservant les informations importantes.
- Une couche dense intermédiaire avec 256 neurones et une activation ReLu est ajoutée pour augmenter la capacité d'apprentissage du modèle.
- Une régularisation par *dropout* (0.5) est appliquée pour réduire le risque d'*overfitting*.
- Enfin, une couche dense finale avec une activation *softmax* est utilisée pour produire les probabilités associées à chaque espèce d'insecte. Le nombre de neurones de cette couche correspond au nombre total d'espèces dans le jeu de données.

Le modèle est ensuite compilé avec l'optimiseur Adam, une fonction de perte catégorielle croisée (*categorical_crossentropy*), et une métrique d'évaluation basée sur la précision (*accuracy*).

L'entraînement est effectué sur 20 epoch avec un lot de 32 images. Une technique d'ajustement dynamique du taux d'apprentissage (*ReduceLROnPlateau*) est utilisée afin de réduire le taux d'apprentissage lorsque la perte de validation cesse de diminuer. Ce mécanisme permet d'améliorer la convergence du modèle. Le jeu de données d'entraînement est divisé en deux sous-ensembles : 80% pour l'entraînement et 20% pour la validation.

En parallèle, une méthode de surveillance des ressources a été mise en place pour enregistrer l'utilisation du CPU, de la mémoire vive, ainsi que les performances du modèle après chaque epoch. Ces statistiques sont enregistrées dans un fichier Excel afin de suivre l'évolution de l'entraînement et d'avoir une vision claire quant aux ressources utilisées.

Une fois l'entraînement terminé, le modèle est évalué sur les données de validation. Ce modèle est ensuite sauvegardé au format ".h5" pour permettre une utilisation ultérieure.

Après cette étape, le modèle est donc testé sur un ensemble d'images inédites afin d'évaluer sa capacité à classifier des insectes non vus. Chaque image est pré traitée de la même manière que pendant l'entraînement (redimensionnement à 224x224 pixels). Le modèle génère des prédictions sous forme de probabilités associées à chaque classe.

Ces prédictions sont ensuite converties en étiquettes finales à l'aide du *LabelEncoder*. Pour chaque image testée, le nom de l'image et l'espèce prédite sont enregistrées dans un fichier Excel. Ce fichier sert à analyser les performances du modèle et à comparer ses prédictions avec d'autres modèles.

Image_Name	Predicted_Species
hydrophilidae_drawer_0008_crop_45.jpg	Dactylosternum_abdominale

FIGURE 4.12 – Fichier résultat de la classification.

L'enregistrement des résultats sous cette forme permet de faire des analyses statistiques ou des visualisations très facilement.

Chapitre 5

Résultats

5.1 Résultats de la détection d'insectes

L'évaluation des performances des modèles de détection a été réalisée en analysant plusieurs métriques et divers graphes. Les résultats sont comparés avant et après l'introduction des améliorations (annotations négatives et pseudo-labelling), mettant en lumière les effets de ces stratégies sur les performances.
Ces résultats sont séparés en deux parties, chacune se concentrant sur un ordre (Coleoptera et Lepidoptera).

5.1.1 Coleoptera

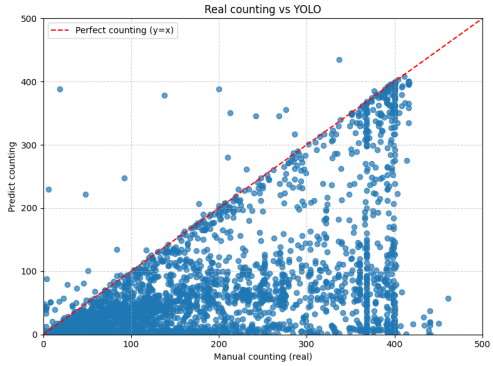
TABLE 5.1 – Résultats des différents modèles de détection pour l'ordre des Coleoptera.

Modèle	**Moyenne**	**Médiane**	**Nombre de prédictions exactes sur 3285**
YOLO	56.68	73.66	27
Faster R-CNN	-11.23	11.39	315
Faster R-CNN (avec annotations négatives)	-13.27	**8.27**	348
Faster R-CNN (avec pseudo-labelling)	-12.22	13.33	284
YOLO (avec pseudo-labelling)	**-2.72**	11.04	**440**

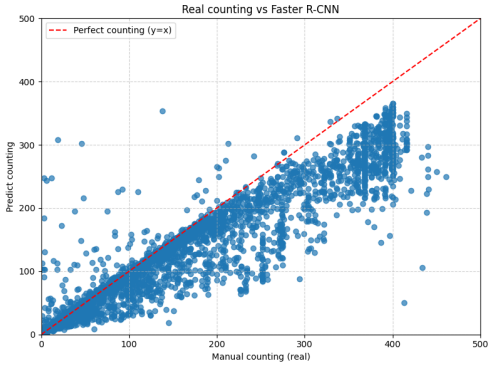
Dans ce tableau (Table 5.1), les moyennes et médianes sont calculées par rapport aux erreurs relatives, c'est-à-dire la différence entre le nombre d'insectes prédit et

le nombre réel, normalisée par le nombre réel d'insectes. Ces métriques permettent d'évaluer la précision des modèles en tenant compte de l'importance relative des erreurs pour chaque photo. Une faible erreur relative indique une bonne concordance entre le comptage prédit et le comptage réel.
Initialement, Faster R-CNN offre de meilleurs de résultats. L'ajout des annotations négatives et la mise en œuvre du pseudo-labelling ont permis des progrès notables. Les annotations négatives ont augmenté le nombre de prédictions parfaites, c'est-à-dire le nombre de fois où le modèle parvient à détecter le nombre exact d'insectes présents dans la photo. Le pseudo-labelling permet une amélioration de la perfomance du modèle de YOLO, en devenant même le modèle le plus efficace.

Visualisation des résultats des deux modèles avant les améliorations :



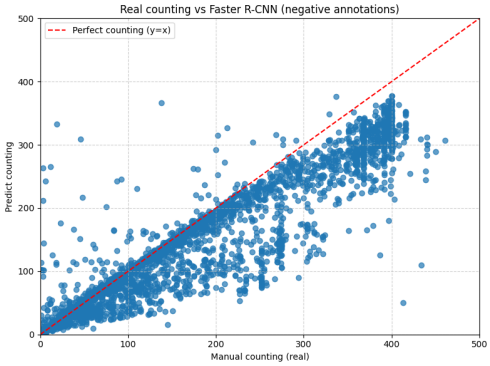
(a) Comparaison entre le comptage manuel et YOLO.



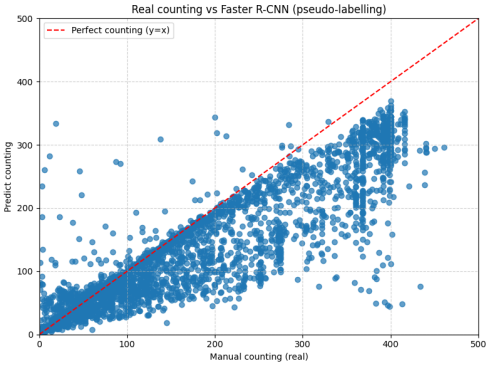
(b) Comparaison entre le comptage manuel et Faster R-CNN.

FIGURE 5.1 – Comparaison entre le comptage manuel et les prédictions des différents modèles pour l'ordre des Coleoptera.

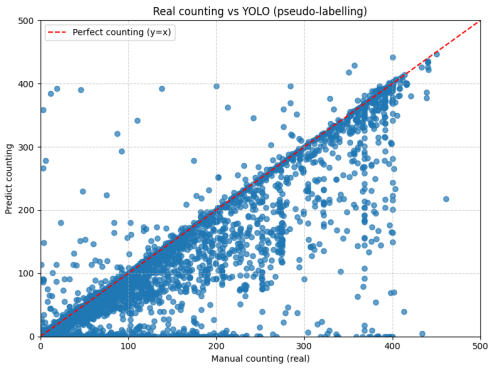
Visualisation des résultats après les améliorations :



(a) Comparaison entre le comptage manuel et Faster R-CNN (avec annotations négatives).



(b) Comparaison entre le comptage manuel et Faster R-CNN (avec pseudo-labelling).



(c) Comparaison entre le comptage manuel et YOLO (avec pseudo-labelling).

FIGURE 5.2 – Comparaison entre le comptage manuel et les prédictions des différents modèles (avec améliorations) pour l'ordre des Coleoptera

Les Figures 5.1 et 5.6 montrent les différences de comptage entre le comptage manuel (comptage correct) et le comptage prédit par les différents modèles. Chaque point représente une image, avec le comptage manuel en abscisse et le comptage prédit en ordonnée. La ligne rouge en pointillés correspond à une correspondance parfaite (y = x). Les écarts par rapport à cette ligne illustrent les erreurs de prédiction.

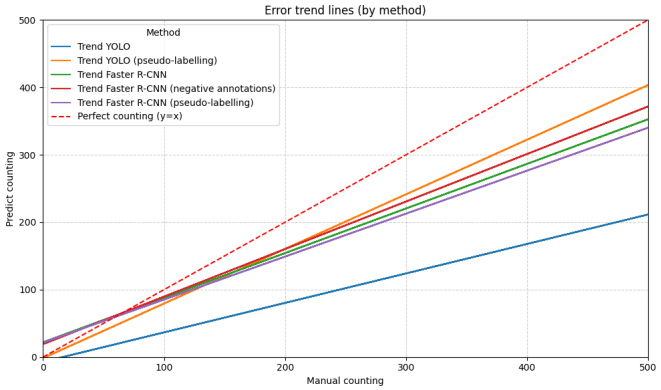
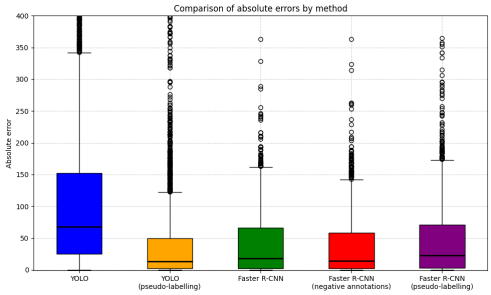
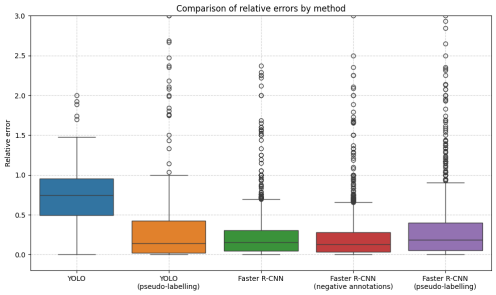


FIGURE 5.3 – Evolution des tendances d'erreur selon les modèles de détection pour l'ordre des Coleoptera.

La Figure 5.3 montre l'évolution des tendances d'erreur pour chaque modèle de détection. Les lignes représentent les tendances de comptage prédit en fonction du comptage manuel. La ligne rouge pointillée indique un comptage parfait (y = x). Les écarts par rapport à cette ligne montrent les biais systématiques de chaque méthode et les gains obtenus après amélioration. Ce graphe permet de voir que plus il y a d'insectes dans une boîte, plus les modèles commettent des erreurs. Il nous permet également de visualiser l'efficacité du modèle YOLO (avec pseudo-labelling) en comparaison avec les autres modèles.



(a) Boxplot des erreurs absolues des différents modèles.



(b) Boxplot des erreurs relatives des différents modèles.

FIGURE 5.4 – Comparaison des erreurs relatives et absolues des différents modèles de détection pour l'ordre des Coleoptera

Ces deux graphiques permettent de voir la dispersion des erreurs pour chaque modèle. Un boxplot étroit indique que les erreurs sont plus concentrées autour de la médiane, tandis qu'un boxplot large indique une plus grande variabilité dans les erreurs. Les modèles avec une médiane plus proche de zéro sont plus précis

dans leur comptage, tandis que les valeurs aberrantes montrent où des erreurs exceptionnelles ont eu lieu.

L'analyse de ces graphes permet de visualiser l'efficacité du modèle YOLO avec pseudo-labelling. Avec ces résultats, il est clair que ce modèle est le plus optimal pour la détection de Coleoptera. Cependant, il peut encore être améliorer pour atteindre une précision plus élevée.

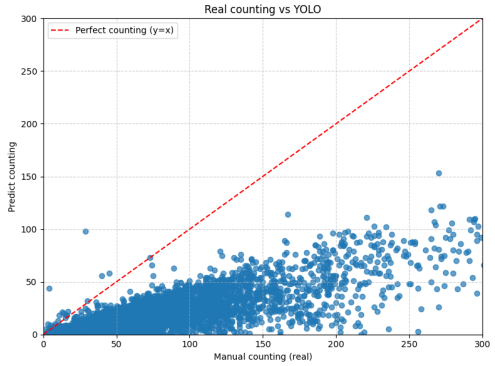
5.1.2 Lepidoptera

TABLE 5.2 – Résultats des différents modèles de détection pour l'ordre des Lepidoptera.

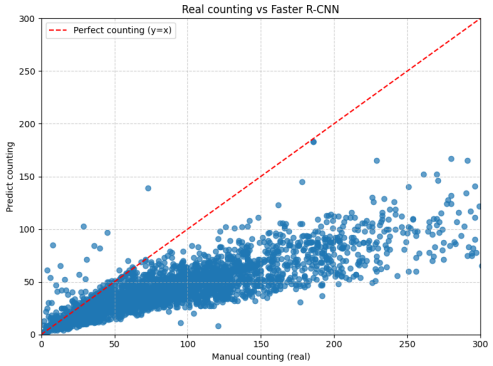
Modèle	**Moyenne**	**Médiane**	**Nombre de prédictions exactes sur 3748**
YOLO	75.62	76.79	4
Faster R-CNN	41.85	48.67	50
Faster R-CNN (avec annotations négatives)	43.68	50.41	27
Faster R-CNN (avec pseudo-labelling)	**40.62**	**47.78**	**61**

Initialement, Faster R-CNN offre de meilleurs de résultats. L'ajout des annotations négatives n'a pas améliorer l'efficacité du modèle. Cependant, la mise en œuvre du pseudo-labelling a permis une légère amélioration.

Visualisation des résultats des deux modèles avant les améliorations :



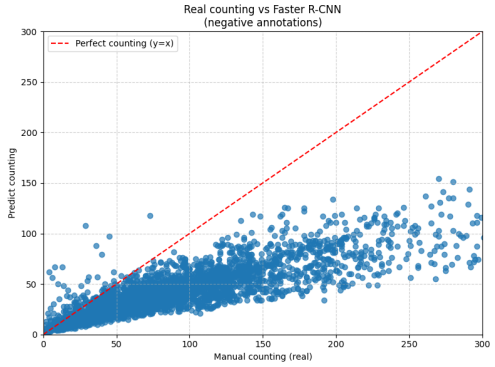
(a) Comparaison entre le comptage manuel et YOLO.



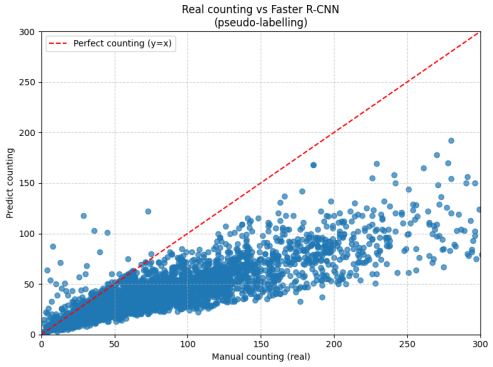
(b) Comparaison entre le comptage manuel et Faster R-CNN.

FIGURE 5.5 – Comparaison entre le comptage manuel et les prédictions des différents modèles pour l'ordre des Lepidoptera.

Visualisation des résultats après les améliorations :



(a) Comparaison entre le comptage manuel et Faster R-CNN (avec annotations négatives).



(b) Comparaison entre le comptage manuel et Faster R-CNN (avec pseudo-labelling).

FIGURE 5.6 – Comparaison entre le comptage manuel et les prédictions des différents modèles (avec améliorations) pour l'ordre des Lepidoptera

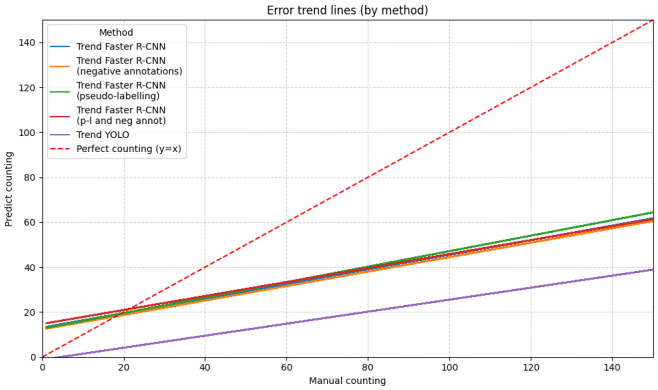
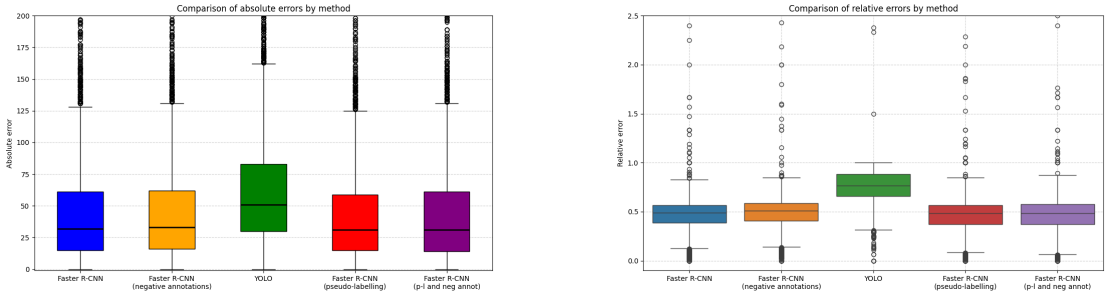


FIGURE 5.7 – Evolution des tendances d'erreur selon les modèles de détection pour l'ordre des Lepidoptera.



(a) Boxplot des erreurs absolues des différents modèles.

(b) Boxplot des erreurs relatives des différents modèles.

FIGURE 5.8 – Comparaison des erreurs relatives et absolues des différents modèles de détection pour l'ordre des Lepidoptera

L'analyse de ces graphes permet de visualiser l'efficacité du modèle Faster R-CNN avec pseudo-labelling. Avec ces résultats, il est clair que ce modèle est le plus optimal pour la détection de Lepidoptera. Il reste cependant bien moins efficace que le modèle de détection des Coleoptera et présente ainsi une plus grande marge d'amélioration.

La difficulté de détection de ces modèles pour l'ordre des Lepidoptera peut être expliqué par le fait que les boîtes entomologiques de cet ordre contiennent beaucoup d'insectes qui se superposent.

FIGURE 5.9 – Exemple de boîte entomologique montrant des insectes qui se superposent.

5.2 Résultats de la classification d'insectes

5.2.1 Comparaison des modèles de classification

La classification des insectes a été évaluée en utilisant deux modèles : ResNet et EfficientNet. Les performances des deux modèles ont été comparées sur plusieurs métriques, notamment la précision, le rappel et les matrices de confusion, afin d'évaluer leur capacité à identifier correctement les espèces d'insectes dans les images.

Les métriques utilisées sont définies comme suit :

— **Précision** :

$$\text{Précision} = \frac{\text{VP}}{\text{VP} + \text{FP}}$$

où VP représente les vrais positifs et FP les faux positifs.

— **Rappel** :

$$\text{Rappel} = \frac{\text{VP}}{\text{VP} + \text{FN}}$$

où FN représente les faux négatifs.

— **F1-Score** :

$$\text{F1} = 2 \times \frac{\text{Précision} \times \text{Rappel}}{\text{Précision} + \text{Rappel}}$$

Voici une comparaison des résultats obtenus pour les deux meilleures versions des modèles ResNet et EfficientNet.

Métriques	*ResNet*	*EfficientNet*
Précision	**0.99**	0.97
Rappel	**1**	1
F1-Score	**0.99**	0.98
Nombre de corrections parfaites (pour un total de 1400 prédictions)	**1388**	1321

FIGURE 5.10 – Tableau récapitulatif des résultats des deux modèles de classification.

Le modèle ResNet s'est donc révélé globalement plus performant qu'EfficientNet, avec une précision globale atteignant 99%. L'analyse détaillée des résultats se concentrera uniquement sur ce modèle.

5.2.2 Analyse détaillée des résultats

Afin d'avoir de la visibilité dans les résultats, une matrice de confusion par ordre est générée. L'analyse de ces résultats est donc séparée en deux parties : Coleoptera et Lepidoptera.

Coleoptera :

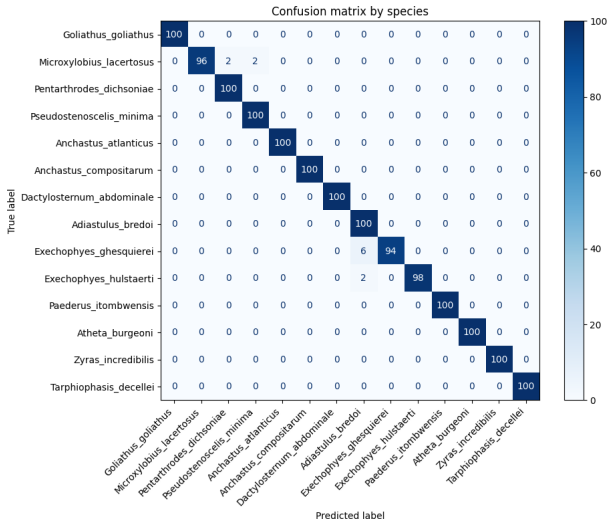


FIGURE 5.11 – Matrice de confusion d'une classification par nom d'espèce pour l'ordre des Coleoptera par le modèle ResNet.

Cette matrice de confusion (Figure 5.11) résulte d'un test sur un jeu de données comportant 100 images de chaque espèce. Celle-ci montre que les erreurs sont principalement observées entre des espèces présentant des différences morphologiques

minimes. Par exemple, le modèle confond parfois les *Microxylobius_lacertotus* et les *Pseudostenoscelis_minima*.

(a) *Microxylobius_lacertotus.*

(b) *Pseudostenoscelis_minima.*

FIGURE 5.12 – Illustration d'une erreur courante du modèle de détection pour l'ordre des Coleoptera

Lepidoptera :

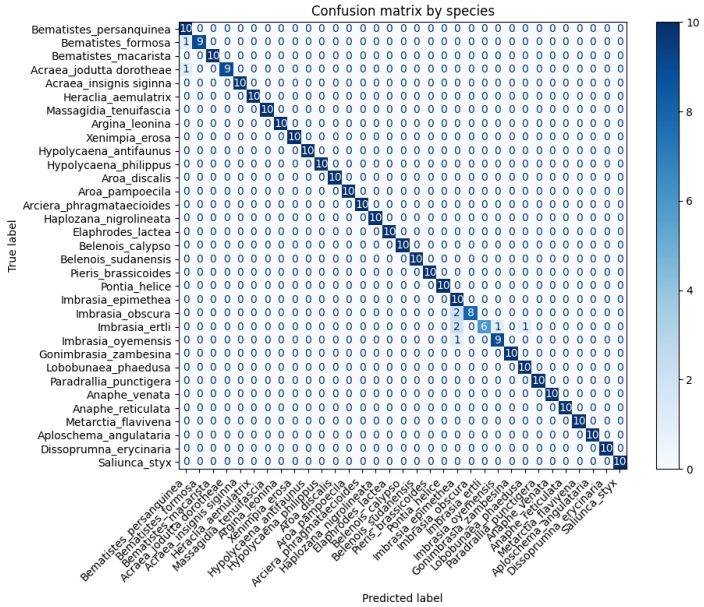
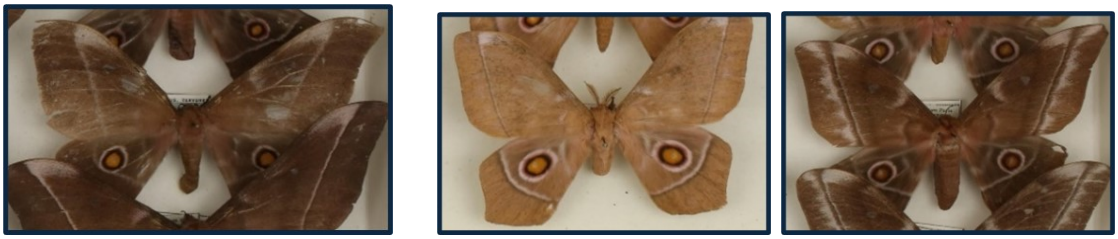


FIGURE 5.13 – Matrice de confusion d'une classification par nom d'espèce pour l'ordre des Lepidoptera par le modèle ResNet.

Cette matrice de confusion (Figure 5.13) est le résultat d'un test sur un jeu de données de 10 images par espèces. Les erreurs les plus fréquentes de ce modèle sont entre ces trois espèces d'Imbrasia.



(a) *Imbrasia_epimethea.* (b) *Imbrasia_ertli.* (c) *Imbrasia_obscura.*

FIGURE 5.14 – Illustration d'une erreur courante du modèle de détection pour l'ordre des Lepidoptera

Les différences entre ces espèces sont minimes, ce qui peut expliquer les erreurs du modèle.

Ces résultats ci-dessus correspondent à une classification précise par nom scientifique de l'espèce. Si on remonte de quelques niveaux taxonomiques, la précision du modèle augmente jusqu'à atteindre une précision de 100

TABLE 5.3 – Précision du modèle de classification en fonction du niveau taxonomique.

Critères de classifications	**Coleoptera**	**Lepidoptera**
Nom scientifique	99%	97%
Genre	99%	
Famille	100%	
Ordre	100%	

5.2.3 Facteurs influant sur la précision des modèles

Afin d'analyser l'impact de la résolution, différents modèles ont été entraînés avec différentes valeurs de résolution d'images.

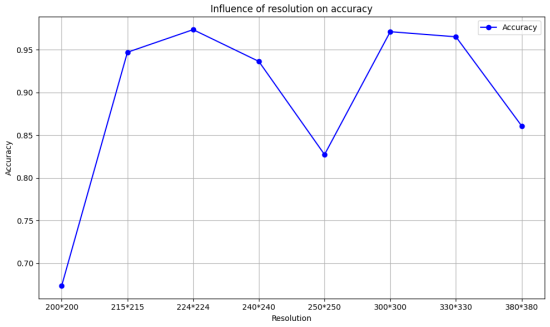


FIGURE 5.15 – Graphique montrant l'influence de la résolution sur la précision.

Ce graphique (Figure 5.15) démontre qu'un maxima est trouvé lorsque la résolution des images est de 224*224 pixels.

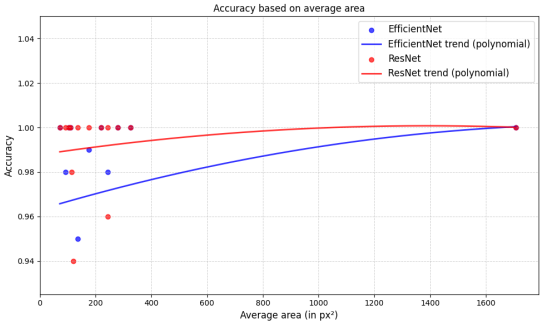


FIGURE 5.16 – Graphique montrant l'influence de la taille des insectes sur la précision des modèles.

Ce graphique (Figure 5.16) révèle que la taille des insectes a un impact significatif sur la précision. Plus ceux-ci sont grands, plus le modèle sera efficace. Cette observation met en évidence, encore une fois, l'importance de la résolution des images d'entrée.

Chapitre 6

Discussion

6.1 Perspectives d'amélioration

6.1.1 Détection

L'étape de détection présente certaines limites, particulièrement dans les cas où les insectes se chevauchent ou sont regroupés de manière dense. Ces situations compliquent l'identification des contours individuels et peuvent entraîner des erreurs, telles que des détections multiples pour un même insecte ou l'omission d'insectes partiellement visibles. Un enrichissement du jeu de données pourrait permettre une meilleure efficacité des modèles. Intégrer davantage d'images contenant des insectes en situations complexes, comme les regroupements denses ou les superpositions, favoriserait un meilleur entraînement du modèle.

De plus, l'analyse de l'efficacité des modèles de détection se fait en comparant les résultats avec les comptages manuels. Cependant, certaines erreurs sont présentes dans ces comptages et faussent ainsi les résultats. Une analyse approfondie des erreurs présentes dans ces comptages manuels utilisés comme référence minimiserait également l'impact sur les résultats.

6.1.2 Classification

Les modèles de classification montrent des limites face aux espèces morphologiquement très proches ou aux insectes de petite taille. Ces défis peuvent être attribués à la difficulté de discerner des détails subtils dans des images à basse résolution. Une approche qui augmenterait la résolution des images permettrait au modèle de capter des détails morphologiques plus fins

6.1.3 Gestion des ressources

L'entraînement de ces modèles est très lourd en ressources computationnelles, nécessitant l'utilisation de superordinateurs tels que LUCIA. Cela constitue une contrainte importante, notamment pour des projets à échelle limitée ou pour des chercheurs disposant de moyens réduits. Pour atténuer ces limitations, une optimisation des modèles permettrait de diminuer ces besoins en calcul. L'utilisation du cloud computing, comme AWS ou Google Cloud, permettrait également de bénéficier d'une puissance de calcul accessible à moindre coût. [25]

6.2 Contribution et bénéfices du projet

Ce projet apporte des contributions significatives pour le domaine de la recherche en entomologie. L'un des apports majeurs de ce travail réside dans l'implémentation d'une approche intégrant des modèles d'intelligence artificielle, notamment pour la détection et la classification des insectes. Cette méthodologie novatrice met en lumière l'adaptation de techniques avancées aux besoins spécifiques de l'analyse entomologique. L'automatisation des processus facilite le traitement à grande échelle des images, réduisant ainsi le temps et les ressources nécessaires. En outre, ce projet a permis d'identifier des limites et des axes d'amélioration des approches actuelles, ouvrant la voie à des avancées futures dans le domaine.

Les retombées pour l'entomologie sont multiples. L'automatisation des tâches de détection et de classification offre une alternative efficace aux méthodes manuelles, en réduisant considérablement le temps nécessaire pour analyser de grandes quantités d'images. De plus, les modèles développés garantissent une identification systématique et standardisée des insectes, limitant les biais humains et augmentant la précision des résultats. Ces avancées ouvrent la porte à des études à grande échelle qui, jusqu'ici, auraient été trop complexes ou coûteuses à réaliser.

Ce projet laisse donc entrevoir la possibilité de développer de nouvelles applications dans des domaines connexes. Il est évidemment possible de pousser ces modèles plus loin. On pourrait imaginer des applications où il suffirait de prendre une photo d'un insecte et de recevoir instantanément le nom de l'espèce. Ou des applications permettant de faire un suivi d'espèces nuisibles, de pollinisateurs, etc. Ce projet offre ainsi des perspectives prometteuses pour le développement d'applications variées.

6.3 Limites et difficultés rencontrées

Malgré les résultats prometteurs, certaines limites subsistent. L'un des principaux défis a été la taille limitée du jeu de données d'entraînement, qui a restreint la généralisation des modèles. Les annotations, bien qu'essentielles, nécessitent un effort manuel considérable, ce qui peut ralentir le processus de développement.

Enfin, les ressources computationnelles nécessaires pour l'entraînement des modèles ont également représenté un défi, rendant certaines expérimentations coûteuses en temps et en énergie.

Chapitre 7

Conclusion

Ce projet a permis de développer et d'appliquer des méthodes basées sur l'intelligence artificielle pour améliorer la détection et la classification des insectes à partir d'images. En combinant des modèles de *deep learning*, ce travail a démontré la pertinence de ces approches pour l'entomologie moderne. L'intégration de ces outils dans des processus automatisés représente un progrès majeur, permettant de gagner en efficacité tout en augmentant la précision des résultats par rapport aux méthodes traditionnelles.

Les défis rencontrés, notamment la gestion des images complexes et les limitations dues à la petite taille des insectes, soulignent la nécessité d'améliorer continuellement les modèles, en enrichissant les jeux de données ou en augmentant la résolution des images. Toutefois, ces difficultés n'ont pas empêché l'obtention de résultats significatifs, permettant ainsi de valider l'application des technologies de détection et de classification dans le domaine de l'entomologie.

Les avantages de ce projet incluent une réduction des coûts et du temps nécessaires, ainsi qu'une précision accrue dans le comptage des insectes. Il existe également des perspectives de généralisation de l'approche développée à d'autres contextes scientifiques et pratiques, tels que le suivi de la biodiversité. Ce projet ouvre en outre des voies de recherche pour affiner les techniques existantes et intégrer de nouvelles approches capables de résoudre des cas complexes.

Ainsi, ce travail démontre le potentiel de l'intelligence artificielle pour révolutionner l'étude des insectes et contribuer à des applications concrètes dans le domaine de l'écologie et de la conservation. La recherche entomologique voit s'ouvrir de nouvelles voies pour le suivi des populations et la compréhension des dynamiques écologiques. Dans un monde où la biodiversité est en constante évolution, l'utilisation de tels outils pourrait jouer un rôle clé dans la préservation des espèces.

Bibliographie

[1] T. Keldenich. Couche de convolution : Tout savoir sur les cnn - meilleur guide. Inside Machine Learning, December 2023.

[2] Contributeurs aux projets Wikipedia. Réseau neuronal convolutif. `https://fr.wikipedia.org/`, November 2024.

[3] Yolov7 : L'intelligence artificielle pour la détection d'objets en temps réel dans une image. Novelis innovation, `https://novelis.io/`.

[4] Faster r-cnn explained for object detection tasks. DigitalOcean, `https://www.digitalocean.com`.

[5] H. P. Aberlenc, V. Albouy, D. Barthélémy, J.-C. Beaucornu, P. Blandin, N. Cliquennois, R. Constantin, A. Cruaud, L. Derharveng, G. Delvare, C. D'Haese, J. DeMarmels, L. Desutter, E. Dominguez, D. A, J. Elouard, S. Fenoglio, G. Fleck, R. Fochetti, and J. Vayssières. *Les insectes du monde : Biodiversité - Classification - Clés de détermination des familles -Tome 1.* Quae & Museo Editions, 2021.

[6] F. Sánchez-Bayo and K. A. G. Wyckhuys. Worldwide decline of the entomofauna : A review of its drivers. *Biological Conservation*, 232 :8–27, 2019.

[7] J. Williams. L'importance des insectes. Learn.Tearfund, 2022. Consulté le 15 octobre 2024, à l'adresse `https://learn.tearfund.org`.

[8] Le déclin des insectes met en péril le vivant. Muséum National D'Histoire Naturelle. Consulté le 15 octobre 2024, à l'adresse `https://www.mnhn.fr`.

[9] A. Athuraliya and Creately. Dichotomous key : comprehensive guide with editable examples. Creately, October 2023.

[10] Contributeurs aux projets Wikimedia. Anatomie des insectes, July 2024. Consulté le 7 juillet 2024.

[11] J. Mignon, E. Haubruge, and F. Francis. *Clé d'identification des principales familles d'insectes d'Europe.* Presses Agronomiques de Gembloux, 2016.

[12] A. Berkani. Étude morphométrique des stades préimaginaux de phyllocnistis citrella stainton (lepidoptera, gracillariidae) en algérie. *Fruits*, 58(2) :83–88, 2003.

[13] M. Leonard-Salle. Variation génétique et aires de répartition des insectes - actualités de l'ulb. Actualités de L'ULB, n.d.

[14] IBM. Vision par ordinateur. https://www.ibm.com/fr-fr/topics, October 2024. Consulté le 3 novembre 2024, à l'adresse https://www.ibm.com/.

[15] OpenClassrooms. Découvrez les différentes couches d'un cnn. OpenClassrooms, n.d.

[16] Melanie. You only look once (yolo) : What is it ? Data Science Courses | DataScientest, https://datascientest.com, March 2024.

[17] R. Kassel. R-cnn (region based convolutional network) : tout sur ce modèle de machine learning. Formation Data Science | DataScientest.com, https://datascientest.com/, March 2024.

[18] GeeksforGeeks. Residual networks (resnet) deep learning. https://www.geeksforgeeks.org/, January 2023.

[19] P. Potrimba. What is efficientnet ? the ultimate guide. Roboflow Blog, https://blog.roboflow.com/what-is-efficientnet/, April 2024.

[20] A. Sarkar. Understanding efficientnet —the most powerful cnn architecture. Medium, https://arjun-sarkar786.medium.com/, December 2023.

[21] A. Rafay and W. Hussain. Efficientskindis : An efficientnet-based classification model for a large manually curated dataset of 31 skin diseases. *Biomedical Signal Processing And Control*, 85 :104869, 2023.

[22] Ceci. https://www.ceci-hpc.be/faq.html#2.4.

[23] P. Cascante-Bonilla, F. Tan, Y. Qi, and V. Ordonez. Curriculum labeling : Revisiting pseudo-labeling for semi-supervised learning. In *Proceedings Of The AAAI Conference On Artificial Intelligence*, volume 35, pages 6912–6920, 2021.

[24] B. F. Chollet. Building powerful image classification models using very little data, n.d.

[25] Qu'est-ce que le cloud computing ? Azure Microsoft. Consulté le 12 novembre 2024, à l'adresse https://azure.microsoft.com.

Annexes

A Code

Tout le code utilisé lors de ce projet peut être retrouvé sur ce dépot GitHub : `https://github.com/gregnoel/Entomo/tree/main`

B Espèces de Coleoptera utilisées lors de la phase de classification

Order	Family	Genus	Species	Scientific_name
Coleoptera	Staphylinidae	Paederus	itombwensis	Paederus_itombwensis
		Atheta	burgeoni	Atheta_burgeoni
		Zyras	incredibilis	Zyras_incredibilis
	Pselaphidae	Adiastulus	bredoi	Adiastulus_bredoi
		Exechophyes	ghesquierei	Exechophyes_ghesquierei
			hulstaerti	Exechophyes_hulstaerti
	Curculionidae	Pseudostenoscelis	minima	Pseudostenoscelis_minima
		Pentarthrodes	dichsoniae	Pentarthrodes_dichsoniae
		Microxylobius	lacertosus	Microxylobius_lacertosus
	Hydrophilidae	Dactylosternum	abdominale	Dactylosternum_abdominale
	Elateridae	Anchastus	atlanticus	Anchastus_atlanticus
			compositarum	Anchastus_compositarum
	Tenebrionidae	Tarphiophasis	decellei	Tarphiophasis_decellei
	Scarabaeidae	Goliathus	goliathus	Goliathus_goliathus

C Espèces de Lepidoptera utilisées lors de la phase de classification

Order	Family	Genus	Species	Scientific_name
Lepidoptera	Acraeidae	Bematistes	persanquinea	Bematistes_persanquinea
			formosa	Bematistes_formosa
			macarista	Bematistes_macarista
		Acraea	jodutta dorotheae	Acraea_jodutta dorotheae
		Acraea	insignis siginna	Acraea_insignis siginna
	Agaristidae	Heraclia	aemulatrix	Heraclia_aemulatrix
		Massagidia	tenuifascia	Massagidia_tenuifascia
	Zygaenidae	Saliunca	styx	Saliunca_styx
	Uraniidae	Aploschema	angulataria	Aploschema_angulataria
		Dissoprumna	erycinaria	Dissoprumna_erycinaria
	Thyretidae	Metarctia	flavivena	Metarctia_flavivena
	Thaumetopoeidae	Paradrallia	punctigera	Paradrallia_punctigera
		Anaphe	venata	Anaphe_venata
		Anaphe	reticulata	Anaphe_reticulata
	Geometridae	Xenimpia	erosa	Xenimpia_erosa
	Arctiidae	Argina	leonina	Argina_leonina
	Saturniidae	Lobobunaea	phaedusa	Lobobunaea_phaedusa
		Gonimbrasia	zambesina	Gonimbrasia_zambesina
		Imbrasia	oyemensis	Imbrasia_oyemensis
			ertli	Imbrasia_ertli
			obscura	Imbrasia_obscura
			epimethea	Imbrasia_epimethea
	Pieridae	Pieris	brassicoides	Pieris_brassicoides
		Pontia	helice	Pontia_helice
		Belenois	calypso	Belenois_calypso
			sudanensis	Belenois_sudanensis
	Notodontidae	Elaphrodes	lactea	Elaphrodes_lactea
		Haplozana	nigrolineata	Haplozana_nigrolineata
		Arciera	phragmataecioides	Arciera_phragmataecioides
	Lycaenidae	Hypolycaena	antifaunus	Hypolycaena_antifaunus
			philippus	Hypolycaena_philippus
	Lymantriidae	Aroa	discalis	Aroa_discalis
			pampoecila	Aroa_pampoecila

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**