UCLouvain

epl

École polytechnique de Louvain

# Analyse de boites entomologiques

## Détection et classification d'insectes

Auteur: **Gautier Black**
Promoteurs: **Axel Legay, Grégoire Noël**
Lecteurs: **Sébastien Jodogne, Benoît Duhoux**
Année académique 2023–2024
Master [120] : ingénieur civil en informatique

**Remerciements**

J'aimerais tout d'abord remercier mon promoteur, le Pr. Axel Legay, de m'avoir donné l'opportunité de travailler sur ce sujet tout en m'ayant laissé une grande liberté dans la réalisation de ce mémoire.

Je remercie ensuite tout particulièrement le Dr. Grégoire Noël, mon référent au laboratoire entomologique de Gembloux Agro-Bio Tech (GxABT). Merci à lui pour sa disponibilité, son enthousiasme pour le sujet ainsi que son implication dans mon intégration au sein de l'équipe.

Je remercie bien sûr le Pr. Frédéric Francis de m'avoir accueilli au sein du laboratoire entomologique de GxABT dont il est le directeur.

J'en profite pour remercier les membres du laboratoire avec lesquels j'ai développé des liens amicaux : Hugo, Colleen, Ricardo, Gloria, Chloé, Émilie, Sandra, Arnaud, Marcellin, Lallie, Rudy, Ottavia, Ibtissem, Nicolas, Kouanda, Kenza, Mathilde, Françoise et Joachim. Merci à eux pour leur gentillesse.

Je tiens également à remercier le Dr. Didier Van den Spiegel pour son accueil à l'AfricaMuseum. Merci à lui d'avoir pris le temps de me montrer le procédé de numérisation mis en œuvre sur place et de m'avoir donné accès à des ressources qui ont été grandement utiles à la réalisation de ce travail.

Merci aussi au Dr. Patrick Semal de l'Institut royal des Sciences naturelles de Belgique (IRSNB) pour ses précieux conseils lors de notre entretien en visioconférence.

Pour finir, je remercie ma maman, ma bien-aimée ainsi que mes colocataires pour le soutien durant les périodes de travail ardu et les moments de doute.

**Résumé**

Les données entomologiques obtenues par le recensement des insectes sont très utiles pour quantifier et interpréter les causes de la diminution de leur effectif. En effet, les insectes épinglés dans les boites entomologiques constituent une trace de la biodiversité qui a existé à une époque et à un lieu définis. Ces données, essentielles pour la compréhension de la dynamique des populations d'insectes, sont cependant difficiles d'accès du fait de leur nature.

La numérisation des boites entomologiques permet à la fois de les rendre consultables partout dans le monde, mais aussi de figer l'état de la boite dans le temps en évitant une perte de données due à diverses dégradations. De plus, l'analyse de ces données numériques permet une meilleure connaissance de la collection examinée, notamment en déterminant son nombre d'individus.

Les avancées majeures apportées par les réseaux de neurones convolutionnels (CNN) ont largement propulsé le domaine de la vision par ordinateur. Cependant, l'utilisation des CNN requiert un nombre conséquent de données d'entrainement, et ces données annotées sont difficilement accessibles pour le cas spécifique des insectes dans une boîte entomologique. En cause, la diversité exceptionnelle des groupes taxonomiques.

Face au manque de données annotées, la méthode de détection par traitement d'image présentée dans ce mémoire constitue une base simple pour pouvoir accomplir la tâche de comptabilisation d'insectes sans données d'entrainement. Non seulement, elle peut être utilisée de manière indépendante, mais elle peut aussi contribuer à l'entrainement d'un modèle pré-entrainé YOLO (You Only Look Once), utilisant un réseau de neurones convolutifs (CNN), en tirant profit des données annotées générées lors de son exécution. Cette combinaison des deux méthodes permet par conséquent de bénéficier de la performance des CNN pour la détection d'insectes, en ayant un jeu de données suffisant.

Une fois les insectes détectés, ils peuvent être isolés dans des images individuelles et annotés en fonction de leur groupe taxonomique pour constituer les données d'entrainement de classificateurs. L'analyse des performances de deux classificateurs différents révèle qu'une distinction entre les insectes jusqu'au genre peut être faite avec une exactitude satisfaisante et que le classificateur utilisant un CNN surpasse légèrement celui basé sur une machine à support de vecteurs (SVM).

# Table des matières

# Liste des abréviations

**CNN** Convolutional Neural Networks (réseau de neurones convolutifs)

**COCO** Common Objects in Context (objets courants dans un contexte)

**CSV** Comma-Separated Values (valeurs séparées par virgules)

**ECC** Enhanced Correlation Coefficient (coefficient de corrélation amélioré)

**EF** Electronic Focus (mise au point électronique)

**GPU** Graphical Processing Unit (unité de traitement graphique)

**GxABT** Gembloux Agro-Bio Tech

**HD** Hight-Definition (haute définition)

**HOG** Histogram of Oriented Gradients (histogramme des gradients orientés)

**HSV** Hue Saturation Value (Teinte Saturation Valeur)

**IRSNB** Institut royal des sciences naturelles de Belgique

**LED** Light-Emitting Diode (diode électroluminescente)

**mAP** mean Average Precision (précision moyenne pondérée)

**OvA** One-vs-All (un contre tous)

**OvO** One-vs-one (un contre un)

**R-CNN** Region Based Convolutional Neural Networks (réseaux de neurones convolutifs basés sur des régions)

**RGB** Red-Green-Blue (Rouge-Vert-Bleu)

**SGD** Stochastic Gradient Descent (descente de gradient stochastique)

**SIFT** Scale-Invariant Feature Transform (transformation invariante à l'échelle des caractéristiques)

**STM** Stepper Motor (moteur pas à pas)

**SURF** Speeded Up Robust Features (caractéristiques robustes accélérées )

**SVC** Support Vector Classification (classification par vecteurs de support)

**SVM** Support Vector Machine (machines à vecteurs de support)

**YOLO** You Only Look Once (on ne regarde qu'une fois)

# Table des figures

# Chapitre 1

# Introduction

## 1.1 Contexte

La biodiversité terrestre est d'une variété impressionnante d'espèces, chacune jouant un rôle crucial dans l'équilibre de notre écosystème. Parmi ces innombrables êtres vivants, les insectes se distinguent par leur abondance, leur diversité et leur rôle fondamental dans le maintien de l'harmonie écologique comme la dégradation de la matière organique et la pollinisation [1] [2] [3]. Cependant, face aux défis contemporains tels que le changement climatique et la dégradation de l'environnement, il devient impératif de documenter et de comprendre la dynamique de ces populations d'insectes [2].

C'est dans ce contexte que la numérisation des boites entomologiques émerge comme une révolution pour l'étude des insectes. Les boites entomologiques, précieuses collections de spécimens soigneusement préservés, ont longtemps servi de banques de données statiques, offrant un aperçu de la diversité entomologique à un moment et à un endroit précis [2]. Cependant, à mesure que les technologies numériques progressent, la numérisation de ces collections devient cruciale pour maximiser leur potentiel informatif et pour faciliter la recherche en entomologie [3].

La numérisation des boites entomologiques outrepasse les limitations traditionnelles de la recherche en entomologie. Elle offre un accès mondial des spécimens, éliminant ainsi les barrières géographiques. Alors que les boites entomologiques physiques sont cantonnées à un emplacement spécifique, leur version numérique permet un partage instantané des données, en les rendant par exemple disponibles sur le web, ce qui favorise la recherche collaborative à l'échelle mondiale [2] [4].

De plus, la numérisation contribue à la préservation à long terme des spécimens, minimisant les risques de détérioration physique et de perte dus à des événements tels que des catastrophes naturelles ou des accidents. En effet, les boites se dégradent avec le temps : le fond jauni, les insectes peuvent se casser, moisir ou subir des contaminations. Le lieu de stockage peut quant à lui endurer des dommages tels que de l'humidité, une infiltration d'eau ou un incendie [4] [5]. Les avancées en matière de stockage numérique et de gestion des données offrent une alternative durable aux collections physiques, garantissant que ces archives précieuses restent accessibles aux générations futures.

Enfin, une fois les données sous forme numérique, celles-ci peuvent être analysées en étant par exemple soumises à des techniques avancées de détection et de classification. L'intégration de ces outils au sein d'une collection offre une compréhension approfondie de son état actuel [2]. Cette approche permet notamment de quantifier de manière précise le nombre d'insectes composant la collection. De plus, la possibilité de trier les spécimens par famille, genre ou espèce, offre aux chercheurs la possibilité d'effectuer des analyses ciblées pour étudier les variations taxonomiques au sein de la collection. Cette démarche permet un examen approfondi de la biodiversité présente dans les boites entomologiques numérisées, offrant des données riches et exploitables pour la recherche scientifique.

## 1.2 Procédé de numérisation

Cette section vise à décrire le procédé de numérisation adopté par le laboratoire entomologique de Gembloux Agro-Bio Tech (GxABT). Il expose également les forces et les faiblesses de ce procédé par rapport à un système de numérisation professionnel tel qu'utilisé à l'Africamuseum et à l'Institut royal des sciences naturelles de Belgique (IRSNB).

Le conservatoire entomologique de Gembloux est le plus grand espace de conservation d'insectes en Wallonie. Il contient plus de 1600 types et potentiellement 3 millions de spécimens [6]. Récemment, pas moins de 9000 boites entomologiques ont été numérisées, incluant une grande diversité à travers les différentes familles, genres et espèces d'insectes.

### 1.2.1 Acquisition des images

Pour la numérisation de ses boites entomologiques, le laboratoire entomologique de GxABT a choisi une installation comprenant une webcam Logitech C920 HD Pro d'une définition de 1920 x 1080 pixels (prix : 109,00€). Celle-ci est montée sur une structure composée de deux bras robotisés permettant à un Arduino de déplacer la webcam verticalement et horizontalement (Fig. 1.1). Ces déplacements assurent la prise d'images sur l'entièreté de la surface de la boite. Un script python permet de numériser la boite en fonction de sa taille, en adaptant le nombre d'images prises. Le pas, c'est-à-dire l'intervalle de distance entre deux images, est quant à lui constant. Il est de 2 cm en Y et de 3 cm en X. Il est important de préciser que certaines zones des images se chevauchent entre elles (Fig. 1.2).

En plus d'une liberté de mouvement en deux dimensions, la webcam dispose d'une mise au point ajustable. Plusieurs images de la même région de la boite sont prises à 14 longueurs focales différentes (de 70 à 135 mm avec un pas de 5 mm) (Fig. 1.3).

L'ensemble est éclairé par deux panneaux LED ainsi que deux bandes LED fixées à la webcam. Les photos sont sauvegardées en format bit map. Ce format contient beaucoup d'informations, ce qui rend le fichier assez lourd, environ 6 Mo. Une boite contient au minimum 36 régions, chacune prise à 14 différentes longueurs focales. Ceci représente plus de 500 fichiers par boite, soit environ 2,6 Go de données.

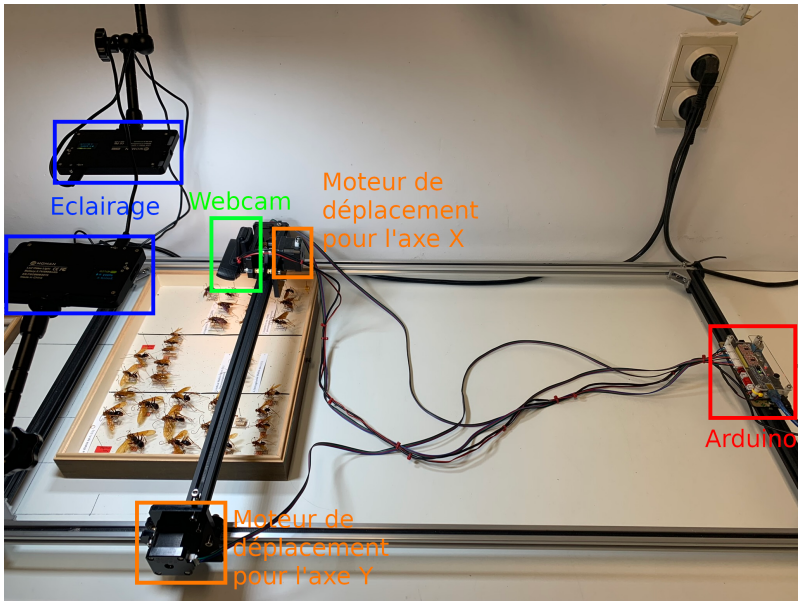



FIGURE 1.1 – Procédé de numérisation du laboratoire entomologique de GxABT

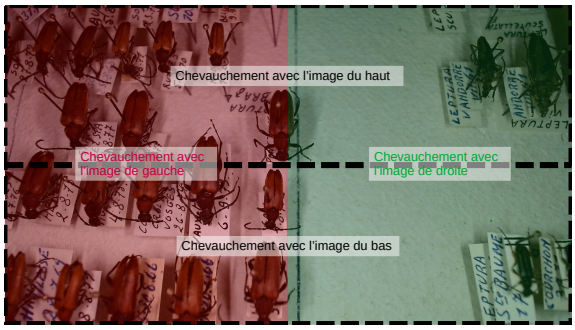



FIGURE 1.2 – Chevauchement sur l'image prise par la webcam

(a) Longueur focale de 70mm

(b) Longueur focale de 135 mm

FIGURE 1.3 – Deux images prises avec des longueurs focales différentes

### 1.2.2 Traitement des images

Les 14 images d'une même région de la boite, mais prises avec des longueurs focales différentes, sont ensuite stackées (Fig. 1.4). Le stacking est une technique avancée largement utilisée dans le domaine du traitement d'images pour améliorer la netteté et la qualité globale d'une photographie. L'idée fondamentale est de fusionner plusieurs images prises au même endroit de sorte que les parties nettes de chaque photo contribuent à la netteté globale de l'image qui en résulte [7].

Un script python qui utilise la librairie OpenCV aligne tout d'abord les différentes images, grâce à l'exécution de l'algorithme Enhanced Correlation Coefficient (ECC) qui maximise le coefficient de corrélation entre deux images ayant des variations. Ensuite, il sélectionne les parties les plus nettes de chaque prise de vue en utilisant l'opérateur Laplacien. Ces zones nettes sont ensuite extraites et combinées pour former une image finale d'une netteté uniforme. Le stacking permet ainsi de surmonter une profondeur de champ étroite, en offrant une netteté accrue sur l'ensemble de la scène photographiée.

FIGURE 1.4 – Résultat de l'image stackée

### 1.2.3 Comparaison avec une configuration professionnelle

Le procédé de numérisation réalisé à l'Africamuseum et à l'Institut royal des sciences naturelles de Belgique (IRSMB) est différent : un objectif à focale fixe Canon EF 40mm f/2.8 STM (prix : 249,00€) est monté sur un boîtier full frame Canon 5DSR (prix : 3019,99€). L'appareil photo est placé sur le dessus d'une armoire blanche dans laquelle un trou permet de prendre en photo une boite entomologique placée sur une étagère (Fig. 1.5). Deux panneaux LED permettent une diffusion uniforme la lumière dans l'armoire. L'objectif étant à focale fixe, il n'est pas possible de zoomer. La solution pour avoir une image aux bonnes dimensions consiste à ajuster la hauteur de l'étagère. Une image de la boite complète est prise une fois la longueur focale ajustée. Il n'est ici pas nécessaire de réaliser un stacking, étant donné la distance suffisante entre la boite et l'appareil. Le stacking est en effet utile lors d'une faible profondeur de champ. L'image capturée a une définition de 8688x5792 pixels (Fig. 1.5).

(a) Configuration

(b) Résultat de la numérisation

FIGURE 1.5 – Procédé de numérisation de l'Africamuseum

L'avantage principal de ce procédé par rapport à celui de GxABT est que l'entièreté de la boite est photographiée en une seule opération. D'une part, ceci augmente considérablement la vitesse de numérisation. D'autre part, il n'y a pas de post-traitement nécessaire pour reconstituer de la boite. La qualité de la photo étant assez satisfaisante pour observer un spécimen spécifique en zoomant dessus, il n'est en effet pas nécessaire de prendre des parties rapprochées de la boite.

La méthode adoptée par Gembloux permet cependant d'acquérir des images de qualité acceptable pour une installation à peu près 30 fois moins coûteuse. Un post-traitement des images pour reconstituer la boite sera néanmoins utile pour prévisualiser la boite dans son ensemble et pour effectuer un comptage manuel des spécimens.

## 1.3 Objectifs

L'objectif de ce travail est de développer des outils fiables de détection et de classification d'insectes, utilisables sur les boites numérisées par le laboratoire entomologique de GxABT. Ces outils permettront un comptage du nombre de spécimens et une classification de ceux-ci à un certain niveau de hiérarchie taxonomique. La précision des outils sera ensuite analysée pour déterminer s'il est envisageable de les adopter. Les données de l'Africamuseum, obtenues par un procédé de numérisation différent, seront également utilisées afin de comparer les deux procédés au niveau de la précision de détection et de classification. Ce travail vise ainsi à contribuer à l'évolution de l'entomologie contemporaine, grâce à une meilleure gestion muséale, et à une meilleure connaissance de la biodiversité, essentielle pour la préservation de l'équilibre de notre planète.

## 1.4 Structure

Ce travail comporte cinq chapitres. Le chapitre 1 introduit le sujet, le chapitre 2 expose différentes méthodes de détection, dans le but d'un comptage du nombre d'insectes dans une boite :

— une méthode basée sur le traitement de l'image
— une méthode utilisant des classificateurs en cascade
— une méthode adoptant la personnalisation d'un modèle pré-entrainé

Le chapitre 3 aborde la classification des insectes et énonce les principes de la taxonomie. Deux méthodes de classification y sont décrites :

— une méthode comprenant un réseau de neurones convolutifs (CNN)
— une méthode utilisant des séparateurs à vaste marge (SVM)

Le chapitre 4 présente les performances des méthodes de détection et le chapitre 5 celles des méthodes de classifications.

# Chapitre 2

# Méthodologie : Détection

De nombreuses techniques de détection d'objets à partir d'une image existent. Ce chapitre se focalise sur trois d'entre elles, qui peuvent s'appliquer à la détection d'insectes dans une boite entomologique. Chacune des trois techniques présente une approche fondamentalement différente des deux autres. La première méthode repose sur l'utilisation d'un masque, exploitant habilement les seuils de valeurs de teinte, de saturation et de luminosité pour isoler les insectes. La deuxième se base sur des classificateurs en cascade, utilisant une série de filtres de caractéristiques. Enfin, la troisième propose une personnalisation d'un modèle pré-entrainé utilisant un réseau de neurones profonds. La détection d'objets permet d'extraire des informations telles que le nombre de détections ainsi que la taille moyenne de ces détections.

## 2.1 Détection par traitement de l'image

La détection des contours des insectes dans une boîte entomologique peut être réalisée en utilisant la segmentation d'image. OpenCV est une bibliothèque puissante et polyvalente qui offre des outils essentiels pour le traitement d'images et la vision par ordinateur [8]. OpenCV est disponible dans plusieurs langages, dont Python. C'est avec ce langage qu'il sera utilisé pour mettre en œuvre cette première méthode.

### 2.1.1 HSV

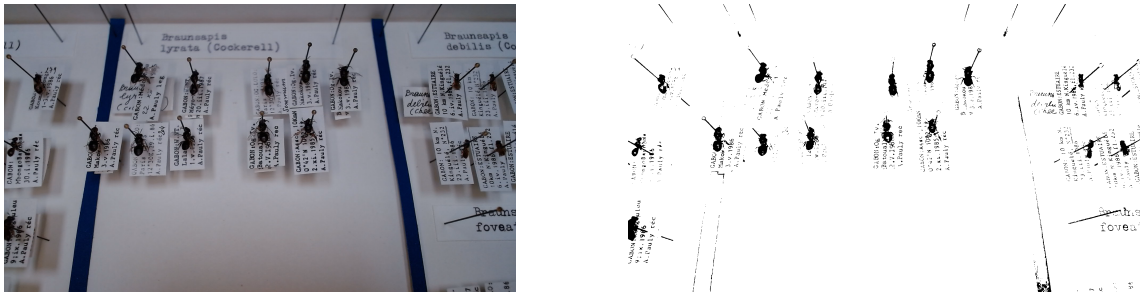
L'espace colorimétrique HSV (Teinte, Saturation, Valeur) représente une image en décomposant l'information de couleur en trois composantes principales : la teinte, la saturation et la luminosité (ou valeur) [9].

- **Teinte :** La teinte représente la couleur dominante de l'image. Elle est mesurée en degrés sur le cercle chromatique (0° à 360°), où 0° et 360° correspondent au rouge, 120° au vert et 240° au bleu. La plage des valeurs de teinte s'étend de 0 à 179 dans OpenCV. La teinte permet de spécifier la "nuance" ou la "tonalité" de la couleur, indépendamment de sa luminosité ou de sa saturation.
- **Saturation :** La saturation mesure l'intensité ou la pureté de la couleur. Elle est exprimée en pourcentage, allant de 0% (gris, aucune couleur) à 100% (couleur pure). Dans OpenCV, cette valeur va de 0 à 255. Une saturation élevée indique des couleurs vives et riches, tandis qu'une saturation faible se traduit par des couleurs plus pâles ou grises.
- **Luminosité :** La luminosité (ou valeur) représente la luminosité de la couleur. Elle est également exprimée en pourcentage, allant de 0% (noir) à 100% (blanc). Tout comme pour la saturation, cette valeur est comprise entre 0 et 255 dans OpenCV. Une valeur élevée signifie une couleur plus lumineuse, tandis qu'une valeur faible indique une couleur plus sombre.

Le principe de l'image HSV consiste donc à représenter chaque pixel de l'image en fonction de ces trois composantes. Contrairement à l'espace RGB, où chaque pixel est représenté par des valeurs de rouge, vert et bleu, l'espace HSV offre une séparation plus compréhensible des caractéristiques de couleur. Il est donc plus facile de manipuler et de segmenter l'information de couleur. De plus, L'espace HSV est plus robuste à l'éclairage. En effet, la luminosité (V) est séparée du reste, réduisant ainsi l'impact des variations d'éclairage sur la détection des objets [9].

### 2.1.2 Masque

Une fois l'image d'origine convertie sous forme HSV, un masque comprenant toutes les parties de l'image qui se situent entre les valeurs minimales et maximales d'un seuil est extrait de l'image (Fig. 2.1). Le seuil est représenté sous la forme d'un triplet [H,S,V] allant de [0,0,0] à [179,255,255] dans OpenCV. Les insectes, le cadre de la boite, les épingles ainsi que l'écriture des étiquettes se différencient du fond blanc de la boite et peuvent donc être compris dans le masque. Il reste à isoler les insectes du reste.

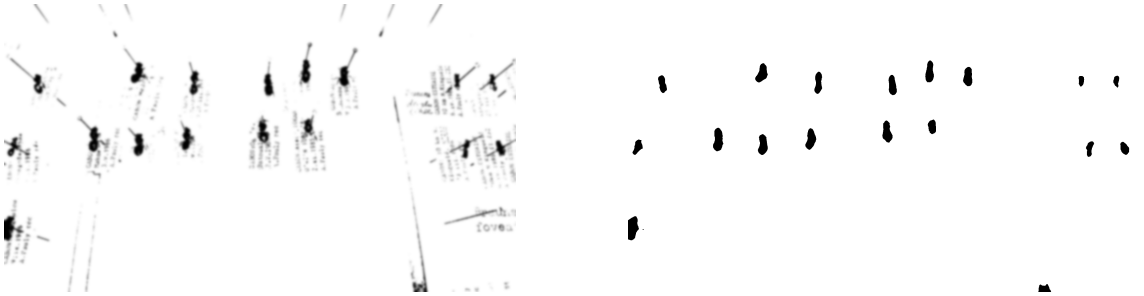


(a) Image originale (b) Masque

FIGURE 2.1 – Application d'un masque pour des valeurs HSV entre [0,0,36] et [179,255,255]

### 2.1.3 Flou médian

Le flou médian est une technique de traitement d'image qui consiste à remplacer la valeur de chaque pixel dans une image par la médiane des valeurs des pixels voisins. Contrairement au flou gaussien qui utilise une moyenne pondérée, le flou médian utilise la valeur médiane pour chaque pixel (Fig. 2.2). Pour chaque pixel de l'image, une fenêtre carrée ou rectangulaire est définie autour du pixel. La taille de cette fenêtre est appelée le noyau du filtre (kernel), elle varie en fonction de la taille du flou à appliquer. Les valeurs des pixels à l'intérieur de la fenêtre de convolution sont ensuite collectées puis triées par ordre croissant de sorte à calculer la médiane. La valeur du pixel d'origine est alors remplacée par cette valeur médiane [10]. Le masque étant constitué de pixels qui ont deux valeurs possibles (tout à fait noir ou tout à fait blanc), un pixel médian qui en découle conserve ce caractère binaire de par la définition même de la médiane, contrairement à un pixel moyen. Le flou médian est par conséquent efficace pour réduire le bruit dans l'image sans perdre la netteté des contours, préservant ainsi les caractéristiques importantes de l'image.

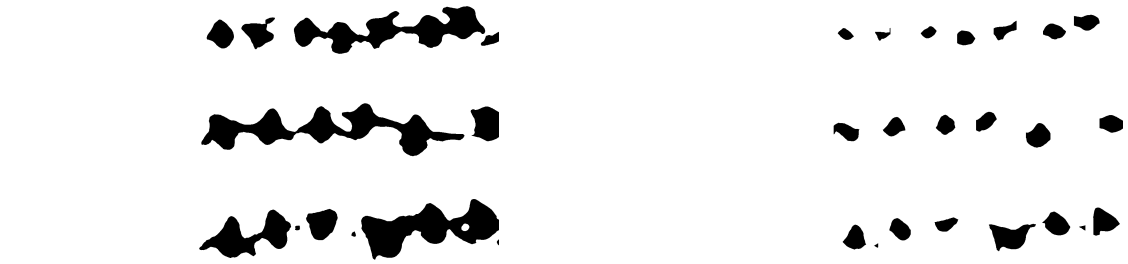


(a) Flou gaussien (b) Flou médian

FIGURE 2.2 – Application d'un flou avec un noyau de (25,25)

### 2.1.4 Érosion

L'érosion est une opération morphologique qui consiste à « grignoter » les contours des objets dans une image. Cette opération est souvent utilisée pour réduire la taille des objets ou éliminer les petits détails indésirables, mais dans ce cas-ci, elle est très utile pour séparer les insectes qui sont proches les uns des autres (Fig. 2.3). Pour chaque pixel, une région définie par le noyau est examinée : si tous les pixels de cette région sont noirs, alors le pixel d'origine est conservé, sinon il est érodé (pixel blanc) [11]. Il est tout à fait possible d'effectuer une érosion verticale ou horizontale en ajustant la forme du noyau. Pour une érosion verticale, le noyau a une seule colonne et plusieurs lignes. De même, pour une érosion horizontale, le noyau a une seule ligne et plusieurs colonnes. Ceci permet de séparer les insectes en fonction de leur disposition. En effet, si les insectes à séparer sont situés les uns à côté des autres, il convient de faire une érosion verticale. De la même manière, une érosion horizontale sera utilisée pour des insectes à séparer situés les uns au-dessus des autres. Il est évidemment possible de combiner les deux directions.



(a) Image avant érosion verticale (b) Image après érosion verticale

FIGURE 2.3 – Application d'une érosion verticale avec un noyau de (65,1)

Il est à noter que cette transformation entraîne une considérable réduction de la taille des insectes. Bien que cela puisse accroître la précision du décompte d'insectes, cela altère les données statistiques relatives à la longueur et à la largeur moyenne des insectes, ainsi que la densité de la boîte qui en découle.

### 2.1.5 Détection des contours

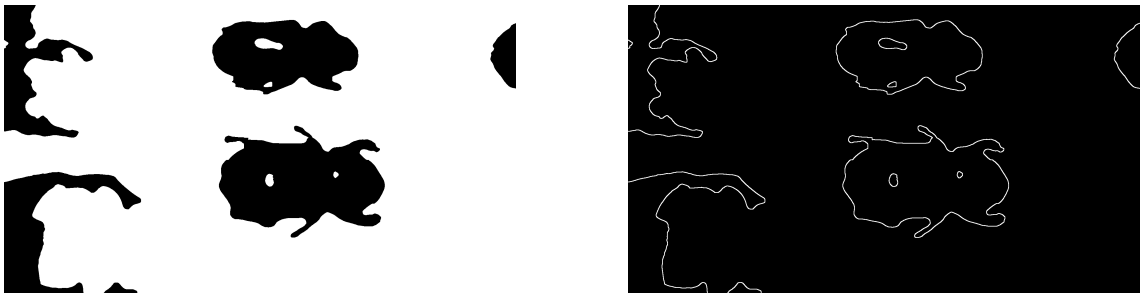
L'algorithme de Canny est utilisé pour la détection des contours. Il repose sur l'utilisation des opérateurs de Sobel. Ces opérateurs sont des filtres de convolution qui permettent d'estimer les dérivées partielles en $x$ et $y$. Le calcul du gradient $G$ se fait en combinant les dérivées partielles $G_x$ et $G_y$. Il indique à quel point l'intensité de l'image change à cet endroit. L'angle du gradient représente quant à lui la direction de ce changement d'intensité [12] :

$$G_x = \frac{\partial I}{\partial x} \text{ et } G_y = \frac{\partial I}{\partial y} \qquad G = \sqrt{G_x^2 + G_y^2} \qquad \theta = \arctan\left(\frac{G_y}{G_x}\right)$$

Les contours sont ensuite affinés en ne conservant que les points qui ont une intensité maximale le long de la direction du gradient. Pour chaque pixel, l'intensité du gradient est comparée avec celles de ses voisins dans la direction de celui-ci. Si le pixel a une intensité de gradient plus grande que celle de ses voisins, il est conservé ; sinon, il est supprimé. Cela permet d'obtenir des contours plus fins et mieux définis, ce qui améliore la localisation précise des bords des insectes.

Le seuillage par hystérésis est la dernière étape. Cette technique utilise deux seuils, pour classer les pixels en trois catégories : forts (changement d'intensité significatif), faibles (changement d'intensité moins prononcé), ou non bords. Ce seuillage permet de maintenir les contours forts qui sont très probablement des contours réels, tout en conservant certains contours faibles qui sont spatialement connectés à des contours forts. Cela contribue à obtenir des contours plus complets et résistants au bruit.

L'algorithme de Canny, initialement conçu pour le traitement d'images en niveaux de gris, est tout à fait capable de traiter des images binaires, malgré l'absence de gradients distincts. (Fig. 2.4). Une légère dilatation du contour est effectuée pour éviter toute discontinuité de celui-ci.



(a) Image avant détection des contours (b) Image après détection des contours

FIGURE 2.4 – Application de l'algorithme de Canny pour la détection des contours

### 2.1.6 Encadrement des contours

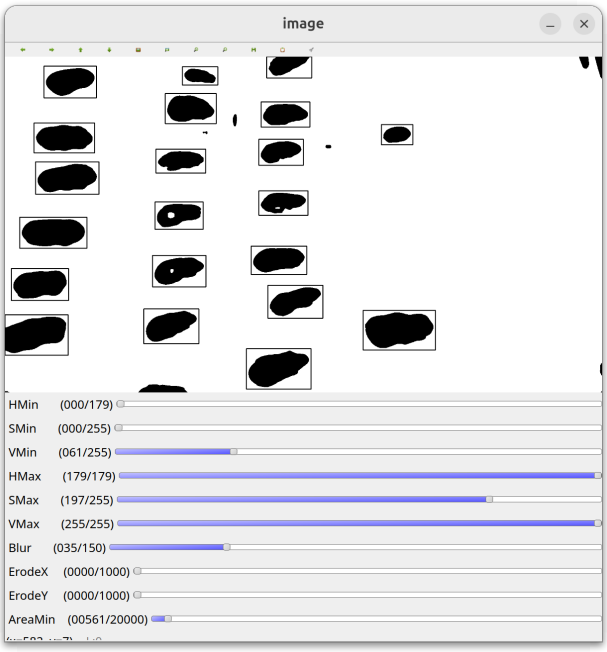
Pour sélectionner les contours pertinents de l'image, OpenCV propose la fonction `findContours` [13]. Cette fonction fournit la liste de tous les contours d'une image ainsi que leur hiérarchie respective. La hiérarchie permet de décrire la relation entre les différents contours, en particulier lorsque ceux-ci sont imbriqués les uns dans les autres (relations de parenté). Dans le cas spécifique des insectes dans une boite entomologique, il convient d'extraire uniquement les contours externes (hiérarchie la plus haute). Cela évite de prendre en compte les trous présents dans les insectes, dû au fait que ces zones n'aient pas été incluses dans le masque. Cette situation est souvent observée, notamment pour des insectes dotés d'une carapace brillante qui réfléchit l'éclairage. La fonction `findContours` permet différents modes de récupération des contours. Pour récupérer uniquement les contours externes, il est nécessaire de définir la valeur du mode sur `RETR_EXTERNAL`.

Toujours dans une optique de ne pas sélectionner des contours non pertinents, il convient de définir une aire minimale et maximale en dehors de laquelle les contours ne seront pas retenus. Les contours trop longs et trop larges seront également rejetés pour éviter une détection des bords de la boite. Les contours sont ensuite encadrés, au moyen de la fonction `rectangle()`, définissant ainsi une largeur et une longueur à l'insecte comptabilisé (Fig. 2.5).

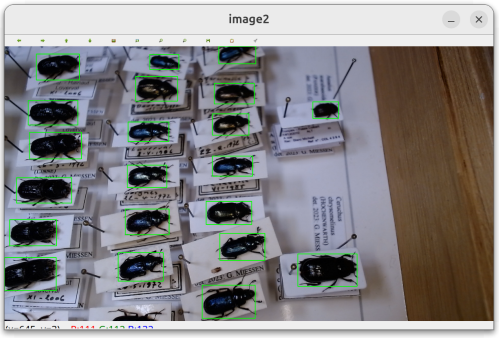
FIGURE 2.5 – Encadrement des contours avec une aire minimale de 1840 pixels carrés et une aire maximale de 2000000 pixels carrés

### 2.1.7 Interface graphique

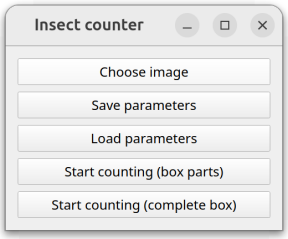
Étant donné que les paramètres optimaux sont spécifiques à chaque boîte entomologique et dépendent des caractéristiques chromatiques des insectes, il est nécessaire de les modifier avant chaque comptage. Pour cela, l'utilisation d'une interface graphique simplifie grandement la tâche. PyQt est un module libre qui permet de créer facilement des interfaces graphiques en Python, il sera employé pour mettre en œuvre cette tâche. Les paramètres sont ajustables à l'aide de curseurs, et l'image est modifiée simultanément dans le visualiseur. Il est ainsi possible de voir en temps réel le masque extrait ainsi que les encadrements des contours sur l'image originale, en fonction des paramètres établis (Fig. 2.6).




(a) Fenêtre de modification des paramètres et de visualisation du masque avec encadrement des contours




(b) Fenêtre de visualisation de l'image originale avec encadrement des contours




(c) Fenêtre de sélection d'actions

FIGURE 2.6 – Interface graphique pour le comptage d'insectes

Le bouton "Choose image" de la fenêtre de sélection d'actions permet de choisir une image afin de calibrer les paramètres de détection. Les boutons "Save parameters" et "Load parameters" permettent de sauvegarder et de charger des paramètres au moyen d'un fichier texte. Deux modes de comptages des insectes sont disponibles : le comptage pour une boite prise en plusieurs photos, selon l'approche adoptée par GxABT (mode 0), et le comptage pour une boite prise en photo dans son entièreté, à l'instar des données fournies par l'Africamuseum (mode 1).

### 2.1.8 Comptage des insectes

Le mode 0 lance un script Python en passant en argument tous les paramètres de seuillage ainsi que le chemin d'accès vers l'image choisie précédemment. Ce script réalise les étapes 2.1.1 à 2.1.6 sur toutes les images présentes dans le dossier de l'image en question, pour ainsi couvrir l'entièreté de la boite entomologique. Compte tenu du fait que les images se chevauchent entre elles, une zone de comptage est définie pour chaque partie de boite (Fig. 2.7). Les insectes ne seront pris en compte que s'ils se situent dans la zone de comptage, le coin supérieur droit de l'insecte ayant été arbitrairement pris pour référence. Les insectes en dehors de cette zone seront comptabilisés sur une autre image. Ceci évite un comptage multiple du même insecte présent sur plusieurs images. La zone de comptage est étendue jusqu'au bord de l'image si cette zone n'est pas couverte par une autre image (bord de la boite). Dans le mode 1, seul l'image sélectionnée est analysée, et tous les insectes détectés sont pris en compte.

FIGURE 2.7 – Zone de comptage (encadré rouge), insectes comptabilisés (encadrés bleus), insectes non comptabilisés (encadrés verts)

### 2.1.9 Données générées

Plusieurs données sont enregistrées durant le processus de comptage :

Un fichier d'annotation est généré pour chaque image analysée. Si la boite entomologique est prise en plusieurs images (mode 0), les fichiers sont regroupés dans un même sous-dossier. Le contenu et l'utilisation de ses fichiers seront détaillés à la sous-section 2.3.3

Pour chaque insecte détecté, une image individuelle est extraite de l'image originale en découpant celle-ci aux dimensions de détection. Celle-ci est enregistrée dans un répertoire avec tous les insectes de la même boîte. Ces données seront utilisées pour la tâche de classification (voir chapitre 3).

Pour le mode 0, une image de sortie incluant les différents encadrés (zone de comptage et insectes) est générée pour toutes les parties de la boite. Ceci permet de vérifier que les paramètres de seuillage sélectionnés plus tôt conviennent à toutes les parties de la boite. Pour le mode 1, l'unique image de sortie, incluant cette fois-ci seulement l'encadrement des insectes, est produite.

Les informations suivantes de toutes les boites sont enregistrées dans un unique fichier au format CSV :

- Les paramètres de seuillage
- Le chemin d'accès vers l'image (mode 0) ou le dossier d'images (mode 1)
- Le nombre d'insectes comptabilisés
- La largeur moyenne des insectes
- La hauteur moyenne des insectes
- Le mode utilisé

Le nombre d'insectes comptabilisés est également imprimé sur la sortie standard.

### 2.1.10 Perspectives

La détection par traitement de l'image présente plusieurs avantages. Tout d'abord, elle offre une flexibilité accrue, car elle permet à l'utilisateur de définir des seuils adaptés aux caractéristiques spécifiques des insectes observés, assurant ainsi une adaptabilité robuste. Ensuite, cette approche demande très peu de puissance de calcul, ce qui permet notamment de visualiser en temps réel la modification des paramètres de seuillage. Enfin, cette méthode a l'avantage non négligeable de ne pas nécessiter de données d'entraînement. Cette caractéristique lui permet de traiter efficacement n'importe quelle famille d'insectes sans la contrainte de devoir ré-entraîner un modèle qui pourrait ne pas inclure des spécimens de ce type dans son ensemble d'entraînement initial.

Cette approche présente cependant quelques limitations. Tout d'abord, l'automatisation est limitée. En effet, elle nécessite une intervention humaine pour calibrer les paramètres. Ensuite, bien que les insectes d'une même boite soient, dans la majorité des cas, de la même famille et partagent par conséquent des spécificités communes, une variabilité au sein de la famille peut être observée au niveau de la taille et de la couleur. Aussi, il est parfois difficile de séparer certains types d'insectes du fond ou les uns des autres à cause d'un contraste insuffisamment marqué ou d'une juxtaposition des insectes.

## 2.2 Détection par la méthode de Viola-Jones

La méthode de Viola-Jones est un algorithme de détection d'objets en vision par ordinateur. Elle a été proposée par Paul Viola et Michael Jones en 2001 [14]. Cet algorithme repose sur l'utilisation de caractéristiques simples appelées "caractéristiques de pseudo-Haar" et d'un classificateur en cascade pour améliorer l'efficacité de la détection [15].

### 2.2.1 Classificateur en cascade

Un classificateur en cascade est basé sur le principe de "diviser pour mieux régner". Plutôt que d'appliquer un seul classificateur complexe, plusieurs classificateurs plus simples dit "faibles" sont utilisés en séquence. Chaque classificateur faible prend une décision binaire (objet présent ou absent) en se basant sur la valeur d'une caractéristique extraite de l'image. L'idée principale de la cascade est de rejeter rapidement les régions qui ne contiennent pas l'objet, en appliquant des classificateurs moins coûteux en calcul au début de la cascade, et en augmentant progressivement la complexité des classificateurs pour les régions qui ont survécu aux étapes précédentes, concentrant ainsi les ressources de calcul sur les régions les plus prometteuses [16].

### 2.2.2 Caractéristiques de pseudo-Haar

Les caractéristiques de pseudo-Haar calculent des variations locales d'intensité dans une image. Ces caractéristiques peuvent être la combinaison de trois types fondamentaux [15] [16] :

- **Rectangles adjacents horizontaux ou verticaux :** Mesure la différence de la somme des pixels entre la partie supérieure (gauche) et inférieure (droite) du rectangle.
- **Ligne horizontale ou verticale :** Mesure la différence de la somme des pixels entre la ligne et les deux régions adjacentes délimitées par cette ligne.
- **Diagonales :** Mesure la différence de la somme des pixels entre les deux diagonales opposées.

Ces caractéristiques peuvent être déplacées et redimensionnées sur l'image, permettant ainsi de capturer des motifs d'intérêt dans des images présentant différentes échelles, orientations et positions.

### 2.2.3 Entrainement du classificateur en cascade

Lors de la phase d'apprentissage, le classificateur est entrainé à partir d'un ensemble d'images positives (contenant l'objet d'intérêt) et négatives (ne contenant pas l'objet). Il détermine quelles caractéristiques sont les plus discriminantes pour distinguer les objets à détecter. Chaque classificateur faible est en effet formé pour maximiser la différence entre les scores des régions positives et négatives. Une fois tous les classificateurs faibles formés, la cascade est construite en ajustant leurs seuils de manière à minimiser les faux positifs tout en maintenant un taux de détection élevé.

Pour créer un classificateur en cascade, OpenCV fournit des outils en ligne de commande [17]. Une interface graphique regroupant ces outils est également disponible [18]. Cette-dernière permet une utilisation plus intuitive et sera donc privilégiée pour créer un classificateur en cascade spécifique à la détection d'insectes.

Étant donné que, généralement, les boîtes entomologiques renferment des insectes d'une unique famille, il n'est pas impératif de disposer d'un classificateur capable de détecter simultanément deux insectes appartenant à des familles différentes. Il est dans ce sens pertinent d'élaborer un classificateur spécifique pour chaque famille d'insectes, car la variabilité morphologique entre les familles, que ce soit au niveau de la forme, de la taille, ou des motifs, est significative. En développant des classificateurs dédiés à chaque famille, le modèle peut se spécialiser davantage dans la reconnaissance des particularités distinctives de chaque famille d'insectes, améliorant ainsi la précision de la détection.

Pour l'acquisition des données d'entrainement, des images individuelles d'insectes de la famille concernée sont nécessaires. La disponibilité de ce genre d'images pour une famille spécifique étant limitée, les données générées par la méthode de détection par traitement de l'image (voir 2.1.9) seront exploitées en choisissant un échantillon des insectes correctement détectés pour constituer l'ensemble des images positives.

Pour les images négatives, la sélection des faux positifs générés par la méthode de détection par traitement de l'image est appropriée, mais insuffisante. Afin d'accroître la quantité d'images négatives, un script Python conçu à cette fin permet de choisir manuellement des zones d'intérêt à l'aide du curseur de la souris (Fig. 2.8), et de les enregistrer en tant qu'images individuelles. Ces zones comprennent le fond de la boite, l'encadrement ainsi que les écritures. Ces zones sont pertinentes, car elles sont présentes sur n'importe quelle image de boite entomologique et peuvent être confondues avec des insectes, car présentant une variation d'intensité. Cette méthode de sélection manuelle peut également être utilisée pour les images positives, dans le cas où il y a eu trop peu d'insectes détectés correctement par la méthode de détection par traitement de l'image, ou encore, si les dimensions des détections diffèrent considérablement de la réalité.

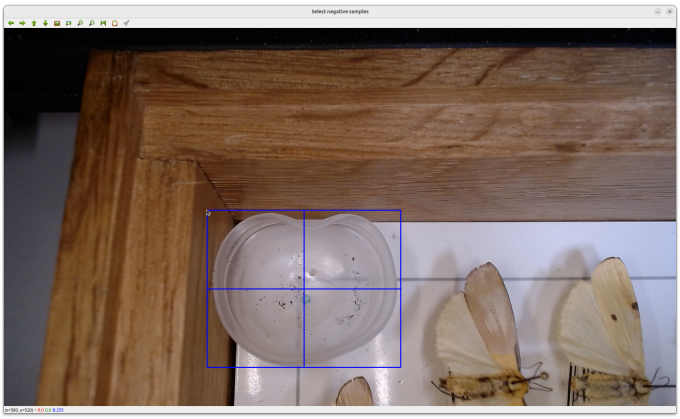


FIGURE 2.8 – Zone négative sélectionnée à l'aide du curseur de la souris

la fonction `opencv_createsamples` permet de créer des fichiers de description Haar à partir des images positives. Ces fichiers contiennent des informations sur les caractéristiques visuelles de l'objet. Ils sont indispensables pour l'emploi de la fonction `opencv_traincascade`, qui entraîne le classificateur. Cette fonction requiert également un nombre approprié de couches de décision pour mener à bien le processus d'entraînement. Un total de 20 couches constitue un compromis entre rapidité d'exécution et efficacité de détection.

### 2.2.4 Utilisation du classificateur en cascade

Une fois l'entrainement du modèle terminé, un dossier contenant les modèles créés pour chaque couche (`stage#.xml`) ainsi que le modèle final (`cascade.xml`) est créé. Ce fichier peut être chargé dans un script Python en utilisant la fonction `CascadeClassifier()` d'OpenCV pour créer une instance de classificateur cascade.

La méthode d'objet `detectMultiScale` est ensuite appelée sur l'instance avec différents paramètres :

- **image :** Il s'agit de l'image, en niveaux de gris, sur laquelle la détection d'objets en cascade est effectuée
- **scaleFactor :** Ce paramètre spécifie le taux de réduction de la taille de l'image à chaque étape de la recherche. Il permet de créer une pyramide d'images échelonnées pour détecter des objets de différentes tailles. Une valeur plus petite augmente la sensibilité de la détection (Fig. 2.9).
- **minNeighbors :** Indique le nombre minimum de voisins requis pour qu'une zone soit considérée comme un objet. Plus la valeur est élevée, plus la détection est stricte (Fig. 2.10).
- **minSize :** Définit la taille minimale d'un objet pour être détecté dans l'image.

Cette méthode retourne la liste des zones ainsi détectées où l'objet est probablement présent en fournissant des quadruplets (x, y, largeur, hauteur). De la même manière que pour la détection par traitement de l'image, chaque zone peut être comptabilisée au moyen d'un compteur, et peut être visualisées sur l'image originale par un encadré (Fig. 2.9 et Fig. 2.10).

(a) scaleFactor = 1.1

(b) scaleFactor = 1.4

FIGURE 2.9 – Détection de Chrysomelidae avec un classificateur en cascade entrainé sur cette famille pour différents taux de réduction de taille (minNeighbor constant = 1.2)

(a) minNeighbor = 1

(b) minNeighbor = 4

FIGURE 2.10 – Détection de Chrysomelidae avec un classificateur en cascade entrainé sur cette famille pour différents nombres minimum de voisins (scaleFactor constant = 3)

### 2.2.5 Comptage des insectes

Les scripts créés pour le comptage, en utilisant la méthode de détection par traitement de l'image, peuvent être adaptés à la méthode de Viola-jones en remplaçant quelques lignes. Le principe reste le même : comptabiliser les insectes se trouvant dans une zone de comptage définie sur toutes les parties d'une boite si celle-ci est de GxABT, et comptabiliser tous les insectes d'une image si celle-ci est de l'Africamuseum. Il est à noter que les fenêtres de détection étant carrées, il n'est pas possible de définir une largeur et une hauteur moyenne de l'insecte, mais bien une dimension moyenne globale.

### 2.2.6 Perspectives

Cette méthode offre une automatisation plus importante que la précédente, car les paramètres sont moins nombreux, et peuvent être optimisés pour être utilisés de manière générale. Un classificateur entrainé sur la famille des insectes concernés est cependant requis. Le temps d'entrainement du modèle est de quelques minutes, et l'utilisation du modèle est presque instantanée.

La détection d'insectes peut nécessiter des modèles plus complexes que ceux adaptés à la détection de visages (application très rependue de la méthode de Viola-Jones). En fonction de la variabilité des insectes au sein d'une même famille et de la complexité de leur forme et de leur texture, le modèle doit être capable de généraliser plus ou moins de conditions.

## 2.3 Détection par un modèle pré-entrainé

La détection d'objets a connu d'énormes progrès grâce à l'avènement des réseaux de neurones profonds. Plusieurs modèles pré-entraînés utilisant cette approche ont émergé ces dernières années, chacun avec ses avantages spécifiques. Parmi eux, YOLO (You Only Look Once) [19] se distingue comme l'un des modèles plus connus et largement utilisés dans le domaine de la vision par ordinateur [20].

### 2.3.1 Fonctionnement général de YOLO

L'idée fondamentale derrière YOLO est de traiter la détection d'objets comme un problème de régression, où un réseau de neurones convolutifs (CNN) est formé pour prédire les coordonnées des boîtes englobantes et les probabilités associées aux classes d'objets [21]. L'algorithme réalise toutes les étapes nécessaires à la détection d'objets (extraction de caractéristiques, prédiction des boîtes englobantes et des classes) en une seule passe, c'est-à-dire une seule itération du CNN.

L'image d'entrée est divisée en une grille de dimensions prédéfinies. Chaque cellule de la grille est responsable de la détection des objets qui s'y trouvent. YOLO utilise un réseau convolutif pour extraire les caractéristiques de chaque cellule et prédire des boîtes englobantes. Chaque boîte est caractérisée par ses coordonnées, la probabilité qu'un objet s'y trouve, et les scores de confiance pour chaque classe d'objet. Les probabilités sont calculées en utilisant une fonction d'activation sigmoïde pour garantir une valeur se situant entre 0 et 1. Le score de confiance final est calculé en multipliant la probabilité que la boîte contienne un objet par la probabilité d'appartenance à une classe la plus élevée. Les boîtes englobantes dont le score de confiance final est inférieur à un certain seuil sont ignorées lors de la phase de post-traitement.

Ces données sont ensuite combinées pour former les sorties finales du modèle : chaque cellule de la grille produit un ensemble de prédictions, et ces ensembles sont regroupés pour former les prédictions globales de l'algorithme. YOLO applique la suppression des non-maxima pour éliminer les boîtes redondantes et ne conserver que les boîtes les plus fiables : toutes les boîtes englobantes sont triées en fonction de leur score de confiance, du plus élevé au plus bas. La boîte englobante avec le score de confiance le plus élevé est sélectionnée comme boîte principale. Les boîtes ayant un chevauchement supérieur à un certain seuil par rapport à la boîte principale sont supprimées. Le processus est répété pour toutes les boîtes englobantes restantes. Cette suppression élimine les boîtes qui ont une grande intersection avec la boîte principale, laissant place à une seule détection pour un objet donné. Les boîtes sont pour finir associées à leur classe respective en se basant sur la probabilité d'appartenance à une classe la plus élevée.

### 2.3.2 Utilisation du modèle YOLO

La dernière version en date de YOLO est YOLOv8. Cette dernière version apporte de nouvelles fonctionnalités tout en améliorant la flexibilité et l'efficacité du modèle [22]. Cette version sera naturellement choisie. L'utilisation de ce modèle pré-entrainé est très intuitive et se fait en quelques lignes de code. Il suffit d'utiliser la classe YOLO du package ultralytics pour charger le modèle.

YOLOv8 est un groupe de modèles de réseaux neuronaux créés et entraînés à l'aide de PyTorch et exportés vers des fichiers avec l'extension .pt [23]. Il y a cinq tailles de modèle différentes ; plus le modèle est grand, meilleure sera la qualité de prédiction, mais au détriment de la rapidité d'exécution. La performance des différentes tailles de modèle est reprise dans l'annexe B. Étant donné que l'application actuelle ne requiert pas une analyse rapide des images, comme cela pourrait être le cas pour le traitement d'un flux vidéo, l'utilisation du modèle le plus grand (YOLOv8x.pt) se justifie.

La méthode `predict` appelée sur le modèle avec en argument une image fournit toutes les informations sur les boites englobantes retenues (coordonnées, classe, niveau de confiance). Pour visualiser le résultat de la détection, la fonction `Image.fromarray()` de la librairie PIL permet de transformer un tableau de valeurs en image (Fig. 2.11).

YOLOv8 est pré-entraîné sur l'ensemble de données COCO (Common Objects in Context), qui

est une énorme collection d'images de 80 objets dans différents contextes [23]. Malheureusement, ni la classe générale "insecte", ni des classes spécifiques d'insectes ne figurent dans cette liste, ce qui rend la détection défaillante.

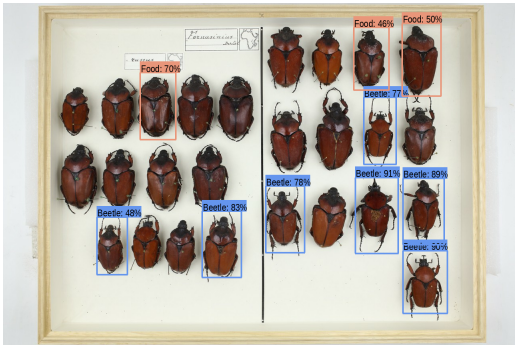
D'autres modèles pré-entrainés comme Faster R-CNN ont plus de classes, dont certaines spécifiques aux insectes [24], mais ne donnent tout de même pas de résultats acceptables (Fig. 2.12).


(a) Boite entomologique de Gembloux Agro Bio-Tech


(b) Boite entomologique de l'Africamuseum

FIGURE 2.11 – Détection d'insectes en utilisant le modèle YOLO (yolov8x.pt)

(a) Boite entomologique de Gembloux Agro Bio-Tech




(b) Boite entomologique de l'Africamuseum

FIGURE 2.12 – Détection d'insectes en utilisant le modèle Faster R-CNN

### 2.3.3 Personnalisation du modèle YOLO

Pour augmenter la précision du modèle YOLO, il est impératif de l'entrainer sur un jeu de données pertinent. En raison de la flexibilité des CNN, il n'est ici pas nécessaire de créer un modèle pour chaque famille d'insecte comme pour l'entrainement du classificateur en cascade. Dans le but de minimiser le nombre de modèles à développer, il convient d'entrainer le modèle avec différentes familles d'insectes du même ordre. Le modèle YOLO permet la détection de plusieurs classes d'objet distinctes, il est donc possible d'entrainer celui-ci avec différents ordres d'insectes, en attribuant un ordre à une classe de détection. Cependant, il est difficile pour le modèle de distinguer des objets similaires en une seule passe, comme le montre la Fig. 2.13. La tâche de classification doit donc être effectuée a posteriori.

Pour constituer le jeu de données, des images d'objets dans leur contexte, et non découpées à la taille de l'objet, sont préférables. En effet, contrairement à la méthode de Viola-Jones qui se base sur des fenêtres coulissantes, YOLO analyse l'image dans son entièreté pendant le processus d'entrainement, ce qui permet d'encoder implicitement des informations contextuelles [25], en incluant notamment les relations spatiales avec les autres objets [26].

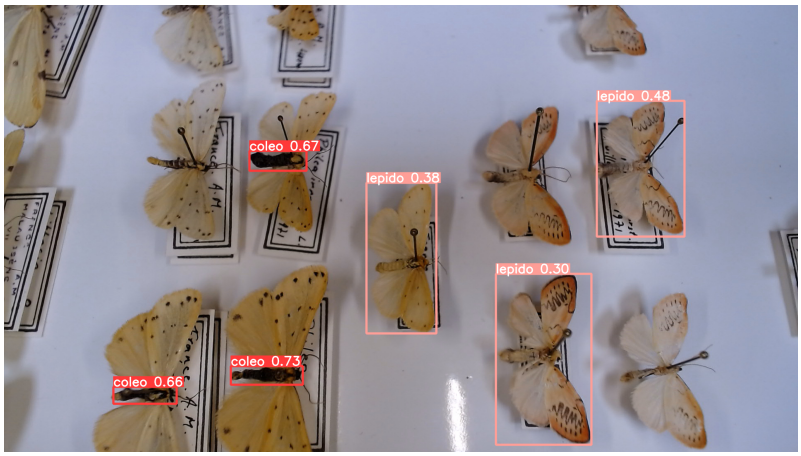



FIGURE 2.13 – Détection de plusieurs ordres d'insectes à la fois par un modèle YOLO personnalisé

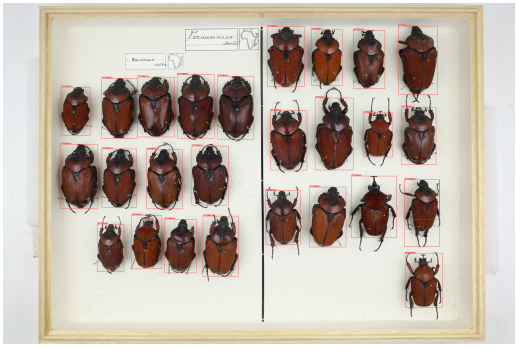
Chaque image nécessite un fichier d'annotation pour localiser les objets présents sur l'image. Cette annotation peut se faire manuellement au moyen d'un script Python créé à cette fin, ou automatiquement avec les données générées par la méthode de détection par traitement de l'image (sous-section 2.1.9). Cette solution sera privilégiée afin d'automatiser la tâche. Un fichier d'annotation répertorie les positions des insectes sur l'image. Une ligne du fichier correspond à un insecte et contient sa classe ("0" est défini pour la classe "insect"), ainsi que les quatre valeurs normalisées suivantes : coordonnées x et y du centre de l'insecte, largeur et hauteur de l'insecte.

Les données sont par la suite séparées en un ensemble d'entrainement pour l'apprentissage du modèle et un ensemble de validation pour mesurer la qualité du modèle. Une répartition de 80% pour l'ensemble d'entrainement et de 20% pour l'ensemble de validation est adoptée. Un fichier de description de l'ensemble de données (`custom_data.yaml`) doit également être créé. Celui-ci contient les chemins d'accès vers les images d'entrainement et de validation, le nombre de classes qu'il existe ainsi que la liste de leur nom dans l'ordre correct. Les index de cette liste correspondent aux nombres utilisés dans les fichiers d'annotation ("0" = "insect"). La structure requise pour l'arborescence des dossiers est décrite dans l'annexe C.

Il suffit enfin d'exécuter la méthode `train` sur le modèle en passant en argument le fichier de description créé plus tôt, le nombre de cycles d'entrainement (epochs) et le modèle pré-entrainé de base. Pour des raisons de rapidité d'exécution, un modèle plus petit (YOLOv8m.pt) que celui utilisé à la sous-section 2.3.2 est privilégié. Chaque cycle a deux phases : l'entrainement et la validation. Pour la phase d'entrainement, le modèle prend en entrée un lot aléatoire d'images d'entrainement. Les résultats sont transmis à la fonction de perte utilisée pour comparer la sortie du modèle avec le résultat des fichiers d'annotation pour ces images.

L'optimiseur ajuste ensuite les poids du modèle en fonction de la quantité d'erreurs de sorte à réduire celle-ci dans le cycle suivant. Par défaut, l'optimiseur de descente de gradient stochastique (SGD) est utilisé. Pour la phase de validation, le modèle prend en entrée un lot aléatoire d'images de validation et le résultat est comparé à celui des fichiers d'annotation pour déterminer la précision du modèle. Une fois tous les cycles d'entrainement terminés, le modèle avec la plus grande précision (`best.pt`) est généré. Des données relatives à la précision du modèle au cours de l'entrainement sont aussi produites. Il reste à appeler la méthode `predict` sur le modèle personnalisé pour visualiser le résultat (Fig. 2.14).

(a) Boite entomologique de Gembloux Agro Bio-Tech



(b) Boite entomologique de l'Africamuseum

FIGURE 2.14 – Détection d'insectes en utilisant le modèle YOLO personnalisé (best.pt)

### 2.3.4 Comptage des insectes

Pour cette méthode, les scripts de comptage réalisés pour les deux méthodes précédentes (détection par traitement de l'image et méthode de Viola-Jones) ne sont pas utilisables. L'Architecture de YOLO en est la cause : YOLO utilise une approche d'inférence unique où il prédit toutes les boîtes englobantes et les classes d'objets simultanément pour l'ensemble de l'image. Il ne permet pas une interaction utilisateur pour la sélection manuelle des objets avant l'inférence. Il est donc nécessaire de filtrer les détections après l'exécution de l'algorithme si besoin. C'est le cas pour les boites numérisées de GxABT. La liste des boites englobante est parcourue en ne prenant en compte que les boites dont les coordonnées du coin supérieur gauche (choix arbitraire) se situe dans une zone définie (zone de comptage) pour éviter un comptage multiple du même insecte présent sur plusieurs images. L'examen des éléments de la liste des boites englobantes permet également d'établir une largeur et une hauteur moyenne des insectes.

### 2.3.5 Perspectives

Cette méthode permet la plus grande automatisation possible pour la détection d'insectes. La détection s'exécute en effet sans la moindre sélection de paramètres. La capacité du modèle YOLO à généraliser et à détecter différents types d'insectes, y compris des insectes de familles différentes de celles sur lesquelles le modèle a été initialement entraîné, est remarquable. Ceci découle de la nature même des CNN, qui sont capables d'apprendre et de généraliser des caractéristiques visuelles pertinentes pour la tâche de détection d'objets.

En dépit de cela, l'entrainement du modèle nécessite des ressources de calcul considérables en raison de la profondeur du modèle et de la complexité des calculs impliqués. Dans le contexte de cette contrainte, l'utilisation des GPU de Google Colab s'avère être une solution viable pour considérablement accélérer le processus, comparé à l'utilisation de matériel local.

# Chapitre 3

# Méthodologie : Classification

Dans le domaine de la vision par ordinateur, les réseaux de neurones convolutionnels (CNN) ont émergé comme des instruments accomplissant des prouesses remarquables dans la classification d'images. Leur capacité à extraire des caractéristiques pertinentes en fait des acteurs clés dans la compréhension et la catégorisation d'images complexes [27]. L'application de cette puissante technologie à la classification des insectes pourrait permettre de distinguer des insectes jusqu'à un certain niveau de leur hiérarchie taxonomique, l'objectif étant de pouvoir les distinguer au niveau le plus bas possible. Les bases théoriques de la taxonomie pour les insectes seront par ailleurs abordées dans ce chapitre pour décrire cette hiérarchie. Pour évaluer l'efficacité des CNN dans cette tâche, une méthode comparative faisant appel aux machines à vecteurs de support (SVM) sera également étudiée.

## 3.1 Taxonomie

Les insectes sont classés en suivant un système de classification biologique basé sur la taxonomie. Ce système hiérarchique a été développé pour organiser les organismes vivants en groupes selon leurs similitudes évolutives. Voici comment sont classés les insectes [28] :

- **Règne :** Les insectes appartiennent au règne animal, qui englobe tous les organismes multicellulaires et hétérotrophes.
- **Embranchement (ou phylum) :** Les insectes font partie de l'embranchement des arthropodes, qui regroupe des organismes caractérisés par un squelette externe, des pattes articulées et un corps segmenté.
- **Sous-embranchement (ou super-classe) :** Les insectes appartiennent au sous-embranchement des hexapodes, qui signifie qu'ils ont six pattes.
- **Classe :** La classe des insectes (Insecta) représente le groupe au sein duquel figurent ces organismes.
- **Ordre :** La classe des insectes est subdivisée en plusieurs ordres en fonction des caractéristiques spécifiques. Par exemple, les coléoptères forment un ordre (Coleoptera), les lépidoptères en forment un autre (Lepidoptera), et ainsi de suite.
- **Famille :** Chaque ordre est ensuite divisé en familles. Les familles regroupent des insectes qui partagent des similitudes plus spécifiques. Les boites entomologiques sont dans la majorité des cas constitués d'insectes d'une même famille.
- **Genre et sous-genre :** Les familles sont divisées en genres, qui rassemblent des espèces ayant des caractéristiques plus étroitement liées.
- **Espèce et sous-espèces :** La plus petite unité de classification est l'espèce, elle regroupe des individus capables de se reproduire entre eux et de produire une descendance fertile.

## 3.2 Classification à l'aide d'un CNN

Les réseaux de neurones convolutifs (CNN) représentent une catégorie spécifique de réseaux de neurones profonds particulièrement adaptée à la reconnaissance d'images. Les CNN ont connu un grand succès dans le domaine de la vision par ordinateur en raison de leur capacité à capturer des caractéristiques d'une image, ce qui les rend opportuns pour la classification.

### 3.2.1 Fonctionnement général d'un CNN

Un CNN est constitué de plusieurs couches [27] :

- **Couche d'entrée :** Prend en charge les images d'entrée,
- **Couches convolutives :** Appliquent des filtres pour extraire des caractéristiques locales de l'image.
- **Couches de regroupement (pooling) :** Réduisent la dimension spatiale des représentations, en conservant les caractéristiques les plus importantes.
- **Couches entièrement connectées :** Réalisent la classification finale en utilisant les caractéristiques extraites.

Les couches convolutives constituent les éléments fondamentaux des CNN. Une couche opère en appliquant un filtre, une petite matrice de poids, à l'image d'entrée afin d'en extraire des caractéristiques. La convolution est réalisée en déplaçant le filtre sur l'image et en calculant le produit scalaire entre les valeurs du filtre et celles correspondantes de l'image, à chaque position. Le résultat de l'opération forme une nouvelle carte de caractéristiques.

Le rembourrage (padding) consiste à ajouter des zéros autour des bords de l'image d'entrée avant la convolution. Il est utilisé pour préserver la taille spatiale de l'image d'entrée et éviter une réduction excessive de la dimension. Le pas (stride), qui est la distance en pixels à laquelle le filtre se déplace sur l'image, influence également la dimension de la carte de caractéristiques résultante. Plus le stride est grand, plus la dimension est réduite. Après la convolution, une fonction d'activation est appliquée pour introduire la non-linéarité dans le réseau, permettant au modèle d'apprendre des représentations complexes [27].

Les couches de regroupement (pooling) réduisent quant à elles la dimension spatiale des cartes de caractéristiques générées par les couches convolutives. Cela rend la représentation invariante aux translations et aux petites variations dans l'image [29]. Comme dans la convolution, le pooling utilise également une fenêtre glissante. Pour chaque région couverte par la fenêtre de pooling, la valeur maximale (max pooling) ou la valeur moyenne (average pooling) est sélectionnée comme valeur représentative de cette région. Les couches entièrement connectées effectuent enfin la classification en prenant les caractéristiques extraites et en les transformant en une sortie de probabilité pour chaque classe.

Le modèle est entraîné sur un ensemble de données annotées, et une fonction de perte évalue la disparité entre les prédictions du modèle et les étiquettes réelles. L'optimiseur intervient ensuite pour ajuster les poids du réseau, en cherchant à minimiser cette perte lors de la rétropropagation du gradient. Pour renforcer la généralisation du modèle, l'augmentation des données est fréquemment employée, appliquant des transformations mineures aux images d'entraînement telles que des rotations et des retournements. Enfin, des techniques de régularisation, comme le dropout, sont intégrées pour prévenir le surapprentisage (overfitting) en désactivant aléatoirement des neurones pendant l'entraînement. Ces stratégies globales contribuent à façonner un modèle robuste et généralisable. Le modèle est finalement évalué sur un ensemble de données de test distinct pour évaluer sa performance et sa généralisation sur des données non vues.

### 3.2.2 Création du CNN

Il existe plusieurs bibliothèques en Python qui permettent la création et l'entraînement de CNN. TensorFlow est une bibliothèque open source développée par Google pour le machine learning et l'apprentissage profond. Depuis la version 2.0, Keras est intégré par défaut à TensorFlow pour la construction de modèles de réseaux neuronaux [30]. L'intégration est transparente, ce qui signifie que les utilisateurs peuvent importer la sous-bibliothèque `tf.keras` pour accéder à l'implémentation Keras tout en bénéficiant des fonctionnalités de TensorFlow. Les modèles Keras sont construits de manière modulaire, ce qui permet de créer facilement des architectures complexes et personnalisées en assemblant des blocs de construction prédéfinis.

### 3.2.3 Utilisation du CNN

Pour constituer les données d'entrainement et de validation, les images individuelles des insectes générées par la méthode de détection par traitement de l'image sont utilisées (voir sous-section 2.1.9), les faux positifs ayant été retirés manuellement. Chaque classe d'instance, composée d'insectes partageant un même groupe taxonomique, est contenue dans un dossier. L'algorithme commence tout d'abord par parcourir toutes les images pour les redimensionner. En effet, les CNN requièrent des images de taille uniforme en entrée pour des raisons liées à leur structure et à leur fonctionnement. Pour que toutes les images aient la même taille sans perdre de l'information, une solution consiste à ajouter des pixels blancs sur les bords des images n'ayant pas les bonnes dimensions pour simuler le fond de la boite entomologique. L'image préserve ainsi les proportions et l'intégrité de l'insecte dans son ensemble. Ces pixels additionnels ne renferment pas d'informations visuelles significatives et, par conséquent, n'interfèrent pas avec le processus d'apprentissage du modèle. Les images redimensionnées et leur classe correspondante sont ajoutées aux listes `images` et `labels`. Le jeu de données est ensuite divisé en un ensemble d'entrainement et de test au moyen de la fonction `train_test_split()` de la librairie `sklearn` avec une répartition 80%-20%.

Le modèle `keras.Sequential()` permet d'empiler les couches de manière séquentielle, c'est-à-dire les unes sur les autres. Les données passent à travers chaque couche dans l'ordre dans lequel elles ont été ajoutées au modèle. Pour ajouter une nouvelle couche au modèle, il suffit d'utiliser la méthode `add`. Trois couches de convolution (`Conv2D`) sont ajoutées au modèle. Entre elles, sont placés des couches de pooling (`Maxpooling2D`). L'avantage du max pooling par rapport à l'average pooling réside dans sa capacité à préserver les caractéristiques saillantes, favorisant ainsi la détection d'éléments distinctifs dans les données. Une couche aplatit ensuite les données en un vecteur unidimensionnel avant de passer aux couches entièrement connectées. Le modèle utilise la fonction d'activation `ReLU` pour toutes les couches, à l'exception de la dernière qui utilise `softmax` pour obtenir des probabilités pour chaque classe. Un autre modèle avec une couche de convolution supplémentaire sera également testé pour observer l'influence de la profondeur du CNN.

Une fois le modèle instancié, la phase suivante consiste à le compiler en spécifiant les paramètres tels que l'optimiseur et la fonction de perte. Le modèle de référence utilisera l'optimiseur "adam" et la fonction de perte "crossentropy". Celui-ci est ensuite entrainé sur l'ensemble d'entrainement. Il faut noter qu'il est possible d'utiliser un ensemble de validation pour suppléer l'entrainement du modèle, et ainsi surveiller le surapprentissage (overfitting) de celui-ci. Cependant, les données étant relativement peu abondantes, il a été décidé de toutes les utiliser pour l'entrainement. La configuration choisie pour l'entraînement comprend 20 epochs et une taille de lot de 32. Encore une fois, l'utilisation des GPU de Google colab permet un entrainement du modèle en quelques minutes.

Le modèle entrainé peut pour finir être évalué au niveau de sa précision grâce à la méthode `evaluate`, et prédire la classe d'une image en l'utilisant dans la méthode `predict`.

## 3.3 Classification à l'aide d'un SVM

Une Machine à Vecteurs de Support (SVM), est un algorithme d'apprentissage supervisé dont l'objectif principal est de trouver un hyperplan dans un espace multidimensionnel qui sépare de manière optimale les exemples de différentes classes [31].

### 3.3.1 Fonctionnement général d'un SVM

L'espace de caractéristiques est un espace multidimensionnel où chaque dimension représente une caractéristique extraite des données d'entrée. Le choix des caractéristiques est crucial, car il influence la capacité du SVM à séparer les classes de manière efficace. Voici quelques exemples de possibilités d'extraction de caractéristiques [32] :

- **Caractéristiques brutes :** Chaque pixel d'une image peut être considéré comme une caractéristique. Par exemple, dans une image de 64x64 pixels en niveaux de gris, l'espace de caractéristiques est de dimension 4096 (64 * 64).
- **Histogrammes de couleurs :** Pour les images en couleur, des histogrammes de couleurs peuvent être extraits, représentant la distribution des valeurs de couleur dans différents canaux (rouge, vert, bleu).
- **Descripteurs de texture :** Des méthodes telles que les descripteurs de texture basés sur la matrice de co-occurrence, les histogrammes de texture, ou les transformées de Gabor peuvent être utilisées pour capturer des informations sur la texture des images.
- **Histogrammes de gradients (HOG) :** HOG est une méthode qui mesure la distribution des gradients d'intensité dans une image.
- **Caractéristiques basées sur des points d'intérêt :** Des méthodes comme les descripteurs SIFT (Scale-Invariant Feature Transform) ou SURF (Speeded Up Robust Features) peuvent extraire des caractéristiques invariantes à l'échelle et à la rotation à partir de points d'intérêt.
- **Caractéristiques géométriques :** Pour des objets spécifiques, des caractéristiques géométriques telles que la forme, l'orientation et la taille peuvent être pertinentes.

L'objectif du SVM est de trouver un hyperplan qui sépare les exemples de deux classes différentes dans cet espace de caractéristiques. Cet hyperplan est défini par une équation linéaire de la forme $w * x + b = 0$ (fonction de décision), où $w$ est un vecteur de poids, $x$ est un vecteur d'entrée, et $b$ est le terme de biais. La classe d'un point $x$ est déterminée par cette fonction. Si $w * x + b > 0$, le point est d'une classe, sinon, il est de l'autre classe. Si le problème de classification comporte plus que deux classes, deux stratégies sont possibles [31] :

- **One-vs-One (OvO) :** Un classifieur SVM binaire est entraîné pour chaque paire de classes et donne un vote pour l'une des deux classes. La classe avec le plus grand nombre de votes est choisie comme classe finale.
- **One-vs-All (OvA) :** Un classifieur SVM binaire est entraîné pour chaque classe, traitant cette classe comme la classe positive et toutes les autres comme la classe négative. La classe associée au classifieur avec la mesure de confiance la plus élevée est la classe finale.

La marge est la distance entre l'hyperplan et les exemples les plus proches de chaque classe (vecteurs de support). Le SVM cherche à maximiser cette marge, car une marge plus grande implique une meilleure généralisation du modèle aux nouvelles données. Seuls ces vecteurs de support ont une influence sur la position et l'orientation de l'hyperplan [31]. Le coût de classification (C) contrôle quant à lui le compromis entre la maximisation de la marge et la tolérance aux erreurs d'entraînement. Dans le cas d'un SVM non-linéaire, des noyaux non-linéaires sont utilisés pour effectuer une séparation dans un espace de dimension supérieure [32].

### 3.3.2 Création et utilisation du SVM

Scikit-learn est une bibliothèque d'apprentissage automatique très populaire et intuitive. Elle implémente la classe SVC (Support Vector Classification) qui utilise un SVM avec une approche OvO. Elle sera utilisée pour la création du modèle. Tout comme pour le CNN, les données d'entrainement d'une classe sont contenues dans un même dossier. Le processus d'annotation est similaire à celui pour le CNN, sauf que les images ne sont pas stockées en deux dimensions : Les images sont aplaties en un vecteur unidimensionnel avec la fonction `flatten()` de NumPy. Le jeu de données est ensuite divisé de la même manière que pour le CNN.

Le modèle est ensuite construit au moyen de la classe SVC de `scikit-learn`. Celui-ci est paramétré en spécifiant le type de noyau et la fonction de coût. Parmi les différents noyaux testés, le noyau linéaire obtient les meilleures performances. Quant à la fonction de coût, la valeur C=1.0 est un choix de départ raisonnable, car elle correspond à une régularisation modérée. Le modèle est par la suite entrainé avec les données d'entrainement.

Comme pour le CNN, le SVM entrainé peut pour finir être évalué au niveau de sa précision au moyen cette fois-ci de la fonction `accuracy_score` et prédire la classe d'une image en l'utilisant dans la méthode `predict`. L'entrainement d'un SVM demande considérablement moins de ressources qu'un CNN. Elle ne nécessite pas l'utilisation de GPU et peut être donc facilement exécutée sur une machine locale.

# Chapitre 4

# Résultats : Détection

Les algorithmes de détection développés nécessitent une analyse au niveau de leur performance pour déterminer s'ils ont une fiabilité suffisante pour pouvoir être utilisés. Différentes techniques ont été utilisées en fonction des contraintes de la méthode de détection testée. Les deux dernières méthodes de détection nécessitent un entrainement considérable et il est donc difficile d'analyser la précision sur un échantillon important de boites entomologiques contrairement à la première méthode. Pour déterminer la précision d'une méthode, il faut tout d'abord définir l'unité expérimentale. Celle-ci correspond à la plus petite entité sur laquelle les mesures sont prises. Les algorithmes sont conçus pour calculer le nombre total d'insectes que compose une boite, il est donc adéquat de prendre la boite entomologique comme unité expérimentale.

## 4.1 Détection par traitement de l'image

### 4.1.1 Mise en place expérimentale

Pour déterminer la précision de l'algorithme de détection de l'image, 30 boites entomologiques pour chacune des deux méthodes de numérisation (GxABT et Africamuseum) ont été sélectionnées de manière à disposer d'une diversité représentative de chaque collection. L'ordre des coléoptères et celui des Lépidoptères étant prédominants dans les deux collections, de nombreuses familles de ces deux ordres ont été choisies. Les boîtes ont également été choisies de manière à garantir une diversité dans la taille des insectes. Ainsi, l'échantillon comprend aussi bien des insectes de quelques millimètres que d'autres de plusieurs centimètres, parfois regroupés dans la même boîte entomologique.

Après avoir exécuté l'algorithme sur les 60 boites, Un script R a été conçu pour traiter les informations générées par ces exécutions, contenues dans un fichier CSV (voir sous-section 2.1.9). Le script permet la réalisation de graphiques pertinents, en détectant notamment des corrélations entre la précision et certains paramètres.

### 4.1.2 Calcul de la précision

Pour déterminer la précision de la méthode de détection par traitement de l'image, le nombre d'insectes détectés par l'algorithme est comparé au nombre d'insectes réels, déterminé par un comptage manuel. La mesure :

$$\frac{\text{Nombre réel} - \text{Nombre détecté}}{\text{Nombre réel}}$$

donne une erreur relative, où une valeur proche de zéro indique une détection précise (la détection est proche du nombre réel d'insectes), et des valeurs plus éloignées de zéro indiquent des écarts plus importants entre la détection et la réalité. Si cette valeur est positive, cela signifie que l'algorithme a sous-estimé le nombre d'insectes, et si elle est négative, qu'il l'a surestimé.

Le nombre d'insectes détectés dépend fortement du calibrage des paramètres. Afin d'éviter tout biais, le calibrage des paramètres est optimisé individuellement pour chaque boîte sans connaître au préalable le nombre réel d'insectes qui la composent. le comptage manuel est en effet effectué a

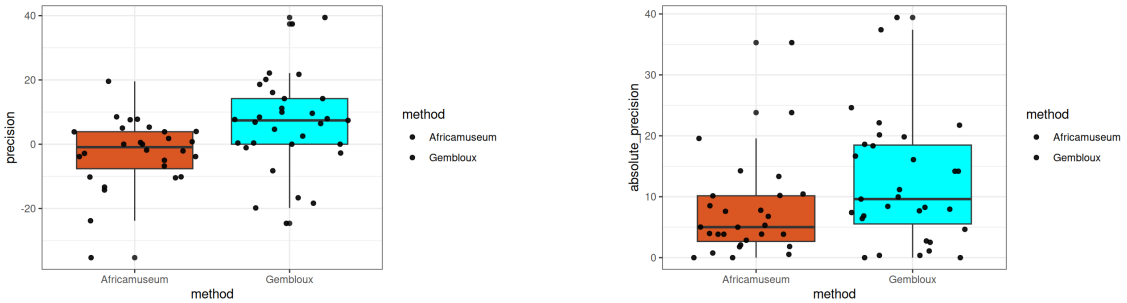
posteriori pour refléter la précision véritable de l'algorithme. Afin de faciliter le comptage manuel sur une boite de GxABT, un script Python a été développé pour la reconstituer le plus fidèlement possible. Ce script permet de réaliser le comptage sans avoir à parcourir toutes les parties de la boite. Chaque partie de la boîte est découpée à des dimensions appropriées, de manière à ne pas se superposer aux parties voisines, tout en garantissant une continuité dans l'image résultante. Les images sont ensuite collées les unes aux autres pour reconstituer la boite entomologique (Fig. 4.1). La jonction entre deux insectes est visible et peut présenter un décalage, mais cela ne gêne nullement le comptage. Une fois celui-ci effectué, le nombre d'insectes comptabilisés manuellement est ajouté à la suite de la ligne correspondante à la boite concernée, dans le fichier CSV.

Il est important de noter que le calcul des faux positifs et des faux négatifs (insectes non détectés) peut fournir une analyse plus approfondie de l'algorithme de détection. Cependant, la complexité de la tâche ne permet pas de traiter un échantillon suffisamment grand. En effet, pour une boite de GxABT, cette tâche nécessite l'examen de chaque partie de boite pour déterminer manuellement les zones faussement détectées et les insectes qui n'ont pas été détectés par l'algorithme. La mesure employée reste pertinente pour l'évaluation de la précision.

FIGURE 4.1 – Reconstitution d'une boite entomologique de GxABT

### 4.1.3 Visualisation des résultats

Le boxplot Fig. 4.2 montre l'erreur relative, en valeur absolue ou non, en fonction des deux méthodes de détection. Le graphique Fig. 4.3 permet quant à lui de visualiser la relation entre le nombre réel d'insectes et le nombre détecté. Plus les points sont proches de fonction identité (ligne rouge), plus la détection est précise.




(a) Erreur relative

(b) Erreur relative en valeur absolue

FIGURE 4.2 – Erreur relative en fonction de la méthode de numérisation

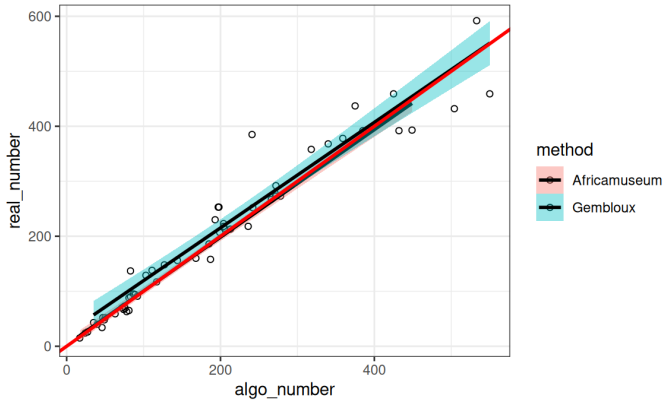



FIGURE 4.3 – Nombre réel d'insectes en fonction du nombre détecté

Plusieurs corrélations avec la précision sont décelées. L'influence de la densité sur la précision est illustrée à la Fig. 4.4. L'aire moyenne des insectes de la boite (définie par la largeur et la hauteur moyennes) est multipliée par le nombre réel d'insectes. Cette valeur correspond au nombre de pixels qui sont occupés par des insectes sur l'image. Pour obtenir la densité, il faut diviser cette valeur par le nombre de pixels que contient une boite. Cependant, étant donné que toutes les boites sont de dimensions similaires, il n'est pas nécessaire d'effectuer cette division pour réaliser une comparaison. L'influence de la densité peut naturellement être observée sur ses composantes, à savoir le nombre réel d'insectes à la Fig. 4.6 et la taille moyenne des insectes (ici, largeur moyenne) à la Fig. 4.5. L'influence du flou appliqué (blur) sur la précision est quant à elle illustrée à la Fig. 4.7.

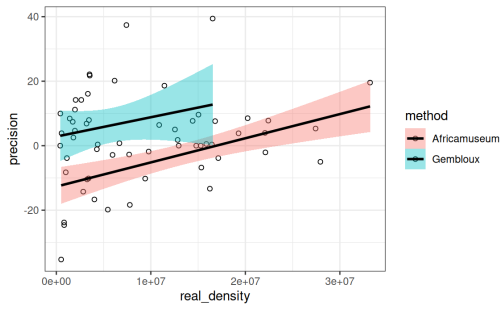



FIGURE 4.4 – Influence de la densité sur la précision

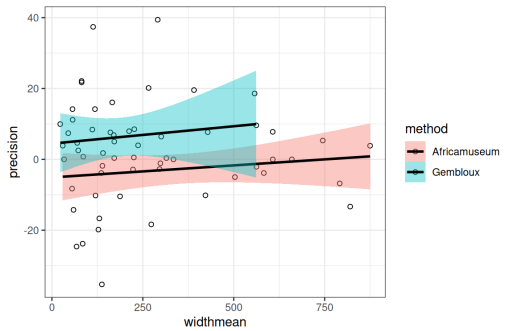



FIGURE 4.5 – Influence de la largeur moyenne sur la précision

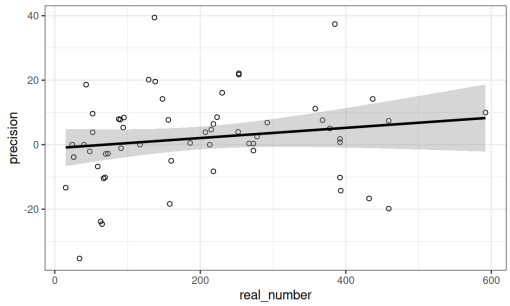



FIGURE 4.6 – Influence du nombre d'insectes sur la précision

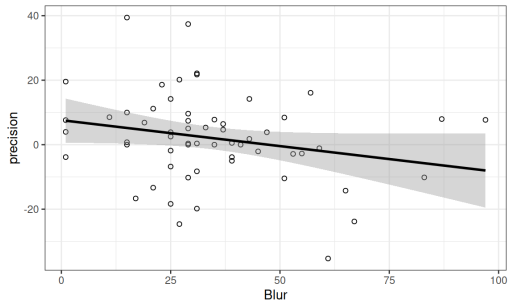



FIGURE 4.7 – Influence du flou appliqué sur la précision

### 4.1.4 Interprétation des résultats

Une constatation notable à la Fig. 4.2 est la présence de moins d'erreurs relatives sur les boites numérisées de l'Africamuseum que sur celles de GxABT. Par ailleurs, il est également observé que le nombre d'insectes détectés tend à être sous-estimé pour les boîtes provenant de GxABT.
Les résultats obtenus à partir de l'analyse du t-test sont :

— t = 2.7067
— df = 54.693
— p-value = 0.009047

La faible p-value indique une différence significative de précision entre les deux méthodes de numérisation, ce qui confirme la première constatation visuelle.

Le premier constat peut s'expliquer d'une part par la visualisation de la boite lors du calibrage des paramètres qui est complète sur une boite de l'Africamuseum alors qu'une seule partie de la boite peut être visualisée sur une boite de GxABT. D'autre part, des erreurs de comptage peuvent se produire lors de l'analyse des différentes parties d'une boite de GxABT. Une erreur se produit quand le même insecte est comptabilisé plusieurs fois, mais sur des images différentes ; ou inversement, lorsqu'il n'est pas comptabilisé du tout, car il n'a été présent dans aucune zone de comptage. Dans les deux cas, cela survient quand les bordures de la zone de comptage ne sont pas bien définies. Les insectes pouvant être épinglés à des hauteurs différentes, même au sein d'une boite, la distorsion focale n'est pas identique pour chaque image. La zone de comptage qui est fixe peut par conséquent ne pas correspondre à la réalité et produire une erreur.

En ce qui concerne l'influence de la densité sur la précision (Fig. 4.4), plus celle-ci est élevée, plus l'algorithme a tendance à sous-estimer le nombre d'insectes. Il en va de même pour le nombre d'insectes de la boite (Fig. 4.6), et pour la taille moyenne des insectes (Fig. 4.5), toutes deux composantes de la densité. Une explication de ce phénomène est que les insectes étant plus difficiles à séparer avec une densité élevée, un groupe d'insectes est comptabilisé comme une unité, ce qui réduit considérablement le nombre total d'insectes détectés.

Pour les paramètres de seuillage, seul le flou à une influence notable sur la précision. Plus il est élevé, plus l'algorithme a tendance à sur-estimer le nombre d'insectes. Les autres paramètres ne présentent pas de corrélation avec la précision, ce qui signifie que leur usage pour calibrer la détection n'a pas d'impact sur la précision.

## 4.2 Détection par la méthode de Viola-Jones

Pour pouvoir analyser la précision de la méthode de Viola-Jones, il faut tout d'abord ajuster les paramètres scaleFactor et minNeighbors de la méthode d'objet `detectMultiscale`, afin de trouver un équilibre entre la minimisation des fausses détections et la maximisation des vraies détections.

### 4.2.1 Calcul de la précision et du rappel

La précision mesure la proportion d'instances positives prédites par le modèle qui sont réellement positives. La formule de la précision est :

$$\frac{\text{Vrais positifs}}{\text{Vrais positifs} + \text{Faux positifs}}$$

Le rappel (recall) mesure la proportion d'instances positives réelles que le modèle a réussi à identifier. La formule du rappel est :

$$\frac{\text{Vrais positifs}}{\text{Vrais positifs} + \text{Faux négatifs}}$$

### 4.2.2 Visualisation des résultats

En calculant la précision et le rappel pour différents paramètres (scaleFactor et minNeighbors), il est possible de tracer la courbe précision-rappel, qui représente la relation entre les deux mesures. La position d'un point particulier sur la courbe représente le compromis entre la précision et le rappel. Le point rouge représente le compromis trouvé pour la méthode de détection par traitement de l'image. La Fig. 4.8 montre la courbe précision-rappel effectuée sur une image type. Plusieurs images ont été testées, et elles présentent toutes des courbes similaires

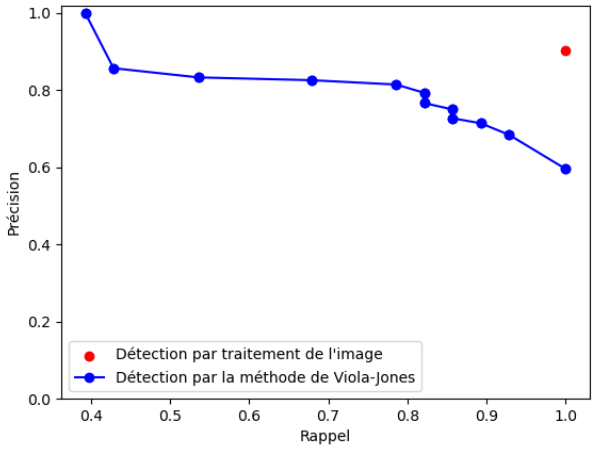



FIGURE 4.8 – Courbe précision-rappel pour différents paramètres

### 4.2.3 Interprétation des résultats

Aucun point de la courbe précision-rappel ne se rapproche du point de la méthode de détection par traitement de l'image. Le calibrage des paramètres pour la méthode de Viola-Jones appliquée à des insectes dans une boite entomologique ne permet pas de trouver un compromis acceptable entre faux positifs et faux négatifs. Il y a soit trop de détections erronées, soit trop de détections manquées. La méthode ne donne pas de performances acceptables pour être utilisée, ses performances ne seront donc pas analysées en profondeur.

## 4.3 Détection par le modèle YOLO personnalisé

### 4.3.1 Mise en place expérimentale

Pour l'analyse de la précision du modèle YOLO personnalisé, seuls deux modèles ont été créés sur la base des ordres prédominants dans les deux collections : un modèle a été entrainé sur l'ordre des coléoptères et un autre sur l'ordre des lépidoptères. Le modèle est suffisamment performant pour pouvoir généraliser plusieurs familles similaires tout en gardant une précision de détection élevée. Les modèles sont entrainés en utilisant des images annotées provenant de 10 boîtes entomologiques de GxABT et de 10 boites de l'Africamuseum. Cependant, il faut noter que la répartition est très déséquilibrée. Une boîte entomologique numérisée par la méthode de GxABT est composée de plusieurs dizaines d'images (dépend de la taille de la boite). Le nombre d'images par boite, multiplié par le nombre de boites utilisées pour l'entrainement, constituent un volume de données suffisent pour entraîner le modèle, contrairement aux boites de l'Africamuseum. La détection a été réalisée sur 10 boites entomologiques de familles d'insectes appartenant aux deux ordres sur lesquels le modèle personnalisé est entrainé, pour chaque méthode de numérisation.

### 4.3.2 Calcul de la précision

Lors de l'entraînement et de la validation du modèle YOLO personnalisé, plusieurs graphiques sont générés pour surveiller et évaluer la performance du modèle au fil du temps. Ces graphiques fournissent des informations sur la convergence du modèle et son apprentissage en fonction de chaque epoch. La fonction de perte (loss) mesure à quel point les prédictions du modèle diffèrent des annotations réelles. Elle est conçue pour que son optimisation conduise à des prédictions plus précises. Elle est divisée en plusieurs composantes :

- **box_loss** qui évalue la précision de la prédiction des boîtes englobantes
- **cls_loss** associée à la classification des objets (non pertinent si une seule classe)
- **dfl_loss** qui se focalise sur les différences de classes (non pertinent si une seule classe)

D'autres mesures sont aussi générées pour chaque epoch : précision et rappel (expliqué à la sous-section 4.2.1), mAP50 et mAP50-95 qui mesurent la précision moyenne (mean Average Precision) sur plusieurs seuils de confiance (score de confiance minimum à atteindre pour considérer la détection). Ces mesures fournissant ainsi une évaluation plus complète de la capacité d'un modèle à détecter des objets à différents seuils de confiance. mAP50 est calculée à un seuil de confiance de 50% ou plus, et mAP50-95 à un seuil entre 50% et 95%.

D'autres courbes sont générées une fois le modèle entrainé et permettent de définir le seuil de confiance optimal à appliquer :

- **Courbe de précision (P-curve) :** représente la précision du modèle pour différents seuils de confiance.
- **Courbe de rappel (R-curve) :** représente le rappel du modèle pour différents seuils de confiance.
- **Courbe Précision-Rappel (PR-curve) :** relation entre les deux mesures (voir sous-section 4.2.2)
- **Courbe F1 :** représente la relation entre le rappel et la précision pour différents seuils de confiance en montrant l'évolution du F1-score (compromis entre les deux valeurs). La formule du F1-score est $2 * \frac{P*R}{P+R}$

Pour déterminer la précision finale une fois le modèle entrainé, la même méthode que celle adoptée pour la détection par traitement de l'image a été utilisée (voir sous-section 4.1.2), afin d'obtenir des résultats comparables.

### 4.3.3 Visualisation des résultats

La Fig. 4.9 montre les données générées lors de l'entrainement du modèle et la Fig. 4.10 montre les performances une fois l'entrainement terminé.

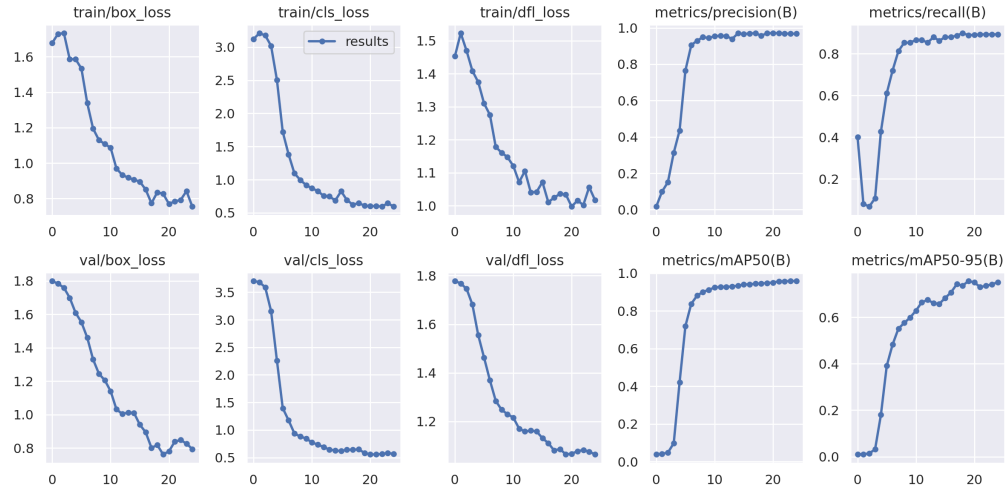
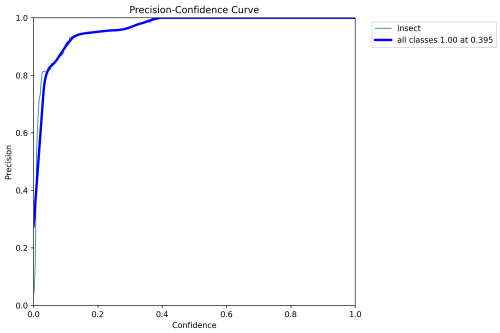
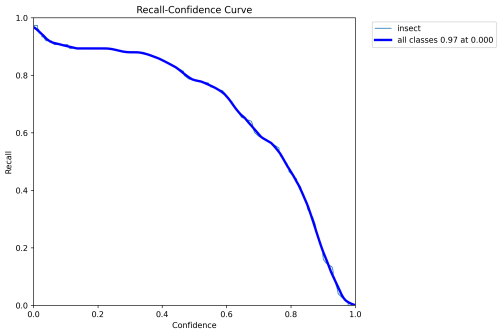


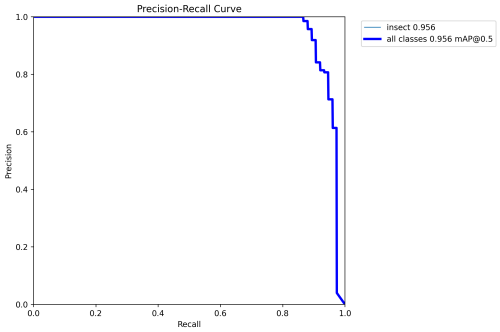

FIGURE 4.9 – Graphiques sur l'évolution de la perte et la précision au fil des epochs



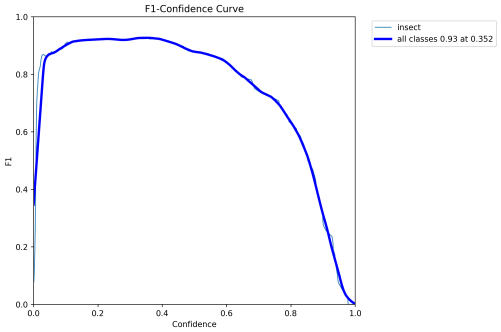

(a) Précision en fonction de la confiance




(b) Rappel en fonction de la confiance



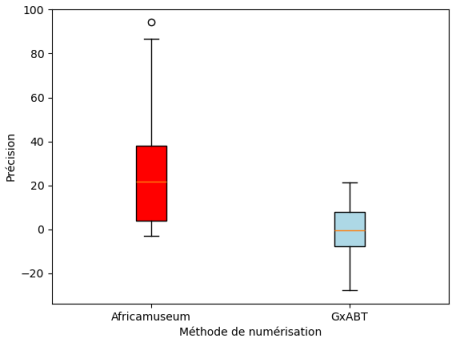

(c) Précision en fonction du rappel



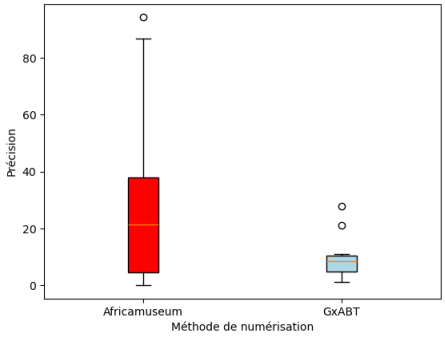

(d) F1-score en fonction de la confiance

FIGURE 4.10 – Graphiques pour l'optimisation de la confiance

La Fig. 4.11 montre l'erreur relative, en valeur absolue ou non, en fonction de la méthode de détection pour le modèle YOLO personnalisé. La Fig. 4.12 montre une comparaison des erreurs relatives entre la méthode de détection par le modèle YOLO personnalisé, et la méthode de détection par traitement de l'image pour les boites de GxABT.

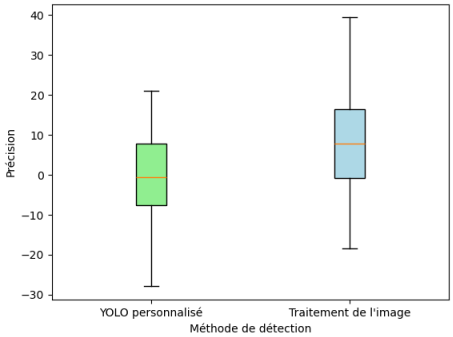



(a) Erreur relative

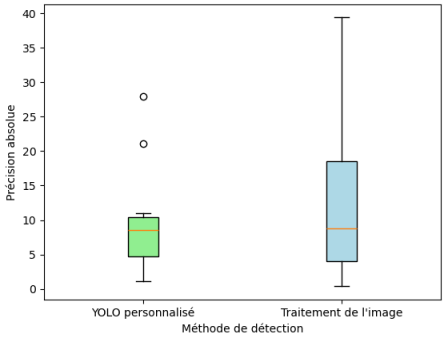



(b) Erreur relative en valeur absolue

FIGURE 4.11 – Erreur relative en fonction de la méthode de numérisation




(a) Erreur relative




(b) Erreur relative en valeur absolue

FIGURE 4.12 – Erreur relative en fonction de la méthode de détection

### 4.3.4 Interprétation des résultats

Les diverses fonctions de perte pour l'entrainement décroissent rapidement, et convergent ensuite vers un point de stagnation. Cela suggère que le modèle en phase d'apprentissage et d'amélioration a un nombre d'epochs suffisant pour atteindre un plateau au-delà duquel des améliorations supplémentaires ne se produiront plus de manière significative. Le fait que les fonctions de perte pour la validation diminuent de la même manière et n'augmentent pas significativement pour les derniers epochs indique qu'il n'y a pas d'overfitting détecté. De manière inverse, les fonctions de précision augmentent rapidement pour atteindre un plateau proche de 1, ce qui indique une bonne précision du modèle pendant l'apprentissage.

La courbe de précision révèle que le modèle a une tendance limitée à générer des prédictions positives incorrectes, même lorsque les seuils de décision sont faibles. La descente rapide de la courbe de rappel suggère quant à elle que le modèle manque un pourcentage important d'instances positives réelles, à partir d'une confiance relativement basse. La courbe PR s'approche du coin supérieur droit du graphique qui représente la performance optimale, où la précision et le rappel sont tous deux élevés. La courbe F1 maintient une valeur élevée en dessous d'une certaine confiance. Cette valeur élevée indique que le modèle atteint un bon équilibre entre précision et rappel, offrant une bonne performance globale. Ces graphiques permettent de trouver un seuil de confiance optimale pour réaliser la détection.

L'erreur relative est cette fois-ci bien plus élevée pour les boites entomologiques de l'Africamuseum que pour celles de GxABT. Pour les boites de l'Africamuseum, cette erreur est systématiquement positive, ce qui indique un nombre important de faux négatifs (détections manquées). Cette observation peut s'expliquer par le manque de compréhension du contexte des boites de l'Africamuseum par le modèle. En effet, la majorité des données d'entrainement provenant des images de boites de GxABT, le modèle n'arrive visiblement pas à interpréter les informations contextuelles des rares données d'entrainement provenant des boites de l'Africamuseum. Pour augmenter la précision sur ces boites, il est impératif d'acquérir davantage de données annotées ayant un contexte similaire.

Les boites de GxABT, quant à elles, obtiennent une bien meilleure précision. Elles sont entrainées sur un jeu de données visiblement suffisant pour que le modèle puisse appréhender le contexte. En comparant les erreurs relatives de la détection par le modèle YOLO personnalisé et de la détection par traitement de l'image, il apparait que la dispersion des erreurs est similaire pour les deux méthodes, mais que les erreurs semblent être mieux réparties autour de zéro pour la détection par le modèle YOLO. Dans l'ensemble, les erreurs positives et négatives sont équilibrées, contrairement à la détection par traitement de l'image qui a tendance à sous-estimer le nombre d'insectes. Le modèle YOLO personnalisé est donc à priori plus précis que celui de la détection par traitement de l'image pour les boites de GxABT

# Chapitre 5

# Résultats : Classification

## 5.1 Mise en place expérimentale

Pour découvrir jusqu'à quel niveau de hiérarchie un classificateur est capable de séparer deux insectes différents, il y a lieu de commencer à un certain niveau de hiérarchie où la séparation se fait correctement, et de descendre progressivement pour voir si les performances se dégradent significativement. Cette approche sera utilisée pour le CNN et le SVM. Les classificateurs seront tout d'abord entrainés sur 11 classes différentes pour les deux méthodes de numérisation (GxABT et Africamuseum). Chaque classe correspond à une famille d'insectes qui peut appartenir au même ordre ou non. Plus précisément, la répartition est la suivante :

- **GxABT :** 8 familles appartiennent à l'ordre des coléoptères, 2 à l'ordre des lépidoptères et 1 à l'ordre des hyménoptères
- **Africamuseum :** 8 familles appartiennent à l'ordre des coléoptères et 3 à l'ordre des lépidoptères

L'objectif est de découvrir si le classificateur est capable de séparer des insectes d'un même ordre, et de voir si la précision est significativement différente pour une classification de familles d'ordres différents.

Ensuite, les classificateurs seront entrainés respectivement sur 23 et 17 classes différentes pour les boites de GxABT et de l'Africamuseum. Chaque classe correspond cette fois-ci à un genre d'insectes qui peut appartenir à une même famille. La répartition est cette fois-ci :

- **GxABT :** 5 genres appartiennent à la famille des Zygaenidae, 3 à la famille des Geotrupidae, 2 à la famille des Lucanidae, 4 à la famille des Scarabaeidae, 2 à la famille des Cerambycidae, 3 à la famille des Nymphalidae et 3 à la famille des Lycaenidae.
- **Africamuseum :** 3 genres appartiennent à la famille des Acraeidae, 3 à la famille des Aphodiidae, 6 à la famille des Cetoniinae et 2 à la famille des Nymphalidae.

Là aussi, l'objectif est de découvrir la capacité du classificateur à séparer des insectes de la même famille ou non. Il n'est malheureusement pas possible de tester les classificateurs à un niveau de hiérarchie plus bas, car le nombre d'images d'entraînement est insuffisant. Il n'y a en effet que très peu d'insectes d'une même espèce dans une boite, ce qui rend l'entraînement du modèle difficile, voire impossible.

## 5.2 Classification par un CNN

Deux CNN avec des niveaux de profondeur différents seront évalués pour observer l'influence de cette variation. Le premier est composé de trois couches de convolutions, le deuxième de quatre.

### 5.2.1 Calcul de la précision

De la même manière que pour le modèle YOLO personnalisé, des informations pour évaluer la performance du modèle au fur et à mesure de son entrainement sont visualisables. La fonction de perte et l'exactitude (accuracy) du modèle sont par ailleurs visibles sur la sortie lors de l'entrainement du modèle, et peuvent être visualisées sur un graphique après coup. L'exactitude, à ne pas

confondre avec la précision, mesure la proportion totale d'instances correctement classées parmi toutes les instances, qu'elles soient positives ou négatives. Elle donne ainsi une vision globale de la performance du modèle sur l'ensemble du jeu de données, sa formule est :

$$\frac{TP+TN}{TP+TN+FP+FN}$$

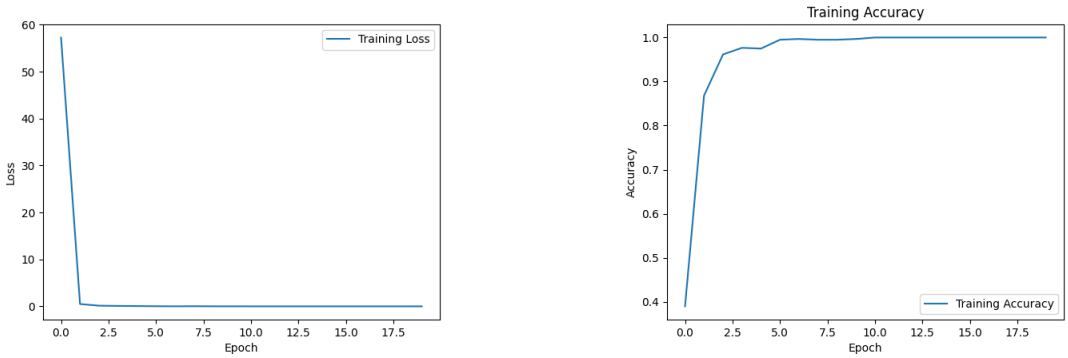
L'exactitude du modèle final est évaluée en calculant sa performance sur différentes répartitions des ensembles d'entraînement et de test, permettant ainsi de déduire une valeur moyenne.

Une matrice de confusion se révèle particulièrement efficace pour visualiser les résultats d'une classification. Celle-ci compare les prédictions du modèle avec les valeurs réelles sur l'ensemble de données testées. Les éléments diagonaux représentent les prédictions correctes du modèle et les éléments non-diagonaux une prédiction incorrecte.

### 5.2.2 Visualisation des résultats

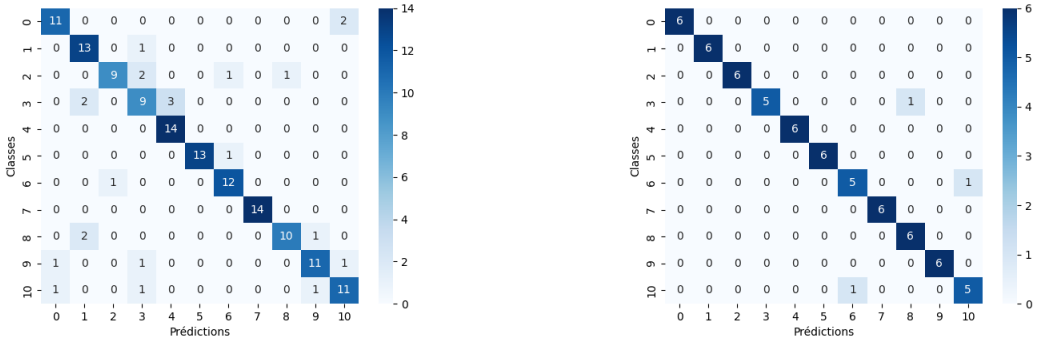
#### classification par familles d'insectes

La Fig. 5.1 montre la fonction de perte et l'exactitude pendant l'entrainement du CNN. Les formes des fonctions sont similaires pour l'entrainement sur les boites de GxABT et de l'Africamuseum, quelle que soit la profondeur du CNN. La Fig. 5.2 montre la matrice de confusion pour la classification des deux méthodes de numérisation. Là aussi, aucune variation notable n'est remarquable en fonction de la profondeur du CNN.




(a) Perte (b) Exactitude

FIGURE 5.1 – Évolution de la perte et de l'exactitude au fil des epochs pour la classification par familles d'insectes (CNN)

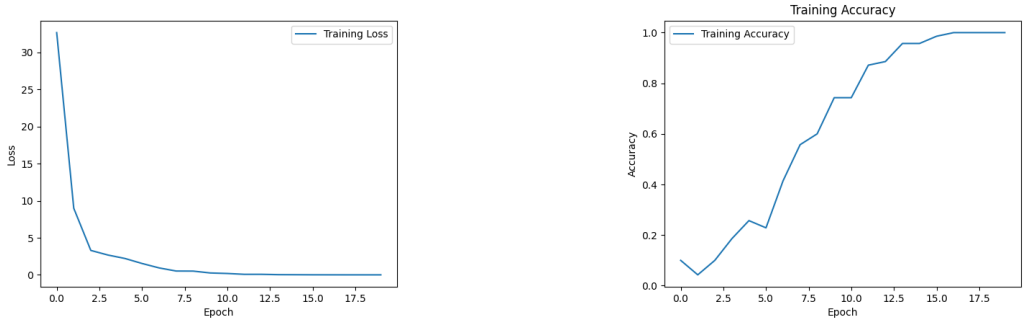



(a) GxABT (b) Africamuseum

FIGURE 5.2 – Matrice de confusion pour la classification par familles d'insectes (CNN)

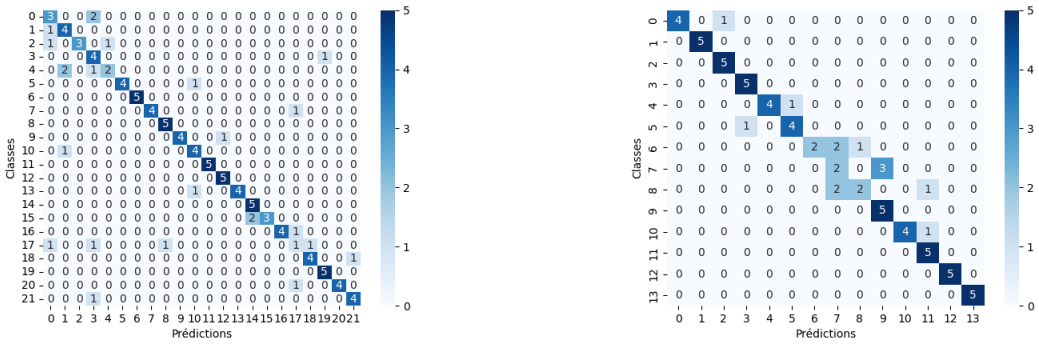
### Classification par genres d'insectes

Les mêmes graphiques sur la perte et l'exactitude sont produits pour la classification par genres d'insectes à la Fig. 5.3. Une fois de plus, il n'y a pas de différence significative en fonction de la méthode de numérisation et de la profondeur du CNN. En revanche, une variation notable est observable pour les matrices de confusion de CNN de profondeurs différentes (Fig. 5.4 et Fig. 5.5).



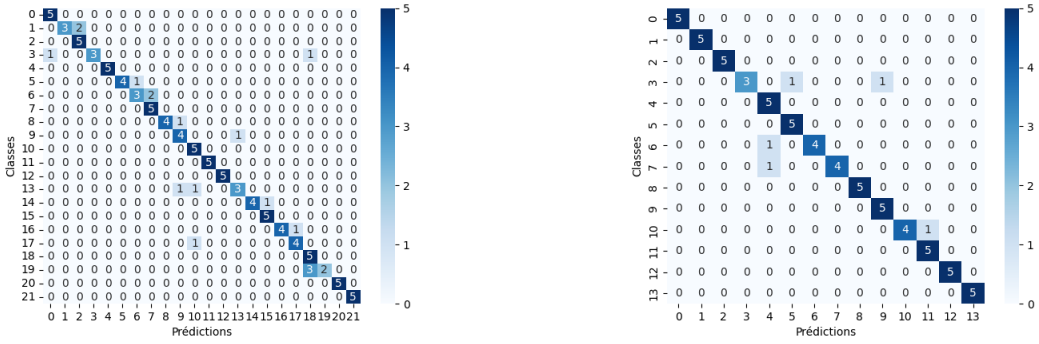

(a) Perte (b) Exactitude

FIGURE 5.3 – Évolution de la perte et de l'exactitude au fil des epochs pour la classification par genres d'insectes (CNN)




(a) GxABT (b) Africamuseum

FIGURE 5.4 – Matrice de confusion pour la classification par genres d'insectes pour le CNN de 3 couches de convolutions




(a) GxABT (b) Africamuseum

FIGURE 5.5 – Matrice de confusion pour la classification par genres d'insectes pour le CNN de 4 couches de convolutions

Les tableaux 5.1 et 5.2 reprennent les exactitudes moyennes sur 10 tirages différents.

| **CNN 3 couches** | GxABT | Africamuseum |
|---|---|---|
| Classification famille | 94% | 96.21% |
| Classification genre | 75.73% | 75% |

TABLE 5.1 – Exactitude (accuracy) pour le CNN de 3 couches de convolutions

| **CNN 4 couches** | GxABT | Africamuseum |
|---|---|---|
| Classification famille | 91.52% | 95.21% |
| Classification genre | 83.27% | 82.28% |

TABLE 5.2 – Exactitude (accuracy) pour le CNN de 4 couches de convolutions

### 5.2.3 Interprétation des Résultats

En comparant les Fig. 5.1 et 5.3, il est observable que l'exactitude monte moins abruptement au cours de l'entrainement de la classification par genres que pour celle par familles. Cela indique que le modèle éprouve une difficulté certaine à discriminer les genres plutôt que les familles.

Les deux matrices de confusion de la Fig. 5.2 témoignent d'une classification efficace des familles, avec une diagonale visible et foncée. L'exactitude moyenne Tab. 5.1 confirme cela. Les données de l'Africamuseum donnent une performance légèrement meilleure que celles de GxABT.

En ce qui concerne la classification par genres, l'exactitude baisse significativement pour les deux méthodes de numérisation. La disparité de profondeur entre les deux CNN devient cette fois-ci perceptible tant sur la matrice de confusion que sur l'exactitude moyenne. L'examen des matrices de confusion à la Fig. 5.4 révèle que des genres de mêmes familles peuvent se confondre pour la classification par le CNN de 3 couches de convolutions. Les indices des genres appartenant aux mêmes familles se succèdent, il est alors possible d'observer des irrégularités sur la diagonale, ce qui indique une confusion entre les genres d'une même famille. Par exemple, les classes 0 à 4 de la Fig. 5.4 (a) se confondent, elles appartiennent toutes à la famille des Zygaenidae. Idem à la Fig. 5.4 (b) pour les classes 6 à 8, qui appartiennent toutes à la famille des Cetoniinae. Cette observation ne se fait pas sur la matrice de confusion du CNN de 4 couches de convolutions. Un CNN assez profond peut donc classifier des insectes de la même famille, mais de genre différent avec une précision relativement acceptable.

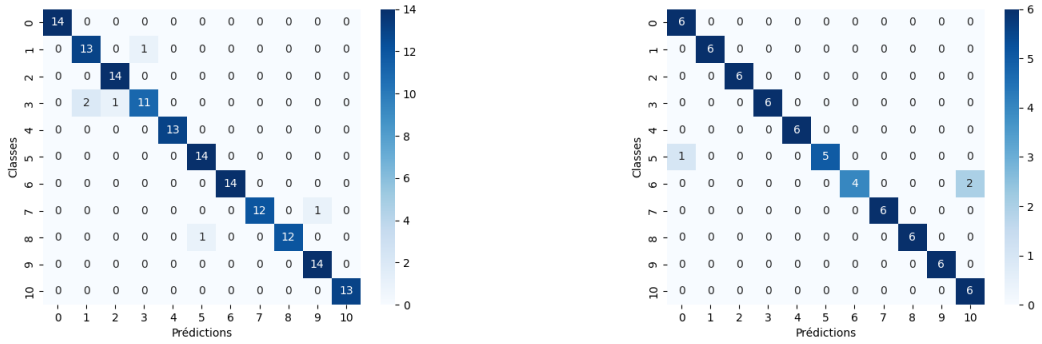
## 5.3 Classification par un SVM

### 5.3.1 Calcul de la précision

Dans `scikit-learn`, il n'est pas directement possible de visualiser les données pendant l'entraînement du SVM en utilisant les outils intégrés dans la bibliothèque. Seule l'exactitude moyenne et les matrices de confusions seront par conséquent interprétées.

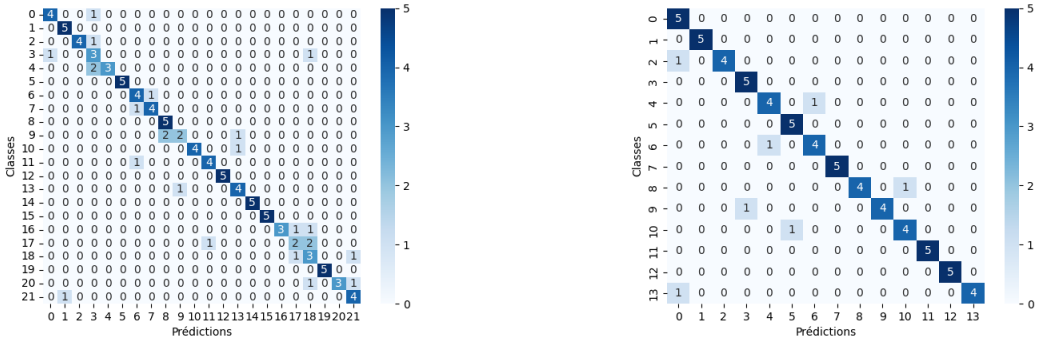
### 5.3.2 Visualisation des résultats

La Fig. 5.6 montre la matrice de confusion du modèle entrainé pour la classification par familles, pour les deux méthodes de numérisation (GxABT et Africamuseum) et la Fig. 5.3 pour la classification par genres. Le Tab. 5.3 reprend les exactitudes moyennes sur 10 tirages différents.




(a) GxABT (b) Africamuseum

FIGURE 5.6 – Matrice de confusion pour la classification de familles d'insectes différentes (SVM)




(a) GxABT (b) Africamuseum

FIGURE 5.7 – Matrice de confusion pour la classification de familles d'insectes différentes (SVM)

| | GxABT | Africamuseum |
|---|---|---|
| Classification famille | 95% | 95.76% |
| Classification genre | 75.45% | 86.71% |

TABLE 5.3 – Exactitude (accuracy) pour le SVM

### 5.3.3 Interprétation des résultats

Les performances du SVM se situent entre celles du CNN de 3 couches de convolutions et celles de celui de 4 couches de convolutions. Ces performances, comparables à celles d'un CNN, peut s'expliquer par le fait que les CNN requièrent généralement une grande quantité de données d'entraînement pour exploiter pleinement leur capacité à extraire des caractéristiques complexes. Lorsqu'il y a abondance de données, les CNN peuvent apprendre des représentations riches et généralisables, ce qui se traduit par des performances exceptionnelles. Les SVM quant à eux sont souvent considérés comme des modèles robustes avec une bonne capacité à généraliser à partir d'un petit ensemble de données. Ils sont moins susceptibles de surajuster (overfitting) dans des scénarios où les données d'entraînement sont limitées. Il est par conséquent intéressant d'utiliser un SVM dans ce contexte-là. Si les données sont plus abondantes, il est probable qu'un CNN atteigne des performances plus élevées.

# Conclusion

L'avènement des CNN a marqué une avancée significative, que ce soit dans le domaine de la détection d'objets, ou dans celui de la classification d'images. La capacité des CNN à apprendre des représentations hiérarchiques complexes, leur adaptabilité aux données non structurées et plus globalement leur performance ont rendu moins attrayantes, voire obsolètes, certaines approches répandues plus anciennes comme la détection d'objets par la méthode de Viola-Jones. Cependant, l'application spécifique des CNN à la détection et à la classification d'insectes dans des boîtes entomologiques présente une limitation cruciale : le manque de données annotées disponibles. Cette problématique résulte de la diversité importante des groupes taxonomiques d'insectes, entraînant une rareté de données annotées pour chaque groupe.

Ce mémoire propose deux approches pour contourner cette limitation. La première est d'utiliser une méthode qui ne nécessite pas de données d'entraînement. Cette approche présente de nombreux avantages, mais ne permet pas une automatisation importante du processus. Il est en effet nécessaire de calibrer de nombreux paramètres pour obtenir des résultats satisfaisants. La deuxième approche est d'utiliser cette même méthode, mais pour générer des données annotées. Ces données, pour autant qu'elles soient en quantité suffisante, peuvent par la suite entrainer un CNN, aussi bien pour la détection que pour la classification, et finalement être utilisées pour détecter ou classifier des insectes similaires d'autres boites entomologiques.

Les résultats obtenus par l'évaluation des performances de cette approche sont prometteurs, mais leur validation pour une échelle beaucoup plus grande que celle adoptée dans cette étude est nécessaire. L'utilisation d'une échelle plus importante permettrait également de réaliser la classification d'insectes pour un groupe taxonomique inférieur hiérarchiquement, et ainsi de déterminer si un classificateur est capable de distinguer des insectes de même genre, mais d'espèces différentes.

Pour finir, il est important de garder à l'esprit que la méthode de numérisation influence grandement la précision de la détection et de la classification. Il est préférable d'utiliser une image unique de bonne qualité pour obtenir les meilleurs résultats

# Bibliographie


[1] G. G. E. Scudder, « The Importance of Insects », en, in *Insect Biodiversity*, R. G. Foottit et P. H. Adler, éd., 1re éd., Wiley, août 2017, p. 9-43, isbn : 9781118945537 9781118945568. doi : 10.1002/9781118945568.ch2. adresse : https://onlinelibrary.wiley.com/doi/10.1002/9781118945568.ch2 (visité le 04/12/2023).

[2] M. Heerlien, J. Van Leusen, S. Schnörr, S. De Jong-Kole, N. Raes et K. Van Hulsen, « The Natural History Production Line: An Industrial Approach to the Digitization of Scientific Collections », *Journal on Computing and Cultural Heritage*, t. 8, no 1, 3:1-3:11, fév. 2015, issn : 1556-4673. doi : 10.1145/2644822. adresse : https://dl.acm.org/doi/10.1145/2644822 (visité le 04/12/2023).

[3] M. Hereld, N. J. Ferrier, N. Agarwal et P. Sierwald, « Designing a High-Throughput Pipeline for Digitizing Pinned Insects », in *2017 IEEE 13th International Conference on e-Science (e-Science)*, oct. 2017, p. 542-550. doi : 10.1109/eScience.2017.88. adresse : https://ieeexplore.ieee.org/abstract/document/8109193 (visité le 04/12/2023).

[4] Z. Wu, J. Kahanpää, P. Sihvonen, A. Koivunen et H. Saarenmaa, « Automated Methods in Digitisation of Pinned Insects », en, *Biodiversity Information Science and Standards*, t. 3, e38260, juill. 2019, issn : 2535-0897. doi : 10.3897/biss.3.38260. adresse : https://biss.pensoft.net/article/38260/ (visité le 04/12/2023).

[5] C. Dietrich, J. Hart, D. Raila et al., « InvertNet: a new paradigm for digital access to invertebrate collections », *ZooKeys*, no 209, p. 165-181, juill. 2012, issn : 1313-2989. doi : 10.3897/zookeys.209.3571. adresse : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3406474/ (visité le 04/12/2023).

[6] E. H. Frédéric Francis, « Entomologie faunistique - Faunistic Entomology — PoPuPS », fr, *Entomologie faunistique - Faunistic Entomology*, issn : 2030-6318. adresse : https://popups.uliege.be/2030-6318/index.php?id=2311 (visité le 06/01/2024).

[7] Wikipedia, *Focus stacking — Wikipedia, The Free Encyclopedia*, http://fr.wikipedia.org/w/index.php?title=Focus%20stacking&oldid=208099332, 2023. (visité le 04/12/2023).

[8] *Home*, en-US. adresse : https://opencv.org/ (visité le 12/12/2023).

[9] Wikipedia, *HSL and HSV — Wikipedia, The Free Encyclopedia*, http://en.wikipedia.org/w/index.php?title=HSL%20and%20HSV&oldid=1189041773, 2023. (visité le 12/12/2023).

[10] *OpenCV: Smoothing Images*. adresse : https://docs.opencv.org/3.4/dc/dd3/tutorial_gausian_median_blur_bilateral_filter.html (visité le 13/12/2023).

[11] *Python OpenCV — cv2.erode() method*, en-US, nov. 2019. adresse : https://www.geeksforgeeks.org/python-opencv-cv2-erode-method/ (visité le 13/12/2023).

[12] Z. Xu, X. Baojie et W. Guoxin, « Canny edge detection based on Open CV », in *2017 13th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)*, oct. 2017, p. 53-56. doi : 10.1109/ICEMI.2017.8265710. adresse : https://ieeexplore.ieee.org/abstract/document/8265710 (visité le 14/12/2023).

[13] *OpenCV: Contours : Getting Started*. adresse : https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html (visité le 14/12/2023).

[14] Wikipedia, *Méthode de Viola et Jones — Wikipedia, The Free Encyclopedia*, http://fr.wikipedia.org/w/index.php?title=M%C3%A9thode%20de%20Viola%20et%20Jones&oldid=201045133, 2024. (visité le 20/12/2023).

[15] Y.-Q. Wang, « An Analysis of the Viola-Jones Face Detection Algorithm », en, *Image Processing On Line*, t. 4, p. 128-148, juin 2014, ISSN : 2105-1232. DOI : 10.5201/ipol.2014.104. adresse : https://www.ipol.im/pub/art/2014/104 (visité le 20/12/2023).

[16] T. Mita, T. Kaneko et O. Hori, « Joint Haar-like features for face detection », in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, ISSN: 2380-7504, t. 2, oct. 2005, 1619-1626 Vol. 2. DOI : 10.1109/ICCV.2005.129. adresse : https://ieeexplore.ieee.org/abstract/document/1544911 (visité le 20/12/2023).

[17] *OpenCV: Cascade Classifier Training*. adresse : https://docs.opencv.org/4.x/dc/d88/tutorial_traincascade.html (visité le 20/12/2023).

[18] *Cascade Trainer GUI*, en-US. adresse : https://amin-ahmadi.com/cascade-trainer-gui/ (visité le 20/12/2023).

[19] *YOLO by Ultralytics*, original-date: 2022-09-11T16:39:45Z, jan. 2023. adresse : https://github.com/ultralytics/ultralytics (visité le 24/12/2023).

[20] A. Kumar et S. Srivastava, « Object Detection System Based on Convolution Neural Networks Using Single Shot Multi-Box Detector », *Procedia Computer Science*, Third International Conference on Computing and Network Communications (CoCoNet'19), t. 171, p. 2610-2617, jan. 2020, ISSN : 1877-0509. DOI : 10.1016/j.procs.2020.04.283. adresse : https://www.sciencedirect.com/science/article/pii/S187705092031276X (visité le 24/12/2023).

[21] J. Du, « Understanding of Object Detection Based on CNN Family and YOLO », *Journal of Physics: Conference Series*, t. 1004, p. 012 029, avr. 2018, ISSN : 1742-6588, 1742-6596. DOI : 10.1088/1742-6596/1004/1/012029. adresse : https://iopscience.iop.org/article/10.1088/1742-6596/1004/1/012029 (visité le 24/12/2023).

[22] Ultralytics, *Accueil*, fr. adresse : https://docs.ultralytics.com/fr/ (visité le 24/12/2023).

[23] AndreyGermanovDev, *How to Detect Objects in Images Using the YOLOv8 Neural Network*, en, mai 2023. adresse : https://www.freecodecamp.org/news/how-to-detect-objects-in-images-using-yolov8/ (visité le 24/12/2023).

[24] P. Jiang, D. Ergu, F. Liu, Y. Cai et B. Ma, « A Review of Yolo Algorithm Developments », *Procedia Computer Science*, The 8th International Conference on Information Technology and Quantitative Management (ITQM 2020 & 2021): Developing Global Digital Economy after COVID-19, t. 199, p. 1066-1073, jan. 2022, ISSN : 1877-0509. DOI : 10.1016/j.procs.2022.01.135. adresse : https://www.sciencedirect.com/science/article/pii/S1877050922001363 (visité le 24/12/2023).

[25] M. Chablani, *YOLO — You only look once, real time object detection explained*, en, août 2023. adresse : https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006 (visité le 25/12/2023).

[26] G. Oreski, « YOLO*C — Adding context improves YOLO performance », *Neurocomputing*, t. 555, p. 126 655, oct. 2023, ISSN : 0925-2312. DOI : 10.1016/j.neucom.2023.126655. adresse : https://www.sciencedirect.com/science/article/pii/S0925231223007786 (visité le 25/12/2023).

[27] W. Zhang, P. Tang et L. Zhao, « Remote Sensing Image Scene Classification Using CNN-CapsNet », en, *Remote Sensing*, t. 11, n° 5, p. 494, jan. 2019, ISSN : 2072-4292. DOI : 10.3390/rs11050494. adresse : https://www.mdpi.com/2072-4292/11/5/494 (visité le 27/12/2023).

[28] C. Martineau, D. Conte, R. Raveaux, I. Arnault, D. Munier et G. Venturini, « A survey on image-based insect classification », *Pattern Recognition*, t. 65, p. 273-284, mai 2017, ISSN : 0031-3203. DOI : 10.1016/j.patcog.2016.12.020. adresse : https://www.sciencedirect.com/science/article/pii/S0031320316304411 (visité le 26/12/2023).

[29] X. Lei, H. Pan et X. Huang, « A Dilated CNN Model for Image Classification », *IEEE Access*, t. 7, p. 124 087-124 095, 2019, ISSN : 2169-3536. DOI : 10.1109/ACCESS.2019.2927169. adresse : https://ieeexplore.ieee.org/abstract/document/8756165 (visité le 27/12/2023).

[30] *Keras — TensorFlow Core*, fr. adresse : https://www.tensorflow.org/guide/keras?hl=fr (visité le 27/12/2023).

[31] Wikipedia, *Machine à vecteurs de support — Wikipedia, The Free Encyclopedia*, http://fr.wikipedia.org/w/index.php?title=Machine%20%C3%A0%20vecteurs%20de%20support&oldid=209205704, 2023. (visité le 28/12/2023).

[32] M. A. Chandra et S. S. Bedi, « Survey on SVM and their application in imageclassification », en, *International Journal of Information Technology*, t. 13, n° 5, p. 1-11, oct. 2021, issn : 2511-2112. doi : 10.1007/s41870-017-0080-1. adresse : https://doi.org/10.1007/s41870-017-0080-1 (visité le 28/12/2023).

[33] Ultralytics, *Détection*, fr. adresse : https://docs.ultralytics.com/fr/tasks/detect (visité le 06/01/2024).

# Annexes

## A Code

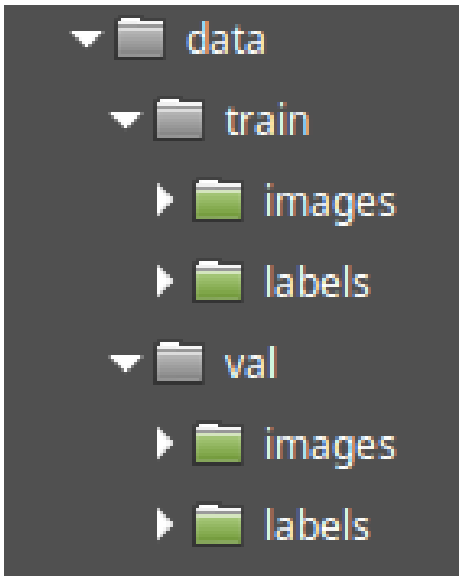
Tous les codes utilisés lors de ce TFE sont disponibles à l'adresse suivante :

`https://github.com/gb9900/TFE.git`

## B Performances de YOLOv8 en fonction de la taille du modèle [33]

| Modèle | Taille (pixels) | $mAP^{val}$ 50-95 | Vitesse CPU ONNX (ms) | Vitesse A100 TensorRT (ms) | Paramètres (M) | FLOPs (B) |
|---|---|---|---|---|---|---|
| YOLOv8n | 640 | 37.3 | 80.4 | 0.99 | 3.2 | 8.7 |
| YOLOv8s | 640 | 44.9 | 128.4 | 1.20 | 11.2 | 28.6 |
| YOLOv8m | 640 | 50.2 | 234.7 | 1.83 | 25.9 | 78.9 |
| YOLOv8l | 640 | 52.9 | 375.2 | 2.39 | 43.7 | 165.2 |
| YOLOv8x | 640 | 53.9 | 479.1 | 3.53 | 68.2 | 257.8 |

## C Structure de l'arborescence des dossiers pour l'utilisation du modèle YOLO personnalisé



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