université de BORDEAUX

Rapport de projet de fin d'études
Bees for life

Auteurs
Louis-Gabriel Barrère
Sébastien Granger
Mehdi Besse

Contents

1 Introduction

1.1 Qui sommes-nous ?

Nous sommes un groupe de 3 étudiants en **Master 2 informatique pour l'image & le son** (MIIS). Lors de cette dernière année universitaire, un **Projet de Fin d'Etudes** (PFE) nous est confié. Ce PFE s'étend sur une durée de 2 mois et un sujet est à choisir parmi une liste proposée. Nous avons décidé de nous investir dans le projet intitulé **Bees for Life**. Nous avons été encadrés par Mr. Desbarats tout au long du projet dans le cadre des cours de **Projet de Fin d'Etudes** et de **Méthode et Outils pour la Conduite de Projets Informatiques** (MOCPI). Nous avons également été mis en relation avec Mr. Willaert Lionel, client membre de Bees for Life, afin de mieux établir ses attentes concernant le projet.

1.2 Contexte

Bees for Life est à la fois le nom du projet et celui d'une équipe de passionnés de la protection de l'abeille et de l'usage de nouvelles technologies appliquées à la protection. Cette initiative s'est donc concrétisée en une plateforme interactive qui permet à des particuliers, des professionnels, des associations, des collectivités de signaler la présence d'un nid de frelons asiatiques sur le territoire.

Le projet de Bees for Life a été créé dans un but de protection face à une menace préoccupante, celle-ci est décrite en détail dans la section **Problématique**. Ce projet fait suite à un travail déjà réalisé, d'après l'article [1], qui implémente un réseau de neurones et expérimente plusieurs approches comme M-RCNN, YOLO et un Ensemble model combinant deux réseaux pour travailler simultanément.

A partir des résultats du précédent projet, la suite de ce projet utilise le Mask-RCNN et l'Ensemble model comme base de deep-learning afin de localiser les nids de frelons en fonction des images couleur et images infrarouges ou vidéos. L'objectif est d'améliorer le projet initial en renforçant le réseau de neurones et en revoyant l'interface utilisateur ainsi que les fonctionnalités de l'application (se référer à [1] pour plus de détails).

1.3 Problématique

Avec le réchauffement climatique, la présence de nids de frelons asiatiques se fait de plus en plus ressentir sur le territoire français et à plus grande échelle, sur le territoire Européen. Les frelons asiatiques sont à la fois agressifs et destructeurs. En effet ils peuvent très facilement détruire des ruches d'abeilles et causer des dommages conséquents sur d'autres espèces, jusqu'à être mortel, même pour l'humain.

C'est pourquoi des opérations sont lancées afin de repérer ces nids, si possible le plus tôt en saison, puis pour détruire ces nids dans le but de limiter le plus possible les dégâts causés par ces frelons.

1.4 Sujet officiel

L'objectif principal du projet est de détecter des nids de frelons asiatiques à partir d'images/vidéos (spectre visible/infrarouge) issues de drones.

Afin d'y parvenir, des méthodes d'apprentissage profond (Mots clés : Machine learning, deep learning) sont employées. Le but de ce projet est de faire évoluer une application de détection de nids de frelons asiatique in situ à partir de données issues de drones.

La détection se fait en phase de post-traitement des données. Initialement, cette application utilise un réseau conjoint de Mask-RCNNs pour segmenter les images et identifier les images contenant un nid (conjointement dans l'infrarouge et le spectre visible).

Le but est donc :

- d'adapter les réseaux aux nouveaux types d'acquisition actuellement sur le drone :
  - Proposer une méthode d'analyse et d'affichagede la température à partir des images infrarouges.
  - Gérer les nouveaux types de données obtenus (TIFF, MOV).
  - Proposer une architecture pour l'analyse des nuages de points (générés par un LiDAR). (si le temps le permet).

- de modifier la GUI de l'application pour y ajouter l'affichage d'informations permettant l'analyse (notamment pour faciliter la localisation du nid).
- de proposer une solution pour permettre à l'application d'apprendre au fur et à mesure des acquisitions/analyses (si le temps le permet).

2 Etat de l'art

De nombreux travaux cités dans [1] sont repris pour ce projet, mais d'autres sources ont été écartées car elles ne font plus l'objet d'étude du projet. Il est donc possible de se référer à ce document pour en savoir plus. Voici les travaux réutilisés pour ce projet ainsi que la raison à chaque document expliquant en quoi il est susceptible d'intérêt spécifiquement pour ce projet :

- [2] Cet article traite de l'activité colonniale des frelons asiatiques. Il permet dans notre cas de mieux comprendre leur quotidien et donc de s'adapter sur les cycles les plus intéressants pour les observer. Il sera plus simple de les localiser selon la température de leur nid sous certaines conditions par exemple.
- [3] Un système de radar permettant de traquer les frelons asiatiques en vol a déjà été déployé, c'est ce qui est décrit dans ce document. Il s'agit ici d'un plus permettant de mieux comprendre le quotidien des frelons en plus d'être une solution pour les localiser.
- [4] Le but étant d'extraire des données en fonction de l'ensemble model (en plus du Mask-RCNN), cet article permet d'obtenir une analyse des performances qui utilise les deux méthodes (fusion). Bien que ceci porte sur des systèmes biométriques unimodaux d'iris et d'empreintes digitales, les scores décrits dans l'article source [1, p. 4] utilisent un procédé qui s'en approche.
- [5] L'optimisation et la vitesse de calcul sont des enjeux majeurs pour tout programme en général, en particulier dans une architecture utilisant un réseau de neurones. C'est pourquoi ce document permet de comprendre quelles opérations peuvent être plus efficaces pour accélérer certains processus, voire de faire des compromis.

- [6] Le projet se base sur le mask-RCNN, il est donc important de savoir sur quoi il porte et comment peut être faite la détection d'éléments dans les images via les réseaux de neurones. L'article explique aussi qu'il s'agit d'une extension du faster RCNN, c'est pourquoi les deux articles ne peuvent se soustraire à une analyse pour ce projet.

- [7] Ce papier explique en détail plusieurs aspects important dans le réseau de neurones profond : loss, accuracy, layers... il est donc utile pour comprendre ce qui est fait dans l'architecture et afficher des graphes de résultats en fonction des entraînements et tests menés

- [8] L'article donne un aperçu des concéquences médicales que peuvent provoquer les frelons asiatiques, ce qui permet de se remettre dans le contexte de la situation qui pousse les études sur les frelons et en quoi il est important de les traiter.

- [9] Ce papier porte son étude sur des systèmes de détection et reconnaissance à l'aide d'un réseau de neurone artificiel et de drones. Le logiciel du prjet prend en entrée des images de drones et utilise un réseau de neurones. Il est donc nécessaire de voir ce qui a déjà été fait avec ces mêmes procédés.

- [10] Cet article rejoint [4], ce qui donne un complément d'informations sur les méthodes de fusion.

- [11] Ce document propose de traquer des frelons en vol afin de détecter leurs nids lorsqu'ils y retournent. La radio-télémétrie est ainsi utilisée et rejoint ce dont parle l'article [3] sur les systèmes de traques.

En plus des précédents articles déjà utilisés par la référence principale [1], d'autres articles peuvent aider à la compréhension de certaines méthodes ou améliorer certains aspects :

- [12] Les nids peuvent être, entre autres, détectés en fonction de leurs températures et produisent des images infrarouges à intégrer au réseau de neurones. L'architecture actuelle se sert d'images FLIR et d'images couleur, ainsi cet article peut être utile pour les images FLIR bien qu'il soit axé sur des problèmes de détection de cancer, certaines approches peuvent être réutilisées dans la détection de nids par source termique.

- [13] Le papier regroupe les méthodes de deep learning employées à travers 210 documents de recherches récents. Ces analyses sont importantes car elles permettent un gain de temps considérable sur les recherches dans le domaine du deep learning.

- [14] Cet article récent présente, en complément, des méthodes de deep learning et de détection d'objets dans des images inrarouge. Ces méthodes faisant partie intégrante du projet, il est nécessaire d'avoir les informations de cet article pour le réseau de neurones du projet.

- [15] Les nids de frelon peuvent être difficiles à détecter à cause de leur petite taille sur certaines images, cet article a donc un intérêt sur ce type de détection car il traite des détection de petits objets sur des images infrarouge.

- [16] Ce papier se contentre sur les activités des frelons asiatiques, mais en particulier en hiver. Ceci permet d'avoir plus d'informations sur les frelons en périodes défavorables.

- [17] Pour éviter la répartition rapide des frelons asiatiques, il est possible de porter une analyse sur les emplacements et l'expension des frelons sur le territoire dans le but d'anticiper les nouveaux emplacements des nids. Cet article fait état de l'invasion des frelons, ce qui permet d'aborder le sujet avec un autre point de vue.

2.1 Résultats préliminaires

Une première étude consistait à suivre des frelons "ouvriers" ("worker") en leur attachant des balises et en vérifiant qu'ils pouvaient voler. Ils se sont aperçus que les méthodes de deep learning n'avaient pas été suffisamment expérimentées pour ce problème.

Le tableau suivant pésente les résultats obtenus suite aux nombreux tests réalisés sur les réseaux Mask R-CNN, YOLO, et Ensemble model (extrait de [1]) :

Figure 1: Résultats préliminaires

RESULTS

Model	GT	TPs	FPs	mAP	LAMR
YOLO FLIR	209	93	140	34.74%	0.71
YOLO Color	523	158	345	22.03%	0.80
Mask R-CNN FLIR	202	191	110	92.95%	0.11
Mask R-CNN Color	203	192	95	92.22%	0.12
Mask R-CNN Ensemble Model	**200**	**186**	**6**	**93.00%**	**0.07**

- En ce qui concerne le réseau YOLO (You Only Look Once), nous pouvons constater que ça n'a pas fonctionné convenablement, avec une mAP (Mean Average Precision) de 34.74 % pour les FLIR et seulement 22.03 % pour les images couleurs.
  La géométrie et la forme des nids changeante d'une image à une autre explique la difficulté qu'a eu YOLO à réaliser de bonnes détections.

- Les Mask R-CNN ont été de loin plus efficaces avec donc une mAP bien plus élevée à savoir 92.95% pour les images FLIR et 92.22% pour les images couleur et un LAMR (Log Average Miss Rate : correspond à la proportion de prédictions ratées sur le nombre d'objets présents) bien inférieur.
  Un bémol des Mask R-CNN s'observe par rapport aux images FLIR qui ont généré un nombre plus important de TP (True Positive) ce qui est une bonne chose, mais le réseau a aussi généré un nombre plus grand de FP (False Positive).
  Une amélioration va donc pouvoir être réalisée en utilisant l'Ensemble model pour enlever les FP et améliorer les résultats.

- Effectivement, l'Ensemble model réduit considérablement le nombre de FP et augmente donc encore un peu la valeur de mAP.

A partir des résultats du précédent projet, avec un taux de réussite de 93%, la suite de ce projet va donc utiliser le Mask R-CNN et l'Ensemble model comme base de deep-learning afin de localiser les nids de frelons en fonction des images couleurs et des images infrarouges ou vidéos.
L'objectif est d'améliorer le projet initial en renforçant le réseau de neurones et en revoyant l'interface utilisateur ainsi que les fonctionnalités de l'application (se référer à [1] pour plus de détails).

3 User stories

- Proposer des méthodes d'analyse de la température des images infrarouges : il s'agit d'adapter l'implémentation initiale avec de nouvelles méthodes dans le but de rendre l'analyse de la température plus efficace. De plus, rendre le visuel sur les nids de frelons plus détaillé et coloré est intéressant pour le projet.

- Faire un apprentissage avec de nouvelles images : un nouveau set de données est fourni, il faut les donner en entrée au réseau de neurones

- Simplifier l'utilisation du logiciel pour un usage moins axé "développeur" : l'emploi de commandes dans une console ou un terminal n'est pas ergonomique, un objectif est de simplifier la mise en place et le lancement du logiciel.

- Afficher plus de détails sur les informations de chaque image : certaines données essentielles peuvent être extraites des images, par exemple les coordonnées GPS pour en savoir plus sur la localisation des nids.

4 Les besoins

4.1 Besoins fonctionnels

- Afficher les informations de l'image (position GPS, nom, taille, date, format, ...)

- Afficher l'image flir et l'image couleur en meme temps après avoir cliqué sur "visualised synchronized"
- Analyser et afficher la température à partir d'images infrarouges sur des images TIF (détecter si un nid est actif ou peu actif)
- Afficher l'image TIF en nuance de couleur
- Reconnaître des nids de frelons à partir de vidéos au format MOV (ajouter les entrées/sorties)
- Reconnaître des nids de frelons à partir d'images au format TIF (ajouter les entrées/sorties)
- Prendre en compte des données LiDAR (si le temps le permet)
- Apprentissage au fur et à mesure (si le temps le permet)

4.2 Besoins non-fonctionnels

Les besoins non-fonctionnels sont difficiles à identifier pour ce projet, voici donc le seul réel besoin non-fonctionnel prévu :
L'interface doit permettre d'accéder au résultat de l'analyse facilement (ergonomie).

5 Choix logiciels

Le projet utilise principalement Python/Tensorflow, Keras/Anaconda. Ce projet étant la suite directe de celui décrit dans l'article de conférence [1], la technologie utilisée reste la même. De plus, la majeure partie du travail consiste à améliorer l'existant.

L'architecture doit donc se baser sur les mêmes bibliothèques et technologies à moins que le sujet du projet soit de basculer vers une nouvelle architecture. Là n'étant pas pas le sujet, l'architecture globale ne change pas.

6 Architecture

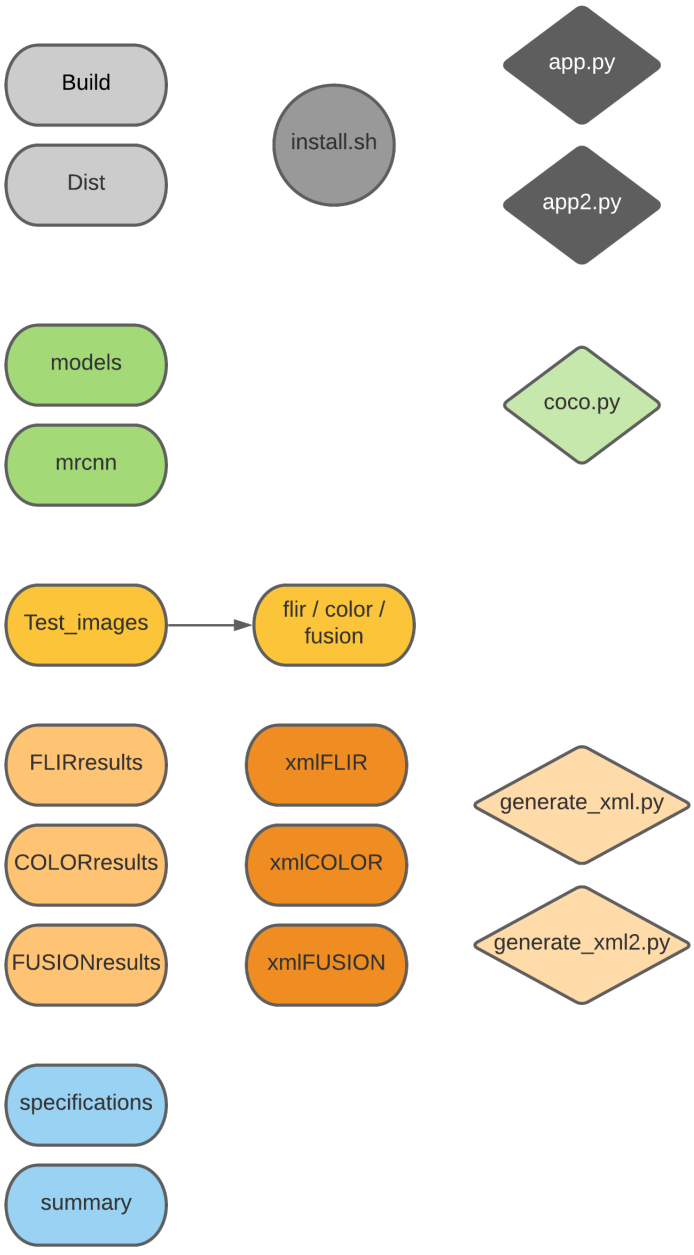
Certains dossiers ou fichiers étant uniquement utilisés comme exemples, voire inutilisés par le programme, une description plus courte est donnée pour ceux qui portent le plus d'intérêt au projet.

- build et dist : permettent surtout de déployer une version "sans terminal" du logiciel, non fonctionnel
- COLORresult, FLIRresults et FUSIONresults : contiennent respectivement les images COLOR, FLIR et FUSION dont des nids sont détectés par réseau de neurones
- models : modèles d'entraînement pour le réseau de neurones
- mrcnn : modules nécessaire au fonctionnement du réseau de neurones
- specifications et summary : graphiques semblant faire état de l'évolution des précisions et échecs du réseau de neurones sur les différents types d'images et tableaux de données
- src : semble contenir des exemples de code permettant de créer des applications ou modules
- Test_images : images à balancer dans le réseau de neurones, à priori pour détecter les éventuels nids de frelons (ou insectes parasites et destructeurs)
- xmlCOLOR, xmlFLIR et xmlFUSION : contiennent respectivement les images COLOR, FLIR et FUSION affichées avec le format xml
- app.py et app2.py : permettent de gérer l'interface de l'application (affichage des boutons, textes, ...), app.py est obsolète car il a servi de base pour app2.py, une version améliorée qui gère les deux types d'images COLOR et FLIR sans problème
- coco.py : code python utilisant l'architecture R-CNN (site de référence coco : https://cocodataset.org/#explore?id=), permet la gestion de dataset
- generate_xml.py et generate_xml2.py : codes python permettant d'écrire un fichier .xml à partir d'une image ou d'un path et d'un name

- install.sh : script d'installation Linux et Mac nécessaire au bon fonctionnement de l'application

Le reste des fichiers ou dossiers ne sont pas nécessaires au logiciel ou n'apportent pas plus de connaissances sur l'architecture. Voici un schéma plus visuel de cette même architecture avec les dépendances :

Figure 2: Architecture Bees for Life



7 Planning prévisionnel

Prévisionnel	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 5	Sem. 6
Louis-Gabriel	Etat de l'art	Analyse de l'interface	Mise en place du squelette de l'interface	Rendre l'application fonctionnelle	Différencier nids actifs/non-actifs	Idem
Sebastien	Analyse du réseau de neurones	Analyse d'images FLIR	Modifier/entraîner le réseau sur des FLIR	Lancer la detection sur des FLIR	Apprentissage au fur et à mesure	Idem
Mehdi	Analyse du réseau de neurones	Anayser format MOV	Lancer la détection sur des vidéos	Enregistrer les résultats des analyses	Analyse des nuages de points	Idem
					Rapport	Rapport

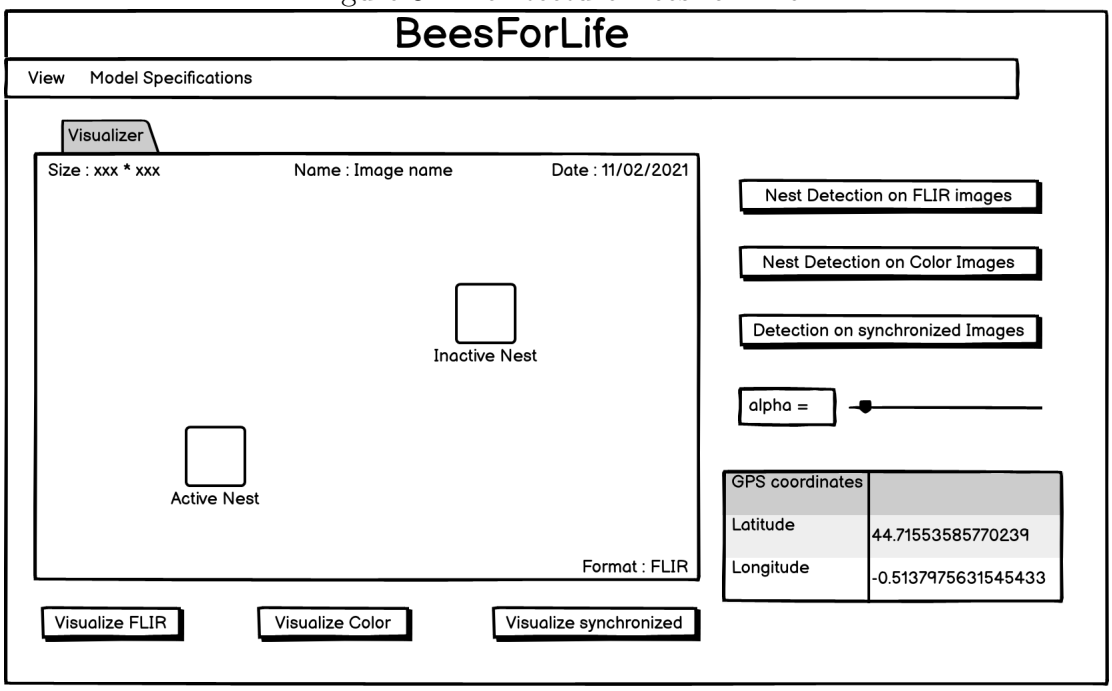
L'objectif des deux premières semaines est de faire état de l'existant et de prendre en main le précédent projet dans le but de l'améliorer plus aisément. A partir de la semaine 3, il faut au moins le "coeur" de la modification attendue, d'où la "mise en place" et la "modification". Dès la semaine 4, il était prévu d'avoir l'application "terminée", car les semaines suivantes étaient prévues pour des ajustements ou des tâches de plus basse priorité (le "si le temps le permet" décrit dans la section **Sujet officiel**) et la rédaction/relecture de ce rapport, bien évidemment pour tous les membres du groupe.

8 Réalisation

8.1 Interface Utilisateur & robustesse

Une maquette a été dessinée avant de s'intéresser à l'amélioration de l'interface utilisateur, elle inclut un tableau d'informations comme fonctionnalité complémentaire. Cette maquette pouvait être modifiée en fonction de nouvelles attentes, en voici un aperçu :

Figure 3: Architecture Bees for Life



C'est à partir de cette base que le tableau d'informations a été implémenté, néanmoins les informations **Size**, **date** et **Format** n'ont pas été incluses dans **Visualizer** par manque d'intérêt du client. Seul le nom de l'image a été demandé dans **Visualizer**, c'est pourquoi il a été implémenté.

Les informations les plus importantes à afficher, à la demande du client, dans le tableau d'informations sont en effet la **latitude** et la **longitude**, le tableau a bien été implémenté avec en plus **l'altitude**.

Un problème lié au nom de certaines images données en entrée pouvait provoquer un crash lors de leur lecture. Ce problème est dû à un caractère spécial qui semble apparaître à cause du matériel qui permet de prendre les potentiels nids en photo. La solution initiale était de supprimer le caractère gênant "à la main", ce qui n'est pas ergonomique. C'est pourquoi une fonction a été implémentée pour traiter ce défaut. Le problème ne semble intervenir que sous Linux et Mac, car Windows modifie probablement les noms des images

lorsqu'ils sont corrompus, ce qui empêche l'apparition du problème.

A la demande du client, plusieurs approche ont été testées afin de retirer la console pour l'exécution du programme et préférer une solution plus ergonomique telle qu'un exécutable. Malheureusement, chaque approche échoue pour des raisons différentes, mais probablement de façon générale à cause de l'environnement virtuel de **Anaconda**. La première méthode était de faire un script Python **setup.py** dans le but de relier toutes les dépendances, faire l'installation et générer un exécutable. La deuxième méthode était d'utiliser **Pyinstaller**, un package externe permettant de faire automatiquement la lecture et le lien entre toutes les dépendances et de créer les dossiers build et dist contenant le résultat des opérations dont l'exécutable. La dernière solution était de faire des scripts pour remplacer l'installation et l'exécutable : ils se contentent de lancer les lignes de commandes qui installent toutes les dépendances et lancent le programme sans avoir à saisir les lignes soi-même. Cette autre opération n'aboutie pas à cause des restrictions d'exécution : sous Windows la commande **Set-ExecutionPolicy Unrestricted** doit nécessairement être entrée dans la console afin d'autoriser l'exécution de TOUS les scripts, ce qui peut être dangereux si des opérations de sécurités ne sont pas appliquées et sous Linux et Mac un chmod est nécessaire pour autoriser un script spécifique de s'exécuter, par exemple : **chmod u+x launch.sh**. Les scripts **launch.sh** (pour Linux et Mac) et **launch.ps1** (pour Windows) ont été créés à l'occasion.

Une dernière méthode a été essayée, mais bien qu'il semblait plus efficace, un autre problème est apparu. La méthode consistait à faire un nouveau script Python très court qui se contente de lancer les scripts sh en outrepassant les restrictions et d'en faire un exécutable. La conversion en exécutable fut un succès, mais au lieu d'exécuter le script dans le répertoire courant le Mac a réinitialisé le chemin d'accès à la racine (donc /), créant ainsi un problème plus difficile à résoudre : il faut retrouver le chemin d'accès au script. Le seul moyen "fiable" était d'utiliser la commande find et de l'inclure au script Python, mais le résultat n'a malheureusement pas abouti. Le nom du script est test.py (donc l'exécutable sous Mac est test).

8.2 Le réseau de neurones Mask R-CNN

Le principe d'un framework Mask R-CNN est de prendre une image en entrée et de fournir en sortie une localisation des boîtes englobantes ainsi que leurs labels correspondants.
Quatre étapes sont définies :

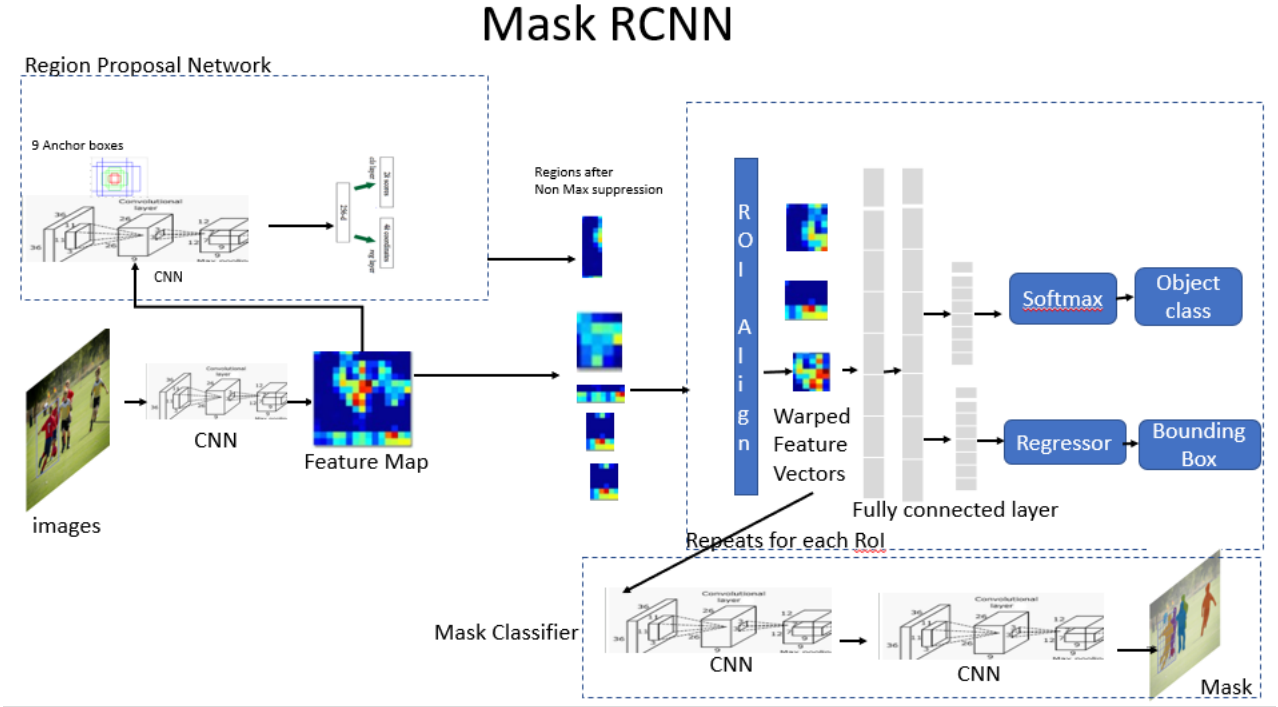
- Proposition de régions
- Calcul des CNN features
- Décision binaire basée sur un Support Vector Machine (SVM)
- Segmentation des niveaux de pixel

R-CNN est donc un SVM modifié pour réaliser de la détection d'objets.
Le modèle s'entraîne sur 2 classes, à savoir "nid" et "background" (BG).
R-CNN utilise un convolutional neural network appelé Resnet101 (pour 101 couches de neurones), implémenté avec TensorFlow.

Le réseau calcule les boîtes englobantes à partir des masques d'entrée.
Pour préserver le bon ratio des images, un zéro padding est effectué sur les images après qu'elles ont été redimensionnées.
Le réseau stocke uniqument les pixels du masque à l'intérieur des boîtes englobantes pour économiser de la mémoire et augmenter la vitesse d'entraînement (le masque peut aussi être redimensionné à une plus petite taille pour plus de vitesse).
La RPN (Region Proposal Network) se charge quant à elle de calculer les anchors.
Ensuite, pour une image de test, la RPN exécute le classifieur sur les anchors des images pour donner un score aux objets trouvés (donc les nids).
Finalement, le réseau va dire si une anchor contient un nid ou non et va attribuer une probabilité aux anchors appelées positives.
Le masque pourra ensuite être généré à partir des boîtes englobantes.

Voici une illustration [18] du Mask R-CNN :

Figure 4: Mask R-CNN



Mask R-CNN ayant tellement surperformé l'algorithme YOLO (une autre approche tentée par Bees for Life), c'est celui ci qui est donc choisi pour créer l'Ensemble model.

Comme présenté plus haut, l'Ensemble model va être conceptualisé en se basant sur le Mask R-CNN qui utilise les modèles Flir et color.
Le principe consiste à s'aider du score de plusieurs features pour essayer d'améliorer encore plus la précision du réseau.
Pus précisemment, chaque model individuel, donc le Flir model et le color model, va nous donner en sortie les boîtes englobantes, un masque, et un score de confiance attribué aux objets trouvés. Les deux scores calculés par les deux modèles vont donc être combinés pour obtenir un score final.
Cette méthode permet donc d'améliorer la précision du réseau assez significativement: par exemple, on sait que le Flir model va avoir tendance à générer un grand nombre de TP (True Positive) mais aussi un plus grand nombre de

FP (False Positive).
Le but va être donc de réduire ce nombre de FP et c'est le color model qui va permettre ceci.
De plus, si un nid est loupé mais le Flir model mais est trouvé par le color model, le nid trouvé sera bien considéré comme un TP ce qui est très avantageux.

On comprend donc facilement que l'Ensemble model soit encore un peu meilleur que le Mask R-CNN.

8.3 Prise en compte des images au format tiff

Une des caméras infrarouges dont est équipé un drone permet d'avoir des images au format tiff, codées sur 16 bits, contenant donc plus d'informations. Ainsi dans l'espoir d'avoir de meilleurs résultats nous devions prendre en compte ce type d'images dans l'application. Pour ce faire, nous devions tout d'abord convertir l'image en nuances de couleurs, puis paramétrer le réseau de neurones Mask R-CNN de manière à ce qu'il puisse s'entraîner à trouver les nids de frelons sur ces images. Étant donné qu'il n'y a pas de nids de frelons a cette époque de l'année, nous n'avons pas pu entraîner nos images tiff, cependant des explications seront données sur ce réseau de neurones ce qui permettra aux futurs programmeurs de facilement interpréter ces images.

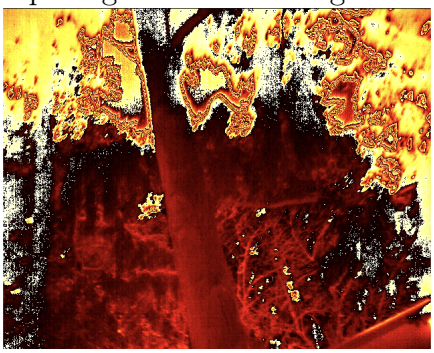
8.3.1 Conversion des images tiff en nuances de couleurs

La bibliotheque skimage nous permet de lire les images au format tiff, voici le résulat obtenu:

Figure 5: Résultat visuel d'une image au format tiff

Grâce à la fonction *applyColorMap()* de la bibliotheque Opencv nous avons pu convertir la nuance de gris en nuance de couleurs. Voici le résultat obtenu en appliquant la map *COLORMAP_HOT* qui est la plus adaptée pour différencier des nuances de température :

Figure 6: Résultat du passage de nuance en gris en nuances de couleurs



8.4 Lecture/écriture des informations GPS

L'Exchangeable image file format ou EXIF est une spécification de format de fichier pour les images utilisées par les appareils photographiques numériques. Les informations de ces images sont contenues dans ce format de fichier. Ainsi

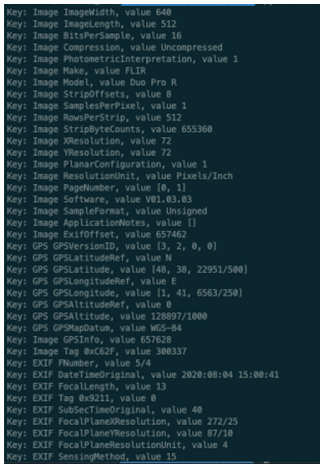
on y trouve des informations telles que la position GPS, la date et l'heure de prise de la photo et bien d'autres informations sur la photo et l'appareil photo. Nous avons donc pu récupérer les informations GPS directement depuis ce format de fichier. Plusieurs bibliothèques open source sont à notre disposition afin de lire les informations sur ces images. La première bibliothèque utilisée est readexif. Elle est très simple a utiliser, mais seulement pour lire les informations sur les photos.

Pour importer les données GPS de l'image d'origine à l'image résultat du réseau de neurones nous avons utilisé la bibliotheque *gpsphoto*. Même si cette bibliothèque fonctionne très bien pour tous les formats d'images, elle reste limitée par le type d'informations. En effet elle ne permet pas de récupérer les informations sur la date ou encore la focale.

Afin de réécrire toutes ces autres informations nous avons utilisé la bibliothèque tifffile pour les images tiff et exif pour les autres types d'images. En effet ces informations sont écrites de manière différente selon le format de l'image traitée, à savoir soit le format tiff ou d'autres types de formats classiques (par exemple JPG).

Voici un exemple d'informations enregistrées par les caméras utilisées par bees for life:

Figure 7: Exemple d'informations stockées pour une image tiff



8.5 Analyse des vidéos

L'application permet d'analyser les vidéos en utilisant les reseaux de neurones existants. La vidéo doit être placée dans le même dossier que les images à analyser. Lors de l'analyse des vidéos, une nouvelle fenêtre Opencv s'ouvre et lit les resultats de l'analyse image par image. Ces résultats sont enregistrés au format d'une vidéo annotée. Cette méthode fonctionne, mais nécessite trop de temps. L'idée serait d'analyser seulement une image par seconde et de tracker la bounding box du nid à l'aide des algorithmes de tracking proposés par open_cv. Bien que l'idée soit intéressante nous n'avons pas encore fini de mettre en place cette technique.

9 Tests et limitations

Le logiciel a été lancé sous chaque système d'exploitation : Windows, Linux et Mac. Chaque fonctionnalité déjà présente semble fonctionner correctement sous chacuns de ces OS. Néanmoins, le menu en cascade ne semble pas "se dérouler" sur Mac pour une raison inconnue...

Le "bug du caractère spécial" a également été vérifié, ce qui permet d'arriver à la conclusion qu'il n'apparaît pas sous Windows contrairement aux autres OS (se référer à la section **Réalisations** pour plus de détails), cependant le correctif n'a pas pu être testé pour des raisons de temps et de compatibilité.

Les 3 méthodes pour gérérer un exécutable (ou substitut) ont échouées d'après les tests, chaque test mène à un crash instantané sans rapport d'erreur en retour (sous Windows) ou "ne fait rien" sous Linux et Mac, rendant les screenshots impossibles. En revanche le script qui lance le programme sans passer par un exécutable affiche un message d'erreur concernant les restrictions d'exécution selon l'OS utilisé.

Bien que le programme soit cross-plateform, certaines fonctionnalités ou rendus visuels semblent différer d'après les tests. Le problème majeur vient du menu en cascade, les autres (sans considérer les problèmes de lancement et le traitement des nom d'images corrompus) sont superflux et ne peuvent être résolus car dépendent de l'architecture de l'OS sur lequel le programme tourne. Par exemple certains boutons peuvent apparaître en bleu sous Mac

et gris sous Windows.

10 Comparaison entre plannings

Prévisionnel	Sem. 1	Sem. 2	Sem. 3	Sem. 4	Sem. 5	Sem. 6
Louis-Gabriel	Etat de l'art	Analyse de l'interface	Mise en place du squelette de l'interface	Rendre l'application fonctionnelle	Différencier nids actifs/non-actifs	Idem
Sebastien	Analyse du réseau de neurones	Analyse d'images FLIR	Modifier/entraîner le réseau sur des FLIR	Lancer la detection sur des FLIR	Apprentissage au fur et à mesure	Idem
Mehdi	Analyse du réseau de neurones	Anayser format MOV	Lancer la détection sur des vidéos	Enregistrer les résultats des analyses	Analyse des nuages de points	Idem
					Rapport	Rapport

Certaines tâches imprévues se sont ajoutées au fil du projet, ce qui a causer de revoir une partie de l'organisation initialement proposée. Par exemple, un problème dans le nom de certaines images cause au programme de crash à la lecture de leurs noms sous Linux et Mac. Il a donc dû faire des compromis entre les précédentes et nouvelles tâches. Certaines nouvelles tâches ont également été incluses à la demande du client, dont certaines purement optionnelles.

Par exemple le fait de générer un exécutable ou de supprimer les caractères spéciaux n'était pas prévu au départ, ces tâches se sont ajoutées dans le planning effectif, il a alors fallut adapter les tâches pour gérer ces cas.

11 Installation & exécution

11.1 Mise en place

Le projet est actuellement une archive à décompresser. Avant toute opération, il est recommandé d'installer Python et Anaconda s'il ne sont pas installés, puis d'ouvrir un terminal et de se déplacer jusqu'au dossier de l'archive décompressée du projet (avec la commande cd).

Si le logiciel doit être lancé sous Linux ou Mac, lancer le script install.sh pour installer toutes les dépendances nécessaires au bon fonctionnement du logiciel. Si le logiciel doit être lancé sous Windows, le script install.sh ne fonctionnera probablement pas (en fonction de dépendances et/ou de restrictions), il faut alors l'ouvrir avec un éditeur de texte, copier-coller chaque ligne dans le terminal.

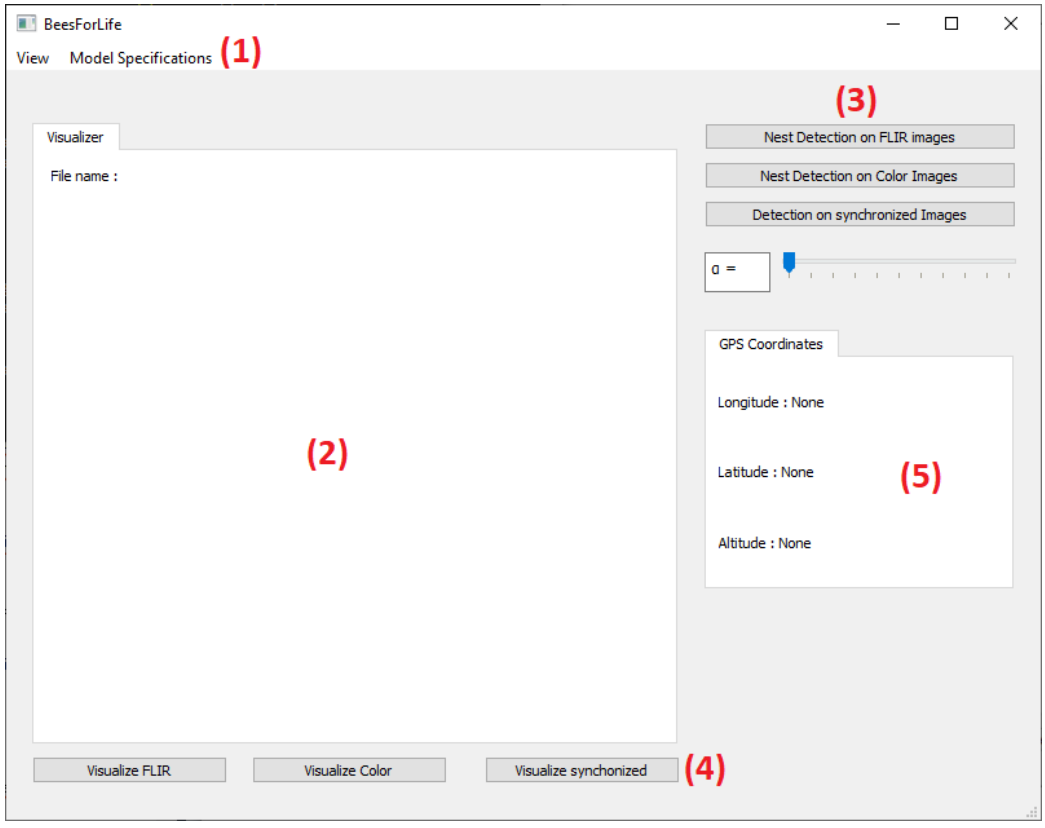
11.2 Lancement

Si l'environnement de Anaconda n'est pas activé, saisir **conda activate beeslife** afin de l'activer. Le logiciel se lance grâce à app2.py, il faut donc saisir **python app2.py** pour le lancer. Tout est prêt à ce stade, le logiciel attend une interaction utilisateur.

Afin que le logiciel prenne en compte les nouvelles images entrantes, il est nécessaire de les mettre dans le sous-dossier adapté.

12 Fonctionnement

Le logiciel permet la détection des nids de frelons asiatiques grâce à plusieurs options que nous décrivons ici. L'affichage ci-après représente l'interface du logiciel qui est présentée par la suite dans ce document.



Les indices en rouge ne font évidemment pas parties du logiciel, elles sont déposées ici pour faire référence à certaines zones du programme afin de mieux les présenter.
Voici d'abord un rappel des fonctionnalités actuelles, puis les nouvelles fonctionnalités implémentées au fil du projet.

12.1 Fonctionnalités initiales

(1) Les menus **View** et **Model Specifications** permettent de visualiser des rapports divers et des résultats des entraînements du réseau de neurones, ainsi que des graphes sur ces résultats.

(2) La zone d'affichage **Visualiser** permet de voir une image choisie aléatoirement contenant les nids de frelons détectés. Cette zone est initialement vide et dépend de certains boutons.

(3) Les boutons à droite **Nest Detection on FLIR Images**, **Nest Detection on COLOR Images** et **Detection on synchronized Images** lancent la détection des nids sur l'ensemble des images respectivement sur les images FLIR, COLOR et FUSION (des dossiers associés), ces détections prennent du temps en fonction du nombre d'images contenues dans ces dossiers. Les images résultantes sont sauvées dans des dossiers spécifiques avec des "cadres" autour des nids détectés par l'application.

Un curseur slider α modulable entre 0 et 1 permet de manipuler l'impact des scores de confiance, provenant des images FLIR et COLOR. Une description plus détaillée des calculs est donnée dans l'article de conférence [1][p. 4] à la base du projet.

(4) Enfin trois boutons **Visualize FLIR**, **Visualize COLOR** et **Visualize synchronized** permettent de changer l'affichage dans **Visualizer** en prennant aléatoirement l'une des images contenue au choix dans FLIR, COLOR ou FUSION selon leurs boutons respectifs.

12.2 Nouvelles fonctionnalités

(5) Il est désormais possible de voir les coordonnées GPS de l'image actuellement affichée dans **Visualizer** grâce au tableau d'informations **GPS coordinates**. Cette option affiche dynamiquement la latitude, la longitude et l'altitude.

Le nom de l'image est désormais affiché au-dessus de l'image dans **Visualizer** **(2)** afin de pouvoir retrouver l'image en cas d'anomalie ou d'un besoin quelconque.

13 Conclusion

13.1 Bilan

Le sujet s'est avéré plus difficile que prévu à cause de certaines contraintes matérielles et virtuelles, mais la pratique a été très intéressante car le fait d'avoir une relation client-développeur permet de se rendre compte à quoi peut ressembler un projet plus concret et de plus grande envergure.

De nouveaux formats d'images sont désormais disponibles dans l'application et l'interface a été revue pour y inclure de nouvelles informations.

13.2 Perspectives

Il est peut être possible de corriger le problème de lancement du logiciel par exécutable avec un code spécifique qui outrepasserait les restrictions, tel que le **exec** en C, puis d'en faire un exécutable (à condition de gérer le problème de chemin d'accès, voir la section **Réalisation** pour plus de détails). Anaconda semble poser problème pour l'utilisation de Pyinstaller, peut être qu'en réadaptant l'environnement virtuel il est possible que Pyinstaller puisse fonctionner et génèrer de lui-même un exécutable, mais c'est une solution difficile et abstraite sans rapport d'erreur.

L'utilisation d'un algorithme de tracking proposé par open_cv lors de l'analyse d'une vidéo permettrait un gain de temps considérable. En effet ces algorithmes, tel que KCF_tracker sont plutôt efficaces et permettent de suivre des objets en temps réel.

14 Remerciements

Nous voudrions remercier les équipes de Bees for Life, ainsi que notre enseignant Monsieur Pascal Desbarats pour nous avoir proposé ce sujet.
Les rendez-vous réguliers hebdomadaires nous ont permis de nous sentir mieux guidés tout au long du projet, un véritable suivi ayant ainsi été réalisé.
Nous tenons également à remercier Monsieur Lionel Willaert pour l'entretien que nous avons eu avec lui, ce qui a rendu le PFE d'autant plus concret et nous a appris à travailler directement avec le client, rendant aussi l'expérience encore plus professionnalisante. Merci enfin d'avoir rendu ce projet possible.

References

[1] T. Shams and P. Desbarats. Detection of asian hornet's nest on drone acquired flir and color images using deep learning methods. In *2020 Tenth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6, 2020.

[2] Adrien Perrard, Jean Haxaire, Agnès Rortais, and Claire Villemant. Observations on the colony activity of the asian hornet vespa velutina lepeletier 1836 (hymenoptera: Vespidae: Vespinae) in france. In *Annales de la Société entomologique de France*, volume 45, pages 119–127. Taylor & Francis, 2009.

[3] Daniele Milanesio, Maurice Saccani, Riccardo Maggiora, Daniela Laurino, and Marco Porporato. Design of an harmonic radar for the tracking of the a sian yellow-legged hornet. *Ecology and evolution*, 6(7):2170–2178, 2016.

[4] David Marius Daniel, Cîşlariu Mihaela, and Terebeş Romulus. Combining feature extraction level and score level fusion in a multimodal biometric system. In *2014 11th International Symposium on Electronics and Telecommunications (ISETC)*, pages 1–4. IEEE, 2014.

[5] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Luc de Haro, Magali Labadie, Pierre Chanseau, Claudine Cabot, Ingrid Blanc-Brisset, Françoise Penouil, and National Coordination Committee for Toxicovigilance. Medical consequences of the asian black hornet (vespa velutina) invasion in southwestern france. *Toxicon*, 55(2-3):650–652, 2010.

[9] Dymitr Pietrow and Jan Matuszewski. Objects detection and recognition system using artificial neural networks and drones. In *2017 Signal Processing Symposium (SPSympo)*, pages 1–5. IEEE, 2017.

[10] Kamel Aizi and Mohamed Ouslim. Score level fusion in multi-biometric identification based on zones of interest. *Journal of King Saud University-Computer and Information Sciences*, 2019.

[11] Peter J Kennedy, Scott M Ford, Juliette Poidatz, Denis Thiéry, and Juliet L Osborne. Searching for nests of the invasive asian hornet (vespa velutina) using radio-telemetry. *Communications biology*, 1(1):1–8, 2018.

[12] Sebastien Jean Mambou, Petra Maresova, Ondrej Krejcar, Ali Selamat, and Kamil Kuca. Breast cancer detection using infrared thermal imaging and a deep learning model. *Sensors*, 18(9):2799, 2018.

[13] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016.

[14] Hao Qu, Lilian Zhang, Xuesong Wu, Xiaofeng He, Xiaoping Hu, and Xudong Wen. Multiscale object detection in infrared streetscape images based on deep learning and instance level data augmentation. *Applied Sciences*, 9(3):565, 2019.

[15] Jinhui Han, Kun Liang, Bo Zhou, Xinying Zhu, Jie Zhao, and Linlin Zhao. Infrared small target detection utilizing the multiscale relative local contrast measure. *IEEE Geoscience and Remote Sensing Letters*, 15(4):612–616, 2018.

[16] Xesús Feás Sánchez and Rebecca Jane Charles. Notes on the nest architecture and colony composition in winter of the yellow-legged asian hornet, vespa velutina lepeletier 1836 (hym.: vespidae), in its introduced habitat in galicia (nw spain). *Insects*, 10(8):237, 2019.

[17] Christelle Robinet, Christelle Suppo, and Eric Darrouzet. Rapid spread of the invasive yellow-legged hornet in f rance: the role of human-mediated dispersal and the effects of control measures. *Journal of Applied Ecology*, 54(1):205–215, 2017.

[18] Renu Khandelwal. Computer vision: Instance segmentation with mask r-cnn. *Towards data science*, 2019.