Detection of Nested Objects in Dense Scenes using Deep Learning - Application to Bee and Varroa Detection

Yassine Kriouile

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École des Mines de Paris

Detection of Nested Objects in Dense Scenes using Deep Learning - Application to Bee and Varroa Detection

Détection d'objets imbriqués dans des scènes denses par Apprentissage Profond - Application à la détection d'abeilles et de varroas

Soutenue par

Yassine Kriouile

Le 08 décembre 2022

École doctorale n°621

Ingénierie des Systèmes, Matériaux, Mécanique, Energétique (ISMME)

Spécialité

Informatique temps réel, robotique et automatique

Composition du jury :

Amar RAMDANE CHERIF Professeur des universités Université Paris Saclay	*Président du jury* *Rapporteur*
Président du jury Anna FABIJAŃSKA Professeure, University of Lodz	*Rapporteur*
Faten CHAIEB Maître de conférence EFREI Paris Panthéon-Assas Université	*Examinatrice*
Hassen DRIRA Professeur des universités Télécom Physique Strasbourg	*Examinateur*
Slim M'HIRI Maître de conférence ENSI - Université de Manouba	*Examinateur*
Lamine BOUGUEROUA Maître de conférence EFREI Paris Panthéon-Assas Université	*Examinateur*
Katarzyna WEGRZYN-WOLSKA Professeure EFREI Paris Panthéon-Assas Université	*Co-directrice de thèse*
Corinne ANCOURT LE QUELLENEC Directrice de recherche, Mines Paris - PSL	*Directrice de thèse*

MINES PARIS | PSL

Acknowledgements

First and foremost, I would like to thank God for all his blessings and for providing me the opportunity to work in a research field that I appreciate.

I would like to express my sincere gratitude to my supervisors Corinne Ancourt, Katarzyna Wegrzyn-Wolska, and Lamine Bougueroua for their continuous support of my Ph.D. study and research, for their motivation, for their encouragement during my thesis. They offered me generously their time and gave me relevant advice about technical and non-technical issues regarding my thesis and its development. They always had faith in me and encouraged me to move forward with my research and my studies. Their guidance helped me in all the time of research and writing of this thesis.

I also express my deepest thanks to Pierre Jouvelot for giving me the necessary advice and guidance, which were extremely valuable for my progress concerning my work during my thesis.

I would also like to express my appreciation to the jury committee members: Amar Ramdane Cherif, Anna Fabijanska, Faten Chaieb, Hassen Drira, Slim M'Hiri for the time spent reading this work, attending the defense and for all their interesting feedback. Specials thanks go to the reviewers Amar and Anna for their valuable remarks that helped me enhance my thesis manuscript.

I would also like to thank Alexandre Dangleant, Lucille Johanet, Constance Beri, Elodie Rumiano, Coline Kouchner, members of PNAPI project, who helped me in software design tasks, annotating tasks and provided me with important information concerning beekeeping domain.

I thank all researchers and colleagues of Efrei Research Lab and Mines Paris – PSL which encouraged me and gave me advises.

I thank all my friends who support me during my thesis. I would also like to express my gratitude to my brother Saad for his constant support during my thesis, and all my brothers and sisters who always encouraged me. I am extremely grateful to my beloved parents for their unlimited support during my years in France and all my life. Thank you for your sacrifices, your love, your encouragement, and your trust.

Special thanks to my mother for her crucial role and invaluable assistance. I am forever indebted to her.

Résumé

Dans cette thèse, nous abordons deux problèmes essentiels lors de la détection d'objets en traitement d'image : 1) maintenir une bonne précision d'objets détectés, essentiellement dans le cas d'images denses et 2) détecter des objets potentiellement imbriqués dans les premiers. Ces deux problématiques existent particulièrement dans le domaine apicole. En effet, un apiculteur doit veiller au niveau d'infestation de son rucher par le parasite varroa qui s'installe sur le dos des abeilles. Les méthodes de traitement d'images par ordinateur peuvent être utilisées pour automatiser cette tâche et la rendre plus précise.

L'apprentissage profond est à l'origine de progrès dans de nombreux domaines liés à la technologie, notamment en reconnaissance faciale et vocale, et dans le traitement automatisé du langage et des images. Nous avons appliqué et adapté ces techniques à nos problématiques. Nos travaux sont basés notamment sur le réseau de neurones Faster R-CNN. Nous proposons une extension de cette architecture pour améliorer la précision de la détection des objets imbriqués dans les scènes denses. Notamment, nous proposons d'ajouter à ce réseau des branches qui détectent les objets à partir de leurs coins. Cela permet de retrouver les objets partiellement cachés. Afin de détecter les objets potentiellement imbriqués, nous nous sommes basés sur l'architecture Mask R-CNN, une extension de Faster R-CNN, dédiée à la segmentation d'objets. Nous ajoutons une branche au Faster R-CNN pour segmenter les objets internes.

Nos expériences sont basées sur un ensemble d'images de scènes denses d'abeilles, contenant des abeilles et des varroas annotés, que nous avons constitué. Les abeilles et les varroas sont les objets imbriqués à détecter. Nos résultats montrent une amélioration de +10% de la précision de détection par rapport à l'approche standard. Après avoir testé différents manières d'extraire les informations pour la segmentation, nous avons défini un seul réseau de neurones capable de détecter les deux types d'objets imbriqués sans diminution de la précision par rapport à deux réseaux de neurones séparés.

Nous montrons également que notre architecture de réseaux de neurones étendue améliore la détection d'objets dans d'autres domaines tels que celui de la détection des personnes dans les scènes denses, où nous avons des résultats qui atteignent +3% par rapport l'approche standard.

Mots clés : détection d'objets, scènes denses, objets imbriqués, varroa, abeille, apprentissage profond, réseau de neurones, Faster R-CNN, Mask R-CNN, segmentation d'objets

Abstract

In this thesis, we address two essential problems related to the detection of objects and image processing: 1) maintaining a good precision of detected objects, essentially in the case of dense images and 2) detecting objects potentially nested in the first. These two problems exist particularly in the beekeeping sector. Indeed, a beekeeper must ensure the level of infestation of his apiary by the varroa parasite which settles on the backs of bees. Computer image processing methods can be used to automate this task and make it more accurate.

Deep learning is driving advances in many technology-related areas, including facial and voice recognition, automated language and image processing. We have applied and adapted these techniques to our problems. Our work is based in particular on the Faster R-CNN neural network. We propose an extension of this architecture to improve the accuracy of nested object detection in dense scenes.

To improve object detection in dense scenes, we propose to add branches to the Faster R-CNN neural network that detect objects from their corners. This allows to find partially hidden objects. In order to detect potentially nested objects, we rely on the Mask R-CNN architecture, an extension of Faster R-CNN dedicated to object segmentation. We add a branch to the Faster R-CNN to segment internal objects.

Our experiments are based on a set of images of dense bee scenes, containing annotated bees and varroa mites, that we have put together. Bees and varroa mites are nested objects to detect. Our results show an improvement that reaches +10% in detection accuracy compared to the standard approach. After testing different ways to extract information for segmentation, we ended up with a single neural network capable of detecting two nested objects without decreasing accuracy compared to two separate neural networks.

We also show that our extended neural network architecture improves object detection in other areas such as people detection in dense scenes, where we have results that reach +3% compared to the standard approach.

Keywords : object detection, dense scenes, nested objects, varroa, bee, deep learning, neural network, Faster R-CNN, Mask R-CNN, object segmentation

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Résumé

Dans ma thèse, je m'intéresse au sujet de la détection d'objets imbriqués dans des scènes denses. Je me base sur les techniques de vision par ordinateur qui permettent d'extraire des informations sémantiques depuis des images numériques. Parmi les tâches qu'accomplissent ces outils, il y a la détection d'objets. Elle consiste à trouver les positions des objets présents dans une image ainsi que leurs classes. Le traitement d'images par ordinateur pour l'extraction d'informations intéressantes nécessite une étape d'extraction de caractéristiques. Les méthodes traditionnelles de reconnaissance d'objets nécessitent de choisir soigneusement cette méthode en tenant compte du type d'images à traiter. L'apprentissage profond permet d'automatiser l'extraction des caractéristiques en utilisant un réseau de neurones entraîné sur les images à traiter. Il existe deux catégories de détecteurs d'objets basés sur l'apprentissage profond; les détecteurs à une seule étape, et les détecteurs à deux étapes. Ces derniers génèrent des propositions à partir de chaque image, puis les classifient pour obtenir les positions des objets dans l'image et leurs classes. Mes approches sont basées sur le réseau de neurones à deux étapes Faster R-CNN.
Les scènes denses correspondent à des situations où il y a plusieurs objets proches entre eux. Dans ce cas, la probabilité qu'un objet soit partiellement caché par un autre est grande. Ainsi le risque de rater ces objets partiellement cachés est plus importante à cause du manque d'information les concernant. En effet, les détecteurs classiques détectent les objets à partir de leurs centres. Pour pallier ce problème, l'idée intuitive employée est de prédire des informations concernant les coins des objets qui restent visibles malgré l'occlusion. Mais la plupart des approches proposées nécessitent d'autres annotations et données. Je propose d'étendre le réseau de neurones Faster R-CNN pour détecter les objets à partir de leur centre et par conséquent limiter le risque de louper les objets partiellement cachés. Mon approche améliore la précision de détection de plus de 10%.

Résumé

Les objets imbriqués sont des objets inclus dans d'autres. Dans ma thèse, je traite le cas de deux objets: l'objet englobant et l'objet inclus. La manière classique de les détecter est d'entraîner un réseau de neurones sur des objets de deux types différents. Cela est difficile quand les types d'objets sont très différents en termes de forme, taille et taux d'apparition. Une autre méthode consiste à entraîner deux réseaux de neurones, chacun pour détecter un objet de type spécifique. Cette approche n'est pas efficace en termes de ressources et de temps car elle extrait les caractéristiques deux fois d'une même image d'entrée. Je propose d'entraîner le réseau de neurones Mask R-CNN pour détecter les objets englobant et les objets inclus.
L'abeille est très importante pour l'être humain. Mais cette insecte souffre d'un problème qui dégrade son état de santé: l'infestation par le parasite varroa. Afin de protéger les abeilles, les apiculteurs doivent surveiller le niveau d'infestation pour éventuellement traiter les ruches. Les méthodes classiques proposées pour l'estimation du pourcentage d'infestation sont manuelles et basées sur l'échantillonnage. Elles nécessitent du temps et des efforts, par conséquent ne sont pas précises. Je propose d'utiliser les réseaux de neurones pour une prédiction automatique du nombre de varroas. Afin d'obtenir un système précis et efficace, je propose des solutions dans ma thèse traitant le cas des abeilles et des varroas: scènes denses (abeilles proches) et objets imbriqués (varroa sur les abeilles).

Thanks to the increase of computer resources, artificial intelligence has showed its benefit in resolving and automating many tasks. Facial and voice recognition, natural language processing, and computer vision are examples of tasks that are nowadays automated through deep learning. In my thesis, I deal with problems related to computer vision and image processing. I propose approaches that perform detection in dense scenes and nested object detection.

Photos are numeric data that reflects real scenes. Each photo contains semantic information that human brain can explain, interpret and make decisions based on it. Depending on the task to fulfill, extracting information from an image could require effort and time. Furthermore, errors are usually present in this type of tasks due to human inattention and distraction. Computer vision are a set of methods implemented on computers to handle information extraction from images. Among the main task that are important in computer vision, there is object recognition. The aim of this task is to find objects of one class or different classes inside an image. There are different types of object recognition: object detection (the output is a set of bounding boxes), semantic segmentation (the output is a mask), instance segmentation [Zhao, 2018]... Computer vision task can be achieved through traditional image processing method that enables extracting information from images. To get these information called "features", a specific method should be carefully chosen, and machine learning algorithms are used to get final results. Thanks to deep learning, feature extraction is more automatic, neural networks are trained on ground truth dataset to obtain the best parameters to achieve the desired task. There are many neural network architectures that were proposed for object detection and segmentation [Jiao, 2019]. They can be classified into two categories: (1) one-stage object detector; objects are directly predicted from images, and (2) two-stage object detectors; the neural network generates proposals, each one is then classified to obtain predicted objects. To have lighter training in terms of required time and dataset size, transfer learning is used. It consists in loading parameters of already trained neural network on big dataset into a new neural network that must be trained for a specific task. My proposed approaches are based on Faster R-CNN [Ren, 2015] which is a state-of-art neural network, it has two-stage architecture and can be extended to perform object segmentation.

Dense scenes correspond to scenes where many objects are present and close to each other. Detecting objects in such situation could be challenging due to high object occlusion. In dense scenes, some objects are partially hidden by others, thus there is a loss in information compared to visible objects. According to used method, partially hidden objects could be missed and accuracy could be degraded. The famous case of dense scenes that are dealt with is scenes of human crowd. In fact, among the main tasks for which computer vision is used is the detection of pedestrians and humans in street in general, this is particularly important for security surveillance or self driving. There are proposed deep learning approaches to resolve this type of issues. The main idea used to resolve this problematic, is to detect different parts or corners of a human body to avoid missing a human when only some of its parts are visible. To achieve this detection, additional annotations corresponding to human parts are usually needed. I propose to resolve the issue without using additional annotations. My approach is based on Faster R-CNN neural network used for object detection. The limit of the standard architecture is that it predicts objects from their centers, it means that objects whose centers are visible

have more probability to be detected. I propose to add branches to the neural network to detect partially hidden objects. Compared to standard Faster R-CNN, my approach increases the detection accuracy up to +10%.

Another interesting issue to handle when detecting objects using computer vision and machine learning is nested object detection. In fact, there are use cases where detecting objects inside others is needed. For example, detecting faces and eyes is important for human detection, face identification and emotion detection. This problem can be seen as multi-class detection, it corresponds to the task of detecting objects of different types in an image. Almost all object detection approaches based on deep learning like Faster R-CNN handles multi-class detection. But this kind of approaches can be limited in case the objects to detect are very different in size, shape, and occurrence rate. When the type of objects is very different, the detector has more difficulties to extract from images specific information to each object type. A difference in occurrence rate is also a factor that can make training an object detector for multi-class detection impossible. When an object is constantly present in all images, but the nested object is rarely present inside the first object, it is practically not easy to build an annotated dataset for both object types. To overcome these two limits, a detection sequence could be considered, the largest object is detected first then another detection is performed to detect the eventual inner object. This structure could be achieved through two different neural networks trained on two different specific datasets. This approach consumes time and resources. I propose to perform detection of nested objects by training one neural network that provides bounding boxes for enclosing objects and region pixels for internal objects. My approach is based on Mask R-CNN [He, 2017] architecture which is an extension of Faster R-CNN. Instead of segmenting the main object, the segmentation branch segments inner objects. The backbone features that should be forwarded to the segmentation layers are carefully chosen to obtain a neural network capable of detecting nested objects without loss of accuracy compared to two separate neural networks.

Bees are very important for the human life. In addition to be the source of interesting products like honey and royal jelly, they participate in food production through pollination. Therefore protecting bee health is a paramount necessity for human being. Nowadays, the bee health is deteriorated because of different factors: pesticides, pollution, parasites spread ...

Fortunately, the advance of human being in science does not just have drawbacks, thanks to this advance, many problems are solved, artificial intelligence and image processing are good examples of advances that bring automatism and efficiency. The increase of computer resources and execution speed helps us to perform many tasks rapidly and precisely, like detecting intruders or self driving. We can benefit from this type of technologies to handle bee issues.

Among the main problems that the bee encouter is varroa infestation. The term "bee" is commonly used to name the "Apis Mellifera" (also named "European bee") which is subspecies of the group of insects "Bee" or "Anthophila" [Moisset, 2021]. In this manuscript, "bee" term is used for "Apis Mellifera". Varroa is a small parasite that attack bees, it causes the diminution of bee number and economic losses. I think that the first infestations started in southeast Asia. In this region, the varroa parasite was living with its original host "Apis cerana" (subspecies of "Anthophila") without causing any issue. To produce more honey, European bees were transferred

to that region, and varroa meets for the first time "Apis Mellifera". Being more vulnerable, bees suffer a lot from this parasite. Due to bee transfer and economic exchange of bee colonies, the spread of varroa was quick. It has reached almost all countries. Varroa infestation is now considered as a worldwide problem [Reid, 2004]. Poor bees have their own Covid that must be dealt with urgently to avoid the decrease of number of bees and deterioration of their health. These small creatures need human help and compassion. Either way, if we do not have any compassion to animals, we still must protect them as their life is crucial for our existence.

Neural networks are among the main technologies behind many advances in different domains. They are already being used in industry to automate production tasks and improve efficiency. In recent years, they started to be used in agriculture, nevertheless its utilization remains still limited compared to the potential and capacity that neural networks have. Especially in beekeeping domain where technology is rarely used. In this manuscript, I demonstrate an aspect of using neural networks as an efficient tool that can participate in protecting bees against varroas.

Beekeepers should monitor the health of their bee colonies. They have to estimate the size of their colonies and check the existence of parasites such as varroas. In fact, the varroa infestation level is an important parameter to monitor for beekeepers to be aware of the health status of their apiaries. In case this level exceeds defined thresholds, specific treatment and actions are triggered [David, 2006]. Computer vision methods based on deep learning could help beekeepers to automate monitoring tasks. I propose to use neural networks to detect bees in images and count varroas on them. To achieve this objective, I must deal with high density constraint. In fact, images of bee frames contain usually many bees that are close to each others, detection in this situation could be challenging. Detecting objects from corners is the approach that I suggest to overcome this issue. Varroas are present on bees, the probability of their occurrence is very rare compared to bees. To deal with detecting two nested objects using only one neural network, I propose to adapt Mask R-CNN structure and training.

To validate the corner approach dedicated for dense scenes, I have trained proposed neural network on public dataset Crowd Human. The obtained results have confirmed the relevance of the approach and its capability to detect partially hidden objects.

My proposed approaches are originally proposed for bee and varroa detection. As they are based on deep learning, training and evaluation dataset are required. I have constructed image set of bee frames. I have annotated it by bee and varroa positions using different annotating tools. I have also developed code to convert annotations to the expected format by the used library Detectron. My approaches, training and evaluating processing are coded under Detectron library which proposes implementation of different neural networks of computer vision. This library has the advantage to be extensible and well documented. To evaluate the accuracy of proposed approaches, I selected relevant standard metrics: average precision, average recall and mean average precision.

This manuscript represents the detailed description of my research work realized during my thesis preparation. To carry out my research work, I have followed a scientific methodology. The first task was to define clearly my problematic to deal with. This means understanding the

context of the thesis and its scopes. Then I have searched for existing methods and approaches that deal with similar objectives. After that, I selected the tools to build my datasets and approaches. After the implementation phase, I performed an evaluation step to assess the relevance of my proposed approaches.

After brief introduction, Chapter 2 presents the context of my research. Then, in Chapter 3, I present definitions related to computer vision and describe approaches used for object detection. In the next chapter, Chapter 4, I explain the conditions of my implementations and tests. After that, in Chapter 5, I explain my first contribution consisting in detecting objects from their corners. Then, in Chapter 6, I explain the second contribution proposed to detect nested objects. Finally, in Chapter 7, I present some experiments carried out to validate my results.

Chapter 2

Context

Résumé

L'abeille est un insecte important pour l'humanité. Elle est source de produits bénéfiques comme le miel, et assure la pollinisation. Les semences de 84% des espèces cultivées de l'UE dépendent de la pollinisation. L'apiculture constitue l'ensemble des tâches qui permettent d'élever les abeilles afin d'exploiter leurs produits. Les abeilles sont souvent hébergées dans des ruches artificielles conçues pour optimiser la production. Ces dernières années, les abeilles souffrent d'un taux élevé de mortalité qui est la cause de différents facteurs: contamination par des produits chimiques, utilisation des pesticides, infestation par le varroa, changement climatique.
Afin de protéger les abeilles et aider les apiculteurs dans leurs tâches, le projet PNAPI a été créé par ITSAP et EFREI Paris, en s'alliant avec d'autres partenaires spécialisés dans l'apiculture. L'objectif du projet est de construire une plateforme d'aide aux apiculteurs. Ce système accompagne, particulièrement, les apiculteurs dans leurs tâches de comptage de varroas, en offrant des méthodes automatisées de collection, de sauvegarde et d'analyse de données. En effet, les méthodes classiques de comptage de varroas se basent soient sur le prélèvement d'un échantillon d'abeilles ou de larves ou l'utilisation d'une lange graissée au dessous de la ruche pour collecter les varroas tombés. Dans les deux cas, l'apiculteur doit compter manuellement les varroas pour estimer le niveau d'infestation. Dans ma thèse, je propose d'employer les méthodes d'apprentissage profond pour faciliter le comptage automatique et améliorer sa précision.

Contents

1 Introduction

In this chapter, I present the context of my research. In Sections 2.2 and 2.3 I introduce bee and beekeeping domains. Then, in Section 2.4, I describe the issues from which bees are suffering. In Section 2.5, I present the project PNAPI that gave birth to my research work. Then, in Section 2.6, I talk about the potential of technology in beekeeping domain. In the next Section, 2.7, I explain the traditional ways used for counting varroas. Finally, I conclude the chapter by a conclusion in Section 2.8.

2 Bee

We tend to believe that honey is provided by the insect called "Bee". But in reality this term or "Anthophila" designates a set of flying insects sharing common characteristics such as pollination. Indeed the honey that humans consume is produced by honey bees or "Apis" which are themselves classified into several subclasses including "Apis mellifera" or "European bee" which is the species the most exploited for breeding and honey production [Moisset, 2021][Rhoné, 2015].

A colony of bees is made up of insects of different types, each of which plays a particular role; the "female" worker bee plays the most important role within the colony; namely cleaning, feeding the larvae, building the cells, collecting nectar and pollen. The colony contains a queen responsible for laying the eggs whose fertilization is the source of birth of new female bees. Finally there are drones; these are the "males" who are born from the unfertilized eggs. Their role is to fertilize the queen's eggs to give birth to bees [Research, 2022].

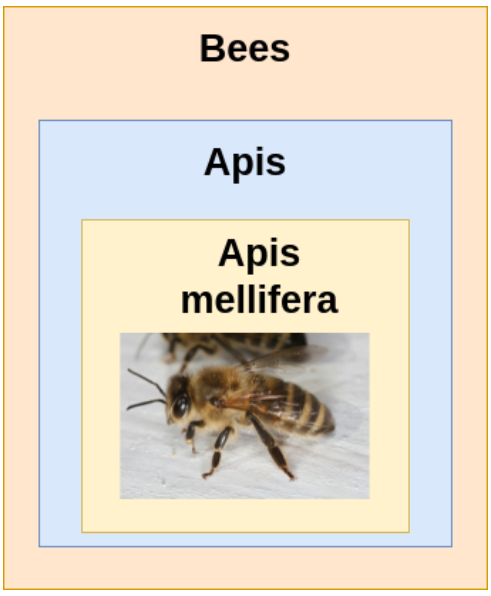


Figure 2.1: Bee Classification [1]

Figure 2.2: Female Bee [1]

[1] https://species.wikimedia.org/wiki/File:Apis_mellifera_carnica_worker_hive_entrance_2.jpg
[2] https://fr.wikipedia.org/wiki/Faux_bourdon#/media/Fichier:Drohn_im_Flug_08-3.jpg
[3] https://coteruche.com/blog/p-le-marqage-de-la-reine-des-abeilles

Figure 2.3: Drone "Male" [2]

Figure 2.4: Queen bee [3]

3 Beekeeping

Beekeeping is a set of skills aimed at breeding honey bees in order to exploit their products, which are honey, wax, royal jelly, pollen and propolis [McGregor, 2022]. The breeding of bees consists of using techniques and methods that preserve the colony from death, keep it healthy and protect it against diseases and predators in order to increase the productivity of bees in terms of the quantity of its products. The bees are bred in a hive set up by the beekeeper to house a colony of bees. The work of a beekeeper is mainly composed of actions in the field; like visits to check the health state of the colony, treatments, some specific maneuvers to protect the colony against the cold, introducing new bees into colonies, transporting hives from location to another ... These actions are usually planned according to the beekeeper calendar.

Figure 2.5: Bee Hives [4]

4 Bee Issues

Recently, the beekeeping industry is threatened by a decline in the number of bee populations and a drop in production. Beekeepers grapple with many issues related to the health of bee

[4] https://commons.wikimedia.org/wiki/File:Beehives_in_Mankato,_Minnesota.jpg

colonies. Scientific work is focused on a deep understanding of the causes of these phenomena in order to help beekeepers make the right decisions [Latioui, 2020]. In France, a study on the mortality rate of bee colonies during the winter of 2020-2021 was carried out with more than 17,000 participating beekeepers [5].

Figure 2.6 gives the estimate of the winter mortality rates of bee colonies in France. In general, the death rate is estimated at 24,8%.

It is now evident that environmental pollution and climate change are a great danger to our planet. Experts have confirmed that they pose a very dangerous threat to many species of terrestrial and marine fauna. However, the death of the bee population necessary for pollination is considered the most important warning for humans.

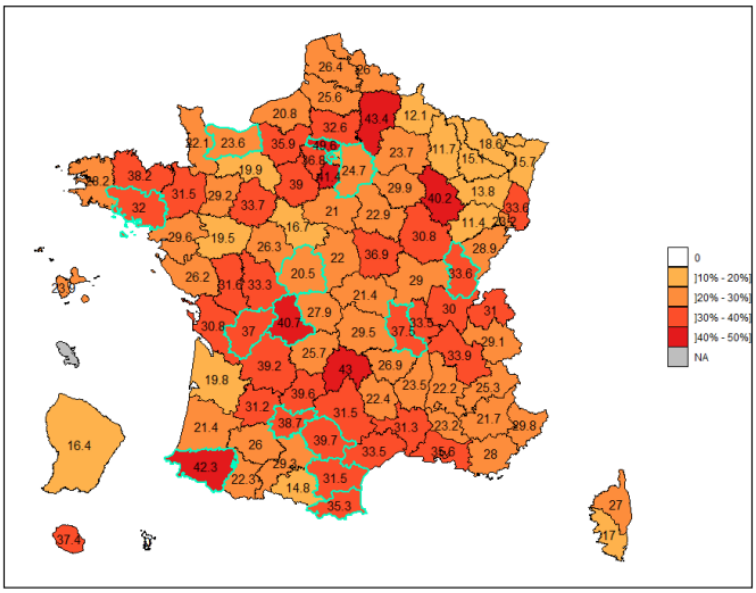


Figure 2.6: Winter mortality rates of bee colonies in France from ESA Platform

The bee is experiencing more and more problems related to its state of health, these issues engenders Colony Collapse Syndrome phenomena. It is the sudden death of a colony of bees due to a lack of adult bees in the hive. The mortality rate increased from 5% in 1990 to 30% in 2018 [Déborah, 2020][Pailloux, 2018]. The exact causes of this syndrome are unknown but some factors are suspected [Hood, 2020][Calavas, 2018]:

- Chemical contamination
- Use of pesticides
- Infection of bees by "varroa destructor" parasites
- Climate change
- Intensive farming

[5] Platform ESA. Mortality of bee colonies during winter 2020-2021, preliminary descriptive results for metropolitan France. Accessed: 2022-07-25 https://www.plateforme-esa.fr/sites/default/files/ENMHA_2020-2021_r%C3%A9sultats_pr%C3%A9liminaires_0.pdf

This syndrome has very heavy consequences on the economy; indeed the fragile health of the bees implies a diminution of honey quantities [FranceAgriMer, 2015][FranceAgriMer, 2017] produced by the bee, therefore an increase in price. In addition, since a significant part of the pollination of food crops is provided by bees, the syndrome generates enormous financial expenses to compensate for the drop in natural pollination activity.

5 PNAPI Project

The technical and scientific institute of beekeeping and pollination (ITSAP) works for the development of the beekeeping sector by carrying out applied research, assisting beekeepers and participating in the creation of services that improve yields and supporting beekeeping actors [ITSAP, 2020]. In 2019, it created a partnership with EFREI Paris school, and other partners specialized in beekeeping domain, and participated in a call for projects funded by the Ministry of Agriculture. The project is called PNAPI [6], it has been accepted with CASDAR[7] funding [8]. The overall objective is to develop a platform to collect and manage data from beekeepers, and to assist them in their decision-making. My thesis is a sub-project of PNAPI, it consists in using advanced techniques of artificial intelligence and computer vision to evaluate the number of destructive parasites (varroa) present in a hive and obtain the infestation level of a bee colony. After that, this information will be coupled with other information related to the environment of the hives in order to help with the diagnosis given by the support platform for beekeepers.

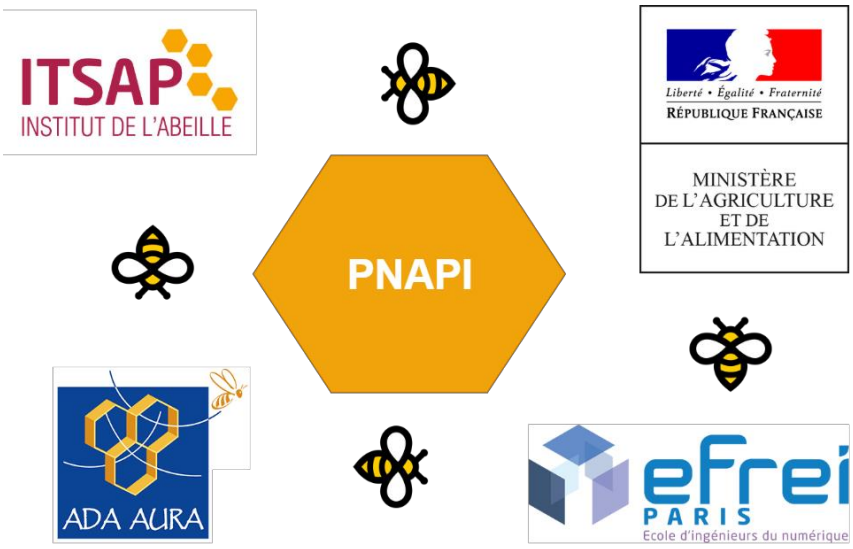


Figure 2.7: PNAPI Project Partners

[6] French: Plateforme numérique d'Accompagnement des APIculteurs, English: Digital Support Platform for Beekeepers

[7] CASDAR project number 18ART1831 (from 2019 to 2022)

[8] French: Compte d'Affectation Spéciale pour Développement Agricole et Rural, English: Special Appropriation Account for Agriculture and Rural Development

6 Technology in Beekeeping

Technology has become an important tool allowing humans to optimize their actions, increase yields and analyze phenomena. With this in mind, agriculture 4.0 is beginning to be born and see the light of day; it is a question of imitating other fields such as industry where technology has become an essential tool with undeniable added value. Indeed, communication and information technologies are becoming more of a necessity than an added value, because of the growing complexity of the systems to be managed. Beekeeping is an example; it is a field of interactions of several and different actors and of massive and heterogeneous data, information and knowledge.

The introduction of technological and computer tools in the field of beekeeping would have multiple advantages; both in terms of information collection and decision-making:

- Data collection: the automation made possible by ICT (communication and information technologies) allows efficient, rapid and less costly collection of massive data. These can be cleaned and structured in order to extract useful information for the beekeeper.

- Decision-making: ICT (communication and information technologies) can support the beekeeper in the decision-making phase thanks to graphical interfaces allowing the analysis of the data collected and presenting relevant indicators, and decision support functionalities which send recommendations and alerts using artificial intelligence algorithms.

Among the possible uses of technology in the field of beekeeping is image processing. Indeed, as it is the case for most areas of agriculture, observation is an essential activity that generally serves to identify the presence of problems or estimate the number or quantity of certain elements. For example, the beekeeper is led to observe the frames of the hive, estimate the number of bees and identify viruses or parasites. The exploitation of technological advances in image processing in beekeeping could therefore have interesting results in terms of efficiency and precision.

7 Varroa Counting

The varroa or more precisely the "varroa destructor" is a species of parasites native to Southeast Asia. It attacks the bee Apis mellifera and is considered among the main causes of colony collapse disorder of bees [Reid, 2004]. The varroa is positioned either inside the brood to attack the bee when it still is larva or nymph, or on the back of the adult bees. In the latter case the varroa is called phoretic varroa.

In order to treat bees against varroa mites, the beekeeper must first estimate the percentage of infestation, indeed there are critical thresholds of parasite pressure or percentage of infestation that should not be exceeded, these thresholds depend on the season of the year and trigger adequate treatments in order to maintain an acceptable pressure and avoid the death of the colony.

There are three types of Varroa:

[9] https://beeheater.com/index.php/our-story/

Figure 2.8: Varroa on a bee back (Phoretic Varroa) [9]

- those located inside brood,
- those located on the back of adult bees (called phoretic varroa),
- varroa mites falling from the hive due to aging.

For each type of varroa, there is a different counting method. Generally, estimating the number of varroa mites in the hive is based on counting varroa mites of a particular type.

With regard to brood varroa, a method called "uncapping the male brood" is used, which consists of collecting about 200 male brood nymphs using a comb and counting the pink dots corresponding to the varroa. If more than 50% of the male brood is infested, it means that the colony is in danger and treatment must be carried out [Défense Sanitaire Apicole du Var, 2022].

Figure 2.9: Comb used for uncapping the male brood [10]

Phoretic mites are counted using two different methods:

- Alcohol wash (lethal to bees)

[10] https://gdsa83.fr/evaluation-varroa/

- Icing sugar wash (not lethal to bees)

In both cases, a sample of bees is taken and placed in a special container to which alcohol or icing sugar is added depending on the chosen method. The objective is to be able to separate the bees from the varroa mites in order to count the number of varroa mites on the sample to deduce an estimate of the percentage of infestation in the hive. The limit values of infected adult bees percentage are between 1% and 5% according to the current season and colony stage. When the limit is exceeded a control must be carried out by the beekeeper [Défense Sanitaire Apicole du Var, 2022].

In order to calculate the number of varroa mites that fall from the hive, the "sticky board" or "greased diaper" method is used. The beekeeper puts a board under the hive, after a number of days varying between 3 to 7 days, he/she revisits the hive to count the number of varroa mites that have fallen on the diaper due to natural fall. The daily fall is calculated by dividing the number of varroa mites by the number of days between the two visits. Subsequently, the level of infestation and its criticality can be deduced from the number of falls per day. The limit values of fallen varroa number per day are between 1/2 and 25 according to the current season and colony stage. When the limit is exceeded a control must be carried out by the beekeeper [Défense Sanitaire Apicole du Var, 2022][Pierred, 2018].

Figure 2.10: Counting fallen varroas on sticky board [11]

Counting varroa mites is a difficult task to do manually and the result of counting on a sticky board is often not very accurate.

Thanks to technological progress, counting tasks can be automated and carried out by machines and by exploiting image processing algorithms. Thus the estimation of the level of varroa infestation of a colony can be achieved by taking a photo using a camera and using a computer providing an algorithm capable of detecting bees and varroa mites and counting them from images.

[11] https://gdsa83.fr/evaluation-varroa/

8 Conclusion

In this chapter I have seen the context of my research. The subject is related to bee domain, it is a part of a bigger project called PNAPI that aims to help beekeepers in their tasks in order to increase production and protect bees. The objective of my thesis is proposing approaches based on computer vision techniques, to detect bees and varroas to monitor the infestation level. Although the subject is related to bees, it remains only an application as the proposed approaches can be extended to other domains.

Chapter 3

State of the Art

Résumé

La vision par ordinateur constitue l'ensemble des traitements qui sont réalisés sur des images ou des vidéos pour en extraire des informations utiles. Une image est une matrice à deux dimensions, chaque point appelé pixel, correspond à une région unitaire de la scène et possède une information de couleur.
On peut classifier les méthodes de reconnaissance en deux catégories: la prédiction d'un rectangle entourant l'objet (appelé boîte englobante) et la prédiction des pixels correspondant à l'objet.
Afin d'extraire des informations depuis les images, des traitements élémentaires sont exécutés sur chaque pixel. Parmi les traitements classiques, on retrouve le filtrage et la convolution qui consistent à réaliser un produit de convolution entre une matrice, appelée noyau, et l'entourage de chaque pixel. Ces méthodes d'extraction permettent de produire un ensemble de caractéristiques pour chaque image. La prédiction d'objets est réalisée à partir de cet ensemble. Généralement, il existe deux manières de réaliser la reconnaissance, en utilisant soit un algorithme classique de traitement d'image, soit une méthode d'apprentissage. Je m'intéresse dans ma thèse à la deuxième car elle est indépendante de la nature des données à traiter. L'apprentissage profond permet d'automatiser toutes les opérations de reconnaissance d'objet, à savoir l'extraction des caractéristiques et l'étape de prédiction des objets. Cela est permis grâce aux structures de réseaux de neurones.
On peut classifier les détecteurs d'objets basés sur l'apprentissage profond en deux catégories. Il y a les détecteurs à une seule étape, comme SSD et YOLO, qui détectent directement les objets à partir des ensembles de caractéristiques extraits de l'image d'entrée. Les détecteurs à deux étapes, comme R-FCN et Faster R-CNN, génèrent d'abord des propositions, puis chaque proposition est classifiée pour déterminer si elle correspond à un objet réel ou pas.

Résumé

J'ai utilisé le réseau Faster R-CNN pour implémenter mes approches. Ce choix se justifie en raison de la performance de ce réseau de neurones qui se base sur le principe des *ancres*. En effet, Faster R-CNN est composé de trois parties principales: le "backbone" qui extrait les caractéristiques de l'image d'entrée. Puis RPN, ou réseau des propositions régionales, qui génère des boîtes englobantes à classifier. Cette génération se base sur l'utilisation des ancres qui représentent des boîtes englobantes générées pour chaque point de l'ensemble de caractéristiques. RPN prédit pour chaque ancre un score donnant la probabilité que l'ancre corresponde à un objet réel et des coordonnées permettant de décaler et corriger la position de la boîte. Les boîtes générées sont ensuite fournies au troisième composant de Faster R-CNN: "box head". Ce sous-réseau de neurones, en se servant d'une couche commune (ROI pooling), extrait les caractéristiques correspondant à chaque boîte proposée, puis à l'aide des couches complètement connectées, un score et des coordonnées de décalage sont de nouveau prédits. Afin de détecter des objets de différentes tailles, le backbone FPN (Feature Pyramid Networks: Réseaux de Pyramide de Caractéristiques) est utilisé, il génère plusieurs ensembles de caractéristiques de différentes tailles à partir de l'image d'entrée.

En ce qui concerne la détection d'abeilles et de varroas à partir des images, il y a différentes approches qui ont été proposées. Les principales limites de ces approches sont : utilisation des équipements spécifiques au niveau de la ruche, non traitement des cas d'occlusion, méthodes intrusives par rapport aux abeilles et des images utilisées non réalistes.

Dans ma thèse, je propose d'étendre le réseau de neurones Faster R-CNN pour améliorer la détection des objets imbriqués dans les scènes denses. L'approche est valorisée par l'application de la détection d'abeilles et de varroas utilisant une base d'images de cadre d'abeilles annotées.

Contents

1 Introduction

In this chapter, I present the state of the art related to my subject. In Section 3.2, I give definitions of some important concepts concerning computer vision and image processing. In Section 3.3, I define object recognition and its different tasks, I also present the types of methods used to perform these tasks. Then, as my objective is to detect bees and varroas from 2D images, I focus on methods based on artificial intelligence used for object detection in Section 3.4. In Section 3.5, I cite existing approaches proposed for varroa and bee detection.

2 Computer Vision and Image Understanding

Computer vision is a scientific field which deals with image processing by computers in order to extract valuable information. It can be defined as the set of methods used for simulating human visual system. Computer vision contains different sub-domains:

- Scene reconstruction
- Action recognition
- Object recognition
- Object tracking
- Human pose estimation
- ...

The data that computer vision handles can be static scenes (images) or dynamic scenes (videos). Each scene is stored as 2D matrix. The points of this matrix are called pixels, each one corresponds to a real quantized region. The value of the pixel is its color, it can be one value or multi-dimensional value. The most used color system is RGB (Red, Green and Blue). In this case, each pixel has three values between 0 and 255 representing these colors. In fact visible colors can be generated by computer screens by additive mixing red, green and blue lights at different intensities.

Another interesting format of representing a 2 dimensional image is to use cube format. An image contains three dimensions: height, width and channels. Each channel corresponds to a different color (red, green or blue).

A digital image represents a real scene. A human, thanks to his brain, can extract information from an image to make decisions. According to the task that he has to fulfill, he understands differently the content of the image (objects presents in the image, their locations, their characteristics ...). To achieve this understanding on computers, the image passes through successive steps, each one extracting information of higher level.

Among the tasks that are possible in the context of computer vision, there are tasks related to object recognition. For this type of tasks, the computer should detect the presence of objects in an image and their positions.

100 124 143	50 12 73	75 147 87	78 194 34
231 124 198	100 124 143	21 124 143	27 32 32
100 14 230	170 10 43	95 148 23	170 147 40
100 124 143	80 94 73	47 121 13	200 124 23

Figure 3.1: RGB image: each cell represents an image pixel, the first line of each cell is red intensity, the second line is green intensity and third line is blue intensity

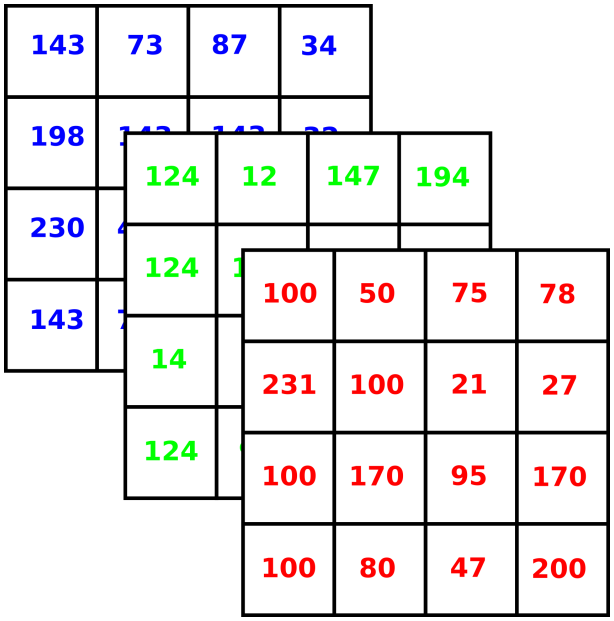


Figure 3.2: RGB image channels, each channel corresponds to different color: red, green or blue

3 Object Recognition

Object recognition consists of recognizing objects from a 2D image. Each object is defined by a region or a set of pixels.

Image Classification The most basic task is classification. The image given at this task contains or not an object of different predefined classes (e.g. car, person, table, dog). The task must predict the presence of the object and its class.

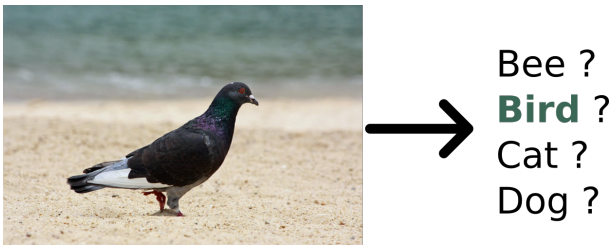


Figure 3.3: Image classification task: find the class of the input image

Object Localization A more advanced task is to detect the location of the object in the image in addition to its class. A rectangle called bounding box surrounding the object is predicted.

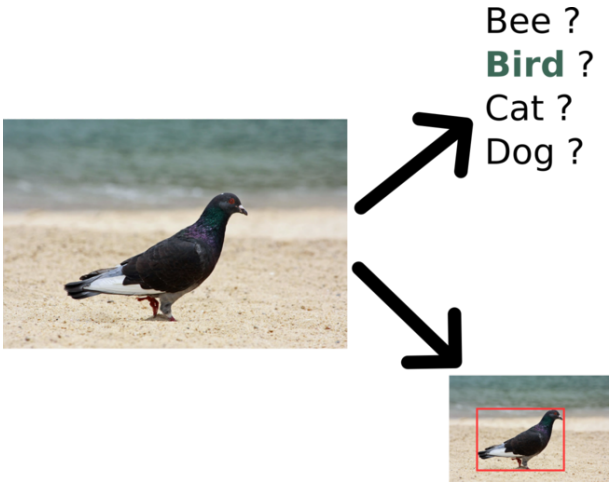


Figure 3.4: Image localization task: find the class of the input image and the location of the object

Image segmentation It is similar to object localization task, but rather than predicting a bounding box surrounding the object, the precise region of pixels corresponding to the object is predicted.

Object detection It consists in detecting multiple objects in the image. For each object, the task must predict its location by providing a bounding box and its class.

Instance segmentation It performs object detection, and predicts also for each object the region of pixels corresponding to the predicted object.

[1] https://www.researchgate.net/publication/337830238_Waterfall_Atrous_Spatial_Pooling_Architecture_for_Efficient_Semantic_Segmentation/figures?lo=1

[2] https://pixabay.com/photos/pets-cat-dog-animals-mammals-3715733/

Figure 3.5: Image segmentation task: find object pixels [1]

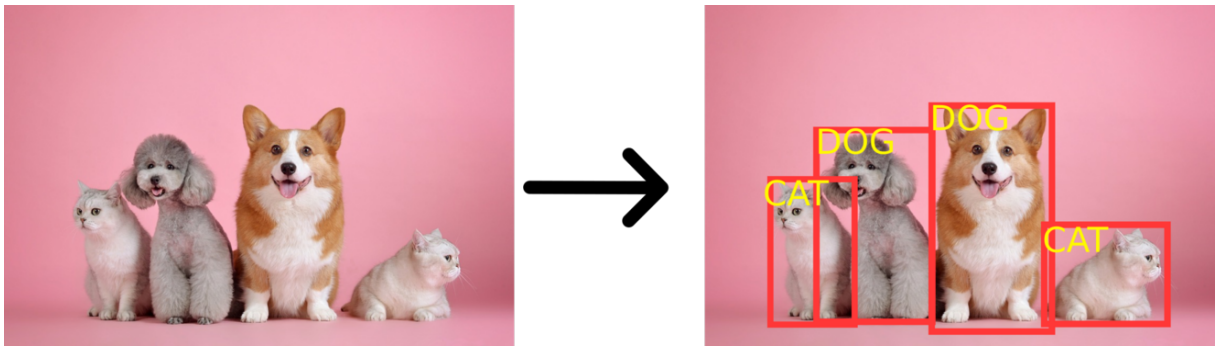


Figure 3.6: Object detection task: find object locations and their classes [2]

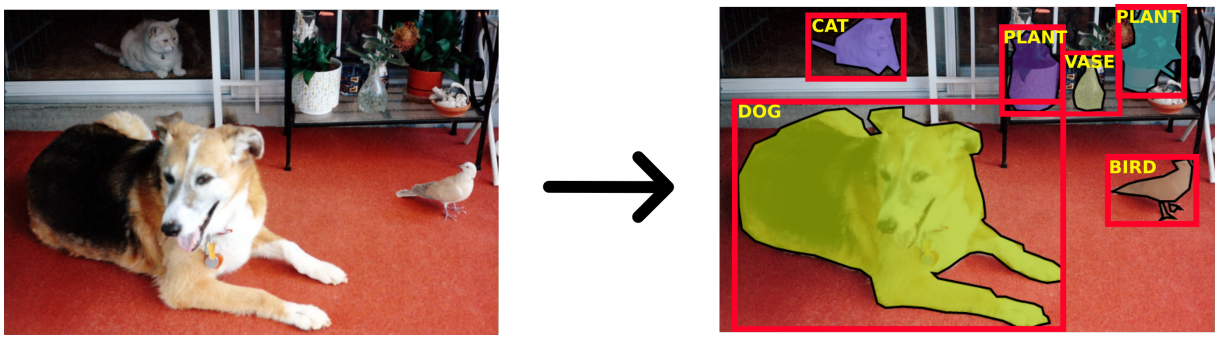


Figure 3.7: Instance segmentation task: find object locations, their classes, and their pixel regions

Semantic segmentation It consists in classifying each pixel in an image, into different classes.

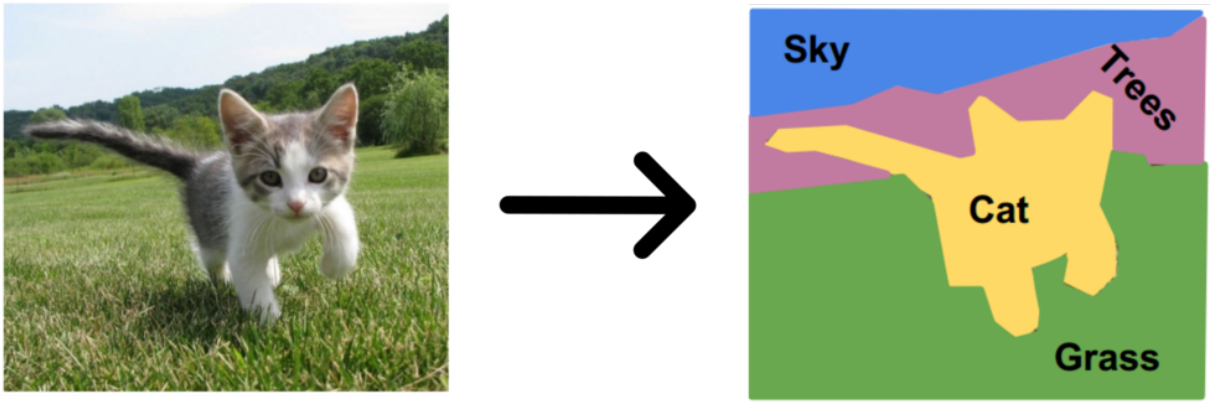


Figure 3.8: Semantic segmentation task: classify each pixel into different classes [3]

Panoptic segmentation It is composed of instance segmentation and semantic segmentation. There are two types of classes: the one used for classifying each pixel. And the ones used for classifying object instances.

Figure 3.9: Panoptic segmentation task: detect object instances and classify each pixel [4]

These tasks must process images to obtain the desired results, in Subsection 3.3.1, I explain some basic operations that can be applied on images, in Subsection 3.3.2, I present some traditional methods used for object recognition. In Subsection 3.3.3, I introduce the concept of feature extraction before explaining how machine learning is used to achieve object recognition tasks in Subsection 3.3.4.

[3] https://medium.com/@wongsirikuln/cassava-semantic-segmentation-eac8e90f1940
[4] https://github.com/facebookresearch/detectron2

3.1 Image Processing

The information extraction from an image could be impacted by some noises. In fact, in some circumstances, the quality of the input image could be degraded due to different reasons: shadows, lights, blur and so on. The image understanding could be more challenging in these kind of situations. In general, to deal with this issue, a preprocessing step is performed to enhance the quality of the image in regards to the desired task. In this section, I present some processing methods that can be applied to an image to enhance its quality and make it ready for information extraction.

3.1.1 Changing Color Space

The color space is the way of encoding colors. I have already presented the RGB space where red, blue and green components are used to represent any color. There are many other color spaces, among them there is HSV (hue, saturation and value) which is used in graphic domain because of its similarity to the human perception. There are equations which transform data from RGB space to HSV space and vice versa.

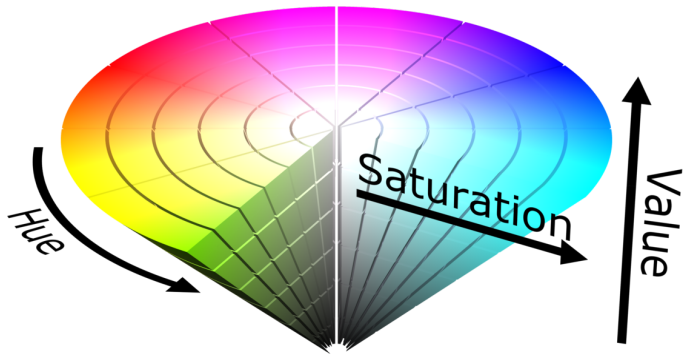


Figure 3.10: HSV color space [5]

3.1.2 Image Filtering and Gradients

Image filtering is an important task that is used for image processing. It is a function that is applied on the 2-dimensional data of an image, and outputs another 2-dimensional data. Let's consider an image of one channel (grayscale for example) I, $I(x, y)$ is the value at (x, y), ω a symmetric matrix of size $(2k+1)*(2k+1)$, and G a matrix defined by $G(x, y) = \Sigma_{i=-k}^{k}\Sigma_{j=-k}^{k}\omega(k+i, k+j)I(x+i, y+j)$. The G matrix is called 2D convolution of kernel ω. It is used to modify the distribution of the pixel values. A common example is to blur images [Li, 2021], this task

[5] https://www.pngwing.com/en/free-png-nhpas

consists in diminishing pixel value differences by using a kernel which calculates mean of each point and its surrounding. The opposite task is to increase differences by using gradient kernel. This kind of kernels (e.g. Laplacian, Sobel) calculates values depending on x and y partial derivatives which permits to extract differences between neighbour pixels. The gradient filters have different application; for example detecting object edges [Kanopoulos, 1988][Bhairannawar, 2018].

3.1.3 Image Thresholding

To eliminate noisy data in an image, a simple thresholding could be efficient. It consists in applying a simple rule on each pixel value based on its comparison with a predefined threshold (e.g. if the value is less than the threshold, the value becomes 0, otherwise it becomes 1) [Sahoo, 1988]. Thresholding steps are generally performed on grayscale images and are part of whole process which extracts valuable information from images.

3.1.4 Histogram Equalization

An image histogram is a distribution of colors in the image. It stores the number of occurrences of each color value. The contrast of an image can be seen as the differences of the colors in the image. When the contrast is high, the viewer differentiate colors more easily, and can therefore understand its details more easily. To improve the contrast of an image, I use histogram equalization method which stretches the image histogram [Patel, 2013].

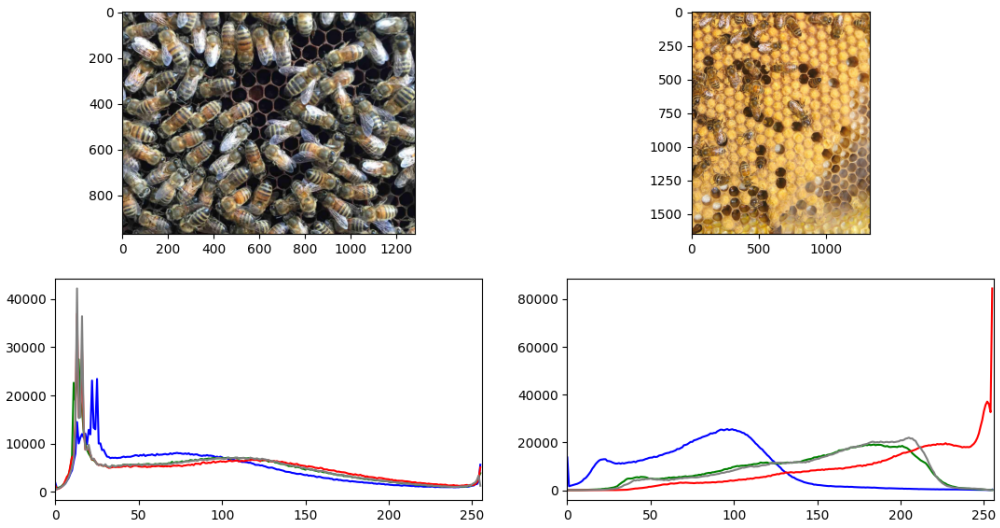


Figure 3.11: Image Histogram: each color corresponds to a different channel: Red, Blue and Green. Gray color corresponds to the image converted to grayscale image.

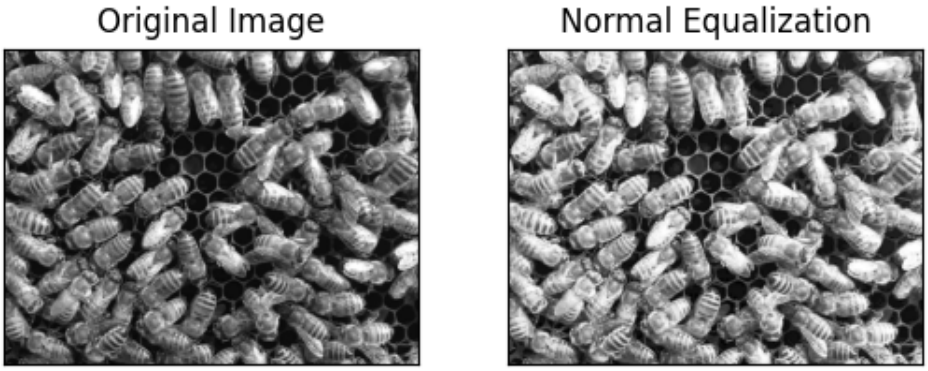


Figure 3.12: Histogram Equalization: it increases image contrast

3.1.5 Morphological Transformation

A morphological transformation is an operation based on shapes. For each pixel, the output value takes the value of its neighbour pixels based on a condition. Morphological operations are normally performed on binary images (each pixel value is 0 or 1, 0 for background and 1 for foreground). The region of neighbour pixels that the operation should check for each pixel is called kernel. The shape and size of the kernel are parameters of the operation. The basic morphological transformations are erosion and dilation. Erosion erodes away foreground boundaries. For each input pixel value, the output is 1 when all neighbour pixels values are 1, otherwise it is 0. Dilation is the opposite of erosion, it dilates foreground boundaries by using this condition: output value is 1 when at least one neighbour pixel value is 1, otherwise the output is 0 [Paris,].

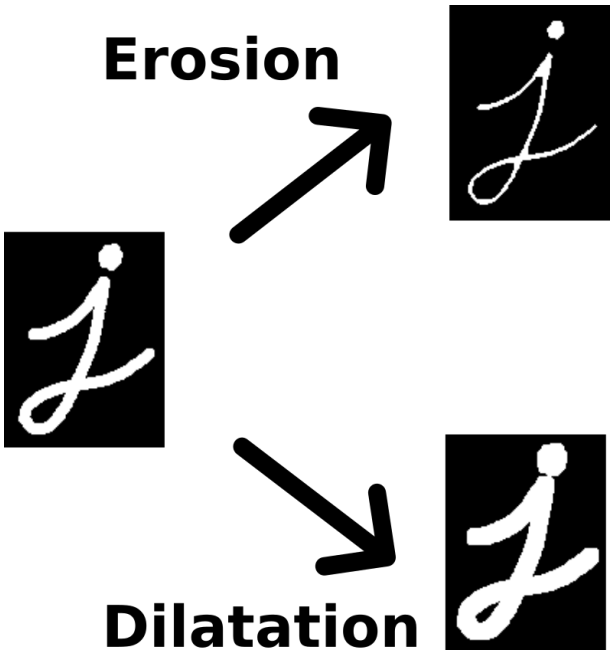


Figure 3.13: Erosion and Dilation operations [6]

By combining these two morphological operations, I obtain other morphological operations. Opening is an erosion followed by dilation, it is used for removing noises. Closing is a dilation followed by erosion, it is used for closing small holes in foreground regions [11]

Figure 3.14: Opening operation [6]

Figure 3.15: Closing operation [6]

3.2 Traditional Object Recognition Methods

After presenting object recognition tasks and describing some basic processing that are performed on images before recognition, in this chapter I show some existing non AI methods that perform this kind of tasks.

3.2.1 Template Matching

Template matching is a simple algorithm which detects a template in an image. It takes as input a template corresponding to an object to find in the image, and the image where the object should be found. It slides the template over the image. For each position, it performs a mathematical operation to evaluate the distance between the template and the position neighbor pixels. The output of template matching is a 2 dimensional map, each point value corresponds

[6] https://docs.opencv.org/4.x/d3/dbe/tutorial_opening_closing_hats.html

to the level of matching. The location detection is obtained by performing a threshold on the output [Cole, 2004].

3.2.2 Hough Transform

Hough transform is a method used for detecting parametric shapes in images. A parametric shape is a shape whose points are defined according to an equation of image coordinates x and y ($y = m.x + c$ in case of a line, m and c being the line parameters). The Hough transform is performed on binary image obtained in general by thresholding. The transform is based on calculating for each parameter combination the number of image foreground points that satisfy the shape equation. For example, in the case of a line, there are two parameters, the output of Hough transform is 2 dimensional matrix, each dimension corresponds to a parameter, each matrix value is the number of points passing by the line defined by the two parameters corresponding to value column and row. The shape detection is performed by taking matrix values that exceed a defined threshold, detected lines are obtained from the corresponding columns and rows. The same principle is performed to other shapes like circles. In standard Hough transform, for each parameter combination, all image foreground points should be checked, which may cause a lot of computations. To overcome this issue, a probabilistic Hough Transform was proposed, it takes in consideration a random subset of points [Fernandes, 2008].

3.2.3 Background Subtraction

Background subtraction is a simple method which detects foreground objects. It takes as input two images, the image corresponding to the background scene, and an image containing the objects to find, the background scene is subtracted from the object image, the result is thresholded to obtain pixels corresponding to objects only. This technique is generally used to detect moving objects in video captured by a static camera (e.g. pedestrians passing by a security camera) [Vacavant, 2012].

3.2.4 Watershed

Watershed is an algorithm used for image segmentation. An image of one channel can be considered as a topographic surface; it is defined by a function from 2D space to 1D space. A topographic surface can be viewed as mountains. The peaks correspond to local maximum and valleys correspond to local minimum. The idea of Watershed algorithm is to fill each valley with water of different colors. The water filling causes the increase of water level, when it reaches a local maximum, a barrier is built to avoid merging waters with different colors. These barriers are the results of the segmentation. The barriers are therefore located at local maximum, and should correspond to object boundaries. It is why Watershed is applied on the gradient of the original image.

In general, an object is more complex than a surface of unique color, so it contains many local maximum delimiting small regions in the object. The standard watershed would segment these regions. To deal with these noises and irregularities in images, an alternative algorithm

was proposed: marker-based Watershed. In this method, the valleys that can be merged are marked with the same label, Watershed fills water starting from these marked regions, it uses a different color for each label.

To illustrate how watershed is used, let's take an example of an image containing some objects. I want to find the pixels corresponding to object regions. These steps can be performed to obtain the desired segmentation [Acharjya, 2012]:

- A threshold is applied to obtain a first binary image. I assume that the background has lower intensity than foreground.
- Opening operation is performed to remove noises.
- Closing operation is performed to close holes.
- A dilation operation is performed to find background.
- An erosion or distance transform operation is performed to find foreground.
- At this step I have regions that I know for sure that they correspond to background, and regions that I know for sure that they correspond to foreground. Of course, the background regions are labeled with the same label. To identify each object pixel, a simple algorithm is applied to obtain connected components, each labeled with a different color.
- Watershed is applied to obtain segmentation.

3.3 Feature Extraction

Features are relevant information extracted from raw data to perform certain computational tasks. More precisely, in the domain of computer vision, features correspond to information describing the content of the image. In fact, as we have seen earlier, an image is made of raw values representing the intensity of colors in each pixel. This information is usually not relevant to perform specific computer vision tasks like image classification or object detection. Therefore, depending on the task to fulfill, specific features are extracted to get more interesting information which is used to give the final result.

Features of an image could be global (information about the whole image) or local (the properties for each region). For example, in object recognition tasks, a relevant feature could be object corners as human vision is naturally based on this kind of data (among others) to detect objects in visualized scenes.

3.3.1 SIFT

An object could be detected by finding the relevant points that define it. These points are called key points. SIFT (Scale-invariant feature transform) algorithm extracts key points from an image and calculates their characteristics to obtain features. The relevant points of an object

[7] https://www.kittenoverload.com/

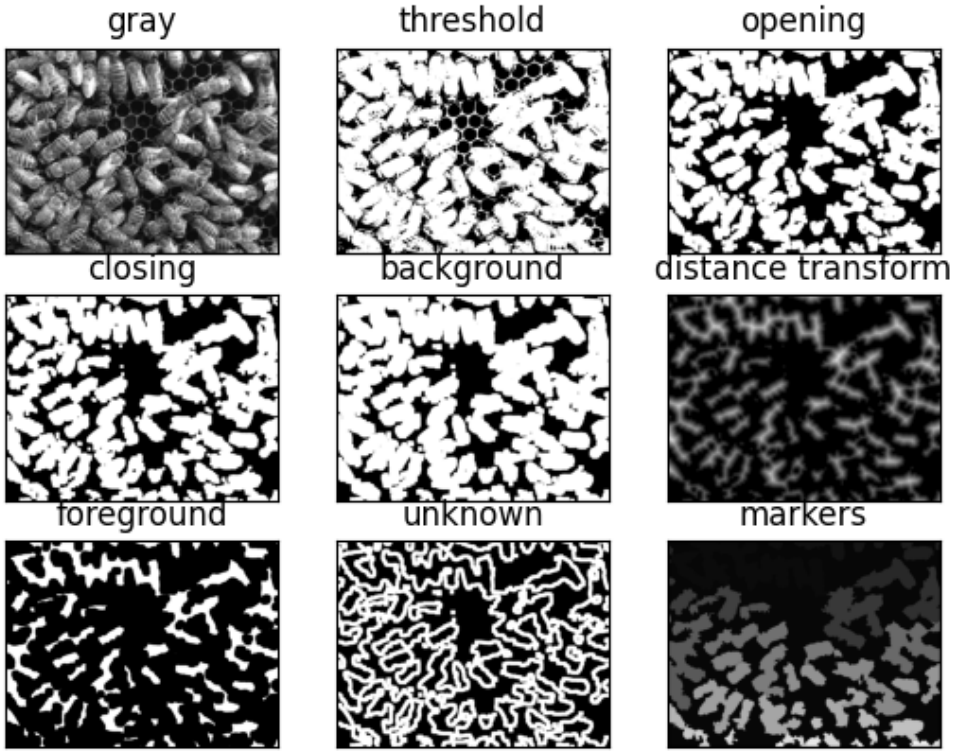


Figure 3.16: Classic steps performed to apply Watershed algorithm. "gray": input image in grayscale. "threshold": result after performing threshold to extract foreground. "opening": result after applying opening transformation to remove noise. "closing": result after applying closing transformation to close holes. "background": result after applying dilatation transformation to obtain sure background region. "distance transform": result after applying distance transform to obtain sure foreground. "foreground": sure foreground region obtained after applying threshold on last result. "unknown": region that should be marked by Watershed, it is simply the difference between sure foreground and sure background. "markers": the segmentation result obtained after applying Watershed algorithm

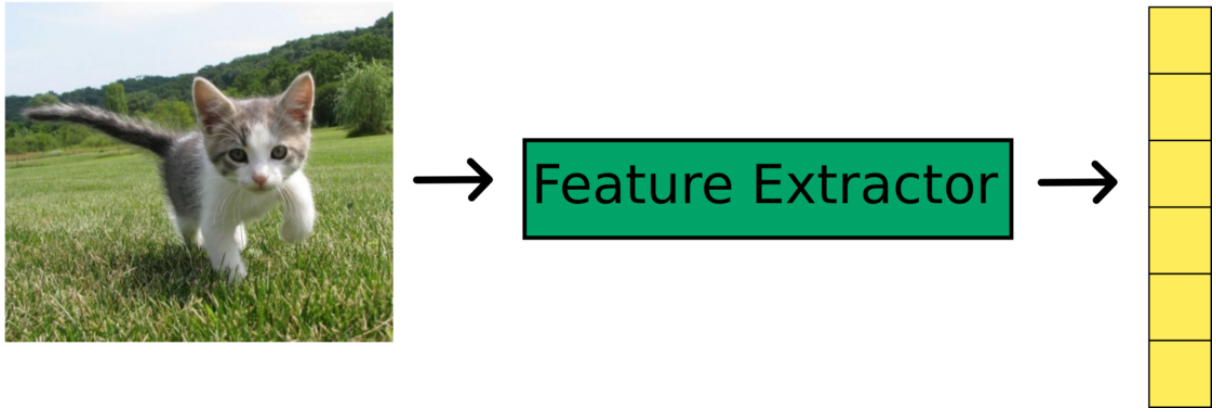


Figure 3.17: A Feature extractor takes as input a 2D image, and outputs a vector of one dimension or a matrix representing the relevant features of the image. In this schema, the features are a vector of one dimension. [7]

are in general points where the difference in intensity is high. To find them, a Difference of Gaussians (DoG) is used. A Gaussian of an image is the result of applying a gaussian filter to it. It has a parameter σ which is the radius of the smooting. Indeed, a gaussian with parameter σ smooths the image and blurs it. It suppresses its peaks whose radius are less than σ. Difference of Gaussians is the difference of two gaussians with different σ, the result keeps the peaks whose radius are included between these two filter parameters. In fact, SIFT, uses a succession of σ, each used for detecting peaks of different scales. The algorithm gets key points by finding maximum local points.

Then, an orientation is assigned to each key point. A neighborhood is considered for each key point, gradient vectors are calculated for each point in the neighborhood. Each vector has its direction, the space of orientation is quantized to 10 intervals, and an histogram is calculated from the direction occurrences, the orientation of the key point is the one with the largest number of occurrences. To have more robust information, this calculation of orientation histograms is repeated for the neighbour points (in this case orientation is calculated compared with respect to the key point orientation), and the final key point descriptor is the concatenation of histogram data [Lowe, 1999][Pascal Monasse, 2022].

3.3.2 Feature Matching

Features are used to perform tasks related to object recognition. For example, they can be used to predict the location of an object in an image. I consider two images, the first one corresponds to a scene, and the second one corresponds to an object to be found inside the scene. Key point features are extracted from each image, and a comparison is performed between each couple of features from the global scene and the query object. The best assignment corresponds to the object found.

[8] https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

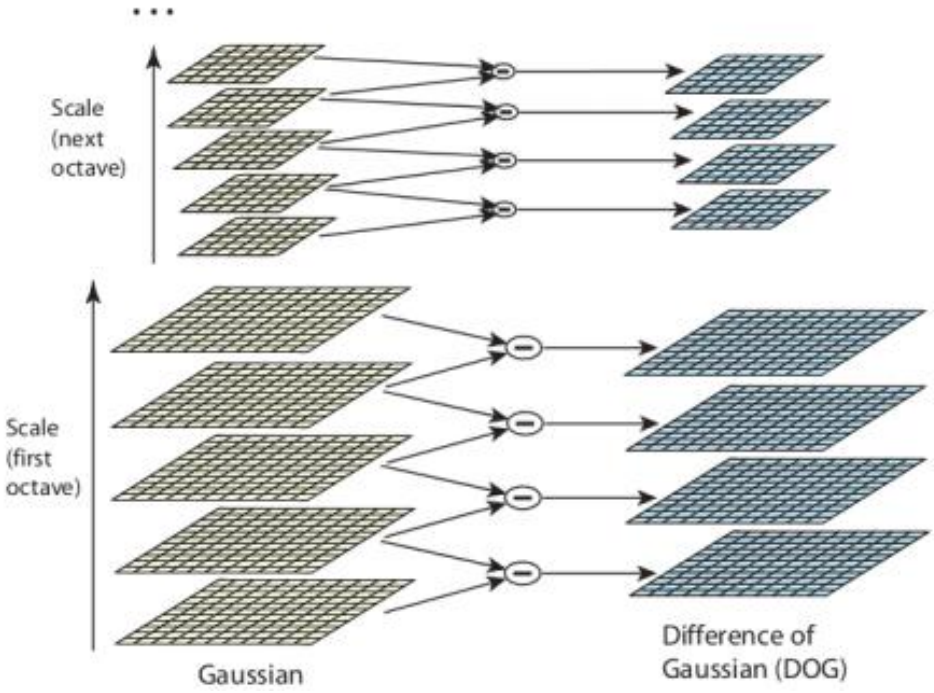


Figure 3.18: SIFT algorithm: succession of DoG [8]

Figure 3.19: SIFT algorithm: example of image key points [8]

3.4 Machine Learning

Feature engineering consists in designing and implementing methods for feature extraction. The extracted features are given to algorithms which perform the desired tasks. A very common class of methods that solve these needs is machine learning. They are particularly relevant in the case where enough data exist. The objective of a machine learning algorithm is to learn how to solve the task based on existing data. In case of computer vision and image understanding, these data are images and eventual annotations. This set of data is called training set, the annotations are in general the correct results of the task applied on the training images. They are called labels and are usually provided through a human labeling step. Supervised learning deals with problems with labels, and unsupervised learning deals with problems with unlabeled data.

3.4.1 KNN

It is a simple supervised learning algorithm dedicated to classification tasks. There is a set of training images, for each image we know its correct class (called ground truth). The algorithm should predict the class of a new image. First, features are calculated for each training image and stored with their corresponding labels. To predict the class of a new image, k-Nearest Neighbour calculates its features and finds the k (k is a predefined parameter of the algorithm) nearest training features to it by using a defined metric like the Euclidean distance, the result label is the one that is repeated the most among the labels corresponding to the retrieved features. The user must choose or design the best feature extractor according to the desired task.

3.4.2 SVM

Support Vector Machines or SVM is a well known machine learning algorithm for supervised learning. It is used for classifying data points into classes. It deals with linearly separable data. We consider some data included in $\mathbb{R}^N$, each element is assigned a class among two classes, the elements of this set are linearly separable when there is an hyperplane that separates the elements of each class.

The objective of SVM is to find this hyperplane using a specific mathematical equation and an algorithm. During the inference time, it can predict the class of unseen data. In the case of computer vision, the data used are features extracted from images, this extraction is carefully chosen by the user according to the task that should be solved. There are other approaches based on SVM that can be used in case of many classes and not linear problems.

3.4.3 K-Means Clustering

K-Means is a clustering algorithm used for unsupervised learning. We consider point data, the objective is to find clusters or groups that separate well the data. This means that the points in each group have relevant similarities with each other that they do not have with elements of

[9] https://dataanalyticspost.com/Lexique/svm/

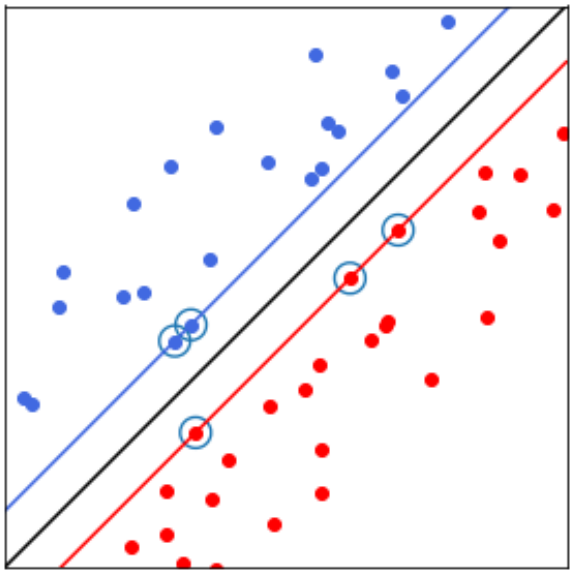


Figure 3.20: SVM algorithm [9]

other groups. K-Means is an iterative algorithm, at each step, it takes the clusters generated from the previous step, and for each one of them it calculates its centroid, then it generates a new set of clusters; for each centroid it creates a cluster and fills it with the closest points to the centroid. In the first step, centroids are chosen randomly from point data. The number of clusters is a parameter of the algorithm. Again this algorithm is applied on features extracted from images in case of image understanding. It can be used for example to classify images without knowing their ground truth labels.

3.4.4 Neural Network

In the three previous algorithms, image feature extraction is needed before the application of the algorithm. In fact, in this kind of approaches, the challenge is to find the relevant extractors for the desired task. Neural Networks are complex operations that are applied to input data to preform predictions by automatically extracting features. We consider a vector V belonging to $\mathbb{R}^n$ (n is the size of vector V), and a matrix W of size $p \times n$, the result of this multiplication $W.V$ gives another vector O belonging to $\mathbb{R}^p$ (p is the size of output vector), this operation is the main operation in neural networks. The vectors are called features or layers. To avoid ambiguity, I will use the term "features" to mean the output vector of the matrix multiplication, and the term "layer" to mean the matrix multiplication operation itself. Each feature dimension value O_i is the result of multiplication of the i^{th} line of W and the vector V, this operation can be represented by a node (called also neuron). Each node is connected to all input dimensions, and has one output, the concatenation of all outputs give the final layer output. The matrix values are learnable parameters of the neural network. They are called weights. These layers can be stacked to obtain a multilayer neural network. In general, real models are composed of

[10] https://www.oreilly.com/

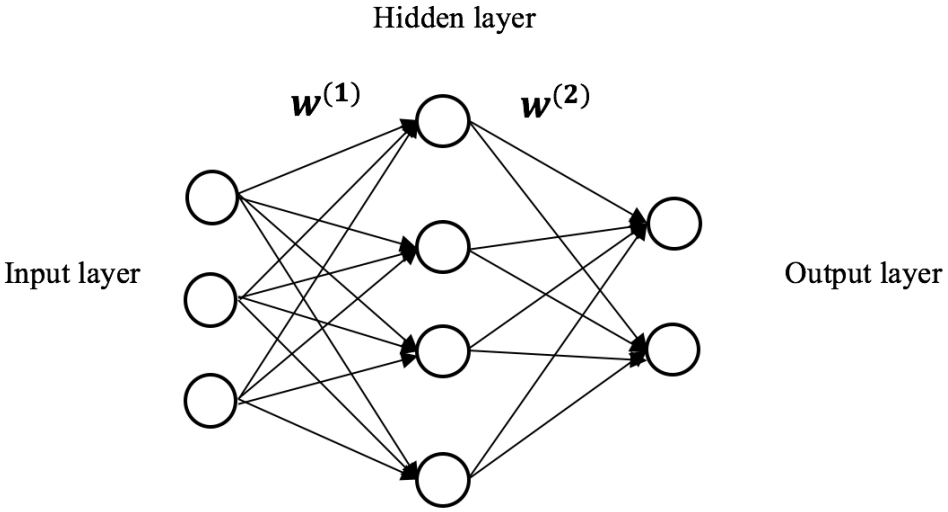


Figure 3.21: Neural Network composed by three layers: input layer, hidden layer and output layer. The input layer provides the input data. The hidden layer calculates intermediate data by multiplying matrix $W^{(1)}$ and input data. The output layer calculates the output data by multiplying matrix $W^{(2)}$ and intermediate data [10]

many consecutive layers, in this case, we deal with deep structures, unlike shallow models. This kind of structures are the origin of the term "Deep Learning".

3.4.5 Biases

Biases are parameters used to calculate neuron outputs. Let b be a vector in $\mathbb{R}^p$, the layer output is $W.V + b$. As matrix weights, biases are learnable parameters of the neural network.

3.4.6 Activation Function

The matrix operations are linear, but in general problems that we solve are not linear, it is why other non-linear operations are applied on the result of the matrix multiplication. These functions are called activation functions. The function is applied on each neuron output.

3.4.7 Training

The final result of the operations is information that we want to predict (classification or segmentation for example). To perform the learning, a function called loss is applied on the neural network results and ground truth data, this function is chosen so as to represent the distance between two information. The objective of the learning is to minimize the loss by changing parameter values. There are different algorithms for minimizing loss quantity, but they use the same mechanism; backward propagation. At each step, the forward pass is performed, it means that the operations are applied on the input data. Then the loss is calculated to compare the final output and ground truth information, and the backward propagation is launched: the aim

[11] https://medium.com/@aborundiya/activation-function-for-multi-layer-neural-networks-a07ac473f69e

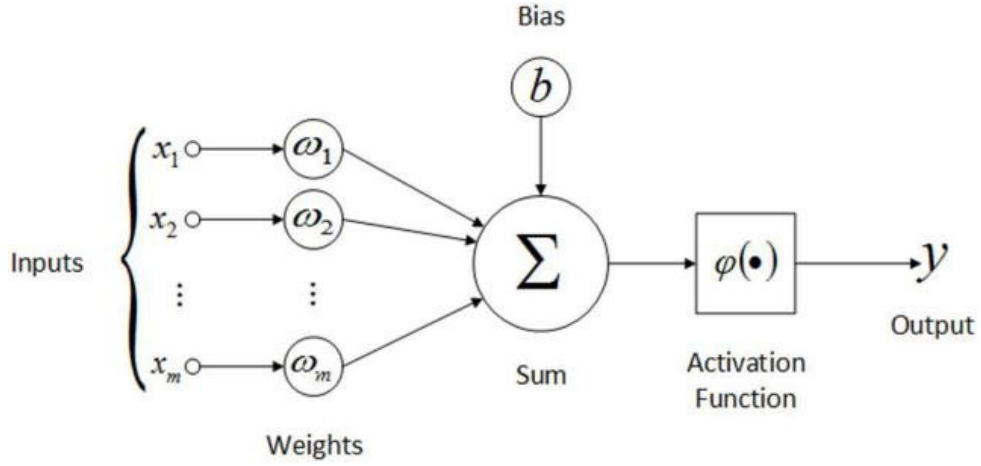


Figure 3.22: Simple neural network of one layer, the output dimension is one, the activation function is applied after applying weights and biases on the input [11]

of this phase is to calculate partial deviations of the loss compared to the network parameters, to accomplish that, it calculates the derivatives of the last layer, then the previous one are calculated based on the last one, and so on. The parameter updates are performed according to calculated derivatives.

3.4.8 Convolutional Neural Network

The previous neural networks are fully connected; each node of a layer is connected to all nodes of the next layer, and for each connection there is a parameter that should be learnt. This rapidly multiplies the number of parameters in the neural networks and make it harder to train and store in machines.

In the case of images, if we consider each pixel a node of the neural network, in a layer, the information that we want to extract from each pixel region is likely the same as the information we want to extract from another pixel region (corners, edges, object presence), this is why matrix parameters should be the same. Fully connected layers use different parameters for each connection.

To solve these two issues, convolutional neural networks have been proposed, instead of simple matrix multiplication, a convolution is calculated on the input data, the parameters to learn are the convolution parameters. We consider a 2D data I of C_{input} channels (3 in case of RGB image), I_c is the c^{th} channel, an output 2D data O of C_{output} channels, a list of kernels $K_{(0\leq o<C_{output},0\leq i<C_{input})}$, and a list of biases $b_{(0\leq o<C_{output})}$, O is the the result of the convolution using kernels K on I when for each o in $[0, C_{output}[$ $O_o = b_o + \sum_{i=0}^{C_{input}} K_{(o,i)} * I_i$, where "$*$" is the convolution operator defined by $(W * \omega)(x, y) = \Sigma_{i=-k}^{k}\Sigma_{j=-k}^{k}\omega(i, j)W(x - i, y - j)$ (W is a matrix and ω is a kernel matrix).

This kind of operation has the characteristic to be spatial invariant, this means that the applied operation does not depend on the region where it is applied. Convolutions are particularly used to extract spatial features from images (corners, edges, specific patterns). In practice, neural networks processing images are composed of a succession of convolution layers, the first

extracting detailed data such as edges, the last layers extracting more semantic data such as object presence. In some cases, depending on the task to perform, fully connected layers are used after convolutional layers to extract global information from images used for example for classification tasks. Therefore instead of using a static algorithm for extracting features, the learning algorithm find the best extractor parameters, only the architecture of the neural network should be designed by the user.

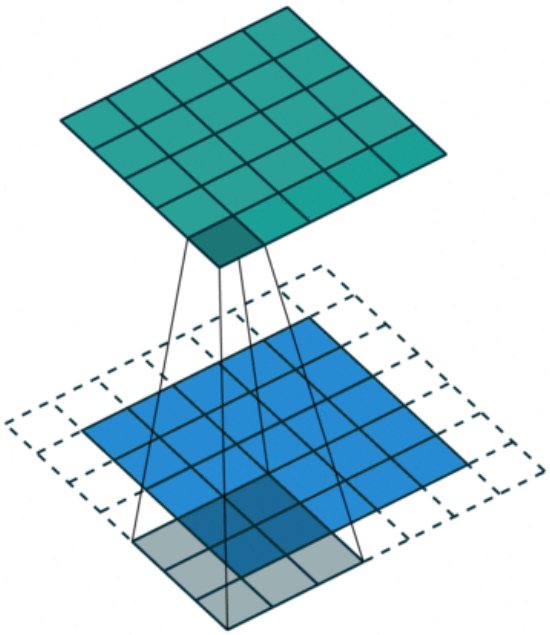


Figure 3.23: Illustration of convolution operation on 2D data. To be able to perform convolution on the matrix ends, a padding is added to input depending on kernel size. [12]

Among the main layers that are used in neural networks for image processing are the pooling layers. They are used to reduce the size of data and noises, the layer takes as input a feature map, it divides it into a defined number of regions according to the desired output size. For each region it aggregates its information using a defined function (maximum, average ...). This operation is performed for each input channel. As output, we obtain a feature map of same number of channels and reduced in size.

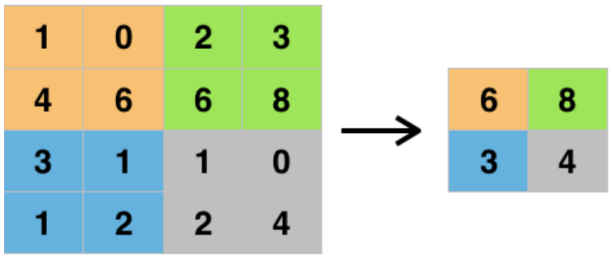


Figure 3.24: 2D Max Pooling [13]

[12] https://medium.com/analytics-vidhya/

3.4.9 Neural Network for Image Classification

The most basic task related to image processing using machine learning is image classification. To perform this task, a neural network composed of convolution layers and fully connected layers is used. The input image is given to neural network, it passes through a stack of convolutions which extract relevant features. This information is then given to a fully connected layer for classification into different classes. In general each convolutional layer is followed by an activation function like ReLU and a pooling layer to reduce size.

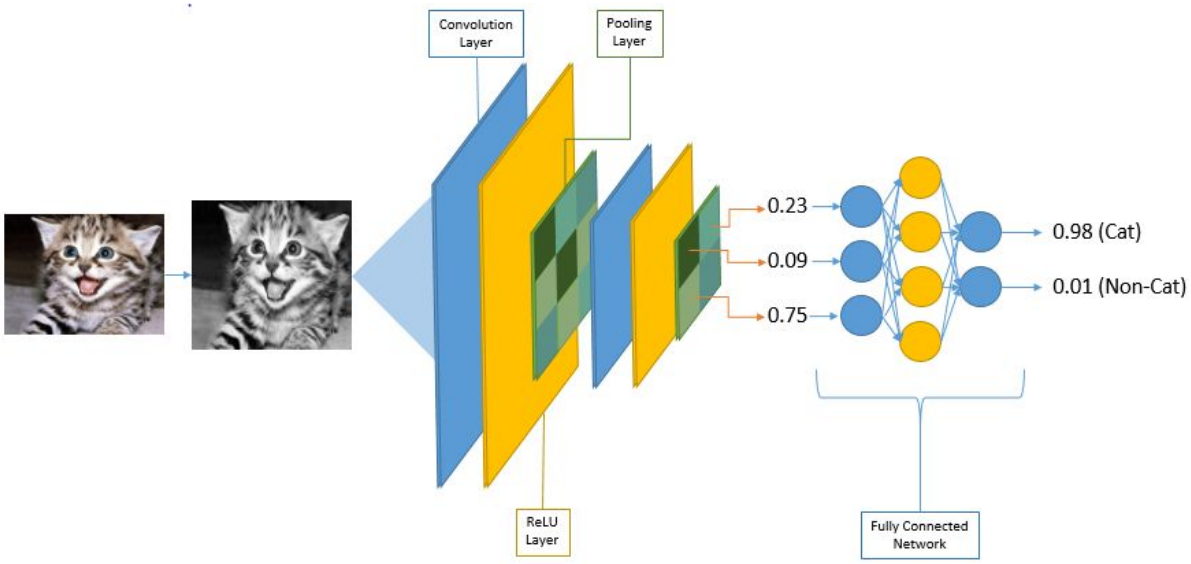


Figure 3.25: A neural network for classifying images into "Cat" and "Non-Cat" according to whether the input image contains a cat or not [14]

4 Object Detection using Machine Learning

After explaining how deep learning and neural networks are used to perform computer vision tasks, I focus in this section on existing neural network architectures proposed for object detection tasks. There are two types of object detectors: two stage detectors presented in Subsection 3.4.1, and one stage detectors presented in Subsection 3.4.2. As my research is based also on segmenting objects, I talk about this subject in Subsection 3.4.3. Then I define transfer learning in Subsection 3.4.4, as it is an important concept used in object detection and segmentation. Finally, in Subsection 3.4.5, I concentrate on Faster R-CNN neural network architecture. My research is based on extending this architecture.

4.1 Two Stage Object Detection

Two stage object detection is an approach of detecting objects using two phases ; generate proposals then classify them.

[13] https://betterprogramming.pub/introduction-to-deep-learning-part-ii-f37f2794715a
[14] https://www.codeproject.com/Articles/5160467/Image-Classification-Using-Neural-Networks-in-NET

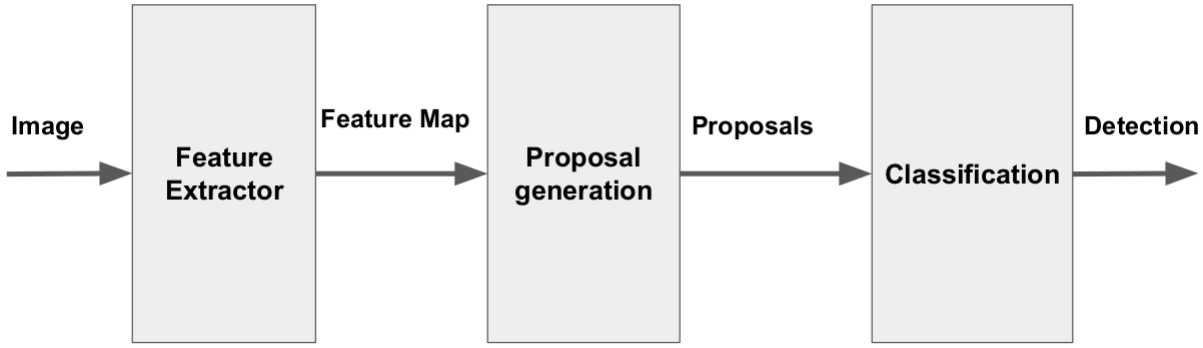


Figure 3.26: General Structure of two stage object detector

4.1.1 R-CNN

R-CNN is based on the selective search vision algorithm which generates proposals that are provided to a convolutional network which checks whether proposal are real objects and performs a classification [Girshick, 2014]. But this detector suffers from some limits:

- Feature extraction should be performed for each proposal in the image, which makes the prediction slow.
- Proposals are warped to respect the CNN input size constraint. Warping could lead to information loss and bad prediction.
- Two different structures are used, a neural network for extracting features and SVM to classify features into output classes. Therefore two training must be performed.
- Selective Search is a non trainable algorithm. The proposal generation is therefore not adaptable to the type of images.

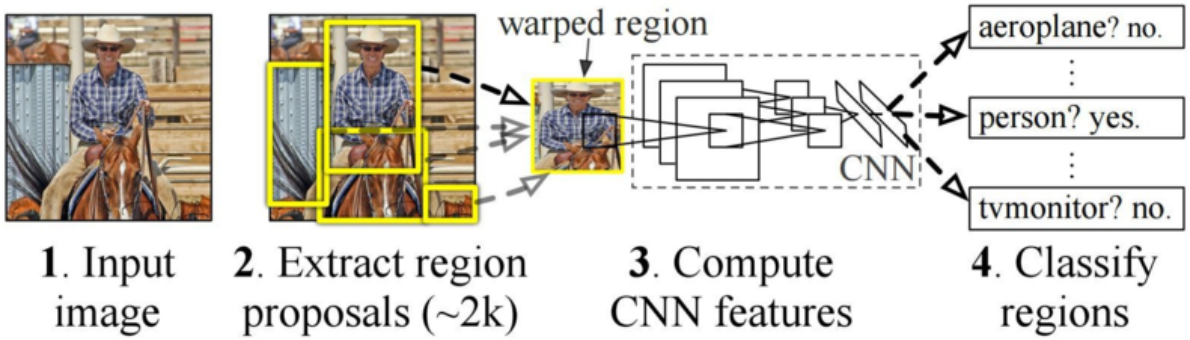


Figure 3.27: R-CNN neural network [Girshick, 2014]

4.1.2 Fast R-CNN

Fast-RCNN improves object detection by using a ROI pooling layer responsible for extracting features from shared feature maps instead of calculating feature map for each proposal [Girshick,

[15] https://learnopencv.com/selective-search-for-object-detection-cpp-python/

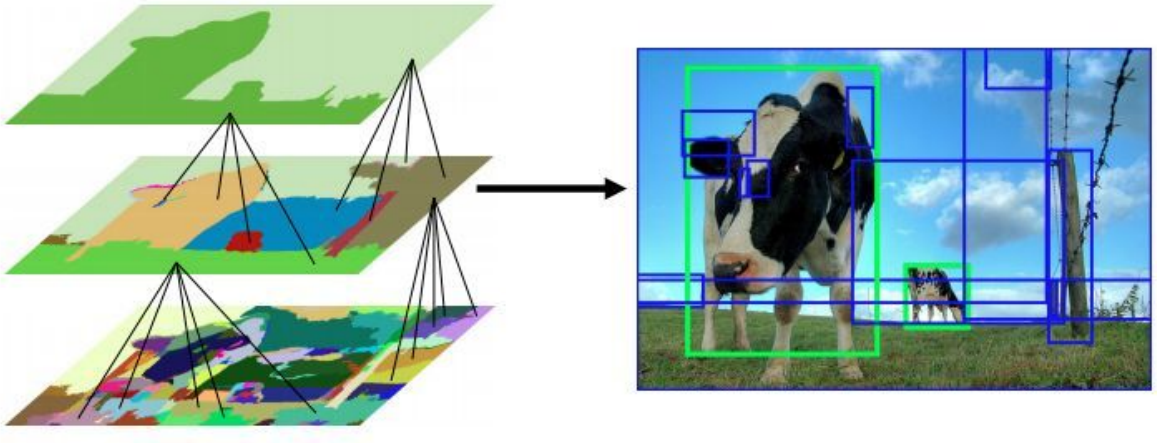


Figure 3.28: Selective Search iterative algorithm [15]

2015]. In addition, it uses a Softmax layer for classification instead of SVM to have one end to end trainable neural network.

4.1.3 Faster R-CNN

Faster R-CNN uses the RPN neural network as a proposal generator [Ren, 2015]. This architecture allows learning to predict object bounding boxes, without relying on a static computer vision algorithm like selective search.

4.1.4 R-FCN

R-FCN is a two stage object detection neural network built only by convolutional layers, the principle is to use many feature maps, each of which contains information about different object regions [Dai, 2016].

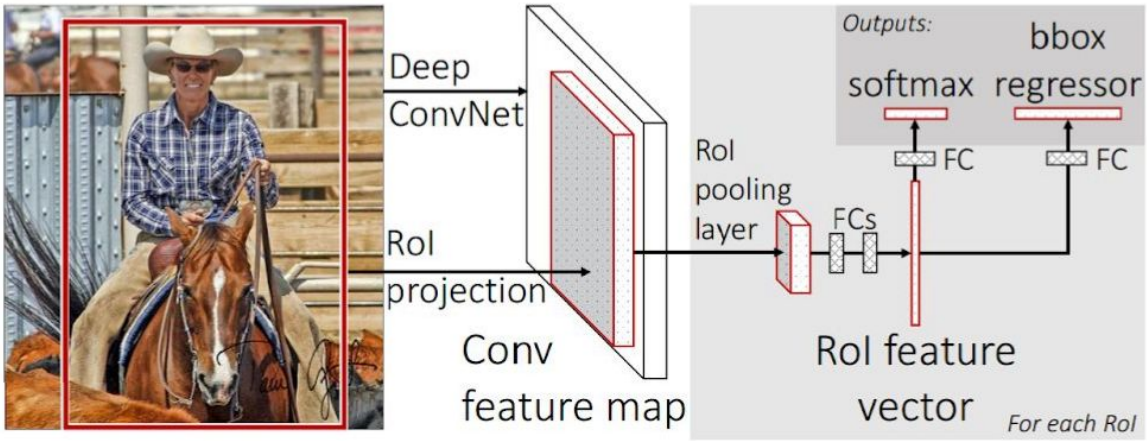


Figure 3.29: Fast R-CNN neural network [Girshick, 2015]

4.2 One Stage Object Detection

One stage object detectors predict object boxes and classes directly from the image features without generating proposals. Therefore, one stage object detectors are faster and less resource greedy than two stage detectors, but in general less accurate.

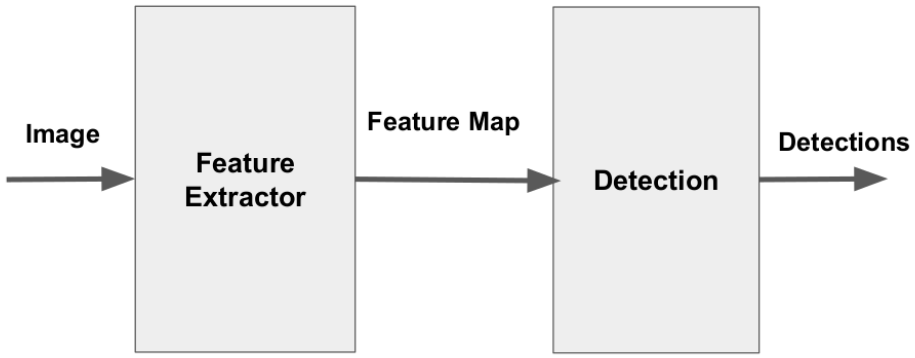


Figure 3.30: General Structure of one stage object detector

SSD and YOLO are the state-of-art neural networks that detect objects without generating proposals [Liu, 2016][Redmon, 2016]. The drawback of these approaches is the imbalance between positive and negative examples during training, which leads to limited performance. To overcome this issue, RetinaNet has been proposed. Their authors suggest to use Focal loss which increases the loss of misclassified object compared to well classified objects [Lin, 2017b]. CornerNet [Law, 2018] is an approach based on using heatmaps to predict corner embeddings that are compared to each other to find corners of same objects. CenterNet detects the centers of the object, from these center points it predicts bounding box offsets [Zhou, 2019a].

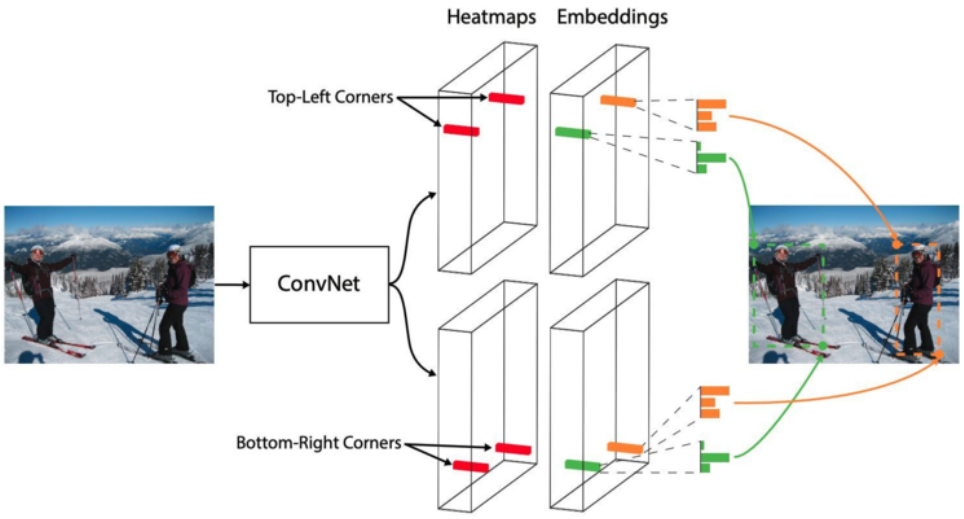


Figure 3.31: Corner Net neural network [Law, 2018]

4.3 Segmentation

As seen previously, there are different types of segmentation: semantic segmentation, instance segmentation and panoptic segmentation. There are different neural network architectures proposed to deal with these tasks. Among the most common, there is Mask R-CNN [He, 2017] dedicated to instance segmentation, it is an extension of Faster R-CNN, it adds a branch that performs segmentation for each detected object. FCN [Long, 2015] (Fully Convolutional Network) performs semantic segmentation, it uses a classification neural network structure, but replaces fully connected layers by convolutions, and uses increasing layers like unpooling or deconvolution [David, 2016]. U-Net [Ronneberger, 2015] is a well known architecture used mainly in the medical field. It has a symmetric structure, the input image passes through classic convolution layers to produce a feature map of reduced size, this feature map passes through increasing layers to reach a size comparable to the input size, then each point is classified according to whether it corresponds to an object region. To have better prediction, a residual structure is used: the increase layers use generated outputs by downsampling layers as inputs.

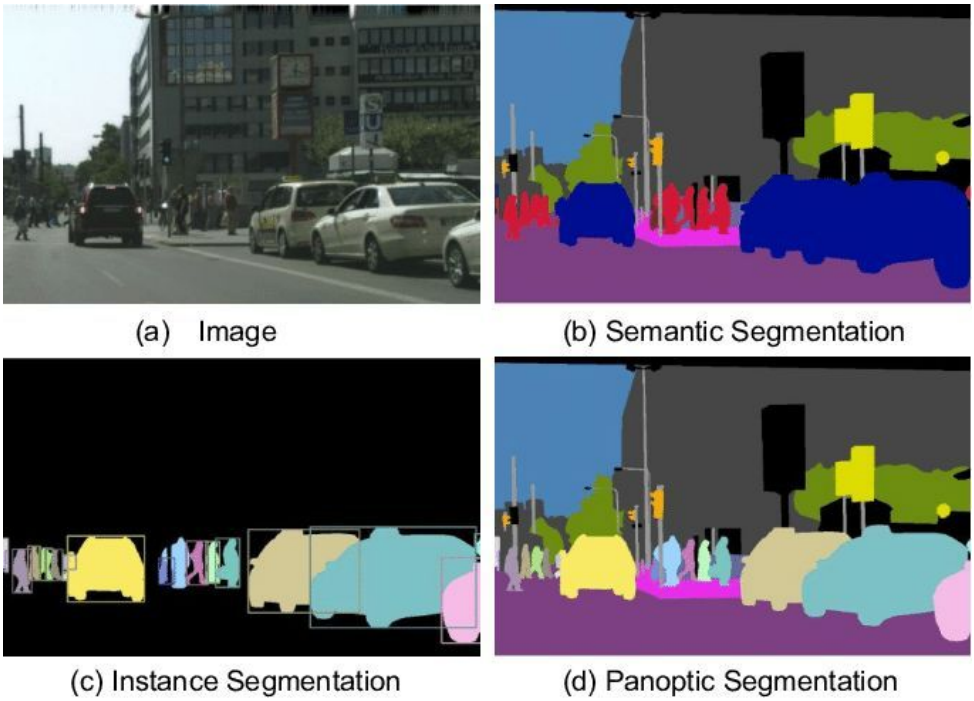


Figure 3.32: Segmentation types [16]

4.4 Transfer Learning

Neural networks are huge structures, the number of trainable parameters can reach millions. Training all these parameters requires large datasets. In the case of tasks that contain very limited datasets, already pre-trained networks are used. A pre-trained network is a network that has been trained using huge datasets on a classification task. Transfer learning consists in

[16] https://www.researchgate.net
[17] https://towardsdatascience.com/nucleus-segmentation-using-u-net-eceb14a9ced4

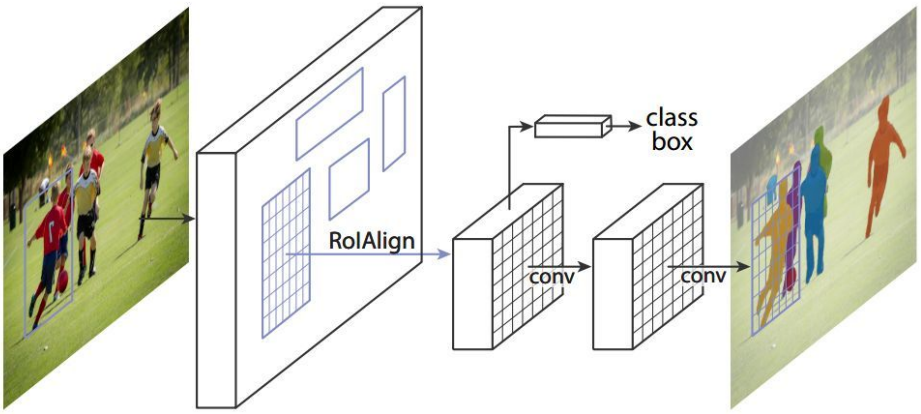


Figure 3.33: Mask R-CNN Neural Network [He, 2017]

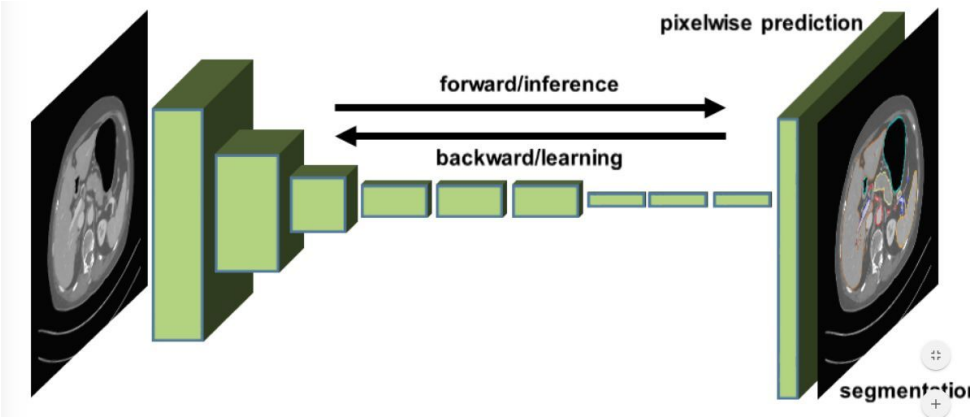


Figure 3.34: FCN neural network

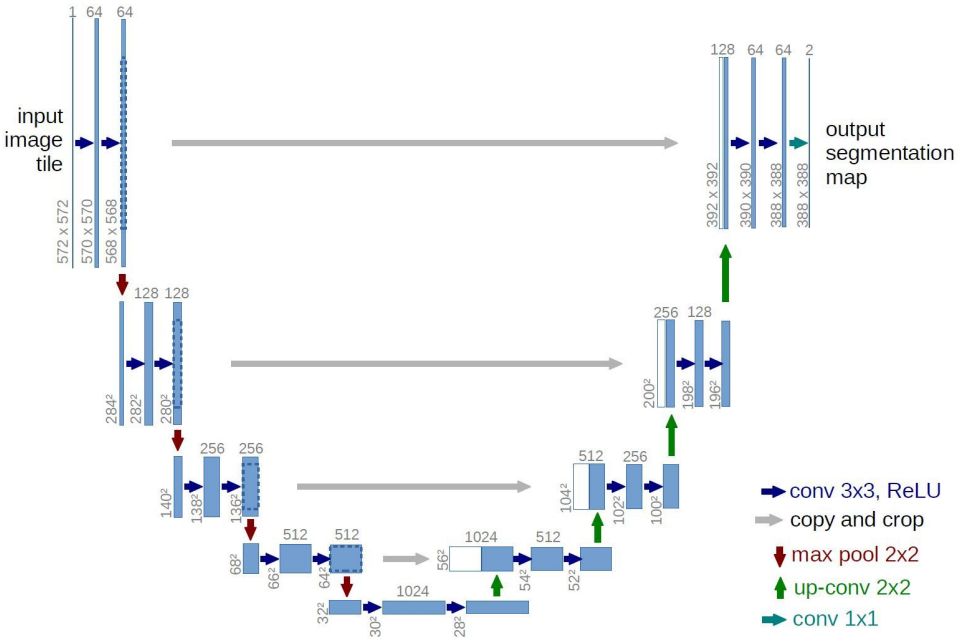


Figure 3.35: U-Net neural network [17]

recovering the values of the parameters obtained after this training, the parameters of the new model to train on the new task are initialized with these values. In general the first layer values are fixed since they extract generic information from images like edges, whereas the final layers extract information specific to the desired task.

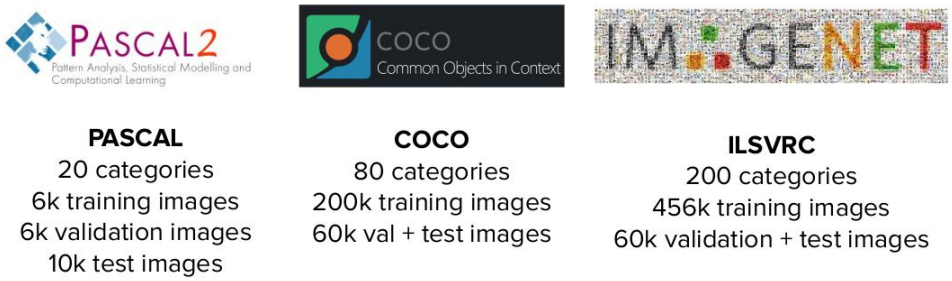


Figure 3.36: Public image datasets

4.5 Faster R-CNN Neural Network Architecture

As my research is mainly based on using Faster R-CNN and adding modifications to it, I will explain this neural network more in details in this subsection.

4.5.1 General Architecture

Faster R-CNN is a two stage object detector. It is composed of several components. First, the input image is processed by a backbone neural network which is responsible of extracting relevant features. To achieve a good training, transfer learning is used, an already pretrained neural network like ResNet [He, 2016] is used as backbone. In training phase, the parameters of the network are updated to fit the specific detection task. After extracting the feature map from the image, the result is passed to the regional proposal network (RPN), it generates proposals from the feature map. To perform this generation, an intermediate convolutional layer is applied, then two pieces of information are predicted, for each feature map point, a classification layer predicts a score corresponding to the probability that the point corresponds to an object, a regression layer predicts four offset coordinates which are used to infere the bounding box position. These two pieces of information are combined to generate proposals. Each proposal is a bounding box corresponding to a potential object. Proposals are then transferred to another neural network. For each proposal, ROI pooling is performed to extract the corresponding region of the proposal from the feature map produced by backbone, the output of this pooling is a feature map with a fixed size. This information is processed by two consecutive fully connected layers to obtain flat features of one dimension, the result is processed by a classification layer to filter only relevant proposals, and a regression layer to correct the coordinates of the bounding box in the image.

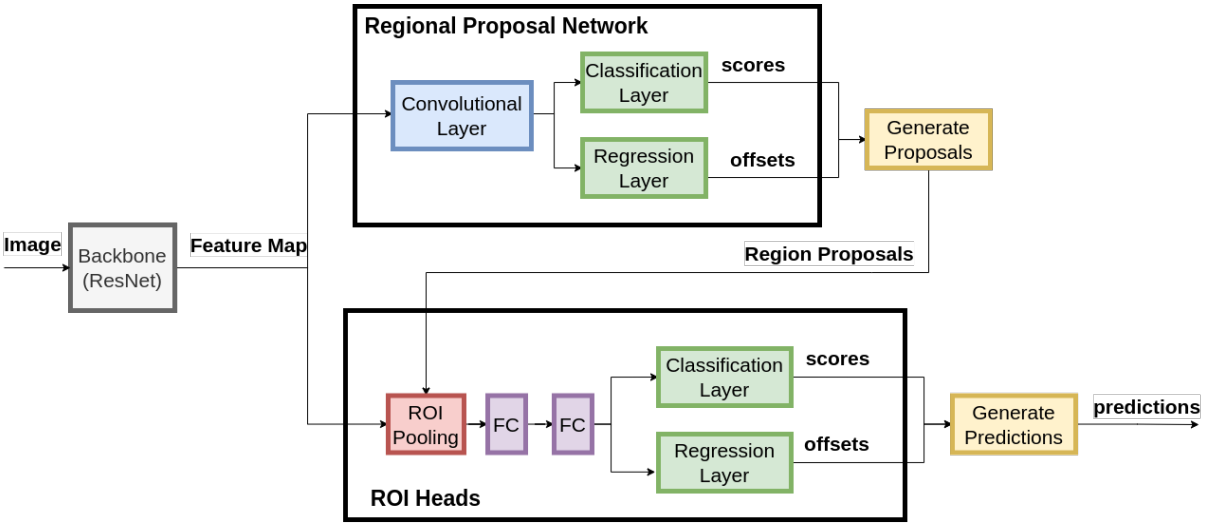


Figure 3.37: Faster R-CNN architecture

4.5.2 RPN

RPN or Regional Proposal Network generates from a feature map several proposals. A set of bounding boxes of defined sizes and ratios is defined. These boxes are called anchors. For each feature map point and for each anchor, RPN predicts two pieces of information, a score and 4 offset coordinates. This is performed through a convolutional layer which calculates the spatial information for each point, then two convolutional branches predict scores and offset coordinates.

4.5.3 ROI Pooling

As seen earlier, pooling operations aggregate feature map information into a smaller feature map. Region Of Interest Pooling aggregates bounding box information extracted from the feature map to a fixed size feature map. In fact, in deep learning, due to training and fully connected layer constraints, output size of each layer should be the same regardless of the input data. Proposals have in general different sizes, for each one of them, the ROI pooling extracts the corresponding region from the feature map extracted by the backbone, taking into account the ratio between image size and feature map size since the proposal coordinates are expressed in the image coordinate system. Each region is subdivided into $N \times N$, each subregion is aggregated to one value using average or mean function, the output is a feature map of size $N \times N$.

4.5.4 Feature Pyramid Networks

Objects in an image can be of different sizes. In fact, an object pixel size depends on its type, the image resolution and its depth from camera. Object detector could suffer from difficulties while detecting objects of different sizes. This is due to the convolutional layers. A convolution extracts information using a kernel of the same size though all input feature map positions.

[18] https://medium.datadriveninvestor.com/review-on-fast-rcnn-202c9eadd23b

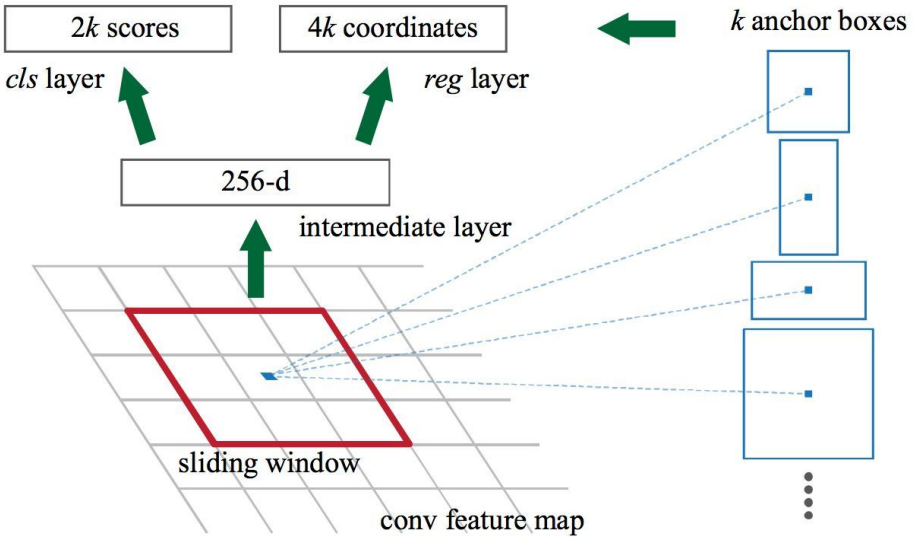


Figure 3.38: Regional Proposal Network [Ren, 2015]

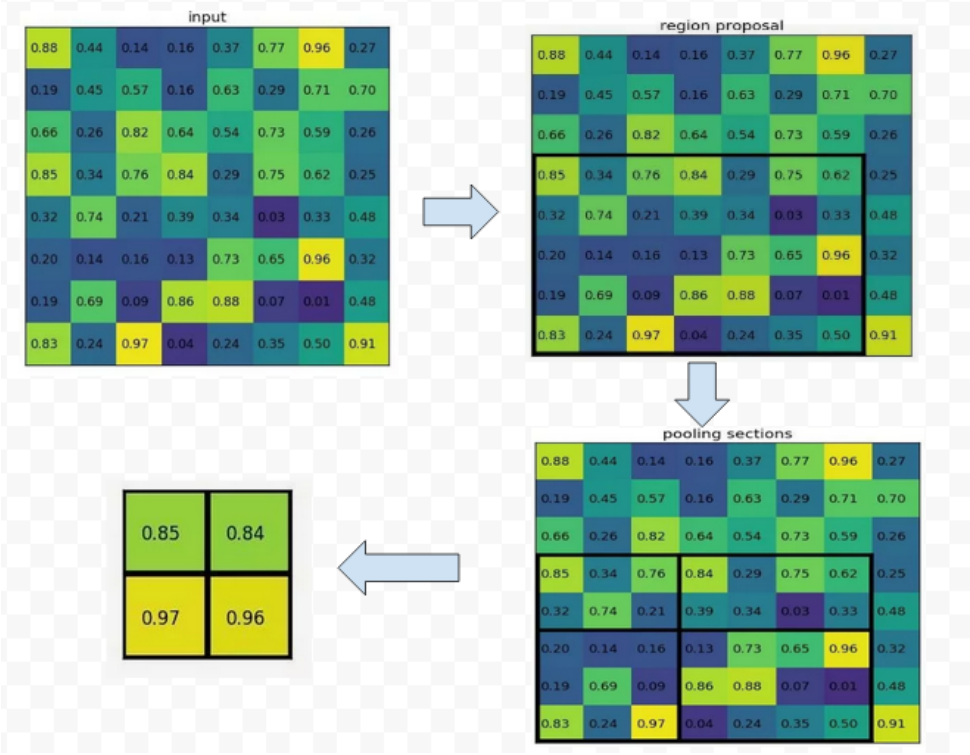


Figure 3.39: ROI Pooling [18]

Therefore only patterns whose sizes are close to the kernel size would be detected. RPN receives the last feature map of the backbone, so only an object of a certain size will be found. FPN approach suggests using many feature maps to detect objects. The backbone is composed of consecutive convolutions, the first ones extract image details, and the last ones extract more global information. FPN uses intermediate feature maps of the backbone to detect objects of different sizes. Since the size of these feature maps is decreasing, it can be viewed as a pyramid of different levels. To enhance the semantic information, FPN creates another pyramid of feature maps. Starting from the top, each one is calculated from the feature map of the higher level in the new pyramid, and the feature map of the same level from backbone pyramid. The feature maps of bigger size detect small objects, and the smaller ones detect big objects [Lin, 2017a].

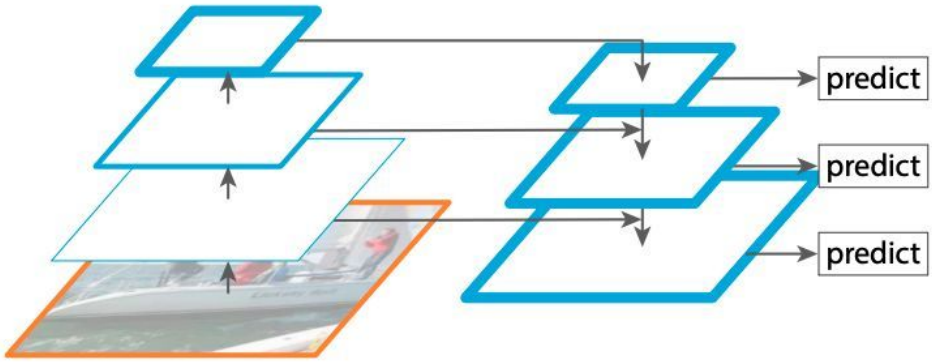


Figure 3.40: Feature Pyramid Networks [Lin, 2017a]

4.5.5 Training Faster R-CNN

Training such neural network is straightforward. There are four losses:

- RPN classification loss, it is calculated between predicted scores and ground truth labels. In fact for each anchor, a score is predicted to check whether the anchor corresponds to a real object. IoU metric is calculated between anchors and ground truth bounding boxes, then the closest ground truth box is associated with each anchor, and a label (0 or 1) is assigned to each anchor according to its IoU with the closest box (using a threshold).
- RPN regression loss, it is calculated between predicted offset coordinates and ground truth boxes. Offset coordinates are predicted for each anchor. The ground truth offset is infered from the associated bounding box to the anchor.
- ROI Heads classification loss, it is calculated between predicted scores and ground truth labels. For each proposal, a score is predicted, and a label is assigned according to its distance with ground truth bounding boxes (in the same way as RPN matching).
- ROI Heads regression loss, it is calculated between offset predicted coordinates and ground truth boxes. Offset coordinates are predicted for each proposal. Ground truth offsets are infered in the same way as RPN.

The backbone parameters are initialized with values obtained by a training on a bigger classification database. In general, the first layer parameters are fixed during training, the other layer parameters are fine tuned during training.

5 Varroa and Bee Detection

There are many approaches that deal with bee and varroa on images. In the following table, I list some of them.

Authors	Date	Objective	Method	Results	Limits
Chazette et al. [Chazette, 2016]	2016	Monitoring hive parameters (temperature, sound, weight), detecting parasites, killing them and informing users	Equipment (camera, mobile laser to kill parasites), Algorithms (CNN for detecting parasites, Information system to inform users)	93% CNN precision	Solution not finalized, and incomplete. Some false positives in the case of large variations in exposure. Not tested in a realistic context.
Toshifumi Kimura et al. [Kimura, 2014]	2014	Bee detection and movement monitoring. Detection of interactions: "crossing", "touching", "passing", "overlapping", "waiting"	Specific laboratory equipment. Algorithm called K-Track (detection using thresholding processing, identification of bees using changes in shape and position, tracking of bees by connecting positions to get trajectories)	77/84 detected events	Method dedicated for laboratories
Magnier et al. [Magnier, 2018]	2018	Bee detection and movement tracking	Using the camera, and a white background. Algorithms: background subtraction, ellipse approximation, edge detection, thresholding		low FPS(Images per seconde)
Vladimir Kulyukin et Sarbajit Mukherjee [Kulyukin, 2019]	2019	Estimation of bee traffic	Motion detection and image classification. Use of CNN, SVM, Random forest. Equipment: BeePi, sensors on beehives	Creation and publication of annotated image and video databases for machine learning	
Magnier et al. [Magnier, 2019]	2019	Follow the movement and trajectory of bees	Image and video processing. Trajectory estimated by polynomials and interpolation		
Hendriks et al. [Hendriks, 2012]	2012	Automatic hive bee tracking	Use of tags on bees. Detection and identification of bees using tags.		No real-time solution. Requires tagging effort. Not suitable for daily use by a beekeeper.
Facchini et al. [Facchini, 2019]	2019	Brood development monitoring	CT-scan scanner to assess pupal development		Only suitable for research and studies

Authors	Date	Objective	Method	Results	Limits
Struye et al. [Struye, 1994]	1994	Monitoring bee traffic	Infrared sensor. 32 bi-directional bee passage channels		
Knauer et al. [Knauer, 2007]	2007	Detect events on bee cells	Image classification: Minimum Distance Classifier, Decision Tree learning, Support vector machines. Using a computer vision system		Cell localization is manual. Requires specific hardware (e.g. Near Infrared Illumination)
Kimura et al. [Kimura, 2011]	2011	Identification, trajectory tracking of bees. Detection of bee dance	Using a camera for recording. Algorithm: Vector quantization	72% identified bees. 50% tracked bees. 1 hour for processing 700 bees from a video of 300 frames	Cases not handled: two bees directly superimposed, or bees hidden in the holes of the hive
Chiu Chen et al. [Chen, 2012]	2012	Estimation of in-and-out activity (entrance to the hive and exit from the hive)	Equipment used: bee passage at the entrance of the hive, LED source for infrared light, infrared CCD camera, labels attached to the backs of the bees. Algorithms: Hough transform for tag detection, SVM for character recognition and bee identification	Symbol recognition accuracy: 98%. Accuracy of bee identification: 86%	Accepting an untested hypothesis: the proposed technique does not have negative effects on the normal behavior of bees
Chiron et al. [Chiron, 2013b]	2013	3D bee tracking	Vision and stereoscopic camera. Algorithm: Global Nearest Neighbors		
Chiron et al. [Chiron, 2013a]	2013	Monitoring the movement of bees	Vision and stereoscopic camera. Algorithms: hybrid segmentation, multi-target tracking, Kalman filter and Global Nearest Neighbor		
Bienefeld et al. [Bienefeld, 2015]	2016	Detection of the hygienic behavior of bees against varroa mites (decapsulation of brood cells)	Video and infrared camera. Specific equipment for recording videos		

Authors	Date	Objective	Method	Results	Limits
Schurischuster et al. [Schurischuster, 2016]	2016	Automatic varroa tracking	System composed of: camera at the entrance of the hive, passage to bees, leds (light sources), Raspberry Pi 3	qualitative assessment	Requires specific hardware. Absence of a rigorous test of the method
apizoom.app [Apizoom, 2022]	2020	Automatic counting of Varroa mites fallen on a greased diaper	Mobile application for: managing apiaries and hives, visualizing the level of infestation over time, configuring parameters. Web application: allowing to download photos for varroa counting	Mobile application in production	Need for a camera with high resolution. Several steps to start the detection
beescanning.com [BeeScanning, 2022]	2020	Calculation of the percentage of infestation by phoretic varroa	Mobile application for managing apiaries and hives and taking photos. Web service for estimating the percentage of infestation: detection and counting of varroa mites	Mobile application in production. Varroa detection: 80% recall. Bee counting: 95% accuracy. Queen detection: 95% accuracy. DWV virus detection: 80% accuracy.	Results may vary depending on optical circumstances. Varroa detection accuracy missing (false positives possibly high). Missing implementation details
Babic et al. [Babic, 2016]	2016	Detection of pollen on bees	Equipment: camera at the hive entrance, Raspberry Pi. Algorithms: Background subtraction, color segmentation, morphology methods, nearest mean classifier, color variance and eccentricity features	Accuracy of 88.7%	Fusion of overlapping bees. False positives (background considered pollen)
Carl Giuffre et al. [Giuffre, 2017]	2017	Estimation of self-grooming rate of bees using videos	Camera, Light Source, Petri. Matlab, Color correction, Thresholding		
Knauer et al. [Knauer, 2005]	2014	Detection of uncapped cells using videos	Camera, illumination. Background modeling		

Authors	Date	Objective	Method	Results	Limits
Elizondo et al. [Elizondo, 2013]	2013	Detection and tracking of parasites from video recordings	Approach based on "Frame by Frame" and background subtraction	90.98% accuracy. 1.25 standard deviation	Method suitable and tested only for videos of a specific type
Davide Bassignana et al. [Bassignana, 2018]	2018	Varroa counting on a greased diaper	Material created called Bee Varroa Scanner: board photo taking. Mobile application to start counting. Neural networks for varroa detection.	Mobile application in production. Interesting Precision-Recall curve.	Need for specific equipment
Maisonnasse A. et al. [Maisonnasse, 2016]	2016	Estimation of certain characteristics of a colony: number of bees, number of cells and area of nutritional reserves	Visual estimate. Learning method to observe: deduction of a correction factor, comparison between observers	Manual method	Lack of automation
Ostiguy et Sammataro [Ostiguy, 2000]	2000	Estimation of the number of Varroa mites that fall on a sticky board	Manual approach: dividing the sticky board into small cells, and counting varroas on a cell sample	Average error: 0.4%. 9.5% standard deviation	Lack of automation. Doesn't handle hidden varroas and debris. Use of the same database for learning and testing
Schurischuster et al. [Schurischuster, 2018]	2018	Parasite detection on bee videos	Algorithms: Foreground detection (Median, GMM, Color thresholding, different color spaces), Patch extraction, pre-processing to improve contrast, feature extraction (color histogram, mean, SURF...), classification (Naive Bayes, SVM, Random Forest) to parasite/non-parasite	Best Accuracy: 81%	Unbalanced image bases (parasites/non-parasites). Method tested under very specific conditions (bees passing through tunnels). Using tunnels is not natural (bees blocked by other dead bees). Limited precision

Authors	Date	Objective	Method	Results	Limits
Bjerge et al. [Bjerge, 2019]	2019	Monitor the level of varroa infestation	Video monitoring unit at the entrance of the hive: use of multispectral illumination and a camera. Algorithm based on deep learning: bee counting and identification of Varroa positions	F2 score for bee count: 0.97. F1 score for varroa detection: <0.91	Inconvenient material. Requires a specific camera for particular wavelengths
Tiwari [Tiwari, 2018]	2018	Recognition of bees from videos. Estimation of bee traffic	Hardware: BeePi is an EBM (electronic beehive monitoring) capable of recording sound, temperature and video. Motion detection algorithm in video. CNN for bee recognition	Best Accuracy: 0.98	Estimation of bee traffic only, not forager traffic (entry-exit activities of bees). Material cost
Ramirez et al. [Ramirez, 2012]	2012	Detection and monitoring of parasites in a controlled environment from video recordings	Noise filtering, detection of regions of interest. Parasite detection using background subtraction	Average accuracy: 93.75%. Standard error: 0.02. Confidence interval: 92.93% - 94.57%	Method not usable in the daily life of a beekeeper. No explanation for heuristics used to set parameters and methods
Gang Jun Tu et al. [Tu, 2016]	2016	Monitoring the behavior of bees and evaluating the state of the hive	Counting of bees at the entrance of the hive and estimation of entry-exit activity. Linear regression	Residual Regression (count, input, output): 0.987, 0.953, 0.888	Does not work for high bee density. Shadows, occlusion and connected bees are unhandled cases
Rodriguez et al. [Rodriguez, 2018]	2018	Recognition of pollen on bees from videos	Capturing videos at the entrance to the hive. CNN for classifying an image containing a bee into "carrying"/"not carrying" pollen	Accuracy of 96.4%	Solution not complete. No automatic solution offered to locate bees
Clara Marc (EPFL) [Marc, 2019]	2019	Automatic counting of Varroa mites fallen on a sticky board	Mobile application: taking photos. Machine learning for varroa counting. Using QrCode to identify a hive		Using a QRCode may not be possible. Missing approach details

Authors	Date	Objective	Method	Results	Limits
Anna Dupleix et al. [Dupleix, 2018]	2019	Automatic counting of Varroa mites fallen on a sticky board	Mobile application: taking photos. Counting using machine learning	Accuracy of 0.938 with small recall. 0.607 precision with large recall	Missing details about used approach
Kulyukin and Reka [Kulyukin, 2016]	2016	Monitoring the traffic of foraging bees using sound and image	Equipment: BeePi, "landing pads", solar panels, camera, microphone, temperature sensor, humidity sensor. Edge detection of a binary image. Pixel separation. Digitization of sound	Accuracies of the proposed algorithms: 73%, 80.5%, 85.5%	Undetected bees in dark images. False positives due to leaves or grass blades. Need to reconfigure for each new hive used
Campbell et al. [Campbell, 2008]	2013	Motion tracking and parasite location	Camera. Background Subtraction		

The main tasks handled by the different approaches and solutions can be summarized by this list:

- Bee counting
- Bee traffic estimation
- Estimation of the in-and-out activity of bees
- Detection of bees carrying pollen
- Detection of brood cell states
- Varroa detection
- Detection of interactions between bees

The main disadvantages and limitations noted are:

- Use of specific equipment (infrared light, infrared camera, tunnels or walkways ...)
- Solution not complete or not finalized (requires manual effort)
- Can only be used for research
- No management of certain cases: overlapping and occlusion, hidden bees, shadows, noises, high density of bees
- Intrusive Methods
- No differentiation between background and bee
- Large variations in exposure lead to false detections
- Limited Accuracy

The three main ready solutions for varroa detection I have found are:

- Apizoom
  - Mobile App and Web App
  - Objective: Varroa detection on boards
  - Main disadvantage: Need to use a professional camera of 24 MP or more
- Beescanning
  - Mobile App and Web Service
  - Objective: Detection of phoretic varroa mites, queen, DWS virus and bees
  - Main drawback: High false positives
- Bee Varroa Scanner

- Mobile application and specific scanner
- Objective: Varroa detection on boards
- Main Drawback: Limited Accuracy

5.1 Bee Detection

[Magnier, 2018] propose an approach to detect and track bees from videos with white background. This detection is based on background subtraction, ellipse approximation, border detection and color threshold, but its FPS (frame per second) is low. In [Magnier, 2019], authors suggest a method to track bees by estimating their trajectories through polynomial and interpolation.

[Kulyukin, 2019] estimate the bee traffic by detecting movements and classifying images based on machine learning methods as CNN, SVM and Random Forest. They published a public database containing annotated images and videos for machine learning.

[Tiwari, 2018] tries to recognize bees and track them from videos using CNN. The recording is managed by BeePi which is a specific hardware dedicated to beehives and makes it possible to collect other kinds of data like sound and temperature.

[Tu, 2016] aim to track bee behavior and evaluate the condition of the hive. This is achieved by counting bees at the beehive entrance and estimating their in-out activity using linear regression. Their method does not handle complicated cases such as high bee density and bee occlusion.

[Kulyukin, 2016] track the traffic of foraging bees using sound and images. After recording videos thanks to BeePi, tracking is performed by pixel separation algorithm and contour detection of binary image.

In [Dembski J, 2020], the authors try to detect bees on video images using three steps: determine the regions of interest ROI for each frame using motion detection, then classify each region whether it contains a bee or not thanks to a convolutional deep neural network, and finally group regions using clustering algorithm.

5.2 Varroa Detection

Varroa is a small parasite that attacks bees. It is considered among the main factors that degrade the bee health and cause colony death [Hood, 2020]. Monitoring varroas is therefore an obligatory task carried out regularly by beekeepers. Thanks to computer vision methods, this task can be automatized. In this subsection, I show some already existing approaches and solutions proposed for detecting varroas in bee colonies.

Varroa mite passes through different phases in a bee colony. First, it is located in brood, more precisely inside larva cells. When the larva becomes an adult bee, it quits its cell with the varroa on its back. The varroa at this phase is called "phoretic varroa". When the varroa gets old, it falls. To estimate the infestation level, a sticky board is put under the hive to collect fallen varroas.

There are different approaches proposed for detecting varroas using computer vision methods. I can classify these approaches in four different classes:

- Detecting varroas in brood cells like [Elizondo, 2013] and [Ramirez, 2012].
- Detecting varroas on sticky boards like [Bassignana, 2018], [Marc, 2019], and [Dupleix, 2018].
- Detecting phoretic varroas at hive entrance like [Schurischuster, 2018], [Bjerge, 2019], [Chazette, 2016], and [Mrozek, 2021].
- Detecting phoretic varroas on hive frames like [Voudiotis, 2022] and [Bilik, 2021].

In my thesis, I am interested in detecting phoretic varroas inside hive frames. My approach requires only taking an image of a bee frame. Detecting phoretic varroas on hive frames is more interesting than other types of detection because of:

- Easier for a beekeeper than taking an image of only brood cells.
- Detecting phoretic varroas by analyzing all bee frame is important for early detection of eventual infestation issue compared to other approaches.
- Sticky board method needs visiting the apiary more than once to put the sticky board and calculate varroas.
- The approaches based on detecting bees at hive entrance require having special equipment and impose a non natural environment for bees.

As I have experienced, using a phone camera to take hive frames photo is quite convenient. This task could be performed by the beekeeper while performing other tasks related to its beekeeping activities. To overcome the issue of handling the phone while wearing beekeeping suit, I have used a phone stylus.

The existing approaches that deal with detecting phoretic varroas inside hive frames have these main limits: Missing implementation details, not realistic bee frame images, using two processing for bee and varroa detection. In my thesis I propose a method based on Faster R-CNN to segment varroas on detected bees by using one neural network on a bee hive frame images. In fact, to have a more complete detection, both bees and varroas should be detected, it is helpful to evaluate infestation level more accurately and to correlate the varroa number with the bee number. My approach can be extended to tasks that need to detect objects inside others.

6 Computer Vision Libraries

To implement computer some vision processing, I used the well known library OpenCV. This library can be used under three different languages: C++, Java and Python. For the purpose of simplicity of coding and integration with other libraries, I used Python implementation.

In fact this language allows straightforward programming without dealing with some technical details, it is therefore chosen by many machine learning libraries. OpenCV proposes many image processing methods like histogram calculation, thresholding, filtering, morphological operations. Concerning the machine learning libraries, there are different ones:

- Tensorflow; it is an open source library developed by Google. It provides a python API for creating and manipulating neural network, loading data and launching and controlling training.
- PyTorch; it is an open source alternative to Tensorflow, developed by Facebook. It has the advantage to be more developer friendly, as its debugging experience is more interesting than Tensorflow one. But the later is more optimized and efficient in production environment. Since my work consists on research, I chose PyTorch for manipulating neural network models.
- Matlab is a whole environment dedicated for performing mathematical operations on data. It provides also some functions dedicated for machine and deep learning.
- Numpy is a well known python library dedicated to manipulating multidimensional arrays.
- Pandas is a python library used for loading and handling data.
- Scikit-Learn is a python library providing implementation for many machine learning algorithms like kNN, k-Means and SVM.
- MatPlotlib is a python library used for displaying graphs. It provides API for handling data and showing specific graphs.
- TensorBoard is a tool used for displaying information about neural networks and their training.

In my research, I have mainly used PyTorch library for neural network implementations. I have also used Numpy for storing and manipulating image data. MatPlotlib was used to display images and charts.

Tensorflow, Matlab and TensorBoard were sometimes used to respectively test some neural network approaches, to annotate images and to display training evolution.

7 Conclusion

Computer vision methods have a potential to automate human tasks. When coupled to deep learning, they can be very efficient in solving object detection tasks. Faster R-CNN is an example of neural network that achieve such objective by automatically extracting features from images, generating proposals and classifying them to obtain bounding boxes corresponding to object positions. In my thesis I propose to improve object detection by extending Faster R-CNN architecture. The issues in which I am interested are related to beekeeping domain: detecting

bees and varroas. To achieve this objective, two problems must be solved: (1) detecting bees in dense scenes and (2) nested object detection to predict bee and varroa positions.

Thus, one of the objectives of this thesis is to propose techniques for automatically detecting and counting bees and varroa mites on a hive frame. I propose to use the following techniques, which coupled within a system, will be distinguished from the existing one by:

- The use of high-performance neural networks for object detection; this will increase the accuracy of the detection system.
- The management and treatment of bee-specific situations such as high density or overlapping by relevant adaptations of neural network models.
- Simplified operation: the system will be connected to a mobile application which will provide it with the images to be analyzed from photos taken by telephone. This system does not require specific and disruptive equipments like bee tunnels, infrared sensors, Raspberry Pi, LED source ...
- Implementation and dynamic evolution, as it is used: the results will improve thanks to continuous learning and thanks to feedback from beekeepers.

Chapter 4

Experimental Context

Résumé

Dans le domaine de l'apprentissage profond, les données sont primordiales, puisque l'entraînement et l'évaluation sont réalisés sur des données annotées. Dans mon cas, j'ai construit une base d'images de cadre d'abeilles. Ces images viennent de différentes sources: images téléchargées depuis Internet, photos prises en utilisant une caméra et un téléphone sur un rucher et images fournies par les partenaires du projet. Pour l'annotation j'ai installé et configuré trois différents outils: Matlab Image Labeler, Via Annotator, et ImageTagger. Ces outils nous ont permis d'annoter les objets par des boîtes englobantes, des ellipses et de collaborer avec les partenaires. Afin d'avoir un entraînement robuste et généralisable, aucun pré-traitement spécifique n'a été réalisé sur les images. Le seul traitement réalisé est le découpage automatique des images pour réduire le nombre d'objets à annoter.
Nous avons annoté les abeilles par des boîtes englobantes. Les varroas par contre ont été annotés par des ellipses car leur forme est plus simple. J'ai suivi quelques étapes pour obtenir des annotations de types différents afin de tester des approches différentes. Premièrement, les abeilles infectées ont été annotées par des boîtes englobantes. Puis, sur chaque image représentant une abeille, les points correspondant à des varroas ont été annotés par des ellipses. Ensuite, les annotations de varroas ont été projetées sur les images d'origine. J'ai, par ailleurs, augmenté la base par des exemples d'abeilles non infectées.
Afin de gérer le stockage et la manipulation des annotations, j'ai utilisé les différents formats de représentation proposés par les différents outils d'annotations. J'ai par conséquent développé des codes pour pouvoir transformer les annotations d'un format à un autre.

Résumé

Pour évaluer les approches, j'ai utilisé des métriques spécifiques au domaine de la détection d'objets. Ces métriques sont basées sur la précision et le rappel. La précision mesure la capacité du détecteur à prédire des objets de manière correcte (pas de faux positifs). Le rappel mesure la capacité du réseau de neurones à ne pas manquer d'objets (pas de faux négatifs). La métrique principale utilisée pour les tests est la moyenne de la précision moyenne (Mean Average Precision) qui est calculée comme la moyenne de précisions calculées sous différentes conditions. J'ai employé cette métrique dans le cas des boîtes englobantes et dans le cas des points. La similarité entre deux boîtes englobantes est calculée en utilisant la métrique IoU (rapport de la surface d'intersection et de l'union). La similarité entre deux points est calculée en utilisant une formule relative à l'inverse de la distance.
Afin d'implémenter mes approches, j'ai utilisé la bibliothèque Detectron2. Elle propose des fonctions pour des réseaux de neurones de reconnaissance d'objets tels que Faster R-CNN et Mask R-CNN. Elle a aussi l'avantage d'être bien documentée et d'offrir des mécanismes efficaces d'extension. J'ai notamment créé de nouvelles classes pour l'approche basée sur les coins, modifié les entraîneurs classiques pour entraîner le réseau de neurones sur deux bases de données, et créé de nouvelles classes pour implémenter des nouvelles méthodes d'évaluation.
L'entraînement et l'évaluation des approches proposées et implémentées sont réalisés sur un ordinateur de 32 Go de RAM, 8 CPUs et carte Nvidia GPU de 16 Go.

Contents

1 Introduction

In this chapter I present experimental context of my research. In Section 4.2, I present information about used dataset for training neural networks and evaluating their performance. In Section 4.3, I explain the metrics that I have used to assess approach accuracy. Finally, in Section 4.4, I present some technical details about used library and code to implement my approaches.

2 Dataset

In this section I describe the built dataset used for training and testing my approaches to detect bees and varroas based on neural networks. I start by describing the dataset sources from which I retrieved images. Then I present the tools used for image annotation. After that, I explain the preprocessing that has been performed on the images. I give some details about the varroa annotation. Then, I describe the formats in which annotations were stored, before showing some statistics about obtained images and annotations.

2.1 Data Source

Since I use machine learning to detect bees and varroas. I needed a database composed of images of bees and varroas. Therefore, one of the first steps I performed was to construct this dataset. The images used for training and testing my approaches come from three sources: (1) images downloaded from the Internet, (2) photos taken using camera and phone in an apiary, and (3) images provided by our project partners. In fact our partner members ITSAP and ADA are specialized in beekeeping domain. Their expertise helped us in annotating bee images.

2.2 Image Annotation Tools

I used three different tools to annotate my images. The first is Imagetagger which is a web application connected to a server for annotating images with bounding boxes [1]. This tool enabled us to work remotely in collaboration with our partners. I installed it and configured it to make it accessible through the Internet. Our project partners used it to help us in the annotation task. The second tool is the Matlab application Matlab Image Labler [2]. Using this tool, I annotated some images of my database with bounding boxes, then exported the annotations to the Matlab workspace. A Matlab code was developed to export annotations from Matlab format to Json format. This tool has the advantage of being easy to use. I used also Via annotator tool [3] to annotate varroas using ellipses, it is a web application that does not require installation.

[1] https://github.com/bit-bots/imagetagger
[2] https://fr.mathworks.com/help/vision/ref/imagelabeler-app.html
[3] https://www.robots.ox.ac.uk/~vgg/software/via/

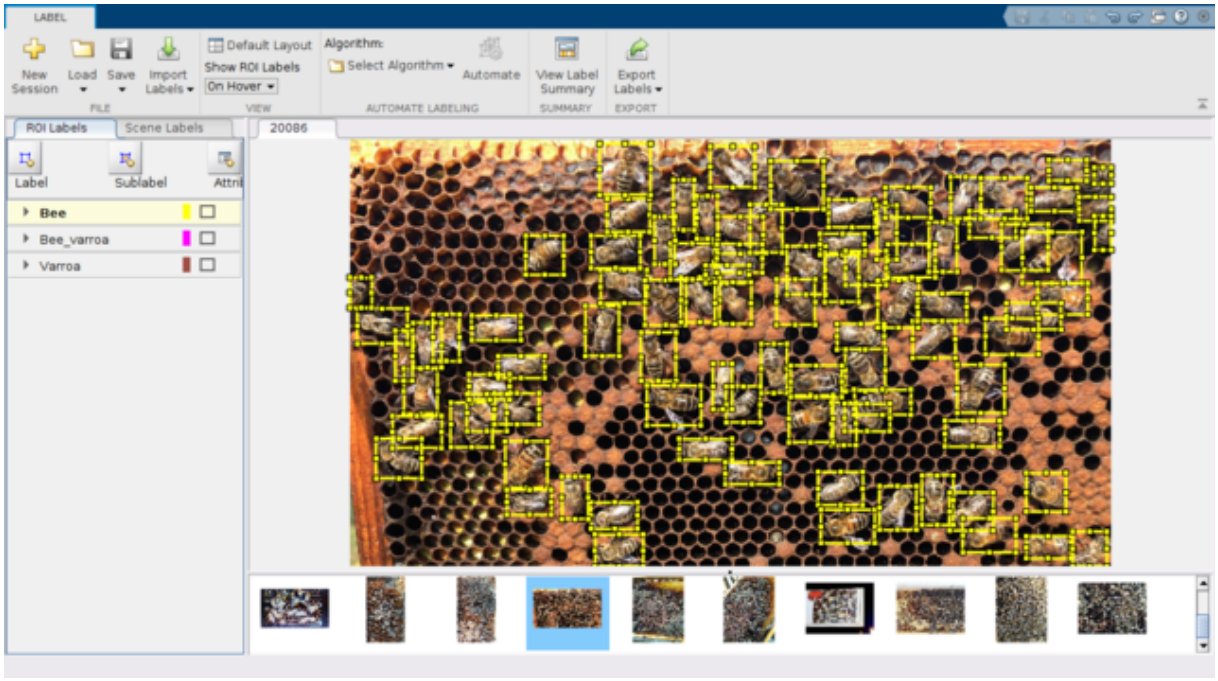


Figure 4.1: Screenshot of Matlab Image Labeler

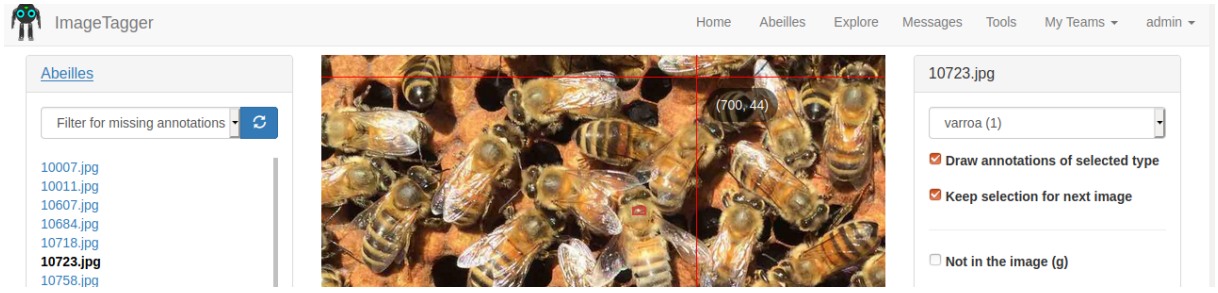


Figure 4.2: Screenshot of ImageTagger Tool

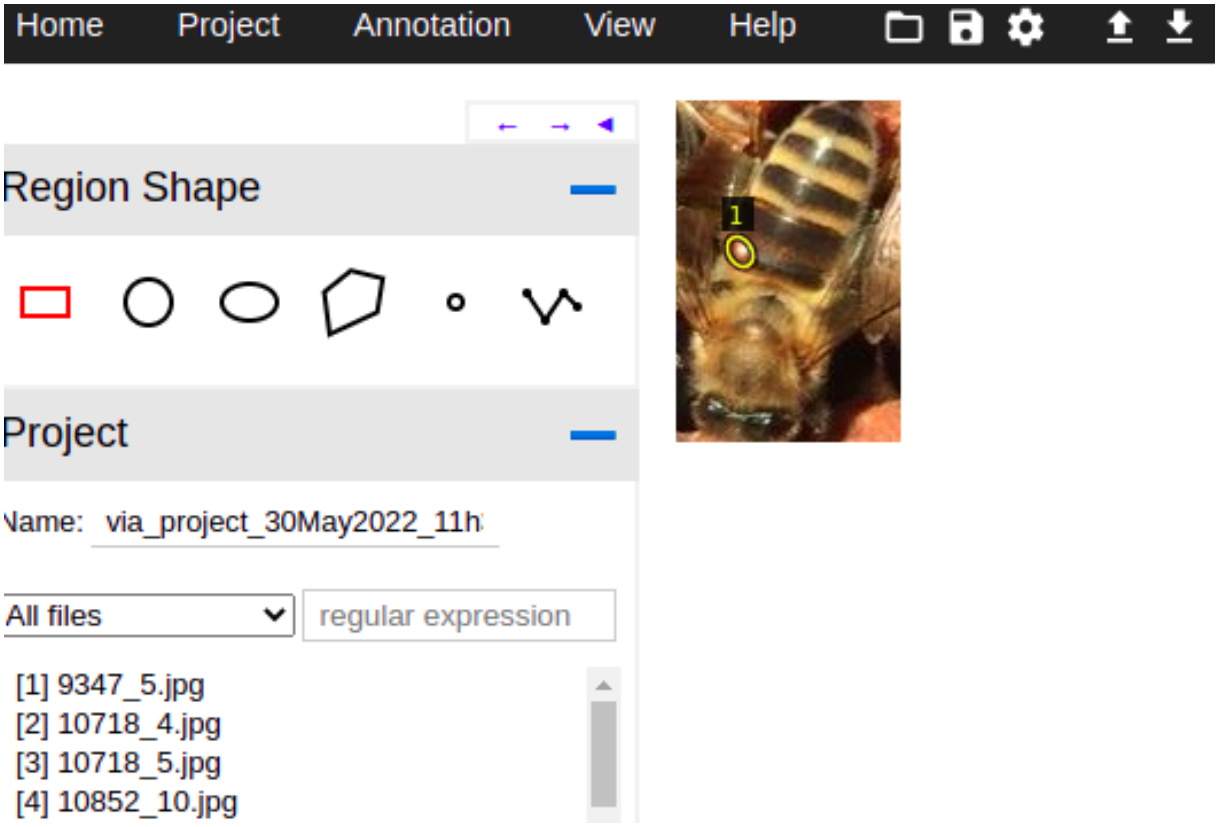


Figure 4.3: Screenshot of Via Annotator Tool

2.3 Preprocessing

The main preprocessing I performed on images is cropping to make the annotation task easier. In fact there were images with many bees, complete annotation of such images takes a lot of time and effort. I performed two kinds of cropping: manual cropping and automatic cropping. The images were first cropped manually to reduce the number of bees to annotate per image. Then the size of the bee in each image is estimated using a trained neural network. In fact, I have annotated some available low resolution images before obtaining more interesting images, and trained a Faster R-CNN neural network, the detection precision was not satisfying, the network was only used to estimate the size of bee in pixels in images, then an automatic cropping was performed to take only a limited number of bees in each image.

2.4 Varroa Annotation

After having annotated the bees, I worked on the annotation of varroas. To have a richer dataset, I annotated infected bees and varroas present in these bees. I started by selecting images of bee frames containing varroas, then I annotated infected bees using bounding boxes. Then I crop images to obtain images of infected bees. For each infected bee, I annotated varroas using ellipses. At the end I obtain these data:

- Image of bee frames

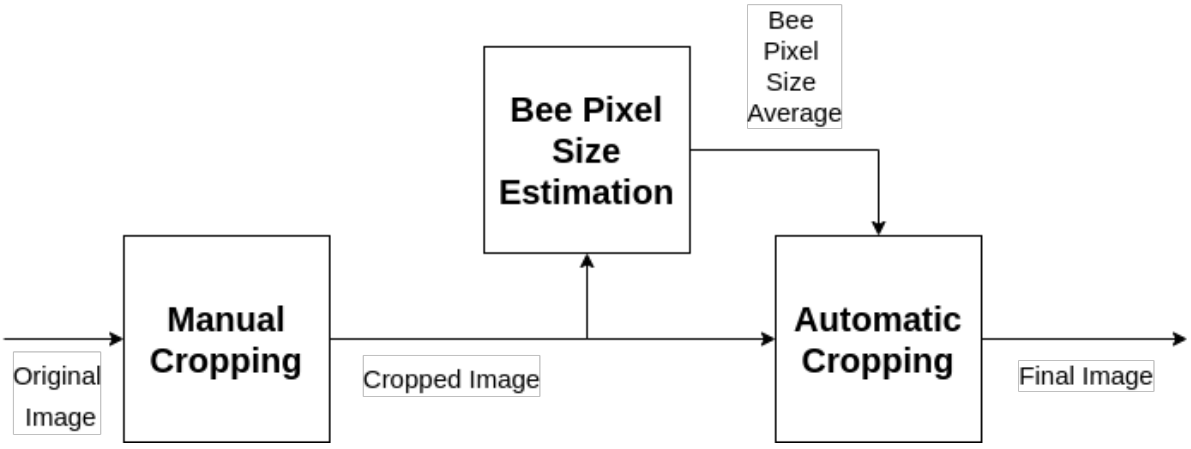


Figure 4.4: Cropping processing performed on images

- Annotations of infected bees in bee frame images
- Images of infected bees
- Annotations of varroas in infected bee images

To test different approaches, I projected varroa annotations on bee frame images. I also augmented my dataset by not infected bees. The result is the following:

- Image of bee frames
- Annotations of infected bees in bee frame images
- Annotations of not infected bees in bee frame images
- Images of infected bees
- Images of not infected bees
- Annotations of varroas in infected bee images
- Annotations of varroas in bee frame images

2.5 Annotation Formats

To process annotations, they are stored in specific formats. As long as I use different tools to annotate images, different formats are managed. Each format specifies how to store bounding boxes on images and optionally masks and ellipses. Python code was developed to convert data from one format to another. In this section I describe these different formats.

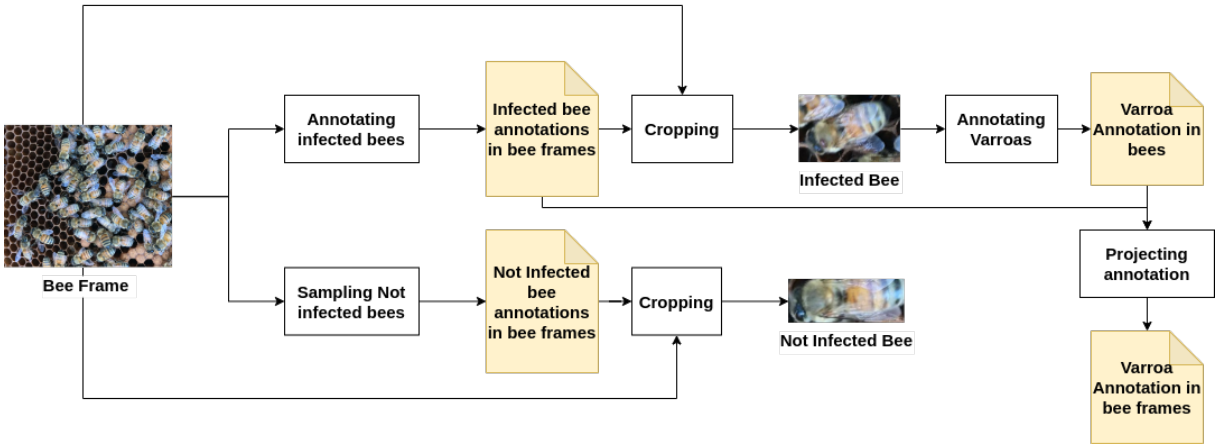


Figure 4.5: Varroa Annotating Workflow

2.5.1 COCO

COCO (Common Objects in Context) format [4], is a well known format used to represent object annotations. It handles bounding boxes, keypoints, segmentation ... Annotations are stored in JSON objects, the most important attributes are "images" and "annotations". "images" is a list of objects, each one contains information about an image: "id", "width", "height", and "file_name". "annotations" is a list of objects, each one contains information for an annotation. In case of bounding box annotations, the most important attributes of each annotation are:

- "id" the identifier of the annotation
- "image_id" the image identifier of the image where the bounding box is present
- "category_id" the identifier of the category of the bounding box (or the class of the object)
- "segmentation" the segmentation mask of the object (the pixel region of the object)
- "area" the area of the bounding box
- "bbox" the coordinates of the bounding box in this format "[x coordinate of the top left corner, y coordinate of top left corner, box width, box height]"
- "keypoints" a list of keypoints used, for example, when detecting human joints.

2.5.2 Detectron2

Detectron2 is a library for computer vision based on artificial intelligence. It is written in Python and uses PyTorch library for building deep learning networks. It contains implementations of several states of the art neural networks for object recognition like Faster R-CNN. In its documentation [5], it specifies the annotation format to respect to train or test neural networks. The annotations are stored in a python list of python dictionaries. Each dictionary contains annotation data of an image, the interesting keys in my case are:

[4] https://cocodataset.org/#format-data
[5] https://detectron2.readthedocs.io/en/latest/tutorials/datasets.html

- "file_name" the file name of the image
- "height" the pixel height of the image
- "width" the pixel width of the image
- "image_id" the identifier of the image
- "annotations" annotations present in the image, it is a list of dictionaries, each one corresponds to one annotation:
  - "bbox" bounding box of the annotation, it is a list of 4 numbers
  - "bbox_mode", the format of the bounding box data, it can be in this format "[x coordinate of the top left corner, y coordinate of top left corner, width, height]" or this format "[x coordinate of the top left corner, y coordinate of top left corner, x coordinate of the bottom right corner, y coordinate of the bottom right corner]"
  - "category_id", the category identifier of the annotation (its object class)
  - "segmentation": it contains segmentation data; pixel region of the object, it can be either a list of polygons (each one is a list of points) or a mask in COCO's compressed RLE format
  - "keypoints": keypoints annotations, it is a list of numbers corresponding to the coordinates of the points

Regarding testing, Detectron2 converts annotation and prediction data into COCO format. In fact, the evaluation is performed using a COCO library [6] dedicated for evaluating object detection results.

2.5.3 Via

Via implements functionality for exporting and importing annotations into different formats like COCO format. In my case, I used the following available JSON format as it was convenient to manipulate: object keys are file identifiers, mapped objects contain annotation information present in corresponding image files, each one contains the following keys:

- "file_attributes": custom attributes that can be assigned to the file, I have not used it
- "filename": the name of the file containing the image
- "size": the size of the file containing the image
- "regions": the regions present in the image, each one corresponds to an annotated object, the value is a list of JSON objects, each one has the following keys:
  - "region_attributes": custom attributes that can be assigned to each region. In my case I used these attributes to implement varroa annotation workflow, in particular, for identifying annotations

[6] https://github.com/cocodataset/cocoapi/tree/master/PythonAPI/pycocotools

  - "shape_attributes": annotation data, in case of bounding boxes, the saved attributes are
    * "height": bounding box height
    * "width": bounding box width
    * "x": bounding box x coordinate
    * "y": bounding box y coordinate

    in case of ellipses:
    * "cx": ellipse center x coordinate
    * "cy": ellipse center y coordinate
    * "rx": ellipse x radius
    * "ry": ellipse y radius
    * "theta": ellipse rotation angle

I implemented Python code to transform annotations from Via format to Detectron2 format and vice versa. Ellipse annotations are converted to segmentation data.

2.5.4 Matlab

Matlab Image Labeler is a tool for annotating images. It is a Matlab application, annotation data can be transferred to Matlab workspace using Matlab data structure: it is a table containing two columns:

- "imageFilename": the file names of annotated images
- "Bee": bee annotations, each cell value is an internal table containing bounding box data. Each line corresponds to an object, the columns of the table correspond to bounding box coordinates (x, y, width and height).

A Matlab code was developed to transform annotations into Detectron JSON format.

2.5.5 ImageTagger

I used ImageTagger tool to annotate bees with bounding boxes. This tool offers the possibility to define the exportation format. I used this functionality to export annotations into Detectron format.

2.6 Annotation Statistics

The images I used for annotations have different resolutions and sizes. The neural network Faster R-CNN has the ability to detect objects of different sizes.

Table 4.1 gives the number of images used for annotation and the number of obtained annotations.

Table 4.1: (1): images completely annotated by bees
(2): images from which infected bees sample were retrieved
(3): images from which not infected bees sample were retrieved
(4): images from which varroas were annotated

(1) Frame images	Annotated Bees
63	3863
(2) Frame images	Annotated Infected bees
254	565
(3) Frame images	Annotated Not Infected bees
63	679
(4) Bee images	Varroas
611	638

3 Evaluation Metrics

There are several metrics proposed for evaluating the performance of tasks related to object recognition. In this section I describe the metrics I selected to assess the accuracy of my proposed approaches. My choices depend on the objectives that I want to reach. In fact, my first approach proposed for detecting objects in dense scenes must reduce the number of missed objects. To assess that, I calculated the recall metric. But when the recall is improved, the precision could be decreased. To evaluate the whole accuracy of the approach, I calculated also the mean average precision to check whether my approach increases recall at the expense of precision or improve it without affecting the precision. Concerning the second approach proposed for detecting nested objects, I calculated as well the mean average precision for comparing the approach to standard methods. For enclosing objects (bees in my case), I used the standard metric calculated on bounding boxes. Whereas, for internal objects (varroas in my case), I used an alternative metric calculated on keypoints. In fact, I considered varroas as points, this gives higher results than bounding boxes, because it is only influenced by the precision of detecting varroa centers, it won't be affected if varroa borders are not accurately predicted. This match my objective, since I am not interested in detecting the precise region of a varroa, I only need its existence status.

3.1 Precision and Recall

In machine learning tasks, precision and recall are performance metrics. Consider data composed of elements. Each is labeled as relevant or not relevant example. Consider a task that must retrieve relevant elements, and an algorithm (can be a neural network structure) that tries to execute that task. To evaluate its performance, the algorithm is executed on data, the retrieved elements are compared to labeled data. There are two types of comparisons:

- Precision is the fraction of relevant retrieved elements among all retrieved elements.
- Recall is the fraction of relevant retrieved elements among all relevant elements.

All the retrieved elements are considered positive in the point of view of the algorithm, and all not retrieved ones are considered negative in the point of view of the algorithm. The relevant positive elements are called true positives, the not relevant ones are called false positives. The relevant negative elements are called false negatives, the not relevant ones are called true negatives. Here is the precision and recall formula:

$$Precision = \frac{\text{Relevant Retrieved Elements}}{\text{Retrieved Elements}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{4.1}$$

$$Recall = \frac{\text{Relevant Retrieved Elements}}{\text{Relevant Elements}} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{4.2}$$

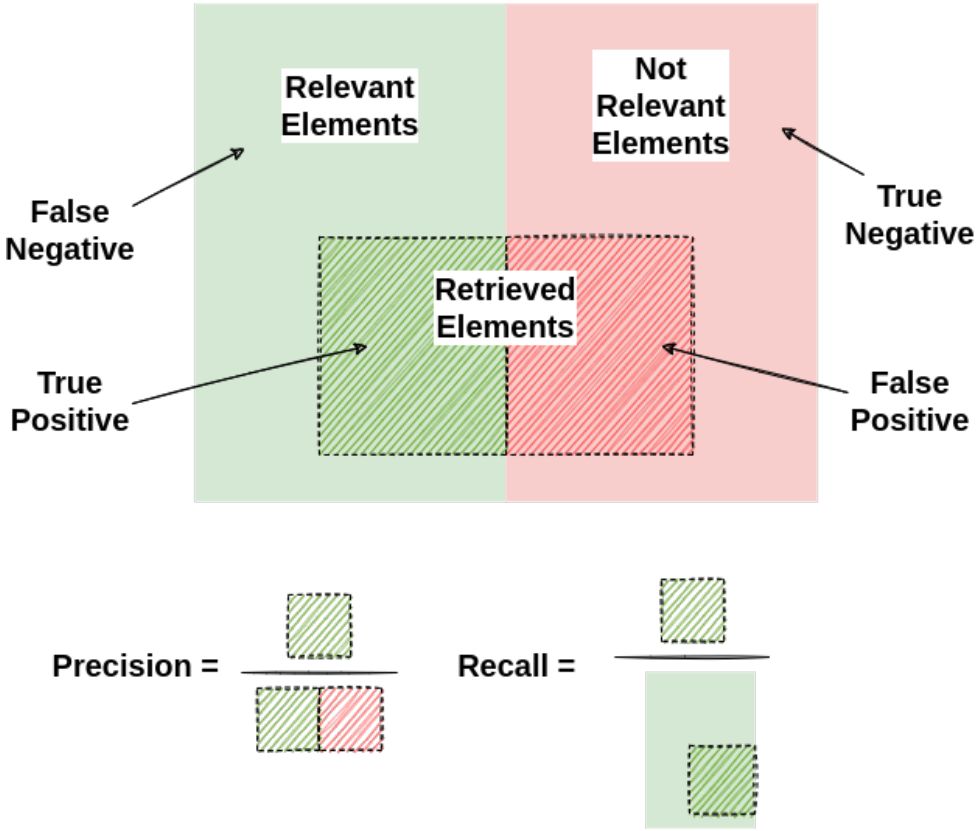


Figure 4.6: Precision and Recall

3.2 Precision and Recall for Classification Task

These two metrics are widely used in tasks related to machine learning. A common example is the classification task. Consider data where each element belongs or not to a predefined class. The desired task is to predict for each input whether it belongs to the class or not. Precision and Recall metrics can be applied to prediction results by simply considering elements belonging to the class as relevant, and retrieved elements the set of elements that the algorithm thinks they

belong to the class:

$$Precision = \frac{\text{Well Algorithm Classified Elements}}{\text{All Algorithm Classified Elements}} \tag{4.3}$$

$$Recall = \frac{\text{Well Algorithm Classified Elements}}{\text{Real Ground Truth Classified Elements}} \tag{4.4}$$

Algorithm Classified: Elements that the algorithm thinks they belong to the class.
Ground Truth Classified Elements: Elements that really belong to the class.

3.3 Accuracy Metric

The accuracy metric is a basic metric that evaluates the accuracy of predictions. It is the fraction of correct predictions among all the predictions. Correct predictions is the set of positive relevant elements and negative not relevant elements:

$$Accuracy = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{False Positive} + \text{True Negative} + \text{False Negative}} \tag{4.5}$$

3.4 Precision and Recall for Object Detection

An algorithm that tries to detect objects must find all the objects in the input image. Consider an input image containing N objects, the algorithm detects M objects, among these M objects there are good ones and bad ones, the precision metric is the fraction of good ones among all predicted objects. The recall is the fraction of well detected objects among all objects present in the image.

$$Precision = \frac{\text{Good detections}}{\text{All detections}} \tag{4.6}$$

$$Recall = \frac{\text{Good detections}}{\text{All objects}} \tag{4.7}$$

The precision metric evaluates the ability of the object detector to produce good predictions. The recall metric evaluates the ability of the object detector to not miss objects.

3.5 IoU: Intersection over Union (Jaccard Index)

I have seen how precision and recall metrics are calculated in previous subsection. Both of them consist in finding good detections, when a detection given by an object detector is considered as good detection? To determine whether a detection is good or not, IoU metric is used. It is a metric defined between two bounding boxes: the area of intersection over the area of union.

$$IoU = \frac{\text{Intersection Area}}{\text{Union Area}} \tag{4.8}$$

This metric evaluates the nearness of two bounding boxes taking into account their sizes. When two bounding boxes correspond to the same object, it is likely that their IoU is high. And when the IoU of two bounding boxes is high, it is likely that they correspond to the same object. The distance between bounding box centers is not sufficient to assess the correspondence to the

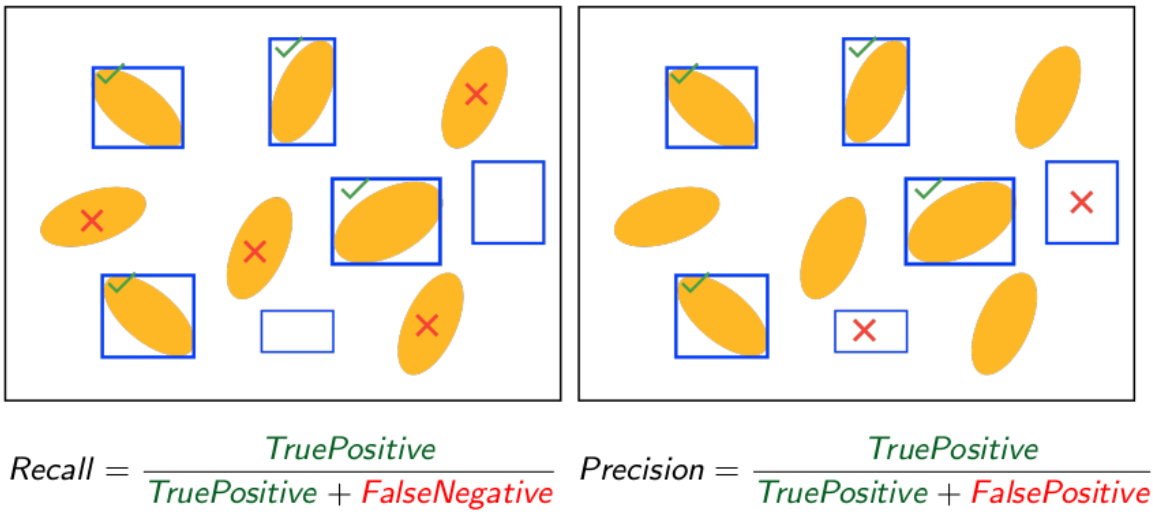


Figure 4.7: Precision and Recall for Object Detection

same object or not, because two different objects can have center points close to each other with very different sizes (e.g. superposition of a small object on a big object). To find good detections, IoU metric is calculated between each predicted bounding box and all ground truth bounding boxes (corresponding to real objects). This information is stored into a 2D matrix. An IoU threshold is defined, for each detected bounding box, the biggest IoU with the remaining ground truth boxes is compared with the threshold, if it is greater, the detected bounding box is considered as good detection, and the corresponding ground truth box is removed from remaining bounding boxes, otherwise the detected box is considered as a false detection. The order of processing the bounding boxes is their scores (neural networks predict a score for each bounding box corresponding to the probability that the box surrounds a real object).

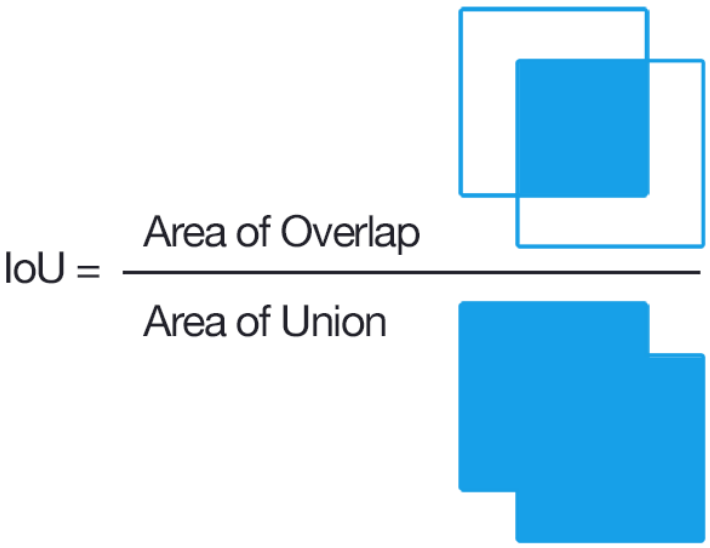


Figure 4.8: Intersection over Union metric [7]

3.6 Average Precision and Average Recall

As explained in the previous section, precision and recall calculated in the context of object detection need a defined IoU threshold. For each different IoU threshold the results are different. To assess the performance more precisely, the metrics are calculated for different thresholds, then the average is calculated across all precisions and recalls, to obtain an average precision and an average recall. In some cases, the neural network detects objects of different classes, its output is the bounding box positions and their classes. In this case, evaluation metrics are calculated for each class, and an average is calculated between the results. Another relevant parameter considered in calculating precision and recall for object detection task is the maximum number of detected bounding boxes taken into account: "MaxDets". The bounding boxes are ordered according to their scores, the first "MaxDets" bounding boxes are retrieved and used for calculating performance metrics. Different values are given to this parameter, an average is calculated between obtained values. As summary, these are the different parameters used to calculate average precision and average recall:

- IoU thresholds
- object classes
- maximum number of detections

3.7 Mean Average Precision

Using two metrics to evaluate an object detector performance is not very convenient when different approaches are compared with each other. In fact, when the recall is increased, the precision tends to decrease, this can be explained by the fact that the detector becomes less restrictive about its prediction when it detects more objects. Therefore, when two approaches are compared, both precision and recall must be taken into account. To decide which approach is better, another metric summarizing precision and recall should be used. Mean average precision is a performance metric that takes into account precision and recall in the same time. This metric is based on precision recall curve. This curve plots the precision under different recalls. Figure 4.9 shows an example of a precision recall curve. To obtain different recall values, the predicted bounding boxes are ordered by their scores, for each bounding box score, a set of true positive elements is constructed by considering only bounding boxes whose scores are higher than the current bounding box score. The recall is calculated for each iteration, it is clear that the recall increases or stays the same after each iteration, and the precision decreases. By using interpolation, the curve can be drawn to plot precision values corresponding to all recall values between 0 and 1. Figure 4.10 illustrates the calculation process of curve values.

3.7.1 Maximum Recall

In some cases, the maximum recall obtained after performing calculation iterations is less than 1. In this case, the neural network does not detect some ground truth objects, it does not even

[7] https://commons.wikimedia.org/wiki/File:Intersection_over_Union_-_visual_equation.png

detects them with a low score. All the values coming after the maximum recall are therefore assigned 0 precision. Figure 4.11 shows an example of Precision Recall curve when the maximum recall is less than 1.

3.7.2 Perfect Curve

A perfect object detector would be precise regardless of the recall, in other words, the detector does not have have false positive whatever the minimum threshold score considered. Furthermore, it should detect all ground truth objects, so the maximum recall must be 1. In this case the precision recall curve is the line $precision = 1$. In general a curve is better than another when the surface under the first one is bigger. In fact, when the surface is bigger, the precision values at same recalls are greater in average than the precision values of the second curve.

3.7.3 Mean Precision

To calculate mean precision, the precision values of 11 recall points are taken into account: $0, 0.1, 0.2, ...1$, the mean is calculated between them. It is clear that a mean precision is degraded when the maximum recall is less than 1.

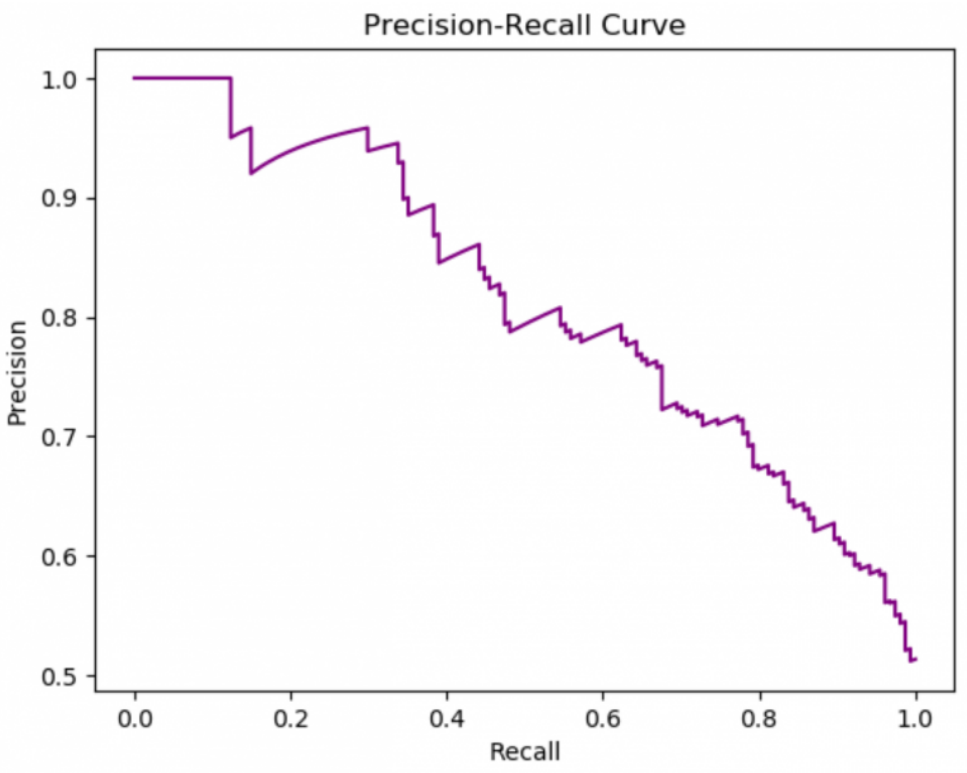


Figure 4.9: Precision Recall Curve [8]

[8] https://www.statology.org/precision-recall-curve-python/

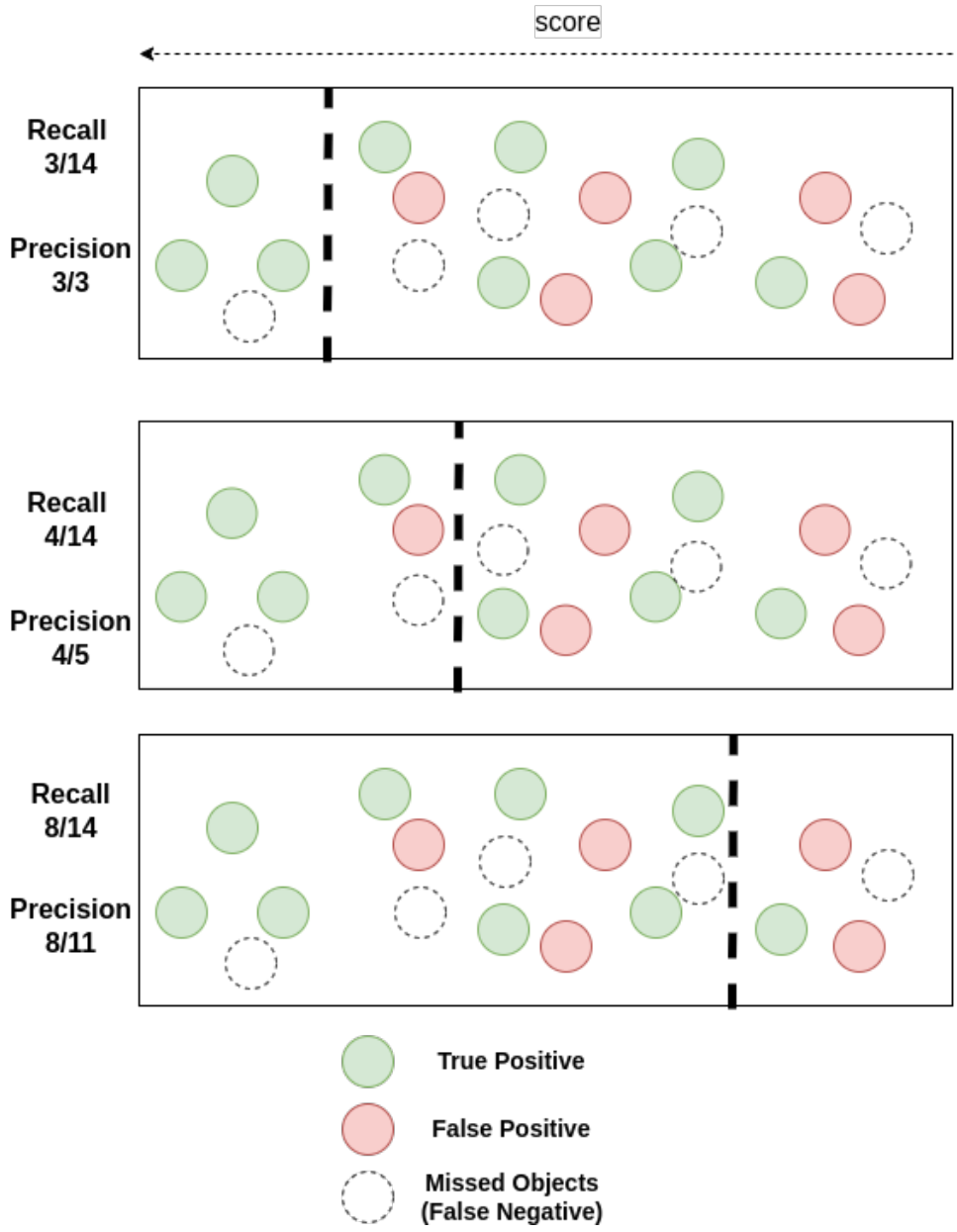


Figure 4.10: Calculation iterations of precision-recall curve values: it shows an example of three ordered iterations, recall increases and precision decreases

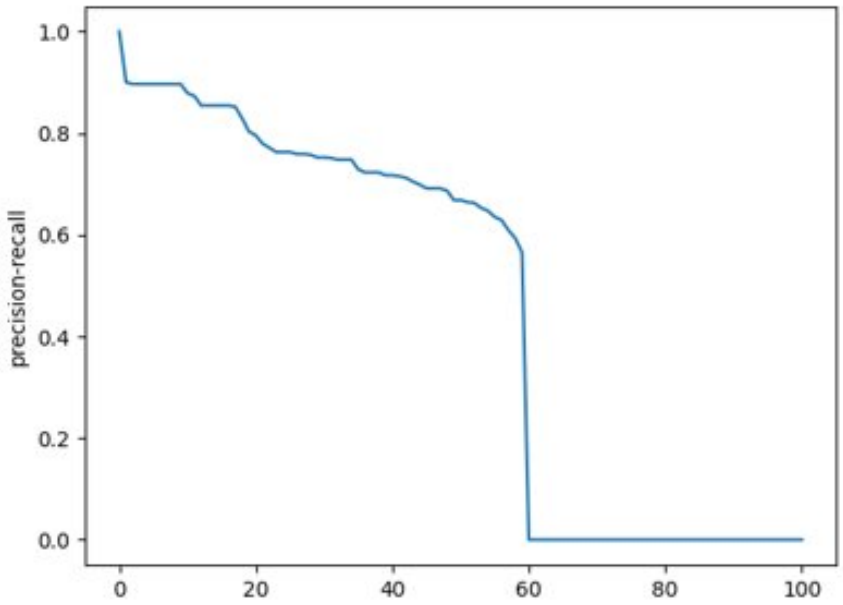


Figure 4.11: Precision Recall curve example when maximum recall is less than 1

3.8 Dice Score

To evaluate the accuracy of varroa detection, I need to find a performance metric that assesses segmentation results. I chose dice score metric because it handles the class imbalance between background and foreground pixels. In fact, in the case of varroa segmentation, the varroa regions are very small compared to the background regions, therefore using traditional metrics like accuracy would give non representative results. As illustrated by Figure 4.12, the dice score formula is [9]:

$$DiceScore = \frac{2 * \text{Overlap Area}}{\text{Predicted Area} + \text{Ground Truth Area}} \tag{4.9}$$

Predicted Area: the pixels labeled as varroa
Ground Truth Area: the real pixels corresponding to varroa

3.9 Keypoints Detection Evaluation

I have used another metric to evaluate varroa detection. I consider a varroa as a point, and check whether used neural network could detect point positions. Therefore I used a metric related to point detection. COCO standards suggest a metric to deal with this type of tasks, it is originally proposed for the detection of human keypoints. The calculation of metric is based on the same principle as Mean Average Precision for bounding boxes, the only difference is that instead of employing IoU metric to compare bounding boxes, a distance metric is used to compare two different sets of keypoints. The library pycocotools offers functions to calculate this metric. I

[9] https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2
[10] https://www.kaggle.com/code/yerramvarun/understanding-dice-coefficient

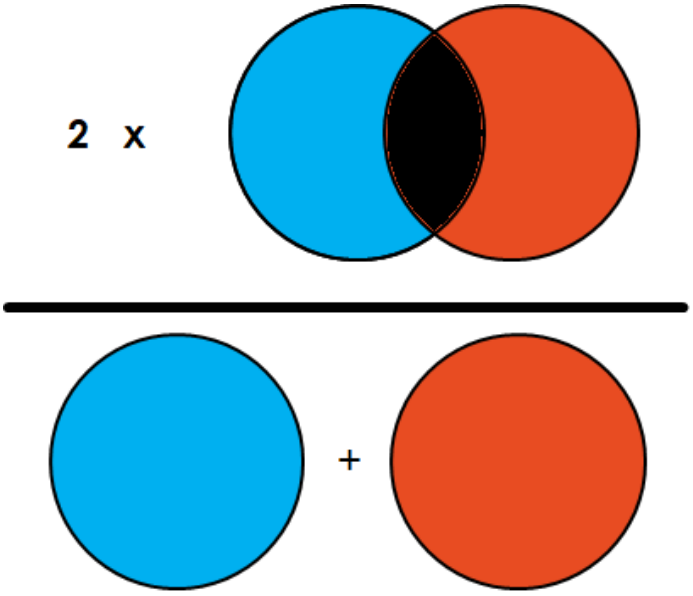


Figure 4.12: Dice Score [10]

configured it to use one keypoint instead of multiple keypoints. This point corresponds to varroa center. A code was developed to extract these points from the ground truth varroa instances and from the segmentation result.

4 Technical Details

The proposed approaches were developed using Detectron2 library. It is a library dedicated for computer vision tasks based on deep learning. Its source code is hosted on this Github repository [11]. It is developed by Facebook, and built on Pytorch library. It contains implementations of state-of-art neural networks for object detection and image segmentation, such as Faster R-CNN, Mask R-CNN and RetinaNet. I have chosen this framework among others because it contains official implementations of Faster R-CNN and has the advantage of being extensible. The functionalities of Detectron2 are arranged into different Python classes. The extensibility is enabled through inheritance mechanism between classes. The schema 4.13 shows some created Python classes and their inheritance relations with Framework classes. In this section I will explain the main aspects of this library, how I extend it, and the used hardware for my training and evaluation phases.

4.1 Detectron Training

Detectron library offers Python functions and classes that help user or developer to launch training easily. The training mechanism is composed of these steps:

[11] https://github.com/facebookresearch/detectron2

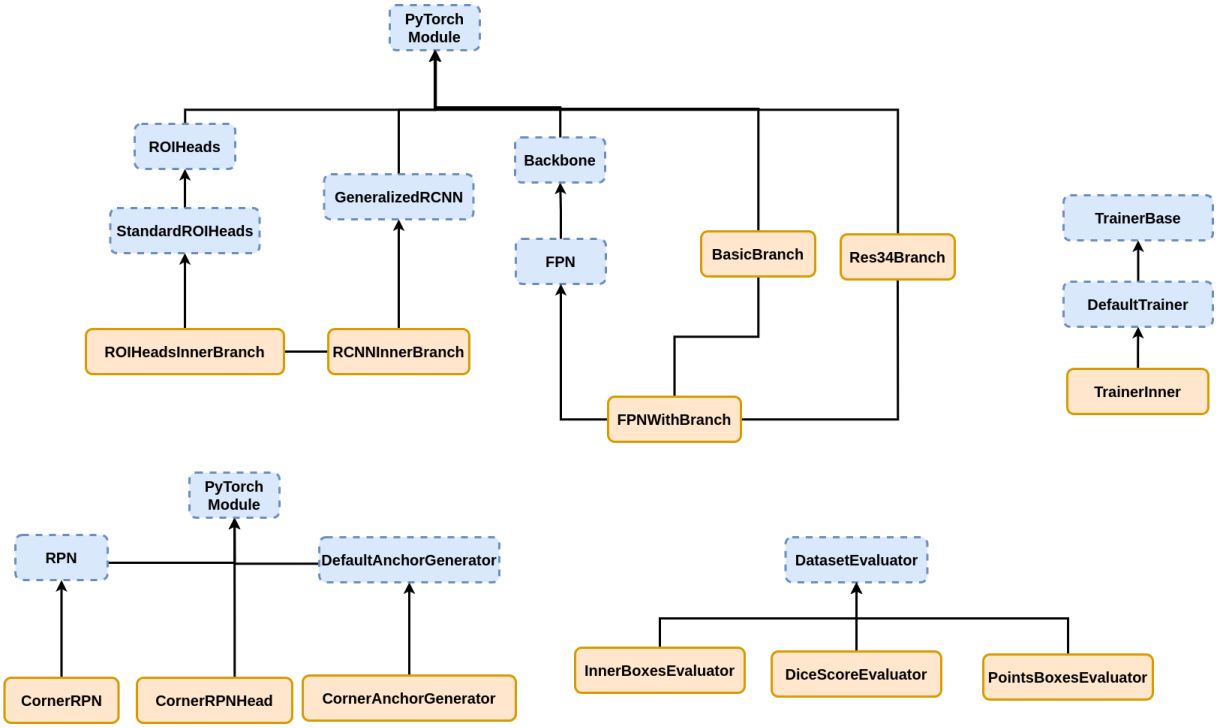


Figure 4.13: Code extension of some functionalities: blue components correspond to PyTorch and Detectron2 classes, orange components correspond to created classes. The arrows correspond to inheritance relation between classes.

- Data registration: in this step, training data and validation data are registered. Detectron has the function "DatasetCatalog.register(dataset name, dataset function)" which registers a dataset into the Detectron catalog. For each dataset a name must be specified and a function must be provided, this function should return the list of annotations using Detectron format. I have developed this function.

- Configuration setting: in this step a Python object corresponding to a training configuration must be created. Detectron offers a pre-initialized configuration, it can be loaded from saved configuration data for a specific neural network like Faster R-CNN or RPN. The user can overwrite some parameters for specific use cases.

- Finally, there is the training step. Detectron offers the class "DefaultTrainer" which encapsulates all the training logic: creating data loader, and performing training iterations. For some particular requirement, I had to extend this class to add other behaviors like using two loaders. The trainer comes with a Python function that starts training, the training progress information and output model are saved in a directory specified by configuration parameters.

4.2 Detectron Prediction

Prediction or inference consists in launching object detection or segmentation on a image. I implemented a general method for inference. Instead of executing inference only on one image, I

developed a technique to execute it on all images of a dataset. The first thing to do is to register the dataset using a Detectron function. Then the configuration object is created and filled with the desired parameters, among these parameters, there is the file location of the saved model. Then a predictor is created using Detectron class "DefaultPredictor". Then an iterator function is launched to perform inference on all images. This function was created by us, its objective is to iterate on all dataset images, for each one it runs a custom processor (in my case it is the predictor which detects objects), and a list of desired displayers, each one is a function that draws a specific information on the image (ground truth bounding boxes, proposals, predictions ...).

4.3 Detectron Evaluation

Evaluation is the task that consists in assessing the performance of a neural network detection by calculating accuracy metrics. A set of annotated images are given to the detector, the detection is launched on these images, the result is compared with real data (ground truth objects). In practice, the dataset used for testing is registered using Detectron library, then configuration parameters are loaded and set, then the evaluation process is executed. Detectron library offers functions and classes to perform evaluations, they are based on COCO metrics calculation and pycocotools library, I had to extend and create my own evaluators to execute specific tasks like displaying more information about calculated metrics, calculating mean average precision from segmentation data and calculating dice score.

4.4 Model Extensibility

Detectron library is very extensible. Thanks to the use of modularity and classes, a developer can add its specific components and behaviors with ease. In fact the neural network architecture is subdivided in different components like RPN, Head layers, Backbone... Each component is implemented through a specific class. A registration mechanism is offered by Detecton library to register components and make them accessible when building neural network architecture. To implement corner approach, I extended the standard class corresponding to RPN into a new class and register it. To activate a desired component, dedicated configuration parameter should be set to the component name.

4.5 Hardware Characteristics

Deep learning is based on neural networks. These structures store huge number of parameters which have to be stored in memory during training and inference. Furthermore, training phase requires more memory as all the layer outputs and intermediary calculated features for each image in the batch should be saved in memory to perform backward propagation. In general, GPUs (graphics processing units) are used to handle this type of processing. In my case, I have used one computer machine with the following characteristics:

- 32 Go of RAM

- 4 cores, 8 CPUs
- 16 Go Nvidia GPU used for training and testing

5 Conclusion

In deep learning, dataset is very essential as it is used for training neural networks. To test my approaches on bee and varroa detection, I have constructed my own annotated dataset by using different tools. I have also developed code to handle annotations and convert them from a format to another.

To assess the performance of proposed approaches, I have selected standard metrics used for object detection and segmentation. Mean average precision is the main metric I used for comparing approaches.

The implementation was realized thanks to Detectron library. It offers ways to construct neural networks, to launch training, inference and evaluation and to extend standard architectures and structures.

Chapter 5

Detection in dense scenes

Résumé

Détecter les objets dans les scènes denses est une problématique fréquente dans le domaine de la détection d'objets. Lorsqu'il y a une densité d'objets élevée dans une scène. La probabilité qu'un objet cache une partie d'un autre objet est grande. Cela signifie que certains attributs d'un objet peuvent manquer lors de la détection. Les détecteurs classiques se concentrent sur la détection du centre. Lorsque cette région centrale est cachée, ce dernier peut être loupé. Afin de pallier cette limite, des approches proposent de prédire des informations en fonction des coins ou des bords de l'objet. La limite majeure de ces méthodes est qu'elles nécessitent des données additionnelles d'annotation. Je propose d'utiliser Faster R-CNN pour détecter les objets à partir de leurs coins. Mon approche prend en considération les régions correspondant aux coins en plus des régions centrales.
Elle se base sur le décalage des "ancres". Une ancre est un modèle de boîte englobante générée pour chaque point de l'ensemble de caractéristiques. Dans l'approche standard, l'ancre est centrée sur chaque point de l'ensemble. Ainsi, l'entraînement du réseau favorise les ancres générées à partir des régions centrales d'objet par rapport aux autres ancres. Je propose de décaler les ancres de façon à ce que le point se situe au coin de l'ancre. Je génère 4 ensembles supplémentaires d'ancres correspondant aux 4 coins d'un objet. De cette façon, lorsque le point correspond à un coin d'un objet, l'ancre correspondante sera favorisée.
Afin de prendre en considération ces nouvelles ancres générées, j'ai dupliqué les autres couches du réseau RPN. J'ai notamment multiplié par 5 les couches de prédiction de scores et les coordonnées de décalage. La génération de propositions est réalisée sur chaque ensemble d'ancres. Cette phase consiste à appliquer les décalages sur les ancres, trier les boîtes obtenues, et appliquer l'algorithme NMS (Non maximum suppression) pour supprimer les doublons. On obtient à la fin un ensemble de boîtes englobantes issu à partir de chaque type d'ancres. Je procède par la suite à la concaténation de ces ensembles pour obtenir les propositions qui sont ensuite classifiées de la même façon que le réseau de neurones standard. Bien évidemment, les fonctions de pertes sont dupliquées 5 fois afin d'entraîner le réseau de neurones RPN et de générer des propositions à partir des coins en plus du centre.

Résumé

Afin de valider la pertinence de mon approche par rapport à la méthode standard, j'ai évalué les métriques de rappel et de précision lors de la détection d'abeilles dans les scènes denses. Le rappel de RPN est amélioré de plus de 10%, celui de Faster R-CNN de 6%.

Contents

1 Introduction

In recent years, bees suffer from many problems related to their health. These issues have different factors. To overcome this kind of issues, beekeepers should monitor bees and their conditions. Computer vision methods based on deep learning could help beekeepers in bee detection which is important for bee monitoring and counting. In fact, there are already advanced neural networks for object detection like Faster R-CNN. This network is detecting objects in an image through two stages. First RPN generates proposals then they are classified and enhanced using another sub-network. RPN takes a set of feature maps calculated by a backbone neural network. For each position in the map, it predicts a predefined number of possible objects. To achieve this prediction, a set of bounding boxes called anchors are generated at each feature map position. RPN predicts whether each anchor matches an object or not and estimates the offset coordinates to correct the object position. The anchors are centered on the position. An anchor corresponds to an object when its IoU (intersection over union) with a ground truth is high. This means that anchors generated at the centers of the objects are more likely classified as objects, while anchors generated at the corners of the objects are more likely classified as non-objects. In dense scenes, there are objects that are partially visible; only the corners are visible. For example, in bee frame images, the probability that a bee part is hidden by another bee is high. The standard approach could miss these kinds of objects. To overcome this issue, I propose to enhance anchor generation by generating anchors from object corners in addition to object centers. Concretely, four additional types of anchors are used; each one corresponds to an object corner (top left, top right, bottom left, bottom right). At each feature map position, the corner anchor is generated so that the feature map point is located at the anchor corner. I duplicate the RPN prediction components and losses to handle the corner data. To test my approach, I constructed and annotated a specific dataset composed by images of bee frames. To analyze my approach, I made different modifications on my proposed approach and tested them on two databases of different density levels. My contribution consists of an extension of standard Faster R-CNN architecture to predict objects from corners and improve recall in dense scenes. Code is available at *https://github.com/yassine-kr/RPNCorner*. I start by citing some related works to my research in Section 5.2. Then, in Section 5.3, I present my motivation to choose this research direction. In Section 5.4, I explain my proposed approach. After that, in Section 5.5, I show my obtained results on bee detection. Finally, I discuss the results in Section 5.6.

2 Related Work

My work aims to improve bee detection using deep learning. In this Section, I cite some works that are related to my subject. First I present the dense scene issue, and the existing approach dealing with this situation. Anchors are also an interesting topic as my method is based on the generation of a new set of anchors, related work about anchors is described. Finally, I cite some already proposed approaches based on corner data to detect objects.

2.1 Dense Scene

In the field of object detection, a dense scene means a scene where the density of objects is very high, in other words it means that the number of objects per unit area is large. In this case, there is a high probability that the image contains two objects that are adjacent or occluding each other. Performing accurate detection using machine learning for computer vision in these conditions could be challenging. There are many works dealing with these issues in the context of detecting people and pedestrians in crowded scenes, such as [Liu, 2019] where authors present a method which predicts whether two overlapping bounding boxes match the same object or two different objects. In [Zhang, 2018], the authors try to improve the detection in pedestrian crowded scenes by proposing another way of calculating the loss which aims to improve the compactness of the predicted bounding boxes around the ground-truth, and to improve the prediction of the human body by detecting its parts. The paper [Xi, 2020] deals with the detection of human faces in crowded scenes with low resolution by exploring the similarity between detected objects. Finally, [Zhang, 2019] propose to use two anchors to detect people in a crowd; one for the head and one for the body.

Concerning object detection in a general context; the paper [Gählert, 2020] attempts to improve object detection in images through using pixel-base bounding box annotations. [Goldman, 2019] detect objects in a dense scene using a new layer of prediction called "Soft-IoU" which predicts IoU of a predicted object bounding box compared to the real object position.

2.2 Anchors

In the state of the art of object detection neural networks, anchors are used to detect objects. They are bounding boxes of different sizes and ratios generated by the neural network. It generates a predefined number of anchors at each position of the feature map. For each of them, a score is predicted, it corresponds to the probability that the anchor contains an object. The learning phase is based on the comparison of anchors with ground-truth bounding boxes. Each anchor is centered on its corresponding feature map position, this matching enables the neural network to predict objects from their centers.

On the other side, there are anchor-free detectors such as [Tian, 2019] which do not use anchors. In this approach, for each feature map point, object bounding box coordinates are directly predicted. This approach allows to predict objects from all their pixels.

2.3 Corner Detection

To deal with dense scene and occlusion constraints, object corner information could be used. My approach is based on generating proposals from corner in RPN generator. There are already existing approaches that are also based on corner information.

The paper [Qiu, 2020] presents an approach called BorderDet which aims to improve the detection of objects in dense scenes by using feature maps of borders in addition to standard feature maps of centers. This way allows not to miss partially hidden objects. But the limit of this method is that it needs a first coarse prediction to extract border points.

[Zhou, 2019b] suggests to detect the extreme points and the central point of an object. The final detection is obtained by grouping these points. The proposed approach is based on the detection of key points using heat maps and CornerNet [Law, 2018]. This method needs extreme points annotations.

[Wei, 2020] uses point-set anchor instead of a rectangular box anchor, this permits to represent objects more accurately. This approach is mainly interesting in the case of image segmentation and pose estimation, where ground-truth annotations are not standard bounding boxes.

[Duan, 2020] proposes an anchor-free and two stage object detector, based on corner proposals. These proposals are generated by predicting two kinds of corner heat maps; top-left and bottom-right. Corners are extracted from these maps and a fixed number of proposals are then generated.

3 Object Detection in Dense Scenes Motivation

In standard Faster R-CNN, RPN generates proposals from object centers. In fact, it takes feature maps as input, and generates anchors for each point in each map. An anchor is placed in a way that the point is located at the anchor center. The network predicts for each anchor, objectness (a score corresponding to the probability of the anchor bounding box to be an object) and offsets to correct the anchor position and match the real object. The training phase compares each anchor to ground truth bounding boxes, using IoU metric (Intersection over Union). The anchors with big IoU are considered as objects (ground-truth objectness is equal to 1). The anchors generated at corner regions are likely to be classified as non-objects as their IoUs with ground truth bounding boxes are small. If the image contains objects whose only visible parts are corners, they are more likely to be missed by the network. In the case of bee images, this situation is frequent due to the high density of bees in the bee frame image. The RPN has a direct influence on the recall of Faster R-CNN; if an object is missed during proposal generation, it is almost impossible to detect it by Faster R-CNN. I think that this issue can be resolved by generating another kind of anchors, using another way of placing them on feature map points. Each one of this new type of anchors is placed in a way that the point is located at a corner anchor. This way would enable RPN training to consider anchors generated at object corner regions as objects (ground truth objectness is equal to 1) and avoid missing partially hidden bees. My work is motivated by these elements:

- In standard anchor-based neural networks for object detection, prediction is based on object center information, corner information is not taken into account.
- Improving anchor generation could be used in all anchor-based neural networks like YOLO.
- Generating anchors influences on detection recall. It is important to generate anchors able to take into account the type of visible region (center or corner).
- Bee detection and counting are interesting tasks for beekeeping domain. Improve accuracy of detection remains an important objective to reach.

4 Approach

Faster R-CNN is a two stage neural network. It uses RPN as a proposal generator. This component generates anchors centered on the feature map positions. This way of placing anchors favors objects whose central regions are visible. I propose to generate another kind of anchors in order to predict objects from corners and improve detection recall.

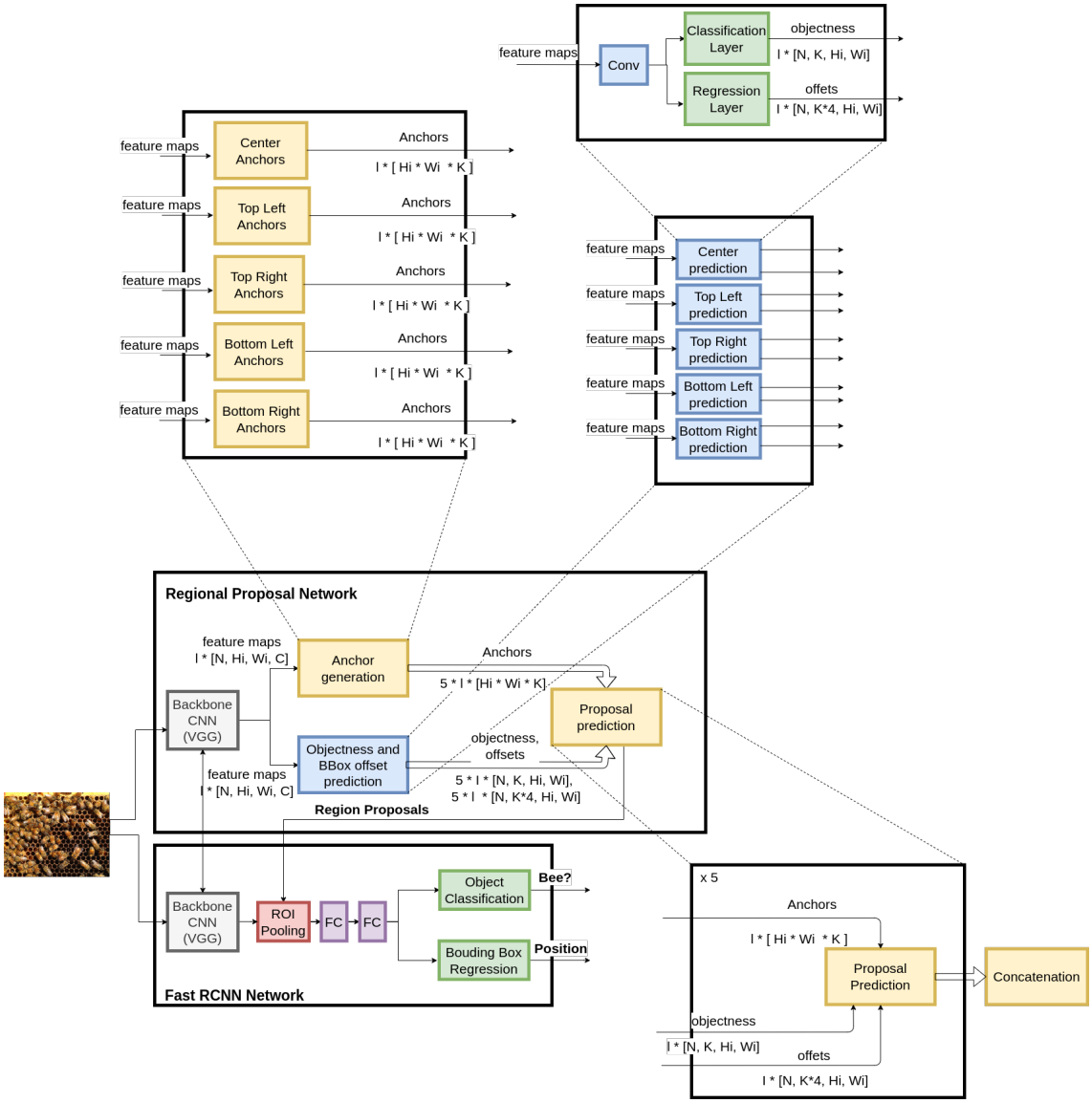


Figure 5.1: General Architecture of Faster R-CNN based on corner detection. l: number of feature maps, N: number of images, Hi: height of feature map i, Wi: width of feature map i, C: number of channels, K: number of anchors per feature map position.

4.1 General Architecture

The proposed neural network is based on Faster R-CNN. Figure 5.1 shows the structure of the network. My work focuses on RPN, the purpose of this sub-network is to generate region

proposals which are then classified and enhanced using another sub-network. The first step computes feature maps using a pre-trained classification neural network like VGG or ResNet. Then objectness and offsets are predicted. After that the predicted offsets are re-used by anchor generation mechanism to create proposals which are ordered using objectness scores. In the standard approach, proposals are generated from object centers. My approach consists in generating proposals from object corners. To achieve this objective, I modified the anchor generation mechanism and extended the neural network to predict corner data. Section 5.4.2 explains the types of generated anchors. Section 5.4.3 presents the added corner predictors. Section 5.4.4 describes how generated anchors and prediction data are combined to create proposals. Section 5.4.5 clarifies the used method for labeling anchors during training. The last Section (5.4.6) cites the used losses for training.

4.2 Anchor Generation

The aim of the standard anchor generation is to detect objects from their centers. In the case of dense scenes, where usually objects are partially occluded, the central region of the objects could be hidden, therefore some objects could be missed. Anchors are a set of bounding box candidates for predicted objects. They are generated on feature maps. In the case of FPN [Lin, 2017a], more than one feature maps are used to detect objects of different sizes. I extend the anchor generation mechanism by adding four anchor generators: top left anchors, top right anchors, bottom left anchors and bottom right anchors. For each generator, K anchors (in my case $K = 3$), of different sizes and ratios, are generated at each position for each feature map. Let s be the dimension of a feature map stride (the image region corresponding to the feature map point), w the anchor width, h the anchor height, and $x1$, $x2$, $y1$ and $y2$ anchor coordinates in the format $X1X2Y1Y2$ ($X1$: x-coordinate of left border, $X2$: x-coordinate of right border, $Y1$: y-coordinate of top border, $Y2$: y-coordinate of bottom border), in the coordinate system whose origin is the center of the stride:

- Central anchor:
  $x1 = -w/2$, $x2 = w/2$, $y1 = -h/2$, $y2 = h/2$
- Top left anchor:
  $x1 = -s/2$, $x2 = w - s/2$, $y1 = -s/2$, $y2 = h - s/2$
- Top right anchor:
  $x1 = -w + s/2$, $x2 = s/2$, $y1 = -s/2$, $y2 = h - s/2$
- Bottom left anchor:
  $x1 = -s/2$, $x2 = w - s/2$, $y1 = -h + s/2$, $y2 = s/2$
- Bottom right anchor:
  $x1 = -w + s/2$, $x2 = s/2$, $y1 = -h + s/2$, $y2 = s/2$

Figure 5.2 illustrates the five types of generated anchors.

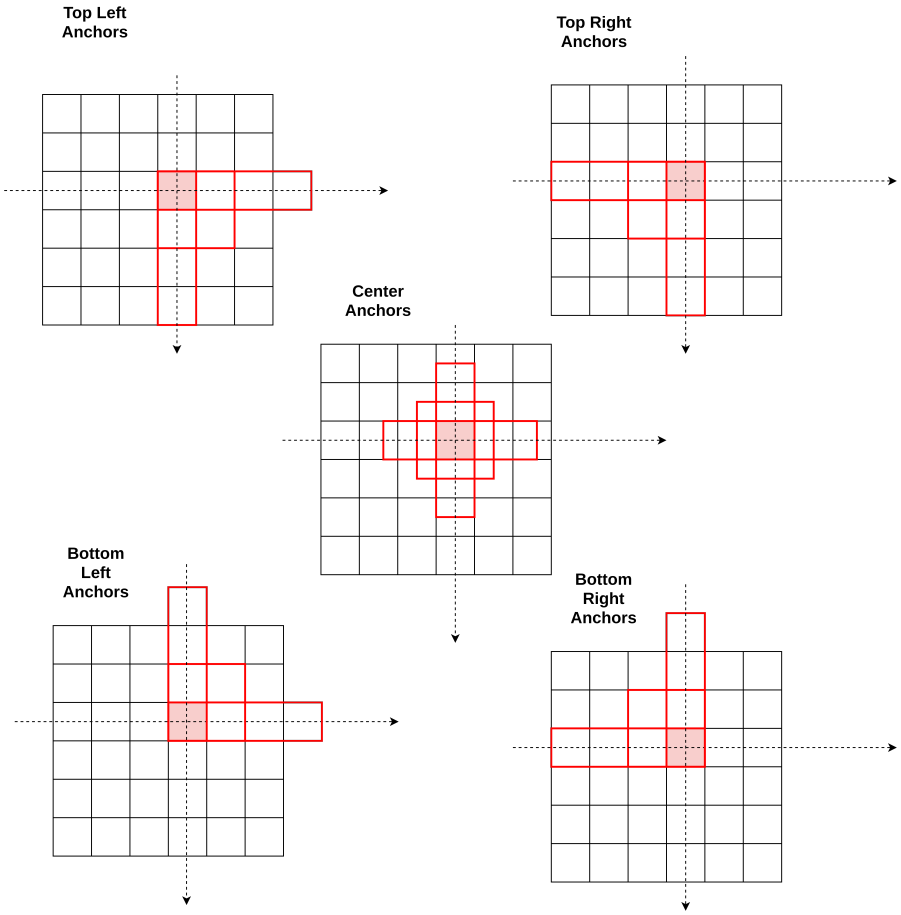


Figure 5.2: Types of generated anchors from feature maps

4.3 Objectness and Offset Prediction

Instead of using one layer for predicting objectness and offsets, four other parallel layers are added to predict corner data. In total there are five branches with different weights; (1) the first to generate proposals from object centers, (2) the second to generate proposals from object top left corners, (3) the third to generate proposals from object top right corners, (4) the fourth to generate proposals from object bottom left corners, (5) the fifth to generate proposals from object bottom right corners. Each branch is made up of two layers; the first one takes as input the feature maps generated by the backbone, and applies a convolution operation. The second layer contains two sub-branches, the first is a convolutional layer to predict objectness at each feature map position, the second is a convolutional layer to predict box offset coordinates to correct proposal positions.

4.4 Generating Proposals

Proposal generation is performed by combining the prediction data and the generated anchors. I duplicate this step four times to generate proposals from corner data. It consists in applying predicted offsets on generated anchors, then a predefined number of best proposals is preserved from each feature map, the NMS (non-maximum suppression) algorithm is applied on each set of feature map proposals to reduce multidetection, and finally a predefined number of best proposals is preserved from all proposals. A proposal is better than the other when its objectness score is higher. The final set of proposals is simply a union of proposals generated by the five generators.

4.5 Anchor Ground Truth Labeling and Offset Assigning

The anchor labeling consists in giving each anchor a value 0 or 1 according to whether the anchor contains an object or not. Offset assigning is the process of giving each anchor the offsets that must be applied to the anchor to reach the correct position. The aim of training is, for each anchor, to predict the objectness and offsets that should be as close as possible to anchor assigned label and offsets. My approach uses standard labeling and assigning; for each anchor, the ground truth bounding box with which the anchor has the greatest overlapping score (IoU: Intersection over Union) is mapped to the anchor, if this IoU is greater than a fixed threshold (0.7 in the standard implementation of Faster R-CNN), the anchor is labeled with 1, if it is less than a fixed threshold (0.3 in the standard implementation of Faster R-CNN), the anchor is labeled with 0, otherwise it is ignored. The differences between the anchor borders and the borders of the mapped bounding box are the assigned ground truth offsets.

4.6 RPN Losses

As demonstrated in Figure 5.3, there are two losses in RPN; objectness loss and offset loss. To balance between positive and negative samples, a sampling is performed. The training consists in optimizing the sum of these two losses:

- Objectness loss:

$$L = -\frac{1}{N_{\text{cls}}} \sum_{i=1}^{N_{\text{cls}}} t_i log(p_i) + (1 - t_i) log(1 - p_i) \tag{5.1}$$

- Offset loss:

$$L = \frac{1}{N_{\text{reg}}} \sum_{i=1}^{N_{\text{reg}}} R(o_i - o_i*) \tag{5.2}$$

Where R, N_{cls}, N_{reg}, p_i, t_i o_i, o_i* are respectively the smooth L1 loss [Girshick, 2015], the number of anchors to classify, the number of anchors to regress, the predicted objectness, the ground truth label, the predicted offsets, and the ground truth offsets.

For training on different layers, I add similar losses for the predicted corner information. The final loss is an equally weighted sum of the five losses.

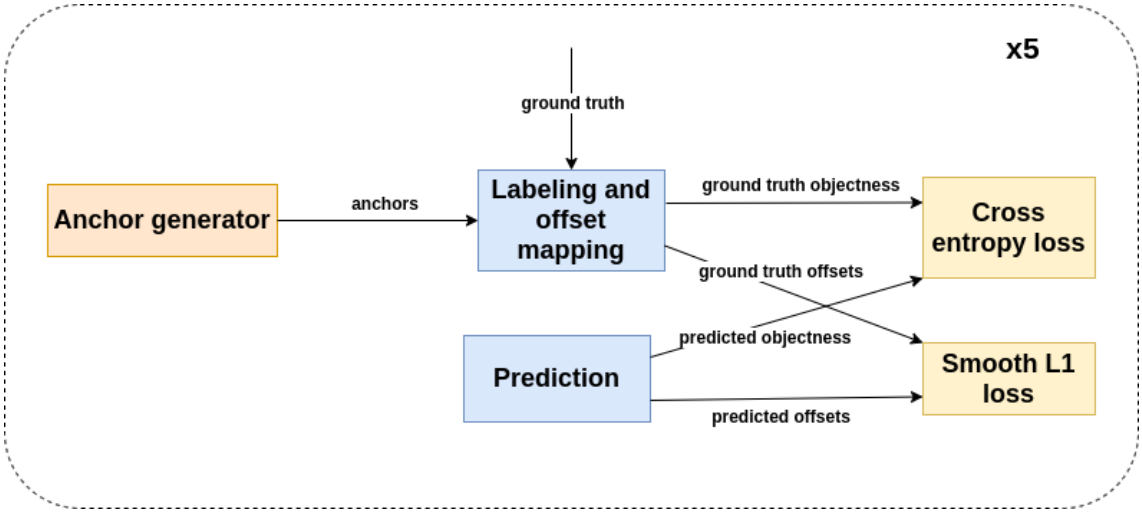


Figure 5.3: RPN losses. These two losses are duplicated five times to train predictors on center and corner data.

5 Experimental Results

To assess the accuracy of my approach, I used conventional object detection metrics: average recall and average accuracy. Since my approach consists of modifying RPN, I tested the recall of RPN in addition to the accuracy of the whole neural network. The extendibility and modularity of Detectron2 framework enabled us to create and integrate my custom subcomponents like anchor generators and RPN predictors. In this Section, I present the results of my corner approach. I start by describing the created datasets for training and testing in Subsection 5.5.1. Then, in Subsection 5.5.2 I explain the library parameters that I set for my use case. In Subsection 5.5.3, I discuss and compare the accuracy results that I obtained after training and testing the standard approach and my approach.

Table 5.1: Accuracy of standard approach and my corner approach. RPN AR@1000: RPN average recall on 1000 first detections using 0.5 IoU, Faster R-CNN AR: Faster R-CNN average recall on 100 first detections using 0.5 IoU, Faster R-CNN AP: Faster R-CNN average precision on 100 first detections using 0.5:0.95 IoU. Number of proposals before RPN NMS is 200, number of porposals after RPN NMS is 1000.

Approach \ Metric		RPN AR@1000	Faster R-CNN AR	Faster R-CNN AP
	Metric type	Recall	Recall	Precision
Standard Approach	Normal density	76.45	76.45	46.03
	High density	56.06	53.87	27.6
Corner Approach	Normal density	86.73	82.8	52.17
	High density	66.88	55.82	28.54

5.1 Train and Test Data

I created three image sets; (1) the first set was annotated by our partners, it contains arbitrary images coming from our dataset sources. They were not completely annotated; highly occluded bees were generally not annotated, because of the effort required by these annotations. (2) The images of the second set were purposely selected to obtain a database with high object density. Moreover, they were almost entirely annotated. (3) The third set contains images with a lower object density. I took 22 images from my first set with 1086 annotations. I created two test sets: (1) the first one contains 11 images with 535 annotations coming from my first image set, (2) the second contains 25 images with 1691 annotations coming from the second set of images. By using two image sets I aimed at comparing my approach under two situations: images with normal density and images with higher density.

5.2 Parameters

I mainly used the default parameter values set by the Detectron library except for some parameters. For training, a pretrained Faster R-CNN model with FPN architecture was used. This model uses a pretrained ResNet-50 backbone trained on ImageNet dataset. The model was trained on COCO dataset. As long as transfer learning is used, 8000 iterations of training on bee images was enough to achieve state-of-art results. The number of preserved proposals before and after NMS algorithm were also fixed through two specific Detectron parameters. I changed the default values because the relevance of my approach depends on that parameters.

5.3 Result Analysis

Table 5.1 shows the results of the tests for the standard approach against the proposed approach. Recall and precision are significantly improved by the corner approach. The recall evaluates the number of detected objects compared to ground truth ones. The precision corresponds to the proportion of correct detections among all detections.

To compare the standard approach to the corner approach, the number of RPN output proposals must be the same. The number of proposals preserved before NMS is 200. When

FPN is used, 200 best proposals are retrieved from each feature map. There are 5 feature maps, it means that 1000 proposals are preserved. After NMS, in the standard case, 1000 best proposals are preserved, it means that all the proposals are preserved. In the case of my approach, 200 best proposals are preserved from each center/corner prediction after NMS, before merging the five sets to obtain 1000 proposals. These parameter values were chosen because of the constraint of the inference. In fact NMS is a greedy algorithm, it requires time and resources. Providing a big number of proposals to NMS is not very convenient if quick detection is required.

The results demonstrate that my approach detects more objects than the standard. This is due to the predicted corner data which enables the neural network to detect bees from corners.

Table 5.2: Accuracy of my corner approach when some modifications are applied: (1) One loss instead of five losses, (2) One branch for detecting corner data instead of four branches, (3) Using shifted convolution in corner branches instead of normal convolution. RPN AR@1000: RPN average recall on 1000 first detections using 0.5 IoU, Faster R-CNN AR: Faster R-CNN average recall on 100 first detections using 0.5 IoU, Faster R-CNN AP: Faster R-CNN average precision on 100 first detections using 0.5:0.95 IoU. Number of proposals before RPN NMS is 200, number of porposals after RPN NMS is 1000.

Approach \ Metric		RPN AR@1000	Faster R-CNN AR	Faster R-CNN AP
	Metric type	Recall	Recall	Precision
One loss	Normal density	85.79	81.31	51.95
	High density	65.58	52.27	28.35
One branch	Normal density	82.82	82.61	48.71
	High density	64.7	59.43	30.02
Shifted convolution	Normal density	84.67	81.87	51.94
	High density	63.87	54.35	29.02

6 Discussion

To understand my results, and explore my approach advantages and limits, I have analyzed some aspects of them. First, I verified the accuracy of the center predictor alone, this is explained in Subsection 5.6.1. In Subsection 5.6.2 I talk about the influence of the number of the preserved proposals on the neural network accuracy. In Subsection 5.6.3 I focus on the possibility of merging the proposals before NMS. On the other hand, I explored using one loss instead of five losses and one branch instead of five branches as explained in Subsection 5.6.4 and Subsection 5.6.5. In Subsection 5.6.6, the new convolution method that I used to enhance corner data representation is described. Finally, in Subsection 5.6.7, I analyze the results that I obtained when I trained my neural network on bee images with low density.

6.1 Accuracy of Center Data

When I compared the accuracy of the standard neural network and the network based on my approach using only center prediction, the standard gives better results. Indeed, it is more difficult

for the training to find backbone parameters that satisfy all the objective losses. Nevertheless, the results demonstrated that the corner data detected by proposed approach compensates the reduced accuracy of center data prediction.

6.2 Number of Preserved Proposals before NMS

I noticed that the accuracy of my approach is not good when the number of preserved proposals from each feature map exceeds 200. It is is certainly related to my training set. This limitation is not present in other cases as I will see later in this thesis. Further research work must be carried out to understand the reason of this limitation.

6.3 Proposal Concatenation

The proposed approach is based on the concatenation of the proposals after NMS. Merging these proposals before NMS algorithm is another possibility: in this case, the predicted offsets for each branch (center/corner) are applied to the anchors to obtain proposals, these bounding boxes are concatenated in a set, then they are ordered by their objectness score, and NMS is applied to reduce multi-detection. The advantage of this method is that it executes the NMS algorithm once instead of five times. But its results were not satisfactory. Furthermore, in this alternative approach, NMS should be applied to a greater number of proposals which requires a greater use of resources.

6.4 One Loss vs Five Losses

Instead of using one loss for each corner or center, only one loss could be used. In fact, this can be implemented by combining corner and center predicted data in a single layer. The five anchor generators are replaced by one generator which generates the five types of anchors. The labeling and mapping step is performed on all the generated anchors. As demonstrated by Table 5.2 this approach gives lower results than the one based on five different losses.

Table 5.3: Accuracy of standard approach and my corner approach when training on dataset with low object density. RPN AR@1000: RPN average recall on 1000 first detections using 0.5 IoU, Faster R-CNN AR: Faster R-CNN average recall on 100 first detections using 0.5 IoU, Faster R-CNN AP: Faster R-CNN average precision on 100 first detections using 0.5:0.95 IoU.

Approach \ Metric		RPN AR@1000	Faster R-CNN AR	Faster R-CNN AP
	Metric type	Recall	Recall	Precision
Standard Approach	Normal density	62.99	45.61	23.01
	High density	49.67	21.94	10.81
Corner Approach	Normal density	65.42	55.33	27.69
	High density	53.93	31.52	14.27

6.5 One Branch for Corner Data

A similar approach was also tested. Instead of using four different branches for the corner data, one branch is used. This method has the advantage of reducing the number of parameters to learn. Despite of the constraint of representing different types of corners with the same set of parameters, the approach results are promising as shown in Table 5.2.

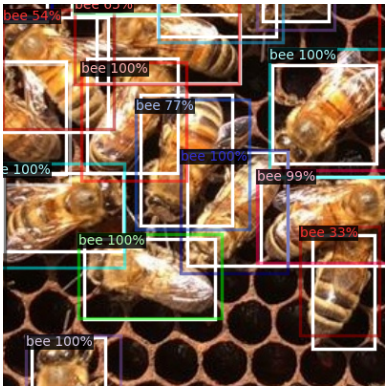
6.6 Shifted Convolution

My approach is based on predicting objectness and offsets from object corners. Whether or not a feature map region corresponds to a corner depends on the region content and its surroundings. Regarding the corners, the prediction is more influenced by the region containing the object. This region is not encircling the corner as it is the case for object center, but it is shifted. For instance, for a top left corner, instead of using a surrounding centered on the corner point, this surrounding should be shifted to right and bottom. Therefore instead of convolving the points of standard surrounding, the convolution is made on this new surrounding which contains more relevant corner data. The results of this approach are presented by Table 5.2.

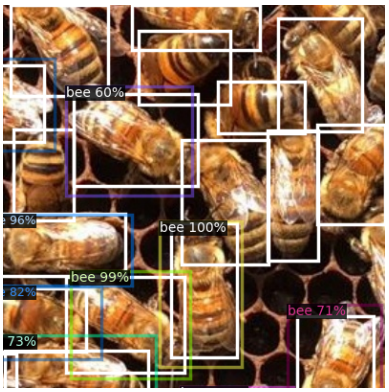
6.7 Sparse Images as Training Set

To check the relevance of my approach, I trained the neural network on a simple database made up of 22 images with only 424 annotations. In fact these images contain sparse bees. The objective was to verify whether the approach could detect partially visible objects using only visible parts which are mainly bee corners. So I used a sparse image set to prevent the network to learn hidden objects by considering them as small. The results are shown in Table 5.3. The limits of those results are that the accuracy is very low compared to the values of state of the art. In fact it is certainly due to the limited number of annotations used for training. But the advantage of that case, is that there is no limit to the number of considered proposals before NMS. More investigation should be conducted in the future to understand the relation between the training annotations and the performance of the approach. Figure 5.4 shows that my approach detects some occluded bees while the normal approach could not predict them.

(a) Detected bees in the first image using standard approach.

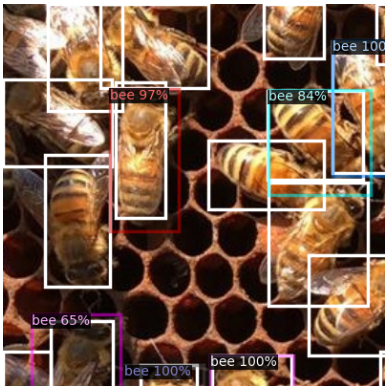


(b) Detected bees in the first image using corner approach.



(c) Detected bees in the second image using standard approach.

(d) Detected bees in the second image using corner approach.



(e) Detected bees in the third image using standard approach.

(f) Detected bees in the third image using corner approach.

Figure 5.4: The detected bees in three images using standard and proposed approaches. (a) and (b) relate to the same image. (c) and (d) relate to the same image. (e) and (f) relate to the same image. Colored rectangles correspond to predicted bounding boxes. White rectangles correspond to ground truth bounding boxes.

7 Conclusion

In this chapter, I propose a new approach to detect objects in dense scenes. It is an extension of the standard Faster R-CNN. In addition to predicting whether the positions of the feature map correspond to the object centers, I predict whether they correspond to the object corners. This is achieved through generating a new set of anchors and predicting their corresponding objectness and offset data. My approach aims to improve object detection in the case of bee frame images. Therefore, I built and annotated a specific dataset. I executed training and testing to evaluate my approach. When the number of the preserved proposals before NMS is less than 200, my approach performs better than the standard. So it can be used in situations where computation resources are limited. Moreover, my proposed neural network can be used in other object detection contexts. Since my approach is only to extend anchor generation mechanism, it is not specific to Faster R-CNN, but it can be exploited in other anchor based neural networks like YOLO. I hope that my approach can help beekeepers to monitor their beehives more accurately, and that my work can contribute to the advancement in computer vision research.

I have also demonstrate that the NMS limitation is not presented when a low density dataset is used for training. But I think that this limitation must be addressed in the future. How to improve recall independently to the number of preserved proposals should be treated in more depth. On the other hand, my approach should be tested in other contexts and neural networks.

Chapter 6

Detection of Nested Objects

Résumé

Dans une image, les objets présentent des relations différentes entre eux. Parmi ces relations, on retrouve l'imbrication: un objet peut se retrouver à l'intérieur d'un autre. Ce cas s'observe dans différentes situations, comme par exemple lorsque des varroas sont situés sur des abeilles. En effet, dans un cadre d'abeilles, les varroas sont souvent positionnés sur le dos des abeilles. La détection des abeilles et des varroas est très utile pour estimer le niveau d'infestation et aider les apiculteurs à prendre des décisions.

La détection des objets imbriqués sur les images peut être coûteuse en terme de ressources. Les méthodes classiques de résolution sont 1) l'entraînement d'un réseau de neurones détectant des objets de types différents ou 2) l'entraînement de deux réseaux de neurones où chacun se charge de détecter les objets d'un type donné. Ces approches ont des limites surtout en terme de ressources utilisées. Je propose d'utiliser le réseau de neurones Mask R-CNN pour la détection des objets principaux (abeilles) et la segmentation des objets internes (varroas). Ce réseau de neurones extrait des informations communes à partir de l'image d'entrée et applique des couches spécifiques pour améliorer les informations extraites. Après des expérimentations et des tests, j'ai constaté que l'extraction des informations issues de la couche "Res4" de Mask R-CNN donne de meilleurs résultats pour la détection d'abeilles et la segmentation de varroas.

Afin d'entraîner le réseau de neurones, et tenir compte de la grande différence des taux d'apparition par image de chaque type d'objet, j'ai construit deux bases d'images annotées. J'ai proposé deux manières de réaliser l'apprentissage sur deux bases de données différentes. La première consiste à traiter par chaque itération d'entraînement deux types de données, les images annotées des abeilles, et les images annotées des varroas. La deuxième consiste à réaliser deux entraînements consécutifs, chacun est réalisé sur une base d'images différente.

Mes expérimentations montrent que le réseau de neurones qui améliore la précision de détection des varroas par rapport à l'approche standard consiste à détecter consécutivement les objets, et ceci sans perte de précision de la détection des abeilles.

Contents

1 Introduction

Varroas are small parasites that attack bees. They are among the main causes of bee health degradation [Sipos, 2021]. Beekeepers monitor varroas by using traditional methods based on sampling. Using artificial intelligence to evaluate the number of these parasites can help beekeepers estimate the level of infestation of their colonies. Varroas and bees are nested objects, detecting them could be demanding in terms of computer resources and time. To overcome this constraint, I propose to perform nested object detection through one neural network based on Mask R-CNN architecture. This network predicts bee positions, for each detected bee, segmentation is performed to find varroa pixels. The neural network extracts common features from the input image, and applies specific layers to enhance the information retrieved for each object type. Due to the large difference between the probability of occurrence in the images of the two object types, two different annotated dataset were built. Neural network was trained on these dataset. I obtained one neural network able to detect nested objects and with results similar to using two separate neural networks. In this chapter, I explain my proposed approach for nested object detection. I start by presenting the state of the art related to my subject in Section 6.2. Then I explain my motivation in Section 6.3. In Section 6.4, I describe the proposed approach. Finally, in Section 6.5, I show and discuss the obtained results.

2 State of the Art

In this section, I start with a reminder of the object detection task in Subsection 6.2.1, then in Subsection 6.2.2 I explain how an object detector performs classification of objects of different types. In Subsections 6.2.3 and 6.2.4, I focus on the segmentation task by describing two neural networks used for semantic segmentation and instance segmentation. Then, in Subsection 6.2.5, I talk about existing approaches dealing with the problematic of detecting nested objects.

2.1 Object Detection

Since the emergence of machine learning and deep learning, computers became more efficient on object detection tasks. Indeed, thanks to the increase in computing power and the ability to process big data, training algorithms executed on neural network structures give more interesting results. The object detection task consists in detecting objects of one or many defined classes. For each object inside the input image, the task should provide a rectangle surrounding the object region and its class. This rectangle is called bounding box. There are different approaches based on deep learning used for detecting objects. There are two main types of object detectors; one-stage object detector and two-stage object detector, for more information, see Section 3.4.

2.2 Multi-class Object Detection

Through using image classification mechanisms, the object detector could detect objects of different classes. When the input image contains objects of different classes (e.g. cats, dogs, bees ...), in addition to detecting object positions, the object detector detects also the class

of each object. In the case of Faster R-CNN, this classification is performed by a stack of convolutional layers. Figure 6.1 shows the structure of the sub-neural network of Faster R-CNN responsible for predicting object classes. After extracting features from the proposal region by the first and second fully connected layers, the next layer calculates the score for each class, the softmax layer calculates the probability that the proposal belongs to each class (background class included), the output proposal class corresponds to the highest probability.

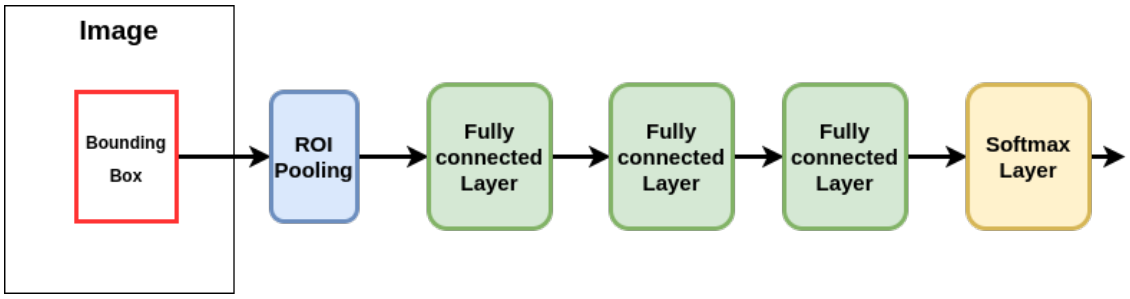


Figure 6.1: Faster R-CNN branch that performs proposal classification

2.3 U-Net

U-Net is a well known neural network proposed for performing segmentation task [Ronneberger, 2015]. It has a hierarchical symmetric structure. Figure 6.2 shows an example of U-Net architecture. First, the input image passes through a downscale phase which extracts features from the image, the result is a feature map with smaller dimension and higher number of channels which contains semantic information about the image. Then the result passes through an upscale phase which increases the size of the feature map to reach the segmentation result. As it is used in many neural network architectures like ResNet for image classification [He, 2016], U-Net contains skip connections which connect downscale output layers to the corresponding upscale layers, this structure enables the network to avoid losing image information and increase learning capacity.

2.4 Mask R-CNN

Mask R-CNN is a neural network used for performing instance segmentation, it is an extension of Faster R-CNN neural network. To detect object pixels, a new branch is added to the neural network. For each bounding box predicted by Faster R-CNN, the corresponding features are extracted from the feature maps generated by the backbone, a smaller feature map is obtained as result. The mask branch classifies feature map points into object/not object. To get the final segmentation, a projection is performed on the original image. Figure 6.3 shows the architecture of Mask R-CNN neural network.

The backbone extracts features from the image, these features are used for generating proposals, classifying them and predicting object pixel regions. When FPN is used, the backbone generates more than one feature map of different sizes. As explained in Subsection 3.4.5, to

[1] https://fr.wikipedia.org/wiki/U-Net

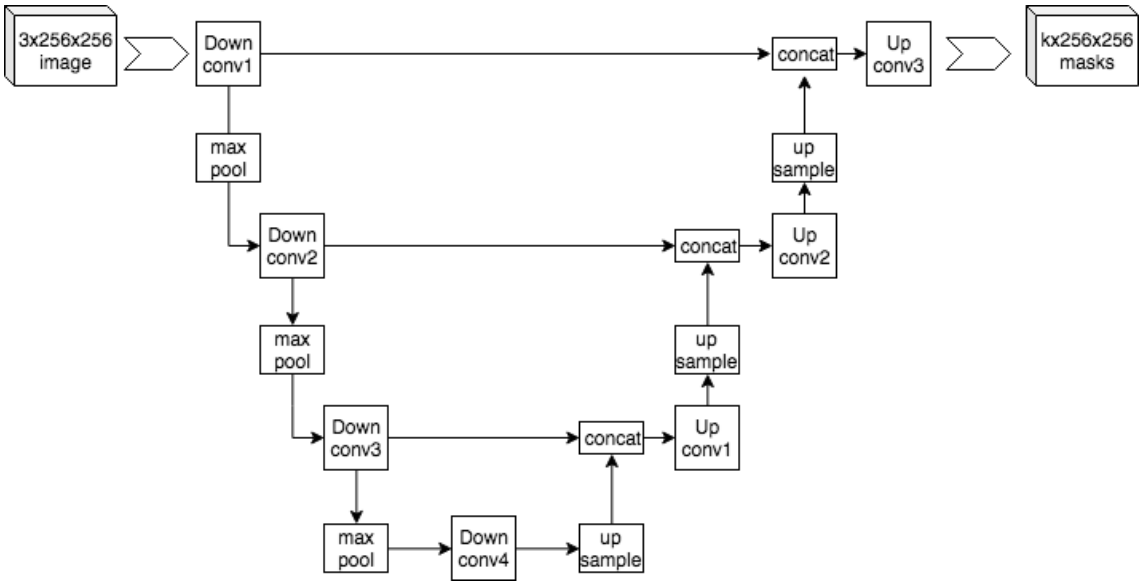


Figure 6.2: U-Net Neural Network Architecture Example [1]

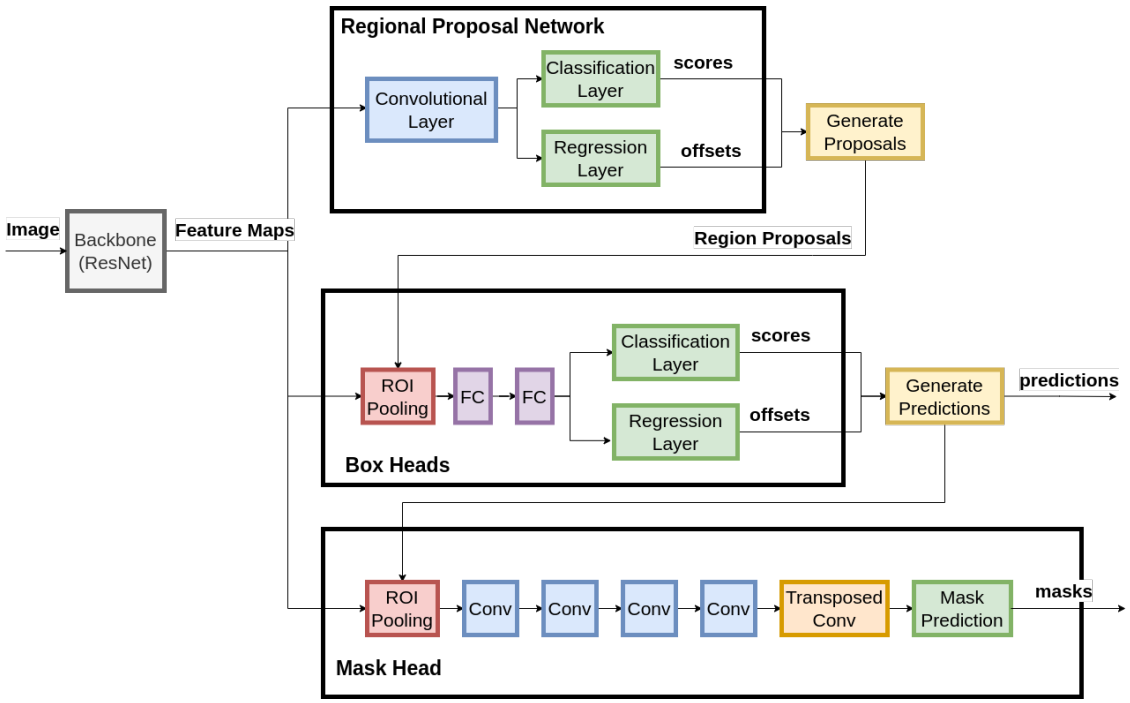


Figure 6.3: Mask R-CNN Neural Network

classify proposals, ROI Pooling selects a corresponding feature map from the generated feature maps according to the bounding box size, it projects the bounding box on it, and extracts a new feature map matching the proposal. Then the classification is performed on these obtained data. If the proposal is classified as an object, segmentation processing is performed on the extracted feature map.

Mask R-CNN Head is composed of a stack of convolution layers (the standard implementation contains 4 convolutions), then the output is given to transposed convolution layer which increases the size of the feature map, then the mask prediction layer is performed to predict masks. This layer classifies each feature map point to object/no object pixel. For more information on transposed convolution see this tutorial [Deep Learning, 2022].

To train Mask R-CNN neural network, instead of using generated bounding boxes predicted by Faster R-CNN as input of Mask Head, generated proposals are used.

2.5 Nested Object Detection

Sometimes we need to detect objects that are localized inside other detected objects. Computer vision domain deals also with such a task. A typical example is the face and eye detecting task. It must detect faces and eyes of humans present in the input images. Two approaches could be used to accomplish this task, detecting eyes and faces directly on input images, or detecting faces first, then detecting eyes on detected face regions. To directly detect faces and eyes, a same object detector is used, it must detect objects of two different classes (faces and eyes), this approach needs to learn face and eyes features by a same neural network which can be more challenging than detecting eyes from face features. Consecutive detection is implemented using two neural networks, the first takes whole human images as input, the second one takes face images as input. Thus, the architecture of the detector is heavier than the first case, it needs more memory to save parameters and more time to train. This can be improved by detecting eyes directly from face features already calculated by the first neural network.

The faces and eyes detection example is similar to detecting bees and varroas, except that the probability of presence of nested object is not the same: the eyes are always present inside a face while the probability of existence of a varroa inside a bee is very low.

This is new and challenging issue. To my knowledge there is no solution published yet for dealing with issue of detecting nested objects of different occurrence probabilities.

3 Nested Object Detection Motivation

Varroas are parasites that attack bees. They infiltrate in bee brood and multiply themselves. They pump nymph and adult bee blood. The bee health is then deteriorated and cannot produce honey [2]. It is why monitoring infestation level is so important. There are classic ways of estimating and monitoring the infestation level. But they need human effort and are not accurate since they are based on sampling. Computer vision methods could be used to automate this type of tasks. In fact, artificial intelligence gives more and more interesting

[2] https://sciencepost.fr/australie-tue-abeilles-par-millions/

results, therefore computer vision approaches based on machine learning could help beekeepers to monitor varroa infestation level and more easily protect their beehives.

There is already research work that deal with using neural network to detect varroa on bee frame images and other works to detect bees alone. But I notice that there is no research done to propose one neural network for detecting bees and varroas at the same time. In fact, to have a more complete detection, both bees and varroas should be detected, it is helpful to evaluate infestation level more accurately and to correlate the varroa number and bee number.

In this chapter, I propose a new approach based on neural networks which aims to detect bees and segment varroas. This approach can be extended to tasks that need to detect objects inside others.

4 Approach

In this section I present the approach. I start by presenting the general architecture of the proposed approach in Subsection 6.4.1. Then, in Subsection 6.4.2, I demonstrate why consecutive detection was chosen over multi-class detection. In Subsection 6.4.3, I explain why I preferred segmentation over object detection. After that, in Subsection 6.4.4, I talk about the advantage of using one neural network instead of using two neural networks for nested object detection. In Subsection 6.4.5, I explain mechanism of feature extraction proposed to detect and segment nested objects. Finally, in Subsection 6.4.6, I explain methods proposed for training neural network for two different dataset.

4.1 General Architecture

The general architecture is based on Mask R-CNN. As it is illustrated in Figure 6.3, the FPN backbone generates five feature maps, these features are used by RPN which predicts proposals, then each proposal is classified as object/ no object by the box heads. The segmentation task is performed by the mask head. In the standard Mask R-CNN, segmentation concerns the same type of objects, in my case bees. As my objective is to segment varroas, the mask head is trained on the varroa regions instead of the bee regions. I chose this approach because of these arguments:

- Faster R-CNN is already a good neural network for bee detection.
- Mask R-CNN is a good state-of-art neural network for instance segmentation, therefore it can be used to detect bees and segment varroas at the same time.

4.2 Multi-class Detection vs Consecutive Detection

The most natural way to detect bees and varroas using the same neural network is to use a multi-class object detector, for example configuring Faster R-CNN to detect two types of objects. To train this neural network, bee frame images containing bee and varroa annotations should be provided to the learning process. But I cannot obtain these annotations. In fact the probability

Table 6.1: Accuracy results of Faster R-CNN detecting varroas from bee frame images and from bee images

Neural Network	BBox Recall @0.5	BBox Average Precision @0.5	Points Mean Average Precision
Faster R-CNN Varroa in Frame	76.92	71.68	80.8
Faster R-CNN Varroa in Bee	87.08	82.55	89.7
Relative Difference	+13.21%	+15.16%	+11.01%

of presence of varroa in a frame image of bees is very rare compared to the presence of bees. Therefore, annotating enough number of varroas for training requires annotating a very big number of bees, which is not possible due to the time needed by the annotation task. While detecting varroa from bees needs only images of infected bees and the corresponding varroa annotation.

Another important argument that pushes us to detect varroas as inner objects is the fact that detecting varroas from bee features is more accurate than detecting them from image frame features. Table 6.1 shows the average precision in both cases using Faster R-CNN neural network. This can be explained by the fact that the percentage of area occupied by a varroa in a frame image is lower than that occupied by a varroa in a bee.

Moreover, detecting objects of two different types and sizes by a neural network is more challenging than detecting objects of one type, because in the first case it needs to learn using the same set of layer parameters to extract bee and varroa features.

4.3 Segmentation vs Object Detection

Locating objects in an image consists of finding their positions. Each position can be coarse like the object center, or more accurate like the bounding box surrounding the object, or very accurate like the object pixels. Object detection consists of giving object bounding boxes, while segmentation is to find object pixels. The segmentation task needs only one branch that classifies each pixel. Object detection needs in general two branches, one to predict the presence of an object at each pixel and another to predict bounding box coordinates. Therefore since I have segmentation annotations, I chose to segment varroas instead of predicting their bounding boxes because the resulting neural network is lighter. Regarding point detection, it is based on segmentation; for each pixel, a score should be predicted corresponding to the probability that it matches an object center. For these reasons, I focused my research on the segmentation task.

4.4 Two Neural Networks vs One Neural Network

A straightforward way to detect bees and varroas on bees consecutively is to detect first bees using a neural network trained on bee detection, then detect varroas on bees using a second neural network trained on detecting varroas from bees. The main drawback of this approach is that it needs to have two different neural networks with different parameters. Thus, the

whole detection consumes more time and resources which are usually not available. In fact, after detecting bees, features are already extracted from the original image, these features can be used to detect varroas instead of using the original bee region.

4.5 Backbone Feature Extraction

Mask R-CNN contains a backbone which extracts features from original images. As it is illustrated by Figure 6.4, these features are used for generating proposals, predicting object bounding boxes and generating the segmentation mask. In the standard approach, generated masks correspond to predicted objects, in my case they represent inner objects (e.g., varroa on a bee). Concretely the backbone generates a set of feature maps. According to the used backbone and the configuration, one feature map is chosen for each bounding box. ROI pooling extracts the corresponding region from the feature map and forwards it to the head layers.

As illustrated by Figure 6.5, when FPN architecture is used, five feature maps "p2", "p3", "p4", "p5" and "p6" are generated by the backbone. These feature maps have decreasing resolutions. Proposals are generated from all these feature maps, according to the size of each bounding box, ROI pooling selects the feature map from which the extraction is performed. In standard Mask R-CNN, Box Head pooling and Mask Head pooling extract features from the same feature map for each bounding box proposal. But when the network is trained for segmenting another type of objects (e.g., varroa), learning features is more challenging, as the same features should contain common information of two different types of objects (e.g., bee and varroa). To overcome this constraint, I decide to select different features for segmenting varroas, this selection must respect the balance between these two constraints:

- Not basic features (like bee images), otherwise there is no benefit of using one neural network for nested detection.
- Not very advanced features, otherwise extracting common features is difficult.

On the other hand, as the varroa is a small object, feature maps of bigger resolution are more adequate for segmentation, so the first FPN output feature maps are more interesting but using them means increasing the number of common layers and making training more challenging. FPN has pyramidal symmetric architecture, "p2", "p3", "p4" and "p5" have the same sizes respectively as "res2", "res3", "res4" and "res5". The "res" features are the outputs of the stages of the standard ResNet neural network. After testing extracting features from different ResNet stages, I conclude that using "res4" for segmentation gives the best results.

To extract more information specific to varroas, I added a branch connected to the backbone. After extracting features from this backbone, the branch performs convolutions on input feature map, then the output is given to the Mask Head for varroa segmentation. Figure 6.6 shows the architecture of the whole configurable neural network, it has two parameters:

- Feature map to select from the backbone, it can be the input image, ResNet output features or FPN output features.
- The branch layers.

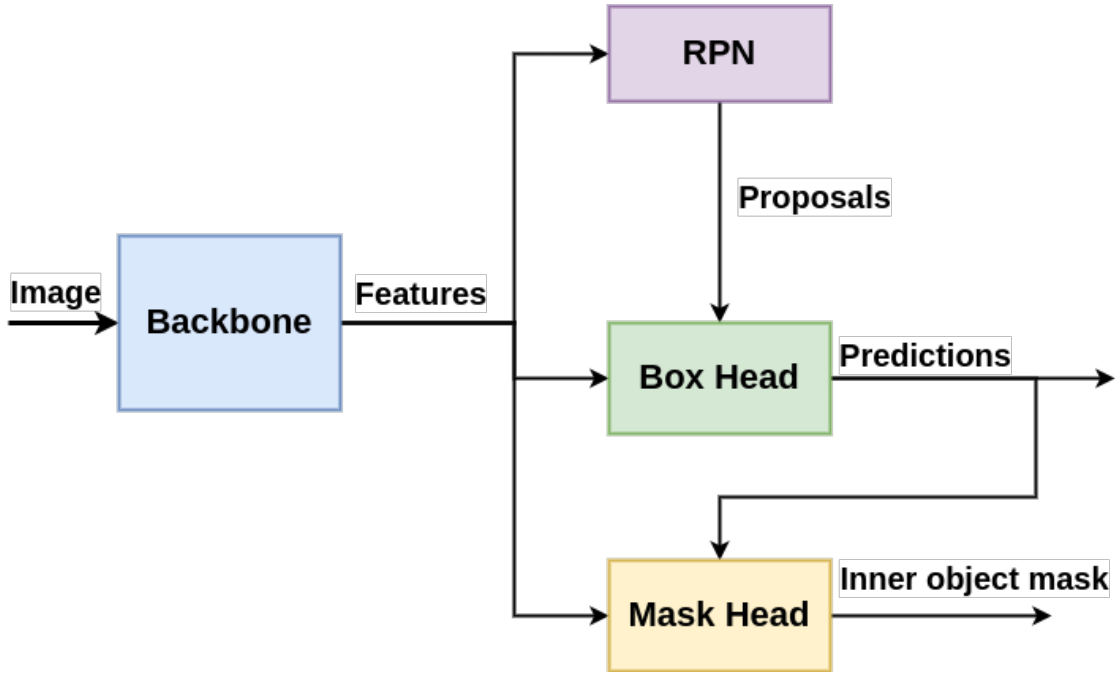


Figure 6.4: Diagram showing how the features generated by Mask R-CNN backbone are used for prediction

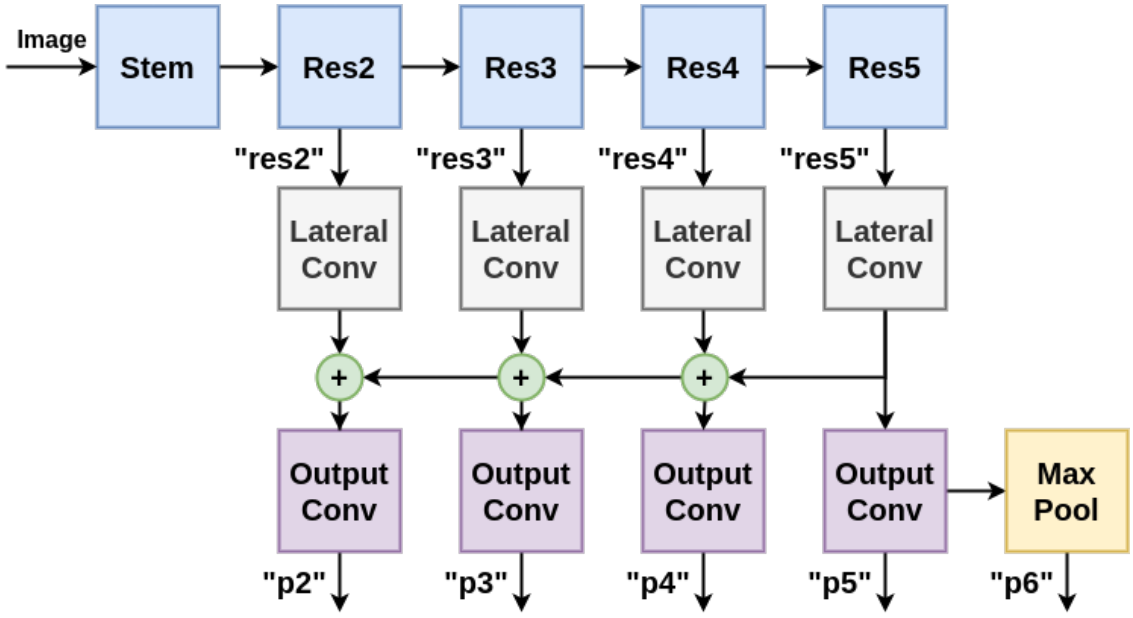


Figure 6.5: Feature Pyramidal Network

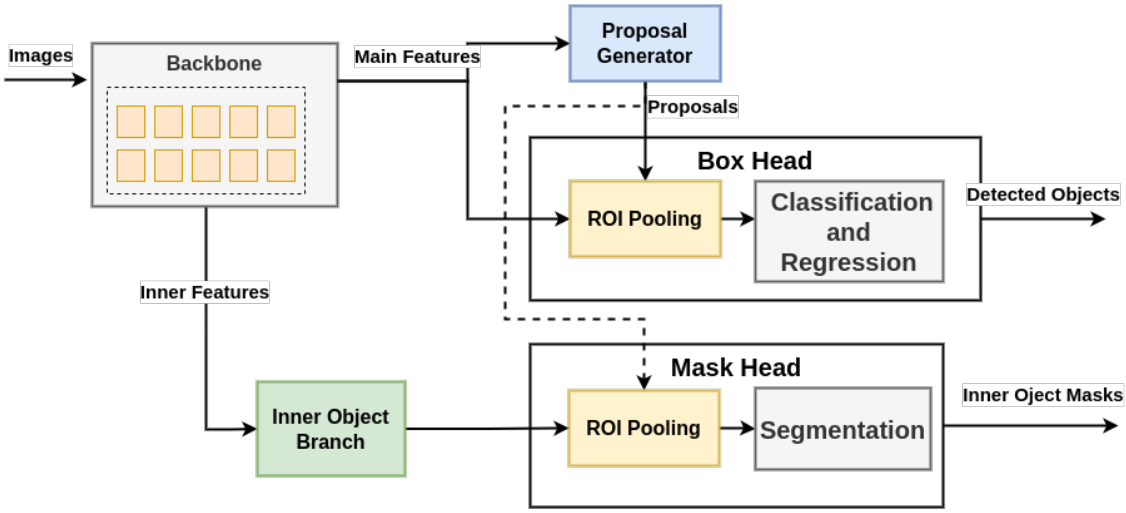


Figure 6.6: Architecture of Mask R-CNN with a branch for extracting features specific to varroa information

4.6 Training Mask R-CNN for Nested Object Detection

As explained in Subsection 6.4.2, I do not have one dataset containing bee annotations and varroa annotations, I have two separate datasets:

- Bee frame images each of which is annotated by bounding boxes corresponding to bees.
- Bee frame images each of which is annotated by bounding boxes corresponding to infected bees and masks and ellipses corresponding to varroas.

Standard training should be modified to consider two different types of data. I have proposed three ways of training.

(1) The first one takes as input two batches of data, bee frame images with bee annotations, and bee frame images with bee infected and varroa annotations. During the forward phase, two backbone features are calculated, one from each input data. Features corresponding to bee annotations are used to calculate RPN losses and Box Head losses, the one corresponding to varroa annotations are used to calculate Mask Head losses. ROI pooling of Mask Head extracts regions from feature maps corresponding to bee infected annotations. The mask loss takes as input varroa positions on infected bees. In other words, the neural network is trained to predict bees of the first dataset and to segment varroas in bees of the second dataset. Figure 6.7 shows this data flow when using two separate batches for training neural network.

(2) The second way of training is based on transfer learning, it consists of performing two separate training, the neural network is trained for varroa segmentation task, the output backbone parameters are saved and imported into the neural network to train it a second time for bee detection task.

(3) The third way of training is to merge the datasets, one backbone forward is performed, the resulting features are then separated to pass them to the next layers.

During inference, the neural network is forwarded normally; the first dataset are used to

evaluate bee prediction, the second one is used to evaluate varroa segmentation. Figure 6.8 shows the inference data flow for detecting bees and segmenting varroas.

Unless otherwise stated, the used trained method to test my approach under different parameters and conditions is the first one.

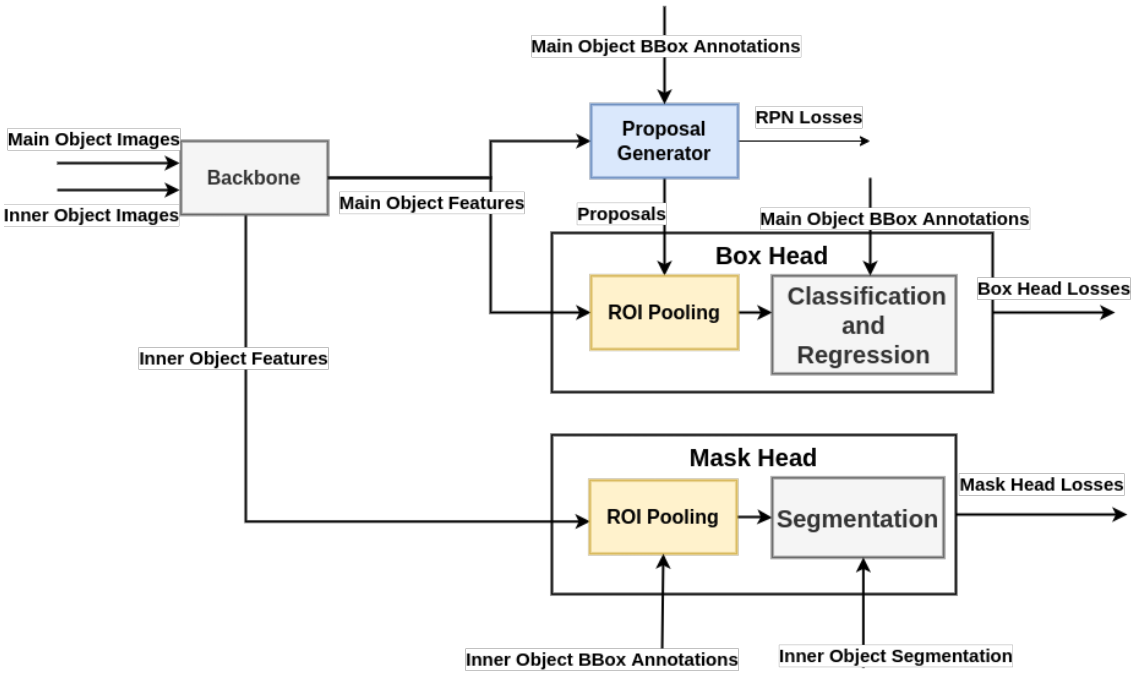


Figure 6.7: Training Mask R-CNN for bee detection and varroa segmentation using two batches from different dataset

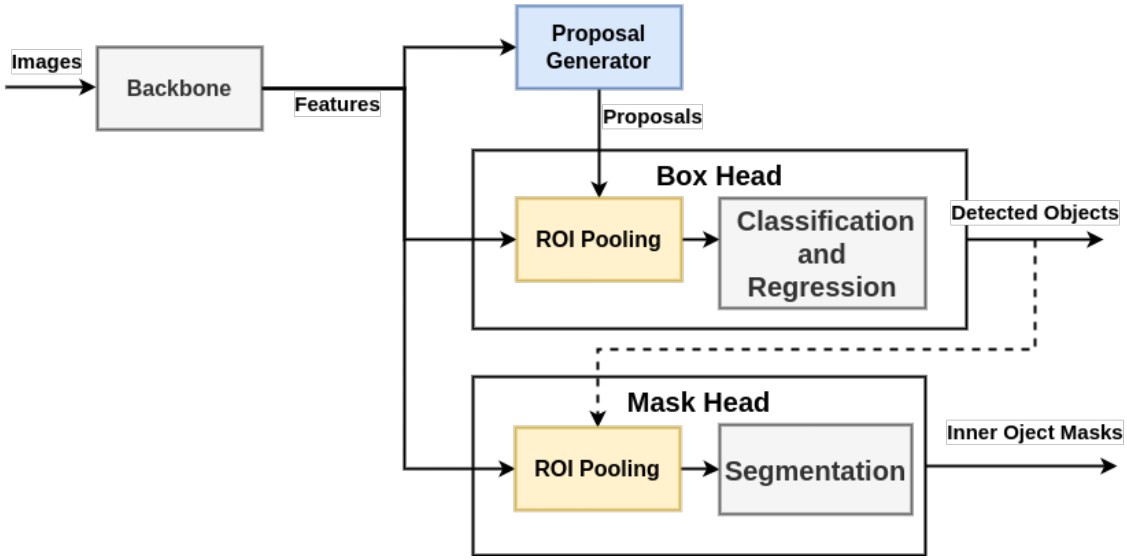


Figure 6.8: Mask R-CNN inference for detecting bees and segmenting varroas

5 Results and Discussion

In this section, I focus on presenting the accuracy results that I obtained by testing different parameters, methods and conditions on proposed approach compared to standard separate detection. The implementations of neural networks are based on Detectron 2, its extensibility and modularity has enabled us to modify standard components and add custom branches. I start the section by describing the dataset I used to evaluate and compare the neural networks in Subsection 6.5.1. Then, I present the used metrics in Subsection 6.5.2. After that, I compare the approach based on different neural network parameters in Subsection 6.5.3. In Subsection 6.5.4, I also compare the results when using different feature maps for extracting information for varroa segmentation. Next, in Subsection 6.5.5, I discuss how accuracy is influenced by using or not an additional layer for extracting varroa features before pooling. After that, I show the results according to mask loss factor in Subsection 6.5.6. Then, in Subsection 6.5.7, I show the results when transfer learning is used. Finally, in Subsection 6.5.8, I compare accuracy results in two cases: when inferring from bee proposals and when inferring from annotated infected bees.

5.1 Varroa Dataset

I aim at building a neural network which detects bees and varroas on bees. Therefore I must train and evaluate this approach on bee detection and varroa segmentation. Thus, I have split my bee frame set composed of 63 images annotated by bees into a train set of 45 images (2581 annotated bees) and a test set of 18 images (1282 annotated bees) to obtain a dataset dedicated to bee detection.

Regarding varroa detection, I have already a set of images composed of 254 images annotated by infected bees and varroas. To augment my dataset by negative examples, I have added images annotated by not infected bees. After filtration I obtained a set of 316 images annotated by infected/not infected bees and varroas. For each annotated bee, I have generated a binary mask

of contained varroas. Then, I have split the set into a train set of 260 images (978 bee masks) and a test set of 56 images (266 bee masks).

To compare varroa detection from bee images and varroa detection from bee frame images, I have built a dataset for each type of detection. The first one is composed of 254 bee frame images that I split into a train set of 200 images (475 annotated varroas) and a test set of 54 images (117 annotated varroas). The second one is composed of 1290 bee images (augmented by not infected bees) that I split into a train set of 979 images (460 annotated varroas) and a test set of 311 images (178 annotated varroas).

5.2 Metrics

To evaluate the performance of the proposed approaches, I used the following metrics: mean average precision for bounding boxes, mean average precision for keypoints and dice score. As explained earlier, the dataset containing varroa annotations are different from the one containing bee annotations, I evaluated the varroa detection/segmentation separately from bee detection. According to the used neural network, the output format can be one of three:

- A binary mask of the whole bee frame image, it corresponds to varroa positions.
- Instance masks, each one corresponds to varroa positions on a bee.
- Bounding boxes surrounding varroas.

5.2.1 Dice Score

As explained in Subsection 4.3.8, dice score metric evaluates the accuracy of the segmentation results. I used it to assess neural networks that segment varroas in order to compare their performances. The ground truth data are binary masks. A binary mask is a 2D matrix whose values are 0 or 1, 1 means that the corresponding pixel belongs to a varroa, 0 means the opposite. This mask can correspond either to a bee or the whole bee frame image depending on the used dataset.

Mask R-CNN generates instance masks. For each detected bee, a binary mask is predicted. To calculate dice score metric, the masks of bees of the same image are summed to obtain a varroa mask of the entire bee frame image. Then dice score is calculated between the result and the ground truth mask.

5.2.2 Mean Average Precision for Bounding Boxes

As explained in Subsection 4.3.7, mean average precision metric evaluates the accuracy of the bounding box results. I used it to compare the performance of the neural networks that detect varroas as objects. The ground truth data are bounding boxes, since the original constructed dataset contains masks, a transformation algorithm was developed to get bounding boxes from varroa masks. For neural networks that generate segmentation results, I used the same transformation to convert segmentation data to bounding boxes.

The transformation algorithm is as follows: the input results are the segmentation data; for each bee, a binary mask for each bee. A global mask corresponding to the whole image is calculated by summing masks, then using the "scipy" library, a connected component algorithm is performed on the global mask to get different regions, the boxes are obtained by acquiring the minimum and maximum coordinates of each region.

5.2.3 Mean Average Precision for Keypoints

Varroas are very small objects, they can be considered as points compared to the whole frame image. Furthermore, the objective of detecting varroas is to estimate the infestation level. Therefore, it is enough to simply find the varroa centers to achieve the objective. Thus, I calculate mean average precision on keypoint results. This metric is explained in Subsection 4.3.9. I notice that the keypoint results are higher than bounding boxes results, this can be explained by the fact that detecting varroa centers is less restrictive than detecting borders of bounding boxes surrounding varroas.

To obtain ground truth data composed of keypoint annotations, I simply calculate the centers of varroa regions. To obtain keypoints from segmentation results I get bounding boxes and then the centers are calculated.

5.3 Neural Network Parameters

ROI pooling takes as input the feature map of the bee frame image and the bounding box surrounding the bee. It projects the bounding box position on the feature map to find the features corresponding to the bee. ROI pooling summarizes the bee features into an output feature map of a fixed size. The output pooling resolution corresponds to the size of the feature map obtained after pooling the bee features. It is a parameter of the neural network.

Unlike bee regions, varroa regions are small compared to the feature map obtained after pooling. Therefore, according to the output pooling resolution, varroa information could be lost while extracting feature maps. To evaluate the influence of this parameter on the results, I launched training under two different pooling resolutions 14x14 and 28x28. The results are depicted in the two first lines of Table 6.2. The accuracy increases when using resolution of 28x28 instead of the standard value which is 14x14. Using higher resolutions than 28x28 does not give better results since the training would be more difficult due to the increase of neural network parameters. Another important reason why the output resolution should be limited is the size of varroas, since their corresponding regions are very small compared to the input feature map, the mask loss tends to be smaller when the resolution is bigger, very small losses could make the training difficult.

Another parameters that I tested is the number of convolutions in Mask Head. These convolutions extract information from the bee feature map, in the standard Mask R-CNN there are 4 convolutions. I launched the training under three different parameters: 2, 4 and 8. The results are presented in Table 6.2. The best parameter value is 4; few convolutions do not extract enough information, many convolutions increase the number of parameters which makes

Table 6.2: Accuracy of Faster R-CNN under different pooling resolution and number of convolutions after pooling.

Parameters	Dice Score	Points Mean Average Precision	Points Average Precision @0.5	Points Average Recall @0.5
Pooling Res: 14x14 Conv Number: 4	42	52.6	53	63.6
Pooling Res: 28x28 Conv Number: 4	50.14	60.6	60.8	74.8
Pooling Res: 28x28 Conv Number: 2	48.27	57.9	58.1	78.5
Pooling Res: 28x28 Conv Number: 8	40.61	57.8	57.9	80.4

the training more difficult.

5.4 Feature Map Extraction

My objective is to detect bees and varroas from bee features by using only one neural network. The backbone generated feature maps are used for generating proposals, classifying bees and segmenting varroas. The bee and varroa objects are not the same, my challenge is to find the common backbone features that can be used for both bee detection and varroa segmentation. My developed architecture allows to choose the features that are used as ROI pooling input for varroa segmentation. To find the best features, I tested neural network training under different extracted features:

- The original input image
- The output of Res3 stage
- The output of Res4 stage

The extracted features were directly forwarded to ROI pooling (I called this a "basic branch"). The results are compared to the standard approaches: Faster R-CNN trained only on bee detection and Mask R-CNN trained only on varroa segmentation. The results are presented in Table 6.3. I have these important remarks:

- Bee detection is stable in all cases, it means that training on varroa segmentation does not influence the bee detection.
- Varroa results become more interesting when the extraction is performed from advanced information.
- The best result that I reached is obtained when extracting from Res4. Extraction from Res5 gives lower accuracy. This may be explained by the fact that Res5 output features does not contain enough details for varroa segmentation.

Table 6.3: Accuracy of Faster R-CNN detecting bees, Mask R-CNN applied only on varroa segmentation, compared to Mask R-CNN using basic branch and different types of extraction

Neural Network	Points Mean Average Precision on Varroas	BBox Mean Average Precision on Bees
Faster R-CNN trained on bee detection	-	53.3
Mask R-CNN trained on varroa segmentation	83.8	-
Mask R-CNN Basic Branch Image Extraction	24.1	54.2
Mask R-CNN Basic Branch Res3 Extraction	45.1	54.2
Mask R-CNN Basic Branch Res4 Extraction	63.7	54.2

Even when using Res4 features, the accuracy is still lower than using a neural network trained only on varroa segmentation. The varroa segmentation accuracy limit can be explained by either (1) the training does not find the neural network parameters that can be used for detecting bees and segmenting varroas at the same time (2) or the bee detection accuracy influences the varroa segmentation accuracy. The second hypothesis is confirmed by the results presented in Subsection 6.5.8.

5.5 Branch Influence

The feature map extracted from the backbone could not contain high semantic information for segmenting varroas. I have tested adding convolution layers before pooling. Table 6.4 compares the accuracy result of forwarding directly the feature to ROI pooling (Basic Branch) and performing a convolution of kernel 3x3 and preserving the number of channels before pooling (Conv Branch). I notice that the accuracy is more interesting in case of not using the convolution. This can be explained by the fact that the training become more difficult due to the increase of neural network parameters.

5.6 Mask Loss

To train the whole neural network for bee detection and varroa segmentation, five losses are used:

- Two losses for generating the proposals: one for objectness, another one for offset coordinates (for more details, see Subsubsection 3.4.5.5)
- Two losses for object prediction: one for classification, another one for offset coordinates (for more details, see Subsubsection 3.4.5.5)

Table 6.4: Accuracy of Mask R-CNN applied on varroa segmentation when using a basic branch and using a Conv branch

Neural Network	Points Mean Average Precision on Varroas	BBox Mean Average Precision on Bees
Mask R-CNN Basic Branch Res4 Extraction loss × 8	67.9	54.3
Mask R-CNN Conv Branch Res4 Extraction loss × 8	65.9	52.5

Table 6.5: Table comparing results of Mask R-CNN applied on varroa segmentation when using standard mask loss and multiplying the loss by x8

Neural Network	Points Mean Average Precision on Varroas	BBox Mean Average Precision on Bees
Mask R-CNN Basic Branch Res4 Extraction	63.7	54.2
Mask R-CNN Basic Branch Res4 Extraction loss × 8	67.9	54.3

- One loss for segmentation task: the used function is the binary cross entropy

Since the varroa region is small, ground truth positive pixels are fewer compared to ground truth negative pixels. This imbalance generates a small loss. Compared to the other four losses, it could be neglected. To overcome this issue, I multiplied the loss by a factor, and compared the results. Table 6.5 shows that multiplying the loss gives better results for varroa segmentation. I have tested two factors x8 and x16, they give almost the same result.

5.7 Transfer Learning

Another approach for detecting objects of two different types using a same neural network is to perform transfer learning. It consists in transferring a learning capacity that some neural network layers have previously acquired from a training to another neural network sharing the same layers. For example, if I have a neural network that has been trained to detect objects of a defined type, its parameters could be imported to another neural network that has common layers, and a new training to detect objects of another type could be launched. The obtained neural network would be able to detect two types of objects.

To accomplish transfer learning in my case, I launched two training each of which corresponds to an object type. The first one is dedicated for only segmenting varroas by extracting features from Res4 since the Res4 features are important for segmenting varroas (Table 6.3). Then,

Table 6.6: Accuracy of Faster R-CNN detecting bees, Mask R-CNN applied only on varroa segmentation, compared to Mask R-CNN trained consecutively on varroa segmentation then on bee detection after freezing backbone branches before Res4 included

Neural Network	Points Mean Average Precision on Varroas	BBox Mean Average Precision on Bees
Faster R-CNN trained on bee detection	-	53.3
Mask R-CNN trained on varroa segmentation	83.8	-
Mask R-CNN Transfer learning Freezing Branch Res4	64.7	46.1

Table 6.7: Accuracy of Faster R-CNN detecting bees, Mask R-CNN applied only on varroa segmentation, compared to Mask R-CNN trained on bee detection and using Res4 and Res3 whose parameters are the results of training on varroa segmentation extracting Res4 features

Neural Network	Points Mean Average Precision on Varroas	BBox Mean Average Precision on Bees
Faster RCNN trained on bee detection	-	53.3
Mask RCNN trained on varroa segmentation	83.8	-
Mask RCNN Transfer learning Res3 +4 Branch	63.7	53.3

I freeze the Res4 layer of ResNet neural network and all the previous stages (especially Res3 because stem and Res2 are freezed by default), and perform a second training dedicated for bee detection. The results are presented in Table 6.6. The varroa segmentation precision is similar to the one obtained when performing a separate batch training (Table 6.3). But the bee accuracy is reduced. The reason of this decrease is because Res3 and Res4 layers are frozen during the second training, their parameters cannot be modified to extract specific bee information.

To avoid this decrease, I consider a Mask R-CNN with a branch dedicated for varroa segmentation. The branch is constructed exactly by the same layers as the concatenation of Res3 and Res4 stages. I trained a Faster R-CNN neural network on bee detection alone. I perform another training for varroa segmentation alone by extracting from Res4. The parameters obtained from the first training are imported into Mask R-CNN structure, the parameters of Res3, Res4 and Mask Head are respectively imported into Mask R-CNN branch sub components Res3, Res4 and the Mask Head. The results are shown in Table 6.7. I notice that the bee precision is increased.

5.8 Inner Object Accuracy

To evaluate varroa segmentation accuracy by the final neural network that performs the detection of objects of both types (bees and varroas), I have two ways:

- Evaluating varroa detection on bee frame images after generating bee proposals from them.
- Evaluating directly varroa segmentation using annotated infected bees.

The first method leads to a connection between varroa accuracy and bee accuracy: varroa detection accuracy could be limited by bee detection accuracy. The second way is more representative of neural network capability of detecting inner objects. Table 6.8 compares accuracy of standard approaches, approaches based on separate batch training, and transfer learning approaches according to three metrics:

- Point mean average precision on varroas (generated proposals): the inference was performed on bee proposals generated by RPN.
- Point mean average precision on varroas (bee infected annotations): the inference was performed on bee infected annotations.
- BBox mean average precision on bees: it corresponds to bee detection accuracy.

From this table I have two important remarks: (1) The precision of varroa detection depends on the type of used bee bounding boxes, the result is always more interesting in case the inference is performed from ground truth annotations instead of predicted bees. This can be explained by the fact that the generated proposals are less accurate than the ground truth annotations; (2) The last line corresponds to the best approach as it has the advantage of segmenting varroas from bees with the best accuracy and detecting bees with an accuracy near to the best one.

Figure 6.9 shows examples of varroa segmentation performed by Mask R-CNN neural network trained only on varroas and extracting features from Res4 output, the inference is performed on ground truth bee infected annotations.

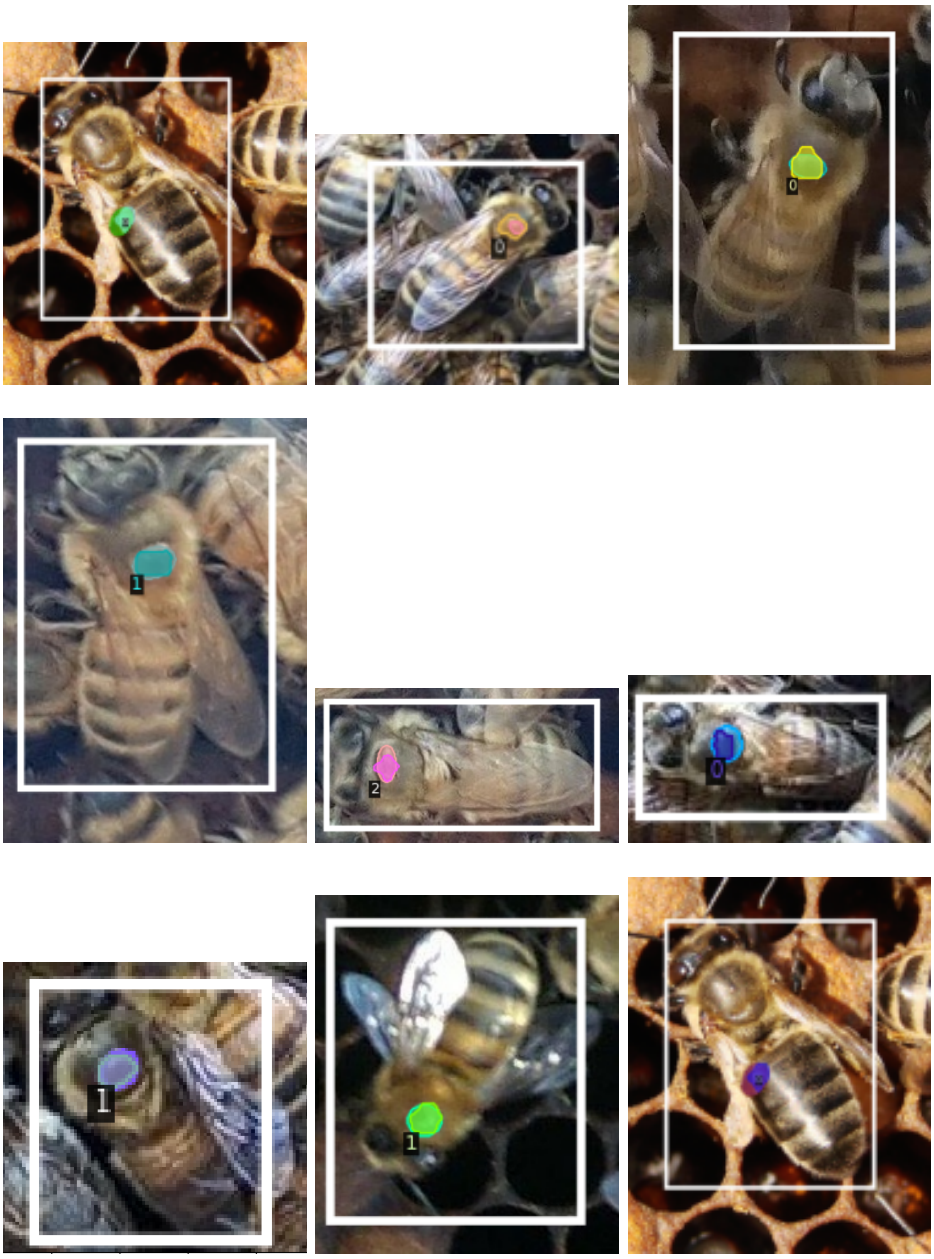


Figure 6.9: Examples of varroa segmentation performed by Mask R-CNN trained only on varroas and extracting features from Res4. Each image contains infected bee bounding box, ground truth varroa region, and predicted varroa region colored by another color. The numbers are the identifiers of bee masks corresponding to predicted varroa regions. Predicted regions can be distinguished through their colors which correspond to identifier colors.

Table 6.8: Accuracy of Faster R-CNN detecting bees, Mask R-CNN applied only on varroa segmentation, compared to different approaches of Mask R-CNN trained on bee detection and varroa segmentation evaluating under two conditions: inference from proposals, inference from annotated infected bees.

Neural Network	Points MAP on Varroas (generated proposals)	Points MAP on Varroas (bee infected annotations)	BBox MAP on Bees
Faster R-CNN trained on bee detection	-	-	53.3
Mask R-CNN trained on varroa segmentation	-	83.8	-
Mask R-CNN Basic Branch Res4 Extraction	63.7	80.5	54.2
Mask R-CNN Transfer learning Freezing Branch Res4	64.7	85.7	46.1
Mask R-CNN Transfer learning Res3 +4 Branch	63.7	85.7	53.3

6 Conclusion

Nested object detection is an important common task in computer vision domain. It consists in detecting two types of objects, one inside another one. My objective is to detect bees and varroas using one neural network. Varroas are parasite that are generally located on top of bee backs. Unlike other use cases, varroas and bees have very different shape and occurrence probabilities in bee frame images. Thus, I could not have one annotated dataset containing bees and varroas data. I built different dataset for each object type. To handle nested object detection in these conditions, I propose to use Mask R-CNN architecture which is initially proposed as an extension of Faster R-CNN to perform object detection and instance segmentation of the same object type. I modified this approach to segment the inner object instead of the detected object. The RPN generates bee proposals, they are then classified to obtain predicted objects, then varroa region pixels are predicted from bees. The most important points that I have dealt with are: training one neural network on two different dataset to detect nested objects, and extracting relevant features for inner object segmentation. Regarding training constraint, I proposed two different approaches; transfer learning where the same neural network is trained consecutively on the two datasets, and separate batch training where two different batches are forwarded by the backbone during each training iteration. Mask R-CNN network is composed of a backbone feature extractor, my proposed approach extracts Res4 output to perform segmentation. Finally I obtained an approach that have similar accuracy performance as the standard approach: two

different neural networks trained separately. Therefore, I have reached my goal to have one neural network that performs detection of bees and varroas by consuming less resources and time during inference compared to standard approach.

Chapter 7

Results

Résumé

Dans ma thèse, j'ai présenté deux principales contributions: l'approche basée sur les coins pour l'amélioration de la détection des objets dans les scènes denses, et une approche efficace pour la détection des objets imbriqués.
Comme cité précédemment, les caractéristiques de la couche "Res4" de Mask R-CNN sont pertinentes pour la détection des objets imbriqués. En plus des tests de précision réalisés pour confirmer ce résultat, j'ai effectué des expérimentations et affiché des ensembles de caractéristiques, illustrant le calcul des différences entre les cartes d'abeilles et celles de varroas et leur permutation. J'ai aussi comparé l'extraction standard FPN et celle basée sur "Res4".
Une question importante concernant la détection des objets imbriqués est la différence entre la détection des objets inclus (varroas) à partir d'une image et leur détection à partir des objets englobants (abeilles). Pour cela, j'ai réalisé un test pour valider que la détection des varroas à partir des images d'abeilles donne de meilleurs résultats que la détection à partir des images de cadre d'abeilles.
Afin de valider la pertinence de la méthode basée sur les coins pour la détection des objets partiellement cachés dans les scènes denses, j'ai testé l'approche sur un sous-ensemble de la base d'images publique Crowd Human qui correspond à des scènes de foules de personnes. Les résultats montrent bien que l'approche des coins proposée est intéressante aussi pour la détection d'objets de nature différente.
Afin de valoriser les approches proposées, et dans le cadre du projet PNAPI, j'ai participé au développement d'une application web qui fera partie de la plateforme d'aide aux apiculteurs. Cette application permet notamment la saisie des données concernant le comptage des varroas. Un travail additionnel doit être réalisé pour intégrer le système de détection des abeilles et des varroas à la plateforme afin d'automatiser la tâche de comptage.
L'objectif de ma recherche est entre autres d'aider les apiculteurs dans leurs tâches de traitement de varroas. Pour cela, le système doit fournir le niveau d'infestation d'une ruche à partir des images prises par l'apiculteur. Pour atteindre cet objectif, un travail additionnel de recherche doit être réalisé. Néanmoins, j'ai calculé le pourcentage d'infestation à partir des abeilles et des varroas détectés par le réseau de neurones et comparé le résultat aux approches standards. J'ai constaté que la précision de l'estimation du pourcentage d'infestation est améliorée.

Résumé

J'ai par ailleurs aussi analysé la consommation de ressources de l'approche proposée pour la détection des objets imbriqués. Mon approche utilise moins de ressources en termes de temps d'exécution, de mémoire, du nombre de paramètres du réseau et du nombre d'opérations réalisées par rapport à la détection consécutive des objets.

Contents

1 Introduction

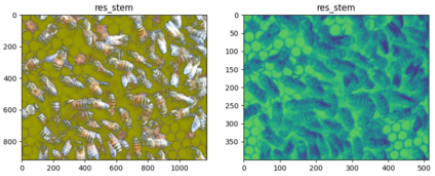
In my thesis, I have proposed two main contributions: corner approach which detects objects from corners to improve detection in dense scenes, and one neural network to perform detection of nested objects. I based my suggestions on hypothesis that I supported by results. The main hypothesis that I defend are: (1) detecting objects from corners can improve detection of partially hidden objects, (2) detecting varroa from bee images is more accurate than detecting from bee frame images, (3) Res4 features are relevant for varroa segmentation, (4) separate batch training and transfer learning are better than merged batch training. To validate my hypothesis I performed some further studies and experiments. In this chapter I present this work and the obtained results. In Section 7.2, I talk about some experiments carried out to check the best features for nested object detection. In Section 7.3, I present accuracy results obtained when merging batches of different dataset is applied during training to detect nested objects. Then, in Section 7.4, I explain a method based on U-Net architecture and its segmentation results when applied on bee frame images and bee images. In Section 7.5, I explain another method of representation and detection of varroa objects based on density estimation. After that, in Section 7.6, I explain the experiments realized to validate my corner approach on a public set of images containing humans. In Section 7.8, I qualitatively compare detecting corners as objects to my corner approach. In Section 7.10, I present some results concerning infestation level estimation. In Section 7.11, I compare the approaches proposed for detecting nested objects and the standard approach in terms of computation. Finally, in Section 7.9, I present the PNAPI global platform that will host my object detector.

2 Choosing Common Features for Nested Object Detection

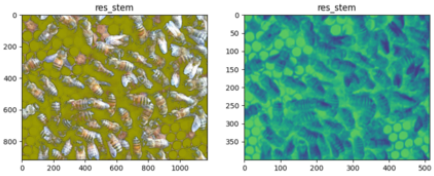
In previous chapter, I proposed a neural network structure to perform bee detection and varroa segmentation at the same time. It is based on Mask R-CNN, but instead of segmenting bees, the network is trained on segmenting varroas. The main modification that I introduced is the feature map selection for inner object segmentation. An intermediary feature map from ResNet output stages is chosen and forwarded to layers responsible for predicting varroas. This choice must respect two constraints: (1) the feature map must contain good semantic information to predict varroas, and (2) the shared layers must be able to extract common features of bees and varroas. In this section, I present some experiments that I carried out to find the best feature map to select.

2.1 Displaying Feature Maps

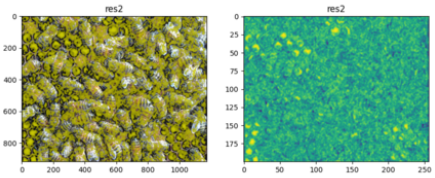
A preliminary analysis of the problem is to display feature maps and make hypothesis based on qualitative estimation of extracted information. Thus I visually compared the output feature maps of backbone ResNet stages in two cases: training on bee detection only and training on varroa segmentation only. Figure 7.1 shows the output feature map of stages "stem", "res2", "res3", "res4" and "res5" obtained by forwarding an image example into a neural network trained



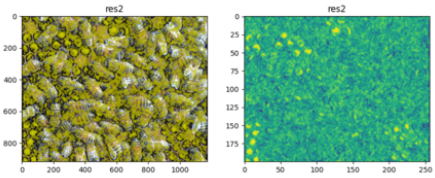
(a) Stem output - bee



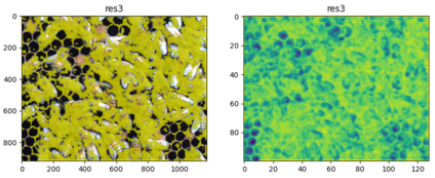
(b) Stem output - varroa



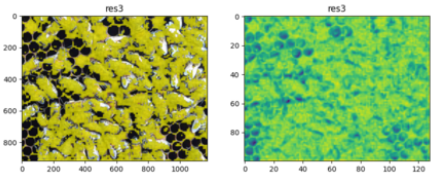
(c) Res2 output - bee



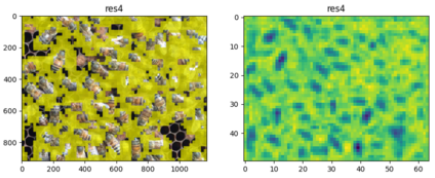
(d) Res2 output - varroa



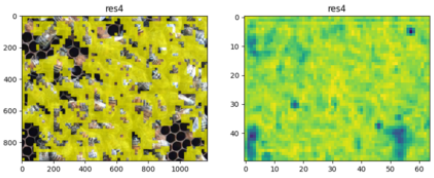
(e) Res3 output - bee



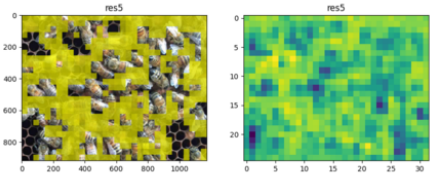
(f) Res3 output - varroa



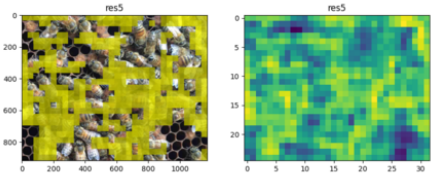
(g) Res4 output - bee



(h) Res4 output - varroa



(i) Res5 output - bee



(j) Res5 output - varroa

Figure 7.1: Backbone output feature maps trained on bees and varroas.

Table 7.1: Means of euclidean distances between feature maps generated by bee detector and ones generated by varroa segmentor on 56 image examples (the results are averaged)

Layer	Mean distance
Res stem	0
Res2	0
Res3	0.51
Res4	0.94
Res5	0.79
Fpn2	1.08
Fpn3	1.17
Fpn4	1.21
Fpn5	1.28
Fpn6	1.27

on bees and another one trained on varroa segmentation. A feature map contains three dimensional data (height, width, channels), to obtain displayable data, I calculate the mean of channel values of each point.

The outputs of stem and Res2 are the same for both cases. In fact, the corresponding layer parameters are frozen during training, the initial values are the results of a training performed on a big dataset for object detection. I can also notice that the two Res3 outputs are quite similar, the Res3 extracts similar information from bees and varroas. The divergence starts from layer Res4, the outputs are quite different, they correspond to bee and varroa positions. The Res5 outputs are also different.

From this analysis, I can conclude that the stage Res3 preserves common information. Therefore, for detecting bees and varroas at the same time, Res3 stage could be shared by the two tasks. I notice that Res4 output features contain high semantic information in both cases. Thus, extracting these features for varroa segmentation could be sufficient, Res3 features seem not very relevant for this task.

2.2 Difference between Feature Maps

To investigate more the common features that can be used for detecting nested objects, I calculate the means of euclidean distances between feature maps generated by bee detector and ones generated by varroa segmentor on 56 image examples (the results are averaged). Table 7.1 shows the obtained results. It contains also values concerning FPN layers. ResNet differences are clearly lower than FPN differences, because FPN layers extract more specific information. In fact, in standard Faster R-CNN and Mask R-CNN, features are extracted from FPN layer outputs. Thus, they contain very specific information. It is why sharing these layers to perform bee detection and varroa segmentation using one neural network is more challenging.

Table 7.2: Standard detection results using Faster R-CNN trained on bee detection only and Mask R-CNN trained on varroas segmentation only, compared to the obtained neural networks modified: each line correspond to the corresponding obtained neural network whose some layers have been swapped with their equivalent in the other neural network.

	Bee Mean Average Precision BBox	Varroa Mean Average Precision Points
Standard Detection (Faster R-CNN / Mask R-CNN)	61.8	83.8
Backbone swapped	0	1.1
ResNet swapped	0.3	18.8
Res3 + Res4 swapped	7.3	8.8
Res3 swapped	51.1	29.5
Res3.0 + Res3.1 swapped	57.7	79.7

2.3 Swapping Layers

Another way to find the common information between detecting bees and detecting varroas is to swap layers. I extract some layers of neural network trained on bee detection and replace them inside the neural network trained on varroa detection and vice versa. I have performed five hierarchical swapping:

- The whole backbone (ResNet + FPN)
- ResNet alone
- The two ResNet stages: Res3 and Res4
- Res3 alone
- Res3.0 and Res3.1 which are consecutive blocks of the stage Res3. Each block is composed of convolutions.

At each extraction, the accuracy of the detection is calculated and compared to the standard approaches. Table 7.2 shows the results. I notice that the accuracy improves when the extraction concerns smaller neural network parts. I notice also that the significant increase in accuracy starts when swapping Res3 layer of bee detector with its equivalent layer in varroa detector and vice versa. This remark confirms that the common information can be preserved by Res3 layer.

2.4 Using Standard Feature Map Selection

My approach is based on Mask R-CNN and FPN. The standard method to select the feature map for proposal classification and instance segmentation is to calculate the area of the proposal generated by RPN and choose a feature map according to that size; bigger feature maps are used for smaller objects and smaller feature maps are used for bigger objects (for more details see Subsubsection 3.4.5.4). I have tested this way of selecting feature map for varroa segmentation.

Table 7.3: Varroa accuracy of Mask R-CNN trained on two separate batches and extracting Res4 features compared to Mask R-CNN trained on two separate batches and extracting all FPN features from backbone.

Neural Network	Points Mean Average Precision	Points Average Precision @0.5	Points Average Recall @0.5
Mask R-CNN Separate Batches Res4 Extraction	63.7	63.7	84.1
Mask R-CNN Separate Batches Extracting all feature maps	52.6	53	63.6

The result is depicted by Table 7.3 which shows the comparison of accuracy between using Res4 extraction and standard feature map selection. I notice that extracting from a fixed feature map as Res4 gives better results.

3 Fusion Batch Training

To train one neural network for two tasks in parallel, I proposed a method based on forwarding one batch composed of two types of images:

- The first set contains images coming from bee detection dataset, each image comes along with bee bounding box annotations.
- The second set contains images coming from varroa detection dataset, each image comes along with bounding box bee infected and not infected annotations, bee infected annotations comes along with ground truth varroa masks.

The backbone forwards the batch. It outputs feature maps corresponding to the two type of data. The batch is then divided into two sub-batches, each one contains data of specific task:

- The batch corresponding to bee annotations is forwarded to RPN and Box Head for proposal generation and object classification.
- The batch corresponding to varroa annotations is forwarded to Mask Head for varroa segmentation.

Table 7.4 shows the result of that training compared to standard approaches. I notice that the accuracy is limited compared to other methods proposed in Subsection 6.4.6, it can be explained by the fact that merging two batches in one reduces the number of examples used for training. Increasing the batch size is not possible due to memory limit.

Table 7.4: Table comparing results of Mask R-CNN applied on varroa segmentation when using fusion batch training and separate batch training

Neural Network	Points Mean Average Precision on Varroas	BBox Mean Average Precision on Bees
Mask RCNN Res4 Extraction separate batches	63.7	54.2
Mask RCNN Res4 Extraction merge batches	49.1	54.1

4 U-Net Segmentation

Among the main neural network architectures used for semantic segmentation, there is U-Net neural network (Subsection 3.4.3). It has a symmetric pyramidal structure; a first phase where the image is downscaled. This phase enables feature extraction from the image. The second phase increases the feature map size, to achieve segmentation. I have tested this architecture to segment varroas from bee frame images and bee images. The used implementation is based on MobileNet neural network. It is a version of ResNet neural network that was proposed for image classification. Thus, it contains only layers that decrease the size of the image to extract features. I have added layers responsible for increasing the feature map sizes to obtain segmentation (for more details see [Tensorflow, 2022]). Table 7.5 shows the dice score metric result of varroa segmentation using U-Net neural network based on MobileNet from bee frame images (each one contains many bees) and from bee infected images (each one contains one bee). The results confirm that detecting varroas from bee images gives better result than detecting from bee frame images.

Table 7.5: Dice score of U-Net detecting varroas from bee frame images and bee images. The used implementation is MobileNet.

Neural Network	Dice Score
U-NET Mobile Net Implementation Varroa in frame	38.1
U-NET Mobile Net Implementation Varroa in bee	53.57

5 Density Estimation

Object segmentation requires predicting the pixels that correspond to the object in the input image. In my case, I am more interested in detecting varroa center positions as my objective is counting varroa number present on bee frame. Some approaches dealing with this problem were published, for example [Xie, 2018] proposes a deep learning method to count humans. It

generates a density distribution for each human head: it uses a Gaussian filter to create a 2D gaussian distribution surrounding head centers. The mean and maximum of this distribution is localized at head center. The neural network is trained to generate a density map that matches these gaussian distributions. The counting is performed by integrating the output.

As I have more specific varroa annotations, I can generate more interesting density maps. Each varroa is annotated by an ellipse. I used the formulas explained in [Do, 2008], to obtain a two-dimensional Gaussian distributions from ellipses. Figure 7.2 shows an example of generated density distribution, yellow intensity corresponds to the distribution value at each position.

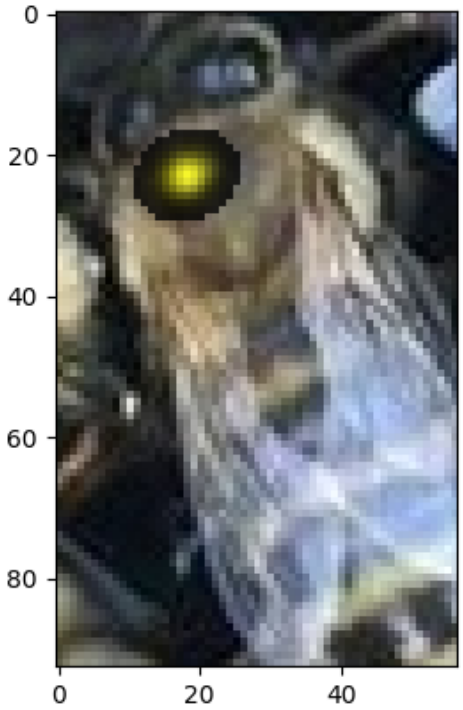


Figure 7.2: Example of generated density map from varroa ellipse annotation

After generating densities, I modified standard Mask R-CNN to handle densities instead of masks, the main modification that I implemented is related to annotation pre-processing: updating density distributions to match image transformations while training. Regarding inference, I used the standard implementation. In fact, to generate masks, probability scores are generated by the network, then a thresholding (default value is 0.5) is performed to obtain final regions corresponding to varroas. The network trained on density distribution estimation generates also probability values. After getting the mask, I retrieve keypoints in the same way as explained in Subsection 6.5.2. Table 7.6 shows the result accuracy, the features used for varroa segmentation are Res4 output features. I notice that the results are almost similar. Density estimation could therefore be an interesting method to handle point detection or counting tasks for nested object detection.

Table 7.6: Accuracy results of Mask R-CNN trained on varroa mask prediction by extracting Res4 features, compared to Mask R-CNN trained on varroa density prediction by extracting Res4 features

Neural Network	Varroa Mean Average Precision Points
Mask R-CNN mask prediction	80.5
Mask R-CNN density prediction	78.8

6 Detection of Humans in Crowd

The corner proposed approach aims to detect objects in dense scenes. To verify the usability of this approach, I have tested it on images containing humans. I start by presenting the conditions of my experiments in Subsection 7.6.1. Then, in Subsection 7.6.2, I explain the corner approach used for training and evaluating accuracy regarding human detection. In Subsection 7.6.3, I show the metrics used for evaluation. Finally, I present accuracy results in Subsection 7.7.

6.1 Experiment Conditions

6.1.1 CrowdHuman Dataset

Among the state-of-art dataset for human detection in dense scenes, there is CrowdHuman dataset [1]. It contains a training set of 15000 images, a validation set of 4370 photos and a testing set of 5000 photos not annotated. Each image is annotated (except for the testing set) with 3 types of bounding box for each person in the photo. One bounding box contains the whole person, another contains only the visible part and the last one contains the head. An example of those three annotation bounding boxes is represented on Figure 7.3. In my case I am concerned by whole person annotation since my objective is to detect bounding box surrounding whole object even when it is partially hidden.

6.1.2 Dataset Construction

The CrowdHuman dataset is available on the Internet for downloading. The format of annotations is different from Detectron required format, a python code was developed to perform the conversion.

For first verification, I created a limited subset of filtered images. The filtration criteria is to take only images whose annotations do not exceed image edges. This choice was made because the Detectron library clips automatically all bounding boxes to fit the image, which is not good for training as my objective is to detect the bounding box surrounding the whole object even when it is outside the image frame. The first dataset that I obtained is as following:

- Training set: 1035 images. The images were taken randomly regardless of their densities.

[1] https://www.crowdhuman.org/

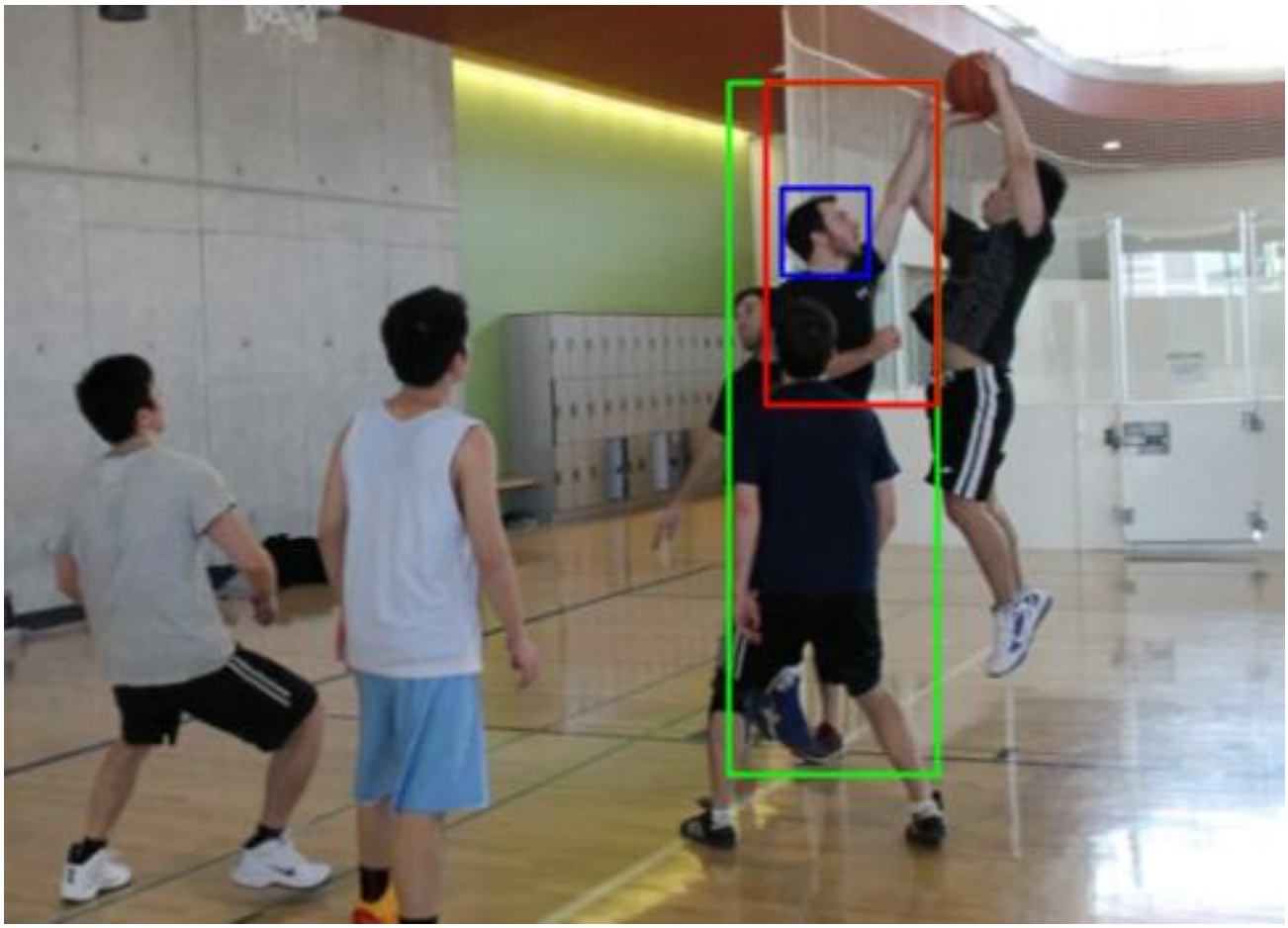


Figure 7.3: Types of annotations for one person in CrowdHuman. Green box contains the whole person, red box contains the visible part and blue box contains its head.

- Testing set of 4370 images divided in three subsets corresponding to three different object densities (Subsection 7.6.3).
  - High density: 357 images
  - Medium density: 998 images
  - Low density: 3015 images

To perform more interesting evaluation, I built a dataset consisting of images that were not filtered. To handle these data, I extended Detectron library to avoid clipping bounding boxes. Due to limited resources, the training set is composed of 1000 random images, the testing set is the same as the first dataset, I have used also a mini set of 193 random images for evaluation.

6.2 Tested Approach

I tested my corner approach on CrowdHuman dataset to assess its capability to detect objects in dense scenes. The approach was used as it is on the first dataset. Whereas, concerning the second dataset, I have encountered limits in resources. In fact, some images of the second dataset are large, and training was not possible due to GPU memory limit. To overcome this issue, I modified corner approach: instead of having four branches to handle corner data, I used three branches: bottom left, bottom right and top center. This choice was made by taking into consideration human body shape, the bottom branches would detect legs and top center branch would detect the head.

6.3 Metrics

In the state of the art, density definition is not discussed a lot, images are often classified manually as dense or not dense. Thus, I propose an interesting way of quantifying object occlusion and density in an image.

6.3.1 Density Score

IoU (Intersection over Union) quantifies well the occlusion of objects between each others. The idea is first to build a matrix containing the IoU between all pairs of bounding boxes. Let A be this squared matrix of size $n \times n$, where n is the number of bounding boxes. For i, j in $[\![1, n]\!]$, $IoU(B_i, B_j)$ is the IoU of boxes B_i and B_j. Notice that A is symmetric and that all its terms on the diagonal are equal to 1, since the IoU of one box with itself is 1.

$$A = \begin{bmatrix} 1 & IoU(B_1, B_2) & \cdots & IoU(B_1, B_n) \\ IoU(B_1, B_2) & 1 & \cdots & IoU(B_2, B_n) \\ \vdots & \vdots & \ddots & \vdots \\ IoU(B_1, B_n) & IoU(B_2, B_n) & \cdots & 1 \end{bmatrix} \tag{7.1}$$

So if I take the mean over all the terms under the diagonal, I could have a good score that quantifies the closeness of all pairs of objects.

However, in practice, most of the terms of A are equal to zero when n increases, and the score would be decreasing with the number of objects, which is not interesting in my case. So my approach consists of dividing the sum of all the terms of the diagonal by n instead of $\frac{n(n-1)}{2}$. This score S_{IoU} can now be between 0 and $\frac{(n-1)}{2}$ but it rarely goes beyond 1.

$$S_{IoU} = \frac{1}{n} \sum_{1 \leq i < j \leq n} IoU(B_i, B_j) \tag{7.2}$$

The results obtained using this formula for calculating density score seem to be a good representation of what I am looking for. Figure 7.4 shows some examples of pictures and their density scores according to Equation (7.2).

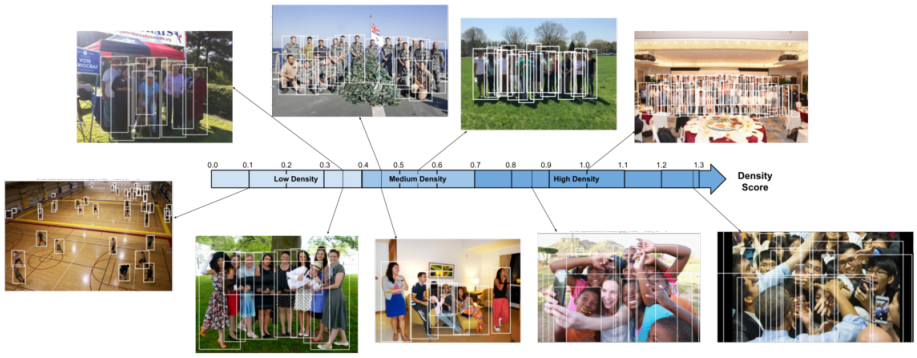


Figure 7.4: Density score examples. The axis is divided in three types of density: low, medium and high.

6.3.2 Object Size

To evaluate the detection behaviours under different circumstances, I calculated accuracy metric for each object size (small, medium and large). The default implementation of mean average precision calculates accuracy for each object size [2]. I have used this functionality to evaluate how the corner approach deals with each object size.

7 CrowdHuman Results

7.1 Filtered Dataset

The results of my first inferences concerning the first dataset are presented in Tables 7.7, 7.8, 7.9, 7.10, 7.11, 7.12, 7.13 and 7.14. We can see that most of the time, corner method works better than the standard one.

To start with the recall, we can observe that the corner recall is higher than standard recall in all cases I deal with. This is quite satisfying because I first want to increase the recall. Moreover, recall is improved with a higher rate when I look at high density photos. That makes sense since

[2] https://github.com/cocodataset/cocoapi/tree/master/PythonAPI/pycocotools

Table 7.7: Average Recall Results of Corner approach trained on a filtered subset of CrowdHuman dataset

	Standard	Corner	Relative Difference
Low density	40.1	42.4	+5.74%
Normal density	38.9	40.6	+4.37%
High density	34.3	36.4	+6.12%

Table 7.8: Average Recall on large objects Results of Corner approach trained on a filtered subset of CrowdHuman dataset

	Standard	Corner	Relative Difference
Low density	48.7	48.9	+0.21%
Normal density	43.4	44.7	+3%
High density	38.9	40.7	+4.63%

Table 7.9: Average Recall on medium objects Results of Corner approach trained on a filtered subset of CrowdHuman dataset

	Standard	Corner	Relative Difference
Low density	39.6	42.3	+6.82%
Normal density	32.6	34.9	+7.06%
High density	24.8	27.2	+9.68%

Table 7.10: Average Recall on small objects Results of Corner approach trained on a filtered subset of CrowdHuman dataset

	Standard	Corner	Relative Difference
Low density	24.2	29.6	+22.31%
Normal density	17.1	20.8	+21.64%
High density	9.1	12.9	+41.76%

Table 7.11: Mean Average Precision Results of Corner approach trained on a filtered subset of CrowdHuman dataset

	Standard	Corner	Relative Difference
Low density	34.6	35.7	+3.18%
Normal density	33.1	33.9	+2.42%
High density	28.9	30.3	+4.84%

Table 7.12: Mean Average Precision on large objects Results of Corner approach trained on a filtered subset of CrowdHuman dataset

	Standard	Corner	Relative Difference
Low density	43.1	42.9	-0.46%
Normal density	37.6	38.2	+1.6%
High density	33	34.4	+4.24%

the corner method had been developed to enhance detection in high density scenes. However, I see that my corner approach does not have a big impact on large objects in low density photos since it is not the kind of detection it aims to improve.

Regarding the mean average precision, I can notice that my corner approach performs better than standard approach in almost all cases. In fact, mean average precision depends on recall, when the recall is improved, this metrics is also improved. In general when the recall is increased, there is a risk that the precision of the detector decreases. But fortunately in my case, the recall improvement is more important than the eventual decrease in precision.

However, in some cases, the standard algorithm is still the best option between the two algorithms. For instance, when the goal is to detect small objects in high density images or large objects in low density images, the standard approach is slightly better.

What I see on those examples is that the corner network does a lot of multi-detections, which makes the precision low. These multi-detections should be avoided with NMS, but as NMS is only applied on each of the five branches, one object could be predicted by different branches and sent to classification component. The NMS algorithm performed at classification layers could be studied to overcome this limit.

7.2 Second Dataset

Tables 7.15 and 7.16 show the result concerning the second dataset. Unlike metrics used for the first dataset, I use less restrictive metrics, instead of averaging precision and recalls at different

Table 7.13: Mean Average Precision on medium objects Results of Corner approach trained on a filtered subset of CrowdHuman dataset

	Standard	Corner	Relative Difference
Low density	33.7	35.6	+5.64%
Normal density	26.4	27.9	+5.68%
High density	20.2	21.8	+7.92%

Table 7.14: Mean Average Precision on small objects Results of Corner approach trained on a filtered subset of CrowdHuman dataset

	Standard	Corner	Relative Difference
Low density	17.5	21	+20%
Normal density	10	11.1	+11%
High density	4.1	4	-2.44%

Table 7.15: Recall @0.5 IoU Results of Corner approach trained on a not filtered subset of CrowdHuman dataset

	Standard	Corner	Relative Difference
Low density	75.76	78.91	+4.16%
High density	67.05	70.86	+5.68%
Mini subset	85.26	88.61	+3.93%

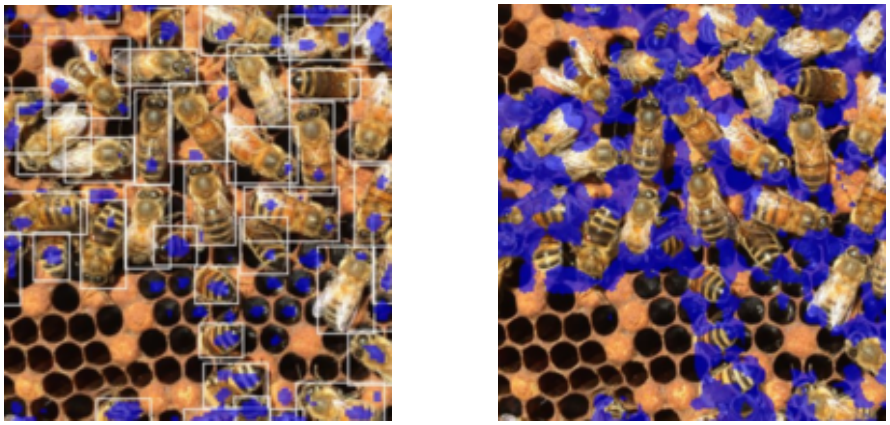
Table 7.16: Mean Precision @0.5 IoU Results of Corner approach applied on a not filtered subset of CrowdHuman dataset

	Standard	Corner	Relative Difference
Low density	66.38	68.87	+3.75%
High density	60.21	63.13	+4.85%
Mini subset	80.68	82.76	+2.58%

IoU values, I calculate only the metrics at 0.5 IoU threshold. I notice again that corner approach performs better than standard approach.

8 Detecting Corners as Objects

Many approaches for object detection are based on detecting objects from their centers. Object centers are considered as visible for each object to detect. But in real situation, especially in dense scenes, there are some objects that are partially hidden, only corner regions are visible. My proposed approach deals with this issue by extending the neural network structure. Another way to deal with this problem would be to train the neural network to detect object corners. I have tested this approach using Faster R-CNN neural network. To obtain corner annotations, I divide each bee bounding box into a grid of 4×4, the top left, top right, bottom left and bottom right tiles are converted to bounding box annotations. The training is performed on this new obtained annotations. Figure 7.5 shows qualitative results. It shows an example of bee frame image on which inference was launched, the blue points correspond to regions that neural network thinks they correspond to object corners (the objectness of these points are bigger than a determined threshold). I notice that my approach detects more precisely object corner regions.



(a) (b)

Figure 7.5: (a) Corners detected using corner approach and (b) Corner detected by training Faster R-CNN on corner annotations

9 PNAPI Platform

My thesis work is part of a global project called PNAPI (Section 2.5). This project aims to provide to beekeepers a platform that help them managing their apiaries. To value my research work, I have participated in the design and the development of this platform. In fact my proposed approaches can be connected to a global system that manage bee colonies data. This integration would add intelligence to the system. To produce this platform, I was involved in a team of software developers, and members of PNAPI project specialized in beekeeping domain. They were specifying the platform requirements. In this section, I describe this platform.

9.1 Platform Architecture

The platform is based on client server architecture. The client side (frontend), is a web application developed using Angular framework. The server side (backend) is composed of two main interconnected components: a REST API developed using Node JS technology, it handles client requests. And a database manager which handles data manipulation, it is developed using MongoDB. Figure 7.6 shows the global architecture of the system.

9.2 Platform Functionalities

This platform has several functionalities that can be used by a beekeeper to manage its apiary and colony data. They are divided into four types:

- Account Management: The user can register itself and manage its credentials.
- Locations and Apiaries: The user can manage its apiary data, their hives and also locations where apiary could be located.
- Entered Data Management: The user can enter some data related to some passed events.
- Planning: The user can plan specific actions in its calendar.

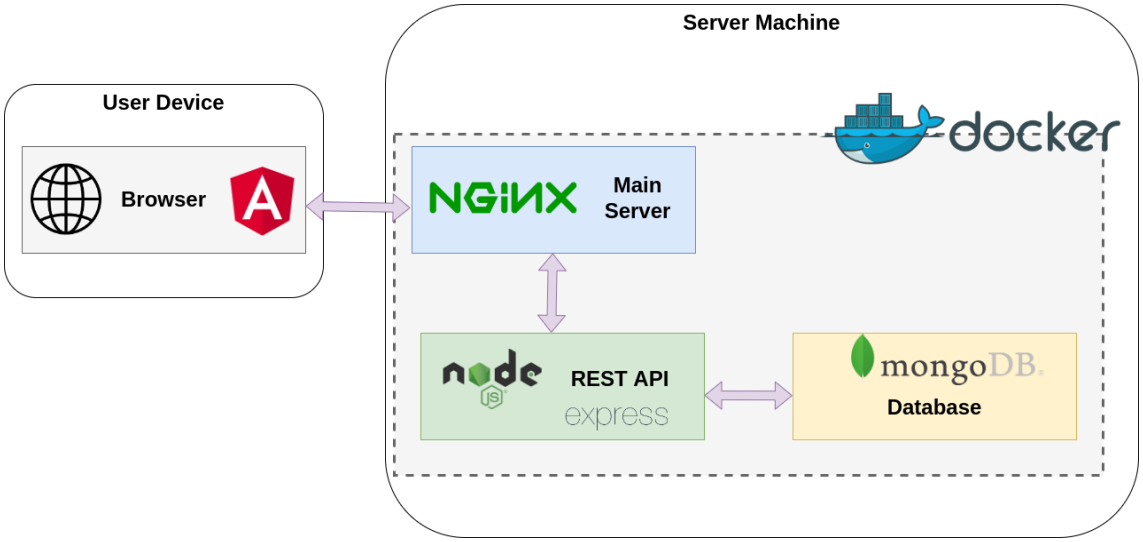


Figure 7.6: PNAPI Platform Architecture

Figure 7.7 presents the functionalities offered by the platform. Figure 7.8, 7.9 and 7.10 are some screenshots of interfaces of the platform application.

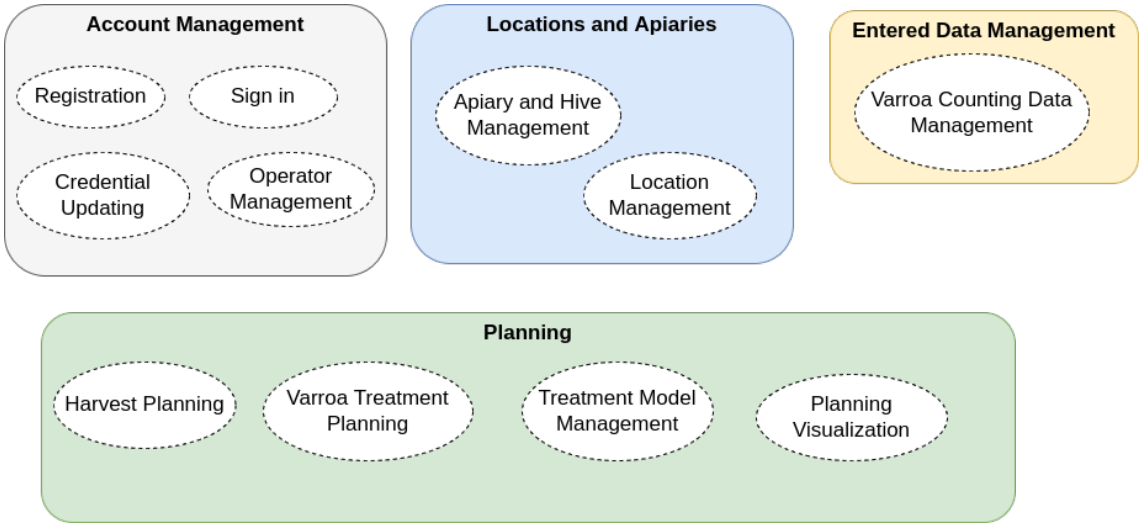


Figure 7.7: PNAPI Platform Functionalities

9.3 Data Model

The platform allows its users to manage their data about apiaries. To reach this objective, I have designed a data model that meets the needs of beekeepers. Each user has its apiaries, each one is located in a location, and has one or more hives. The user can manage a list of operators. He can plan some actions in its calendar like harvests or varroa treatments. He can also enter data about counted varroas in an apiary. Figure 7.11 shows the data model of the platform.

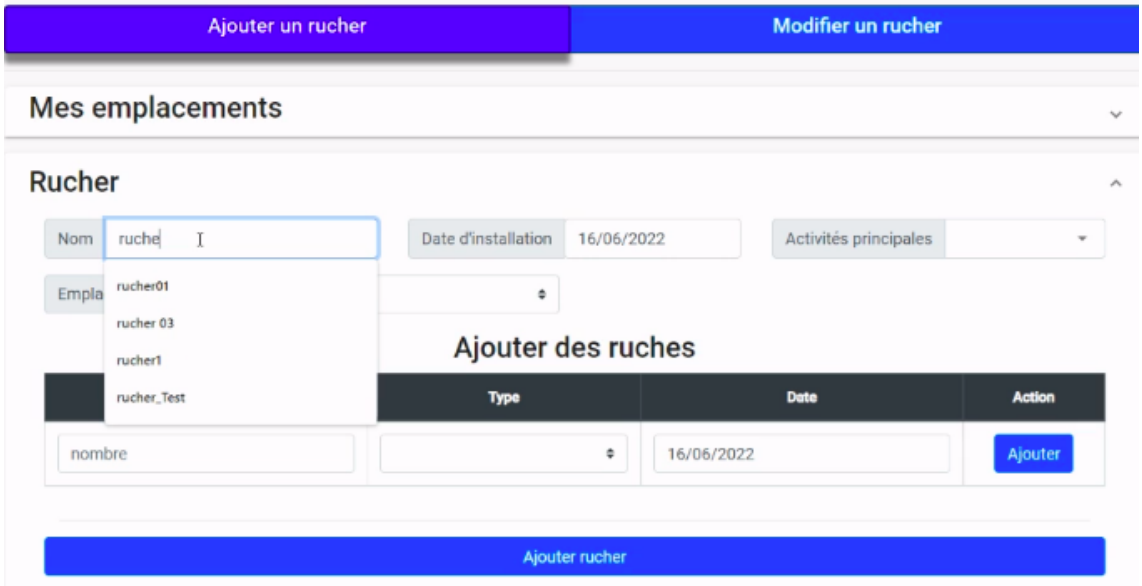


Figure 7.8: Apiary Interface Screenshot

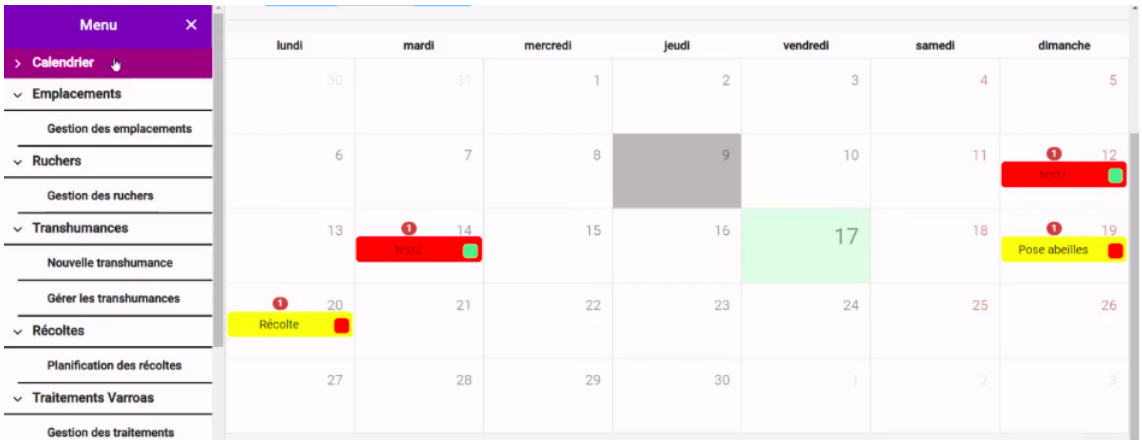


Figure 7.9: Calendar Interface Screenshot

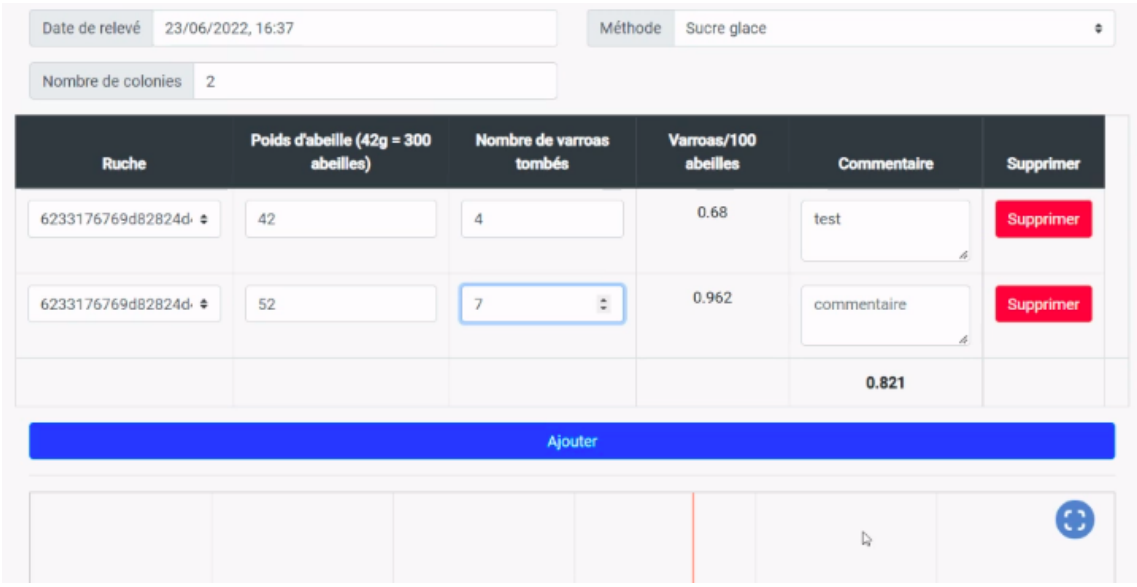


Figure 7.10: Varroa Counting Interface Screenshot

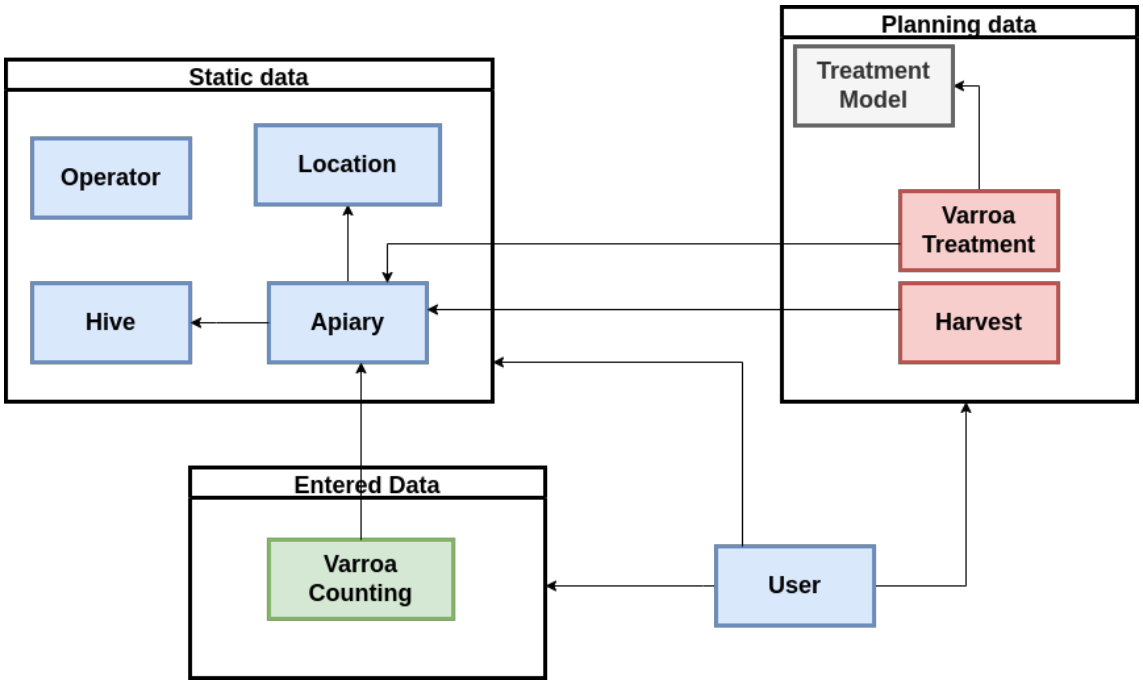


Figure 7.11: PNAPI Platfrom Data Model

9.4 Varroa Counting Module

As seen previously, the developed platform allows users to enter data about counted varroas. In fact this is an important data for beekeepers to decide whether treatment action should be triggered. My object detection approaches could be directly transformed to software services that can be called by the platform to calculate the infestation level. Instead of counting manually varroas, the beekeeper would take a photo of bee frame using its phone, he would upload this photo using the platform web application. Then the image is sent to the neural network module that launches the detection of bees and varroas to predicts the infestation level (percentage of infested bees), the output value is returned to the user and the counting form is automatically filled.

10 Infestation Level

The final objective of my thesis is to estimate the infestation level from detected bees and varroas. The infestation level depends on some variables like percentage of varroa per bee and season. To validate my approaches with respect to infestation level estimation, I need to build a specific dataset of annotated images. These images should be annotated by their hives and the time of capturing. The ground truth infestation level should be given by some beekeeper experts. In my case, I don't have the resources and time to build such dataset. Neverthless, I built a dataset of 33 images annotated by 1251 bees and 78 varroas and compared ground truth infestation percentage and predicted infestation percentage. The table 7.17 shows the results of my experiments, I notice clearly that the approach "Res3+4" reduces the error by 0.22%, this is interesting as we must be precise in case of infestation estimation. In fact, the thresholds used by beekeepers for treating varroas are between 1% and 5% (section 2.7).

Table 7.17: Results of varroa percentage estimation

Approach	Mean Absolute Error	Variance	Confidence Interval
Consecutive Detection	1.16%	2.57	[-2.05%, 4.36%]
FPN extraction	2.14%	4.09	[-1.91%, 6.18%]
Res4 extraction	1.23%	2.50	[-1.94%,4.39%]
Res3+4 Branch	0.94%	2.17	[-2%, 3.89%]

11 Computational Overhead

As I have explained previously, my proposed approach for detecting nested objects (bees and varroas) is more efficient than the standard separate detection. This efficiency is in terms of resources and time consumed during inference. To evaluate the relevance of the approach compared to the standard one, I have compared these metrics:

- Inference time

- Number of parameters of the neural network
- Model memory consumption
- Flops (number of floating point operations) and Macs (number of multiply-accumulate operations)

The results are depicted by table 7.18. They were calculated by repeating 10 times inference on 30 images. I notice that the approach giving the best results "Res3+4" reduces each time resource consumption compared to consecutive detection.

Table 7.18: Computational overhead comparing proposed approaches and standard approach for detecting bees and varroas

	Time (ms)	Params Number (G)	Model Memory (MB)	flops (G)	macs (GMACS)
Faster R-CNN bee prediction	99	41	168	341	170
Mask R-CNN varroa segmentation	82	31	121	230	115
FPN extraction	11	27			
Consecutive Detection	181	72	289	571	285
FPN extraction	104	46	188	355	177
Res4 extraction	114	46	188	430	215
Res3+4 Branch	131 (-28%)	54 (-26%)	221 (-23%)	510 (-11%)	255 (-11%)

12 Conclusion

In this chapter, I have presented experiments I carried out to validate my approach results. By displaying and swapping backbone feature maps, I have validated that Res4 features can be used for varroa segmentation. I have also noticed that extracting from Res4 feature map is more interesting than using standard feature map selection implemented by FPN. This can be explained by the fact that detecting varroas needs different information than detecting bees.

Regarding training, I have tested merging data coming from different dataset (bee annotations and varroa annotations) into one batch. The obtained results are less interesting than forwarding two batches or transfer learning.

I have also confirmed that detecting varroa from bees is more interesting than detecting them from bee frames. To confirm that, I have used U-Net architecture based on MobileNet and launch training on dataset containing varroa annotations on bees and another dataset containing varroa annotations in bee frames. The results show that the first method is more relevant.

I have proposed another way to represent varroas. Instead of predicting varroa regions, the neural network estimates varroa densities. I have used elliptical gaussian density for each varroa as it is originally annotated by an ellipse. The obtained results show that density estimation is promising. It can be improved to achieve counting task by integrating the output densities.

To validate the relevance of corner approach, I have trained it on a subset of CrowdHuman dataset, the obtained result are positive and promising. Due to library constraint and resource

limit, I have filtered dataset and modified corner approach to reduce the number of corner branches to three instead of four. More work can be carried out to assess the relevance of the approach on the whole dataset.

I have also qualitatively compared corner approach to an approach consisting of training neural network on annotations obtained by cropping each bee corner. I have concluded that corner approach detects corner regions better.

Finally, I have participated in realizing PNAPI platform which will host the implementation of my proposed approaches. Thus, my research results are useful for beekeepers to automatically monitor the infestation level of their colonies by varroas.

Conclusion and Perspectives

Résumé

L'objectif de ma thèse était de proposer des approches pour la détection des objets imbriqués dans les scènes denses. Cette problématique est fréquente dans différentes situations de la vie courante. Grâce aux avancées fournies par les technologies d'apprentissage profond, la détection rapide et efficace des objets sur des images est possible par le biais des réseaux de neurones. Mais cette détection devient plus difficile dans le cas des scènes denses et objets imbriqués.
En effet, la détection standard des objets par réseaux de neurones se base sur la détection des objets à partir des centres des régions. Dans le cas des scènes denses, les centres peuvent être cachés par d'autres objets, par conséquent des objets peuvent être loupés par le détecteur. Pour pallier cette limite, on détecte les objets par leurs coins, mais souvent de nouvelles données d'annotation doivent être fournies.
Je propose de résoudre cette problématique en étendant le détecteur d'objets Faster R-CNN. Ce dernier est un réseau de neurones proposé pour la détection précise des objets. Il est composé du sous-réseau de neurones RPN qui génère les propositions. Je propose d'ajouter quatre branches additionnelles à RPN pour la détection d'objets à partir des différents coins d'un objet. Après expérimentation, mon réseau de neurones améliore le rappel, dans le cas de la détection d'abeilles dans les scènes denses, jusqu'à 10% et améliore la précision jusqu'à 6%. Dans le cas de la détection des personnes, la précision et le rappel sont améliorés de 3%.
Concernant la détection des objets imbriqués, l'approche standard consiste à réaliser une détection successive des objets. Elle se base sur l'utilisation de deux réseaux de neurones, chacun est entraîné pour trouver les positions d'un type donné d'objets. Cette approche manque d'efficacité car les caractéristiques sont extraites deux fois à partir d'une même image. Je propose d'utiliser le réseau de neurones Mask R-CNN pour la détection en une seule fois des objets imbriqués, dans mon cas, les abeilles et les varroas. Ce réseau est entraîné pour la détection des abeilles et la segmentation des varroas. Pour cela j'ai proposé deux méthodes d'entraînement, la première consiste à fournir à chaque itération des images provenant de deux bases d'images différentes, et la deuxième consiste à réaliser deux entraînements consécutifs. Pour extraire les informations pertinentes de l'image pour la détection d'objets de types différents, j'ai procédé à quelques tests et expérimentations. J'ai conclu que l'ensemble de caractéristiques de la couche "Res4" du réseau Mask R-CNN contient des informations pertinentes pour la détection d'abeilles et la segmentation de varroas.

Résumé

Mes propositions permettent d'aboutir à un réseau de neurones qui améliore la détection de varroas de 1.9% pour la précision sans perte de précision au niveau de la détection des abeilles par rapport à une détection consécutive. J'aboutis à un réseau qui améliore la détection des varroas de 11% de précision par rapport à l'extraction standard des caractéristiques.

Pour entraîner et évaluer mes approches, j'ai construit une base d'images annotées par des abeilles et des varroas: 63 images de cadre d'abeilles, 3863 abeilles annotées, 565 abeilles infectées annotées, 679 abeilles non infectées annotées et 638 varroas annotés.

Il y a bien évidemment des modifications à réaliser sur les approches afin d'améliorer leurs résultats. En effet, j'ai remarqué que parfois l'approche des coins génère des faux positifs, cela peut être résolu en améliorant l'algorithme NMS. Un autre point intéressant à traiter est l'occupation de mémoire additionnelle utilisée par l'ajout de 4 branches au réseau. J'ai testé l'utilisation d'un nombre réduit de branches et le résultat est prometteur. Une autre piste intéressante de recherche serait de tester mes approches sur différentes architectures et sur différents domaines. Par ailleurs, afin de répondre aux besoins des apiculteurs, un travail de recherche supplémentaire doit être réalisé pour obtenir un système capable d'estimer le niveau d'infestation à partir des images de cadre d'abeilles. Pour valoriser ce système, il faut l'intégrer dans la plateforme d'aide aux apiculteurs.

The objective of my research is to propose methods to detect nested objects in dense scenes. This kind of tasks is very useful in real life. For example it is used in medicine domain to detect infected regions in human cells, or in agriculture domain to detect parasites on animals. Detecting objects in dense scenes can be a solution for situations where many objects close to each other must be detected. For example, it can be used to accurately estimate the number of people in a public space, or to predict the size of colony of insects where occlusion if very frequent.

Thanks to the advances enabled by deep learning methods, several approaches were proposed to deal with object detection. Nowadays, we can accurately and quickly detect objects in normal situations thanks to neural networks. But when the density of objects in an image is very high, the miss rate of object detector increases. The standard approaches do not have the capability to detect partially occluded objects. Regarding nested object detection, the main issue of the existing approaches is the time and resources required to detect objects of different types.

Neural networks proposed for object detection, are usually trained to predict objects from their center. It means that object center regions are used to predict the presence and the position of objects. Thus objects whose center regions are hidden can be missed. To overcome this issue, other approaches were proposed to detect objects from corner regions. But they usually require corner annotations.

Thus I solve these issues by proposing the corner approach based on the Faster R-CNN neural network through its component RPN. In addition to detecting object from their center, they are also detected from their corners without providing additional annotations. Therefore, partially occluded objects have more chance to be found by the neural network.

More precisely, RPN is a sub neural network trained for generating proposals. It generates, at each feature map position, anchors which are bounding boxes with defined size and ratio. To get the final proposal set, each obtained bounding box is classified as object or not. My proposed approach extends this neural network structure by adding four branches to detect objects from their corners. To implement this approach, I have modified the anchor generation mechanism so that generated anchors at corners are shifted to match object regions.

I obtained the following satisfactory positive results:

- The corner approach improves the recall of Faster R-CNN on bee detection up to +6%
- The corner approach improves the recall of RPN on bee detection up to +10%
- The corner approach improves the precision of Faster R-CNN on bee detection up to +6%
- The corner approach improves the precision on CrowdHuman dataset up to +3%
- The corner approach improves the recall on CrowdHuman dataset up to +3%

To handle detection of nested objects, the common approach is to use two neural networks. Each one is trained on objects of a specific type. This approach requires memory and time. The alternative method, is to use one multi-class object detector trained to detect objects of different types. This approach suffer from two issues: (1) difficulty of finding common features

using the same set of neural network parameters, (2) difficulty to train the neural network on two different dataset. In fact, sometimes, especially when the occurrence probabilities of the nested objects are very different, the training set is divided into two different image sets each of which is annotated with annotations of a specific object type. As far as I know, the challenge of detecting nested objects of different occurrence probabilities using one neural network is not yet addressed.

To overcome the problems, I proposed an approach based on Mask R-CNN. It detects enclosing objects and segment internal objects using one neural network. I proposed a training method that uses two dataset during learning phase.

More precisely, after generating bee proposals, each proposal is classified, then varroa regions are predicted from bee features. I have proposed two ways of training: either (1) training neural network using two batches at each iteration, each batch corresponds to an object type, or (2) perform two consecutive training of the same neural network on the two datasets. Moreover, I concluded that extracting Res4 features from the backbone for varroa segmentation is relevant to obtain similar accuracy as traditional separate neural networks.

I have reached the second objective of my thesis, thanks to the following contributions:

- The Mask R-CNN approach proposed for nested object detection uses one neural network and has similar performance as two separate neural networks:
  - The transfer learning method based on using branch Res3+4 improves the precision of varroa point prediction compared to separate neural networks by +1.9% and has similar precision of bee bounding box prediction
- The Res4 feature extraction improves the precision of varroa point prediction compared to standard FPN selection up to +11%

My proposed approaches are applied in beekeeping domain to help beekeepers in their monitoring tasks. In fact to protect their bee colonies, beekeepers should estimate bee quantities and the level of infestation by the varroa parasite. Varroas are located on bees which are usually occluding each other.

- I have proposed an approach based on density estimation, that can be used to count directly varroas from densities. The result are promising: just -5% of varroa prediction precision decrease compared to Mask R-CNN trained only on varroa segmentation

To achieve my objectives I have developed my techniques according to the following points:

- I have constructed a dataset of 63 bee frame images, 3863 annotated bees, 565 annotated infected bees, 679 annotated not infected bees, and 638 annotated varroas.
- I have proposed a keypoint metric to evaluate varroa position prediction. It is less restrictive and more adequate for detecting objects of small size and same shape.

The proposed approaches for nested object detection can be directly implemented as software services. Thus, they can be used for inference on real photos taken by beekeepers to help them monitor varroa infestation level.

Perspectives

NMS (Non Maximum Suppression) is used in object detectors to delete duplicated detections: predicted bounding boxes that correspond to already detected objects. It favour bounding boxes with high scores. In standard Faster R-CNN, NMS is applied two times, the first execution is after generating proposals by RPN, the second one is after classifying them. In corner approach, in RPN, NMS is launched on each branch output, then the results are concatenated to get the final set of proposals. I have tested concatenating bounding boxes before applying NMS, but the accuracy was not satisfactory. I think that it is because of the bounding box scores of center branch that are not in balance with other branch scores. In fact, detecting objects from center gives scores that are greater in general than scores predicted by corner branches. Thus, after concatenating boxes coming from different branches, ordering them would put center boxes at first positions. Therefore NMS would neglect boxes predicted by corner branches. Concatenating proposals after applying NMS gives rise to duplicated detections. The second NMS execution could remove this double detection, but the results that I got show that sometimes the second NMS execution does not filter out duplicated predictions. Further investigation should be carried out to find out the reason of this issue that degrades the precision and increase the number of false positives.

The main drawback of the corner approach is its memory usage. This can be annoying in case the memory is limited. I have proposed other alternatives that are more or less interesting and need less memory such as using two branches instead of five (one for center detection, and the other for corner detection), and using four branches instead of five for human detection. The results are promising, more research can be realized to validate my findings.

To validate my corner approach I have built a dataset of human images. It is a subset of CrowdHuman dataset. Due to resource limits, I have not tested the approach on official set dedicated for training. Using the whole dataset is very important step that should be realized to analyze more deeply the approach and compare it to published approaches proposed for predicting humans of CrowdHuman dataset.

The approach that I have proposed to detect nested objects has been tested and applied on bee and varroa case. The particularity of this case is that the types of objects to detect (bee and varroa) are very different in shape and occurrence probability. These two characteristics are usually present in infestation phenomena. Therefore my proposed approach can be extended to other domains where living beings are infected by smaller living beings.

The nested object detection approach that I have proposed aims to facilitate the counting task of varroas present in colonies. Further work must be carried out to evaluate the relevance of the approach towards infestation level estimation. This work needs some ground truth data about the infested level of a set of colonies. A counting method should be proposed, it would take as input the detected bees and varroa segmentation result for each bee, as output it must give the number of bees and number of varroas or number of infected bees. In my thesis, I have proposed another method that predicts varroa density from which varroa number can be automatically inferred. These two approaches can be improved and tested for predicting infestation level.

Finally, I believe that my work could benefit beekeeping domain. In fact, the project PNAPI gave the opportunity to construct tools and approaches that would help beekeepers in their task and protect bee health. This work should be continued to reach a complete system equipped with artificial intelligence functionalities that bring automation and improve work efficiency. It must be able to accompany and assist beekeeper in his daily actions to take care of its bees and increase its products.

List of Publications

Journal Papers in Preparation

Jean-Charles HUET, Lamine BOUGUEROUA, **Yassine KRIOUILE**, Katarzyna WEGRZYN-WOLSKA, Corinne ANCOURT. "*Digital transformation of beekeeping through the use of a decision making architecture*". *in review for the journal MDPI Applied Science.*

Yassine KRIOUILE, Corinne ANCOURT, Katarzyna WEGRZYN-WOLSKA, Lamine BOUGUEROUA. "*Nested Object Detection using Mask R-CNN*". *Journal to be decided.*

International Conference

Jean-Charles HUET, Lamine BOUGUEROUA, **Yassine KRIOUILE**, Alain MORETTO. "*Toward an intelligent system architecture for smart agriculture: application to smart beehives*". ISDA 2020: 20th International Conference on Intelligent Systems Design and Applications.

Yassine KRIOUILE, Corinne ANCOURT, Katarzyna WEGRZYN-WOLSKA, Lamine BOUGUEROUA. "*Generating proposals from corners in RPN to detect bees in dense scenes*". VISAPP 2022: 17th International Conference on Computer Vision Theory and Applications.

Presented Poster

Yassine KRIOUILE, Lamine BOUGUEROUA, Corinne ANCOURT, Katarzyna WEGRZYN-WOLSKA, Jean-Charles HUET. "*PNAPI Digital Support Platform for Beekeepers*". A2IA 2020 conference: Artificial Intelligence and Industrial Applications.

References

[Acharjya, 2012] Pinaki Pratim Acharjya and Dibyendu Ghoshal. "Watershed Segmentation based on Distance Transform and Edge Detection Techniques". *International Journal of Computer Applications* 52 (2012), pp. 6–10.

[Apizoom, 2022] Apizoom. *Apizoom varroas detection*. 2022. URL: https://www.apizoom.app/ (visited on 09/02/2022).

[Babic, 2016] Z. Babic, R. Pilipovic, V. Risojevic, and G. Mirjanic. "Pollen bearing honey bee detection in hive entrance video recorded by remote embedded system for pollination monitoring". *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* III-7 (2016), pp. 51–57.

[Bassignana, 2018] Davide Bassignana, Chiara Flora Bassignana, Davide Cuttini, Gianluca Francini, Maurizio Ghirardi, Daniela Laurino, et al. "Automatic Varroa Counting Bee Varroa Scanner" (2018).

[BeeScanning, 2022] BeeScanning. *BeeScanning - Detect varroa mites*. 2022. URL: https://beescanning.com/ (visited on 09/02/2022).

[Bhairannawar, 2018] Satish S. Bhairannawar. "Chapter 4 - Efficient Medical Image Enhancement Technique Using Transform HSV Space and Adaptive Histogram Equalization". *Soft Computing Based Medical Image Analysis*. Ed. by Nilanjan Dey, Amira S. Ashour, Fuqian Shi, and Valentina E. Balas. Academic Press, 2018, pp. 51–60.

[Bienefeld, 2015] Kaspar Bienefeld, Fred Zautke, and Pooja Gupta. "A novel method for undisturbed long-term observation of honey bee (*Apis mellifera*) behavior – illustrated by hygienic behavior towards varroa infestation". *Journal of Apicultural Research* 54.5 (2015), pp. 541–547.

[Bilik, 2021] Simon Bilik, Lukas Kratochvila, Adam Ligocki, Ondrej Bostik, Tomas Zemcik, Matous Hybl, et al. "Visual Diagnosis of the Varroa Destructor Parasitic Mite in Honeybees Using Object Detector Techniques". *Sensors* 21.8 (2021).

[Bjerge, 2019] Kim Bjerge, Carsten Eie Frigaard, Peter Høgh Mikkelsen, Thomas Holm Nielsen, Michael Misbih, and Per Kryger. "A computer vision system to monitor the infestation level of Varroa destructor in a honeybee colony". *Computers and Electronics in Agriculture* 164 (2019), p. 104898.

[Calavas, 2018] Didier Calavas. *Mortalité des colonies d'abeilles domestiques pendant l'hiver 2017-2018*. 2018.

[Campbell, 2008] Jason Campbell, Lily Mummert, and Rahul Sukthankar. "Video Monitoring of Honey Bee Colonies at the Hive Entrance" (2008), p. 4.

[Chazette, 2016] Larissa Chazette, Matthias Becker, and Helena Szczerbicka. "Basic algorithms for bee hive monitoring and laser-based mite control". *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*. Athens, Greece: IEEE, 2016, pp. 1–8.

[Chen, 2012] Chiu Chen, En-Cheng Yang, Joe-Air Jiang, and Ta-Te Lin. "An imaging system for monitoring the in-and-out activity of honey bees". *Computers and Electronics in Agriculture* 89 (2012), pp. 100–109.

[Chiron, 2013a] Guillaume Chiron, Petra Gomez-Krämer, and Michel Ménard. "Detecting and tracking honeybees in 3D at the beehive entrance using stereo vision". *EURASIP Journal on Image and Video Processing* 2013.1 (2013), p. 59.

[Chiron, 2013b] Guillaume Chiron, Petra Gomez-Krämer, Michel Ménard, and Fabrice Requier. "3D Tracking of Honeybees Enhanced by Environmental Context". *Image Analysis and Processing – ICIAP 2013*. Ed. by David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, et al. Vol. 8156. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 702–711.

[Cole, 2004] Luke Cole, David J. Austin, and Lance Cole. "Visual Object Recognition using Template Matching". 2004.

[Dai, 2016] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. "R-FCN: Object Detection via Region-Based Fully Convolutional Networks". *International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2016.

[David, 2006] David. *CSL : "Varroa model" et "Monitoring methods"*. 2006. URL: http://gdsa27.free.fr/spip.php?article110 (visited on 09/01/2022).

[David, 2016] Eli David and Nathan Netanyahu. "DeepPainter: Painter Classification Using Deep Convolutional Autoencoders". 2016, pp. 20–28.

[Déborah, 2020] Déborah. *30% c'est le taux de mortalité actuelle des abeilles*. 2020. URL: http://www.univers-nature.com/chiffre-cle/30-cest-le-taux-de-mortalite-actuelle-des-abeilles-63958.html. (visited on: 26.08.2020).

[Deep Learning, 2022] Dive into Deep Learning. *Transposed Convolution*. 2022. URL: https://d2l.ai/chapter_computer-vision/transposed-conv.html (visited on 08/26/2022).

[Défense Sanitaire Apicole du Var, 2022] Groupement de Défense Sanitaire Apicole du Var. *Méthodes d'évaluation de l'infestation par Varroa destructor au cours de l'année et interprétation diagnostic*. 2022. URL: http://gdsa83.fr/evaluation-varroa/. (visited on: 01.09.2022).

[Dembski J, 2020] Szymański J. Dembski J. "Weighted Clustering for Bees Detection on Video Images". *Computational Science*. Springer, 2020.

[Do, 2008] Chuong B. Do. *The Multivariate Gaussian Distribution*. 2008.

[Duan, 2020] Kaiwen Duan, Lingxi Xie, Honggang Qi, Song Bai, Qingming Huang, and Qi Tia. "Corner Proposal Network for Anchor-free, Two-stage Object Detection". *European Conference on Computer Vision – ECCV*. 2020.

[Dupleix, 2018] Anna Dupleix, Shu Mui, François Pfister, and Victor Reutenauer. "Application smartphone pour le comptage automatisé du varroa" (2018), p. 2.

[Elizondo, 2013] Víctor Elizondo, Juan C. Briceño, Carlos M. Travieso, and Jesús B. Alonso. "Video Monitoring of a Mite in Honeybee Cells". *Advanced Materials Research* 664 (2013), pp. 1107–1113.

[Facchini, 2019] Elena Facchini, Laura Nalon, Maria Elena Andreis, Mauro Di Giancamillo, Rita Rizzi, and Michele Mortarino. "Honeybee pupal length assessed by CT-scan technique: effects of Varroa infestation, developmental stage and spatial position within the brood comb". *Scientific Reports* 9.1 (2019), p. 10614.

[Fernandes, 2008] Leandro A.F. Fernandes and Manuel M. Oliveira. "Real-time line detection through an improved Hough transform voting scheme". *Pattern Recognition* 41.1 (2008), pp. 299–314.

[FranceAgriMer, 2015] FranceAgriMer. *La production française de miel et de gelée royale en France en 2014*. 2015.

[FranceAgriMer, 2017] FranceAgriMer. *Bilan de campagne Miel en 2017*. 2017.

[Gählert, 2020] Nils Gählert, Niklas Hanselmann, Uwe Franke, and Joachim Denzler. *Visibility Guided NMS: Efficient Boosting of Amodal Object Detection in Crowded Traffic Scenes*. 2020.

[Girshick, 2015] Ross Girshick. "Fast R-CNN". *2015 IEEE International Conference on Computer Vision (ICCV)*. 2015.

[Girshick, 2014] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014.

[Giuffre, 2017] Carl Giuffre, Sharon R. Lubkin, and David R. Tarpy. "Automated assay and differential model of western honey bee (Apis mellifera) autogrooming using digital image processing". *Computers and Electronics in Agriculture* 135 (2017), pp. 338–344.

[Goldman, 2019] Eran Goldman, Roei Herzig, Aviv Eisenschtat, Jacob Goldberger, and Tal Hassner. "Precise Detection in Densely Packed Scenes". *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[He, 2017] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask R-CNN". *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 2980–2988.

[He, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[Hendriks, 2012] Cris L Luengo Hendriks, Ziquan Yu, Antoine Lecocq, Teatske Bakker, Barbara Locke, and Olle Terenius. "Identifying All Individuals in a Honeybee Hive – Progress Towards Mapping All Social Interactions" (2012), p. 4.

[Hood, 2020] Mike Hood. *colony collapse disorder*. 2020. URL: https://www.britannica.com/science/colony-collapse-disorder. (visited on: 17.08.2020).

[ITSAP, 2020] ITSAP. *ITSAP*. 2020. URL: https://itsap.asso.fr/. (visited on: 17.08.2020).

[Jiao, 2019] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, et al. "A Survey of Deep Learning-Based Object Detection". *IEEE Access* 7 (2019), pp. 128837–128868.

[Kanopoulos, 1988] N. Kanopoulos, N. Vasanthavada, and R.L. Baker. "Design of an image edge detection filter using the Sobel operator". *IEEE Journal of Solid-State Circuits* 23.2 (1988), pp. 358–367.

[Kimura, 2014] Toshifumi Kimura, Mizue Ohashi, Karl Crailsheim, Thomas Schmickl, Ryuichi Okada, Gerald Radspieler, et al. "Development of a New Method to Track Multiple Honey Bees with Complex Behaviors on a Flat Laboratory Arena". *PLoS ONE* 9.1 (2014).

[Kimura, 2011] Toshifumi Kimura, Mizue Ohashi, Ryuichi Okada, and Hidetoshi Ikeno. "A new approach for the simultaneous tracking of multiple honeybees for analysis of hive behavior". *Apidologie* 42.5 (2011), pp. 607–617.

[Knauer, 2005] Uwe Knauer, Michael Himmelsbach, Frank Winkler, Fred Zautke, Kaspar Bienefeld, and Beate Meffert. "Application of an adaptive background model for monitoring honey-bees" (2005), p. 6.

[Knauer, 2007] Uwe Knauer, Fred Zautke, Kaspar Bienefeld, and Beate Meert. "A Comparison of Classiers for Prescreening of Honeybee Brood Cells" (2007), p. 12.

[Kulyukin, 2016] Vladimir A Kulyukin and Sai Kiran Reka. "Toward Sustainable Electronic Beehive Monitoring: Algorithms for Omnidirectional Bee Counting from Images and Harmonic Analysis of Buzzing Signals" (2016), p. 11.

[Kulyukin, 2019] Vladimir Kulyukin and Sarbajit Mukherjee. "On Video Analysis of Omnidirectional Bee Traffic: Counting Bee Motions with Motion Detection and Image Classification". *Applied Sciences* 9.18 (2019), p. 3743.

[Latioui, 2020] Zine Eddine Latioui, Lamine Bougueroua, and Alain Moretto. "Social Media Chatbot System - Beekeeping Case Study". *Hybrid Intelligent Systems*. Ed. by Ana Maria Madureira, Ajith Abraham, Niketa Gandhi, and Maria Leonilde Varela. Cham: Springer International Publishing, 2020, pp. 302–310.

[Law, 2018] Hei Law and Jia Deng. "CornerNet: Detecting Objects as Paired Keypoints". *European Conference on Computer Vision (ECCV)*. 2018.

[Li, 2021] Pei Li, Hongjuan Wang, Mengbei Yu, and Yeli Li. "Overview of Image Smoothing Algorithms". *Journal of Physics: Conference Series* 1883 (2021), p. 012024.

[Lin, 2017a] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. "Feature Pyramid Networks for Object Detection". *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[Lin, 2017b] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal Loss for Dense Object Detection". *IEEE International Conference on Computer Vision (ICCV)*. 2017.

[Liu, 2016] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, et al. "SSD: Single Shot MultiBox Detector". *Lecture Notes in Computer Science* (2016).

[Liu, 2019] Yu Liu, Lingqiao Liu, Hamid Rezatofighi, Thanh-Toan Do, Qinfeng Shi, and Ian Reid. *Learning Pairwise Relationship for Multi-object Detection in Crowded Scenes*. 2019.

[Long, 2015] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440.

[Lowe, 1999] D.G. Lowe. "Object recognition from local scale-invariant features". *Proceedings of the Seventh IEEE International Conference on Computer Vision*. Vol. 2. 1999, 1150–1157 vol.2.

[Magnier, 2018] Baptiste Magnier, Gaëtan Ekszterowicz, Joseph Laurent, Matthias Rival, and François Pfister. "Bee Hive Traffic Monitoring by Tracking Bee Flight Paths:" *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 563–571.

[Magnier, 2019] Baptiste Magnier, Eliyahou Gabbay, Faysal Bougamale, Behrang Moradi, François Pfister, and Pierre R. Slangen. "Multiple honey bees tracking and trajectory modeling". *Multimodal Sensing: Technologies and Applications*. Ed. by Shahriar Negahdaripour, Ettore Stella, Dariusz Ceglarek, and Christian Möller. Munich, Germany: SPIE, 2019, p. 29.

[Maisonnasse, 2016] Alban Maisonnasse, Julie Hernandez, C. Quintec, Marianne Cousin, Constance Beri, and Andre Kretzschmar. "Evaluation de la structure des colonies d'abeilles, création et utilisation de la méthode ColEval (Colony Evaluation)" (2016).

[Marc, 2019] Clara Marc. *De l'intelligence artificielle pour sauver les abeilles*. 2019. URL: https://actu.epfl.ch/news/de-l-intelligence-artificielle-pour-sauver-les-abe/ (visited on 03/29/2020).

[McGregor, 2022] Samuel Emmett McGregor. *beekeeping*. 2022. URL: https://www.britannica.com/topic/beekeeping (visited on 09/01/2022).

[Moisset, 2021] Beatriz Moisset. *Anthophila (Apoidea) - Bees*. 2021. URL: https://bugguide.net/node/view/8267#classification (visited on 09/01/2022).

[Mrozek, 2021] Dariusz Mrozek, Rafał Grny, Anna Wachowicz, and Bożena Małysiak-Mrozek. "Edge-Based Detection of Varroosis in Beehives with IoT Devices with Embedded and TPU-Accelerated Machine Learning". *Applied Sciences* 11.22 (2021).

[Ostiguy, 2000] Nancy Ostiguy and Diana Sammataro. "A simplified technique for counting *Varroa jacobsoni* Oud. on sticky boards". *Apidologie* 31.6 (2000), pp. 707–716.

[Pailloux, 2018] Eloi Pailloux. *30 % de mortalité pour les abeilles sur l'hiver 2018-19*. 2018. URL: https://campagnesetenvironnement.fr/abeilles-30-de-mortalite-lors-de-lhiver-2017-18/. (visited on: 26.08.2020).

[Paris,] CMM Mines Paris. *COURS DE MORPHOLOGIE MATHEMATIQUE*.

[Pascal Monasse, 2022] Kimia Nadjahi Pascal Monasse. *Décrivez efficacement les features détectées avec SIFT*. 2022. URL: https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5053196-decrivez-efficacement-les-features-detectees-avec-sift (visited on 09/02/2022).

[Patel, 2013] Omprakash Patel, Yogendra Maravi, and Sanjeev Sharma. "A Comparative Study of Histogram Equalization Based Image Enhancement Techniques for Brightness Preservation and Contrast Enhancement". *Signal Image Processing : An International Journal* 4 (2013).

[Pierred, 2018] Pierred. *Le comptage des varroas. Pourquoi? Quand? Comment?* 2018. URL: https://asapistra.fr/?q=node/1241. (visited on: 01.09.2022).

[Qiu, 2020] Han Qiu, Yuchen Ma, Zeming Li, Songtao Liu, and Jian Sun. "BorderDet: Border Feature for Dense Object Detection". *European Conference on Computer Vision – ECCV*. 2020.

[Ramirez, 2012] Melvin Ramirez, Juan P. Prendas, Carlos M. Travieso, Rafael Calderon, and Oscar Salas. "Detection of the mite Varroa destructor in honey bee cells by video sequence processing". *2012 IEEE 16th International Conference on Intelligent Engineering Systems (INES)*. Lisbon, Portugal: IEEE, 2012, pp. 103–108.

[Redmon, 2016] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[Reid, 2004] Brendan Reid. *Varroa Mite (Varroa destructor)*. 2004. URL: http://www.columbia.edu/itc/cerc/danoff-burg/invasion_bio/inv_spp_summ/varroa_destructor.html#Taxonomy (visited on 09/01/2022).

[Ren, 2015] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015.

[Research, 2022] Mid-Atlantic Apiculture Research and Extension Consortium. *The Colony and Its Organization*. 2022. URL: https://canr.udel.edu/maarec/honey-bee-biology/the-colony-and-its-organization/ (visited on 09/01/2022).

[Rhoné, 2015] Fanny Rhoné. "L'abeille à travers champs : quelles interactions entre Apis mellifera L et le paysage agricole (Gers 32) ? : le rôle des structures paysagères ligneuses dans l'apport de ressources trophiques et leurs répercussions sur les traits d'histoire de vie des colonies". Theses. Université Toulouse le Mirail - Toulouse II, 2015.

[Rodriguez, 2018] Ivan F. Rodriguez, Remi Megret, Edgar Acuna, Jose L. Agosto-Rivera, and Tugrul Giray. "Recognition of Pollen-Bearing Bees from Video Using Convolutional Neural Network". *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Lake Tahoe, NV: IEEE, 2018, pp. 314–322.

[Ronneberger, 2015] O. Ronneberger, P.Fischer, and T. Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Vol. 9351. LNCS. Springer, 2015, pp. 234–241.

[Sahoo, 1988] Prasanna Sahoo, Sasan Soltani, and Andrew Wong. "A Survey of Thresholding Techniques". *Computer Vision, Graphics, and Image Processing* 41 (1988), pp. 233–260.

[Schurischuster, 2018] Stefan Schurischuster, Beatriz Remeseiro, Petia Radeva, and Martin Kampel. "A Preliminary Study of Image Analysis for Parasite Detection on Honey Bees". *Image Analysis and Recognition*. Ed. by Aurélio Campilho, Fakhri Karray, and Bart ter Haar Romeny. Vol. 10882. Cham: Springer International Publishing, 2018, pp. 465–473.

[Schurischuster, 2016] Stefan Schurischuster, Sebastian Zambanini, Martin Kampel, and Benjamin Lamp. "Sensor Study for Monitoring Varroa Mites on Honey Bees (Apis mellifera)" (2016), p. 4.

[Sipos, 2021] Tamás Sipos, Tamás Donkó, Ildikó Jócsák, and Sándor Keszthelyi. "Study of Morphological Features in Pre-Imaginal Honey Bee Impaired by Varroa destructor by Means of Computer Tomography". *Insects* (2021).

[Struye, 1994] M. H. Struye, H. J. Mortier, G. Arnold, C. Miniggio, and R. Borneck. "Microprocessor-controlled monitoring of honeybee flight activity at the hive entrance". *Apidologie* 25.4 (1994), pp. 384–395.

[Tensorflow, 2022] Tensorflow. *Image segmentation*. 2022. URL: https://www.tensorflow.org/tutorials/images/segmentation (visited on 08/21/2022).

[Tian, 2019] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. "FCOS: Fully Convolutional One-Stage Object Detection". *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019.

[Tiwari, 2018] Astha Tiwari. "A deep learning approach to recognizing bees in video analysis of bee traffic". PhD thesis. 2018.

[Tu, 2016] Gang Jun Tu, Mikkel Kragh Hansen, Per Kryger, and Peter Ahrendt. "Automatic behaviour analysis system for honeybees using computer vision". *Computers and Electronics in Agriculture* 122 (2016), pp. 10–18.

[Vacavant, 2012] Antoine Vacavant, Thierry Chateau, Alexis Wilhelm, and Laurent Lequievre. "A Benchmark Dataset for Outdoor Foreground/Background Extraction". *ACCV Workshops*. 2012.

[Voudiotis, 2022] George Voudiotis, Anna Moraiti, and Sotirios Kontogiannis. "Deep Learning Beehive Monitoring System for Early Detection of the Varroa Mite". *Signals* 3.3 (2022), pp. 506–523.

[Wei, 2020] Fangyun Wei, Xiao Sun, Hongyang Li, Jingdong Wang, and Stephen Lin. "Point-Set Anchors for Object Detection, Instance Segmentation and Pose Estimation". *European Conference on Computer Vision – ECCV*. 2020.

[Xi, 2020] Yue Xi, Jiangbin Zheng, Xiangjian He, Wenjing Jia, Hanhui Li, Yefan Xie, et al. "Beyond context: Exploring semantic similarity for small object detection in crowded scenes". *Pattern Recognition Letters* (2020).

[Xie, 2018] Weidi Xie, J. Alison Noble, and Andrew Zisserman. "Microscopy cell counting and detection with fully convolutional regression networks". *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization* 6.3 (2018), pp. 283–292. eprint: https://doi.org/10.1080/21681163.2016.1149104.

[Zhang, 2019] Kevin Zhang, Feng Xiong, Peize Sun, Li Hu, Boxun Li, and Gang Yu. *Double Anchor R-CNN for Human Detection in a Crowd*. 2019.

[Zhang, 2018] Shifeng Zhang, Longyin Wen, Xiao Bian, Zhen Lei, and Stan Z. Li. "Occlusion-Aware R-CNN: Detecting Pedestrians in a Crowd". *Computer Vision – ECCV*. Springer International Publishing, 2018.

[Zhao, 2018] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. "Object Detection With Deep Learning: A Review". *IEEE Transactions on Neural Networks and Learning Systems* 30 (2018), pp. 3212–3232.

[Zhou, 2019a] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. *Objects as Points*. 2019.

[Zhou, 2019b] Xingyi Zhou, Jiacheng Zhuo, and Philipp Krähenbühl. "Bottom-Up Object Detection by Grouping Extreme and Center Points". *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

RÉSUMÉ

Dans cette thèse, nous abordons deux problèmes essentiels lors de la détection d'objets en traitement d'image : 1) maintenir une bonne précision d'objets détectés, essentiellement dans le cas d'images denses et 2) détecter des objets potentiellement imbriqués dans les premiers. Ces deux problématiques existent particulièrement dans le domaine apicole. En effet, un apiculteur doit veiller au niveau d'infestation de son rucher par le parasite varroa qui s'installe sur le dos des abeilles. Les méthodes de traitement d'images par ordinateur peuvent être utilisées pour automatiser cette tâche et la rendre plus précise.

L'apprentissage profond est à l'origine de progrès dans de nombreux domaines liés à la technologie, notamment en reconnaissance faciale et vocale, et dans le traitement automatisé du langage et des images. Nous avons appliqué et adapté ces techniques à nos problématiques. Nos travaux sont basés notamment sur le réseau de neurones Faster R-CNN. Nous proposons une extension de cette architecture pour améliorer la précision de la détection des objets imbriqués dans les scènes denses. Notamment, nous proposons d'ajouter à ce réseau des branches qui détectent les objets à partir de leurs coins. Cela permet de retrouver les objets partiellement cachés. Afin de détecter les objets potentiellement imbriqués, nous nous sommes basés sur l'architecture Mask R-CNN, une extension de Faster R-CNN, dédiée à la segmentation d'objets. Nous ajoutons une branche au Faster R-CNN pour segmenter les objets internes.

Nos expériences sont basées sur un ensemble d'images de scènes denses d'abeilles, contenant des abeilles et des varroas annotés, que nous avons constitué. Les abeilles et les varroas sont les objets imbriqués à détecter. Nos résultats montrent une amélioration de +10% de la précision de détection par rapport à l'approche standard. Après avoir testé différents manières d'extraire les informations pour la segmentation, nous avons défini un seul réseau de neurones capable de détecter les deux types d'objets imbriqués sans diminution de la précision par rapport à deux réseaux de neurones séparés.

Nous montrons également que notre architecture de réseaux de neurones étendue améliore la détection d'objets dans d'autres domaines tels que celui de la détection des personnes dans les scènes denses, où nous avons des résultats qui atteignent +3% par rapport l'approche standard.

MOTS CLÉS

détection d'objets, scènes denses, objets imbriqués, varroa, abeille, apprentissage profond, réseau de neurones, Faster R-CNN, Mask R-CNN, segmentation d'objets

ABSTRACT

In this thesis, we address two essential problems related to the detection of objects and image processing: 1) maintaining a good precision of detected objects, essentially in the case of dense images and 2) detecting objects potentially nested in the first. These two problems exist particularly in the beekeeping sector. Indeed, a beekeeper must ensure the level of infestation of his apiary by the varroa parasite which settles on the backs of bees. Computer image processing methods can be used to automate this task and make it more accurate.

Deep learning is driving advances in many technology-related areas, including facial and voice recognition, automated language and image processing. We have applied and adapted these techniques to our problems. Our work is based in particular on the Faster R-CNN neural network. We propose an extension of this architecture to improve the accuracy of nested object detection in dense scenes.

To improve object detection in dense scenes, we propose to add branches to the Faster R-CNN neural network that detect objects from their corners. This allows to find partially hidden objects. In order to detect potentially nested objects, we rely on the Mask R-CNN architecture, an extension of Faster R-CNN dedicated to object segmentation. We add a branch to the Faster R-CNN to segment internal objects.

Our experiments are based on a set of images of dense bee scenes, containing annotated bees and varroa mites, that we have put together. Bees and varroa mites are nested objects to detect. Our results show an improvement that reaches +10% in detection accuracy compared to the standard approach. After testing different ways to extract information for segmentation, we ended up with a single neural network capable of detecting two nested objects without decreasing accuracy compared to two separate neural networks.

We also show that our extended neural network architecture improves object detection in other areas such as people detection in dense scenes, where we have results that reach +3% compared to the standard approach.

KEYWORDS

object detection, dense scenes, nested objects, varroa, bee, deep learning, neural network, Faster R-CNN, Mask R-CNN, object segmentation