Département Génie Informatique

ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

Mémoire de Projet de Fin d'Études

Présenté en vue de l'obtention du titre

D'Ingénieur d'État

Spécialité : Intelligence Artificielle et Génie Informatique

❋ ❋BELLAL Mohamed Yassine❋ ❋

Titre :

La surveillance automatique de la ruche en utilisant les techniques de vision par ordinateur

❀❀❀❀❀❀

Organisme d'Accueil : LIRMM

Soutenu le : 0 4 /07/2022 devant le jury composé

- MOUTACHAOUIK HICHAM : Professeure à l'ENSAM de Casablanca Président
- HAIN Mustapha : Professeur à l'ENSAM de Casablanca Examinateur
- CHERGUI Adil : Professeur à l'ENSAM de Casablanca Encadrant
- Pascal Poncelet : Professeur à l'université de Montpellier Parrain

Dédicaces

A celle qui a attendu avec impatience les fruits de sa bonne éducation, à ma mère.

A celui qui m'a indiqué la bonne voie en me rappelant que la volonté fait toujours les grands hommes, à mon père.

En témoignage de ma profonde gratitude et de mon incontestable reconnaissance envers vous.

A mon frère et ma sœur auprès desquels j'ai trouvé un soutien sans complaisance et des encouragements sincères, à Omar et Zhour.

A toute ma famille et mes amis qui ont fait preuve de soutien et qui m'ont donné une motivation sans prix.

A tous mes chers amis avec qui j'ai passé des instants inoubliables.

A mon école l'ENSAM.

A toutes les personnes qui ont cru en mes succès.“

-Mohamed Yassine

Remerciements

Avant d'entamer la description de mon expérience professionnelle, il m'est nécessaire d'adresser quelques expressions de remerciements et reconnaissance à toutes les personnes ayant contribué dans la réussite de ce travail.

Je remercie en premier lieu mon encadrant pédagogique, M. Adil CHERGUI, pour m'avoir fait l'honneur de m'encadrer tout au long de mon stage de fin d'études et de m'avoir guidé dans mon projet, on le remercie vivement de la qualité de sa formation, de sa disponibilité et de son encadrement exceptionnel.

C'est avec un profond respect que j'exprime ma gratitude envers mon maître de stage, Pascal Poncelet, qui m'a accompagné tout au long de mon stage. M. Poncelet m'a donné de nouvelles perspectives, appris de nouvelles méthodes et techniques. J'ai beaucoup appris sur la vision par ordinateur et le machine learning au cours de ce stage, et les discussions autour du projet et des résultats étaient toujours enrichissants.

Que les membres du jury trouvent ici l'expression de notre reconnaissance pour avoir accepté d'évaluer notre travail.

Mes remerciements s'adressent aussi à l'ensemble du corps professoral et administratif de l'école national des arts et métiers pour l'effort qu'ils fournissent afin de nous garantir une bonne formation et à l'équipe administrative et technique pour tous les services offerts.

Finalement, que tous celles et ceux qui ont contribué de près ou de loin à l'accomplissement de ce travail trouvent l'expression de nos remerciements les plus chaleureux.

Résumé

Surveiller une ruche en suivant le nombre d'abeilles et leur évolution tout au long de la journée est primordiale pour connaître la santé de la ruche, ainsi que pour prédire la qualité du miel. À cette fin, les portes avec des capteurs infrarouges sont généralement utilisés pour calculer le nombre d'abeilles entrant et sortant de la ruche.

Dans ce projet, nous avons utilisé une approche basée sur les techniques de computer vision pour détecter les abeilles et les compter. Les résultats obtenus à l'aide de différentes approches sont comparés et la meilleure approche a été utilisée avec une API pour faciliter la consommation de données par les utilisateurs.

Keyword: computer vision, machine learning, deep learning, yolo , django, docker, deepsort, labgen.

Abstract

Monitoring a honeycomb by tracking the number of bees and their change throughout the day is important to know the health of the beehive, as well as to predict the quality of honey.

For this purpose, infrared sensor gates are usually used to calculate the number of bees entering and leaving the hive.

In our projects, we used an approach based on computer vision to detect bees and count them. The obtained results using different approaches are compared with each other, and the best approach was used with an API to facilitate data consumption by users.

Keyword: computer vision, machine learning, deep learning, yolo , django, docker, deepsort, labgen.

Liste des acronymes

CCD colony collapse disorder

CNN Convulutional Neural Network

DT Decision Tree

FC Fully connected layer

FPN Feature Pyramid Networks

IoU Intersection over union

KNN k-nearest neighbors

mAP Mean average precision

MVC Model–view–controller

PCA Principal component analysis

R-CNN Regions with convolutional neural networks

RoI Region of Interest

RPN Region Proposal Network

SAT Self-Adversarial Training

SGD Stochastic gradient descent

SVM Support vector machines

t-SNE t-distributed stochastic neighbor embedding

YOLO You Only Look once

liste des figures

liste des tableaux

Contents

1 Introduction générale

Les abeilles jouent un rôle important dans l'agriculture, parce qu'elles sont responsables d'une grande partie de la pollinisation des végétaux dans le monde. Ce qui a pour effet de maintenir la fructification des plantes. La diminution du nombre d'abeilles dans une région, il est souvent accompagné par une baisse de rendement agricole. Par conséquence, elles affectent négativement les courbes de famine, notamment dans les pays émergents.

Dans ce cas, l'équilibre des différents écosystèmes se retrouverait menacé alors même que l'industrie agro-alimentaire est déjà sous tension, et que les ressources sont, elles aussi, menacées par le réchauffement climatique. Il a également été remarqué ces dernières années La disparition soudaine des abeilles et leur abandon de la ruche , ce qui pose un majeur problème pour les apiculteurs et les agriculteurs, d'où la nécessité de surveiller la ruche pour éviter et prévoir le comportement malicieux des abeilles**.**

C'est dans ce cadre que s'inscrit notre projet de fin d'études, effectué à l'université de Montpellier au sein du LIRMM (Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier) dans le cadre du projet Superbeelive, sous l'encadrement de Dr. Pascal Poncelet, le responsable de l'équipe ADVANCE et sous la supervision du Dr. Matthieu Rousset Chercheur INSERM à l'IBMM, ce stage consiste à fournir une solution pour l'automatisation de la surveillance de la ruche à travers la détection des abeilles et le suivi ainsi la reconstruction du cadre de cire pour faire une étude ultérieure sur les alvéoles. À fin de suivre en détail l'état de santé de la ruche.

Le rapport est organisé de la manière suivante, dans la première Chapitre, nous avons étudié et comparé entre les algorithmes qu'ils existent actuellement.

Dans la deuxième partie, nous expliquerons brièvement le rôle des outils que nous avons utilisés, car cela est nécessaire à la compréhension du reste du rapport.

La troisième partie quant à elle sera consacrée à expliquer la démarche quand a suivi ainsi que le détaille des travaux quand a effectué tout au long de ce stage.

En fin, on a parlé sur les outils utilisés pour l'organisation de projet et pour conclus, nous ferons le bilan du projet et les choses à faire dans la suite de mon stage.

2 Cadre de stage et présentation du projet

2.1 Organisme d'accueil

2.1.1 LIRMM

Fondé en 1992, le Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier (LIRMM[1]) est une unité mixte de recherche, dépendant conjointement de l'Université de Montpellier (UM) et du Centre National de la Recherche Scientifique (CNRS).

Le LIRMM comporte trois départements : informatique, microélectronique et robotique. Chaque département est lui-même organisé en équipes.

Département d'informatique

Le département informatique s'étend du domaine des mathématiques à celui de la recherche appliquée : algorithmique des graphes, bio-informatique, cryptographie, réseaux, bases de données et systèmes d'information (intégration de données, fouille de données, maintien de la cohérence), génie logiciel (langages de programmation, objets, composants, modèles), intelligence artificielle (apprentissage, contraintes, représentation des connaissances, systèmes multi-agents), et interaction homme-machine (langage naturel, visualisation, web sémantique et e-learning).

Département de microélectronique

Le département de microélectronique est spécialisé dans la recherche de solutions innovantes pour embarquer, dans des systèmes électroniques intégrés, toujours plus d'intelligence et de technologies émergentes afin d'améliorer la qualité, la fiabilité, l'adaptabilité, l'efficacité (notamment énergétique) et la sécurité de ces systèmes. La plupart de ses activités trouvent des applications dans le domaine large des objets communicants pour l'environnement, y compris les environnements difficiles (spatial, radiatif, haute température) et le vivant.

Département de robotique

Le département de robotique développe de nouveaux systèmes robotiques et des outils fondamentaux associés avec pour objectif de les amener jusqu'à la valorisation et le transfert industriel. Cette politique scientifique s'inscrit également dans une démarche qui vise à répondre, pour un certain nombre de travaux, à des problèmes sociétaux, économiques et environnementaux. Le département mène ainsi des activités de recherche appliquées à l'industrie manufacturière, la santé, l'environnement et l'homme dans son environnement quotidien.

Les missions du LIRMM sont de former des chercheurs, de produire des connaissances (à travers des publications d'articles scientifiques à portée internationale), des objets matériels et logiciels prototypes, de l'activité économique (partenariat industriel, création d'entreprises innovantes), et de l'animation scientifique à l'échelle nationale et internationale.

Lors de ce stage je faisais partie du département informatique du LIRMM au sein des équipes de recherche ADVANSE. Les travaux de l'équipe ADVANSE (ADVanced Analytics for data SciencE) s'inscrivent dans le domaine de l'analyse des grandes bases de données afin d'en extraire de nouvelles connaissances : fouille de données, visualisation analytique et apprentissage automatique.

[1] https://www.lirmm.fr/lirmm-en/

2.1.2 IBMM

l'Institut des Biomolécules Max Mousseron (IBMM[2] ou UMR5247 CNRS-Université Montpellier 1-Université Montpellier 2), dont le directeur est le Professeur Pascal Dumy, est un établissement de recherche de dimension internationale et l'un des 4 Instituts fondateurs du Pôle Chimie Balard de Montpellier.

Les activités de recherche sont centrées sur les biomolécules essentielles comme les lipides, les nucléosides, les nucléotides et acides nucléiques, les peptides et protéines, les glycosides, les biopolymères, les molécules prébiotiques et les molécules fluorées.

Les programmes de recherche menés autour des biomolécules concernent leur conception, leur synthèse et leur pharmacologie. Ces biomolécules et leurs familles de composés sont utilisées comme molécules de base en pharmacologie pour l'étude des mécanismes physiologiques et pathologiques qui leur sont associés, elles sont également les molécules essentielles précurseurs des futurs médicaments. Les activités de recherche de l'IBMM se situent à l'interface de la chimie et de la biologie, elles visent à étudier et à comprendre les mécanismes d'action des biomolécules et le traitement des pathologies humaines et animales (infectieuses, cardio-vasculaires, dégénératives, cancer...) avec des applications dans la médecine moléculaire de demain. Parallèlement, les applications des biomolécules couvrent de vastes domaines tels que la cosmétologie, l'agroalimentaire, l'industrie vétérinaire et l'agrochimie respectueuse de l'environnement et s'inscrivant dans un cadre de développement durable (chimie verte).

Les forces de l'IBMM résident dans la diversité des compétences et dans la complémentarité des différentes équipes qui le composent, dans la reconnaissance internationale des équipes, dans les nombreuses collaborations académiques et industrielles qu'il développe et dans ses actions de valorisation, d'innovation et de transfert de technologie. L'IBMM est également un centre d'attractivité pour les futurs jeunes chercheurs grâce aux diverses possibilités de formations et de stages qu'il offre sur la base de ses capacités d'expertises et de son savoir-faire en terme de conception, analyse et découverte des futurs médicaments qui font l'originalité du site montpelliérain dans le domaine des biomolécules.

2.1.3 Rôle au sein de l'équipe

Mon rôle et mes objectifs au sein de l'équipe Advance est de mettre en place un modèle de machine learning. qui permet de compter et localiser les abeilles en temps réel pour utiliser ces données en deuxième étape pour contrôler la santé de la ruche.

2.2 Présentation du projet SuperBeeLive

Le projet SuperBeeLive résulte d'une collaboration entre plusieurs laboratoires de Montpellier, à savoir l'IBMM, le LIRMM. Coordonné par Matthieu Rousset, enseignant-chercheur à l'IBMM, il s'articule autour de l'étude des effets des changements environnementaux, dus à l'activité humaine, sur les abeilles.

SuperBeeLive a un objectif d'observer une ruche combinant à la fois la diffusion (et l'enregistrement) en direct du comportement d'un superorganisme (à l'échelle individuelle - l'abeille - et à l'échelle de la colonie d'abeilles) et des mesures de plusieurs paramètres physiques (température, humidité)

[2]https://ibmm.umontpellier.fr/

grâce à un dispositif spécialement conçu basé sur l'introduction de cartes électroniques entre les peignes de cire.

Des caméras vidéo enregistrant le nid d'abeille des deux côtés, nous donneront les informations nécessaires sur les changements de comportement de la colonie. A l'origine, le dispositif permet de comprendre le lien entre le comportement du superorganisme et les paramètres physiques externes.

2.3 Présentation de sujet et Objectifs du stage

Les abeilles jouent un rôle central dans l'agriculture, car elles sont statistiquement les plus importantes responsables de la pollinisation de la plupart des plantes dans le monde et elles affectent directement le rendement agricole.

L'importance des abeilles ne se limite pas à la pollinisation, mais aussi à la production de miel, la cire d'abeille, propolis et la gelée royale de leurs ruches. Les abeilles jouent un rôle central dans l'agriculture et de plus ils produisent tant de produits utiles, c'est pour cette raison il est si important pour les apiculteurs de pouvoir gérer et surveiller la santé de leurs ruches pour un but final d'évité le phénomène de disparition les abeilles de la ruche connue scientifiquement Colony Collapse Disorder CCD, ce phénomène présente actuellement un vrai danger pour l'agriculture sur plusieurs pays.

C'est dans ce contexte, l'importance de ce projet est évidente dans la sociométrie des abeilles pour l'analyse des ruches par les biologistes et les aider à prédire les maladies avant qu'elles ne se propagent et à les traiter le plus tôt possible.

La sociométrie des abeilles consiste à détecter les abeilles ainsi que les suivit en temps réels pour savoir plus d'information sur leur mouvement.

Dans la suite du rapport, on va essayer de trouver la meilleure approche qu'on peut utiliser pour la détection des abeilles.

Les objectives de stages sont les suivants :

- Surveillance de la ruche pour prévoir l'état de santé des abeilles et éviter précocement le phénomène de collapse colony disorder ;
- Trouver l'approche la plus adéquate à la détection de *multi-objet* avec des objets nombreux et petits d'une manière rapide pour l'utilisé en temps réels pour compter les abeilles ;
- Déployée le modèle plus performant avec un REST api pour faciliter la consommation des données aux experts ;
- Mettre en place l'algorithme de *deepsort* qui permet à suivre les abeilles en temps réels en affectant à chaque abeille un identifiant unique;
- La reconstruction du cadre de cire avec les algorithmes de génération d'arrière-plan pour le préparer à une classification ultérieure sur les alvéoles.

3 Gestion du projet

3.1 ClickUp - Diagramme de Gantt

ClickUp est également un logiciel de gestion de projet en ligne. Il rassemble plusieurs outils de gestion tels que des listes de tâches, des tableaux de notes, etc. Nous l'avons utilisé afin de créer un diagramme de Gantt ce qui nous permettait de respecter les délais du projet. Voici un aperçu de notre diagramme ci-dessous Fig. 1.

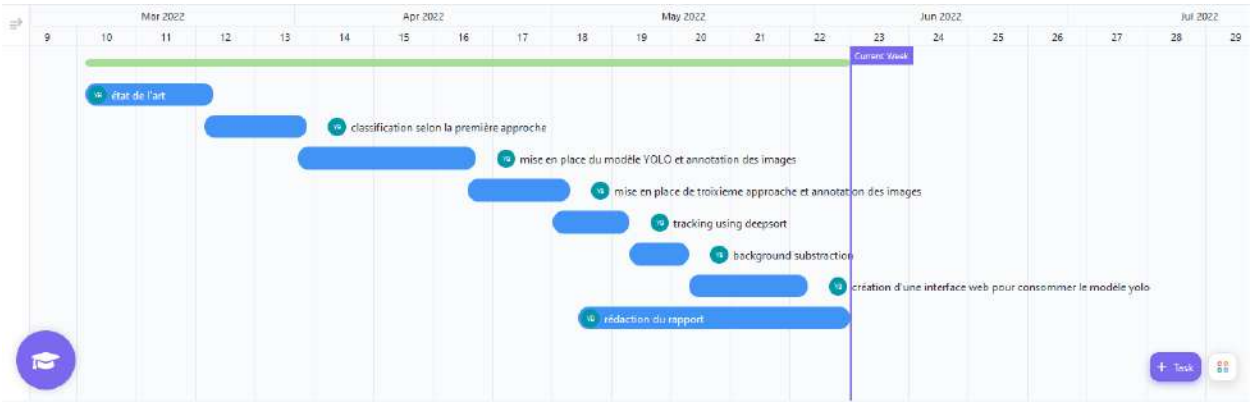


Figure 1: diagramme de Gantt

Les deux premières semaines de stage sont dédiées pour la couverture de sujet de stage et les technologies et les algorithmes de l'état de l'art qu'on peut utiliser, après nous avons entamé notre projet par le jeu de données préparé par karl , nous avons utilisé ce jeu de données pour tester la première approche pendant deux semaines après nous avons passé trois semaines pour l'étiquetage d'un nouveau jeu de données et l'utilisé pour entrainer le modèle yolo , par la suitenous avons passé à peu près deux semaines pour étiqueter un nouveau jeu donné pour l'utilisée avec un modèle U-NET .

Plus tard nous avons passé les sept semaines suivant dans quatre taches différent, suivit des abeilles en utilisant deep sort , reconstruction du cadre de cire avec LaBgen et déploiement du modèle performent avec django et docker et en parallèle de ces différentes taches, nous avons commencé la rédaction du rapport.

3.2 Google Colab

Colaboratory, souvent raccourci en "Colab", est un produit de Google Research. Colab permet à n'importe qui d'écrire et d'exécuter du code Python par le biais d'un navigateur. C'est un environnement particulièrement adapté à l'apprentissage machine, à l'analyse de données et à l'éducation. En termes plus techniques, Colab est un service hébergé de notebooks Jupyter (document JSON) qui ne nécessite aucune configuration et permet d'accéder sans frais à des ressources informatiques.

3.3 Trello

Trello est un outil de gestion de projet en ligne. Il est utilisé afin de découper ces projets en sous-tâches. Ces sous-tâches sont assignables à des utilisateurs et sont mobiles d'une section à l'autre (en cours d'avancement, fini, etc). Nous avons choisi de diviser nos tâches en différentes catégories : les tâches à faire, celles en cours, celles terminées et celles validées par notre encadrant de projet. Nous avons fait les mêmes catégories spécifiquement pour la rédaction du rapport de projet, afin de bien définir les parties à rédiger. Nous pouvons voir un aperçu de notre planche à la figure 12 ci-dessous.

4 Étude comparative

4.1 Détection

En vision par ordinateur, le problème de prédiction de la classe et de l'emplacement des objets contenus dans une image est connu sous le nom de problème de détection d'objets. Contrairement à la classification, l'image est catégorisée quel que soit le nombre d'objets ou leur emplacement.

Avant la popularité de deep learning dans le domaine de computer vision, la détection d'objets se faisait en utilisant des caractéristiques d'apprentissage automatique artisanales, telles que shift invariant feature transform (SHIFT) [1], histogram of oriented gradients (HOG) [2] etc, en revanche , actuellement, Nombreux algorithmes basés sur deep learning ont prouvé leur utilité.

Après 2012, différents chercheurs ont introduit des nouvelles stratégies telles que les méthodes de *pooling*, *object proposals algorithm*, des nouvelles fonctions de pertes, les méthodes d'estimation de bounding box[3] avec des CNN pour améliorer la tâche de détection d'objets.

Les modèles de détection d'objets basés sur les cnn peuvent être classés en deux catégories :

(i) approche en deux étapes et (ii) approche en une étape.

Les modèles de ces deux catégories sont discutées dans les deux sous-sections suivantes.

4.1.1 Approche en deux étapes :

Dans la détection d'objets en deux étapes, la première étape génère des propositions de régions[4] sur l'image qui peuvent contenir des objets, dans la deuxième étape, ces propositions sont classées et détectées à l'aide de bounding box.

Les modèles de pointe qui entrent dans cette catégorie sont expliqués un par un.

Region based convolutional neural network

R-CNN est l'un des principaux réseaux neuronaux profonds conçus pour effectuer la détection d'objets. Le R-CNN utilise les propositions d'objets générées par la recherche sélective pour entraîner le CNN à effectuer la détection d'objets et à générer des bounding boxes potentielles à contenir des objets.

L'architecture du RCNN contient trois blocs différents comme le montre Fig. 2 . La première étape comprend le premier bloc où les auteurs ont utilisé l'algorithme de recherche sélective [3] pour générer environ 2000 propositions de régions à partir de chaque image d'entrée. Dans le deuxième bloc, en suivant l'architecture d'AlexNet [4], ils ont utilisé un CNN avec cinq couches convolutionnelles (Conv) et deux couches fully connected (FC) pour extraire un vecteur caractéristique de chaque proposition de région (region proposal). Comme CNN nécessite une image de taille fixe en entrée, les auteurs ont utilisé une déformation d'image affine [5] pour obtenir une image d'entrée de taille fixe à partir de chaque proposition de région, indépendamment de leur taille ou de leur rapport d'aspect. Ensuite, ces images déformées sont introduites dans des réseaux CNN individuels pour extraire des vecteurs caractéristiques de longueur fixe de chaque région. Le

[3]bounding box : connu aussi par boite englobante ou boite délimitation

[4]Region proposal : région avec forte probabilité de contenir des objets

troisième bloc classe chaque proposition de région à l'aide de SVM [6] spécifique à la catégorie. Les deuxième et troisième blocs constituent ensemble la deuxième étape.

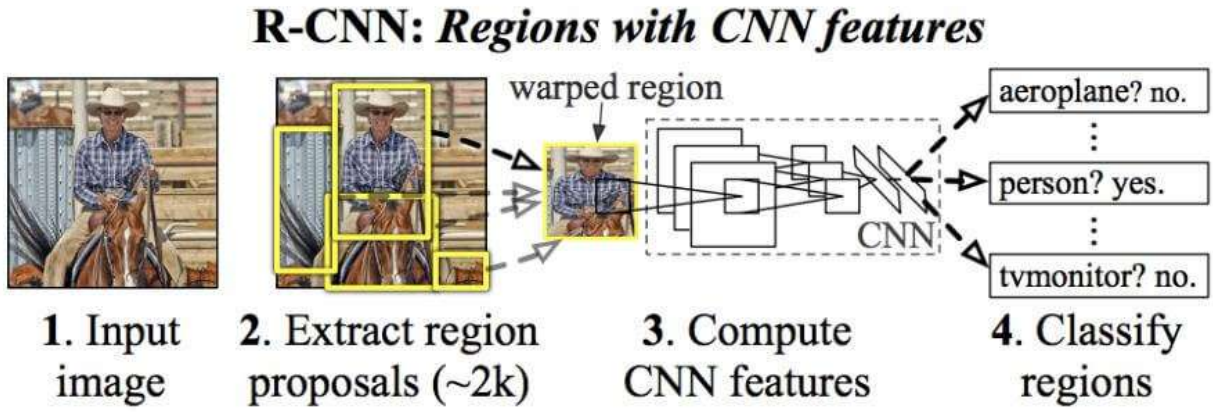


Figure 2: the flow chart of R-CNN

Détails de l'entraînement

Les auteurs de R-CNN ont pré-entraîné le CNN en utilisant le jeu de données ILSVRC-2012 [7]. Ils ont ensuite remplacé le classificateur softmax 1000-way à ImageNet par un classificateur 21-way (pour les 20 classes PASCAL VOC et une classe d'arrière-plan) et ont entraîné les paramètres CNN en utilisant SGD avec des propositions de régions déformées tirées des images de PASCAL VOC. Si le chevauchement IoU (Intersection over union) des propositions de régions avec les vraies régions correspondantes est supérieure ou égale à 0.5, ils traitent la proposition comme positive pour cette classe de boîte et le reste comme une proposition négative. Le taux d'apprentissage de SGD était de 0,001. La taille du mini-batch par itération SGD était de 128 (32 fenêtres positives de premier plan et 96 fenêtres négatives d'arrière-plan). Ils ont optimisé les SVM linéaires par classe en utilisant le standard hard négative mining [8] pour réduire la consommation de mémoire.

L'utilisation de CNN pour la détection d'objets a permis à R-CNN d'obtenir une meilleure précision dans le problème de détection, mais il y a un inconvénient majeur de ce modèle. Les CNN ont besoin d'une image d'entrée de taille fixe, mais les propositions de régions générées par le R-CNN sont arbitraires. Pour répondre aux exigences de CNN, l'échelle ou le rapport d'aspect ou l'originalité d'une image est compromis en raison du recadrage, de la déformation, des échelles prédéfinies.

Les limites du R-CNN :

- Une complexité temporelle énorme, ce qui rend le R-CNN non adapté pour les applications de la vie réelle.
- Génération imprécise des propositions de régions candidates en raison de l'absence de capacité d'apprentissage inhérente à l'algorithme de recherche sélective

Fast R-CNN

Les auteurs de l'article [9] ont proposé un réseau de neurones profonds modifié, à savoir Fast R-CNN, pour surmonter les limitations de R-CNN tels que (i) l'entraînement en plusieurs étapes

(feature extraction, réglage fin du réseau, entraînement SVM, régression par bounding box), (ii) phase d'entraînement coûteuse en termes d'espace et de temps, et aussi (iii) une détection d'objet lente.

Fast R-CNN est un algorithme de detection en une seule étape qui apprend à classer les propositions de régions et corrige leurs emplacements spatiaux ensemble. Fast R-CNN peut entraîner un réseau de détection profonde comme le VGG-16 [10], en un temps réduit neuf fois plus rapide que le R-CNN.

R-CNN utilise CNN pour chaque proposition de région générée pour la suite du processus de détection**.** Mais le R-CNN rapide prend en entrée l'image entière et un ensemble de propositions d'objets. À partir de la carte de caractéristiques CNN produite, les régions d'intérêt (RoI) sont identifiées en utilisant la méthode de recherche sélective.

Les auteurs ont ensuite utilisé une couche de mise en commun des couches de *Region of Interests pooling (RoI)* pour les remodeler en un vecteur caractéristique de longueur fixe. Ensuite, les couches FC prennent ces vecteurs de caractéristiques en entrée et transmettent la sortie à deux branches de sortie. Une branche est destinée à la classification et une autre à la régression de bounding box. Fig. 3 illustre l'architecture du R-CNN rapide.

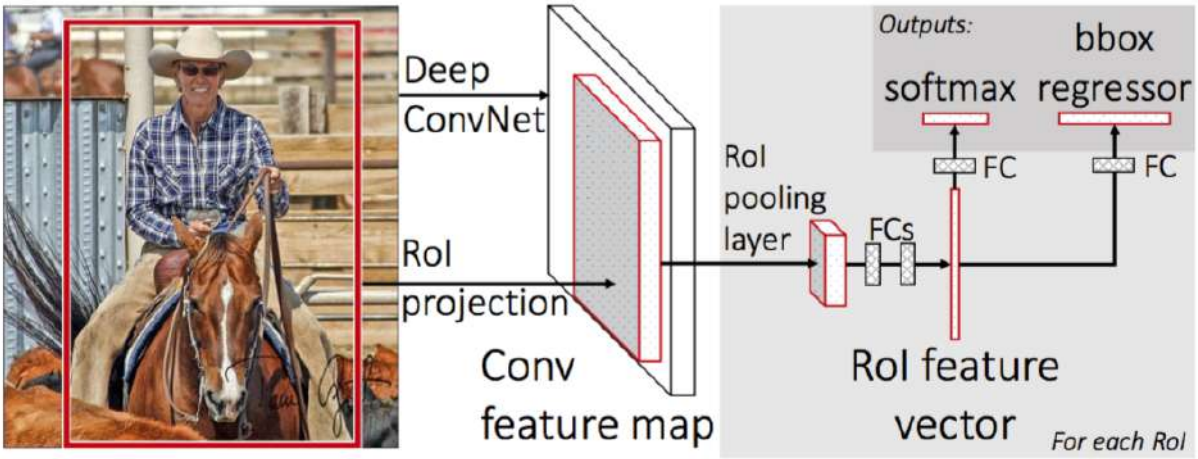


Figure 3: Architecture du Fast R-CNN

Détails de l'entraînement

Les auteurs ont expérimenté leur modèle en utilisant trois types de réseaux qui sont pré-entraînés sur le jeu de données ImageNet [11]. Trois réseaux sont CaffeNet [12], AlexNet [4] et VGG_CNN_M_1024 [13] avec une autre version VGG16 [10]. Les réseaux ont 5 couches de max-pooling au total et le nombre de couches de convolutions dans ces réseaux était compris entre cinq et treize. Lors de l'initialisation de ces réseaux pré-entraînés, les auteurs ont d'abord remplacé la dernière couche max-pooling par une couche de pooling RoI. Ensuite, la dernière couche FC et la couche softmax des réseaux ont été remplacées par deux couches de sortie sœurs pour la classification et la régression de la boîte englobante. Les réseaux sont également modifiés pour prendre deux entrées: une image d'entrée complète et une liste des ROI présentes dans ces images. Les auteurs ont formé Fast R-CNN en utilisant la descente de gradient stochastique (SGD) avec des mini-batch échantillonné de manière hiérarchique.

Tout d'abord, les auteurs ont choisi un nombre N d'échantillons d'images au hasard. Ensuite, à partir de chaque image, un nombre R/N de RoI est échantillonné pour chaque mini-batch. Lors du Fine-tuning, ils ont choisi N = 2 et R = 128 pour construire le mini-batch. Le taux d'apprentissage initial pour 30K itérations en mini-batch était de 0.001 et pour les 10K itérations suivantes en mini-batch était de 0.0001. Ils ont par ailleurs utilisé un momentum de 0,9 et une décroissance des paramètres de 0,0005. Ils ont utilisé la fonction de perte multi-task pour entraîner conjointement leur réseau pour la classification et la régression de la boîte englobante en une seule étape.

L'utilisation du CNN une seule fois pour une image entière et la procédure d'entraînement en une seule étape réduisent la complexité temporelle du R-CNN rapide à grande échelle par rapport au R-CNN précédent. De plus, l'utilisation d'une couche de Roi pooling et de quelques astuces lors de l'entraînement a permis à ce modèle d'atteindre une plus grande précision.

Limitations de Fast R-CNN :

- L'introduction de la couche de pooling RoI à bien réduit la complexité temporelle de Fast R-CNN dans une certaine mesure par rapport à R-CNN, Mais le problème de la génération de propositions de régions inexactes, dû à la capacité de non-apprentissage de l'algorithme de recherche sélective, existe aussi dans le R-CNN rapide car, comme dans le R-CNN, les propositions de régions sont sélectionnées à l'aide de l'algorithme de recherche sélective.

Faster R-CNN

Afin d'éviter la limitation de r-cnn et fast r-cnn, et de réduire la complexité temporelle et également de générer des propositions de régions précises, un réseau nommé Faster RCNN est conçu dans [14] en fusionnant Fast R-CNN et un nouveau réseau neuronal entièrement convolutif, nommément région proposal network (RPN). RPN est un réseau fully convolutional (FCN) qui prend une image de taille arbitraire en entrée et produit en sortie un ensemble de propositions d'objets candidats rectangulaires. Chaque proposition d'objet est associée à un score pour détecter si la proposition contient un objet ou non. En réalité, RPN génère non seulement des propositions de régions de haute qualité, mais peut également proposer simultanément des limites d'objets et des scores pour confirmer la présence d'objet à chaque position.

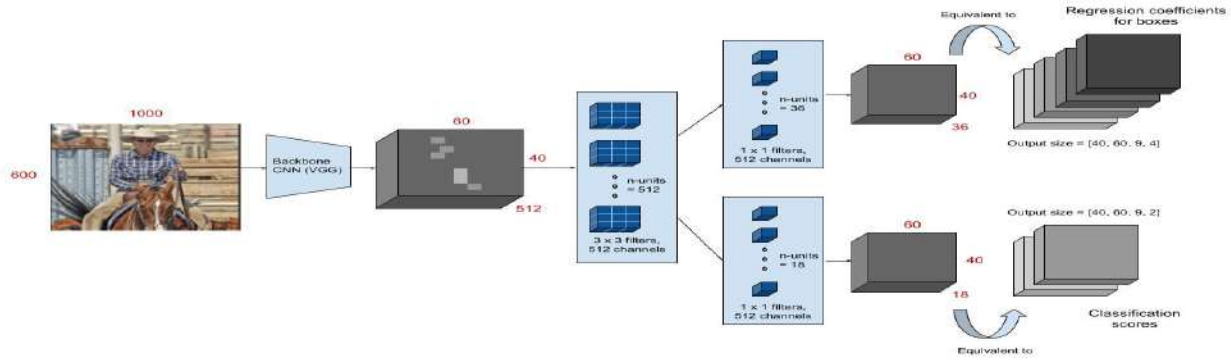


Figure 4: L'architecture du Faster R-CNN

Semblable à Fast R-CNN, l'image entière est fournie en tant qu'entrée aux couches Conv de Faster R-CNN pour produire une carte de caractéristiques convolutives. Ensuite, au lieu d'utiliser un algorithme de recherche sélective sur la carte des caractéristiques pour identifier les propositions de région, un RPN est utilisé pour prédire les propositions de région. Dans RPN, les propositions de régions détectées par emplacements de fenêtres glissantes sont appelées Anchors. Un Anchor box[5] pertinent est sélectionnée en appliquant une valeur seuil sur le score d'Objectness[6]".Anchor boxes sélectionnées et les cartes de caractéristiques calculées par le modèle CNN initial sont transmises ensemble à la couche RoI pooling pour le remodelage et la sortie de la couche de RoI pooling est introduite dans les couches FC pour la classification finale et la prédiction de la boîte englobante[7].

Détails de l'entraînement :

Le RPN est essentiellement un réseau neuronal entièrement convolutif [15] pré-formé sur l'ensemble de données ImageNet et il est affiné sur l'ensemble de données PASCAL VOC. Les propositions de région générées à partir de RPN avec des boîtes d'ancrage sont utilisées pour former le Fast R-CNN (partie ultérieure de Faster R-CNN après RPN). Pour entraîner le RPN, chaque Anchor est affectée à une étiquette de classe binaire. L'étiquette positive est attribuée à deux types d'ancres :

- Anchor avec le plus grand Intersection-over-Union (IoU) chevaucher avec ground-truth[8] box.
- Anchor qui présente un chevauchement IoU supérieur à 0,7 avec n'importe quelle ground-truth box.

Une étiquette négative est attribuée à un Anchor si sa valeur IoU est $< 0{,}3$ pour tous les ground-truth boxes.

Les auteurs ont entraîné RPN et Fast R-CNN de manière indépendante. Ils ont simplement suivi le multi-task loss de Fast R-CNN pour entraîner leur réseau. RPN est entraînable de bout en bout

[5]Anchor boxes : un ensemble de boîtes englobantes prédéfinies d'une certaine hauteur et largeur.
[6]Objectness : le score d'être un objet
[7]Boite englobante : bounding box
[8]Ground-truth : une information dont on sait qu'elle est réelle ou vraie

avec la rétro propagation et la SGD. Les auteurs ont également suivi la stratégie d'échantillonnage "image centric" du Fast R-CNN. Un mini-batch de SGD est construit avec un certain nombre de boîtes d'ancrage positives et négatives prédites à partir d'une image d'entrée. Dans chaque mini-batch, le rapport entre les ancres positives et négatives peut aller jusqu'à 1:1. Ils ont utilisé un taux d'apprentissage initial de 0.001 pour 60K en min-batches et 0.0001 pour les 20K en mini-batches suivants. Les auteurs ont utilisé 0,9 comme momentum et 0,0005 comme weight decay[9].

Les modèles de détection d'objets mentionnés précédemment ont considéré une carte de caractéristiques (feature map) à une seule échelle. De cette façon, ces modèles n'ont pas la localisation exacte des instances de l'objet, car la carte de caractéristiques de la dernière couche du CNN est invariable à l'échelle.

Limitations de Faster R-CNN:

- Le RPN est formé où toutes les anchors du mini-batch, de taille 256, sont extraites d'une seule image. Étant donné que tous les échantillons d'une même image peuvent être corrélés (c'est-à-dire que leurs caractéristiques sont similaires), le réseau peut prendre beaucoup de temps avant d'atteindre la convergence.

Mask R-CNN

Mask R-CNN étend les techniques de détection d'objets R-CNN précédentes pour aller plus loin et localiser les pixels exacts de chaque instance d'objet (segmentation d'instance [16]) au lieu de simples boîtes de délimitation . Comme ce modèle masque indépendamment chaque instance d'un objet, ils l'ont nommé Mask R-CNN . Mask R-CNN dispose du RPN exact de Faster R-CNN pour produire des propositions de région. Les auteurs ont appliqué la couche RoIAlign sur les propositions de région au lieu de la couche de RoI pooling pour aligner les entités extraites avec l'emplacement d'entrée d'un objet. Les RoI alignés sont ensuite introduits dans la dernière section du masque R-CNN pour générer trois sorties : une étiquette de classe, un décalage de boîte englobante et un masque d'objet binaire. Pour le masquage, un petit réseau neuronal FC est utilisé sur chaque RoI. L'architecture de Mask R-CNN est celle de la Fig. 5.

[9]est une technique de régularisation par l'ajout d'une petite pénalité, habituellement la norme L2 des poids

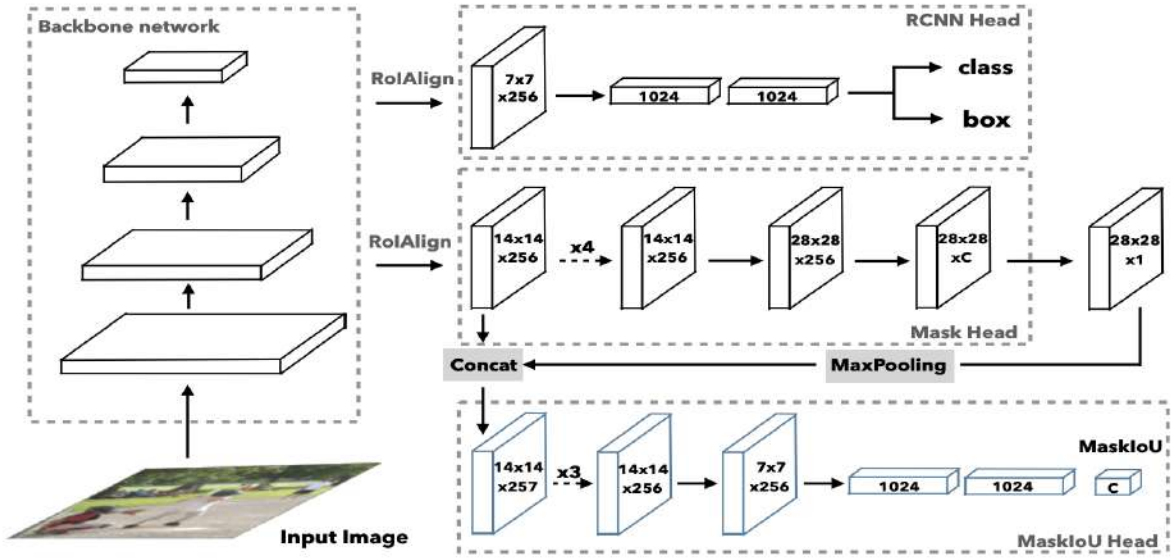


Figure 3. Network architecture of Mask Scoring R-CNN. The input image is fed into a backbone network to generate RoIs via RPN and RoI features via RoIAlign. The RCNN head and Mask head are standard components of Mask R-CNN. For predicting MaskIoU, we use the predicted mask and RoI feature as input. The MaskIoU head has 4 convolution layers (all have kernel=3 and the final one uses stride=2 for downsampling) and 3 fully connected layers (the final one outputs C classes MaskIoU.)

Figure 5: L'architecture du Mask R-CNN

Détails de l'entraînement :

Les auteurs ont utilisé les réseaux ResNet [17] et ResNeXt [18] comme CNN de base du R-CNN Faster. Le Feature Pyramid Network (FPN) [19] est utilisé avec le Faster R-CNN comme structure de base. Les régions d'intérêt (RoI) sont extraites de différents niveaux du FPN avec des échelles variables. L'utilisation de ResNet et de FPN a permis à Mask R-CNN de gagner en précision et en rapidité. Les auteurs ont défini tous les hyperparamètres de leur modèle en suivant [14]. Si l'IoU d'un RoI avec une boîte de vérité terrain est $\geq 0{,}5$ alors ce RoI est considéré comme positif sinon négatif. Mask R-CNN a également suivi un entraînement centré sur l'image. La fonction de perte multi-task de Mask R-CNN combine la fonction de perte du masque de classification, de localisation et de segmentation. La perte de masque est définie uniquement sur les positifs (RoIs). Le réseau ResNet FPN est entraîné en utilisant SGD avec une taille de mini-batch de 2 images. Chaque image est échantillonnée en N (RoIs).

Le rapport entre les résultats positifs et négatifs (RoIs) pour chaque mini-batch était de 1:3. Le taux d'apprentissage pour 160K itérations était de 0,02. Il a été diminué d'un facteur 10 pour l'itération suivante de 120K. De même, le momentum et weight decay étaient respectivement de 0,9 et 0,0001. L'entraînement du ResNeXt FPN comprend une taille de mini-batch de 1 image et un taux d'apprentissage initial de 0,01.

4.1.2 Approche en une étape

Dans l'approche à une étape, la classification et la régression sont effectuées en une seule fois en utilisant un échantillonnage régulier et dense en ce qui concerne les emplacements, les échelles et le rapport d'aspect. L'exemple le plus populaire de l'approche en une étape est décrit ci-dessous :

Yolo

Le modèle *You Only Look Once* (YOLO) [20] est un modèle de réseau à une seule étape qui prédit les probabilités de classe et les boîtes englobantes directement à partir de l'image d'entrée en utilisant un CNN simple. Le modèle divise l'image d'entrée en un nombre fixe de grilles. Chaque cellule de cette grille prédit un nombre fixe de boîtes englobantes avec un score de confiance. Le score de confiance est calculé en multipliant la probabilité de détecter l'objet avec l'IoU entre les boîtes prédites et les boîtes de ground-truth. Les boîtes de délimitation dont la probabilité de classe est supérieure à une valeur seuil sont sélectionnées et utilisées pour localiser l'objet dans l'image.

YOLO v2 utilisait une architecture profonde personnalisée connue sous le nom de darknet-19, un réseau à 19 couches à l'origine, complété par 11 couches supplémentaires pour la détection des objets. Avec une architecture à 30 couches, YOLO v2 avait souvent du mal à détecter les petits objets. Ceci était attribué à la perte de caractéristiques fines lorsque les couches ont sous-échantillonné l'entrée.. Pour remédier à ce problème, YOLO v2 a utilisé une cartographie d'identité, en concaténant les cartes de caractéristiques[10] d'une couche précédente pour capturer les caractéristiques de bas niveau.

Cependant, l'architecture de YOLO v2 manquait encore de certains des éléments les plus importants qui sont désormais essentiels dans la plupart des algorithmes de pointe. Pas de blocs résiduels, pas de skip connections et pas de upsampling. YOLO v3 intègre tout cela.

[10] carte de caractéristique : feature map

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
	Convolutional	32	1 × 1	
1×	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
	Convolutional	64	1 × 1	
2×	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
	Convolutional	128	1 × 1	
8×	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
	Convolutional	256	1 × 1	
8×	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
	Convolutional	512	1 × 1	
4×	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 6: L'architecture de yolo

Tout d'abord, YOLO v3 utilise une variante de Darknet, qui dispose à l'origine d'un réseau à 53 couches entraîné sur Imagenet. Pour la tâche de détection, 53 couches supplémentaires sont empilées dessus, ce qui nous donne une architecture sous-jacente entièrement convolutive à 106 couches pour YOLO v3. C'est la raison de la lenteur de YOLO v3 par rapport à YOLO v2. La Fig. 6 à quoi ressemble maintenant l'architecture de YOLO.

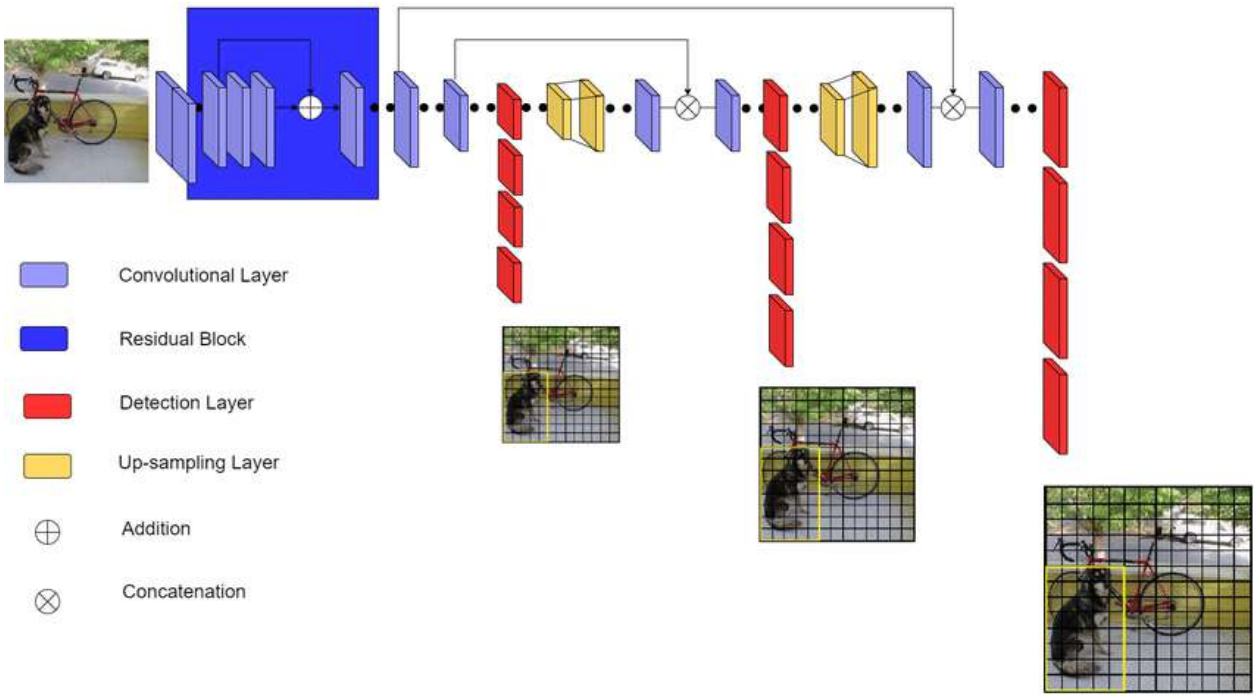


Figure 7: l'architecture de yolo v3

YOLO est un réseau entièrement convolutif et sa sortie éventuelle est générée en appliquant un noyau 1 x 1 sur une carte de caractéristiques. Dans YOLO v3, la détection est effectuée en appliquant des noyaux de détection 1 x 1 sur des cartes de caractéristiques de trois tailles différentes à trois endroits différents du réseau.

La forme du noyau de détection est 1 x 1 x (B x (5 + C)). Ici, B est le nombre de bounding boxes qu'une cellule de la carte de caractéristique peut prédire, "5" est pour les 4 attributs de bounding boxes et une confiance d'objet, et C est le nombre de classes.

Dans YOLO v3 entraîné sur COCO, B = 3 et C = 80, la taille du noyau est donc de 1 x 1 x 255. La carte de caractéristiques produite par ce noyau a une hauteur et une largeur identiques à la carte de caractéristiques précédente et aux attributs de détection le long de la profondeur comme décrit ci-dessus Fig. 7.

Le stride du réseau, ou d'une couche, est défini comme le ratio par lequel il sous-échantillonne l'entrée. Dans les exemples suivants, nous supposerons que nous avons une image d'entrée de taille 416 x 416. YOLO v3 fait des prédictions à trois échelles, qui sont précisément données par le sous-échantillonnage des dimensions de l'image d'entrée par 32, 16 et 8 respectivement.

La première détection est effectuée par la 82ème couche. Pour les 81 premières couches, l'image est sous-échantillonnée par le réseau, de sorte que la 81e couche a un stride de 32. Si nous avons une image de 416 x 416, la carte de caractéristiques résultante sera de taille 13 x 13. Une détection est effectuée ici en utilisant le noyau de détection 1 x 1, ce qui nous donne une carte de caractéristiques de détection de 13 x 13 x 255.

Ensuite, la carte de caractéristiques de la couche 79 est soumise à quelques couches de convolution avant d'être échantillonnée de 2x pour atteindre des dimensions de 26 x 26.Cette carte de caractéristiques est ensuite concaténée en profondeur avec la carte de caractéristiques de la couche

61. Ensuite, les cartes de caractéristiques combinées sont à nouveau soumises à quelques couches de convolution 1 x 1 pour fusionner les caractéristiques de la couche précédente (61). Ensuite, la deuxième détection est effectuée par la 94e couche, ce qui donne une carte de caractéristiques de détection de 26 x 26 x 255.

Une procédure similaire est à nouveau suivie, où la carte de caractéristiques de la couche 91 est soumise à quelques couches conv avant d'être concaténées en profondeur avec une carte de caractéristiques de la couche 36. Comme précédemment, quelques couches convolutionnelles 1 x 1 suivent pour fusionner les informations de la couche précédente (36). Nous faisons la dernière des 3 couches à la 106ème couche, ce qui donne une carte de taille 52 x 52 x 255.

Les détections à différentes couches permettent de résoudre le problème de la détection des petits objets. Les couches sur-échantillonnées concaténées avec les couches précédentes permettent de préserver les caractéristiques à grain fin qui aident à détecter les petits objets.

La couche 13 x 13 est responsable de la détection des gros objets, tandis que la couche 52 x 52 détecte les objets plus petits, la couche 26 x 26 détectant les objets moyens.

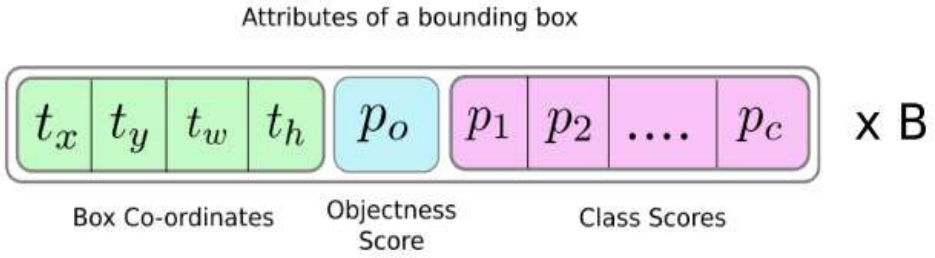


Figure 8: Le variable de sortie de yolo

Le variable de sortie de modèle Yolo est construit par trois parties, premièrement les cordonnés de boite englobante, puis le score d'Objectness et enfin par les scores de chaque classe, la Fig. 8 montre le variable de sortie de Yolo.

Yolo v4

YOLO version 4 est conçue en s'inspirant de plusieurs méthodes de détection d'objets de type Bag-of-Freebies et Bag-of-Specials. La méthode Bag-of-Freebies augmente le temps d'inférence et le coût d'apprentissage du détecteur, mais améliore sa précision, tandis que la méthode Bag-of-Specials augmente le coût d'inférence de la méthode dans une certaine mesure, mais améliore sa précision.

En dehors de ces modifications, les autres améliorations apportées au modèle YOLO version 4 sont la sélection des valeurs optimales des hyper-paramètres en utilisant des algorithmes génétiques, introduction de méthodes d'enrichissement des données telles que Self-Adversarial Training (SAT) et Mosaic, modification de méthodes existantes telles que Cross mini-Batch Normalization, Spatial Attention Module, etc

4.2 Comparaison

Comme nous avons examiné des modèles de détection d'objets de deux types différents (à deux étapes et à une étape), la comparaison des performances de ces modèles est une tâche difficile.

Dans une application réelle, la performance fait essentiellement référence à la précision et à la vitesse.

Nous avons vu que la performance d'un modèle de détection d'objets dépend de différents aspects tels que le réseau d'extracteur de caractéristiques, la résolution de l'image d'entrée, la stratégie de correspondance et le seuil IoU, le nombre de propositions ou de prédictions, l'encodage de la boîte frontière, la fonction de perte, la fonction d'optimisation, la valeur choisie pour les hyperparamètres d'apprentissage, etc.

Sur la base des articles étudie auparavant, nous avons montré les performances comparatives des différents modèles de détection d'objets sur le jeu de données PASCAL VOC dans Table 1.

La précision moyenne (mAP) permet de mesurer la précision de la détection des objets. Les deux dernières colonnes donnent un aperçu de la vitesse et de l'approche organisationnelle de ces modèles.

Selon les résultats indiquer sur Table 1 , il semble clair que yolo surpasse les autres algorithmes au niveau de mAP ainsi qu'il est plus rapide et l'algorithme le plus adapté à la détection en temps réel.

	yolov4	R-CNN	FAST R-CNN	FASTER R-CNN
pascal voc 2007	78.6	58.5	70	73.2
pascal voc 2012	73.4	53.3	68.4	70.4
Real time detection	yes	no	no	no
Number of Stage	one	tow	tow	tow

Table 1: comparaison entre yolo et R-CNN et fast r-cnn et faster r-cnn

5 Outil et Technologies

Introduction

Ce chapitre est consacré à la présentation des outils et technologies utilisés lors de la réalisation de notre projet. En commençant par Django puis Docker et finalement en présentant makesense.

5.1 Django

Django est l'un des frameworks web basés sur Python les plus utilisés. Il y a de nombreuses raisons pour lesquelles il a été choisi pour développer l'api. En recherchant ses attributs, les principaux que nous rencontrons sont la rapidité, l'évolutivité et la maturité avec multiples composants intégrés qui facilitent le développement ainsi qu'il est compatible avec les scripts de python.

Le principe principal utilisé par Django est "Don't Repeat Yourself" (DRY). Il est largement utilisé, en raison de la réutilisation du code de Django, qui minimise les répétitions et permet d'en faire le plus possible avec peu de lignes de code. Cela vient également du fait que Django est créé sur l'architecture MVC, ce qui encourage une réutilisation efficace du code.

Ainsi, comme nous venons de le voir, il est basé sur le paradigme MVC, mais avec un petit changement. Le terme MVC est changé en TV qui signifie : Model – Template - View. Le modèle Django n'a pas changé la fonctionnalité. Il fonctionne toujours et traite les opérations avec la base de données. Le template est le composant qui affiche le contenu à l'utilisateur.

De l'autre côté, la vue est plus qu'un simple remplacement du contrôleur. Elle communique avec le modèle pour obtenir des données de la base de données, et après les avoir obtenues, elle formate les données, crée un objet de réponse HTTP et le renvoie au client .

Ainsi, la vue est le décideur sur quelles données doivent être renvoyées au template. Les vues dans Django peuvent être créées comme une méthode ou une classe Python et chaque vue a son propre template à partir duquel elle reçoit des entrées et renvoie des données.

Malgré ces attributs principaux, Django possède, à mon avis, d'autres caractéristiques qui le rendent préférable à d'autres frameworks web. Les paragraphes suivants, en décrivent quelques-unes. Certaines d'entre elles :

Développement rapide : Il permet aux développeurs d'être en mesure de démarrer et de terminer un projet Django en peu de temps. Les performances ne sont pas affectées par la vitesse de développement d'une application. Vitesse à laquelle une application peut être développée.

Le système de template : Il est lié au principe DRY, car les templates utilisent l'héritage, ce qui évite la répétition de code redondant.

Génération simple de la base de données et des tables : Après avoir défini la conception de la base de données, la relation entre les modèles et les champs à inclure, la commande migrate crée automatiquement toutes les tables qui n'existaient pas auparavant. Elle fournit également une variété d'options pour le programmeur afin de définir les modèles.

Scalability : La capacité de Django à évoluer de manière flexible pour répondre aux demandes et aux exigences du trafic est un grand avantage pour le framework. Cela vient du fait que ses composants sont séparés en différentes couches.

Sécurité. Django propose de nombreux moyens pour protéger les données, les comptes et les mots de passe. Il protège contre les injections SQL, le clickjacking, les attaques CSRF et bien d'autres.

Pour facilité la détection et le rendre accessible au maximum à l'utilisateur et les experts de biologie, nous avons utilisé Django pour créer une api REST qui consomme le modèle de détection plus performant

5.2 Docker

Docker est une plateforme ouverte pour le développement, l'expédition et l'exécution d'applications. Docker vous permet de séparer vos applications de votre infrastructure afin que vous puissiez livrer des logiciels rapidement. Avec Docker, vous pouvez gérer votre infrastructure de la même façon que vos applications. En tirant parti des méthodologies de Docker pour expédier, tester et déployer rapidement le code, vous pouvez réduire considérablement le délai entre l'écriture du code et son exécution en production.

Docker permet d'empaqueter et d'exécuter une application dans un environnement faiblement isolé appelé conteneur. L'isolation et la sécurité vous permettent d'exécuter plusieurs conteneurs simultanément sur un hôte donné. Les conteneurs sont légers et contiennent tout ce qui est nécessaire à l'exécution de l'application. Vous n'avez donc pas besoin de vous fier à ce qui est actuellement installé sur l'hôte. Vous pouvez facilement partager des conteneurs pendant que vous travaillez, et être sûr que toutes les personnes avec qui vous partagez obtiennent le même conteneur qui fonctionne de la même manière.

Docker fournit des outils et une plateforme pour gérer le cycle de vie des conteneurs :

- Développez l'application et ses composants de support à l'aide de conteneurs.
- Le conteneur devient l'unité de distribution et de test de l' application.
- Lorsque nous sommes prêts, déployez votre application dans votre environnement de production, en tant que conteneur ou service orchestré. Cela fonctionne de la même façon, que votre environnement de production soit un data center local, cloud provider ou un hybride des deux.

En plus des caractéristiques précédentes, Docker présente d'autres avantages qui rendent l'utilisation de cette technologie préférable.

- Livraison rapide et cohérente de vos applications : Docker rationalise le cycle de vie du développement en permettant aux développeurs de travailler dans des environnements standardisés en utilisant des conteneurs locaux qui fournissent vos applications et services. Les conteneurs sont parfaits pour les flux de travail continuous integration and continuous delivery (CI/CD).
- Déploiement et mise à l'échelle réactifs : La plateforme de Docker, basée sur les conteneurs, permet de workload hautement portables. Les conteneurs Docker peuvent fonctionner sur l'ordinateur portable local d'un développeur, sur des machines physiques ou virtuelles dans un datacenter, sur un cloud provider ou dans une environnements hybride.
- Exécution de plus de charges de travail sur le même matériel : Docker est léger et rapide. Il constitue une alternative viable et rentable aux machines virtuelles basées sur un hyperviseur.

Vous pouvez ainsi utiliser une plus grande partie de votre capacité de calcul pour atteindre les objectifs techniques de client.

Le déploiement du projet dans un autre environnement après la fin de la phase de développement est une tâche complexe, de même qu'il faut beaucoup de temps pour installer toutes les dépendances avec des versions compatibles, Pour contourner ces problèmes, nous avons utilisé docker pour conteneuriser le projet.

5.3 Makesense

makesense.ai est un outil en ligne gratuit pour étiqueter des photos. Grâce à l'utilisation d'un navigateur, il ne nécessite aucune installation compliquée - il suffit de visiter le site Web et vous êtes prêt à partir. Le système d'exploitation que vous utilisez n'a pas d'importance non plus - nous faisons de notre mieux pour être vraiment multiplateforme. Il est parfait pour les petits projets d'apprentissage profond en vision par ordinateur, car il rend le processus de préparation d'un ensemble de données beaucoup plus facile et rapide. Les étiquettes préparées peuvent être téléchargées dans l'un des multiples formats pris en charge. L'application a été écrite en TypeScript et repose sur le duo React/Redux.

Nous nous sommes servis de cet utile pour l'étiquetage des abeilles et la génération des fichiers d'annotation sous la forme de YOLO

Conclusion

Dans ce chapitre, nous avons présenté les outils utilisés lors de réalisation de notre projet, ainsi que la raison pour laquelle nous avons précisément choisi ces technologies plutôt que d'autres.

Dans le chapitre suivant, nous présentons la démarche générale à suivre pour la réalisation et mise en place d'un système de détection en temps réels, ainsi que le suivi des abeilles et la reconstruction du cadre de cire.

6 Réalisations

Introduction

Nous allons consacrer ce chapitre présenter la démarche suivie de la réalisation de notre projet. En commençant par la détection et l'utilisation de trois approches différentes à fin de déployer le modèle performent, puis nous allons traiter l'algorithme utilisé pour la suivie des abeilles ainsi que la méthode utilisée pour la reconstruction de cadre de cire.

La démarche suivie

Dans ce projet, nous avons utilisé trois approches différentes pour la détection des abeilles, et suite à une étude comparative, nous avons convenu que YOLO est l'approche la plus appropriée à notre problème, puis nous avons créé une API REST avec Django pour consommer le modèle rapidement et de manière simple pour tous les utilisateurs de ce projet, de plus nous avons mis en place l'algorithme deepsort pour le suivi des abeilles, enfin nous avons utilisé LaBgen pour reconstruire le cadre de cire en éliminant les abeilles, Fig. 9 suivante résume le travail effectué.

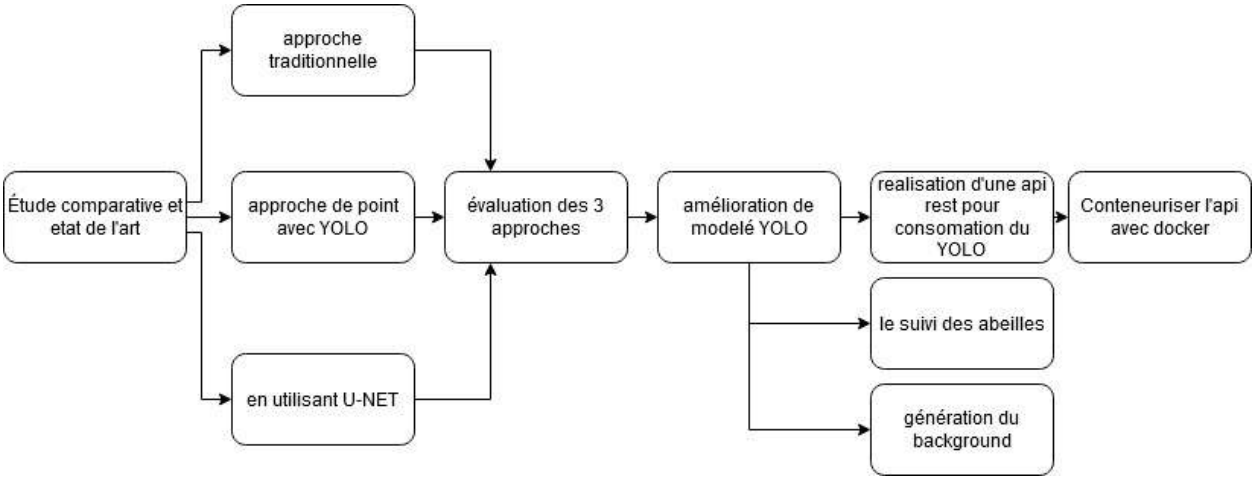


Figure 9: résumé de la démarche utilisé

6.1 Source des données

La ruche de l'étude est composée de distincts modules reliés, chacun consistant en un cadre portant deux fondations en cire symétriques. Ce dispositif permet d'insérer entre les deux fondations en cire une carte électronique portant une matrice de micro-capteurs dédiés à l'enregistrement des mesures de matériaux, avec notamment des capteurs de vibrations et des paramètres physiologiques de la colonie, avec des capteurs de température, etc.. Un duo de caméras, en noir et blanc et infrarouge, placées face à chaque module, permet d'étudier les alvéoles, le comportement des abeilles et la charge parasitaire des colonies. Trois caméras grand angle sont placées à l'extérieur de la ruche pour surveiller la prédation des frelons et la charge pollinique des butineurs. Voir Fig. 10

Les vidéos collectées sont stockées par la suite sur un serveur local et sur le serveur seafile de lirmm pour être accessible à distance aux différents membres d'équipe contribuant à ce projet. La vidéo est capturée par la caméra en segments d'une minute à 60 images/sec et en résolution 1920∗1080.

Les vidéos d'une minute sont enregistrées et encodées en H264[11].

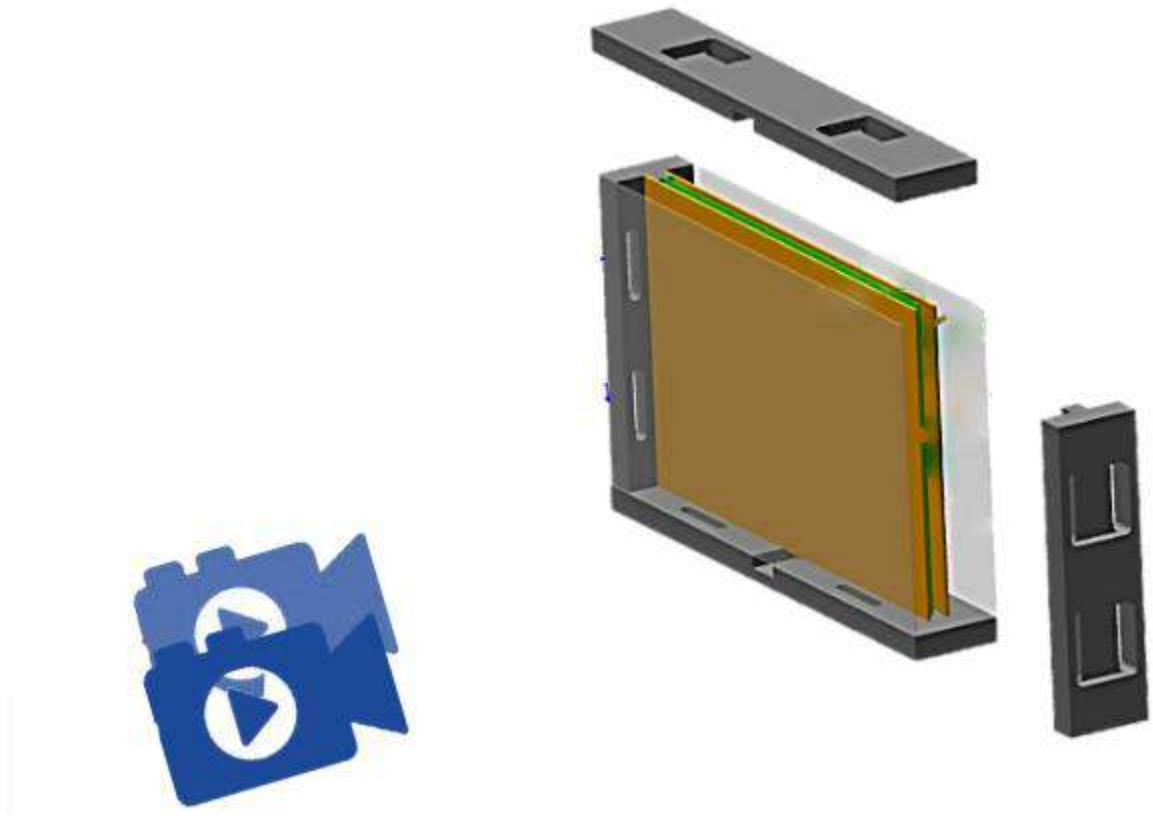


Figure 10: schéma du dispositif de surveillance de la ruche

6.2 Détection

Pour résoudre le problème de comptage des abeilles, il faut en premier temps détecter les abeilles en précisant les coordonnées de chacune, pour cette raison, nous avons testé plusieurs approches Commençant par une approche traditionnel en utilisant la technique de windows sibling puis en utilisant l'algorithme de pointe de l'état de l'art et enfin par l'utilisation d'un algorithme basé sur U-NET pour la détection des abeilles et leur orientation.

[11]H264 est une norme de compression vidéo basée sur un codage orienté bloc et compensé en mouvement.

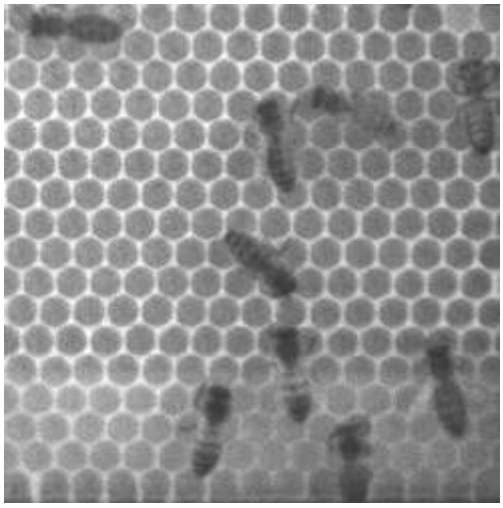


Figure 11: Image de la ruche avec un nombre d'abeilles faible

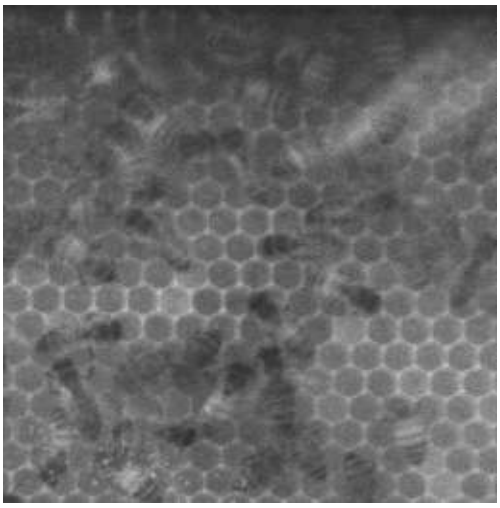


Figure 12: Image de la ruche avec un nombre d'abeilles très élevé

En observant l'image ci-dessus Fig. 12, il peut sembler clair de compter les abeilles manuellement . En prenant la deuxième image Fig. 12, on se rend compte que la tâche est bien difficile même pour un expert en biologie au vu du nombre élevé d'abeilles qu'ils bougent dans un espace étroit ainsi que la résolution médiocre de l'image.

L'utilisation de différentes approches nous a aidés à choisir le modèle le plus fiable pour le déployer en deuxième étape, dans le but de rendre la tâche facile aux experts de biologies. Qu'ils ne sont pas forcément des experts en informatique.

6.2.1 Approche basée sur windows siblings

Pour réussir la tâche de détection des abeilles, nous avons utilisé en premier lieu une approche traditionnel consiste à diviser chaque frame à des fenêtres d'une taille proche à la taille de chaque alvéole, approximativement 25∗20 px. pour le but d'appliquer des modèles de classification à chaque fenêtre à fin de détecter s'il s'agit d'une abeille ou d'une alvéole et grâce au travail du

K.Paygambar , nous avons obtenu un jeu de donnés avec 2086 image étiquetée en total d'une taille de 25∗20px, 1043 pour les abeilles et 1043 pour les alvéoles.

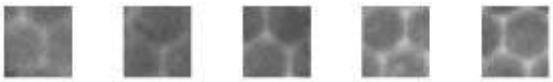


Figure 13: Images d'alveole

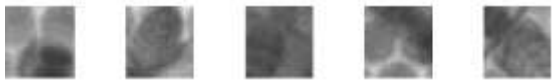


Figure 14: Images d'abeille

Les figures ci-dessous Fig. 13 et Fig. 14 montrent une partie de notre jeu de données étiqueté par K.Paygambar[12] dans le cadre de stage de m1, on peut clairement distinguer les deux classes à l'œil, mais dans d'autres cas comme illustré sur la Fig. 15, la tâche est plus complexe, car dans la phase d'étiquetage, les images avec une petite partie d'abeilles étaient considérées comme une image d'abeille, même si les alvéoles occupent la majorité de l'image

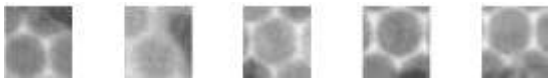


Figure 15: images classées comme abeille

Étude statistique

Sur la Fig. 17 et Fig. 17 nous pouvons voir en orange les alvéoles et en bleu les abeilles. Nous remarquons que pour la plupart, elles se distinguent assez bien avec la plupart des abeilles en haut du graphique et la plupart des alvéoles en bas pour t-sne et la plupart des abeilles en gauche du graphique et la plupart des alvéoles en droit pour *PCA* . Néanmoins, nous voyons certains points qui sont mal répartis, avec des bleus au milieu des oret inversement. Ces quelques erreurs seront alors mauvaises pour nos classifieurs qui vont alors mal classifier certaines images.

[12]K.Paygambar a effectué leur stage de m1 à LIRMM et il a préparé un jeu de donnée étiqueté des abeilles pour la première approche .

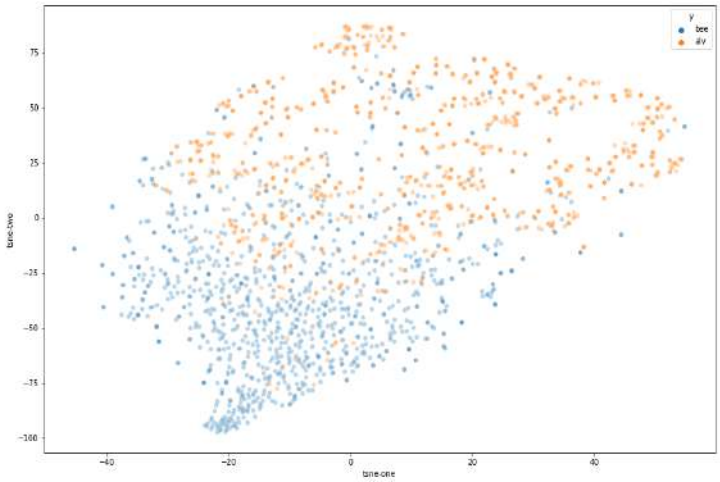


Figure 16: Images segmentées et séparées en deux catégories par t-SNE

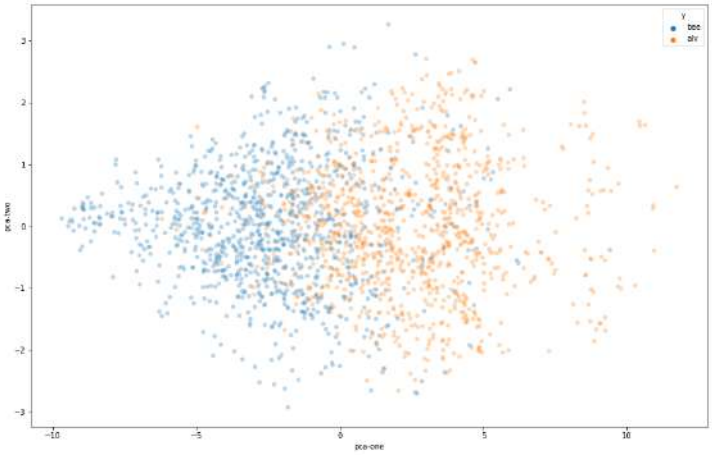


Figure 17: Images segmentées et séparées en deux catégories par PCA

Prétraitement

Dans les problèmes de classification des images, l'étape de prétraitement est très importante et peuvent influer sur les résultats, pour cette raison, il est appréciable de mentionner la démarche de chargement des images et de prétraitement.

Nous avons chargé les images en Grayscale avec une taille de 25∗20,ce qui nous a aidé à réduire le nombre des features pour fin d'accélérer la phase d'entraînement des modèles de Machine learning ainsi pour éviter les problèmes de saturation de la RAM.

Évaluation Vu qu'on a un jeu de données équilibré, nous avons utilisé l'accuracy pour évaluer les modèles de machine learning et le k-fold avec k=10 pour faire un benchmark entre les modèles et choisir le modèle qui donne les meilleurs résultats.

K-fold cross validation

La méthode k-fold signifie que l'ensemble de données se divise en un nombre K. Il divise l'ensemble de données au point où l'ensemble de test utilise chaque pli. Comprenons le concept à l'aide de la validation croisée 3 fois ou K + 3. Dans ce scénario, la méthode divisera l'ensemble de données en trois parties. Le modèle utilise le premier pli de la première itération pour tester le modèle. Il utilise les ensembles de données restants pour former le modèle. Le deuxième pli aide à tester l'ensemble de données et d'autres supports avec le processus de formation. Le même processus se répète jusqu'à ce que l'ensemble de test utilise chaque pli parmi les trois plis.

Iteration 1 :	Validation	Apprentissage	Apprentissage
Iteration 2 :	Apprentissage	Validation	Apprentissage
Iteration 3 :	Apprentissage	Apprentissage	Validation

Table 2: schéma explicatif de la méthode k-fold

Pour confirmer notre hypothèse, nous avons utilisé diverse modèle de machine learning, que ce soit des modèles traditionnels (decision tree, Logistic Regression , svm, Gaussian,knn), des modèles d'ensemble learning (Random Forest,Gradient Boosting,AdaBoost) ou des modèles de cnn. La Fig. 18 montre l'architecture cnn utilisée.

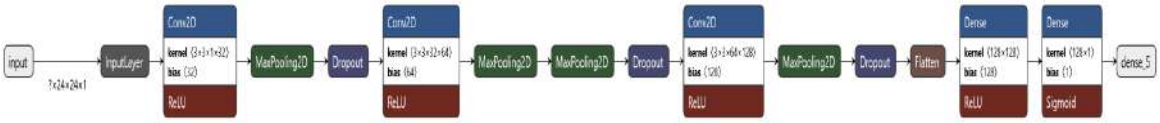


Figure 18: l'architecture utilisée pour cnn

L'algorithme SVM nous a donnée des résultats impressifs, puis on a utilisé une approche de grid search pour trouver les paramètre qui génère les meilleurs résultats, par conséquence, nous avons trouvé que svm avec le kernel rbf donne des meilleurs résultats en comparant avec les autres algorithmes. Le modèle cnn est le seul modèle qui donne des résultats proches au svm mais avec un standard déviation plus important et avec un temps de calcul plus élevé. Table 3 montre les résultats obtenus

model ML	**Decision Tree**	**Logistic Regression**	**KNN**	**Random Forest**	**Gradient Boosting**	**AdaBoost**	**GaussianNB**	**svm-rbf**	**cnn**
accuracy moyen	0.843	0.817	0.883	0.894	0.897	0.849	0.815	**0.926**	0.92
standard deviation	0.021	0.037	0.012	0.018	0.02	0.014	0.017	0.016	0.019

Table 3: résultats obtenus

6.2.2 Approche basée sur YOLO

Après une étude comparative entre les modèles de détection de point, nous sommes convenu d'utiliser en premier temps YOLO pour ces différents avantages, aussi, parce qu'il est l'algorithme le plus adapté pour un système de surveillance en temps réel.

Les différences entre les versions de yolo

Les principales différences entre l'architecture YOLOv3, YOLOv4 et YOLOv5 sont les suivantes : YOLO v3 utilise le backbone Darknet53. L'architecture YOLOv4 utilise CSPdarknet53 comme backbone et YOLOv5 utilise la structure Focus avec CSPdarknet53 comme backbone. La couche Focus est introduite pour la première fois dans YOLOv5. La couche Focus remplace les trois premières couches de l'algorithme YOLOv3 algorithme. L'avantage d'utiliser une couche Focus est de réduire la mémoire CUDA requise, réduction de la couche, augmentation de la propagation vers l'avant et de la rétro propagation.

De plus, YOLOv5 se base sur le framework PyTorch Contrairement à ses prédécesseurs qu'il se base sur darknet.

L'entrainement du modele YOLOv5

Afin d'entraîner le modèle yolo v5 et par la suite évaluer notre approche, nous avons constitué un jeu de données comportant 1200 images divisées en images d'entraînement et test et validation.

Les images sont extraites des vidéos d'une durée de moins 5min, la vidéo, il est divisé à des frames et à chaque frame nous avons pris une partie de l'image aléatoirement pour diversifier notre jeu de données ainsi pour alimenter nos modèles avec des images d'abeilles en des situations différentes.

L'étiquetage des données

L'étiquetage ou l'annotation d'images est un élément crucial de la détection d'objets. Parce qu'il influe les résultats et en même temps, c'est une tâche exigeante qui nécessite beaucoup de temps. Dans notre cas, nous avons passé plus d'une semaine pour annoter 2270 images, chacune contenant cinq abeilles en moyenne.

Nous avons utilisé makesense.io pour annoter les images avec des boîtes englobantes. Ces boîtes englobantes constituent notre vérité terrain (ground-truth).

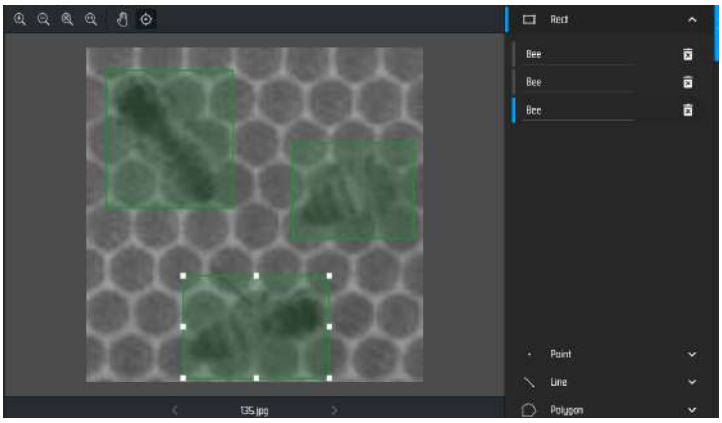


Figure 19: l'annotation avec makesense

makesense génère des fichiers d'annotation pour chaque image en format yolo contenant les coordonnées de chaque boîte englobantes avec la hauteur et la largeur de la boite de plus le type de classe. Dans notre cas, on a considéré 1 classe : "bee"

Environnement et hyperparamètre

La plupart des modèles de détection modernes nécessitent plusieurs GPU pour l'entraînement avec une grande taille de mini-batch, et le faire avec un CPU rend l'entraînement très lent et peu pratique. Pour cette raison, nous avons utilisé le serveur de calcul de LIRMM qu'il offre 4 gtx-1080 TI. Grâce à ce serveur, la phase d'entraînement était très rapide et on a pu optimiser les résultats de notre modèle rapidement.

Batch size	64
Subdivision	16
Taux d'apprentissage	0.001
Taille des entrées	416x416
pixels Momentum	0.9
Decay	0.005
Nombre de classes	2

Table 4: Hyperparamètres pour l'entraînement

Métrique d'évaluation

La mAP (mean Average Precision) représente la valeur moyenne des prévisions moyennes de chaque classe. La précision représente le pourcentage des détections correctes par rapport au nombre total de détections et le rappel représente le pourcentage des détections correctes par rapport à la vérité terrain.

Une prédiction est considérée comme vraie (TP : True Positive) si l'IoU est supérieur à un seuil donné (0.5 dans notre cas), et fausse (FP : False Positive) si c'est inférieur. La vérité terrain non détecté correspond au Faux Négatif (FN : False Negative)

$$p = \frac{TP}{TP + FP}$$

$$r = \frac{TP}{TP + FN}$$

L'AP (précision moyenne) est calculée comme la moyenne des 11 valeurs de précision pour les valeurs de rappel = {0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1}.

Metrics	*valeur*
mAP@.5	0.978
mAP@.5:.95:	0.556
P	0.965
R	0.938

Table 5: resultats pour 640 labels

D'autres graphiques montrant l'évaluation des métriques par epoch sont données dans les annexes , , , Ces graphiques nous montrent qu'à partir de 70 epoch le modèle atteint les meilleurs résultats. Les graphes Fig. 38 et Fig. 38 montrent l'évaluation de la fonction de perte par epoch des coordonnées du bounding box et de objectness il semble claire que les deux courbes converge vers le zéro pour cette raison on a obtenu des bons résultats. Et, car nous avons utilisé un seul classe, la fonction de perte de classe est toujours égale 0. Fig. 38

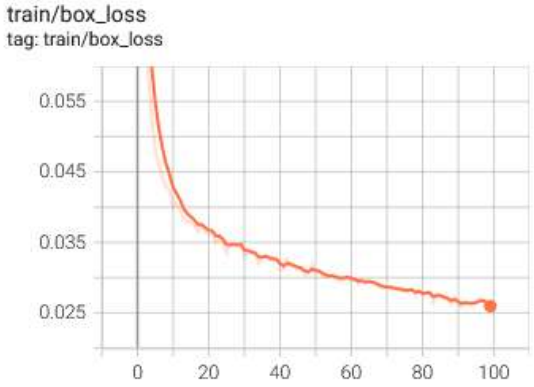


Figure 20: évolution de la fonction de perte de coordonnées de du bounding box

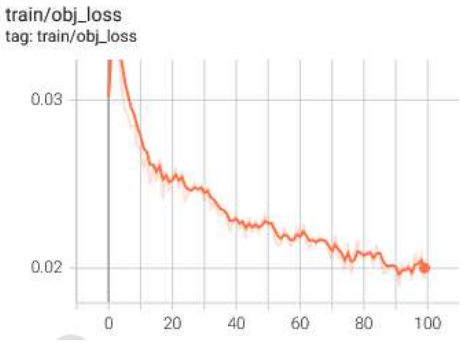


Figure 21: évolution de la fonction de perte de score d'objectness en entrainement

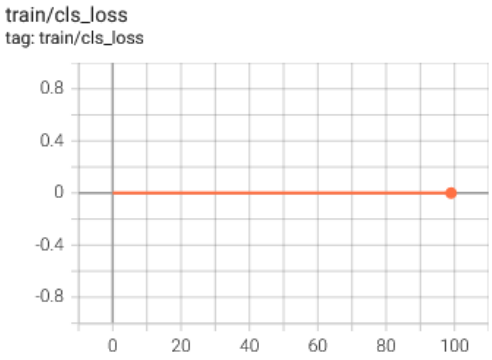


Figure 22: évolution de la fonction de perte de class

L'image suivante Fig. 23 montre un exemple de résultat avec la boite englobante, la classe et le taux de confidence

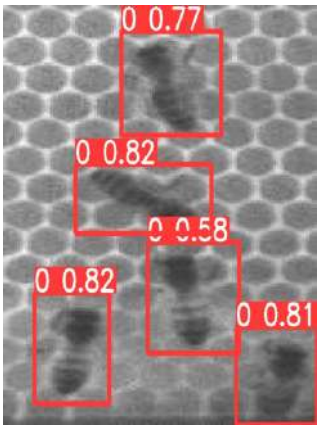


Figure 23: Résultat obtenu avec yolo

6.2.3 Approche basée sur U-NET

Cette approche se base entièrement sur l'article de [21] [21], ils ont fait un travail similaire au nôtre, mais ils ont utilisé l'architecture d'U-Net [22] [22] avec une fonction de perte qui dépend des coordonnées d'abeille ainsi que leur orientation.

Ils ont proposé une solution intégrant le réseau neuronal U-Net Fig. 24 avec un composant récurrent pour une détection précise des objets dans une séquence vidéo. Afin de permettre la reconnaissance d'instances d'objets, ils ont défini un étiquetage adapté couvrant uniquement la partie centrale de chaque objet et non adjacente aux autres objets.

Pour mieux indiquer la direction de la tête sur l'axe principal du corps, ils ont proposé une fonction de perte approximant l'angle d'orientation individuel et étendent la segmentation du fond du premier plan avec l'estimation de l'angle d'orientation des objets. En outre, la composante récurrente

du réseau exploite les informations encodées dans la séquence vidéo et améliore la précision, tout en maintenant le réseau à une fraction de la taille du U-Net original.

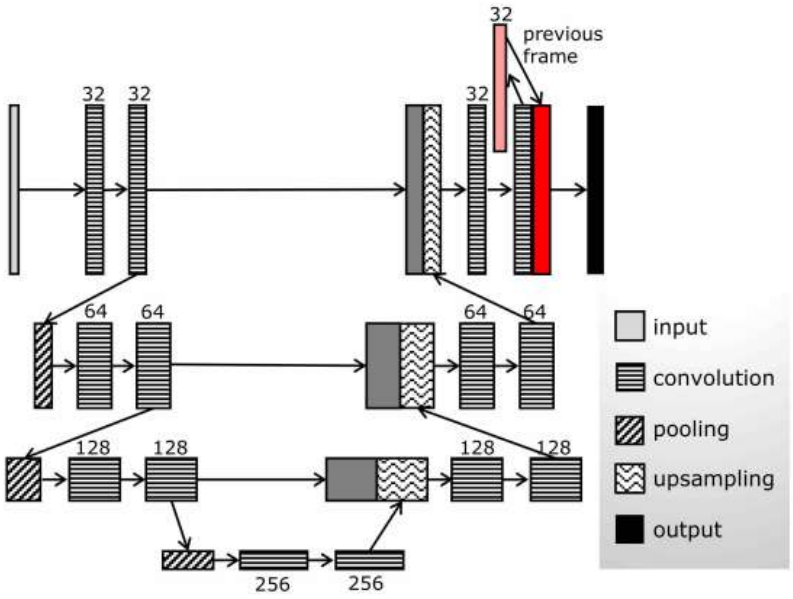


Figure 24: l'architecture d'U-Net

La première étape était d'utiliser directement les modèles entraîné sur leur jeu de donnés, mais ce process n'a pas amené des bons résultats (en annexe) pour la raison majeure que les qualités d'image sont différentes et les figures Fig. 25 et Fig. 26 montrent clairement cette différence.

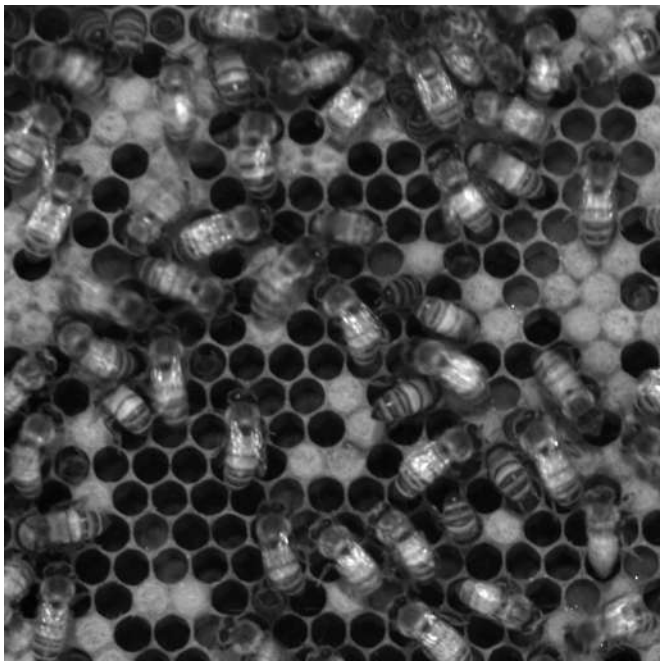


Figure 25: exemple du jeu de données utilisé en U-NET

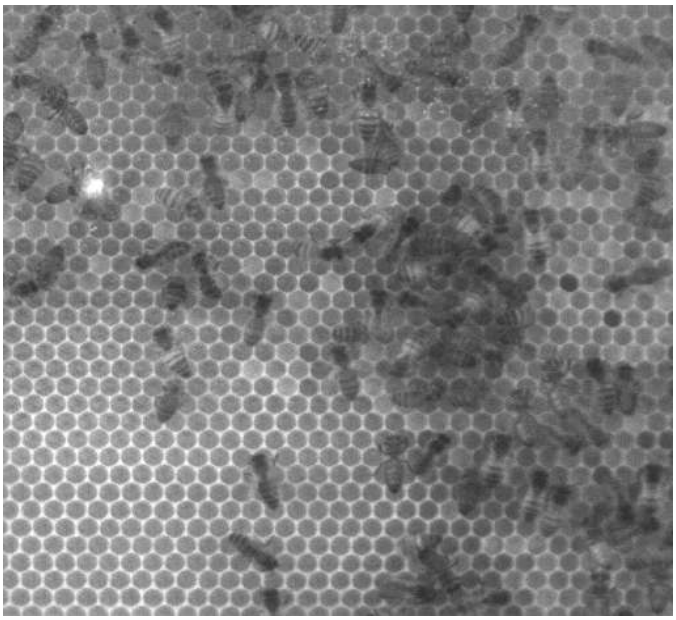


Figure 26: Exemple de notre jeu de données

Annotation et étiquetage

Pour les raisons illustrées auparavant, nous avons essayé de construire un jeu de donnée basant sur nos images. En utilisant l'outil d'annotation publié avec cet article, on a réussi à étiqueter 235 images, chacune contient 23 abeilles en moyenne et 5406 d'abeilles étiquetées en total.

Des fichiers d'annotation sont générés associés à chaque image, le fichier d'annotation comporte les données suivant pour chaque abeille (x, y, t, α) tel que x,y pour les coordonnées, t pour la classe et α pour l'orientation.

Évaluation

La figure suivante montre l'évolution de la fonction de perte pendant les 25 epoch, on peut noter que la fonction de perte n'a pas assez convergé vers le zéro, Ce qui reflète sur les résultats illustrés en figure Fig. 39

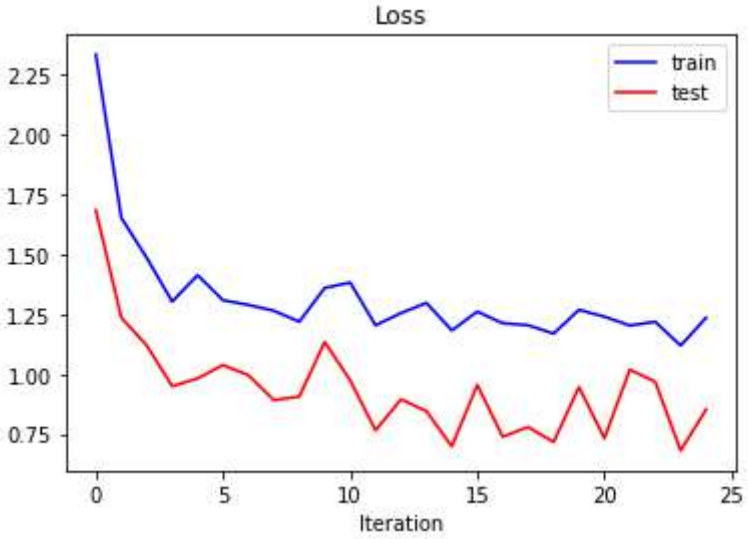


Figure 27: Évolution de fonction de perte en entrainement et en test

La Fig. 28 illustre clairement les limite et les faiblesses de cette approche. Cette dernière peut amener des meilleurs résultats par rapport les résultats obtenu précédemment, mais il faut avoir un plus large jeu de données, 150 images il n'était pas suffisante selon les résultats.

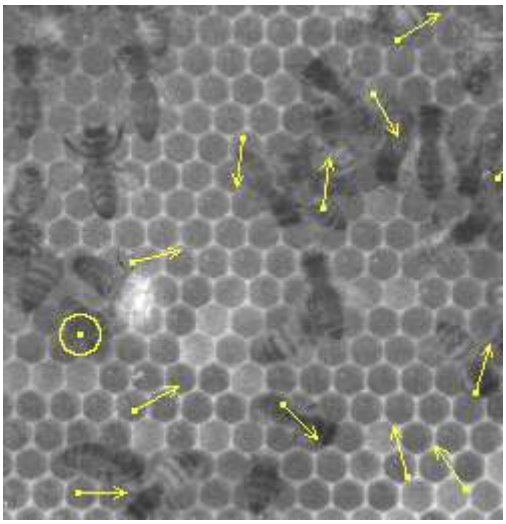


Figure 28: Exemple de résultat obtenu par U-Net

6.2.4 Discussion

Dans cette section, nous avons utilisé trois différentes approches à fin de comparer les trois est utilisé la meilleure pour déployer notre modèle avec une REST api.

En se basant sur le travail de k.Paygambar et le jeu de donnée qu'il nous a préparé on a pu appliquer la technique de windows sibling, mais le problème majeur de cette méthode, c'est qu'il peut compter une abeille plusieurs fois, car l'abeille est repartie sur plusieurs fenêtres, par conséquent on obtient

un chiffre loin du nombre réel d'abeilles qu'il apparaisse effectivement sur l'image, ainsi qu'il est long et incompatible avec les applications de traitements en temps réel.

À partir de la comparaison entre les algorithmes de l'état de l'art, on est convaincu que yolo est le modèle le plus adéquat avec notre problème, après le test il a approuvé qu'il soit capable de détecter les abeilles même avec des situations difficiles (reflet de la lumière),

La troisième approche en utilisant un modèle U-Net pour la prédiction du centre d'abeille ainsi que leur orientation, mais en raison du faible nombre d'observations étiqueté, le modèle n'a pas approuvé leur utilité et généralement il n'a pas réussi à détecter les abeilles, même les abeilles claires sur l'image. Nous avons essayé de contourner ce problème par l'utilisation du jeu de donnée de [21] pour entrainer le modèle, mais en effet de la différence de qualité d'image entre nos vidéos et les vidéos de sources de [21], on n'a pas pu régler le problème, par conséquent on est convaincu que YOLO est le modèle le plus performant dans notre cas, pour cette raison, nous avons déployé ce modèle avec sur REST api.

6.2.5 Déploiements

Le process d'installation et de préparation de l'environnement pour utiliser le modèle yolo est très compliqué ainsi qu'il prend beaucoup de temps même pour les personnes familiarisés avec ces utiles, et presque impossible pour les utilisateurs un peu loin de l'informatique, pour cette raison, nous avons déployé le modèle, pour facilité la tâche de consommation de modèles facile et rapide ainsi que disponible à partir différentes machines.

nous avons servi de Django pour les différents avantages qu'il nous offre et qu'on a déjà mentionné en chapitre des outils.

Le schéma suivant Fig. 29 illustre clairement l'architecture utilisée, le client envoie une image via une interface web, l'image passe par l'api Django pour se préparer au modèle YOLOv4 après yolo traite les images, une nouvelle image avec la boîte englobante est renvoyée au client. On a également utilisé docker pour faciliter la portabilité du projet.

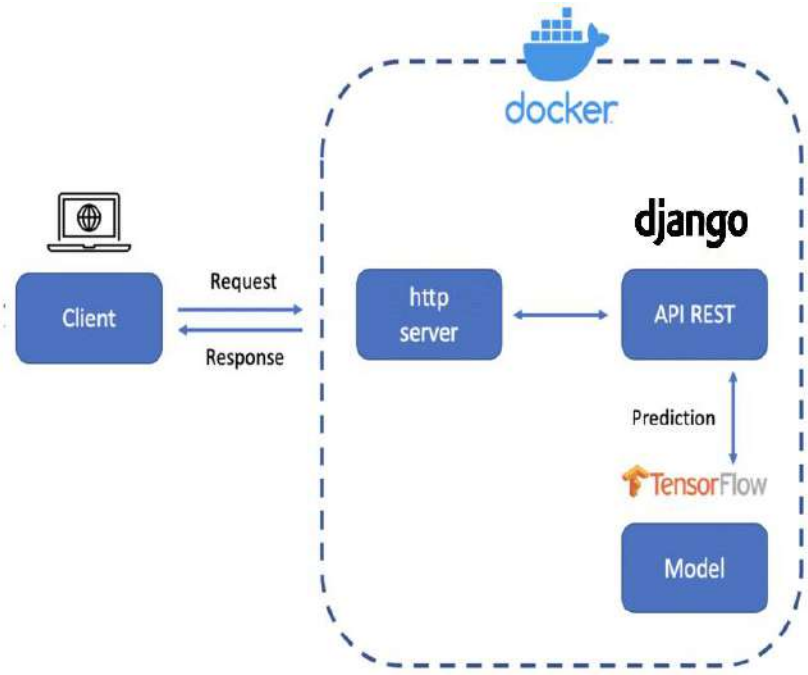


Figure 29: Schéma de déploiement du modèle yolo

La Fig. 30 illustre l'interface web qu'il consomme l'api, cette interface web est une Proof of concept pour facilité la consommation des données est extensible à extraire les données soit au format json ou à travers dessiné les boite de délimitation avec l'étiquète et taux de confiance sur l'image.

Figure 30: l'interface web qu'il consomme l'api

Figure 26: l'interface web qu'il consomme l'api

6.3 Tracking

Le suivi d'objets (*tracking*) est une tâche très importante de computer vision. Faire du tracking d'objets signifie savoir détecter et identifier univoquement chacun des objets présents dans le cadre d'une image (c'est-à-dire un frame d'une vidéo), et pouvoir les suivre sur la séquence des frames jusqu'à ce qu'ils quittent la scène. À partir de cette définition, il est clair que, à différence de la tâche de détection, il n'est pas suffisant de trouver une boite qui englobe les pixels de l'image appartenant à chaque objet, mais il est aussi nécessaire d'assigner un identifiant à chacun d'entre eux sur la base de certaines caractéristiques univoques, pour pouvoir le suivre sur la série d'images sans le confondre avec les autres, même s'il est temporairement non visible. Cela implique qu'on ne peut pas traiter chaque frame de la vidéo de manière indépendante, mais il est fondamental de tenir compte des liens spatio-temporels (distance, vitesse, ...) sur la séquence.

Afin d'obtenir une description plus détaillée sur les abeilles ainsi que leur comportement , il est nécessaire d'effectuer une étape de tracking pour obtenir la trajectoire de chaque abeille et identifier leur mouvement durant les vidéos. Pour cette raison, nous avons utilisé deep sort, on a choisi cet algorithme pour plusieurs raisons, l'un des raisons, il donne des meilleurs résultats au cas de collision, en plus il est compatible avec yolo.

6.3.1 Deep sort

Parmi les algorithmes de suivi de multiples objets, Deep SORT s'est révélé être l'une des approches les plus rapides et les plus robustes [23]. Au départ, il s'agissait de l'algorithme SORT (Simple Online and Real time Tracking) [24], qui a été développé pour avoir une approche minimaliste du suivi en ligne basé sur la détection, qui se concentrait sur l'association efficace des détections d'objets sur chaque image. Il a tiré parti de la grande réputation des réseaux neuronaux convolutifs en matière de détection précise des objets. En outre, deux méthodes classiques de prédiction de mouvement et d'association de données, l'algorithme de Hungarian [25] et le filtre de Kalman [26], ont été mises en œuvre comme composants de suivi. En raison de sa complexité modeste, SORT était 20 fois plus rapide que les autres trackers de pointe [24]. En servant YOLO comme détecteur, il a également eu de meilleures performances par rapport aux méthodes de suivi en ligne traditionnelles dans le MOT (Multiple Object Tracking) Challenge 2015 [27].

Le principal inconvénient de SORT était les occlusions et le changement de point de vue. Pour résoudre ce problème, [23] ont développé Deep SORT, qui est une version étendue de SORT (illustrée à la Fig. 31).

Dans Deep SORT, au lieu de s'appuyer uniquement sur des métriques basées sur le mouvement pour l'association des données, nous avons également intégré une métrique profonde basée sur l'apparence, dérivée du réseau de neurones convolutifs. Ce changement a permis d'obtenir une plus grande robustesse face à la collusion, au changement de point de vue et à l'utilisation d'une caméra non stationnaire pour des changements d'identité moins fréquents.

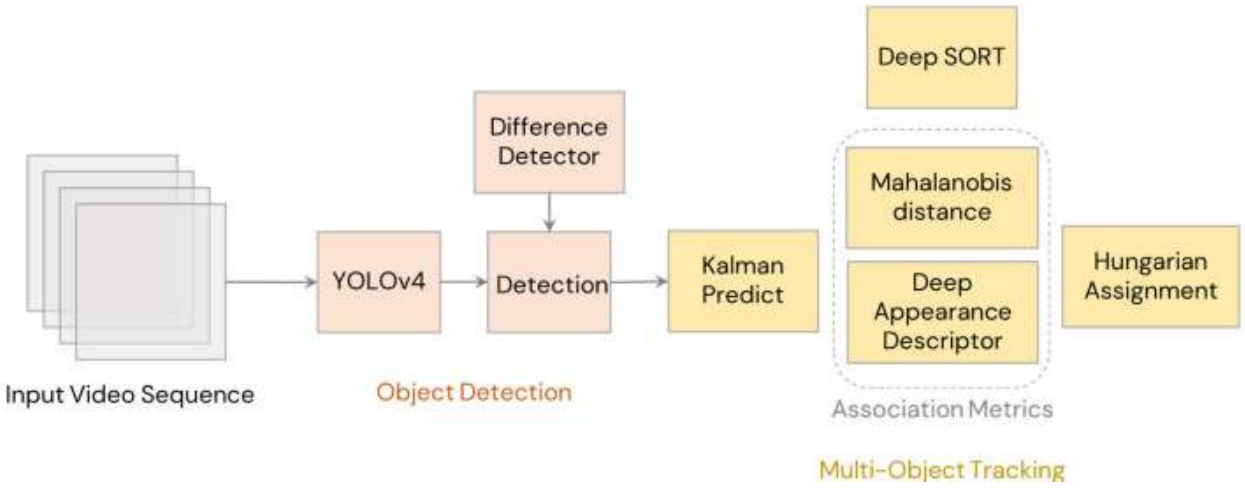


Figure 31: l'architecture de deepsort

Grâce au travail de mikel-brostrom qu'il a pu combiner l'algorithme de deepsort avec yolo et aussi en employant le modèle qu'on a entrainés de yolo on a réussi à affecter chaque abeille à un identifiant pour différencier aux autres abeilles, cette vidéo montre clairement les résultats obtenus.

Actuellement, on peut obtenir le trajet de chaque abeille, ces informations sont très utiles pour les experts de biologie pour avoir une idée sur l'état de santé de chaque abeille ainsi que l'état de santé de toute la ruche par la suite.

Remarque Dans les cas où les abeilles se croisent, l'algorithme de deepsort crée parfois un nouvel identifiant.

6.4 Background Subtraction

L'objectif final de ce projet est de pouvoir mettre en place un pipeline fiable permettant de différencier les différents types des alvéoles constituant la ruche selon leur contenu , ainsi que compter le nombre et la variation de chaque type avec le temps. Afin d'atteindre cet objectif, la première étape consiste à reconstruire le cadre en supprimant les éléments mobiles. Dans ce cas, les abeilles sont les seules qui bougent.

Pour cette raison, nous avons appuyé sur Labgen-p [28] [28] pour reconstruire le cadre de cire, nous avons adopté cet algorithme, car elle a achevé les meilleures performances lors de la compétition du Scene Background Modeling and Initialization (SBMI) organisé en 2015, et du IEEE Scene Background Modeling Contest (SBMC) organisé en 2016. LaBGen combine un filtre médian temporel pixel par pixel et un mécanisme de sélection de patch basé sur détection de mouvement. Pour détecter le mouvement, un algorithme de soustraction d'arrière-plan décide, pour chaque image, quels pixels appartiennent à l'arrière-plan.

Le Table 6 ci-dessous montre les paramètres utilisés avec LaBgen

Algorithme	Subsense
S	3
N	12
P	150

Table 6: Paramètres utilisés avec LaBgen

Tel que P pour le nombre de pass et N pour le nombre de lignes et colonnes utilise pour spatial area

L'image Fig. 32 est le résultat d'une vidéo d'une minute, vous pouvez noter qu'il y a encore des abeilles, Cela s'est produit, car les abeilles n'ont pas bougé durant les séquences de la vidéo

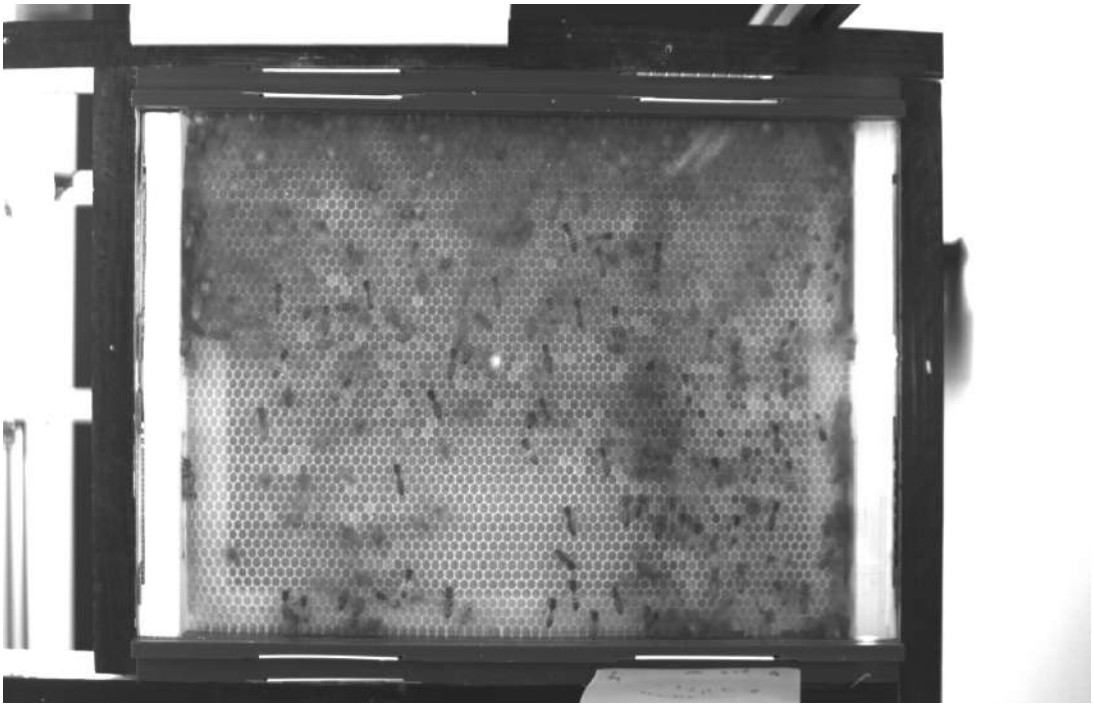


Figure 32: resultat du background substraction

Conclusion

Ce chapitre a présenté la démarche utilisée pour choisir l'approche la plus adéquate à notre contexte pour la détection et le suivi des abeilles. Et grâce à ce travaille on a pu aider les experts à construire une base solide pour surveiller la ruche à fin de prévoir leur santé.

7 Conclusion générale et perspective

Ce stage s'inscrit dans le cadre de l'étude de sociométrie d'une ruche d'abeilles, nous nous sommes servis de techniques de computer vision à fin de comprendre le comportement d'abeilles. Nous avons abordé la tache de comptage des abeilles, on appuyait sur trois approches différentes ,windows siblings, YOLO et U-net à la fin, nous avons containérisé le modèle de yolo avec REST api, car il a été l'approche plus consistante et plus performante, et pour offre plus de donnée au expert sur les mouvements des abeilles, nous avons utilisé l'algorithme de deepsort pour la suivie des abeilles en temps réel finalement nous avons reconstruit le cadre de cire pour but à étudier les alvéoles en deuxièmes étapes à l'aide de LaBgen.

Ce stage m'a permis de découvrir l'univers de la recherche scientifique sur ses aspects les plus importants. Elle m'a permis aussi de familiariser avec les concepts de computer vision.

Perspective

- Étiquetage des nouvelles images spécifiant les différents types d'abeilles, en utilisant la sortie de yolo qu'il détecte à ce moment les abeilles sans différencier entre leurs types.
- Déploiement du modèle de deepsort avec REST api pour la consommation des données d'agrégations comme (nombre d'abeilles qu'ils ont bougées - nombre d'abeilles qu'ils sont restées dans leur position pendant la durée de la vidéo - la distance parcourir par la rêne et les ouvriers)
- L'utilisation de cadre de cire que nous avons reconstituée grâce à l'algorithme d'extraction de l'arrière-plan labgen, pour la détection des différents types d'alvéoles (alvéoles avec **Pollen**, alvéoles avec du miel ...) et l'utilisation de ces informations pour suivre l'évolution de la ruche en fonction de temps pour but de prédire la santé des abeilles de cette ruche et évite la disparition des abeilles.

Annex

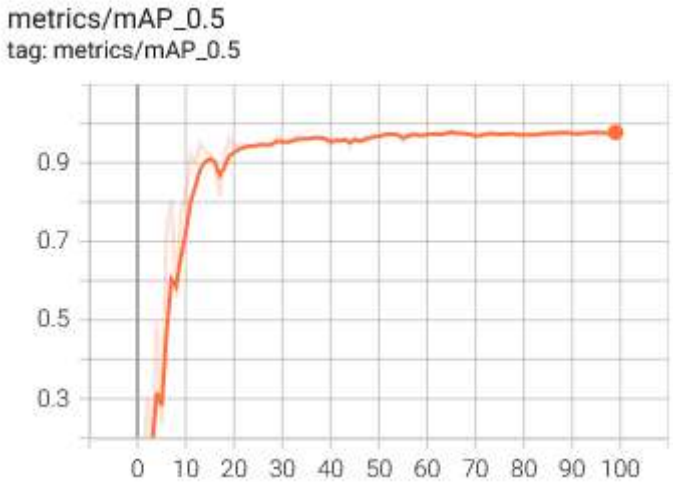


Figure 33: evolution de mAP 0.5 par epoch

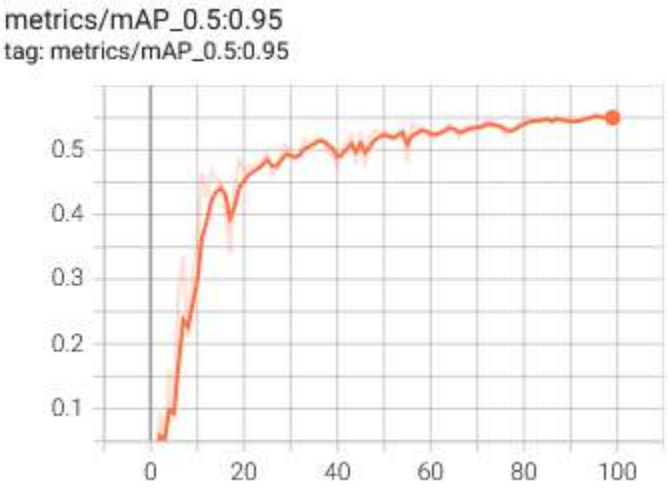


Figure 34: evolution de mAP 0.5:0.95

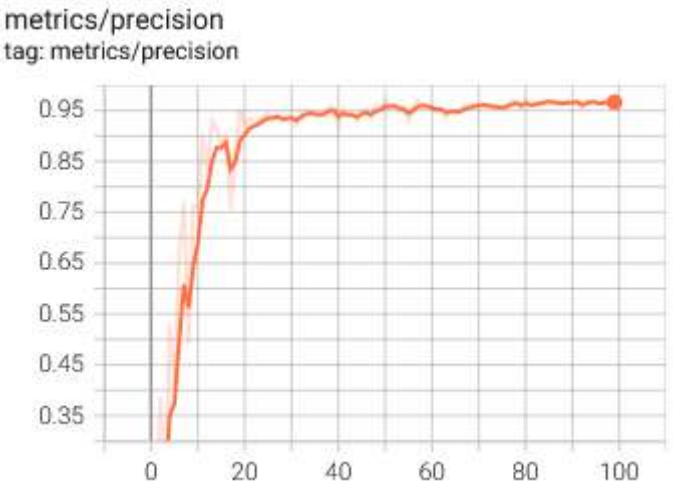


Figure 35: evolution de precision

Figure 36: evolution de recall

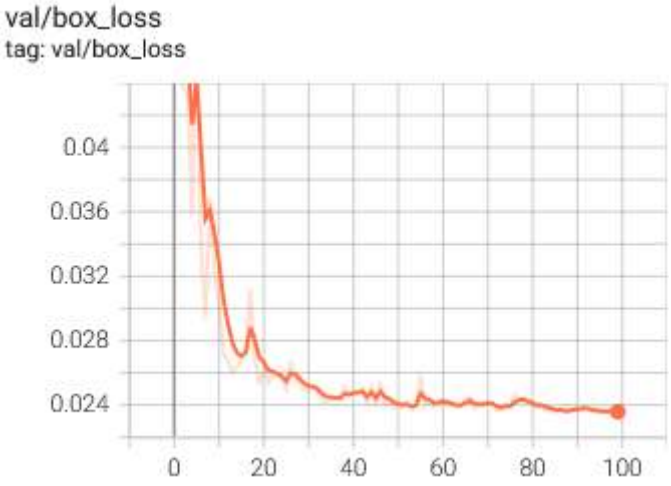


Figure 37: évolution de fonction de perte des coordonnées du bounding box en validation

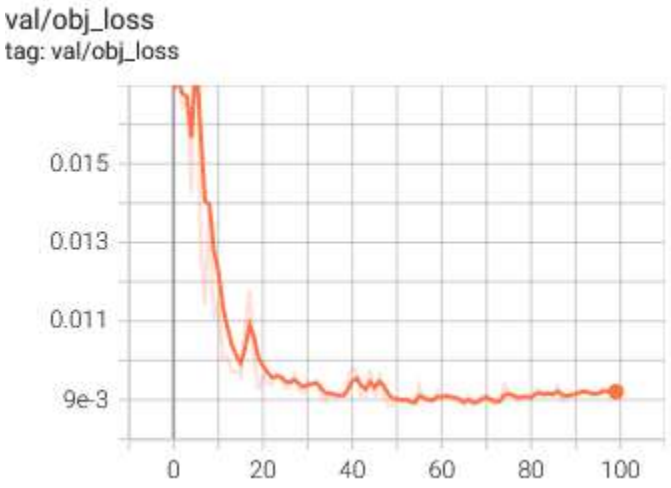


Figure 38: évolution de la fonction de perte de score d'objectness en validation

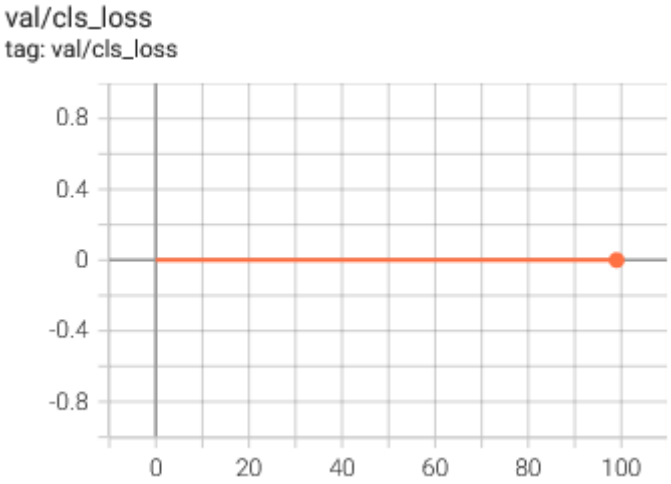
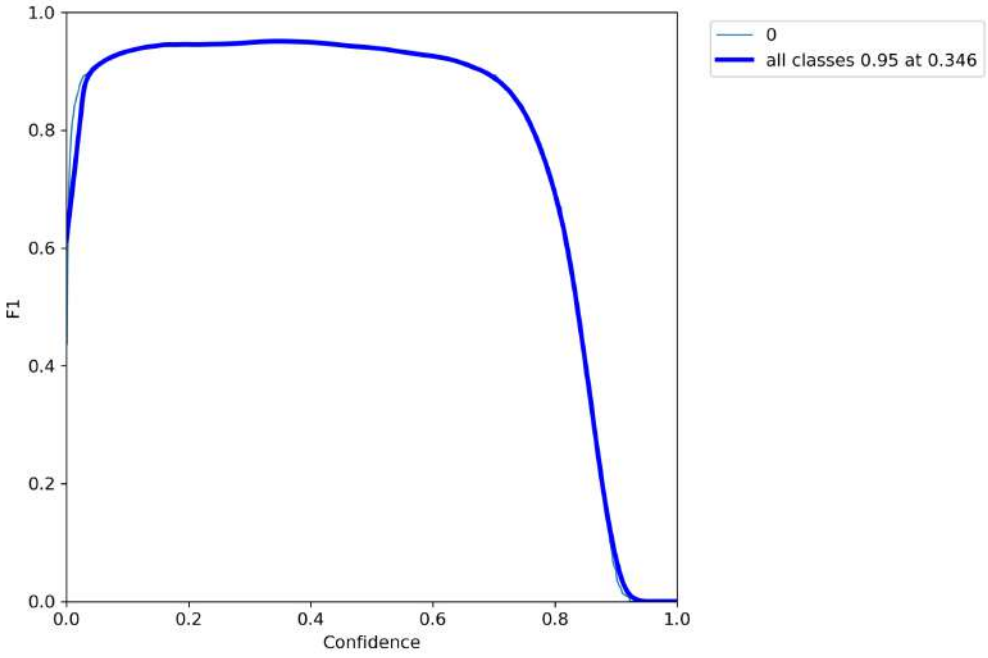
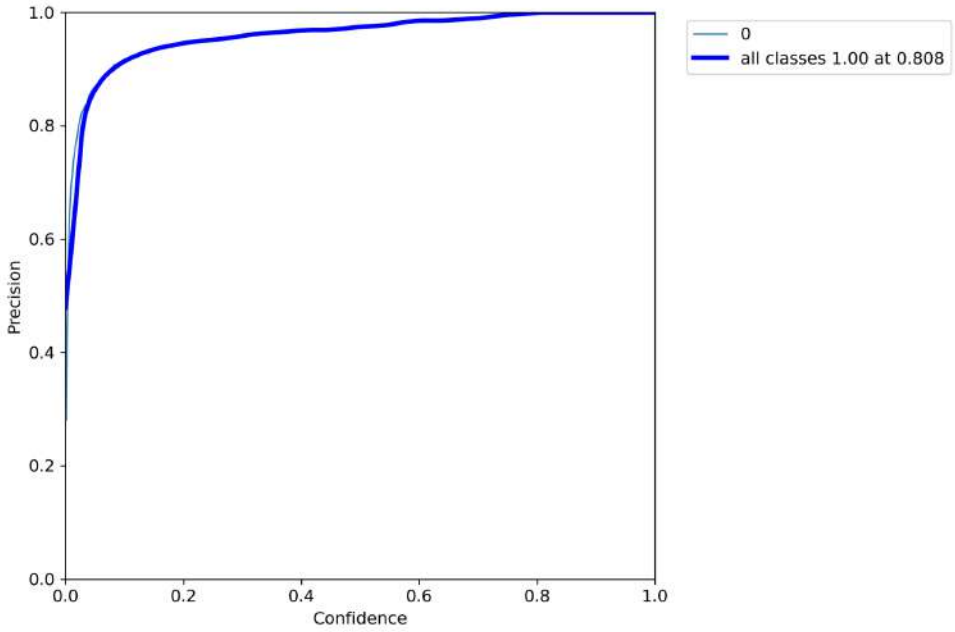
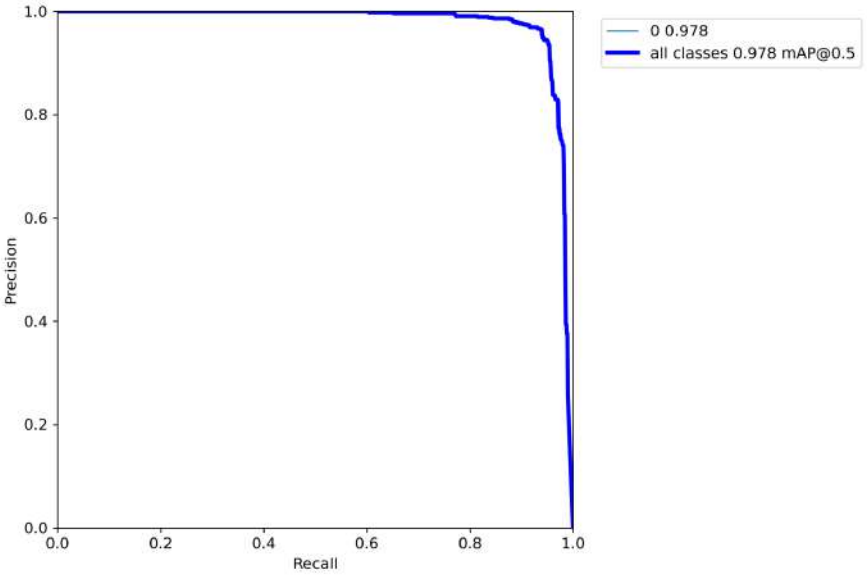
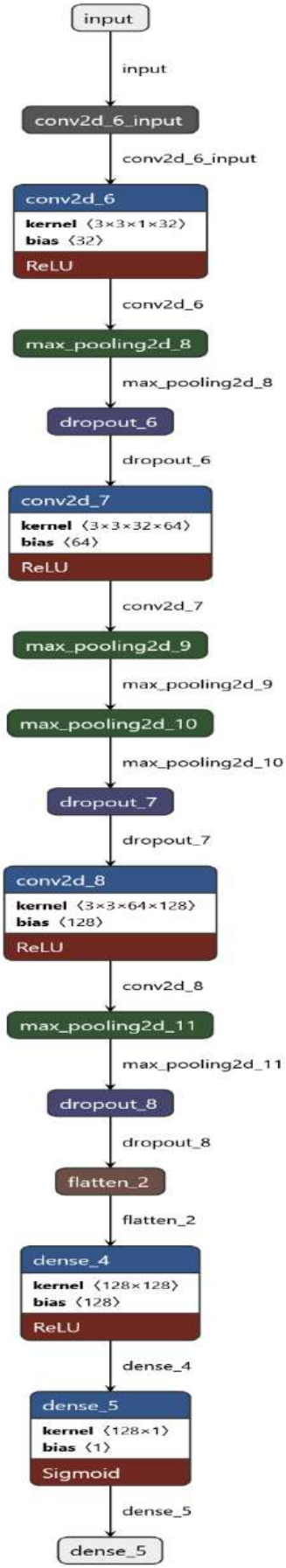


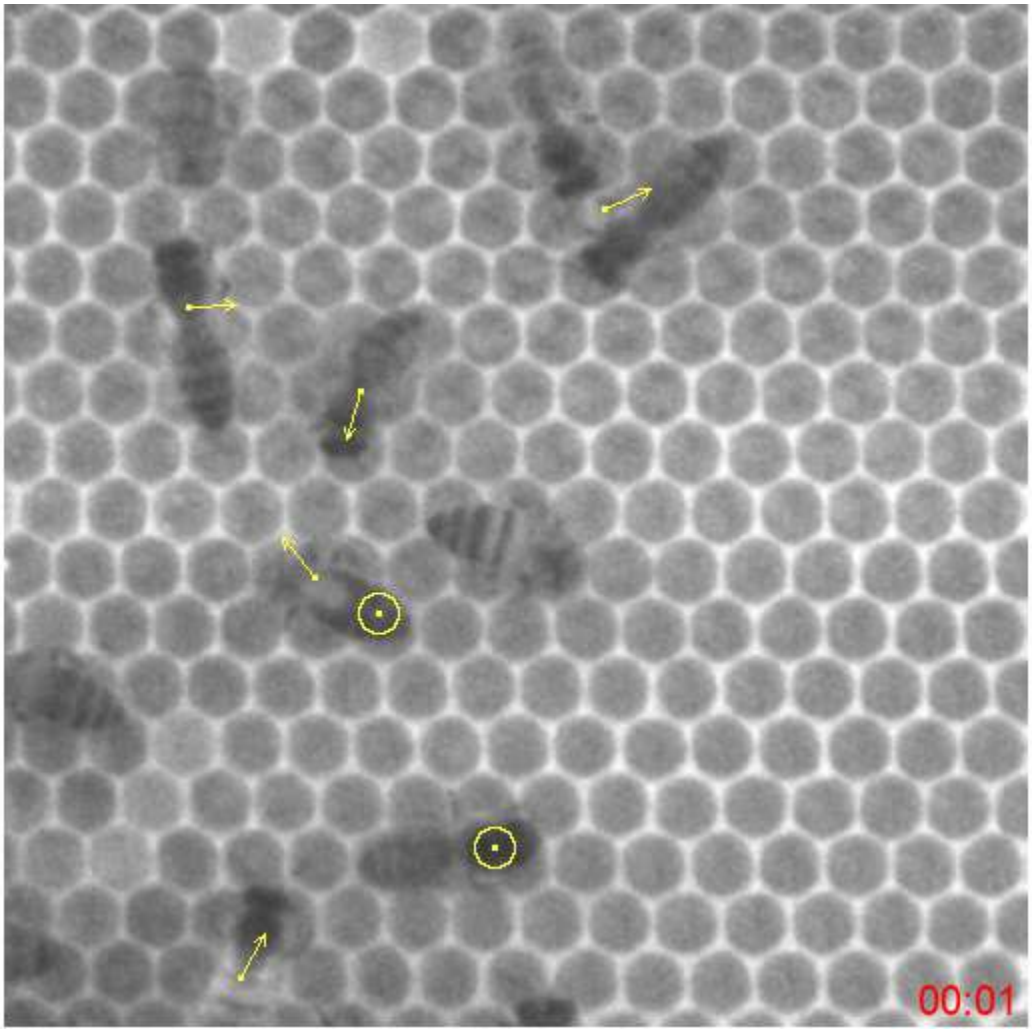
Figure 39: évolution de fonction de perte de class en validation

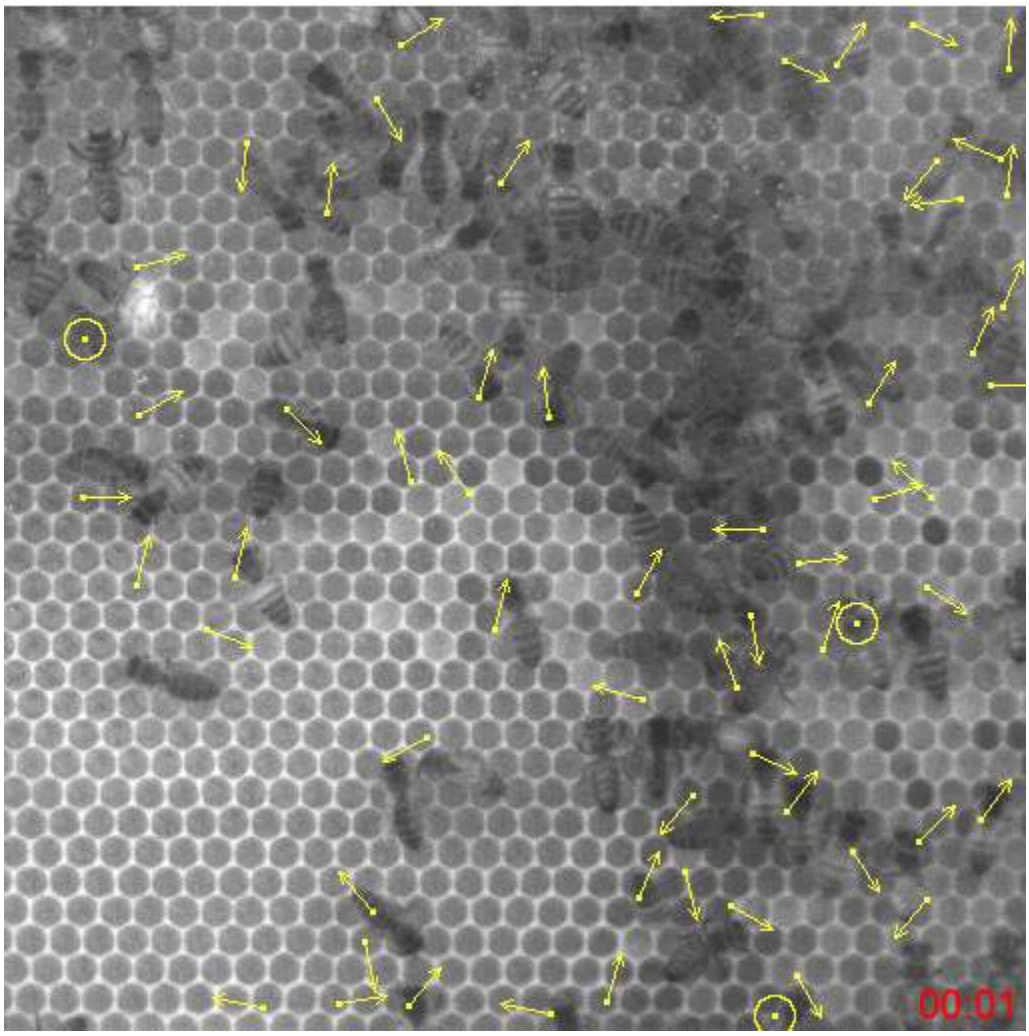

0
all classes 0.95 at 0.346
F1
Confidence
0.0
0.2
0.4
0.6
0.8
1.0


0
all classes 1.00 at 0.808
Precision
Confidence
0.0
0.2
0.4
0.6
0.8
1.0


1.0
0.8
0.6
0.4
0.2
0.0
Precision
0.0
0.2
0.4
0.6
0.8
1.0
Recall
0 0.978
all classes 0.978 mAP@0.5


input
input
conv2d_6_input
conv2d_6_input
conv2d_6
kernel 〈3×3×1×32〉
bias 〈32〉
ReLU
conv2d_6
max_pooling2d_8
max_pooling2d_8
dropout_6
dropout_6
conv2d_7
kernel 〈3×3×32×64〉
bias 〈64〉
ReLU
conv2d_7
max_pooling2d_9
max_pooling2d_9
max_pooling2d_10
max_pooling2d_10
dropout_7
dropout_7
conv2d_8
kernel 〈3×3×64×128〉
bias 〈128〉
ReLU
conv2d_8
max_pooling2d_11
max_pooling2d_11
dropout_8
dropout_8
flatten_2
flatten_2
dense_4
kernel 〈128×128〉
bias 〈128〉
ReLU
dense_4
dense_5
kernel 〈128×1〉
bias 〈1〉
Sigmoid
dense_5
dense_5


00:01


00:01

References

[1] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 0920-5691. DOI: 10.1023/b:visi.0000029664.99615.94. URL: http://dx.doi.org/10.1023/b:visi.0000029664.99615.94.

[2] N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. IEEE, None. DOI: 10.1109/cvpr.2005.177. URL: http://dx.doi.org/10.1109/cvpr.2005.177.

[3] Koen E. A. van de Sande et al. "Segmentation as selective search for object recognition". In: *2011 International Conference on Computer Vision*. IEEE, Nov. 2011. DOI: 10.1109/iccv.2011.6126456. URL: http://dx.doi.org/10.1109/iccv.2011.6126456.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386. URL: http://dx.doi.org/10.1145/3065386.

[5] T.E. Boult and G. Wolberg. "Correcting chromatic aberrations using image warping". In: *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Comput. Soc. Press, None. DOI: 10.1109/cvpr.1992.223201. URL: http://dx.doi.org/10.1109/cvpr.1992.223201.

[6] M.A. Hearst et al. "Support vector machines". In: *IEEE Intelligent Systems and their Applications* 13.4 (July 1998), pp. 18–28. ISSN: 1094-7167. DOI: 10.1109/5254.708428. URL: http://dx.doi.org/10.1109/5254.708428.

[7] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115.3 (Apr. 2015), pp. 211–252. ISSN: 0920-5691. DOI: 10.1007/s11263-015-0816-y. URL: http://dx.doi.org/10.1007/s11263-015-0816-y.

[8] Maxime Bucher, Stphane Herbin, and Frdric Jurie. "Hard Negative Mining for Metric Learning Based Zero-Shot Classification". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2016, pp. 524–531. ISBN: 9783319494081. DOI: 10.1007/978-3-319-49409-8_45. URL: http://dx.doi.org/10.1007/978-3-319-49409-8_45.

[9] Ross Girshick. "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Dec. 2015. DOI: 10.1109/iccv.2015.169. URL: http://dx.doi.org/10.1109/iccv.2015.169.

[10] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: (Sept. 2014). arXiv: 1409.1556v6 [cs.CV]. URL: http://arxiv.org/abs/1409.1556v6.

[11] "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2009. DOI: 10.1109/cvpr.2009.5206848. URL: http://dx.doi.org/10.1109/cvpr.2009.5206848.

[12] Yangqing Jia et al. "Caffe. Convolutional Architecture for Fast Feature Embedding". In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, Nov. 2014. DOI: 10.1145/2647868.2654889. URL: http://dx.doi.org/10.1145/2647868.2654889.

[13] Ken Chatfield et al. "Return of the Devil in the Details: Delving Deep into Convolutional Nets". In: *Proceedings of the British Machine Vision Conference 2014*. British Machine Vision Association, 2014. DOI: 10.5244/c.28.6. URL: http://dx.doi.org/10.5244/c.28.6.

[14] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence*

39.6 (June 2017), pp. 1137–1149. ISSN: 0162-8828. DOI: 10.1109/tpami.2016.2577031. URL: http://dx.doi.org/10.1109/tpami.2016.2577031.

[15] Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.4 (Apr. 2017), pp. 640–651. ISSN: 0162-8828. DOI: 10.1109/tpami.2016.2572683. URL: http://dx.doi.org/10.1109/tpami.2016.2572683.

[16] Vladimir Iglovikov et al. "TernausNetV2: Fully Convolutional Network for Instance Segmentation". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, June 2018. DOI: 10.1109/cvprw.2018.00042. URL: http://dx.doi.org/10.1109/cvprw.2018.00042.

[17] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016. DOI: 10.1109/cvpr.2016.90. URL: http://dx.doi.org/10.1109/cvpr.2016.90.

[18] Saining Xie et al. "Aggregated Residual Transformations for Deep Neural Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.634. URL: http://dx.doi.org/10.1109/cvpr.2017.634.

[19] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017. DOI: 10.1109/cvpr.2017.106. URL: http://dx.doi.org/10.1109/cvpr.2017.106.

[20] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: (Apr. 2018). arXiv: 1804.02767v1 [cs.CV]. URL: http://arxiv.org/abs/1804.02767v1.

[21] Katarzyna Bozek et al. "Towards Dense Object Tracking in a 2D Honeybee Hive". In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018. DOI: 10.1109/cvpr.2018.00440. URL: http://dx.doi.org/10.1109/cvpr.2018.00440.

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: (May 2015). arXiv: 1505.04597v1 [cs.CV]. URL: http://arxiv.org/abs/1505.04597v1.

[23] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. "Simple online and realtime tracking with a deep association metric". In: *2017 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sept. 2017. DOI: 10.1109/icip.2017.8296962. URL: http://dx.doi.org/10.1109/icip.2017.8296962.

[24] Alex Bewley et al. "Simple online and realtime tracking". In: *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sept. 2016. DOI: 10.1109/icip.2016.7533003. URL: http://dx.doi.org/10.1109/icip.2016.7533003.

[25] H. W. Kuhn. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2.1-2 (Mar. 1955), pp. 83–97. ISSN: 0028-1441. DOI: 10.1002/nav.3800020109. URL: http://dx.doi.org/10.1002/nav.3800020109.

[26] R. E. Kalman. "A New Approach to Linear Filtering and Prediction Problems". In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. ISSN: 0021-9223. DOI: 10.1115/1.3662552. URL: http://dx.doi.org/10.1115/1.3662552.

[27] Patrick Dendorfer et al. "MOTChallenge: A Benchmark for Single-Camera Multiple Target Tracking". In: *International Journal of Computer Vision* 129.4 (Dec. 2020), pp. 845–881. ISSN: 0920-5691. DOI: 10.1007/s11263-020-01393-0. URL: http://dx.doi.org/10.1007/s11263-020-01393-0.

[28] Benjamin Laugraud, Sbastien Pirard, and Marc Van Droogenbroeck. "LaBGen-P-Semantic: A First Step for Leveraging Semantic Segmentation in Background Generation". In: *Journal*

of Imaging 4.7 (June 2018), p. 86. ISSN: 2313-433X. DOI: 10.3390/jimaging4070086. URL: http://dx.doi.org/10.3390/jimaging4070086.

Université Hassan II Casablanca
Ecole Nationale Supérieure d'Arts et Métiers Casablanca

Nom : BELLAL
Prénom : Mohamed Yassine
Filière : Intelligence Artificielle et Génie Informatique

Titre du rapport :

La surveillance automatique de la ruche en utilisant les techniques de vision par ordinateur

Résumé :

Surveiller une ruche en suivant le nombre d'abeilles et leur évolution tout au long de la journée est primordiale pour connaître la santé de la ruche, ainsi que pour prédire la qualité du miel. À cette fin, les portes avec des capteurs infrarouges sont généralement utilisés pour calculer le nombre d'abeilles entrant et sortant de la ruche.

Dans ce projet, nous avons utilisé une approche basée sur les techniques de computer vision pour détecter les abeilles et les compter. Les résultats obtenus à l'aide de différentes approches sont comparés et la meilleure approche a été utilisée avec une API pour faciliter la consommation de données par les utilisateurs.

Mots clés :

IA – computer vision – machine learning – deep learning – yolo – django – docker –deepsort – LaBGen.