Analyse vidéo pour la détection, le suivi et la reconnaissance du comportement pour l'animal en situation d'élevage

Djahlin Nikue Amassah

UNIVERSITÉ D'ORLÉANS

ÉCOLE DOCTORALE MATHÉMATIQUES, INFORMATIQUE, PHYSIQUE THÉORIQUE ET INGÉNIERIE DES SYSTÈMES

Laboratoire Pluridisciplinaire de Recherche Ingénierie des Systèmes, Mécanique, Énergétique

THÈSE présentée par :

Djahlin NIKUE AMASSAH

soutenue le : **15 mars 2024**

pour obtenir le grade de : **Docteur de l'Université d'Orléans**

Discipline/ Spécialité : **Informatique**

Analyse vidéo pour la détection, le suivi et la reconnaissance du comportement pour l'animal en situation d'élevage

THÈSE dirigée par :

M. EMILE Bruno	Professeur des Universités, Université d'Orléans
M. DESQUESNES Xavier	Maître de conférences, Université d'Orléans

Encadrant :

Mme TREUILLET Sylvie	Professeure des Universités, Université d'Orléans

RAPPORTEURS :

Mme CAPLIER Alice	Professeure des Universités, Université Grenoble-Alpes
Mme ACHARD Catherine	Professeure des Universités, Sorbonne Université

JURY :

M. CHATEAU Thierry	Professeur des Universités, Université Clermont Auvergne, Président du jury
Mme CAPLIER Alice	Professeure des Universités, Université Grenoble-Alpes
Mme ACHARD Catherine	Professeure des Universités, Sorbonne Université
Mme TREUILLET Sylvie	Professeure des Universités, Université d'Orléans
M. EMILE Bruno	Professeur des Universités, Université d'Orléans
M. DESQUESNES Xavier	Maître de conférences, Université d'Orléans

Remerciements

Je tiens tout d'abord à exprimer ma profonde gratitude envers M. EMILE Bruno, mon directeur de thèse, pour son accompagnement, son soutien constant et ses conseils éclairés. Sa passion pour la recherche et son engagement envers l'excellence académique ont été une source d'inspiration constante au cours de cette aventure.

Je souhaite également adresser mes remerciements à Mme TREUILLET Sylvie, ma co-directrice de thèse et M. DESQUESNES Xavier, mon encadrant, pour leurs conseils précieux et leur expertise qui ont grandement enrichi ce travail de recherche.

Encore une fois merci à vous, directeur, co-directrice et encadrant, pour votre gentillesse, votre compréhension, votre patience, votre disponibilité permanente et pour les nombreux encouragements que vous m'avez prodigué tout au long de ce parcours académique exigeant. Vous m'avez aidé à mieux traverser les moments de pression, en transformant les défis en opportunités d'apprentissage et de croissance. Je tiens à dire que c'est un plaisir de travailler avec vous.

Mes sincères remerciements vont également à la région Centre-Val de Loire, qui a soutenu ce projet de recherche, et aux partenaires de ce projet : l'Université d'Orléans, les laboratoires PRISME et INRAe, les sociétés TEKIN et ACTI'COM, le Pôle Capteurs et Automatismes et la chambre d'agriculture de l'Indre, le ZOOPARC de Beauval, l'association Beauval Nature pour la Conservation et la Recherche.

Je suis reconnaissant envers l'IUT de l'Indre, qui a fourni un environnement stimulant pour la recherche et qui a mis à ma disposition des ressources indispensables à la réalisation de ce projet.

J'adresse tous mes remerciements à Mme ACHARD Cathérine, Professeure à la Sorbonne Université, ainsi qu'à Mme CAPLIER Alice, Professeure à l'Université Grenoble-Alpes, de l'honneur qu'elles m'ont fait en acceptant d'être rapportrices de cette thèse. Je tiens à remercier également M. CHATEAU Thierry d'avoir accepté être examinateur de cette thèse.

Mes collègues de laboratoire et de recherche méritent également ma reconnaissance. Leur collaboration, leurs idées et leur support ont contribué de manière significative à la qualité de ce travail.

Enfin, je tiens à exprimer ma profonde gratitude envers ma famille et mes amis, pour leur soutien indéfectible, leurs encouragements et leur compréhension pendant cette période exigeante.

Ce travail n'aurait pas été possible sans la contribution précieuse de chacun, et je suis reconnaissant envers tous ceux qui ont participé de près ou de loin à cette aventure.

Sommaire

Liste des figures

Liste des tableaux

Chapitre 1

Introduction

Ces dernières années, l'intelligence artificielle (IA) [1] a connu des développements importants dans divers domaines, notamment le traitement d'images, le traitement du langage naturel, la robotique et les systèmes automatisés, révolutionnant différents aspects de nos vies [36]. Les réseaux de neurones profonds, avec leur capacité à modéliser des relations complexes et des représentations structurées, ainsi que la croissance exponentielle des données et de la puissance de calcul des ordinateurs ont créé un terrain fertile pour l'apprentissage et le développement de nouveaux algorithmes d'IA, en particulier dans le domaine de la vision par ordinateur [2](un domaine scientifique et une branche de l'intelligence artificielle qui traite de la façon dont les ordinateurs peuvent acquérir une compréhension de haut niveau à partir d'images ou de vidéos numériques.). Ces progrès s'appuient sur des techniques d'apprentissage profond, en particulier les réseaux neuronaux convolutifs (CNN) [3] et leurs variantes [37], pour extraire et comprendre les informations visuelles des images et des vidéos. La vision par ordinateur peut être appliquée à de nombreux secteurs d'activités tels que la médecine, l'automobile, l'aéronautique, le multimédia, l'agriculture, afin de réduire l'erreur humaine et de fournir une surveillance ou une prise de décision plus efficiente. En élevage par exemple, le suivi du bétail joue un rôle crucial pour assurer le bien-être et la productivité des animaux. En détectant les anomalies et en fournissant des informations en temps réel sur les comportements du troupeau, les technologies de vision par ordinateur, couplées avec un système de vidéosurveillance, peuvent permettre une gestion plus efficace du bétail, contribuer à améliorer la productivité et permettre des pratiques agricoles durables [38].

1.1 Projet ANIMOV

Éleveurs agricoles et soigneurs en parc animalier partagent un même besoin d'observations et de surveillance des animaux dont ils ont la charge. Un éleveur de chèvres, par exemple, veut connaître le comportement des animaux afin de disposer d'indicateurs précis pour piloter l'alimentation et la repro-

1. https ://fr.wikipedia.org/wiki/Intelligence_artificielle
2. https ://fr.wikipedia.org/wiki/Vision_par_ordinateur
3. https ://fr.wikipedia.org/wiki/R%C3%A9seau_neuronal_convolutif

duction du troupeau. Il souhaite également prévenir une situation anormale (agressivité, maladie, etc.). De même, un soigneur veut connaître le comportement des animaux dont il s'occupe, tout particulièrement la nuit lorsque l'observation directe n'est plus possible. L'observation permanente et directe des animaux 24 heures sur 24 par un humain n'est évidemment pas envisageable pour des raisons de coût mais également d'altération du comportement des animaux.

La reconnaissance d'activités ou reconnaissance d'actions, est un domaine de recherche en soi en vision par ordinateur et en apprentissage automatique, avec diverses applications. L'application la plus courante est l'identification et la compréhension des activités humaines. Les techniques de reconnaissance d'actions peuvent être appliquées également à la surveillance du bétail [39] [40], où elles contribuent à améliorer le bien-être des animaux, la productivité et les pratiques de gestion agricole. Il s'agit d'analyser des données visuelles capturées par des caméras ou des dispositifs équipés de capteurs, et d'appliquer des algorithmes d'apprentissage automatique pour reconnaître et classer des comportements spécifiques tels que se nourrir, boire, se coucher, se toiletter, s'accoupler, ou encore la mise-bas, le comportement territorial, les interactions sociales et aussi des comportements anormaux comme un problème de santé, un repos anormal, une alimentation et/ou un abreuvement anormal, les agressions etc.

L'analyse vidéo permet une surveillance moins coûteuse que d'autres techniques se basant sur des capteurs portés par les animaux. Elle se fait également plus discrète lorsqu'il s'agit de comportements comme la reproduction ou la mise-bas [6]. En effet, la présence humaine influence le comportement animal et interdit bien souvent d'observer des situations spécifiques. L'utilisation de colliers accéléromètres ou d'étiquettes RFID [41] placés sur l'animal est possible. Toutefois, en raison du grand nombre de lecteurs et d'étiquettes RFID nécessaires, l'utilisation de ces techniques peut s'avérer coûteuse pour la surveillance dans les moyennes et grandes exploitations. La fixation et le détachement des capteurs peuvent entraîner également des coûts de main-d'œuvre supplémentaires et une altération des informations [6]. Avec un système de vidéosurveillance, on peut effectuer l'analyse automatique du comportement des animaux à un coût plus réduit, surtout dans les grandes exploitations. On peut également combiner les techniques à base de capteurs portables et celles à base d'analyse vidéo pour une surveillance plus efficace.

Mes travaux de recherche ont été réalisés au sein du projet ANIMOV « Animal Movements Observation from Videos », un projet de recherche pluridisciplinaire mis en œuvre sur la période 2019-2023 par un consortium régional en Centre-Val-de-Loire composé de l'Université d'Orléans (laboratoire PRISME et Pôle Capteurs), le centre de recherche INRAe Val de Loire (sites de Tours et Bourges), la Chambre d'Agriculture de l'Indre, l'Institut de l'Elevage et trois entreprises (TEKIN, Acti'Com, ZooParc de Beauval). Ce consortium permet de regrouper des compétences complémentaires (tableau 1.1) : développement d'algorithmes appliqués à l'imagerie ; systèmes vidéo, objets connectés ; physiologie animale, comportement animal ; développement agricole ; conservation des espèces sauvages.

L'objectif principal du projet est de fournir aux éleveurs et/ou aux soigneurs un outil permettant d'analyser automatiquement le comportement des animaux en situation d'élevage pour détecter les cycles d'activité et les situations anormales. Ce projet porte principalement sur deux espèces animales : les

éléphants en parc animalier et les chèvres en stabulation d'élevage. Cette thèse porte principalement sur les recherches menées pour l'analyse automatique des activités chez les chèvres. L'objectif principal de nos recherches est donc de développer et mettre en œuvre un système de vision permettant d'analyser automatiquement le comportement chez les chèvres. Afin de mieux atteindre notre objectif, nous l'avons découpé en trois sous tâches :

- La détection des animaux ;
- Le suivi des animaux ;
- La classification des activités des animaux.

Il s'agit des trois sous-tâches classiquement réalisées dans les systèmes de reconnaissance d'activités aussi bien chez l'homme que chez les animaux. Ainsi la détection va nous permettre de reconnaître une instance de l'objet (ici une chèvre) et de la localiser précisément dans l'image ou la vidéo. Le suivi va nous permettre de suivre chaque chèvre détectée grâce à un identifiant unique que nous allons maintenir tout au long de la séquence vidéo. Et enfin la classification des activités va nous permettre de reconnaître ce que la chèvre est en train de faire (couchée, debout, s'abreuve, s'alimente, etc.), en fonction du temps. Une fois ces tâches terminées et rassemblées en un seul système, les éleveurs auront un ensemble d'informations sur le comportement du troupeau qu'ils pourront utiliser comme des indicateurs pour une meilleure exploitation de leur ferme. Ces informations doivent être fournies en temps réel pour être plus pertinentes, il faut donc développer un système simple, rapide et efficace capable de faire du traitement temps réel.

1.2 Problématique et contraintes

Le projet ANIMOV a été mis en place pour aider les éleveurs et soigneurs avec un système de vision qui va permettre d'analyser automatiquement les vidéos afin de suivre les cycles d'activité (alimentation, abreuvement, reproduction, etc.) de leur troupeau, en temps réel, de jour comme de nuit. Pour développer un système d'analyse automatique de vidéos, nous avons besoin de données. Dans notre cas, il s'agit des vidéos sur les chèvres dans un environnement d'élevage en troupeau clos qui présentent quelques contraintes par rapport aux données les plus utilisées dans la littérature pour la reconnaissance d'activités humaines par exemple. Ces contraintes sont notamment :

- Une forte densité d'animaux dans chaque enclos : cela augmente le risque d'occultations pouvant entraîner des échecs de détection et de suivi ;
- Une ressemblance élevée entre les caprins : la robe de tous les individus de l'espèce étant très similaire (figure 3.3), il est difficile de trouver des caractérisques pour les séparer efficacement par rapport au suivi des personnes où on pourrait utiliser des caractéristiques de couleur des vêtements par exemple ;
- Plusieurs caméras pour un même enclos avec des chevauchements impliquant une réidentification des chèvres d'une caméra à une autre, tâche dont la complexité augmente avec la ressemblance entre les sujets et aussi avec l'occultation ;

TABLE 1.1 – Description des partenaires du projet

Partenaire	**Type**	**Rôle dans le projet**
Laboratoire PRISME	Université	— Développement d'algorithmes pour le suivi du cycle d'activités des animaux, la détection et la classification de comportements anormaux — Coordination du projet
INRAe de Tours Laboratoire PRC	Recherche	— Expertise comportementale — Fourniture d'un cadre d'expérimentation en stabulation
INRAe de Bourges	Recherche	Vérité terrain dans une ferme caprine
Chambre d'agriculture de l'Indre	Organisme consulaire	— Expertise comportementale — Fourniture d'un cadre d'expérimentation en exploitation (chèvre)
TEKIN	Entreprise	— CDC — Produits applicatifs
ACTI'COM	Entreprise	— CDC — Produits applicatifs
Zoo parc de Beauval	Entreprise	— Expertise comportementale — Fourniture d'un second cadre d'expérimentation en enclos (éléphants)
Beauval Nature	Association	Vulgarisation des découvertes scientifiques

– Manque de données publiques existantes sur la détection et le suivi des caprins ;

– Traitement temps réel : notre système d'analyse automatique des comportements nécessite un traitement en temps réel des flux vidéos.

1.3 Plan de la thèse

Ce manuscrit est organisé en six chapitres. Ce chapitre introductif décrit le contexte général, la problématique à résoudre ainsi que les objectifs de travail. Le chapitre 2 présente un état de l'art sur les méthodes de détection et de suivi des animaux basées sur l'analyse vidéo. Nous commencerons par une présentation des différentes solutions appliquées au suivi d'animaux, suivie d'une description détaillée des méthodes générales de détection et suivi d'objets, en distinguant les méthodes classiques de celles basées sur l'apprentissage profond.

Le chapitre 3 sera consacré à l'évaluation de méthodes de la littérature sur notre base de données, pour la détection et le suivi. Après la présentation des données et de la vérité terrain, nous présentons les résultats obtenus avec plusieurs méthodes populaires et performantes, ainsi qu'une analyse de ceux ci.

Dans le chapitre 4, nous proposons une nouvelle architecture, dérivée du modèle YOLOX, pour la détection. Nous montrons, notamment, que cette architecure améliore significativement les résultats de la partie détection.

Dans le chapitre 5, nous présentons l'analyse de quelques comportements réalisée avec la méthode de détection et de suivi mise en place.

Le chapitre 6 résume nos différentes contributions et présente les perspectives d'amélioration et d'autres aspects de recherche non traités dans ce manuscrit.

Chapitre 2

État de l'art

2.1 Introduction

La surveillance du bétail fait référence au processus d'observation et de suivi du comportement des animaux d'élevage, tels que les vaches, les porcs ou les chèvres. Cette surveillance peut être un outil précieux pour la gestion du bétail en permettant aux agriculteurs et aux gestionnaires de bétail d'obtenir des informations sur la santé, le bien-être et les activités de leurs animaux. Les méthodes de surveillance des animaux existantes utilisent souvent des capteurs électroniques portés par les animaux. Ces méthodes peuvent engendrer des coûts d'installation et de maintenance élevés, surtout pour les grandes exploitations. Avec les progrès récents en IA, surtout en vision par ordinateur, des applications basées sur le traitement d'image ou l'analyse vidéo, pour la surveillance du bétail, ont vu le jour et permettent de réduire le travail manuel et d'économiser du temps. Elles peuvent permettre d'améliorer la santé des animaux, augmenter les bénéfices et réduire l'empreinte écologique [42]. Dans ce chapitre nous passons en revue les travaux de la littérature existants sur l'automatisation de la surveillance vidéo du bétail, ainsi que les défis et les perspectives qui restent à mettre en œuvre. La reconnaissance d'actions, utilisée pour surveiller le comportement des animaux, se base sur les méthodes de détection et de suivi d'objets pour analyser une vidéo et en extraire des informations pertinentes afin de comprendre la scène. La détection est utilisée afin de reconnaître et de localiser un objet d'intérêt dans une image ou une vidéo. Le suivi, quant-à lui, permet de suivre cet objet dans toute la séquence vidéo, avec un identifiant unique. Ces deux étapes sont nécessaires pour mettre en place un système automatique de vidéosurveillance du bétail. Nous allons également présenter l'état de l'art sur ces méthodes (détection d'objets et suivi d'objets).

2.2 Suivi du bétail par l'analyse vidéo

La surveillance vidéo du bétail emploie des techniques de traitement d'images et/ou, plus récemment, des méthodes d'apprentissage profond [1] afin de détecter les comportements recherchés. Les travaux re-

1. https ://fr.wikipedia.org/wiki/Apprentissage_profond

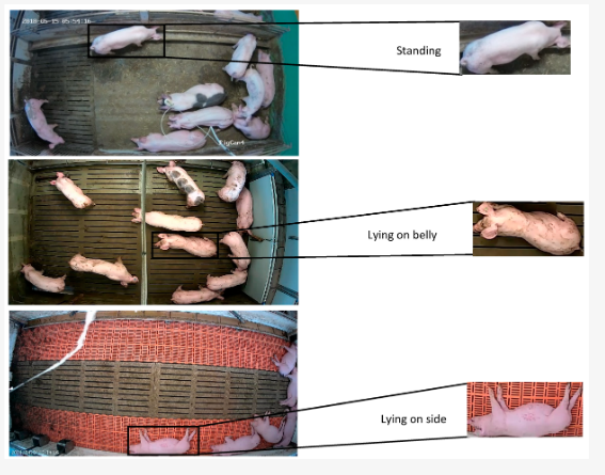


FIGURE 2.1 – Exemple d'images utilisées pour la détection des postures dans [1]

censés ci-après concernent principalement les bovins, les porcs, et les chèvres dans un environnement d'élevage.

Yujuan et al. [43] proposent une méthode basée sur l'algorithme Mean Shift [44] pour suivre avec précision le mouvement de la mâchoire des vaches laitières. Pour vérifier la précision de la méthode, six vidéos, d'une durée totale de 99 minutes et de vingt-quatre mille (24000) images ont été sélectionnées. D'après les résultats des tests effectués dans leur étude, un taux de réussite de 92,03 % a été obtenu pour la méthode de détection de rumination proposée. Les résultats démontrent que cette méthode, qui surveille la rumination des vaches laitières, est efficace et réalisable.

Abozar Nasirahmadi et al. [1] ont mené une étude afin de déterminer si un système d'imagerie bidimensionnelle (2D), associé à des approches d'apprentissage profond, pouvait être utilisé pour détecter les postures debout et couchées sur le ventre, et sur le côté (figure 2.1), des porcs dans les conditions d'une exploitation commerciale. Les images utilisées dans cette étude ont été enregistrées par des caméras de vue de dessus (VIVOTEK IB836BA-HF3 et Hikvision DS-2CD2142FWD-I). Ils ont testé trois méthodes (Faster R-CNN, R-FCN et SSD) pour la détection des postures. Les résultats expérimentaux ont montré que la méthode R-FCN (avec ResNet101) était capable de détecter les postures couchées et debout avec une précision moyenne plus élevée.

Kashiha et al. [2] ont étudié la faisabilité d'une méthode automatisée pour estimer le poids des porcs individuellement en utilisant des techniques de traitement d'images. Cette étude comprenait des mesures dans quatre enclos de porcs en croissance, chacun composé de dix porcs. Chaque enclos était surveillé par une caméra CCD [2] posée au plafond en vue de dessus. Les vidéos ont été enregistrées au format MPEG-1, avec une fréquence de 25 images par seconde, une largeur d'image de 720 pixels et une hauteur d'image de 576 pixels. Chaque porc est automatiquement identifié par sa marque unique peinte sur son dos (figure 2.4) à l'aide d'un seuillage d'Otsu [45] et de descripteurs de Fourier [46]. Le processus d'estimation du poids se déroule comme suit : la première étape du traitement a consisté à segmenter l'image afin de

2. https ://fr.wikipedia.org/wiki/Capteur_photographique_CCD

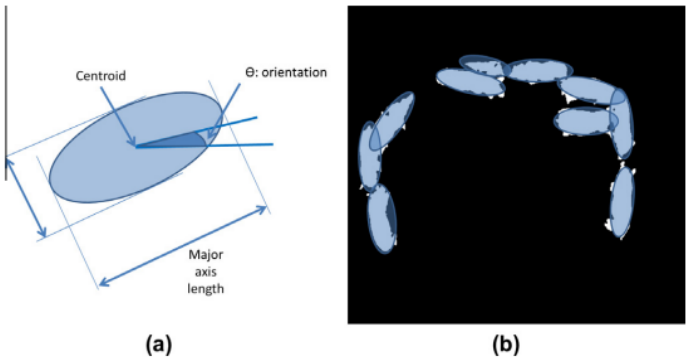


FIGURE 2.2 – Plan des enclos dans [2]

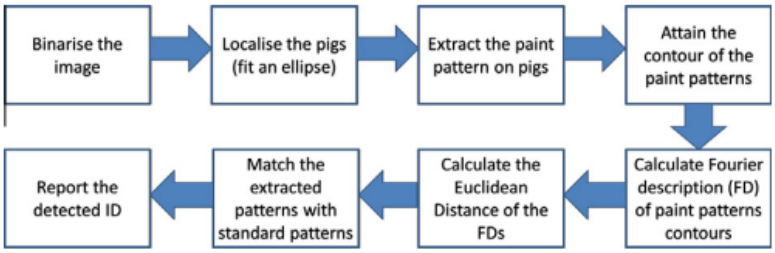


FIGURE 2.3 – Organigramme de l'identification des porcs marqués dans [2]

trouver l'emplacement des porcs et l'emplacement du motif d'identification appliqué. Ensuite, la surface occupée par le porc est extraite sous forme d'ellipse, car les régions lumineuses liées aux porcs présentent un contraste assez élevé avec l'arrière-plan (sol de l'enclos) et son poids est estimé par une modélisation dynamique [47]. Pour extraire la surface des porcs, des ellipses ont été ajustées aux objets de l'image binaire à l'aide de la méthode d'ajustement des ellipses par les moindres carrés. Ensuite, les paramètres des ellipses tels que l'orientation, la longueur de l'axe majeur, la longueur de l'axe mineur et le centroïde ont été calculés pour tous les objets de l'image (figure 2.2). La figure 2.3 présente les différentes étapes de cette approche utilisée pour l'identification des porcs marqués dans un enclos. Cette méthode exige que les porcs soient marqués individuellement, ce qui n'est pas toujours pratique à l'échelle commerciale.

Nasirahmadi et al. [3] ont étudié l'évolution du repos en groupe des porcs en fonction du changement de la température ambiante. La segmentation d'Otsu, les opérations morphologiques et l'ajustement d'ellipse ont été utilisés pour segmenter les porcs de l'arrière-plan de la vidéo. Ensuite, la méthode de triangulation de Delaunay a été employée pour analyser le comportement de repos de groupe des porcs. Ils se basent sur l'hypothèse que les porcs aux repos ou couchés ont une proximité élevée (figure 2.5). Ainsi, grâce aux propriétés de la région et au périmètre de chaque triangulation de Delaunay, on peut estimer la proximité des porcs les uns par rapport aux autres.

Ahrendt et al. [4] ont développé un algorithme de suivi individuel des porcs dans les étables à stabulation libre (figure 2.6). L'algorithme prend en entrée une séquence temporelle d'images couleur et, après une procédure d'initialisation, il estime les positions spatiales de chaque porc pour chaque nouvelle image. L'idée fondamentale de l'algorithme est de mettre à jour en permanence une carte de positions

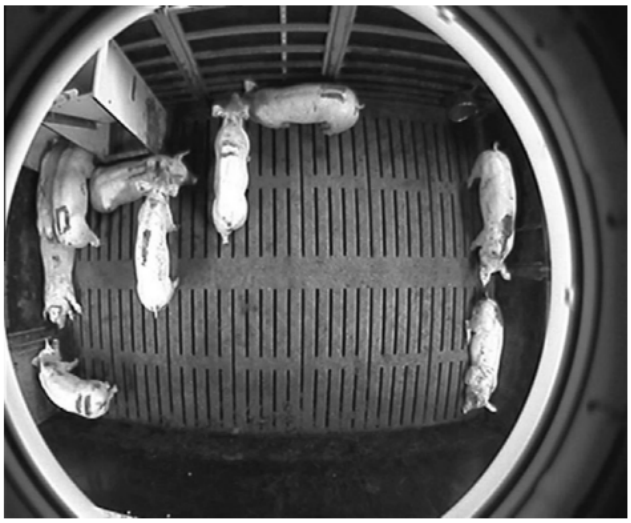


FIGURE 2.4 – Exemple d'image de la base utilisée dans [2]

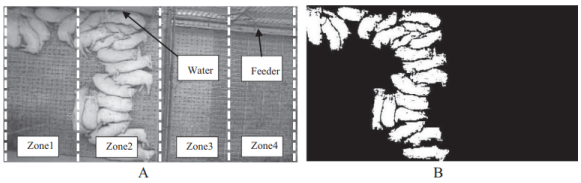


FIGURE 2.5 – Exemple d'image de la base utilisée dans [3] : l'enclos et ses quatre zones (A), image binaire après application des opérateurs morphologiques (B).

ainsi qu'un modèle pour chaque porc. La carte de positions pour un porc spécifique est constituée des coordonnées des pixels de l'image qui font partie de la région du porc. Le modèle pour chaque porc a été choisi pour être une distribution gaussienne dans un espace 5D qui comprend les coordonnées (x, y) de l'image ainsi que les coordonnées des couleurs RGB. Lorsqu'une nouvelle image est reçue, l'algorithme met d'abord à jour la carte de coordonnées à partir du modèle de porc précédent, puis, en utilisant la nouvelle estimation de la carte de coordonnées, met à jour le modèle. Pour améliorer les performances de l'algorithme, une estimation de l'arrière-plan, par soustraction, est effectuée et permet d'éviter les erreurs lorsque l'arrière-plan a des couleurs similaires à celles des porcs. La principale limite de cette méthode est que le suivi peut être perdu si les mouvements des porcs sont rapides, et également en cas d'occultations entre les porcs.

Pour analyser les zones d'occupation des porcs, Nilsson et al. [5] ont proposé une méthode de segmentation des porcs basée sur l'apprentissage. Afin d'évaluer l'approche en conditions réelles, un enclos contenant neuf jeunes porcs a été filmé en vue de dessus par une caméra Axis M3006 d'une résolution de 640×480 pixels, en trois sessions de 10 minutes dans différentes conditions d'éclairage. Leur approche est basée sur une segmentation par apprentissage contrairement à la segmentation d'Otsu utilisée dans les approches précédentes. Pour l'apprentissage de la segmentation, la régression logistique est appliquée sur 12 bandes de caractéristiques extraites manuellement : l'espace couleur CEI-LUV (3 bandes), l'amplitude normalisée du gradient (une bande), les gradients orientés (6 bandes), un canal d'Otsu et un filtre

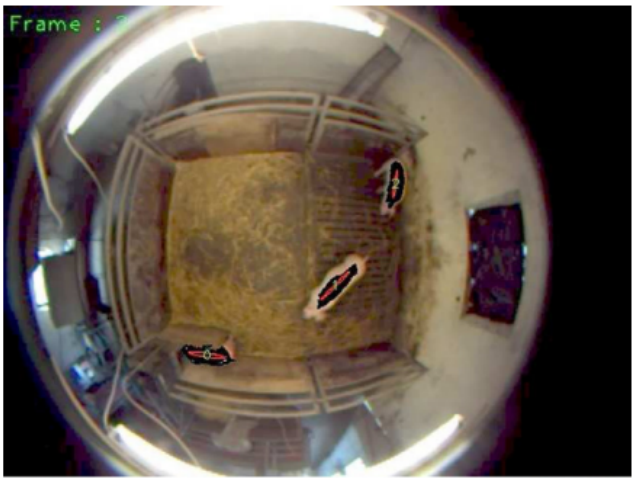


FIGURE 2.6 – Exemple d'image de la base utilisée dans [4]

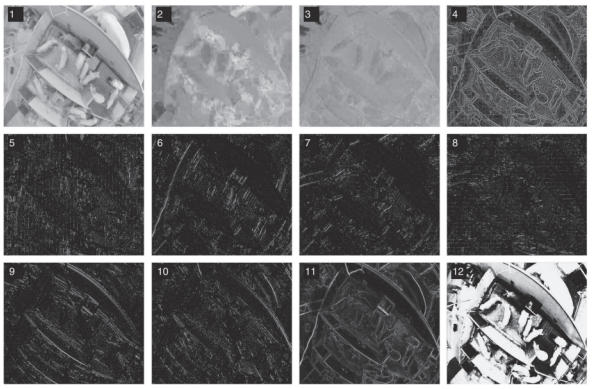


FIGURE 2.7 – Les types de caractéristiques pour l'apprentissage de la segmentation dans [5]

maximum-minimum (respectivement de 1 à 12 sur la figure 2.7). La segmentation basée sur l'apprentissage n'est pas une mesure exacte car elle est influencée par le nombre de pixels de chaque porc, qui varie en fonction de la taille du cochon ainsi que de sa posture et de son orientation. Un cas d'échec typique de cette méthode est dû aux occultations dans l'enclos.

Lei Zhang et al. [6] proposent une méthode basée sur une caméra vidéo 2D et des techniques récentes de réseaux de neurones pour détecter et suivre automatiquement des porcs individuellement dans un hangar, sans qu'il soit nécessaire de marquer manuellement ou d'identifier physiquement les porcs (figure 2.8). Pour la détection, ils ont comparé trois méthodes : le R-FCN, le Faster R-CNN et le SSD ; pour le suivi, le filtre de corrélation a été utilisé.

Yuan Rao et al. [7] décrivent un système de surveillance du bien-être des chèvres à la ferme, avec une combinaison de l'internet des objets et de l'apprentissage automatique. L'expérience a été menée dans une ferme caprine commerciale appartenant à Hefei Angu Agricultural Ltd en Chine. Environ 60 chèvres ont été sélectionnées et réparties dans 10 enclos entièrement sur caillebotis (3,0 m de large x 3,5 m de long) constitués de tubes d'acier creux et dotés d'un plancher entièrement sur caillebotis, soit environ 6 chèvres par enclos (figure 2.9). Ils proposent une classification et quantification automatiques des comportements

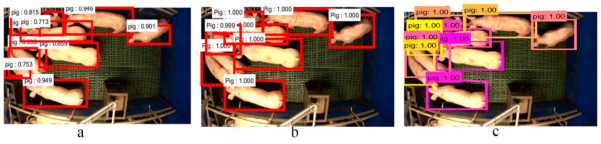


FIGURE 2.8 – Résultats de détection des porcs dans les travaux de [6] : a- R-FCN, b- Faster R-CNN et c- SSD

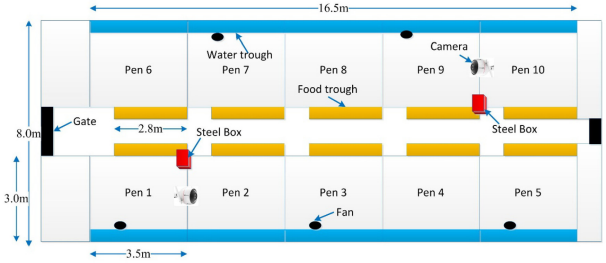


FIGURE 2.9 – Plan des enclos dans [7]

des chèvres basées sur l'apprentissage profond en faisant d'abord la détection individuelle des chèvres avec le Faster R-CNN. Pour détecter l'alimentation et l'abreuvement des chèvres, ils comparent les positions des chèvres détectées par rapport aux traits représentants la mangeoire et l'abreuvoir (figure 2.10).

Patrizia Tassinari et al. décrivent dans [8] le développement d'un système de vision par ordinateur, basé sur l'apprentissage profond, visant à reconnaître en temps réel les vaches, à détecter leurs positions, leurs actions et leurs mouvements et à enregistrer l'historique des sorties pour chaque animal. Les vidéos ont été enregistrées par une caméra HDR-CX115E (Sony) dans un standard de haute qualité (résolution HD, 25 images par seconde). L'enregistrement a été effectué sur un trépied placé à 2 m au-dessus du sol de la grange, de sorte que la hauteur totale de l'enregistrement était d'environ 3,50 m (figure 2.11). Quatre vaches ont été sélectionnées pour entraîner et valider le réseau neuronal YOLOv3 capable de reconnaître une vache à partir de son pelage.

Hai Ho Dac et al. [48] ont mené une étude visant à développer un système de reconnaissance faciale pour les vaches des fermes laitières en utilisant des modèles d'apprentissage profond et des techniques

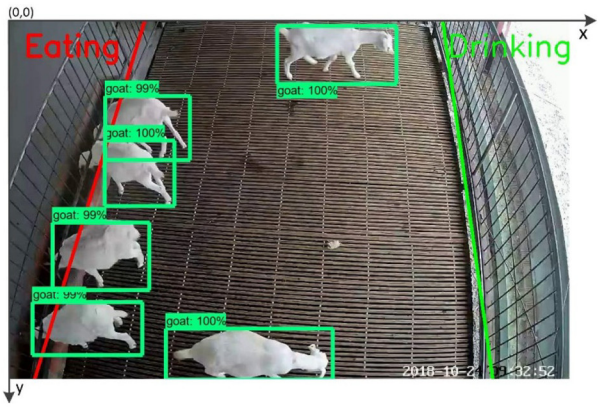


FIGURE 2.10 – Résultats de sortie dans [7]

FIGURE 2.11 – La caméra HDR-CX115E positionnée pour l'enregistrement vidéo dans [8]

de vision par ordinateur. Cette approche, potentiellement applicable à d'autres animaux de ferme, est importante pour l'identification et l'évaluation du bien-être. Leur pipeline d'analyse vidéo suit les systèmes standard de reconnaissance des visages humains, qui se composent de quatre étapes importantes : (i) détection des visages, (ii) recadrage des visages, (iii) encodage des visages, et (iv) recherche des visages. Trois modèles d'apprentissage profond ont été utilisés dans le pipeline d'analyse : détecteur de visage (YOLOv5), prédicteur de points de repère (Resnet18), et encodeur de visage (Resnet101 et ArcFace). Tous les modèles d'apprentissage profond ont été affinés grâce à l'apprentissage par transfert sur un ensemble de données de vaches laitières recueillies dans une ferme laitière robotisée située sur le campus Dookie de l'Université de Melbourne, en Australie.

Alvaro et al. [49] proposent une approche pour la surveillance multivues du comportement individuel de la vache basée sur la reconnaissance d'actions à l'aide de données vidéo. L'étable expérimentale avait une taille de 30 × 12 mètres et abritait 21 vaches âgées de 1 à 7 ans, dont 3 veaux. Deux caméras de surveillance HIK (Hikvision) d'une résolution de 4K (3840 × 2160 pixels) ont été installées dans l'étable pour capturer des données vidéo en continu. Le système proposé prend une séquence d'images en entrée et utilise un détecteur d'objets YOLOv5 [50] pour localiser et identifier les classes qui représentent en même temps les actions des objets. Ils ont entraîné le détecteur sur un jeu de données des activités des vaches qui comprend :

— les actions individuelles : debout, se déplacer, se reposer, se lever, se toiletter ;

— les actions de groupe : nourrir veau, se battre et lécher une autre vache ;

— les actions des parties : bouger la tête, remuer la queue et ruminer.

Les régions détectées par YOLOv5 sont ensuite introduites dans un mécanisme de suivi et d'identification, ce qui permet au système de suivre en permanence chaque individu dans la scène et de lui attribuer un numéro d'identification unique.

Pour détecter la reproduction chez les vaches laitières, Yangyang Guo et al. [51] proposent une méthode

qui se base sur des caractéristiques géométriques et le flot optique des régions d'images identifiées dans les vidéos prises dans les fermes laitières.

Dihua Wu et al. [52] proposent une méthode qui permet de surveiller le comportement respiratoire de plusieurs vaches à travers des vidéos. Dans un premier temps, 4000 images étiquetées manuellement ont été utilisées pour affiner le modèle YOLACT (You Only Look At CoefficienTs) pour la reconnaissance et la segmentation [53] de plusieurs vaches. Ensuite, les états de repos spécifiques (couché, debout) ont été identifiés en fusionnant un réseau de neurones convolutionnel et les algorithmes bidirectionnels de mémoire à long et à court terme (LSTM). Enfin, les algorithmes de détection de la posture (couchée et debout au repos) ont été utilisés pour le suivi du comportement respiratoire.

Pour aider les zoologistes, dans le cadre du projet ANIMOV, Souhaieb et al. [54] ont utilisé des techniques d'apprentissage profond pour localiser automatiquement les éléphants debout et les éléphants couchés dans leur enclos surveillé par plusieurs caméras. La position des éléphants détectés est ensuite projetée sur le plan de l'étable. Ainsi, au lieu de regarder toutes les vidéos, les zoologistes examinent l'historique des positions dans le plan, ce qui leur permettra de mesurer plus rapidement les phases de sommeil des éléphants.

Su Myat Noe et al. [55] proposent une approche basée sur les algorithmes de suivi multi-objets (MOT) par apprentissage profond pour détecter et suivre automatiquement et en continu les bovins à l'aide d'une caméra RVB. Cette étude compare les algorithmes de pointe, tels que Deep-SORT, Strong-SORT et des algorithmes de suivi personnalisés. Afin d'améliorer la précision du suivi de ces méthodes d'apprentissage profond, les auteurs présentent une approche de réidentification améliorée pour un ensemble de données de bovins noirs dans Strong-SORT. Les images ont été capturées en continu par une caméra fish-eye GV-FER5700 placée au sommet d'une grange contenant entre 10 et 20 bovins.

Récemment, des caméras vidéo 3D (capteur de profondeur) ont été utilisées pour surveiller les animaux en vue de dessus dans plusieurs études. Kulikov et al. [56] ont proposé une méthode pour suivre automatiquement les porcelets avec plusieurs capteurs d'images 3D Microsoft Kinect à l'aide du logiciel EthoStudio. M. Mittek et al. [9] ont proposé une méthode de suivi d'objets multiples permettant de produire des données détaillées, à long terme et en continu sur les mouvements en 3D, qui peuvent être utilisées pour détecter les alimentations et les abreuvements, les agressions et une multitude d'autres interactions sociales. Ils ont utilisé un capteur Kinect v2 [3] développé par Microsoft qui produit des images en couleur, en infrarouge et en profondeur (figure 2.12). En plus de faciliter la mesure de la profondeur, l'illuminateur infrarouge permet d'effectuer des repérages de jour comme de nuit sans avoir recours à la lumière visible. Bien que les systèmes de vision existants basés sur une caméra vidéo de profondeur aient obtenu certains succès dans la détection et le suivi des animaux en élevage, ils présentent certains inconvénients en raison des limites du capteur d'imagerie [6] (par exemple, la caméra de profondeur Kinect a une portée limitée à 4 mètres ; un champ de vision restreint avec un angle horizontal de 58,5° et un angle vertical de 45,6°) et la précision des données de profondeur est sensible à la position de la caméra. Ces

3. https ://fr.wikipedia.org/wiki/Kinect

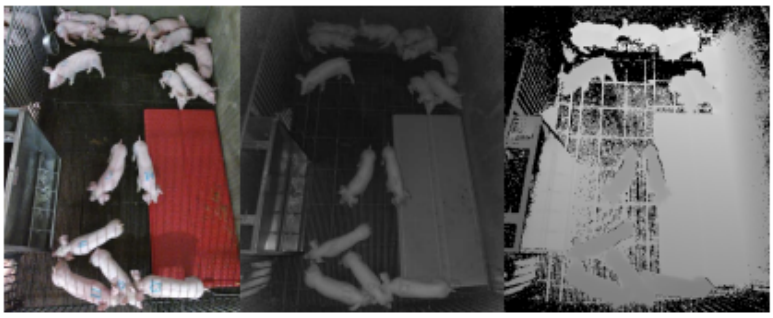


FIGURE 2.12 – Images couleur, infrarouge et de profondeur capturées par la caméra Kinect v2 dans [9]

limitations ne se manifestent pas au niveau des caméras 2D RVB. Nous pouvons retrouver dans [57], une revue des algorithmes d'apprentissage profond pour les systèmes de vision par ordinateur dans le secteur de l'élevage.

Les travaux de la littérature présentés ci-dessus explorent les techniques de détection et de suivi de comportements chez les animaux dans les vidéos et montrent des résultats assez satisfaisants. Les tâches les plus réalisées dans ces travaux sont principalement la détection, l'identification (à partir des marquages sur l'animal), le suivi et l'identification de comportements des individus. La plupart des travaux de la littérature traite un nombre d'animaux compris entre 5 et 20, ce qui réduit le risque d'occultations par rapport à notre cas où nous avons entre 20 et 60 chèvres par enclos et par caméra. Ces travaux utilisent généralement une caméra par enclos et ne traitent pas le problème de réidentification des animaux. En effet, lorsqu'on utilise plusieurs caméras pour l'analyse, on pourrait avoir besoin de suivre le même animal d'une caméra à une autre qui revient à faire de la réidentification [58]. Cette tâche reste très complexe et peu réaliser dans la vidéosurveillance des animaux surtout sur des individus très simillaires et dense comme dans notre cas. Bien qu'ils soient une solution interessante pour la surveillance du bétail, les capteurs vidéos présentent quelques limitations à prendre en compte. Une de ces limitations est le champ de vue et l'occultation dans les vidéos qui peut entraîner des échecs lors de la détection et du suivi des animaux. L'identification et la catégorisation des comportements complexes (l'anxiété, les interactions sociales, le stress, etc.) à base de la surveillance vidéo peut s'avérer difficile, surtout pour des espèces dont les actions sont nuancées. L'entraînement d'un modèle d'apprentissage automatique pour reconnaître ces comportements peut être difficile et nécessiter beaucoup de données étiquetées. Techniquement, la vidéosurveillance nécessite des algorithmes de pointe de vision par ordinateur qui peuvent être très gourmands en puissance de calcul et nécessiter des matériels ou logiciels spécifiques. Malgré ces limitations, l'analyse vidéo pour le suivi du bétail reste quand même une solution prometteuse grâce aux avancées de l'Intelligence Artificielle, au progrès de la puissance de calcul et le développement des algorithmes plus sophistiqués qui permettent d'améliorer de plus en plus les traitements.

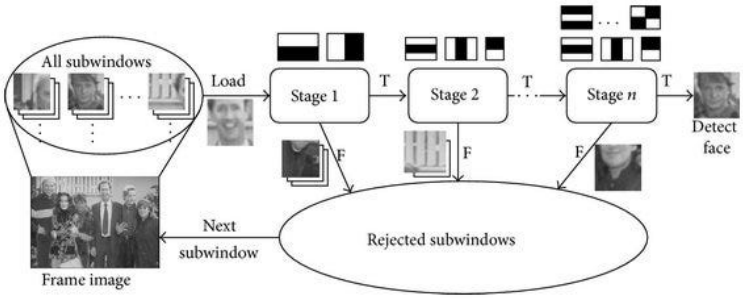


FIGURE 2.13 – Processus de détection par Viola Jones [10].

2.3 Méthodes de détection d'objets

En vision par ordinateur on désigne par détection d'objets une tâche réalisée par un algorithme capable de localiser la présence d'une instance ou plusieurs instances d'objets dans une image numérique [59] au moyens de boîtes englobantes. Historiquement, on peut noter un clivage avant et après les années 2020 en deux catégories [19] : les approches sans réseau de neurones et les approches par apprentissage profond. Nous allons d'abord commencer par voir les approches sans réseau de neurones présentées dans la section 2.3.1 et ensuite les approches par apprentissage profond présentées dans la section 2.3.2.

2.3.1 Approches sans réseau de neurones

Les approches sans réseau de neurones étaient utilisées pour faire de la détection d'objets avant l'avènement de l'apprentissage profond. Elles s'appuient sur l'extraction préalable des caractéristiques dans les images et des algorithmes conventionnels d'apprentissage automatique (KNN, SVM, Adaboost, etc.). Sans chercher à être exhaustifs, nous citons ici quelques méthodes emblématiques qui ont marqué des étapes dans la recherche :

a) Le détecteur Viola Jones : en 2001, P. Viola et M. Jones [60] ont réussi pour la première fois à détecter en temps réel des visages humains. Ce détecteur utilise une technique de fenêtres coulissantes : il s'agit de passer en revue, par des blocs ou fenêtres de taille fixe, tous les emplacements et échelles possibles d'une image pour voir si l'une des fenêtres contient un visage humain (figure 2.13). Cette méthode utilise les caractéristiques de type Haar [61] pour capturer les motifs et les variations locales de l'image. Le détecteur Viola Jones a considérablement amélioré sa vitesse de détection en intégrant trois techniques importantes : "image intégrale", "sélection des caractéristiques" et "détection en cascade". Il utilise l'algorithme AdaBoost [62] pour sélectionner un petit ensemble de caractéristiques les plus discriminantes parmi un grand nombre de caractéristiques potentielles de type Haar.

b) Le détecteur HOG : en 2005, N. Dalal et B. Triggs [63] ont proposé le descripteur de caractéristiques Histogramme des Gradients Orientés (HOG) qui capture les informations locales dans une image. En caractérisant les bords ou les frontières des objets, ce descripteur a été utilisé pour la détection

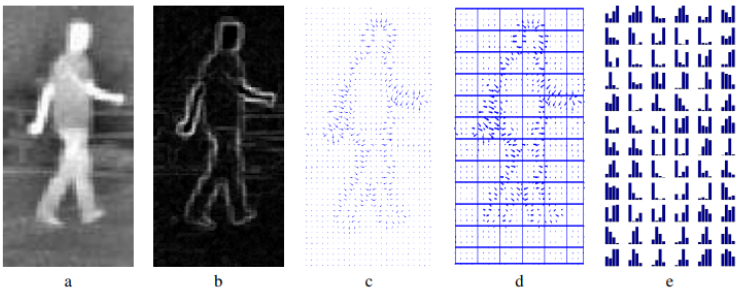


FIGURE 2.14 – Caractérisation d'une image infrarouge par HOG pour la détection d'une personne [11].

des objets. Pour assurer une certaine invariance des caractéristiques en translation, changement d'échelle, d'illumination, le descripteur HOG est calculé sur une grille dense de cellules uniformément espacées et utilise une normalisation du contraste local par chevauchement. Le descripteur HOG a rencontré un certain succès pour détecter des objets dont les bords et les formes sont bien définis, comme les piétons (figure 2.14), les voitures et d'autres objets dans les images. Cependant, ce descripteur peut échouer dans certaines situations telles que les apparences complexes des objets ou des arrière-plans encombrés.

c) Le descripteur SIFT [64] : est un algorithme de détection et de description de points d'intérêt qui identifie les caractéristiques locales communémant employées pour la mise en correspondance des objets à différentes échelles et rotations. Le descripteur SIFT s'appuie aussi sur le calcul d'histogrammes des gradient orientés dans une fenêtre de 16 x 16 pixels sub-divisée en 4 en apportant des améliorations pour plus de robustesse. Cette caractérisation extrêmement locale, en multipliant les points d'intérêt, permet de détecter de manière robuste les objets malgré les obstacles et les occultations en faisant correspondre des caractéristiques à une base de données d'objets connus à l'aide d'un algorithme comme le plus proche voisin (K-NN [65]), suivi d'une transformée de Hough [66] pour identifier les groupes appartenant à un seul objet, et enfin d'une vérification par une solution des moindres carrés pour des paramètres de pose cohérents. Ces descripteurs sont utilisés dans [67] pour la reconnaissance faciale, avec quelques modifications afin d'améliorer les performances. Bien que SIFT fut largement utilisé dans le passé, il rencontrait des limitations face aux changements d'échelle et sa complexité de calcul est devenue une limitation pour les applications en temps réel et à grande échelle.

d) Le modèle DPM (Deformable Part-based Model), vainqueur des défis de détection d'objets VOC-07, VOC-08 et VOC-09 [68], est le summum des méthodes sans réseau de neurones. Le défi PASCAL Visual Object Classes (VOC) est une référence en matière de reconnaissance et de détection de catégories d'objets visuels. Il fournit aux communautés de la vision et de l'apprentissage automatique un ensemble de données d'images et d'annotations, ainsi que des procédures d'évaluation standard [68]. Le DPM fut proposé à l'origine par P. Felzenszwalb [69] en 2008 en tant qu'extension du détecteur HOG. Il suit la philosophie de détection "diviser pour régner", où l'entraînement peut

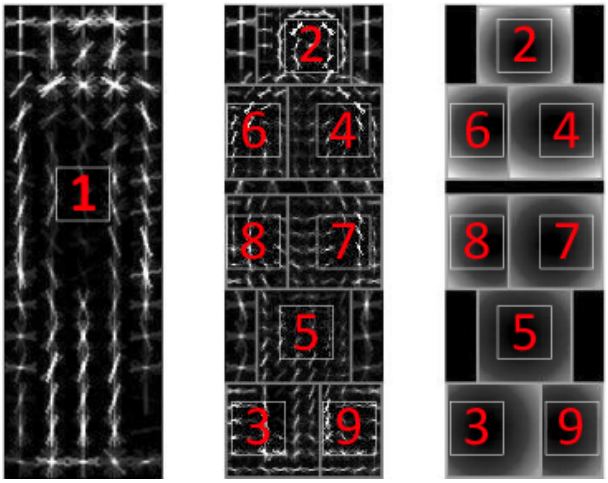


FIGURE 2.15 – DPM modèle pour la détection de piétons [12].

être simplement considéré comme l'apprentissage d'une manière appropriée de décomposer un objet (figure 2.15), et l'inférence peut être considérée comme un ensemble de détections sur différentes parties de l'objet. La figure 2.15 nous montre les étapes, de gauche à droite, pour la détection d'une personne avec le modèle DPM : la définition du modèle de base par un filtre de racine générale, application de filtre de partie de haute résolution et enfin l'application de model spatial pour la localisation de chaque partie de la racine.

Les méthodes sans réseau de neurones pour la détection d'objets nécessitent une extraction préalable des caractéristiques et ensuite une sélection d'algorithme approprié pour la classification et la localisation. Elles se heurtent alors à la complexité ou à la grande variabilité de l'apparence des objets. L'introduction des méthodes d'apprentissage profond a révolutionné la détection d'objets, surtout grâce aux modèles de réseau de neurones convolutionnel (CNN) [70]. Ces modèles peuvent apprendre automatiquement des caractéristiques complexes à partir de données brutes des pixels, éliminant ainsi l'extraction préalable des caractéristiques dans les méthodes sans réseau de neurones par ce qui est appelé un apprentissage profond (deep learning). Lorsqu'ils sont entrainés sur de très grandes bases de données, ces modèles s'avèrent plus performants et plus robustes aux occultations et aux changements d'échelle, d'illumination, de points de vue, etc.

2.3.2 Méthodes par apprentissage profond

Les méthodes par apprentissage profond se basent sur les réseaux de neurones convolutionnels et aussi sur les transformeurs [71]. Ces méthodes se regroupent en deux grandes catégories à savoir : la détection en deux étapes et la détection en une étape. Nous allons d'abord revoir quelques principes des réseaux de neurones convolutionnels, ensuite les méthodes de détection en deux étapes, les méthodes de détection en une étape et, pour finir, les métriques d'évaluation de la détection d'objets.

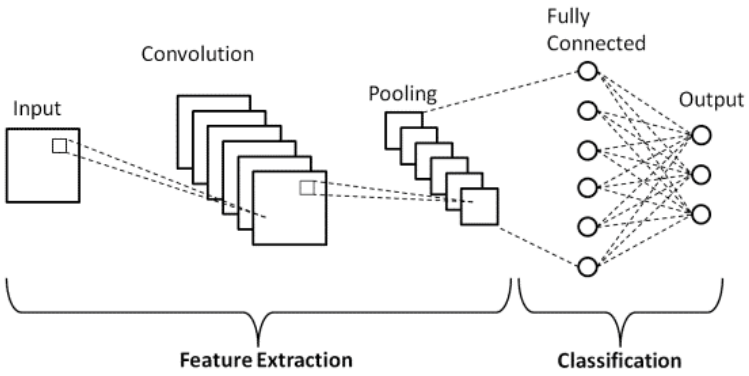


FIGURE 2.16 – Architecture basique d'un réseau de neurones convolutionnel [13].

Réseau de neurones convolutionnel [70]

Un réseau de neurones convolutionnel (CNN) est un type spécialisé de réseau de neurones qui utilise la convolution dans au moins une de ses couches. Il est essentiellement utilisé pour traiter les données telles que les images et les vidéos. Un CNN comporte généralement plusieurs couches, comme on peut l'observer sur la figure 2.16.

1. La couche convolutive : c'est l'élément central du CNN. Elle permet d'extraire des caractéristiques de l'image à travers des opérations de convolution dont les noyaux représentent l'ensemble des paramètres ou poids pouvant être appris. Le noyau est spatialement plus petit qu'une image et représente le champ réceptif de l'image d'entrée. Le noyau glisse sur la hauteur et la largeur de l'image produisant une réponse de ce champ réceptif en chaque position visitée (pixel central) en fonction des poids du noyau. Il en résulte une représentation bidimensionnelle appelée carte d'activation (figure 2.17), pour chaque noyau. L'intervalle de glissement du noyau est appelée "stride" : s'il vaut 1, tous les pixels seront visités, si on augmente l'intervalle, cela produit un sous-échantillonage de l'image. Si nous avons une image d'entrée de taille W x W x C ($C = 3$ pour une image RVB), avec D_{sortie} noyaux de sortie de taille F, un stride S et un padding P (diminution des bords de l'image), la taille de la carte d'activation de sortie sera W_{sortie} x W_{sortie} x D_{sortie}. La taille W_{sortie} est calculée comme suit :

$$W_{sortie} = \frac{W - F + 2P}{S} + 1 \tag{2.1}$$

2. La couche de mise en commun (pooling en anglais) réduit la taille spatiale de la représentation de sortie de la couche de convolution (figure 2.18) en prenant le maximum ou la moyenne d'un sous ensemble de sorties. Si nous disposons d'une carte d'activation de taille W x W x D, d'un noyau de mise en commun de taille spatiale F et d'un "stride" S, la taille du volume de sortie (W_{sortie} x W_{sortie} x D) peut être déterminée par la formule suivante :

$$W_{sortie} = \frac{W - F}{S} + 1 \tag{2.2}$$

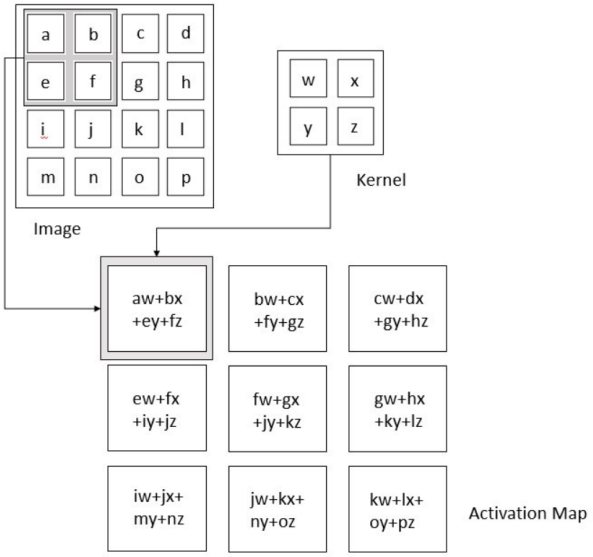


FIGURE 2.17 – Opération de convolution sur une image [14]

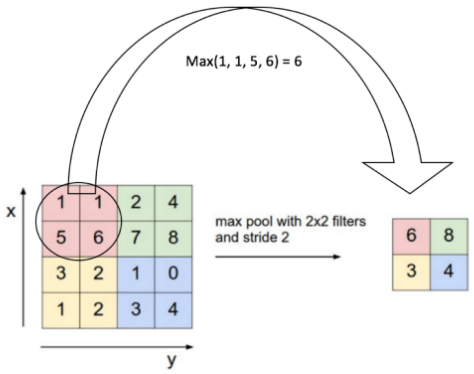


FIGURE 2.18 – Opération de regroupement par le maximum [15]

Un CNN peut enchainer plusieurs couches de convolution-pooling, hiérarchisant une information contenue dans l'image en allant du local au global.

3. La couche entièrement connectée (FC) reliant tous les neurones de sortie avec tous les neurones de la couche précédente permet par exemple, d'affecter une classe d'objets à la représentation extraite par les couches de convolution.

De nombreux modèles CNN ont initialement été conçus et entraînés sur de vastes bases d'images publiques comme ImageNet pour la classification d'images. Parmi les plus connus, on peut citer AlexNet [72], Inception [73], VGG [74], ResNet [75] et MobileNet [76]. La partie encodeur de caractéristiques de ces réseaux pré-entraînés (couches de convolutions successives) a ensuite été réutilisée pour d'autres tâches de vision comme la détection ou la segmentation d'objets.

Détection en deux étapes

En 2014, R. Girshick et al. [77] proposent le R-CNN (Regions with CNN features), dont le principe repose sur une détection d'objets en deux étapes : la première étape consiste à générer des régions d'intérêt de l'image susceptibles de contenir un objet potentiel. La seconde détermine si chaque région contient ou pas un objet et prédit sa classe le cas échéant. L'extraction d'un ensemble de propositions d'objets ou de boîtes d'objets candidats est réalisée par l'algorithme de recherche sélective [78]. L'algorithme de recherche sélective génère des sous-segmentations de l'image qui pourraient appartenir à un objet (sur la base de la couleur, de la texture, de la taille et de la forme) et combine itérativement des régions similaires pour former des objets. Il en résulte des "propositions d'objets" à différentes échelles. Ensuite, chaque proposition est redimensionnée en une image de taille fixe et introduite dans un modèle CNN pré-entraîné sur ImageNet (par exemple, AlexNet [72]) pour extraire les caractéristiques. Enfin, le classifieur SVM [79] et la régression linéaire [80] sont utilisés pour prédire, d'une part la présence d'un objet dans chaque région ainsi que sa catégorie et, d'autre part, les coordonnées des boîtes englobantes. R-CNN apporte une amélioration significative des performances sur le défis VOC-07 par rapport aux méthodes traditionnelles, avec une amélioration importante de la précision moyenne (mAP), qui passe de 33,7% (pour le DPM-v5 [81]) à 58,5%. Bien que le R-CNN ait fait de grands progrès, il présente quelques limitations à prendre en compte : les calculs redondants des caractéristiques sur un grand nombre de propositions chevauchantes (plus de 2000 boîtes pour une image) conduisent à une vitesse de détection extrêmement lente (14 secondes par image avec un GPU).

La même année, K. He et al. [16] proposent le SPPNet afin d'améliorer la vitesse de détection. Les modèles CNN précédents requièrent souvent une taille fixe d'image en entrée. La principale contribution de SPPNet est l'introduction d'une couche Spatial Pyramid Pooling (SPP), qui permet à un CNN de générer une représentation de longueur fixe quelle que soit la taille de l'image sans la remettre à l'échelle. La couche SPP divise la carte des caractéristiques d'entrée en un ensemble de grilles de taille fixe à différentes échelles (figure 2.19). Pour chaque grille, elle calcule la valeur maximale (ou moyenne) de chaque bande de la carte de caractéristiques à l'intérieur de cette grille. Les résultats de ces opérations de regroupement sont concaténés pour créer un vecteur de caractéristiques de taille fixe, ce qui permet d'éviter le calcul répété des caractéristiques par le CNN. SPPNet est 20 fois plus rapide que le R-CNN sans sacrifier la précision de la détection (défis VOC-07 mAP=59,2%). Bien que SPPNet ait effectivement amélioré la vitesse de détection, il présente encore quelques inconvénients : premièrement, l'apprentissage se fait toujours en plusieurs étapes, deuxièmement, SPPNet ne fait qu'affiner ses couches entièrement connectées tout en ignorant simplement toutes les couches précédentes.

En 2015, R. Girshick a proposé le détecteur Fast R-CNN [82], qui constitue une nouvelle amélioration du R-CNN et du SPPNet. Le Fast R-CNN permet d'entraîner simultanément un détecteur et un régresseur de boîte englobante dans les mêmes configurations de réseau. Sur l'ensemble du jeu de données du défi VOC-07, le Fast R-CNN a augmenté le mAP de 58,5% (du R-CNN) à 70,0% tout en ayant une vitesse de détection 200 fois supérieure à celle du R-CNN. Bien que le Fast-R-CNN intègre avec succès les avantages

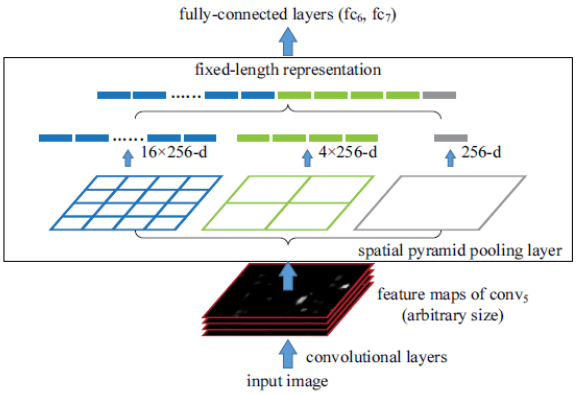


FIGURE 2.19 – Architecture d'une couche SPP [16].

du R-CNN et du SPPNet, sa vitesse de détection est toujours limitée par la proposition de régions car les régions sont toujours proposées séparément, par un autre algorithme.

La même année, S. Ren et al. [17] ont proposé le détecteur Faster R-CNN, peu après le Fast R-CNN. Il s'agit du premier détecteur à base de CNN proche du temps réel (COCO mAP@.5=42.7%, VOC07 mAP=73.2%, 17fps avec ZF-Net [83]). La principale innovation de Faster-R-CNN est l'introduction d'un réseau de propositions de régions (RPN) qui permet de proposer des régions avec un coût de calcul faible. Le réseau RPN est un réseau convolutionnel complet qui glisse sur la carte des caractéristiques (figure 2.20) et qui indique pour chaque position s'il y a un objet ou non, sans tenir compte de la classe de l'objet. Afin d'avoir un système qui soit robuste à la translation et à l'échelle, le RPN utilise un algorithme basé sur les ancres [4] qui sont centrées sur la fenêtre coulissante. Pour chaque position de la fenêtre coulissante sur la carte des caractéristiques, 9 ancres sont placées.

Du R-CNN au faster R-CNN, la plupart des blocs individuels d'un système de détection d'objets, comme la proposition de régions, l'extraction de caractéristiques, la régression de la boîte englobante, la classification, ont été progressivement intégrés dans un système d'apprentissage unifié de bout en bout. Bien que le Faster R-CNN élimine le goulot d'étranglement de la vitesse du Fast R-CNN, il y a toujours une redondance de calcul à l'étape suivante de détection. Par la suite, diverses améliorations ont été proposées, notamment le R-FCN [84] et le Light head R-CNN [85].

En 2017, T.-Y. Lin et al. [86] ont proposé l'architecture FPN (Feature Pyramid Networks) pour améliorer la détection d'objets et d'autres tâches impliquant des représentations de caractéristiques à plusieurs échelles. L'idée principale de FPN est de créer une pyramide de caractéristiques en agrégeant et en fusionnant les caractéristiques de différents niveaux d'un CNN. Cela permet au réseau de détecter des objets à différentes échelles et d'améliorer la précision de la détection d'objets, en particulier pour des objets de tailles et de rapports d'aspect différents. L'utilisation du FPN dans l'architecture Faster R-CNN de base permet d'obtenir des résultats de détection de pointe sur l'ensemble de données COCO. Le FPN est maintenant devenu un élément de base de nombreux détecteurs récents.

4. https ://towardsdatascience.com/anchor-boxes-the-key-to-quality-object-detection-ddf9d612d4f9

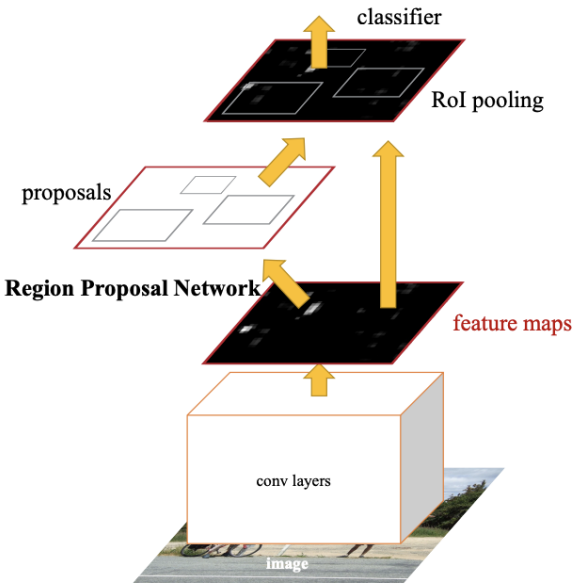


FIGURE 2.20 – Module RPN du Faster R-CNN [17].

Bien que les approches de détection d'objets en deux étapes offrent une grande précision et une meilleure localisation, elles ont tendance à être plus lentes en termes de vitesse d'inférence et complexes à entraîner. Elles sont bien adaptées aux scénarios où la précision est une priorité absolue. Dans la section suivante, nous allons présenter les approches de détection en une étape qui sont plus rapides et arrivent à égaler, voir même surpasser la précision des approches en deux étapes.

Détection en une étape

La détection d'objets en une seule étape fait référence à une classe d'architecture de réseaux de neurones conçues pour détecter des objets dans une image en un seul passage, sans qu'il soit nécessaire de procéder à une étape distincte de génération de propositions comme dans la détection en deux étapes. En effet, la prédiction des classes et des coordonnées des boîtes englobantes se fait directement à partir de l'image d'entrée. Les détecteurs à une étape sont optimisés pour la vitesse et sont souvent utilisés dans des applications en temps réel où la détection rapide d'objets est cruciale. L'une des méthodes de détection en une étape les plus populaires est YOLO (You Only Look Once) proposé par R. Joseph et al. [87] en 2015. Il s'agit du premier détecteur à une étape de l'ère de l'apprentissage profond. YOLO suit un paradigme totalement différent de celui des détecteurs à deux étapes : il applique un seul réseau de neurones à l'ensemble de l'image. Ce réseau divise l'image en plusieurs cellules et prédit les boîtes englobantes et les probabilités de classe simultanément. Les algorithmes YOLO prédisent la boîte englobante par un décalage par rapport à une boîte d'ancrage prédéfinie. Les dimensions de ces boîtes d'ancrages sont calculées sur les boîtes de la vérité terrain annotées sur la base d'images d'apprentissage, en utilisant l'algorithme de clustering K-Means [88]. Malgré leur vitesse, les premières versions de YOLO (YOLOv1 [87] et YOLOv2 [89]), souffraient d'une baisse de la précision, surtout sur les petits objets, par rapport aux

détecteurs à 2 étapes. Pour corriger cela, d'autres versions plus élaborées ont été développées : YOLOv3 [31], YOLOv4 [27], Scaled-YOLOv4 [90]. Récemment, YOLOv7 [91], un travail réalisé dans la continuité de YOLOv4, a été proposé et améliore davantage le détecteur YOLO en terme de précision et de vitesse. Il faut noter que d'autres versions de YOLO existent comme : YOLOv5, YOLOv6 et YOLOv8, développées par l'entreprise Ultralytics [92].

En 2015, W. Liu et al. [93] ont proposé SSD (Single Shot Detection) pour détecter efficacement des objets de tailles et de rapports d'aspect différents dans les images. La principale contribution de SSD est l'introduction des techniques de détection multiréférence et multirésolution lui permettant de détecter des objets de différentes tailles, en particulier les petits objets. En 2017, T.-Y. Lin et al. [94] ont proposé RetinaNet afin d'améliorer la précision des détecteurs à une étape par rapport aux détecteurs à deux étapes. En effet, ils ont constaté que le déséquilibre entre les classes d'avant-plan et d'arrière-plan rencontré lors de l'entraînement des détecteurs denses en est la cause principale. Ainsi, une nouvelle fonction de coût appelée "focal loss" [94] a été introduite dans RetinaNet en remodelant la perte d'entropie croisée standard de manière à ce que le détecteur se concentre davantage sur les exemples difficiles et mal classés au cours de l'entraînement.

Les méthodes présentées ci-dessus utilisent principalement des boîtes d'ancrage prédéfinies pour prédire les coordonnées des boîtes englobantes. Fondamentalement, une boîte d'ancrage est un moyen d'aider le modèle afin qu'il n'ait pas à prédire directement la boîte englobante. Les boîtes d'ancrage sont en fait des paramètres supplémentaires et posent des questions comme : combien d'ancres le modèle doit-il utiliser ? Quelle doit être la taille des ancres ? Ces questions conduisent à plus de réglages d'hyperparamètres et à moins de diversité dans le modèle. Le modèle peut souffrir d'un déséquilibre entre les classes et mettre plus de temps pour converger. Pour résoudre ces problèmes, H. Law et al. proposent le réseau CornerNet [95] en abandonnant le paradigme de détection précédent et considèrent la tâche comme un problème de prédiction des points d'intérêt. En 2019, X. Zhou et al. [96] proposent le CenterNet qui suit également le paradigme de détection basé sur les points d'intérêt, mais élimine les post-traitements coûteux tels que l'affectation des points d'intérêt par groupes et le NMS [5]. CenterNet considère un objet comme un point unique (le centre de l'objet) et régresse tous ses attributs (tels que la taille, l'orientation, l'emplacement, pose, etc.) sur la base du point central de référence. Le modèle peut intégrer la détection d'objets en 3D, l'estimation de la pose humaine, l'apprentissage du flot optique, l'estimation de la profondeur et d'autres tâches dans une seule architecture. En 2021, Zheng Ge et al. [18] ont proposé une version de YOLO dénommée YOLOX où l'utilisation des boîtes d'ancrages est supprimée et quelques techniques avancées de détection sont ajoutées (découplement de la "tête de détection" et attribution de labels par l'algorithme SimOTA). Les modifications ont porté principalement sur l'architecture YOLOv3. Sur la figure 2.21 nous pouvons remarquer les différents changements apportés par YOLOX par rapport à YOLOv3. Dans YOLOX, la "tête de détection" est découplée en 3 têtes, une pour chaque tâche : classification, régression des boîtes englobantes et la confiance (présence d'un objet ou pas).

5. https ://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c

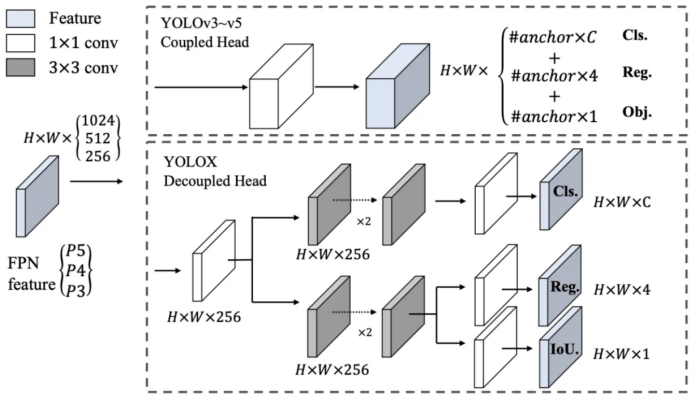


FIGURE 2.21 – Différence entre YOLOX et YOLOv3 [18].

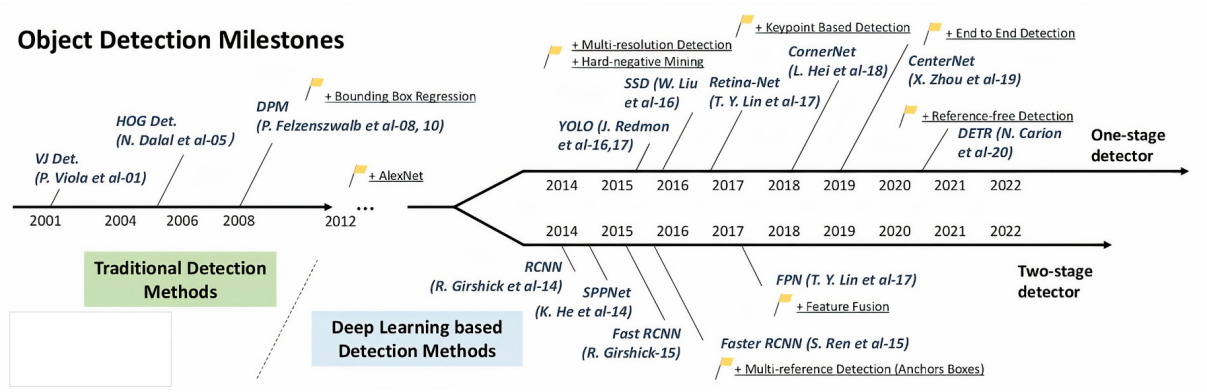


FIGURE 2.22 – Évolution des méthodes de détection d'objets [19].

Ces dernières années, les transformeurs ont profondément influencé l'ensemble du domaine de l'apprentissage profond, en particulier le domaine de la vision par ordinateur. Les transformeurs abandonnent l'opérateur de convolution traditionnel en faveur d'un calcul basé sur l'attention afin de surmonter les limites des CNN et d'obtenir un champ réceptif à l'échelle globale. En 2020, N. Carion et al proposent DETR [97] où la détection d'objets est considérée comme un problème de prédiction d'ensembles. Le DETR est un réseau de détection de bout en bout avec des transformeurs [71]. Plus tard, X. Zhu et al ont proposé le DETR déformable [98] pour remédier au long temps de convergence du DETR et à ses performances limitées en matière de détection des petits objets.

La figure 2.22 nous montre l'évolution des méthodes de détection d'objets au cours de ces vingt dernières années.

2.3.3 Métriques d'évaluation de la détection d'objets

Les performances d'un modèle ou d'un algorithme de détection d'objets sont évaluées en comparant les prédictions du modèle par rapport à la vérité terrain (annotation manuelle correspondant au résultat souhaité). Ces dernières sont représentées sous forme de boîtes englobantes encadrant les objets détectés. Il existent plusieurs métriques standardisées qui proviennent principalement des concours comme PASCAL VOC [68], COCO [99] ou encore Google Open Images Dataset V4 [100]. Avant de présenter ces métriques,

il est nécessaire d'introduire quelques termes techniques qui sont utilisés pour les calculer [101] :

- L'Intersection sur Union (IoU) : il s'agit d'une mesure basée sur l'indice de Jaccard [6] qui évalue le chevauchement entre deux masques binaires ou deux boîtes englobantes (celle de la vérité terrain, Bgt et celle prédite par le modèle, Bp). Pour calculer l'IoU, on utilise la formule suivante :

$$IoU = \frac{aire(B_p \cap B_{gt})}{aire(B_p \cup B_{gt})} \tag{2.3}$$

  Plus l'IoU est grand, plus la prédiction est proche de la vérité terrain. L'IoU permet de déterminer si une détection est valide (vrai positif) ou non (faux positif) en fonction d'un seuil ;
- Vrai positif (TP) : désigne une prédiction de détection correcte ($IoU > seuil$). Le seuil considéré est généralement supérieur à 0.5 ou 50% ;
- Faux positif (FP) : désigne une prédiction de détection incorrecte ($IoU < seuil$) ;
- Faux négatif (FN) : désigne une absence de détection d'un objet présent dans la vérité terrain ;
- Vrai négatif (TN) : la détection d'objets ne prend pas en considération le fond mais seulement les objets d'intérêt, en conséquence, on ne considère pas les vrais négatifs dans l'évaluation de la prédiction ;
- Précision : mesure la capacité d'un modèle à identifier les objets. Il s'agit du pourcentage de prédictions correctes sur le total des prédictions :

$$P = \frac{TP}{TP + FP} = \frac{TP}{all\ detections} \tag{2.4}$$

- Rappel : mesure la capacité d'un modèle à trouver tous les objets corrects. Il s'agit du pourcentage de prédictions correctes sur tous les objets de la vérité terrain. Ci-après la formule de calcul du rappel :

$$R = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths} \tag{2.5}$$

- FPS : il s'agit du nombre d'images traitées par seconde et permet d'évaluer la rapidité d'un algorithme. Il peut varier en fonction du matériel informatique utilisé.

Les termes présentés ci-dessus sont utilisés dans les métriques suivantes d'évaluation de la détection d'objets :

Courbe de précision-rappel (AUC)

Cette métrique permet d'évaluer les performances du détecteur d'objets en fonction de différents seuils de IoU. La courbe est construite pour chaque classe d'objet. Un détecteur d'objets est considéré comme bon si sa prédiction reste élevée lorsque le rappel augmente. Plus l'aire sous la courbe (AUC) est élevée, meilleures seront les performances. Cette affirmation peut être comprise plus intuitivement en examinant

6. https ://fr.wikipedia.org/wiki/Indice_et_distance_de_Jaccard

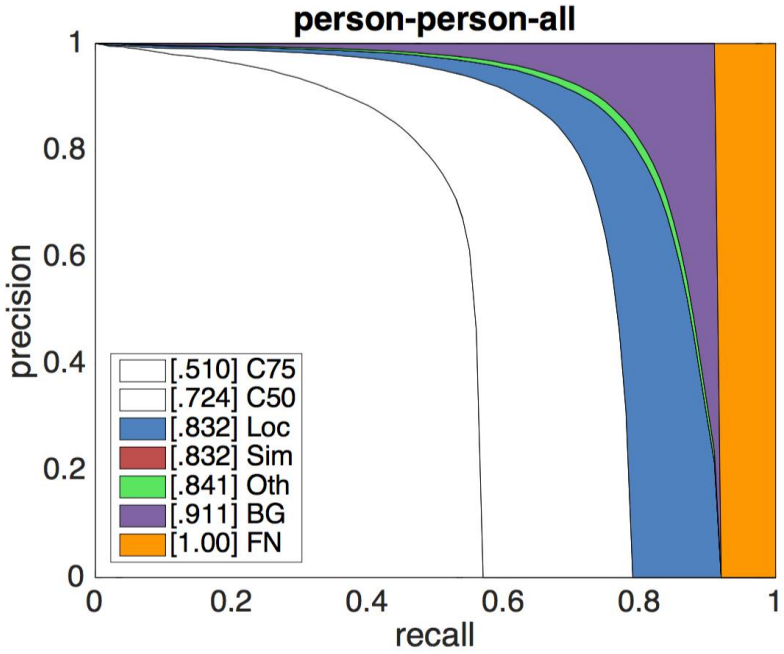


FIGURE 2.23 – Exemple de courbe AUC avec différents seuils [20].

les équations ci-dessus de P (2.4) et R (2.5). En gardant à l'esprit que $TP + FN =$ une constante qui dénombre toutes les détections de la vérité terrain (constante), si le rappel augmente c'est que TP augmente et que FN diminue. La figure 2.23 nous montre un exemple de courbe AUC pour la détection des personnes.

Précision moyenne (AP)

AP est la précision moyenne pour toutes les valeurs de rappel comprises entre 0 et 1. Dans la pratique, il existe deux approches pour calculer l'AP à partir de la courbe AUC pour une classe donnée : l'interpolation en 11 points (PASCAL VOC 2008) et la surface sous la courbe AUC (PASCAL VOC 2012). Dans l'interpolation en 11 points la valeur de rappel est divisée en 11 points (0 à 1.0) et la moyenne de la précision maximale est calculée pour ces 11 valeurs de rappel. Pour la surface de la courbe AUC on calcule la somme des aires des trapèzes formés par les segments de droite reliant des points consécutifs de la courbe à l'axe horizontal.

Ci-dessous nous avons les équations pour le calcul de ces deux approches de AP :

$$AP_{11} = \frac{1}{11} \sum_{R \in \{0, 0.1, \ldots, 0.9, 1\}} P_{interp}(R) \tag{2.6}$$

avec $P_{interp}(R) = maxP(R)$,

$$AP_{all} = \sum_{n} (R_{n+1} - R_n) P_{interp}(R_{n+1}) \tag{2.7}$$

L'équation 2.6 correspond à l'interpolation en 11 points et 2.7 correspond à la surface sous la courbe AUC. L'approche pour le calcul de la précision moyenne utilisée actuellement dans PASCAL VOC est la surface sous la courbe AUC.

A partir de l'AP, nous pouvons calculer également la mAP (Mean Average Precision en anglais) qui correspond à la moyenne de la précision moyenne pour toutes les classes et/ou pour 1 ou plusieurs seuils de IoU. Ci-après nous avons l'équation de calcul du mAP :

$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_i \tag{2.8}$$

avec AP_i étant la AP de la *ieme* classe et N le nombre total de classes.

L'évaluation du mAP des modèles de détection peut être effectuée suivant la modalité du concours PASCAL VOC ou MS COCO. En effet, dans PASCAL VOC la mAP est évaluée en utilisant un seul seuil de IoU (0.5) tandis que dans MS COCO plusieurs seuils sont utilisés (0.5 à 0.95).

Le tableau 2.1 nous montre la comparaison du mAP entre les différentes méthodes de la littérature sur les bases PASCAL VOC et MS COCO.

TABLE 2.1 – Comparaison du mAP des méthodes de détection d'objets sur les données PASCAL VOC et MS COCO [33]

Méthode	CNN Backbone	PASCAL VOC	MS COCO
Détecteurs à une étape			
SSD513	ResNet-101	76.8	31.2
DSSD513	ResNet-101	81.5	33.2
YOLO	∼GoogLeNet	66.4	-
YOLOv2	DarkNet-19	78.6	21.6
YOLOv3	DarkNet-53	-	33.0
YOLOv4	CSPDarkNet-53	-	43.5
YOLOX-L	DarkNet-53	-	50%
CornerNet511	Hourglass-104	-	42.1
RetinaNet	ResNet-101	-	39.1
EfficientDet-D7	BiFPN	-	52.2
DETR-DC5	ResNet-101	-	44.9
Swin Transformer	HTC++	-	58.7
Détecteurs à deux étapes			
SPP-Net	ZF-5	60.9	-
R-CNN	AlexNet	58.5	-
Fast R-CNN	VGG-16	70.0	19.7
Faster R-CNN	VGG-16	73.2	21.9
FPN	ResNet-101	-	36.2
Mask R-CNN	ResNeXt-101	-	39.8
Cascade R-CNN	ResNeXt-101	-	45.8
DetectoRS	ResNeXt-101	-	55.7

2.4 Méthodes de suivi d'objets

Le suivi d'objets est une tâche de vision par ordinateur qui consiste à localiser et à suivre un objet spécifique ou plusieurs objets sur une séquence d'images consécutives dans une vidéo. L'objectif du suivi d'objets est de maintenir un identifiant unique pour chaque objet lorsqu'il se déplace et change d'apparence dans la vidéo. Il existe deux types de suivi : le suivi en ligne qui consiste à traiter les images en temps réel au fur et à mesure qu'elles arrivent, et le suivi hors ligne qui traite l'ensemble de la séquence vidéo après son enregistrement. Pour être efficaces, les algorithmes de suivi d'objets doivent être capables de gérer les problèmes tels que les occultations, les variations d'échelle, les changements d'éclairage, le mouvement flou et les changements d'apparence des objets. Le suivi d'objets multiples (MOT) [7] est une sous-classe du suivi d'objets qui suscite un intérêt croissant en raison de son potentiel académique et commercial. Plusieurs approches existent dans la littératue permettant de faire le suivi d'objets. Nous allons voir ces approches dans cette section en commencant d'abord par les concepts clés du suivi d'objets, puis les algorithmes de suivi sans apprentissage profond, ensuite nous verrons les méthodes à base d'apprentissage profond et nous terminerons par les métriques d'évaluation du suivi d'objets.

2.4.1 Concepts clés du suivi d'objets

Dans le suivi d'objets, il y a trois concepts clés qu'il faut prendre en compte. Le premier concept concerne la représentation de l'objet. Les objets peuvent être représentés à l'aide de boîtes de délimitation, de contours, de points clés ou même de masques sémantiques, en fonction de l'algorithme de suivi et du niveau de détails requis. La figure 2.24 nous montre quelques exemples de représentation d'objets. Le deuxième concept est la modélisation du mouvement qui permet de décrire le comportement attendu des objets dans les images suivantes de la vidéo. Les modèles de mouvement courants supposent souvent la vitesse d'un objet comme étant constante. D'autres modèles plus complexes prennent en compte la dynamique de l'objet. Le troisième concept est le modèle d'apparence qui permet de capturer l'aspect visuel de l'objet et peut être basé sur des histogrammes de couleur, des descripteurs de texture, des caractéristiques profondes de réseaux de neurones convolutionnels (CNN) ou encore une combinaison de ces éléments [21].

2.4.2 Algorithmes de suivi sans apprentissage profond

Ces algorithmes de suivi d'objets sont des techniques qui ont été développées avant l'adoption généralisée de l'apprentissage profond et qui sont basées sur des principes de vision par ordinateur et des approches classiques d'apprentissage automatique. Ces algorithmes sont généralement développés pour suivre des objets dans des vidéos en se basant sur des caractéristiques telles que la couleur, la texture, le mouvement, la forme, etc.

7. https ://motchallenge.net/

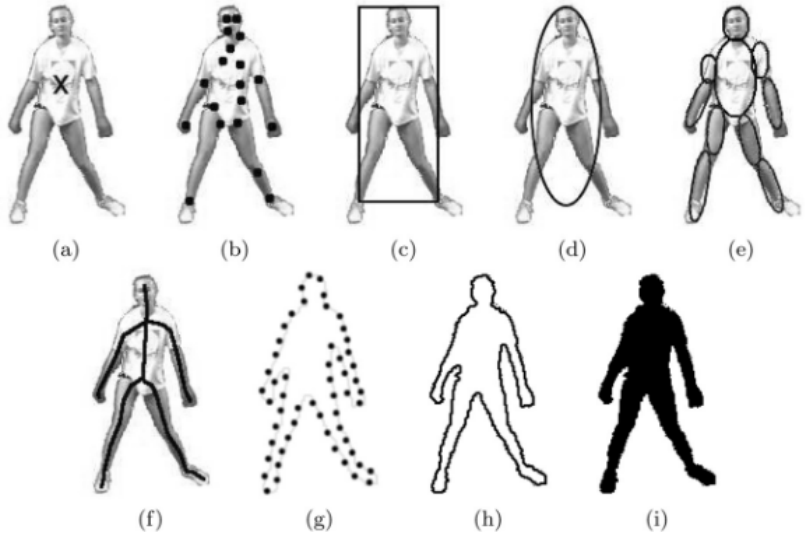


FIGURE 2.24 – Différents exemples de représentation d'objets : (a) par centroïde, (b) par points-d'intérêt, (c) par boîte englobante, (d) par ellipse, (e) par partie, (f) par squelette, (g) par contour en point, (h) par contour plein et (i) par silhouette [21].

Parmi ces algorithmes nous avons :

L'algorithme Mean-Shift [44]

Il est particulièrement connu pour sa capacité à trouver des modes ou des pics dans une fonction de densité de probabilité (PDF) de points de données, ce qui permet de localiser efficacement des clusters ou des régions d'intérêt dans un ensemble de données. Il est largement utilisé pour suivre de manière robuste et rapide l'emplacement d'un objet dans une séquence d'images à l'aide de l'histogramme des couleurs de l'objet. Ido Leichter et al. [102] proposent une version améliorée du Mean-shift utilisant plusieurs histogrammes de référence obtenus à partir de différentes vues de l'objet à suivre. Cette amélioration est réalisée tout en préservant les propriétés de convergence et de rapidité de l'outil de suivi original. Le Mean-shift est un algorithme polyvalent qui fonctionne bien dans les scenarii où les données sont distribuées en clusters ou en régions avec des densités différentes. Cependant, il peut rencontrer des difficultés avec des distributions complexes ou dans les cas où les points de données sont très éloignés les uns des autres. Des extensions telles que le CamShift [103] ont été développées pour améliorer les capacités de Mean-Shift pour le suivi d'objets.

Le filtre de Kalman [104]

Il s'agit d'un algorithme récursif qui estime l'état d'un objet sur la base de mesures bruitées et de modèles de mouvement. Il est utilisé pour le suivi d'objets en combinant les prédictions de mouvement avec les observations courantes pour fournir une estimation plus précise de la position de l'objet. L'idée principale est d'équilibrer les informations provenant des prédictions (dynamique du système) et des mesures, tout en tenant compte des incertitudes associées. Le filtre conserve deux variables, l'estimation

de l'état a posteriori x et la matrice de covariance de l'estimation a posteriori P. Dans la tâche de suivi d'objets, le processus du filtre de Kalman est décrit avec le modèle de transition d'état F, le modèle d'observation H, le bruit de processus Q et le bruit d'observation R. À chaque étape t, compte tenu des observations z_t, le filtre de Kalman alterne les étapes de prédiction et de mise à jour. Dans l'étape de prédiction, les mesures précédentes sont utilisées pour prédire la position actuelle. L'étape de mise à jour utilise les mesures actuelles pour corriger la position prédite précédemment. Ci-après nous avons les équations de calcul de l'état x et de la matrice de covariance P :

$$\text{predict} = \begin{cases} \hat{x}_{t|t-1} &= F_t \hat{x}_{t-1|t-1} \\ P_{t|t-1} &= F_t P_{t-1|t-1} F_t^T + Q'_t \end{cases} \tag{2.9}$$

$$\text{update} = \begin{cases} K_t &= P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1} \\ \hat{x}_{t|t} &= F_t \hat{x}_{t|t-1} + K_t (z_t - H_t \hat{x}_{t|t-1}) \\ P_{t|t} &= (I - K_t H_t) P_{t|t-1} \end{cases} \tag{2.10}$$

Le tableau 3.1 nous présente la description des paramètres des équations précédentes. Xi Chen et al.

TABLE 2.2 – Termes de référence du filtre de Kalman [34]

Symbole	Description	Dimensions
x	Variable d'état	$n \times 1$ vecteur colonne
P	Matrice de covariance de l'état	$n \times n$ matrice
z	Mesures	$m \times 1$ vecteur colonne
F	Matrice de transition des états	$n \times n$ matrice
H	Matrice d'état-mesure	$m \times n$ matrice
R	Matrice de covariance de la mesure	$m \times m$ matrice
Q	Matrice de covariance du bruit	$n \times n$ matrice
K	Gain de Kalman	$n \times m$

[105] proposent une approche pour la détection et le suivi de plusieurs objets en mouvement avec des occultations en se basant sur le filtre de Kalman. Hitesh A. Patel et al. [106] ont également utilisé le filtre de Kalman pour le suivi d'un objet en mouvement. SORT [107] et Deep SORT [108], deux méthodes de suivi très populaires, emploient aussi le filtre de Kalman pour l'estimation du mouvement. L'une des limites du filtre de Kalman est l'hypothèse selon laquelle la dynamique du système et celle des modèles de mesure sont linéaires et que le bruit est gaussien [8]. En réalité, de nombreux systèmes réels sont non linéaires et l'hypothèse d'un bruit gaussien peut ne pas se vérifier, ce qui peut conduire à des résultats sous-optimaux. Une version étendue du filtre de Kalman [109], non linéaire, a été proposée pour corriger cela. Cette limitation est également relevée par le filtre particulaire [110].

Le filtre particulaire [110]

Il s'agit d'une méthode récursive de calcul statistique de Monte Carlo [9] qui est souvent utilisée pour les modèles de mesure du bruit non gaussien. Dans le filtre particulaire la densité d'état conditionnelle

8. https ://fr.wikipedia.org/wiki/Loi_normale
9. https ://en.wikipedia.org/wiki/Monte_Carlo_method

$p(X_t|Z_t)$ à l'instant t est représentée par un ensemble d'échantillons $\{s_t^{(n)} : n = 1, ..., N\}$ (particules). Chaque particule a un poids de probabilité $\pi_t^{(n)}$, qui représente la probabilité d'échantillonnage de cette particule basée sur la fonction de probabilité de densité. Dans [111], les auteurs traitent en détail de l'application du filtre particulaire pour le suivi visuel, dans le cas d'un seul objet et de plusieurs objets.

Les filtres à corrélation [112]

Ils sont également utilisés dans le suivi d'objets. Ils apprennent l'apparence de l'objet et utilisent des opérations convolutives pour suivre l'objet dans les images suivantes. Deux types populaires de filtres de corrélation sont le filtre MOSSE [113] "Minimum Output Sum of Squared Error" et le filtre DCF [114] "Discriminative Correlation Filter". Henriques et al. [115] ont proposé le filtre de corrélation à noyau (KCF) qui ajuste davantage les caractéristiques du canal aux caractéristiques multicanaux et introduit des caractéristiques CNN pour le suivi. Tianzhu Zhang et al. [116] proposent une méthode de suivi d'objets en combinant le filtre de corrélation multi-tâche et le filtre particulaire (MCPF).

Bien qu'ils n'atteignent pas toujours le même niveau de précision que les méthodes modernes d'apprentissage profond, ces algorithmes restent pertinents et utiles pour divers scenarii de suivi.

2.4.3 Méthodes de suivi d'objets par apprentissage profond

La plupart des méthodes modernes de suivi d'objets s'appuient sur des techniques d'apprentissage profond ou des combinaisons de ces techniques avec les algorithmes traditionnels pour atteindre une meilleure précision. Ces dernières années, de nombreuses méthodes de suivi d'objets multiples (MOT [10]) ont été proposées pour résoudre les problèmes de suivi existants, tels que le suivi en temps réel, le changement d'identifiant et l'occultation. Une méthode de suivi d'objets multiples comprend généralement 4 étapes :

- la détection : pour localiser l'objet cible, par une boîte englobante par exemple, dans chaque image en utilisant un modèle pré-entraîné de détection d'objet ;
- l'estimation du mouvement : consiste à estimer la prochaine position de l'objet dans les images suivantes, en utilisant par exemple un algorithme d'estimation d'état d'un objet ;
- l'association : consiste à associer les objets détectés dans l'image suivante avec l'estimation obtenue à l'étape précédente et ainsi maintenir un identifiant unique pour cet objet ;
- la création et la suppression des identifiants de suivi : lorsque des objets entrent et sortent de l'image, des identifiants uniques doivent être créés ou détruits en conséquence.

Certaines méthodes essayent de regrouper ces quatres étapes en trois ou deux étapes en combinant plusieurs architectures, d'autres proposent une seule architecture permettant de regrouper ces quatres étapes en une seule et faire le suivi de bout en bout. Ces méthodes peuvent être classées, en fonction du paradigme utilisé, en plusieurs groupes. Parmis ces groupes, nous avons :

10. https ://paperswithcode.com/task/multi-object-tracking

Suivi par détection

Dans cette approche, le suivi des objets découle de leur détection dans chaque image sous forme d'une boîte englobante ; une seconde étape permet d'associer les détections qui correspondent au même objet dans les images successives. Dans ce groupe nous avons la méthode SORT (Simple Online Real-time Tracking) proposée par Alex Bewley et al. [107] en 2017. Il s'agit d'une approche pragmatique du suivi d'objets multiples dont l'objectif principal est d'associer des objets de manière efficace pour des applications en ligne et en temps réel. Bien qu'elle n'utilise qu'une combinaison simple de techniques familières telles que le filtre de Kalman [104] et l'algorithme Hongrois [117] pour les composants de suivi, cette approche permet d'obtenir une précision comparable à celle des systèmes de suivi en ligne les plus modernes. Nicolai Wojke et al. [108] proposent une version améliorée de SORT dénommée Deep SORT en ajoutant l'information d'apparence. En effet dans SORT, l'association s'effectue à base d'une métrique donnée de distance (distance IoU) entre les boîtes englobantes des objets suivis. L'association est résolue de manière optimale à l'aide de l'algorithme Hongrois. L'algorithme hongrois, également connu sous le nom d'algorithme de Kuhn-Munkres, est utilisé pour résoudre le problème d'affectation dans le cadre de l'optimisation combinatoire. Le problème d'affectation consiste à trouver l'affectation optimale d'un ensemble de tâches à un ensemble d'agents, en minimisant le coût total ou en maximisant le bénéfice total. Il est utilisé dans le suivi pour affecter les nouveaux objets détectés aux objets existants en se basant sur une matrice de coûts de distance entre ces objets. Dans Deep SORT, en plus d'avoir la métrique de distance IoU, une métrique d'apparence basée sur des caractéristiques CNN est ajoutée. Cette nouvelle métrique d'association est entraînée hors ligne par un modèle CNN. Grâce à cette extension, il est possible de suivre les objets pendant de plus longues périodes d'occultation, ce qui permet de réduire efficacement le nombre de changements d'identité. Cependant, le temps de traitement augmente à cause de l'extraction des descripteurs d'apparence inopportuns. Récemment, de nouvelles versions plus performantes de SORT et Deep SORT ont été proposées respectivement par Jinkun Cao et al. [118], Yunhao Du et al. [119]. Yongyi Lu et al. [120] proposent également une méthode de suivi par détection en utilisant l'architecture LSTM [121] pour effectuer l'association et permettre l'exploitations de riches informations temporelles inhérentes aux données vidéo. Ces méthodes présentent deux inconvénients. Premièrement, l'association de données ne tient pas compte des caractéristiques d'apparence de l'image ou nécessite un extracteur de caractéristiques coûteux en termes de calcul. Deuxièmement, la détection est séparée du suivi. Yifu Zhang et al. proposent la méthode FairMOT [122] qui permet d'entraîner conjointement la détection et la réidentification dans une seule architecture pour le suivi.

Suivi par segmentation

Dans cette approche, une segmentation des objets à suivre est préalablement appliquée image-par-image ; le suivi s'opère sur les masques de segmentation. L'objectif est de fournir une représentation plus précise de la forme et des limites de l'objet. La segmentation sémantique [11] et la segmentation par

11. https ://www.jeremyjordan.me/semantic-segmentation/

instance [12] sont souvent employées dans cette approche. Elle est particulièrement efficace pour le suivi d'objets ayant des formes complexes, des apparences variables et pour les occultations. Aljosa Osep et al. [123] proposent une méthode de suivi d'objets multiples basée sur la segmentation d'image agnostique par catégorie. Matthieu Paul et al. [124] proposent un pipeline de suivi centré sur la segmentation produisant un masque de segmentation très précis. Les méthodes de segmentation sont souvent plus gourmandes en ressources informatiques. Cela peut entraîner une augmentation du temps de traitement, ce qui rend le suivi en temps réel plus difficile, en particulier sur les appareils dont les ressources sont limitées.

Suivi par régression

Dans cette approche, le suivi est formulé sous la forme d'une régression des coordonnées des boîtes englobantes pour prédire la position de l'objet dans l'image suivante à partir des coordonnées précédentes. Ainsi il n'est plus nécessaire d'associer les détections entre les images. C'est dans ce contexte que Christoph Feichtenhofer et al. [22] proposent le "Detect to Track and Track to Detect (D&T)" qui utilise le modèle de détection d'objets R-FCN [84] et l'étend à la détection et au suivi d'objets multiples en prenant en entrée deux images consécutives. Les entrées passent d'abord dans un réseau entièrement convolutif pour produire des cartes de caractéristiques. Une couche de corrélation opère sur les cartes de caractéristiques à différentes échelles (figure 2.25) et calcule des cartes de corrélation pour toutes les positions dans la carte de caractéristiques. Un regroupement de régions d'intérêt (ROI pooling) est appliqué sur ces cartes de corrélation pour la régression inter-images des boites englobantes permettant d'obtenir les décalages des coordonnées des boîtes englobantes $(\Delta_x, \Delta_y, \Delta_w, \Delta_h)$ entre l'image t et l'image $t+1$ dans les deux images $\{I^t, I^{t+1}\}$. Pour chaque image on obtient donc les coordonnées des détections et leur décalage par rapport à l'image précédente. Un score de liaison par classe est défini pour combiner les détections et les décalages dans le temps. L'architecture D&T peut être gourmande en ressources informatiques, car elle exige que la détection, basée sur une approche en deux étapes, et le suivi des objets soient effectués en temps réel.

Afin de simplifier l'architecture de détection et de suivi simultané, Philipp Bergmann et al. [125] proposent le Tracktor en exploitant la branche de régression d'un détecteur pour effectuer un réalignement temporel des boîtes englobantes des objets. Dans un premier temps, un regroupement de régions d'intérêt (ROI pooling) est appliqué sur les caractéristiques de l'image actuelle mais avec les coordonnées des boîtes englobantes précédentes, afin d'étendre les trajectoires actives (objets existants dans l'image précédentes) de l'image $t-1$ à l'image t actuelle en supposant que la cible se déplace légèrement entre les images. De cette façon, les identifiants des objets sont propagés d'image en image jusqu'à la fin de la séquence. Afin de tenir compte des nouvelles cibles, le détecteur d'objets fournit également les détections pour l'ensemble de l'image t. Tracktor dispose de deux extensions : un réseau siamois de réidentification et un modèle de mouvement, pour gérer les cas complexes et les occultations de longue durée. Cette architecture est basée principalement sur un détecteur à 2 étapes, ce qui le rend un peu gourmand en calcul pour le temps réel.

12. https ://blog.roboflow.com/difference-semantic-segmentation-instance-segmentation/

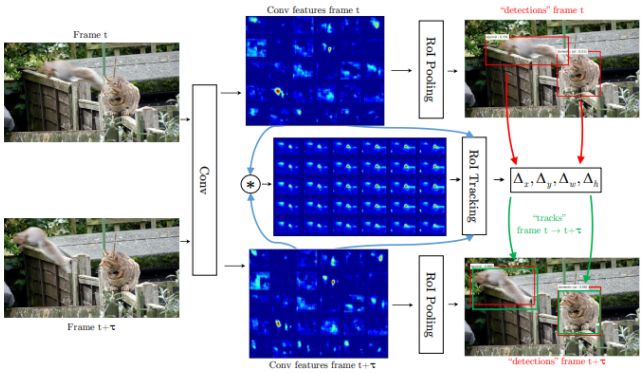


FIGURE 2.25 – Fonctionnement de l'architecture D&T [22].

Aussi pour gérer les cas de longues occultations il faut ajouter les deux extensions supplémentaires, ce qui peut augmenter la complexité de l'architecture.

Pour faire très simple, Xingyi Zhou et al. [23] proposent le "CenterTrack" pour suivre les objets comme des points. CenterTrack applique un modèle de détection pour localiser les objets et prédire leurs associations avec l'image précédente en utilisant le décalage entre le centre des objets. Pour cela, ils modifient le détecteur CenterNet pour prendre en entrée deux images consécutives et également les résultats précédents de détections (sous forme de carte de chaleur "heatmap"). La nouvelle architecture est entraînée de bout en bout pour prédire trois sorties : les détections sur l'image actuelle, le décalage avec les détections précédentes et la "heatmap" des détections courantes (figure 2.26). Pour associer les détections dans le temps, CenterTrack utilise le décalage en 2D entre les centres (x, y) des objets de l'image courante et ceux de l'image précédente. Dans CenterTrack, lorsqu'un objet quitte l'image ou est occulté et réapparaît, une nouvelle identité lui est attribuée. Cela limite ses capacités pour gérer les longues périodes d'occultation.

Suivi par attention

C'est une approche qui exploite les mécanismes d'attention [71] pour suivre les objets. Chu et al. [126] ont introduit un mécanisme d'attention spatio-temporelle (STAM) pour gérer la perte de suivi causée par l'occultation et l'interaction entre les objets. De même, Zhu et al. [127] ont proposé un "Dual Matching Attention Networks" (DMAN) avec des mécanismes d'attention spatiale et temporelle pour effectuer l'association des données.

Les méthodes existantes de suivi d'objets multiples (MOT) suivent le paradigme du suivi par détection pour effectuer séparément la détection des objets, l'extraction des caractéristiques et l'association des données, ou intègrent deux des trois sous-tâches pour former une solution partiellement de bout en bout. Jinlong Peng et al. [24] proposent un modèle en ligne simple appelé Chained-Tracker (CTracker), qui intègre les trois sous-tâches dans une solution de bout en bout. Un module d'attention conjoint utilisant des cartes de confiance prédites est ajouté à l'architecture afin d'améliorer encore ses performances (figure 2.27). En 2021, Peize Sun et al. [25] ont proposé le TransTrack en s'appuyant sur l'architecture du

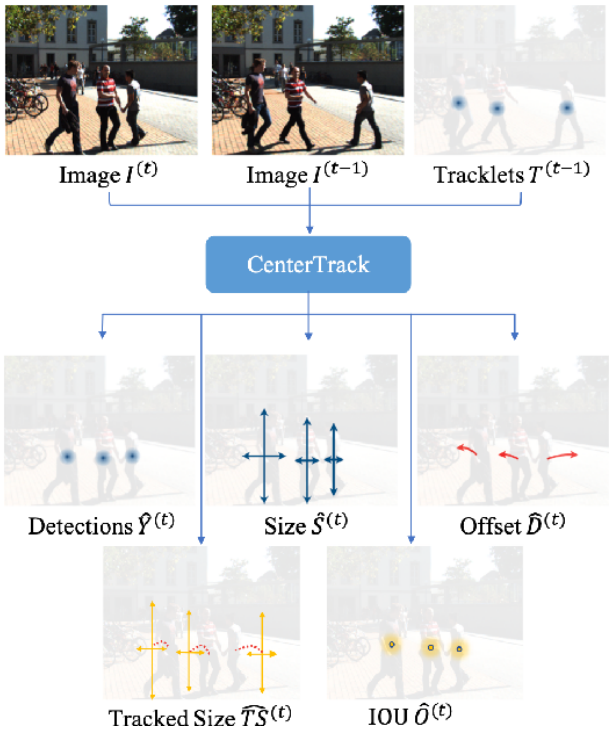


FIGURE 2.26 – Fonctionnement de l'architecture CenterTrack [23].

transformeur [71], qui est un mécanisme de clé de requête basé sur l'attention. Il créé un nouveau paradigme de détection et de suivi conjoints en réalisant la détection et l'association d'objets en une seule fois. Dans TransTrack, les requêtes de caractéristiques d'objets de l'image précédente et les requêtes d'objets apprises sont prises en entrée. Les cartes des caractéristiques de l'image constituent une clé partagée. La requête d'objet apprise détecte les objets dans l'image actuelle. La requête de suivi de l'image précédente associe les objets de l'image actuelle aux objets précédents (figure 2.28). Ce processus est exécuté séquentiellement sur toutes les images adjacentes et complète finalement la tâche de suivi d'objets multiples. Récemment, des amélorations du TransTrack ont été proposées : TrackFormer [128] et MOTR [129] afin d'obtenir une architecture de bout en bout plus efficace pour le suivi d'objets.

2.4.4 Métriques d'évaluation du suivi d'objets

Pour évaluer les performances des algorithmes ou méthodes de suivi d'objets, sur les séquences vidéos, certaines métriques sont utilisées pour quantifier la précision, la robustesse et l'efficacité du suivi [130]. Une plate-forme dénommée le MOTChallenge [13] est mise en place afin de faire une comparaison équitable des méthodes de suivi les plus modernes. En effet, cette plate-forme fournit des données de vérité terrain standardisées, des mesures d'évaluation, des scripts, ainsi qu'un ensemble de modèles de détection pré-entraîné pour évaluer les méthodes de suivi plus rapidement. Parmis ces métriques nous avons :

13. https ://motchallenge.net/

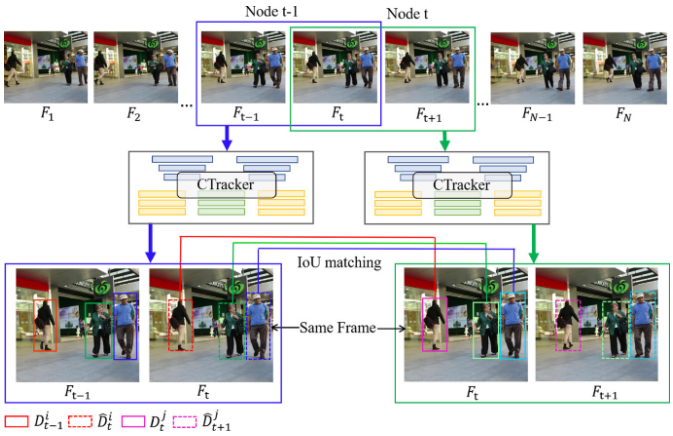


FIGURE 2.27 – Fonctionnement de l'architecture CTracker [24].

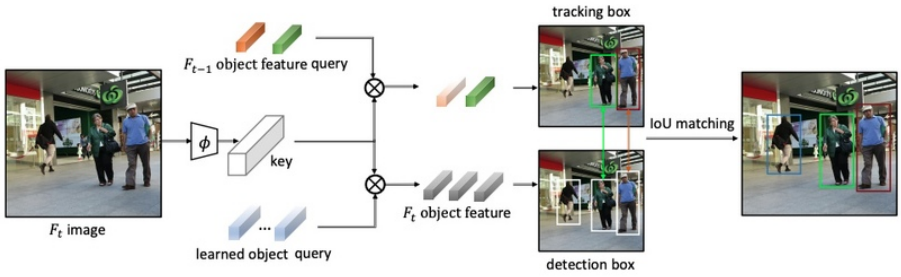


FIGURE 2.28 – Fonctionnement de l'architecture TransTrack [25].

MOTP

Il s'agit d'une métrique d'évaluation de la précision du suivi d'objets (Multiple Object Tracking Precision en anglais) et permet de mesurer la dissimilarité moyenne entre tous les vrais positifs et les objets de la vérité terrain correspondants. Ci-après la formule de calcul du MOTP pour un vrai positif :

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{2.11}$$

avec c_t le nombre de correspondances entre la vérité terrain et le résultat de la détection dans l'image t, et $d_{t,i}$ est le chevauchement (IoU) de la boîte englobante de la cible i avec l'objet de vérité terrain qui lui est assigné dans l'image t. MOTP donne donc le recouvrement moyen entre les prédictions correctes et leur verité terrain correspondante. Cette métrique quantifie la précision de localisation du détecteur et fournit donc peu d'informations sur les performances réelles du suivi.

MOTA

MOTA (Multiple Object Tracking Accuracy) est une métrique plus complète qui prend en compte trois types de mesure pour évaluer les performances du suivi, notamment les faux positifs (FP), les faux négatifs (FN) et les changements d'identité (IDSW). Pour la calculer, nous avons l'équation suivante :

$$MOTA = 1 - \frac{\sum_t (FP_t + FN_t + IDSW_t)}{\sum_t GT_t} \tag{2.12}$$

avec GT le nombre de vérité terrain des boîtes englobantes, FN le nombre de faux négatifs (nombre de vérité terrain non détectée), FP le nombre de faux positifs (le nombre de détections incorrectes) et IDSW le nombre de changement des IDs c'est-à-dire le nombre de fois qu'une trajectoire donnée change d'un objet de vérité terrain à un autre. Il faut noter que MOTA peut également être négative dans les cas où le nombre d'erreurs commises par l'algorithme de suivi dépasse le nombre d'objets présents dans la scène.

Précision de l'identification, Rappel de l'identification et Score F1 (IDF1)

La métrique IDF1 est une mesure basée sur l'identité qui met l'accent sur la capacité de préservation de l'identité de l'objet suivi sur l'ensemble de la séquence. Dans ce cas, la correspondance entre les prédictions et la vérité terrain est établie en résolvant un problème de correspondance bipartite, en reliant les paires ayant le plus grand chevauchement temporel. Une fois la correspondance établie, on peut calculer le nombre d'identifiants vrais positifs (IDTP), le nombre d'identifiants faux positifs (IDFP) et le nombre d'identifiants faux négatifs (IDFN). A partir de ces mesures, la précision de l'identification peut être calculée comme suit :

$$IDP = \frac{IDTP}{IDTP + IDFP} \tag{2.13}$$

et le rappel de l'identification :

$$IDR = \frac{IDTP}{IDTP + IDFN} \tag{2.14}$$

Il faut noter que IDP et IDR représentent la fraction des détections calculées (vérité terrain) qui sont correctement identifiées. L'IDF1 est ensuite exprimé comme un rapport entre les détections correctement identifiées ; et le nombre moyen de détections réelles et calculées :

$$IDF1 = \frac{2 \cdot IDTP}{2 \cdot IDTP + IDFP + IDFN} \tag{2.15}$$

Mesures de la qualité du suivi

Il s'agit des mesures qualitatives permettant d'évaluer le pourcentage de trajectoire de vérité terrain que les algorithmes de suivi arrivent à recouvrir. Ainsi, chaque vérité terrain peut être classée comme : MT(majoritairement suivie, environ 80% de sa durée), ML (majoritairement perdue, environ 20% de sa durée) et PT (partiellement suivie, entre 20 et 80% de sa durée). Un plus grand nombre de MT et moins de ML sont souhaitables. Dans certaines situations, il peut être intéressant d'obtenir les trajectoires longues et persistantes sans interruption. Pour cela, le FM fut introduit et permet de compter le nombre

de fragmentation de la trajectoire c'est-à-dire le nombre de fois où une trajectoire de vérité terrain est interrompue.

HOTA

HOTA (High Order Tracking Accuracy) est une nouvelle métrique développée en 2021 afin de palier aux limitations des métriques précédentes qui accordent trop d'importance soit à la détection, soit à l'association. Elle équilibre explicitement l'effet d'une détection, d'une association et d'une localisation précises en une seule mesure unifiée pour comparer les algorithmes de suivi. HOTA se décompose en une famille de sous-métriques qui sont capables d'évaluer séparément chaque type d'erreur de base, ce qui permet une analyse claire des performances de suivi.

HOTA peut être considérée comme une combinaison de trois scores IoU. Il divise la tâche d'évaluation du suivi en trois sous-tâches (détection, association et localisation) et calcule un score pour chacune d'entre elles à l'aide de l'IoU :

– **La localisation** : elle mesure l'alignement spatial entre une détection prédite (TP) et une détection de la vérité terrain. Ainsi on peut calculer la précision de la localisation comme suit :

$$LocA = \frac{1}{|TP|} \sum_{c \in TP} LocIoU(c) \tag{2.16}$$

avec $LocIoU(c)$ l'IoU entre la prédiction correcte (c) et la vérité terrain correspondante.

– **La détection** : elle mesure l'alignement entre l'ensemble des détections prédites et l'ensemble des détections de la vérité terrain. Il s'agit ici de définir quelles sont les détections qui se croisent entre l'ensemble des détections prédites et les détections de vérité terrain (détection avec $LocIoU > 0.5$ par exemple). L'algorithme Hongrois est utilisé pour réaliser la correspondance unique entre les détections prédites et les détections de la vérité terrain. On peut alors calculer la précision de la détection comme suit :

$$DetA = \frac{|TP|}{|TP| + |FP| + |FN|} \tag{2.17}$$

– **L'association** : elle mesure l'efficacité avec laquelle un algorithme de suivi relie les détections dans le temps aux mêmes identités (ID), compte tenu de l'ensemble des liens d'identité dans les trajectoires de la vérité terrain. Pour cela, on prend une détection prédite et une détection de la vérité terrain qui sont appariées ensemble (en utilisant l'algorithme Hongrois), et on mesure l'alignement entre la trajectoire entière de cette détection prédite et la trajectoire entière de la détection de vérité terrain. Ainsi nous avons le TPA qui correspond au nombre de correspondances vraies positives entre les deux trajectoires ; FPA qui correspond au nombre de trajectoires qui ne sont pas dans la vérité terrain ; toutes les détections restantes dans la trajectoire de la vérité terrain sont des FNA (faux négatif associations). On peut calculer ainsi la précision de l'association avec l'équation suivante :

TABLE 2.3 – Comparaison de quelques méthodes modernes de suivi d'objets sur les données MOT17 privées

Méthode	MOTA(↑)	IDF1(↑)	MT(↑)	ML(↓)	IDSW(↓)	HOTA(↑)	FPS(↑)
CenterTrack	67.8	64.7	34.6	24.6	3039	52.2	17.5
FairMOT	73.7	72.3	43.2	17.3	3303	59.3	25.9
CTracker	66.6	57.4	32.2	24.2	5529	-	6.8
TransTrack	74.5	63.9	46.8	11.3	3663	54.1	10
TrackFormer	74.1	68.0	47.3	10.4	2829	-	7.4
SORT	43.1	39.8	11.5	35.3	718	34.0	143
Deep SORT	60.3	61.2	31.5	20.3	2442	61.2	6.4
ByteTrack [131]	80.3	77.3	53.2	14.5	2196	63.1	30
MOTR	73.4	68.6	50.3	13.1	2439	57.8	7.5
OC-SORT	78	77.5	41	20.9	1950	52.4	700
StrongSORT++	79.6	79.5	53.6	13.9	1194	63.5	7.1

$$AssA = \frac{1}{|TP|} \sum_{c \in TP} \frac{|TPA(c)|}{|TPA(c)| + |FPA(c)| + |FNA(c)|} \tag{2.18}$$

En combinant les trois scores de l'IoU définis ci-dessus, nous pouvons calculer la HOTA comme suit :

$$HOTA_\alpha = \sqrt{DetA_\alpha \cdot AssA_\alpha} \tag{2.19}$$

avec α le seuil d'IoU du $LocIoU$.

Le tableau 2.3 nous montre la comparaison de quelques méthodes de la littérature avec les métriques présentées précédemment et sur la base de test du MOT17 [14] qui contient une seule classe (personne) et 7 séquences de 17757 images au total. On remarque que certaines méthodes dont MOTA ou HOTA sont élevées (StrongSORT par exemple), ont un temps de traitement plus lent (faible FPS) qui peut s'expliquer par leur architecture un peu complexe, sauf la méthode OC-SORT [118]. En cherchant à avoir une bonne précision dans le suivi, on peut augmenter la complexité de l'architecture et son temps de traitement.

2.5 Conclusion

Dans ce chapitre, nous avons présenté les méthodes de la littérature permettant le suivi du bétail à base de capteur vidéo (vidéosurveillance), dans un environnement d'élevage clos. La plupart de ces méthodes se base sur la détection et le suivi d'objets. Pour la détection, plusieurs méthode ont été testées dans les 2 types de détection (en une étape et en deux étapes) de la littérature. Le choix de l'une de ces méthodes dépendra de l'objectif visé en terme de complexité de l'architecture, de la précision et aussi de la rapidité. Les méthodes de détection d'objets en une étape sont plus rapides et plus adaptées au temps réel par rapport aux méthodes en deux étapes. Pour le suivi, il existe également, dans la littérature, plusieurs méthodes allant des architectures en plusieurs étapes (SORT et Deep SORT par exemple),

14. https ://motchallenge.net/data/MOT17/

simple et rapides, aux architectures de bout en bout, plus complexes, intégrant la détection et le suivi dans un seul réseau.

Ces méthodes, bien qu'elles soient performantes, présentent également quelques limitations. Pour la détection par exemple, la rapidité de détection est loin d'égaler celle de l'homme ; une faible performance de détection des petits objets dans de grandes scènes ; des architectures souvent basées sur une affectation de un à plusieurs et nécessitant une étape supplémentaire de NMS (suppression non maxiamle) [15] pour filtrer les détections. La plupart des détecteurs d'objets sont généralement conçus pour la détection au niveau de l'image et ignorent les corrélations entre les images vidéo [19]. Pour le suivi également il reste encore des progrès à faire compte tenu de la complexité de cette tâche (variation dans les scènes, longue occultation, variation d'apparence, application en temps réel, etc.).

Il serait interessant d'expérimenter quelques unes de ces méthodes pour l'analyse du comportement chez les chèvres afin de détecter leurs limites, face à nos données, et proposer d'autres approches pour les améliorer.

15. https ://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/

Chapitre 3

Détection et suivi des chèvres

3.1 Introduction

Ce chapitre est consacré principalement aux expérimentations effectuées pour la détection et le suivi des chèvres. Cette étape est effectuée préalablement à l'analyse du comportement et la reconnaissance d'actions. Sur la figure 3.1, nous présentons l'architecture adoptée pour notre système de détection et suivi des chèvres. Nous effectuons la détection sur chaque image de la vidéo, puis chaque individu est suivi dans la séquence par une estimation de sa position dans l'image suivante et un algorithme d'association afin de conserver un unique identifiant par animal. La section suivante précisera les conditions d'acquisition de nos données et la méthode adoptée pour constituer nos bases d'apprentissage et de test. Puis les algorithmes de détection et suivi retenus pour nos tests seront présentés avant d'exposer l'analyse des résultats obtenus.

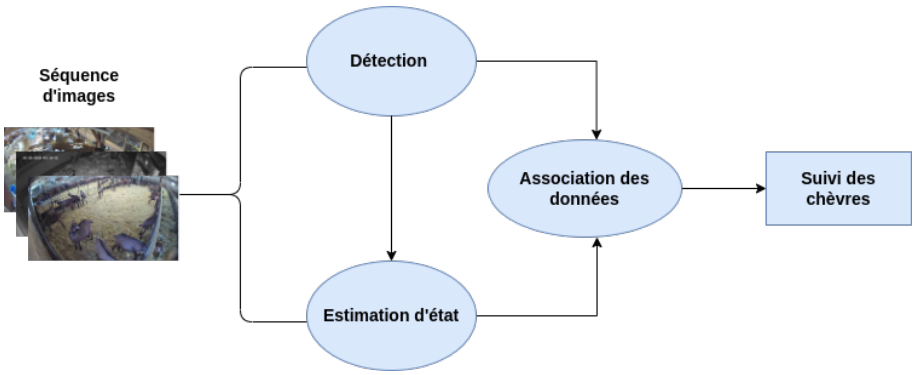


FIGURE 3.1 – Architecture générale du système de détection et suivi des chèvres.

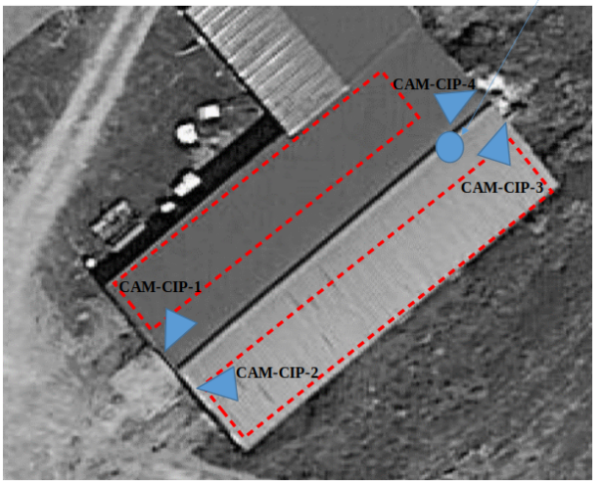


FIGURE 3.2 – Installation des caméras dans la ferme de Nohant.

3.2 Acquisition des données

Pour l'acquisition des données sur les chèvres, des caméras Hikvision DS-2CD2325FWD [1] de haute résolution (1920x1080 pixels) sont utilisées. L'installation des caméras a été réalisée par l'entreprise ATICOM, partenaire du projet. Afin de diversifier les contextes d'acquisitions, trois lieux ont été retenus pour les scènes caprines :

- un enclos d'élevage situé sur l'exploitation expérimentale de l'INRAe à Avord, avec 1 lot de 60 chèvres de race alpine ;
- un enclos d'élevage situé sur une exploitation agricole à Nohant sur Vic, avec 2 lots de 30 à 60 chèvres composés de race alpine et Saanen ;
- un enclos d'élevage situé dans une ferme commerciale à Lusignan avec 1 lot de 20 chèvres.

Dans les deux premières fermes d'élevages (Avord et Nohant), quatre caméras sont positionnées dans chaque coin supérieur des enclos (figure 3.2) offrant quatre différents angles de vue pour observer les postures et le comportement. Comme nous pouvons l'observer sur la figure 3.3, les images (a) et (b) proviennent respectivement de Nohant et Avord. Nous avons utilisé ces données pour construire une première base d'apprentissage et de test.

Par la suite, nous avons récupéré d'autres données provenant de la troisième ferme (Lusignan), offrant une vue de dessus au centre de l'enclos (figure 3.3 (c) et (d)) et ajouté aux données de la première base.

Dans les fermes, les vidéos sont enregistrées 24 heures sur 24 (jour et nuit) en utilisant l'enregistreur DS-7608NI-K2 avec un disque dur de 3To, à 25 images par seconde. Chaque vidéo fait environ 2 heures et sur une journée nous avons autour de 45 Go de données. L'architecture matérielle pour l'acquisition des données est composée de deux parties principales : une partie locale, située sur le site de l'exploitation, permettant l'acquisition et le stockage des images et une partie distante, responsable du traitement des données.

1. https ://www.hikvision.com/content/dam/hikvision/fr/datasheets/series-pro/DS-2CD2325FWD-I_Fiches_Techniques_V5.pdf

La partie locale est composée de :

- d'un réseau de capteurs, permettant l'acquisition des images ;
- d'un système de stockage local des images ;
- d'une interface utilisateur permettant la consultation des images et l'accès aux outils de pilotage : la visualisation des flux vidéos ainsi que les résultats d'analyse en temps réel produit par notre système. L'utilisateur a la possibilité de faire certaines manipulations comme récuprér l'analyse des données sur une période donnée ;
- d'un ordinateur, interconnectant les trois composants précédents et responsable de la partie logique. Il permet également de réaliser une partie des traitements ;
- d'un accès au réseau internet pour les interactions avec la partie décentralisée (partie distante) : connexion à distance sur le serveur de traitement.

La partie distante se compose principalement d'un serveur de stockage et d'un serveur de calcul permettant de réaliser le traitement des images, l'extraction des indicateurs et aussi le stockage des traitements.

La base de données, extraite de vidéos enregistrées, contient des scènes présentant les situations suivantes :

- alimentation ;
- rumination ;
- abreuvement ;
- debout, couchées, repos ;
- entrée et sortie de l'enclos ;
- reproduction ;
- mise-bas ;
- passage hommes, chiens et machines.

Les vidéos utilisées pour les tests sont représentatives des conditions réelles d'acquisition avec :

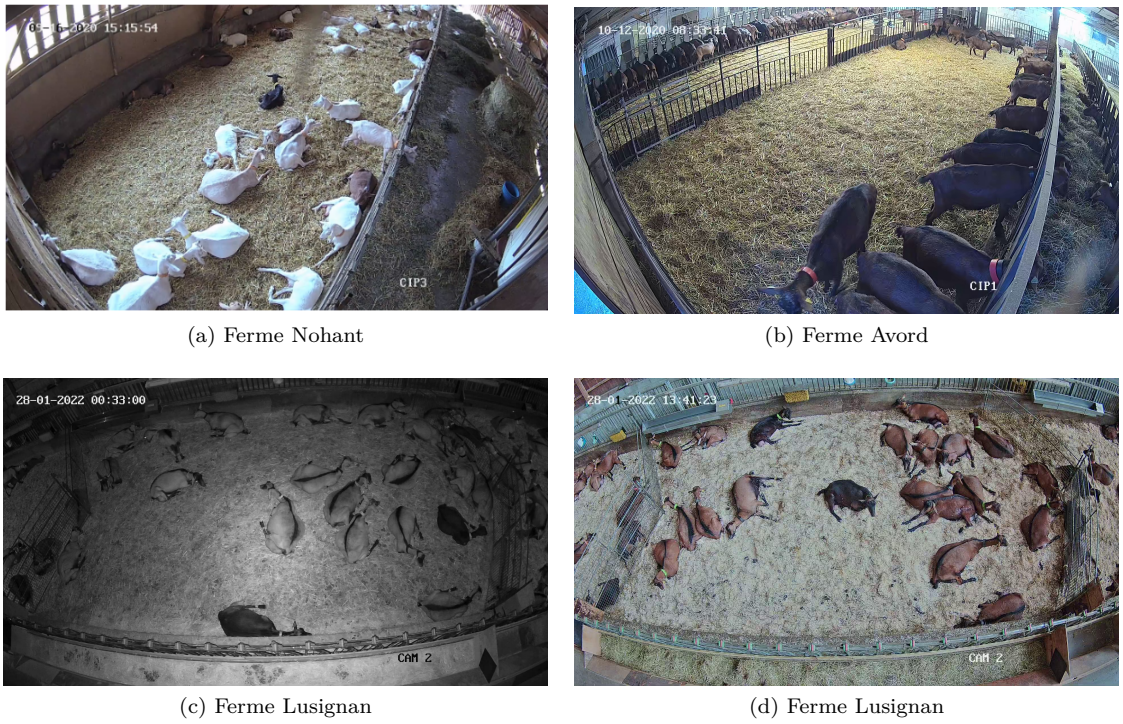
— différents niveaux de luminosité (y compris des scènes nocturnes) ;

— présence de poussière, brume, insects, etc ;

— différentes densités animales dans la scène.

3.3 Vérités terrain

Pour pouvoir entraîner et évaluer notre modèle de détection des chèvres, nous avons construit manuellement une base d'images annotées à partir des vidéos provenant des fermes d'observation (Nohant, Avord et Lusignan). Les coordonnées réelles (vérité terrain) de la position des chèvres dans les images sont représentées sous forme de boîtes englobantes. Nous avons défini deux classes pour qualifier l'état



(a) Ferme Nohant (b) Ferme Avord

(c) Ferme Lusignan (d) Ferme Lusignan

FIGURE 3.3 – Exemples d'images provenant des fermes.

des chèvres dans les enclos : "chevre_debout" et "chevre_couche". Nous avons construit deux jeux de données en mélangeant plusieurs images extraites de différentes vidéos (jour comme nuit) :

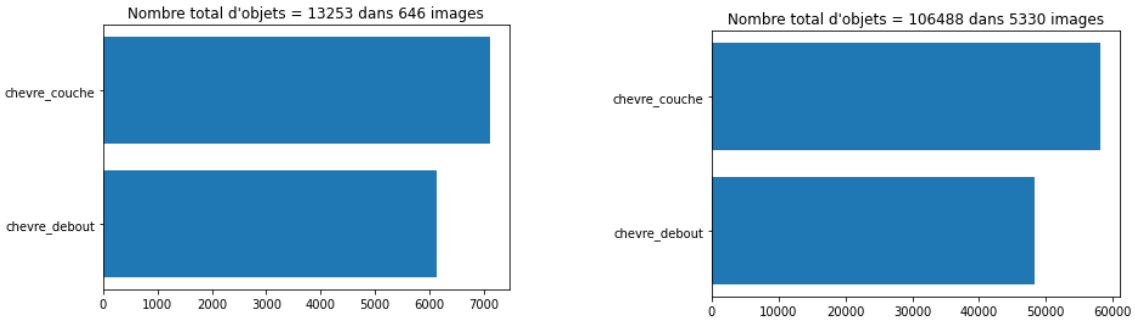
- jeu de données 1 : 796 images qui proviennent des deux premières fermes Nohant et Avord manuellement annotées en utilisant l'outil labelImg [2] (figure 3.5) et divisées en 646 images pour l'entraînement et 150 images pour le test ;
- jeu de données 2 : 5922 images qui proviennent de l'augmentation du jeu de données 1 en ajoutant de nouvelles images des deux fermes précédentes et de la ferme de Lusignan, séparées en 5330 images pour l'entraînement et 592 pour le test. Ici nous avons utilisé l'outil CVAT [3] pour faire l'annotation.

Ces Images sont extraites de plusieurs vidéos différentes avec des scènes variées (on extrait une image à chaque 10 minutes). La figure 3.4 nous montre la répartition par classe du nombre de chèvres présentes dans chacun de nos jeux de données d'entraînement. Le deuxième jeu de données est construit afin d'améliorer les performances de détection par rapport au premier jeu de données car les modèles entraînés sur ce dernier présentaient quelques échecs de détection, surtout sur les images de la troisième ferme.

3.4 Détection des chèvres

Comme nous l'avons vu dans le chapitre 2 de ce document, il existe plusieurs approches dans la littérature permettant de faire de la détection d'objets. Pour effectuer la détection des chèvres, nous avons

2. https ://www.v7labs.com/blog/labelimg-guide
3. https ://www.cvat.ai/



(a) Jeu de données 1

(b) Jeu de données 2

FIGURE 3.4 – Répartition du nombre de chèvres par classe pour les 2 jeux de données

FIGURE 3.5 – Annotation des données avec l'outil LabelImg.

expérimenté deux méthodes à base des réseaux de neurones convolutifs, l'une en deux étapes (Faster R-CNN) et l'autre en une étape (YOLOv4) afin de comparer les deux types de détection à base de CNN (en deux étapes et en une étape). En effet, l'apprentissage profond utilise des couches de convolutions (CNN) qui permettent d'extraire automatiquement des caractéristiques de haut niveau qui représentent mieux les formes complexes des objets par rapport aux caractéristiques classiques de bas niveau des méthodes sans apprentissage profond. Les approches à base de CNN sont également plus performantes et robustes aux occultations. Il existe des modèles de CNN pré-entraînés sur de grandes bases d'images génériques dont l'apprentissage a été affiné sur nos jeux de données (transfert learning ou apprentissage par transfert) pour accélérer l'entraînement et permettre au modèle de converger plus rapidement. Cela nous permet de gagner en temps sur l'entraînement des nouveaux modèles.

Nous avons effectué les expérimentations de la détection des chèvres sur un serveur Ubuntu avec une machine intel(R) Core(TM) i9-9900K CPU 3.60GHz, RAM 64GB avec une carte graphique Nvidia Titan RTX GPU de 24GB de mémoire vidéo.

3.4.1 Faster R-CNN

Dans cette section, nous allons expérimenter un exemple de méthode de détection en deux étapes. Dans cette méthode, la détection s'effectue en deux étapes : la première étape consiste à générer des

régions d'intérêt de l'image susceptibles de contenir un objet potentiel (ROI). Ensuite, chaque région est classée afin de déterminer si elle contient vraiment un objet et prédire sa classe le cas échéant. D'où le nom de détection en deux étapes. La méthode testée dans cette partie est le faster R-CNN [17].

Architecture

Il s'agit de l'une des méthodes les plus performantes de la catégorie de détection en deux étapes. Cette méthode est une amélioration du Fast R-CNN [17] qui permet d'effectuer la détection en temps réel. L'architecture du Faster R-CNN comprend deux branches après les couches de convolutions :

- la première branche est un réseau de proposition de régions (RPN) [17] sur la base de la dernière carte de caractéristiques. Il s'agit d'un réseau convolutionnel complet qui glisse sur la carte des caractéristiques et indique pour chaque position s'il y a un objet ou non (avec probabilité et décalage potentiel des contours), sans tenir compte de la classe de l'objet ;
- la deuxième branche classe les propositions du RPN.

Afin d'avoir un système qui soit robuste à la translation et au changement d'échelle, le RPN utilise un algorithme basé sur les boîtes d'ancrages ou ancres. Pour chaque position de la fenêtre coulissante sur la carte des caractéristiques, 9 boîtes d'ancrage sont placées. La sélection des boîtes d'ancrage dépend généralement de l'ensemble de données de la vérité terrain. Le choix du nombre spécifique de boîtes d'ancrage est souvent un choix de conception effectué par les développeurs du modèle sur la base de leur ensemble de données et des exigences de la tâche. Il est possible d'expérimenter diverses configurations du nombre d'ancres pour trouver celle qui convient le mieux pour un cas d'utilisation spécifique et un ensemble de données. Cependant, les 9 ancres utilisées par défaut dans Faster R-CNN arrivent à donner de bons résultats sur la majorité des données car elles sont générées sur un grand ensemble de données, comme ImageNet [4], qui contient presque toutes les catégories d'objets. Les ancres sont toutes centrées sur la fenêtre coulissante, seules leur échelle et leur rapport changent (figures 3.6 et 3.7). La figure 3.7 nous montre l'utilisation de k ancres sur la carte de caractéristiques produit par le RPN. Cette carte passe dans des couches intermédiaires qui vont prédire la classe et les décalages x, y, w, h des k boîtes d'ancrages, par rapport à la boîte de la vérité terrain. Il y a trois échelles (128^2, 256^2 et 512^2 pixels) et trois rapports (1 :1 , 2 :1 et 1 :2), ce qui correspond aux 9 ancres. La taille et la forme de ces boîtes d'ancrage sont générées de manière à ce qu'elles soient prochent des objets de la vérité terrain. Chaque ancre est traitée à travers les couches convolutionnelles du RPN qui produisent la probabilité que cette ancre représente un objet et potentiellement un décalage pour corriger les dimensions de l'ancre. Ainsi, si nous avons une carte de caractéristiques de dimensions hauteur = 32 et largeur = 64, avec 9 ancres ont aura au total 18432 ancres à analyser. Un filtrage est effectué en fonction du score de probabilité afin de ne conserver qu'une partie de ces ancres, qualifiées de positives (celles comportant des objets plus probables). Le reste des ancres va correspondre aux exemples négatifs ou arrière-plan.

Ainsi une étiquette positive sera donnée si :

4. https ://image-net.org/about.php

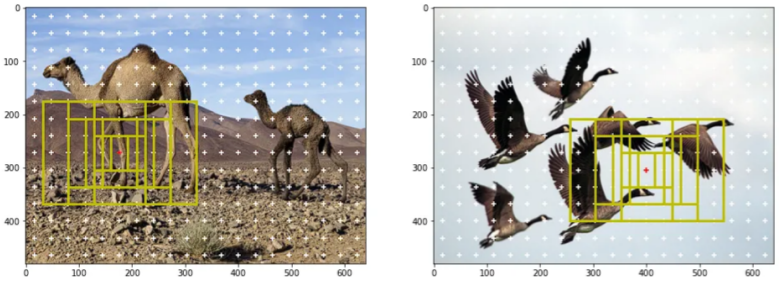


FIGURE 3.6 – Exemple de génération des ancres dans Faster R-CNN [26].

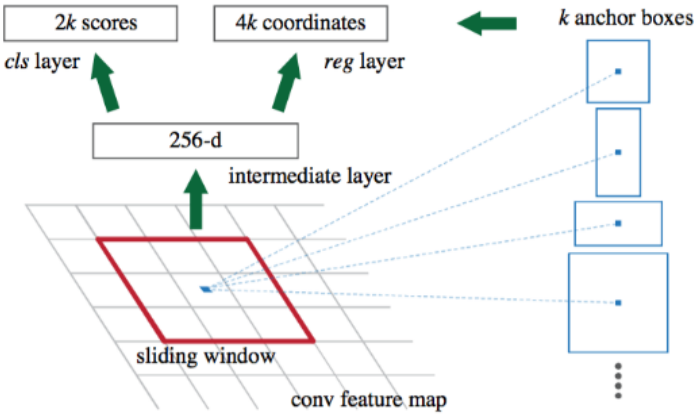


FIGURE 3.7 – Utilisation des ancres dans Faster R-CNN [17].

– l'ancre ou les ancres dont l'intersection sur l'union (IoU) est la plus élevée avec une boîte de la vérité terrain ;

– l'ancre dont l'intersection sur l'union (IoU) est supérieure à un seuil donné (0.7 dans Faster R-CNN) avec n'importe quelle boîte de la vérité terrain.

Les ancres qui ont un score d'IoU inférieur à 0.3 pour toutes les boîtes de la vérité terrain, obtiennent une étiquette négative. Les ancres qui ne sont ni positives ni négatives ne contribuent pas à la minimisation de la fonction objectif dont l'équation est la suivante :

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{3.1}$$

$$L_{cls}(p_i, p_i^*) = -(p_i \cdot log(p_i^*) + (1-p) \cdot log(1-p_i^*)) \tag{3.2}$$

$$L_{reg}(t_i, t_i^*) = smooth_{L1}(t_i - t_i^*) \tag{3.3}$$

avec i l'index de l'ancre dans le lot, p_i la probabilité que l'ancre i contienne un objet, $p_i^* = 1$ si l'ancre i à une étiquette positive et 0 si son étiquette est négative, t_i les coordonnées de la boîte prédite, t_i^* les coordonnées de la boîte de la vérité terrain à laquelle elle est associée, L_{cls} représente la fonction de coût

pour la classification (l'entropie croisée binaire) et L_{reg} représente la fonction de coût pour la régression des boîtes englobantes qui est activée que lorsque $p_i^* = 1$.

Pour la classification, la fonction de coût logarithmique entre 2 classes (objet et non objet) est utilisée. Pour la régression, la fonction de coût "Smooth L1 loss" est utilisée. Dans la régression des coordonnées des boîtes englobantes, les coordonnées sont calculées comme suit :

$$t_x = (x - x_a)/w_a, t_y = (y - y_a)/h_a, \tag{3.4}$$

$$t_w = log(w/w_a), t_h = log(h/h_a), \tag{3.5}$$

$$t_x^* = (x^* - x_a)/w_a, t_y^* = (y^* - y_a)/h_a, \tag{3.6}$$

$$t_w^* = log(w^*/w_a), t_h^* = log(h^*/h_a) \tag{3.7}$$

où x, y correspondent aux coordonnées du centre de la boîte englobante prédite et h, w respectivement sa hauteur et sa largeur ; x_a, y_a, w_a, h_a correspondent aux coordonnées et dimensions de l'ancre ; x^*, y^*, w^*, h^* correspondent aux coordonnées et dimensions de la boîte de la vérité terrain. L'idée de la régression ici est de corriger les coordonnées et dimensions des ancres de façon à ce qu'elles se rapprochent des coordonnées et dimensions de la boîte de la vérité terrain.

Entraînement

Le Faster R-CNN comprend donc deux réseaux : un réseau RPN pour la proposition de régions et un réseau pour la détection d'objets (Fast R-CNN). Pour pouvoir entraîner ces deux réseaux de façon à ce qu'ils partagent les mêmes caractéristiques CNN, il existe trois techniques d'entraînement :

- apprentissage alterné : dans cette technique, le RPN est entraîné dans un premier temps séparément et ensuite les propositions sont utilisées pour entraîner le Fast R-CNN pour la détection. Le réseau mis au point par Fast R-CNN est ensuite utilisé pour initialiser le RPN, et ce processus est itéré ;
- apprentissage conjoint approximatif : dans cette technique, les réseaux RPN et Fast R-CNN sont fusionnés en un seul réseau pendant l'entraînement. Dans chaque itération du gradient, la rétropropagation génère des propositions de régions qui sont traitées comme des propositions fixes et précalculées lors de l'apprentissage d'un Fast R-CNN. Mais cette technique ignore les gradients par rapport aux coordonnées des boîtes de proposition qui sont également des réponses du réseau, et est donc approximative ;
- apprentissage conjoint non approximatif : similaire à la technique précédente. Ici, une couche de mise en commun (RoI pooling) est utilisée pour permettre de calculer les gradients non seulement par rapport aux paramètres du réseau, mais aussi par rapport aux coordonnées de la boîte englobante (régression), puisqu'il s'agit de paramètres pouvant être appris.

Pour la détection des chèvres avec le Faster R-CNN, nous avons utilisé la troisième technique d'apprentissage pour aller plus rapidement, comparé à l'apprentissage alterné. Nous avons d'abord effectué l'entraînement sur nos deux jeux de données en utilisant les hyperparamètres optimaux de Faster R-CNN (tableau 3.1). Nous avons utilisé le framework Pytorch [5] pour l'implémentation du réseau et l'entraînement. Nous avons utilisé la technique d'apprentissage par transfert en initialisant les poids de notre modèle par les poids du modèle Faster R-CNN pré-entraîné sur le jeu de données Pascal VOC avec Resnet-50 comme extracteur de caractéristiques.

TABLE 3.1 – Hyperparamètres pour l'entraînement du Faster R-CNN

Paramètre	Valeur
Batch size	16
Taux d'apprentissage	0.001
Optimiseur	SGD
Taille des entrées	640x640 pixels
Momentum	0.9
Nombre de classes	2
Nombre d'époques	300

Résultats

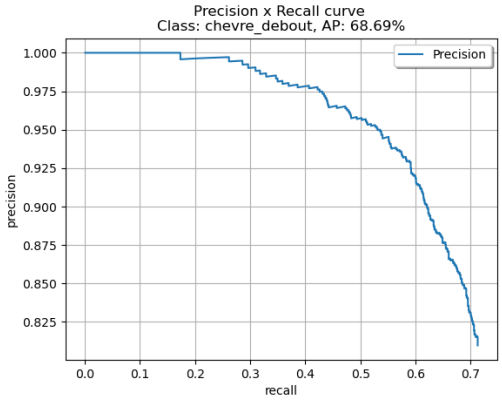
L'évaluation des résultats de la précision moyenne (mAP) est effectuée en suivant le protocole du défi PASCAL VOC [68] en considérant : la précision moyenne (AP), la moyenne de la précision moyenne (mAP), la courbe AUC (précision-rappel) et le nombre d'images par seconde (FPS). Nous avons entraîné et testé le Faster R-CNN sur nos deux jeux de données. Pour le jeu de données 1 (150 images de test), nous avons obtenu une précision moyenne sur les 2 classes de 72.41% et 20.7 FPS, et pour le jeu de données 2 nous avons une précision moyenne de 97.9%. Le tableau 3.2 montre les résultats de l'évaluation du Faster R-CNN sur le jeu de données 1 et le tableau 3.3 montre ceux du jeu de données 2.

TABLE 3.2 – Évaluation de la détection avec Faster R-CNN sur le jeu de données 1

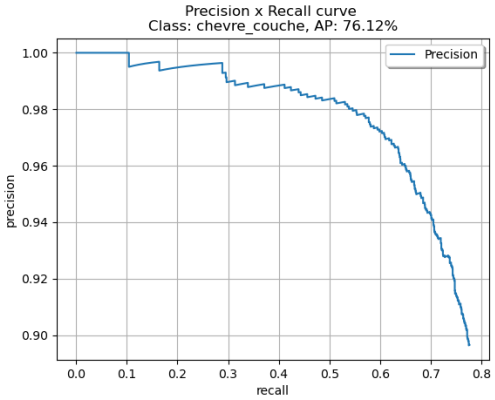
Jeu de données 1 : test sur 150 images - apprentissage 646 images	
Métrique	Valeur
mAP@.5	72.41%
AP@.5 chevre_couche	76.12%
AP@.5 chevre_debout	68.69%
FPS	20.7

Nous pouvons également visualiser les courbes AUC de précision-rappel obtenues pour les jeux de données 1 et 2 sur la figure 3.8 : (a) et (b) ; (c) et (d) respectivement. Ces courbes, confirment que la performance est supérieure sur le jeu de données 2. Les tableaux 3.2 et 3.3 nous montrent les performances du Faster R-CNN sur nos deux jeux de données. On peut remarquer que les résultats sur le premier jeu de données (tableau 3.2) était peu performants, ce qui nous a amené à construire le deuxième jeu de

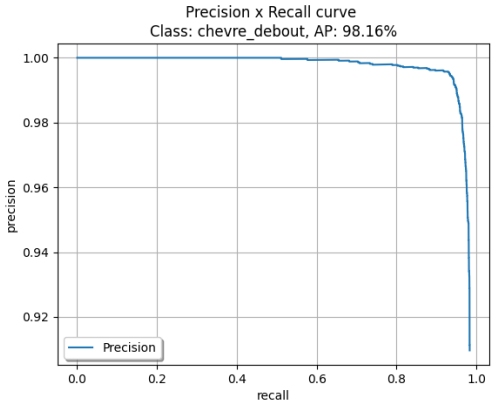
5. https ://pytorch.org/



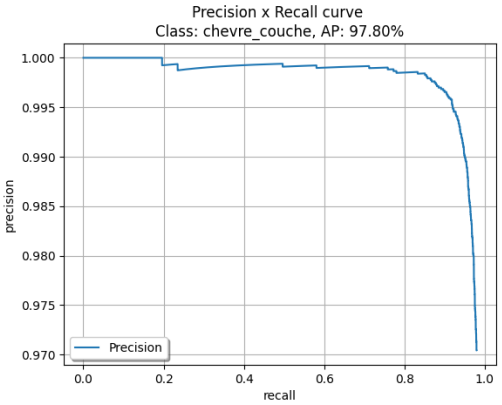
(a) Classe chevre_debout : jeu de données 1



(b) Classe chevre_couche : jeu de données 1



(c) Classe chevre_debout : jeu de données 2



(d) Classe chevre_couche : jeu de données 2

FIGURE 3.8 – Évolution de la précision moyenne en fonction du rappel, par classe, du Faster R-CNN sur les jeux de données 1 et 2

données (tableau 3.3) où les résultats sont meilleurs. Cette amélioration peut s'expliquer par le fait que, le jeu de données 2 contient plus d'exemples et plus de variations que le jeu de données 1.

TABLE 3.3 – Évaluation de la détection avec Faster R-CNN sur le jeu de données 2

Jeu de données 2 : test sur 592 images - apprentissage 5330 images	
Métrique	Valeur
mAP@.5	97.9%
AP@.5 chevre_couche	97.80%
AP@.5 chevre_debout	98.16%
FPS	20.7

3.4.2 YOLOv4

Dans cette section, nous allons tester un exemple de méthode de détection en une étape. Dans cette méthode, la détection s'effectue directement sur la carte de caractéristiques produite par les couches de convolutions, contrairement à la détection en deux étapes. La méthode testée dans cette partie est YOLOv4 [27], une des méthodes performantes de détection en une étape. Il s'agit de la quatrième version

de la famille des détecteurs YOLO, et était la plus récente lors de nos expérimentations. Cette version apporte plusieurs améliorations, par rapport aux versions précédentes (YOLOv1 [87], YOLOv2 [89] et YOLOv3 [31]), qui augmentent ses performances en terme de précision et de temps de calcul.

Les améliorations apportées dans YOLOv4 concernent principalement l'architecture, les fonctions de coût et l'augmentation des données :

1. l'augmentation de la variabilité des images d'entrée, afin que le modèle de détection d'objets soit plus robuste. Il existe plusieurs techniques pour augmenter les données. Dans YOLOv4, les chercheurs ont utilisé les techniques d'augmentation suivantes : le **MixUp** [132] qui utilise deux images pour les multiplier et les superposer avec différents coefficients, puis ajuste leur label en fonction de ces coefficients superposés, le **CutMix** [133] où on coupe et colle des patchs aléatoires entre les images d'apprentissage. Les labels de la vérité terrain sont mélangés proportionnellement à la surface des patchs dans les images et le **CutOut** [134] qui consiste à masquer aléatoirement des régions carrées de l'entrée pendant l'apprentissage. YOLOv4 a introduit également deux autres méthodes d'augmentation de données : la méthode Mosaïque et la SAT [27] ;

2. les méthodes de régularisation : appliquées aux cartes de caractéristiques pour éviter le surapprentissage des réseaux CNN. Dans YOLOv4, les techniques de régularisation suivantes peuvent être utilisées : le **DropOut** qui consiste à désactiver un ensemble de neurones avec une probabilité aléatoire à un moment donné de l'apprentissage afin d'éviter un surapprentissage du réseau, le **DropBlock** [135] qui élimine des régions contiguës d'une carte de caractéristiques d'une couche au lieu d'éliminer des unités aléatoires indépendantes comme dans le DropOut, le **DropPath** [136] qui élimine aléatoirement des exemples pendant l'apprentissage avec une probabilité donnée ;

3. la fonction de coût pour la régression des boîtes englobantes prenant en compte la zone de couverture de l'objet. En effet, les détecteurs d'objets précédents utilisent l'erreur quadratique (MSE) pour effectuer la régression des coordonnées des boîtes englobantes qui se focalise plus sur les points (valeurs des coordonnées) que sur la zone de couverture de la boîte englobante de l'objet qui fait que l'erreur augmente avec le changement d'échelle de l'objet. Pour corriger cela, les chercheurs ont introduit la fonction de coût **"IoU Loss"** qui prend en compte la zone de couverture de l'objet et reste invariante au changement d'échelle de l'objet. Il existe plusieurs versions du "IoU Loss" :

   — **GIoU Loss** [137] prend en compte les cas qui ne se chevauchent pas (contrairement au **IoU Loss**).

   $$GIoULoss = 1 - IoU + \frac{|C - (B \cup B^{gt})|}{|C|}, \tag{3.8}$$

   avec B la boîte prédite, B^{gt} la boîte de la vérité terrain et C la plus petite surface convexe entre B et B^{gt} ;

   — **DIoU Loss** [137] considère la distance du centre d'un objet comme un terme de pénalité supplémentaire avec l'équation :

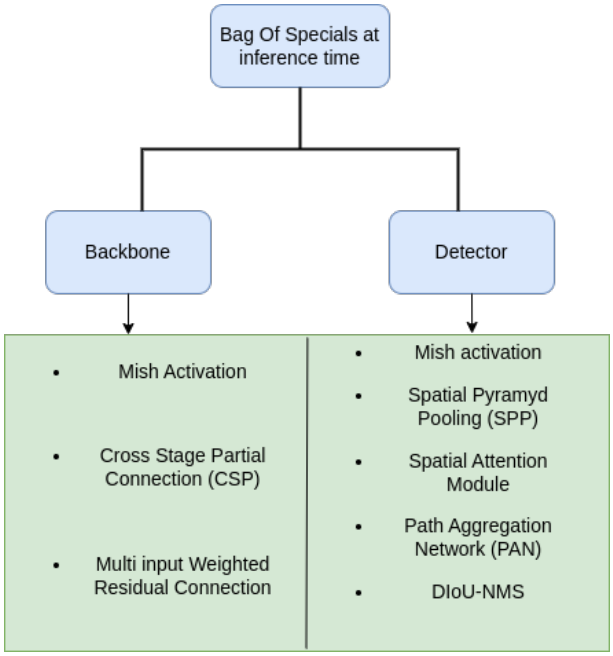


FIGURE 3.9 – Stratégies de **"Bag Of Specials"** utilisées dans YOLOv4 [27].

$$DIoULoss = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2}, \tag{3.9}$$

avec b le centre de la boîte prédite, b^{gt} le centre la boîte de la vérité terrain et c est la longueur de la diagonale de la plus petite boîte englobante couvrant les deux boîtes (B, B^{gt}). ρ désigne la distance entre les deux centres b et b^{gt} ;

— **CIoU Loss** [137] prend simultanément en compte trois paramètres : la zone de chevauchement, la distance entre les points centraux et le rapport d'aspect. Il s'agit d'une extension du **DIoU Loss** :

$$CIoULoss = DIoULoss + \alpha v, \tag{3.10}$$

avec α un paramètre de pondération positif dans lequel les chevauchements sont plus prioritaires que les cas de non-chevauchement et v donne des informations sur la cohérence du rapport d'aspect.

$$v = \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h})^2 \tag{3.11}$$

w, h sont les dimensions de la boîte prédite, w^{gt}, h^{gt} sont celles de la boîte de la vérité terrain.

$$\alpha = \frac{v}{(1 - IoU) + v} \tag{3.12}$$

Nous avons également d'autres stratégies de post-traitement dénommées **"Bag Of Specials"** qui contiennent différents plugins et modules qui n'augmentent que légèrement le coût de l'inférence, mais peuvent amélio-

rer considérablement la précision du détecteur d'objets. Les différentes techniques utilisées sont résumées dans la figure 3.9. Nous pouvons voir pour chaque partie du réseau, quelle est la technique utilisée lors de l'inférence pour améliorer les performances du réseau. Ces techniques sont principalement :

— l'activation Mish [138] : la fonction d'activation utilisée dans YOLOv4. Dans un réseau de neurones convolutif (CNN), une fonction d'activation est une opération non linéaire appliquée à la sortie de chaque couche convolutive. Les fonctions d'activation introduisent la non-linéarité dans le réseau, ce qui lui permet d'apprendre à partir de modèles et de relations complexes dans les données. ReLu fut la fonction d'activation la plus utilisée dans la littérature pour la détection d'objets. Mais la dérivée de cette fonction pour les valeurs négatives est de 0, ce qui entrave la méthode de descente du gradient car 40% des neurones du réseau ne seront pas mis à jour. Pour corriger cela, l'activation Leaky Relu a été introduite mais n'atteint pas les performances souhaitées. L'activation Mish corrige ces problèmes en préservant les valeurs négatives, qui stabilisent le flux de gradient du réseau et aide à apprendre des caractéristiques plus représentatives. La fonction d'activation Mish est donnée par :

$$f(x) = x \cdot \tanh\left(\ln\left(1 + e^{x}\right)\right) \tag{3.13}$$

— le Module d'Attention Spatiale (SAM) : aide à résoudre les questions "où ?" et "quoi ?" en demandant au réseau d'accorder plus d'importance aux informations contextuelles autour d'un objet et aux caractéristiques qui sont importantes. Dans le module SAM original [139], le regroupement maximum (MaxPooling) et le regroupement moyen (AvgPooling) sont appliqués séparément aux cartes de caractéristiques d'entrée pour créer deux ensembles de cartes de caractéristiques. Les résultats sont introduits dans une couche de convolution suivie d'une fonction sigmoïde pour créer l'attention spatiale. Ce masque d'attention spatiale est appliqué à la caractéristique d'entrée pour produire les cartes de caractéristiques affinées. Dans YOLOv4, une version modifiée du SAM est utilisée en remplaçant le regroupement maximum et moyen par une convolution 1 x 1 (figure 3.10) ;

— DIoU Supression Non Maximale (NMS) : NMS filtre les autres boîtes englobantes qui prédisent le même objet et retient celle qui présente le plus haut degré de confiance. Le DIoU (évoqué précédemment) est utilisé comme facteur dans la suppression non maximale (NMS). Cette méthode prend en compte le DIoU et la distance entre les points centraux de deux boîtes englobante lors de la suppression des boîtes redondantes. Elle est ainsi plus robuste dans les cas d'occultations.

Architecture

L'architecture de YOLOv4 est constituée de 3 parties principales comme on peut le voir sur la figure 3.11. Nous avons :

– l'extracteur de caractéristiques ou "backbone" : il s'agit d'un ensemble de couche de convolutions et de connexions résiduelles permettant d'extraire des caractéristiques pour la détection. YOLOv4 utilise le CSPDarknet53 comme "backbone". L'architecture CSP est dérivée de l'architecture Den-

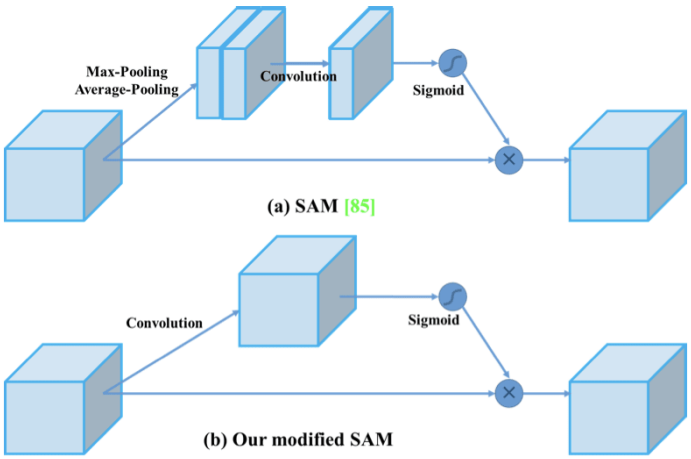


FIGURE 3.10 – Module SAM dans [27].

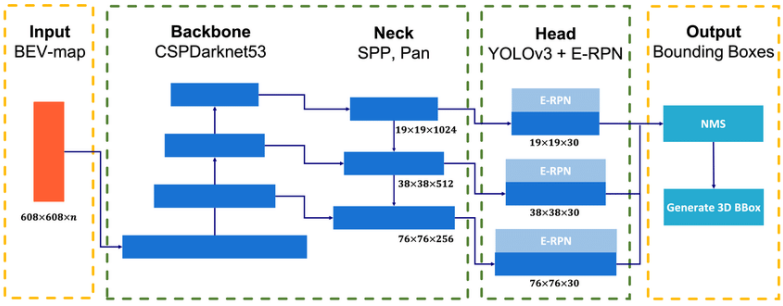


FIGURE 3.11 – Architecture de YOLOv4 [27].

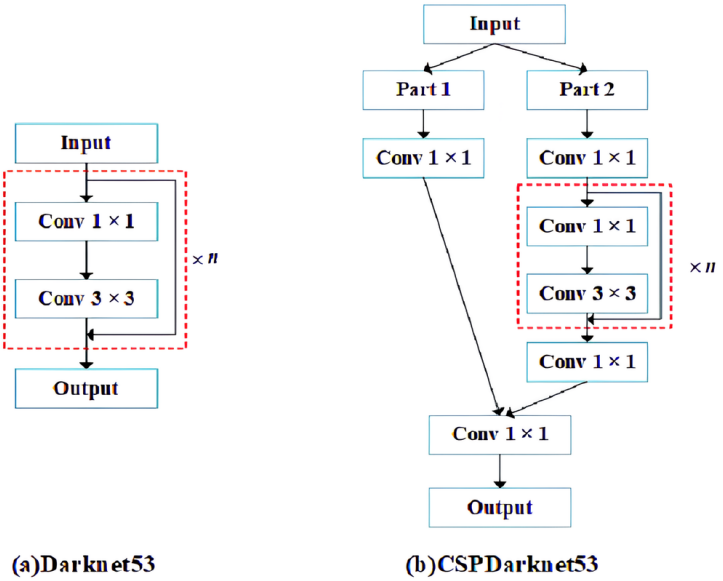


FIGURE 3.12 – Darknet53 vs CSPDarknet53 [28].

seNet ; elle divise l'entrée i en deux parties $Part_1$ et $Part_2$ (figure 3.12). La première partie $Part_1$ passera par la couche dense (Conv 1x1), la seconde $Part_2$ sera concaténée à la fin avec le résultat à la sortie de la couche dense . Cela permet d'augmenter le flux de gradient à travers le réseau, de faciliter le flux d'informations et d'améliorer l'apprentissage des caractéristiques dans les réseaux neuronaux profonds ;

– le réseau intermédiaire ou "neck" : son rôle essentiel est de collecter les cartes de caractéristiques aux différentes échelles. En général, le "neck" est composé de plusieurs chemins ascendants des couches inférieures vers les couches supérieures et de plusieurs chemins descendants des couches supérieures vers les couches inférieures (figure 3.13). Pour la détection, un réseau nécessite une taille fixe d'images obligeant à redimensionner les images et entraîne la perte d'informations. La couche de mise en commun SPP [16] permet de générer des caractéristiques de taille fixe, quelle que soit la taille de nos cartes de caractéristiques sans perdre des parties de l'objet. Pour permettre une meilleure propagation des informations sur les couches de bas en haut ou de haut en bas, le réseau PAN [140] est utilisé avec une légère modification (utilisation de la concaténation à la place de l'addition) ;

– Le réseau de prédiction ou le "head" combine des caractéristiques de différentes échelles pour créer une représentation efficace des caractéristiques pour la détection d'objets. Il prédit ensuite un vecteur dense contenant les coordonnées des boîtes englobantes, le score de confiance et les probabilités de classe. Pour la classification, YOLOv4 utilise le "Focal Loss" [94] qui permet de mieux gérer le déséquilibre entre les classes.

Afin de s'adapter à un large éventail de contraintes en matière de ressources, les auteurs ont proposé une version dénommée Scaled-YOLOv4 [90], dans laquelle on trouve plusieurs versions de YOLOv4 pour différents type de matériels.

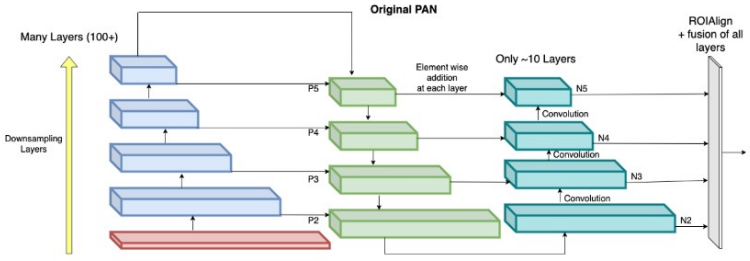


FIGURE 3.13 – Architecture du réseau PAN [29].

YOLOv4 utilise également les boîte d'ancrages prédéfinies lors de l'apprentissage pour prédire les coordonnées des boîtes englobantes. Ces boîtes peuvent être générées en utilisant l'algorithme de clustering K-Means sur la vérité terrain. L'idée est de regrouper les vecteurs (largeur, hauteur) en clusters de manière à ce que l'IoU moyenne soit la plus élevée possible. Les clusters ici représentent les vecteurs dont les dimensions sont assez proches. Par défaut, YOLOv4 dispose de 9 boîtes d'ancrages prédéfinies obtenues sur la base d'images ImageNet [6]. Ces boîtes d'ancrages sont réparties sur 3 échelles de détection comme suit :

— échelle 1 : (12, 16), (19, 36) et (40, 28)

— échelle 2 : (36, 75), (76, 55) et (72, 146)

— échelle 3 : (142, 110), (192, 243) et (459, 401)

La carte de caractéristiques, issue de l'image en entrée, est divisée en plusieurs cellules pour chaque échelle de détection. Pour la première échelle de détection, nous avons un pas de 32, pour la deuxième échelle nous avons un pas de 16 et 8 pour la dernière. Pour une image d'entrée de taille 416x416 pixels, on aura donc une carte de caractéristiques de taille (416/32)x(416/32)x255 qui correspond à 13x13 = 169 cellules au total pour l'échelle 1 et 255 filtres. Nous aurons un total de 10647 boîtes (13x13x3 + 26x26x3 + 52x52x3) prédites par le réseau. Pour l'entraînement de YOLOv4 sur l'ensemble des données MS COCO avec C=80 (nombre de classes), B=3 (nombre de boîtes d'ancrages par cellule) et une image de dimensions 416x416 pixels, la première échelle de prédiction, après un seul passage du CNN, produit un tenseur de dimension = [(13, 13, 3 * (5 + 80)] comme on peut l'observer sur la figure 3.14. Sur cette figure, nous avons l'image d'entrée qui est divisée en 169 cellules (13x13) à la première échelle de détection et aussi les 3 boîtes prédites par une cellule sur cette échelle.

Sur la figure 3.15, nous pouvons observer les équations de calcul des coordonnées (b_x, b_y, b_w, b_h) de la boîte calculée pour une échelle donnée : t_x, t_y correspondent au centre de la boîte prédite par le réseau, t_w, t_h sont sa largeur et sa hauteur ; p_w, p_h représentent les dimensions de la boîte d'ancrage (largeur et hauteur) ; c_x, c_y correspondent aux coordonnées de la cellule sur la carte de caractéristiques, par rapport au coin supérieur gauche.

Une fois les prédictions obtenues, on utilise le NMS (suppression non maximale) pour ne garder que les détections pertinentes.

6. https ://www.image-net.org/update-mar-11-2021.php

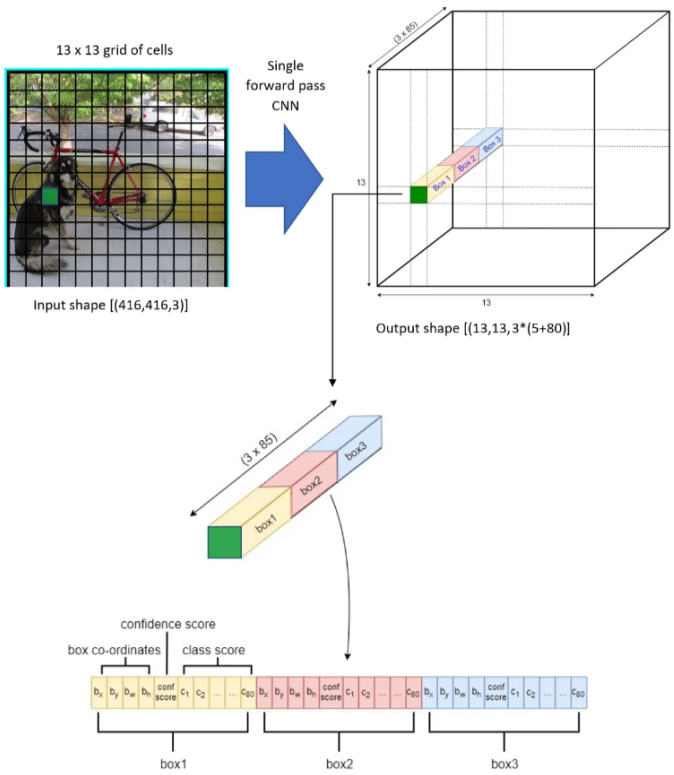


FIGURE 3.14 – Exemple de prédiction de YOLOv4 sur MS COCO [30].

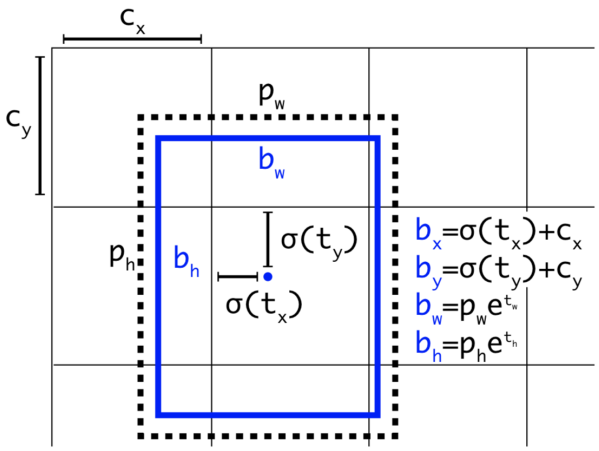


FIGURE 3.15 – Utilisation de la boîte d'ancrage pour la régression des boîtes dans YOLOv4 [31].

Entraînement

L'entraînement de YOLOv4 a été effectué en utilisant le framework darknet avec un apprentissage par transfert à partir des poids pré-entraînés (yolov4.conv.137) sur le jeu de données MS COCO. Le tableau 3.4 nous montre les hyperparamètres utilisés pour l'entraînement. Le "backbone" utilisé est le CSPDarknet53 et la fonction de coût pour la régression est le CIoU Loss.

TABLE 3.4 – Hyperparamètres pour l'entraînement de YOLOv4

Paramètre	Valeur
Batch size	64
Subdivisions	16
Taux d'apprentissage	0.001
Optimiseur	SGD
Taille des entrées	640x640 pixels
Momentum	0.949
Decay	0.0005
Nombre de classes	2
Nombre d'itération	4000

Résultats

L'évaluation des résultats est effectuée en suivant le même protocole que précédemment sur nos deux jeux de données. Pour le jeu de données 1, nous avons obtenu une précision moyenne sur les 2 classes de 88.64% et 54 FPS comme on peut le voir dans le tableau 3.5. Pour le jeu de données 2, nous avons une précision moyenne de 98.54% sur les deux classes (3.6).

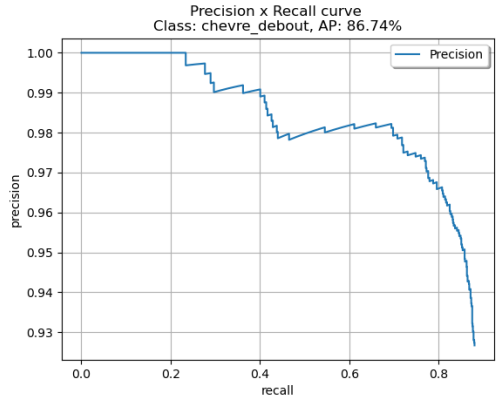
TABLE 3.5 – Évaluation de la détection avec YOLOv4 sur le jeu de données 1

Jeu de données 1 : test sur 150 images - apprentissage 646 images	
Métrique	Valeur
mAP@.5	88.64%
AP@.5 chevre_couche	90.56%
AP@.5 chevre_debout	86.74%
FPS	54

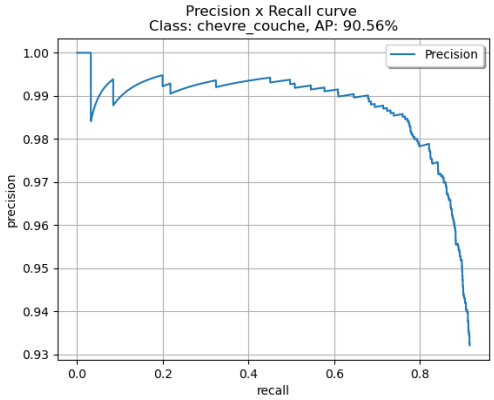
TABLE 3.6 – Évaluation de la détection avec YOLOv4 sur le jeu de données 2

Jeu de données 2 : test sur 592 images - apprentissage 5330	
Métrique	Valeur
mAP@.5	98.54%
AP@.5 chevre_couche	98.97%
AP@.5 chevre_debout	98.11%
FPS	54

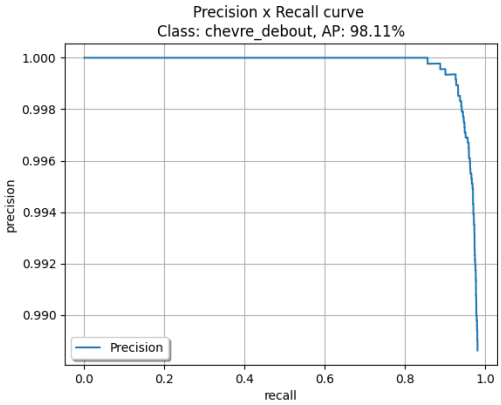
Dans le tableau 3.6, les résultats de la détection sur le jeu de données 2 sont meilleurs que ceux du jeu de données 1. Nous présentons également les courbes AUC de précision-rappel sur nos deux jeux de données (figure 3.16). L'aire en dessous des courbes sur le jeu de données 2 est plus importante que celle du jeu de données 1. Les performances en terme de précision du modèle ont beaucoup augmenté à la suite



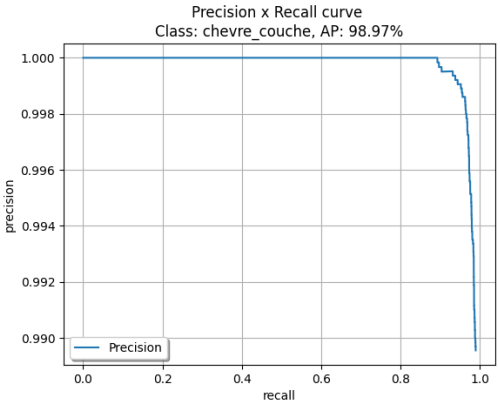
(a) Classe chevre_debout : jeu de données 1



(b) Classe chevre_couche : jeu de données 1



(c) Classe chevre_debout : jeu de données 2



(d) Classe chevre_couche : jeu de données 2

FIGURE 3.16 – Évaluation de la précision moyenne par classe de YOLOv4 sur les jeux de données 1 et 2

de l'augmentation du jeu de données. La figure 3.17 nous montre un exemple de détection des chèvres sur une image de Nohant avec YOLOv4.

D'après ces résultats, nous observons que les performances de YOLOv4 sont supérieures à celles du Faster R-CNN non seulement en terme de précision, mais aussi en terme de vitesse. Cela est dû aux différentes améliorations et nouvelles techniques intégrées dans YOLOv4. Le tableau 3.7 compare les performances de YOLOv4 et Faster R-CNN sur le jeu de données 2. Pour la partie de suivi des chèvres, nous avons besoin d'un modèle rapide et assez précis. Ainsi, nous avons continué avec YOLOv4 comme méthode de détection dans la partie suivi.

TABLE 3.7 – Comparaison de YOLOv4 et Faster R-CNN

Métrique	*YOLOv4*	*Faster R-CNN*
mAP@.5	**98.54%**	97.9%
chevre_debout AP@.5	**98.11%**	98.16%
chevre_couche AP@.5	**98.97%**	97.80%
FPS	**54**	20.7

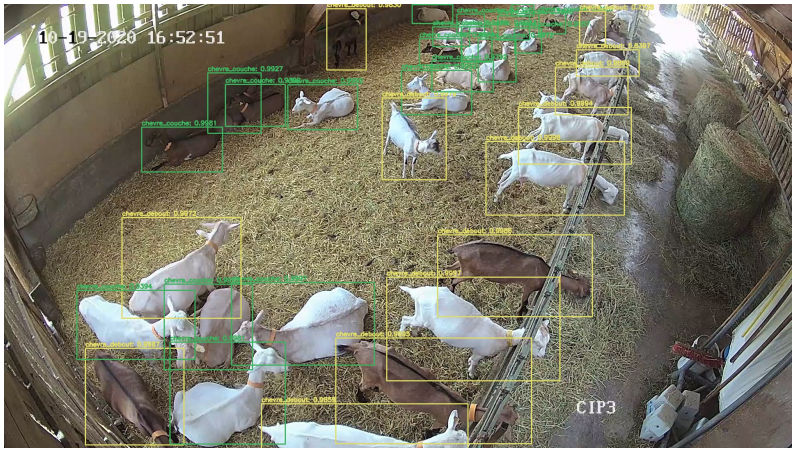


FIGURE 3.17 – Détection des chèvres sur une image avec YOLOv4

3.5 Suivi des chèvres

La détection des chèvres est effectuée dans chaque image. Un algorithme de suivi multi objets permet ensuite d'associer un identifiant unique à chaque animal qui sera conservé durant toute la séquence vidéo, pour une analyse du comportement individuel. Pour réaliser le suivi des individus, nous avons implémenté et comparé deux méthodes performantes de l'état de l'art : SORT [107] et Deep SORT [108]. Il s'agit de deux méthodes de suivi par détection où on utilise un modèle d'apprentissage profond pour détecter, dans un premier temps, les objets à suivre et ensuite on utilise des algorithmes d'estimation du mouvement (le filtre de Kalman par exemple) et d'association de données (l'algorithme Hongrois par exemple) pour suivre ces objets détectés. Ce sont des méthodes simples, avec des architectures peu complexes, mais qui peuvent être efficace pour le suivi en temps réel. Le suivi par détection est plus rapide, comparé au suivi par segmentation et au suivi par attention. L'idée ici est de tester dans un premier temps ce type de suivi, vu que la partie détection est déjà effectuée, et par la suite tester une architecture de suivi par régression afin d'avoir une architecture semi ou complètement de bout-en-bout pour le suivi et la détection.

3.5.1 SORT

La méthode SORT comprend principalement 4 étapes :

— La détection et localisation des individus réalisées ici avec YOLOv4 dans chaque image : chaque chèvre est représentée par sa classe (debout ou couchée), les coordonnées de sa boîte englobante (x_c^t = centre x, y_c^t = centre y, w^t = largeur et h^t = hauteur) à l'instant t et un identifiant unique ID.

— L'estimation de mouvement pour prédire la position de chaque objet (détecté à l'instant $t-1$) dans l'image courante de la vidéo est réalisée par un filtre de Kalman : chaque objet est représenté par un vecteur d'état $E_t(x_c^t, y_c^t, v_x^t, v_y^t, w^t, h^t)$ avec v_x^t et v_y^t la vitesse de l'objet par rapport à x et y. On suppose que les chèvres n'ont pas de déplacement très rapide la plupart du temps et donc le filtre de Kalman peut bien estimer le mouvement des objets entre 2 images consécutives. Le filtre de Kalman

utilise le modèle de transition (F) pour décrire comment le système évolue dans le temps. Ici l'état E_t représente la position et la vitesse de chaque objet suivi. Le modèle de transition est utilisé pour estimer l'état E_t et la covariance de l'estimation (P_t), en se basant sur l'état et la covariance à l'instant précédent $(t-1)$ avec les équations :

$$E_t = F \cdot E_{t-1} \tag{3.14}$$

$$P_t = F \cdot P_{t-1} \cdot F^T + w_t \tag{3.15}$$

avec w_t le bruit du processus, P_{t-1} la matrice de covariance à l'instant précédent. On initialise P_t, au début du traitement, en donnant une grande incertitude aux vitesses initiales non observable. Le vecteur d'état (E_t) et la matrice utilisée pour F sont :

$$E_t = \begin{bmatrix} x_t \\ y_t \\ v_x \\ v_y \\ w_t \\ h_t \end{bmatrix} ; F = \begin{bmatrix} 1 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

avec Δt le temps écoulé entre les mesures. Le modèle d'observation (H) est représenté par :

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Ensuite, dans la mise à jour du filtre, le calcul du gain de Kalman (K) est calculé et l'estimation de l'état (E_t) et la matrice de covariance (P_t) sont mises à jour comme suit :

$$K = P_t \cdot H^T (H \cdot P_t \cdot H^T + v_t)^{-1} \tag{3.16}$$

$$\hat{E}_t = E_t + K \cdot (z_t - H \cdot E_t) \tag{3.17}$$

$$\hat{P}_t = (I - K \cdot H) \cdot P_t \tag{3.18}$$

avec $\hat{E}_t$ et $\hat{P}_t$ l'état et la matrice de covariance mis à jour, I une matrice identité, z_t l'observation du système et v_t le bruit d'observation.

— L'association des prédictions de YOLOv4 à celles du filtre de Kalman : nous avons, d'une part,

les prédictions du filtre de Kalman sur chaque objet dans la liste ordonnée $T = (t_1, t_2, ..., t_M)$ qu'on peut appeler liste de trajectoires et, d'autre part, les prédictions de YOLOv4 dans la liste $D = \{d_1, d_2, ..., d_N\}$ qu'on peut appeler liste de détections. Dans T, nous avons les positions des objets, leur ID et leur classe respectifs et dans D nous avons les positions et les classes des nouvelles détections. Nous voulons savoir, parmi les nouvelles positions, lesquelles sont proches des positions prédites par Kalman en comparant la distance de chaque position des objets dans D aux positions des objets dans T. Une matrice de coûts de taille (N, M) est calculée avec N le nombre total d'objets dans D et M le nombre total d'objets dans T. On applique l'algorithme Hongrois [117] sur la matrice de coûts qui va coupler chaque objet de la liste T avec un objet de la liste D en fonction de la distance qui les sépare (les objets avec une faible distance seront couplés). Ainsi, si l'objet d_2 (détection) est couplé avec l'objet t_1 (trajectoire) alors on considère que c'est le même objet et on met à jour la liste T avec ces nouvelles informations de D (la position, les dimensions de la boîte et la classe).

— création et suppression des IDs : dans cette étape on créé un ID unique pour chaque objet détecté et on le propage durant l'étape d'association entre les prédictions. Les objets dans D qui ne sont pas associés sont considérés comme de nouveaux objets et les objets dans T qui ne sont pas associés sont gardés afin de pouvoir les associer dans les prochaines images ou supprimés après un certain nombre d'images. Dans notre cas on considère que les chèvres ne quittent pas définitivement l'enclos donc on garde les objets de T, sur 100 images consécutives, pour les associer de nouveau dans les prochaines images.

3.5.2 Deep SORT

Cette méthode est une extension de l'algorithme SORT avec l'ajout d'un descripteur d'apparence lors du calcul de la distance pour associer deux prédictions. Dans le calcul de la matrice de coûts pour effectuer l'association, Deep SORT [108] utilise la distance de Mahalanobis [141] et la distance cosinus [142] entre les vecteurs de descripteurs d'apparence (prédits par un CNN entraîné hors ligne) pour estimer le score de similarité entre deux objets. Si ce score est inférieur à un certain seuil, les objets sont considérés comme différents et ne sont pas associés ; sinon cela signifie que les deux objets correspondent au même objet. Cependant, dans l'expérience, les poids de la distance cosinus et de la distance de Mahalanobis sont fixés à 1 et 0, respectivement, ce qui signifie que seule la distance cosinus est utilisée pour créer la matrice de coûts.

Pour implémenter le Deep SORT pour le suivi des chèvres, nous avons entraîné un classificateur softmax [35] comme un encodeur capable de générer la représentation d'une chèvre (descripteur d'apparence) afin de la comparer à une autre, au moment du suivi, selon le principe du plus proche voisin [35]. La base d'apprentissage doit contenir les images des chèvres détectées, regroupées par identité, dans les vidéos d'entraînement comme dans le jeu de données MARS [143] utilisé pour la réidentification des personnes. Nous avons généré notre base sur de courtes vidéos (environ 1 minute) en utilisant la méthode SORT

Algorithme 1 : Algorithme de suivi SORT

Entrées : vidéo ;
Sorties : liste mise à jour des trajectoires d'objets $T = (t_1, t_2, ..., t_M)$;
Initialiser une liste vide T de trajectoires d'objets ;
Initialiser un entier $ObjID$ à 0 ;
tant que *pas à la fin de la vidéo* **faire**
    Détecter les objets de l'image courante ;
    Initialiser une liste des détections $D = (d_1, d_2, ..., d_N)$;
    Initialiser une liste DST vide de distance ;
    Initialiser une liste $NObj$ de 1 pour les nouveaux objets de même taille que D ;
    pour *Chaque objet dans la liste des trajectoires* **faire**
        Prédire l'état suivant de cet objet avec le Filtre Kalman ;
        Remplacer l'état existant dans T par l'état prédit;
    fin
    si T *non vide* **alors**
        pour *Chaque objet détecté* d_i **faire**
            Calculer son IoU avec les objets de la liste T ;
            Calculer la distance euclidienne Q entre son centre et les centres des objets T ;
            Calculer sa distance $dist$;
            si $Classe(d_i) \neq Classe(t_j)$ **alors**
                $dist = Q + (1 - IoU) + 500$
            sinon
                $dist = Q + (1 - IoU)$
            fin
            Ajouter la distance $dist$ dans la liste DST ;
        fin
        Générer l'association M avec l'algorithme Hongrois sur les distances dans DST ;
        pour *Chaque indice* $m_{i,j}$ *dans* M **faire**
            si $Classe(t_{m_j}) ==$ "*couche*" **alors**
                Mettre à jour t_{m_j} avec les même informations de T
            sinon
                Mettre à jour t_{m_j} avec les nouvelles informations de détection d_{m_i}
            fin
            Marquer l'indice m_i correspondant dans $NObj$ comme objet existant en mettant sa valeur à 0 ;
        fin
    fin
    pour *Chaque objet détecté* **faire**
        Vérifier si c'est un nouveau objet ;
        si *Nouveau objet* **alors**
            Initialiser une nouvelle trajectoire d'objet t_j ;
            Incrémenter la valeur de $ObjID$ et l'affecter à la trajectoire t_j ;
            Ajouter la nouvelle trajectoire à la liste T
        fin
    fin
    Retourner la liste des trajectoires T ;
fin

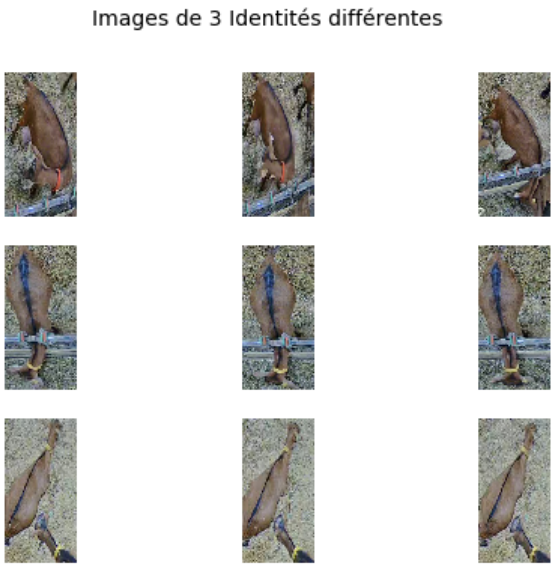


FIGURE 3.18 – Base d'entraînement du descripteur d'apparence sur les chèvres.

déjà implémentée pour extraire les chèvres détectées et les regrouper par leur identifiant de suivi. La figure 3.18 montre un extrait de notre base d'apprentissage (chaque ligne correspond à une chèvre d'une même identité sur trois images successives d'une vidéo). Cette base est constituée de 160 identifiants (ou classes) avec 10400 images de chèvres comme sur la figure 3.18. Nous avons utilisé 80% de ces images pour l'entraînement du descripteurs et 20% pour le test.

Classificateur cosinus softmax [35]

Etant donné un ensemble de données $D = \{(x_i, y_i)\}_{i=1}^{N}$ de N images d'apprentissage $x_i \in \mathbb{R}^D$ et les étiquettes de classe associées $y_i \in \{1, ..., C\}$, l'approche standard de la classification dans le cadre de l'apprentissage profond consiste à traiter les images d'entrée par un ensemble de couches CNN et à placer un classificateur softmax à la fin du réseau afin d'obtenir des scores de probabilité pour chacune des C classes. Le classificateur softmax choisit la classe ayant la probabilité maximale, par la règle de Bayes, en fonction de la fonction paramétrique suivante :

$$p(y = k|r) = \frac{exp(w_k^T r + b_k)}{\sum_{n=1}^{C} exp(w_n^T r + b_n)} = \frac{p(y = k|r)p(y = k)}{\sum_{n=1}^{C} p(r|y = n)p(y = n)} \text{ [35]}, \tag{3.19}$$

avec $w_k = \sum^{-1} \mu_k$ et $b_k = -\frac{1}{2}\mu_k^T \sum^{-1} \mu_k + log\ p(y_i = k)$ et $r = f(x)$, $r \in \mathbb{R}^D$ la représentation des caractéristiques sous-jacentes d'un réseau de codage paramétré qui est entraîné conjointement avec le classificateur. Avec quelques adaptations, le classificateur softmax standard peut être modifié pour produire des groupes compactes dans l'espace de représentation [35]. Ces modifications sont principalement :

— l'application de la normalisation l2 à la dernière couche du réseau de codage pour garantir que la représentation est de longueur unitaire $||f_\theta(x)||_2 = 1,\ \forall x \in \mathbb{R}^D$;

— les poids doivent également être normalisés à une longueur unitaire : $\widetilde{w}_k = w_k/||w_k||_2,\ \forall k = 1, ..., C$.

Ainsi, le nouveau classificateur dénommé cosinus softmax, peut s'énoncer comme suit :

$$p(y = k|r) = \frac{exp(K \cdot \widetilde{w}_k^T r)}{\sum^{-1} exp(K \cdot w_n^T r)}, \tag{3.20}$$

avec K un paramètre d'échelle libre. Cette paramétrisation comporte $C - 1$ paramètres de moins que la formulation standard dans (3.19) car les termes de biais b_k ont été supprimés. C'est donc cette nouvelle formulation (3.20) que nous allons utiliser pour entraîner notre descripteur d'apparence. Le tableau 3.8 nous montre l'architecture du CNN utilisée pour entraîner le descripteur d'apparence. Elle prend en entrée une image de taille 128 x 64 pixels et est constituée de couches convolutives et résiduelles. En sortie nous avons un vecteur descripteur de 128 valeurs. Ce descripteur sera utilisé, dans la phase d'association des données, pour estimer la similarité entre deux objets à partir de leur apparence.

TABLE 3.8 – Architecture du réseau de descripteur d'apparence [35]

Couche	**Taille du patch/Stride**	**Taille de Sortie**
Conv 1	3 × 3/1	32 × 128 × 64
Conv 2	3 × 3/1	32 × 128 × 64
Max Pool 3	3 × 3/2	32 × 64 × 32
Residual 4	3 × 3/1	32 × 64 × 32
Residual 5	3 × 3/1	32 × 64 × 32
Residual 6	3 × 3/2	64 × 32 × 16
Residual 7	3 × 3/1	64 × 32 × 16
Residual 8	3 × 3/2	128 × 16 × 8
Residual 9	3 × 3/1	128 × 16 × 8
Dense 10	-	128
$l_2 - norm$	-	128

Pour l'entraînement, deux fonctions de coût sont utilisées :

— **la perte de triplet (triplet loss)** : la fonction objectif considère des triplets r_a (l'ancre), r_p (un exemple positif) et r_n (un exemple négatif) qui comprennent une paire positive $y_a = y_p$ et une paire négative $y_a \neq y_n$ et contraint la différence de distance entre la paire négative et la paire positive à être supérieure à une marge prédéfinie $m \in \mathbb{R}$:

$$L_t(r_a, r_p, r_n) = \{||r_a - r_n||_2 - ||r_a - r_p||_2 + m\}_+, \tag{3.21}$$

avec $\{\}_+$ la fonction pivot qui s'évalue à 0 pour les valeurs négatives et à l'identité dans le cas contraire.

— **le magnet loss** : est une mesure du rapport de vraisemblance qui force la séparation en termes de distance de chaque échantillon par rapport aux moyennes des autres classes. Cette fonction a été proposée comme une alternative aux formulations de fonction de coût du réseau siamois [144] qui

fonctionne sur la distribution de la classe entière plutôt que sur des échantillons individuels. Cette fonction de coût est définie par l'équation suivante :

$$L_m(y,r) = \left\{ -\log \frac{e^{-\frac{1}{2\delta^2}||r-\hat{\mu}_y||_2^2 - m}}{\sum_{k\in\overline{C}(y)} e^{-\frac{1}{2\delta^2}||r-\hat{\mu}_k||_2^2}} \right\}_+, \tag{3.22}$$

avec $\overline{C}(y) = \{1,...,C\}\backslash\{y\}$, m est un paramètre de marge, $\hat{\mu}_y$ est la moyenne de l'échantillon de la classe y et δ^2 est la variance de tous les échantillons par rapport à la moyenne de leur classe.

La figure 3.19 nous montre les courbes de l'évolution de l'erreur totale (a) (triplet loss et magnet loss) et également de la précision de classification (b) (classement des chèvres comme appartenant à un même identifiant) lors de l'entraînement du descripteur d'apparence sur nos données.

Association des données

Pour effectuer l'association entre les prédictions du filtre de Kalman et les prédictions du détecteur, Deep SORT intègre les informations relatives au mouvement et à l'apparence en combinant deux métriques appropriées :

— pour le mouvement : la distance de Mahalanobis (au carré) entre les états prédits par Kalman et les mesures nouvellement obtenues avec l'équation suivante :

$$d^{(1)}(i,j) = (d_j - y_i)^T S_i^{-1}(d_j - y_i), \tag{3.23}$$

avec (y_i, S_i) la projection de la distribution de la trajectoire i dans l'espace de mesure et d_j la boîte de détection j. Ainsi, l'association entre i et j est définie par : $b_{i,j}^{(1)} = \mathbb{1}[d^{(1)}(i,j) \leq t^{(1)}]$, où $t^{(1)}$ représente le seuil de la distance de Mahalanobis. $b^{(1)} = 1$ si l'association entre la détection j et la trajectoire i est possible ;

— pour l'apparence : un descripteur d'apparence r_j est calculé pour la détection j et la plus petite distance cosinus entre i et j est calculée par :

$$d^{(2)}(i,j) = min\{1 - r_j^T r_k^{(i)} | r_k^{(i)} \in R_i\}, \tag{3.24}$$

avec $\{r_k^{(i)}\}_1^{L_k}$, l'historique des L_k descripteurs d'apparence associés pour chaque trajectoire k. Ainsi, l'association entre la détection j et la trajectoire i, en utilisant cette seconde métrique, est donnée par : $b_{i,j}^{(2)} = 1[d^{(2)}(i,j) \leq t^{(2)}]$

Ces deux métriques se complètent pour résoudre différents aspects de l'association. D'une part, la distance de Mahalanobis fournit des informations sur les emplacements possibles des objets en fonction du mouvement, qui sont particulièrement utiles pour les prédictions à court terme. D'autre part, la distance cosinus prend en compte des informations sur l'apparence qui sont particulièrement utiles pour retrouver des identités après des occultations à long terme, lorsque le mouvement est moins discriminant.

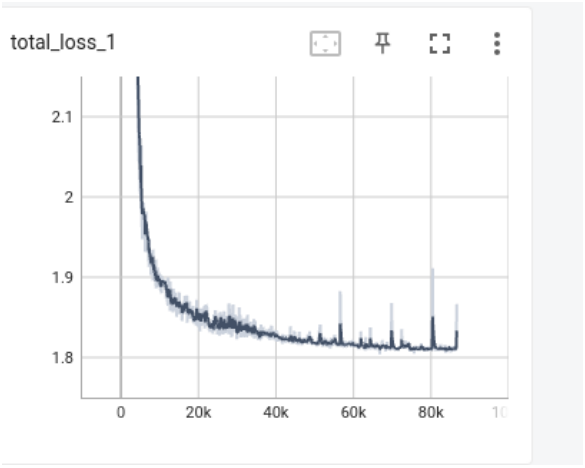
Pour effectuer l'association finale entre la détection i et la trajectoire j, ces deux métriques sont combinées à l'aide d'une somme pondérée comme suit :

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda) d^{(2)}(i,j). \tag{3.25}$$

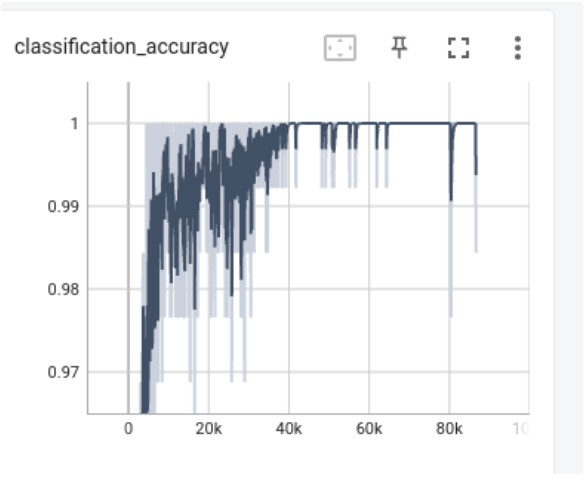
L'influence de chaque métrique sur le coût d'association combiné peut être contrôlée par l'hyperparamètre λ. Dans nos expérimentations, nous avons testé $\lambda = 0$ et $\lambda = 1$ mais leur impact sur le résultat final est négligeable.

Algorithme 2 : Algorithme de suivi Deep SORT

Entrées : vidéo ;
Sorties : liste mise à jour des trajectoires d'objets $T = (t_1, t_2, ..., t_M)$;
Initialiser une liste vide T de trajectoires d'objets ;
Initialiser un entier $ObjID$ à 0 ;
tant que *pas à la fin de la vidéo* **faire**
    Détecter les objets de l'image courante ;
    Initialiser une liste des détections $D = (d_1, d_2, ..., d_N)$;
    Initialiser une liste DST vide de coût de distance ;
    Initialiser une liste $NObj$ de 1 pour les nouveaux objets de même taille que D ;
    pour *Chaque objet dans la liste des trajectoires* **faire**
        Prédire l'état suivant de cet objet avec le Filtre Kalman ;
        Remplacer l'état existant dans T par l'état prédit;
    fin
    si T *non vide* **alors**
        pour *Chaque objet détecté* d_i **faire**
            Extraire son descripteur d'apparence r_i;
            Calculer son coût $c_{i,j}$ avec chaque trajectoire t_i ;
            Ajouter $c_{i,j}$ dans la liste DST ;
        fin
        Générer l'association M sur les coûts $c_{i,j}$;
        pour *Chaque indice* $m_{i,j}$ *dans* M **faire**
            Mettre à jour t_{m_j} avec les nouvelles informations de détection d_{m_i} ;
            Marquer l'indice m_i correspondant dans $NObj$ comme objet existant en mettant sa valeur à 0 ;
        fin
    fin
    pour *Chaque objet détecté* **faire**
        Vérifier si c'est un nouveau objet ;
        si *Nouveau objet* **alors**
            Initialiser une nouvelle trajectoire d'objet t_j ;
            Extraire son descripteur r_k ;
            Incrémenter la valeur de $ObjID$ et l'affecter à la trajectoire t_j ;
            Ajouter la nouvelle trajectoire et son descripteur à la liste T
        fin
    fin
    Retourner la liste des trajectoires T ;
fin



(a) Courbe de l'erreur d'entraînement du descripteur d'apparence CNN



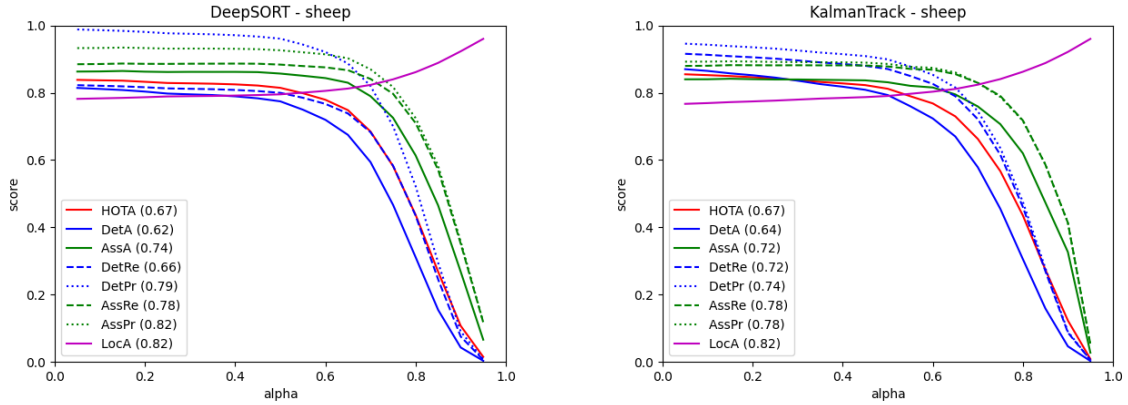
(b) Courbe de la précision de classification du descripteur d'apparence lors de l'entraînement

FIGURE 3.19 – Résultats de l'entraînement du descripteur d'apparence sur les chèvres

3.5.3 Résultats et interprétations

Pour évaluer les performances des deux méthodes de suivi des chèvres, nous avons utilisé le protocole d'évaluation de suivi d'objets du "MOT Challenge" [130]. Nous avons annoté manuellement 4 séquences vidéos d'environ 1 minute chacune (environ 1500 images au total). Ces vidéos sont constituées de scènes de nuit comme de jour et avec des mouvements plus ou moins rapides des chèvres. Nous avons ensuite utilisé la méthode d'évaluation HOTA [145] sur les trajectoires prédites par rapport à la vérité terrain annotée manuellement par nos soins.

La figure 3.20 nous montre l'évaluation des 4 vidéos. Nous pouvons y observer les scores de chaque métrique de HOTA en fonction du seuil d'IoU (alpha). D'après le protocole d'évaluation du "MOT Challenge", chaque vidéo est évaluée individuellement et ensuite une moyenne est calculée sur les résultats de toutes les vidéos pour donner l'évaluation finale. Pour la méthode de suivi SORT, nous avons une précision moyenne d'association (AssA) égale à 72 % et dans la méthode Deep SORT (avec $\lambda = 1$) cette valeur est de 74 %. Cela indique que, dans la méthode Deep SORT, l'association est légèrement mieux effectuée par rapport à la méthode SORT. La précision moyenne de l'association ici mesure l'efficacité de la méthode de suivi à relier, dans le temps, les détections aux mêmes identités. On peut expliquer la valeur légèrement élevée sur l'association dans Deep SORT par le fait que l'association prend en compte les caractéristiques CNN, en plus du mouvement. Cependant, comme on peut le voir dans le tableau 3.9, le temps de traitement dans Deep SORT est beaucoup plus important à cause de l'extraction des caractéristiques CNN qui est gourmande en puissance de calcul (50 images par seconde pour SORT contre 12 images par seconde pour Deep SORT). Le rappel de détection (DetRe) mesure à quel point la méthode de suivi trouve toutes les détections réelles, tandis que la précision de détection (DetPr) mesure à quel point la méthode de suivi parvient à ne pas prédire des détections supplémentaires qui n'existent pas. Dans Deep SORT, DetPr (79%) est plus élevé comparé à SORT (74%). On peut dire que moins



(a) Évaluation du Deep SORT

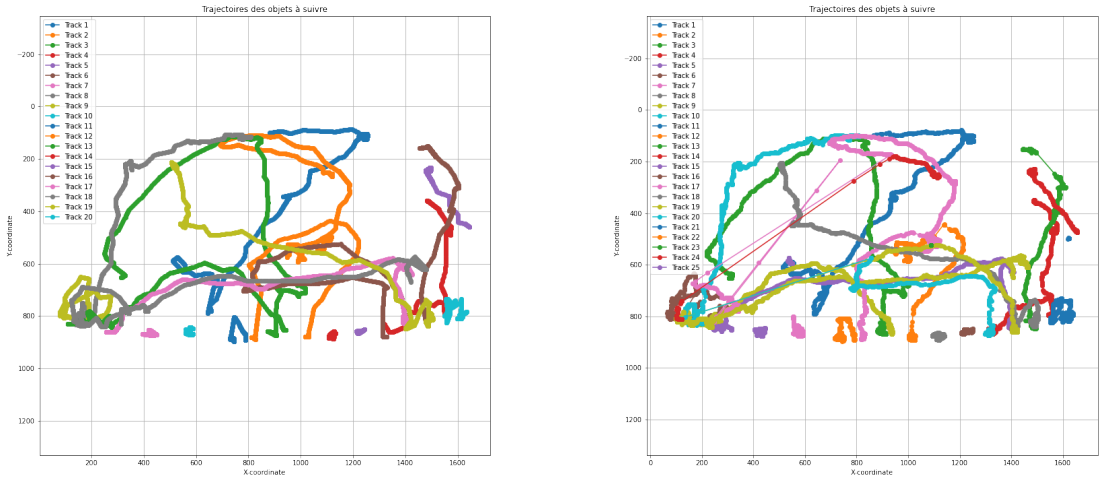
(b) Évaluation de SORT

FIGURE 3.20 – Résultats de l'évaluation des méthodes de suivi avec la métrique HOTA

de détections supplémentaires sont générées dans Deep SORT que dans SORT. La différence entre le DetRe (66%) et le DetPr (79%) dans Deep SORT est assez grande et peut indiquer un suivi instable. Par contre, DetRe (72%) est plus élevé dans SORT ce qui indique que SORT arrive à mieux trouver toutes les détections réelles et est plus stable au niveau du DetPr (74%). La LocA correspond à la précision de la localisation du détecteur et elle est la même pour les deux méthodes de suivi, vu qu'on utilise le même détecteur.

Le tableau 3.9 nous montre également la comparaison entre les méthodes SORT et Deep SORT implémentées pour le suivi des chèvres, sur d'autres métriques d'évaluation comme MOTA et MOTP [130]. Ces résultats nous confirment la stabilité du suivi avec la méthode SORT.

Les figure 3.21 et 3.22 nous montrent la comparaison entre les trajectoires prédites par notre méthode de suivi (SORT) et les trajectoires de la vérité terrain, annotées manuellement, de deux vidéos différentes. Il s'agit des vidéos de Lusignan, d'une durrée de 1 minute 30 secondes environ, dans lesquelles nous avons 20 chèvres par enclos. Un fichier texte est généré pour chaque vidéo et contient, pour chaque ligne, l'image, l'ID, les coordonnées (x, y, w, h) de la boîte englobante d'une chèvre détectée dans la vidéo. Ensuite on extrait les lignes ayant le même ID ainsi que les coordonnées (x, y) du centre de la boîte englobante correspondante à cet ID qu'on affiche dans un plan 2D de coordonnées (x,y). La figure 3.21 correspond à un cas où les chèvres sont très mobiles avec beaucoup de croisements et on peut y observer que les trajectoires prédites par notre méthode de suivi sont assez proches des trajectoires de la vérité terrain. On remarque aussi le nombre d'IDs qui est de 25 dans les trajectoires prédites, au lieu de 20, et quelques changements d'ID. Cela s'explique par des fausses détections du détecteur ou des fausses estimations du filtre de Kalman mais qui sont supprimées après quelques images (après 100 images où ces détections/estimations n'ont pas été associées). On observe aussi quelques changements d'ID sur la figure 3.21 (b)(changement de couleur sur une même trajectoire) dûs à une mauvaise association, lorsque les chèvres sont très proches par exemple. Par contre, sur la figure 3.22, on observe que les trajectoires prédites sont les mêmes que les trajectoires de la vérité terrain et le nombre total de détections reste 20



(a) Trajectoires de la vérité terrain

(b) Trajectoires prédites par SORT

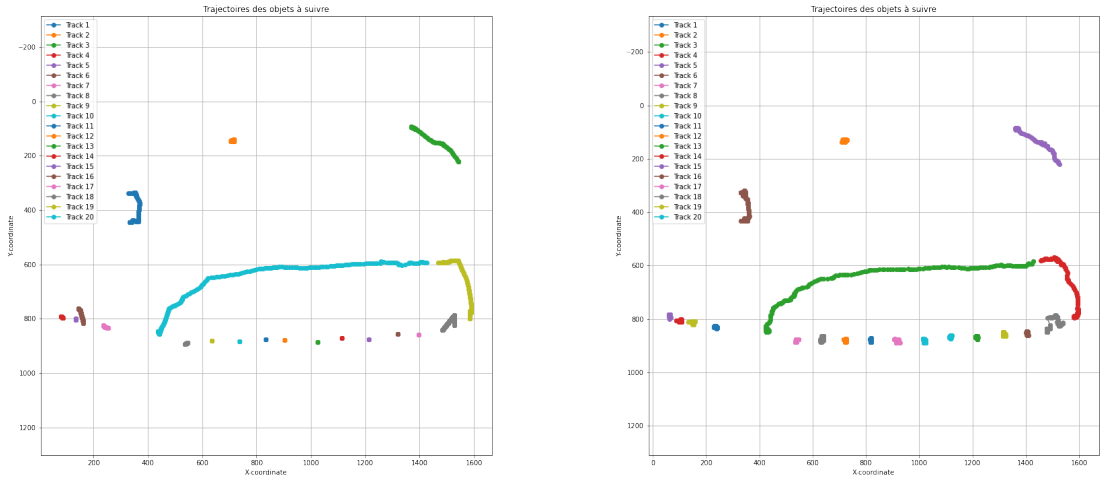
FIGURE 3.21 – Comparaison des trajectoires pour une vidéo où les chèvres sont très mobiles

car c'est une vidéo où les chèvres ne se déplacent pas beaucoup et se croisent donc très peu. Vu que les mouvements ici sont plus lents, le filtre de Kalman arrive à prédire correctement les trajectoires (pas de fausses estimations du mouvement).

3.6 Conclusion

Dans ce chapitre, nous avons présenté les expérimentations effectuées sur la détection et le suivi des chèvres, sur une base de données créée par nos soins. Pour la détection nous avons testé et comparé deux méthodes populaires de la littérature : YOLOv4 (méthode à une étape) et Faster R-CNN (méthode à deux étapes). La deuxième base (5330 images au total) enrichie de vidéos provenant des 3 fermes différentes permet d'entraîner les réseaux avec de meilleures performances. Parmi les deux méthodes de détection, YOLOv4 présente de meilleures performances avec une précision moyenne de plus de 98% sur les deux classes (debout et couché) et s'avère 2.5 fois plus rapide que Faster R-CNN. Pour le suivi des chèvres, nous avons testé et comparé également deux méthodes populaires de la littérature : SORT et Deep SORT. L'évaluation sur les vidéos de test montre une légère amélioration de Deep SORT par rapport à SORT en terme d'association des données. Ce qui peut s'expliquer par l'utilisation du descripteur d'apparence du Deep SORT. Cependant, SORT reste plus rapide et plus adapté à un système temps réel. Les deux méthodes fonctionnent assez bien pour le suivi et quelques cas d'occultations de courtes durées.

Les résultats obtenus dans ces expérimentations nous montrent que la vidéo peut être utilisée pour extraire des informations sur l'état ou le comportement des animaux en situation d'élevage. Dans le chapitre 5, nous allons présenter comment nous avons utilisé le système de détection et de suivi SORT pour effectuer l'analyse de quelques comportements chez les chèvres.



(a) Trajectoires de la vérité terrain

(b) Trajectoires prédites par SORT

FIGURE 3.22 – Comparaison des trajectoires pour une vidéo où les chèvres sont moins mobiles

TABLE 3.9 – Comparaison entre SORT et Deep SORT

Métrique	SORT	Deep SORT
HOTA	67%	67%
AssA	72%	74%
LocA	82%	82%
MOTA	83%	79%
MOTP	79%	78%
FPS	50	12

Chapitre 4

Proposition d'amélioration pour la détection

4.1 Introduction

Dans ce chapitre nous allons présenter notre proposition d'amélioration de la détection de notre système. Bien que les algorithmes de détection soient généralement très efficaces, ils présentent parfois des échecs dans la détection des animaux, ce qui rend le suivi complexe. L'idée est d'explorer les avantages potentiels de l'intégration des détections antérieures dans les algorithmes de détection. Étant donné que notre travail se concentre sur des vidéos avec des objets qui restent généralement dans le champ de vision, cette approche pourrait améliorer la cohérence des détections. Par ailleurs, nous souhaitons investiguer la possibilité de fusionner les aspects détection et suivi en développant une approche plus avancée. Cela impliquerait de propager de manière cohérente les identifiants des animaux détectés d'une image à la suivante, afin d'obtenir une solution plus complète et intégrée.

Dans le suivi d'objets avec SORT, les traces des objets détectés dans les images précédentes sont maintenues et utilisées pour les retrouver dans les images suivantes. Ainsi, une perte de détection sur l'image suivante peut parfois induire en erreur la méthode de suivi. La Figure 4.1 illustre un exemple de cette situation avec trois objets détectés (d_1, d_2, et d_3) à l'instant t, associés à trois trajectoires (t_1, t_2, et t_3) en fonction de la distance qui les sépare. Le tableau coloré sur cette figure représente les coûts ou les distances entre chaque couplage à un instant donné. Durant la phase d'association dans SORT, l'objet précédent est associé à l'objet actuel avec lequel il a la plus petite distance (ou le plus grand recouvrement IoU). Si un objet n'est plus détecté à un moment donné et qu'un nouvel objet est détecté au même endroit (comme l'objet d_3 perdu sur la Figure 4.1), la trace de l'objet perdu est affectée au nouvel objet (objet d_4 sur la Figure 4.1) à cet instant, entraînant ainsi un changement d'identité. Bien que l'utilisation du descripteur d'apparence du Deep SORT puisse être envisagée pour vérifier si l'objet détecté correspond vraiment à l'objet dans la trajectoire, elle est peu robuste dans notre cas, car les chèvres se ressemblent

beaucoup. Une détection plus stable, avec moins de pertes entre les images intermédiaires, améliorerait significativement les résultats du suivi.

Comme nous envisageons d'intégrer à la fois la détection et le suivi d'objets, il serait plus avantageux d'opter pour une approche sans boîtes d'ancrage, telle que YOLOX. L'architecture de YOLOX est basée sur une approche plus souple, permettant au modèle de prédire les paramètres de la boîte englobante directement à partir des points d'ancre. Ces points d'ancre peuvent être remplacés par les centres des détections antérieures en les intégrant, sous forme de "heatmap", dans la détection sur les images suivantes. Ainsi, on peut prédire directement le décalage avec le centre de l'objet détecté précédemment et effectuer le suivi en même temps. La flexibilité de YOLOX peut être aussi avantageuse dans les cas où la distribution des tailles et des formes des objets n'est pas bien représentée par les boîtes d'ancrage prédéfinies. L'architecture YOLOX donne également des résultats comparables à YOLOv4.

4.2 Approche proposée

Une solution pour améliorer la détection serait d'utiliser l'information spatio-temporelle dans la phase de détection, comme dans [146], afin de permettre au modèle de se rappeler des objets précédents et d'améliorer sa précision sur l'image actuelle. Dans [22], les auteurs proposent également une autre façon d'exploiter la corrélation entre les caractéristiques de deux images consécutives d'une vidéo pour ajuster les variances des boîtes englobantes d'un même objet sur plusieurs images consécutives. Le fait que la plupart de ces travaux utilisent un détecteur à deux étapes et sont basés sur les boîtes d'ancrages rend difficile la détection en temps réel sur les vidéos.

Pour améliorer la détection des objets dans les vidéos, et faire toujours du temps réel, nous proposons une approche qui va permettre au modèle de prendre en compte les informations spatiales (sur les positions) des détections précédentes pour faire la détection sur l'image courante. L'idée ici est d'aider le modèle à se rappeler des objets qu'il a déjà détecté précédemment (en supposant que ces objets seront toujours présents dans les images suivantes) et, en même temps, lui permettre de détecter les nouveaux objets. Pour cela, nous allons introduire dans le réseau de détection, les positions des objets précédents, et pondérer la carte de caractéristiques avec une carte de chaleur basée sur les objets précédents. Dans notre approche, la carte de chaleur est générée à partir des résultats de la détection précédente et est agrégée avec la carte de caractéristiques de l'image courante. La carte de chaleur correspond à une image en noir et blanc où les pixels correspondants au centre d'un objet ont une valeur à 1 et les autres pixels sont à 0. Cette approche est inspirée de la méthode de suivi "CenterTrack" [23]. Pour mettre en place notre approche, nous allons utiliser l'architecture de YOLOX [18] car c'est une architecture de détection à une étape, qui n'utilise pas de boîtes d'ancrage, plus rapide et plus adaptée pour un système temps réel. Cette architecture se base également sur les points de caractéristiques (comme le centre des objets) pour effectuer la régression des boîtes des objets. On pourra exploiter, plus tard, cette information pour ajuster les positions des objets précédents vers leur nouvelle position dans l'image courante et ainsi effectuer le suivi de bout en bout. En effet, les centres de deux objets sont rarement situés au même endroit et peuvent

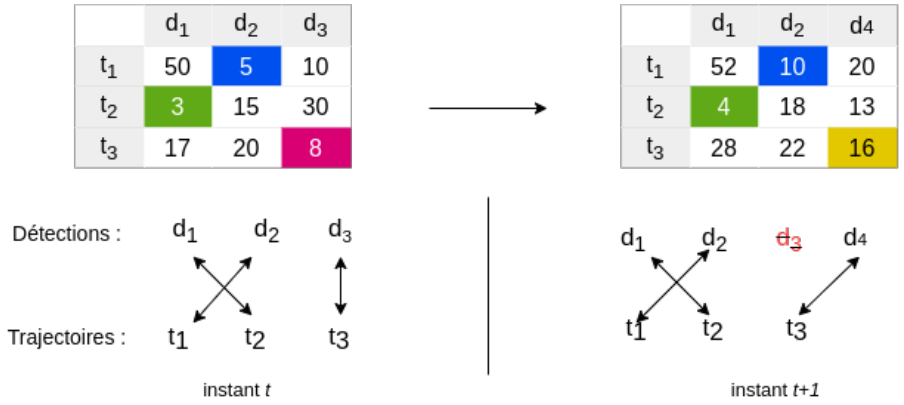


FIGURE 4.1 – Exemple de changement d'identifiant de suivi d'un objet

être robustes aux occultations. Dans les sections suivantes, nous allons présenter le fonctionnement de l'architecture YOLOX et les modifications effectuées pour l'adapter à notre approche.

4.3 Architecture de YOLOX

L'architecture YOLOX est une version de la famille des détecteurs à une étape YOLO. Cette version permet d'améliorer les performances des versions précédentes de YOLO. Elle est principalement basée sur l'architecture de YOLOv3 [31] et comprend 3 parties principales : le réseau de base ("backbone"), le réseau intermédiaire ("neck") et le réseau de prédiction ("head"). YOLOX utilise également une nouvelle stratégie d'affectation de labels SimOTA [147] et la régression des boîtes englobantes est basée sur la fonction "GIoU Loss" [137]. C'est une architecture qui n'utilise pas de boîtes d'ancrage, mais utilise les points d'intérêt (centre des objets) comme ancre. Il existe 6 versions de YOLOX, en fonction de la profondeur et de la largeur des couches de l'architecture. Ces versions sont regroupées en deux grandes catégories :

— version "light" : YOLOX-Nano, YOLOX-Tiny ;

— version standard : YOLOX-S, YOLOX-M, YOLOX-L et YOLOX-X.

Le tableau 4.1 nous présente les détails sur chaque version de YOLOX.

TABLE 4.1 – Caractéristiques des versions de YOLOX

Model	Taille entrée	Temps (ms)	Paramètres (10^6)	FLOPs (10^9)
YOLOX-Nano	416	-	0.91	1.08
YOLOX-Tiny	416	-	5.06	6.45
YOLOX-S	640	9.8	9.0	26.8
YOLOX-M	640	12.3	25.3	73.8
YOLOX-L	640	14.5	54.2	155.6
YOLOX-X	640	17.3	99.1	281.9

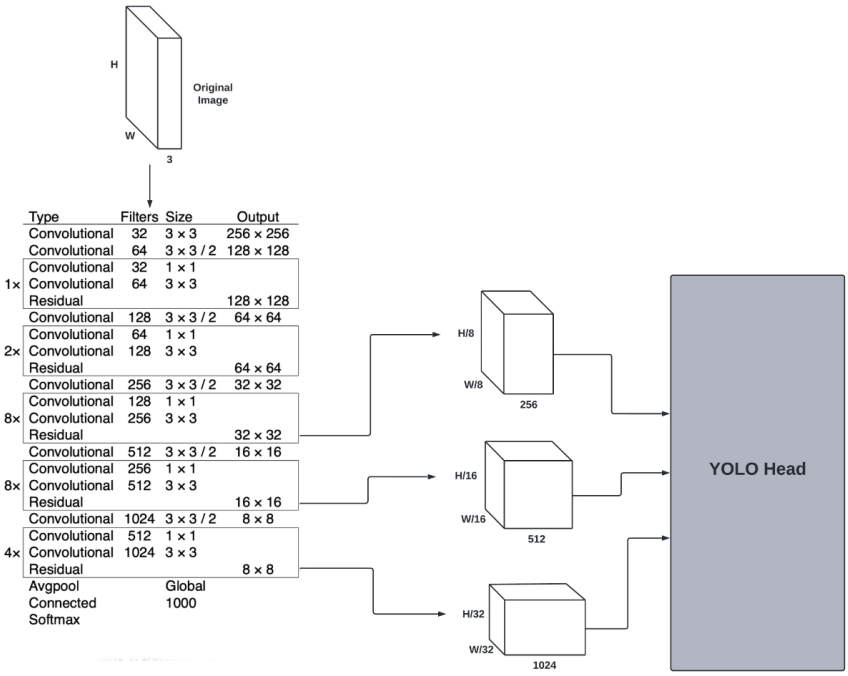


FIGURE 4.2 – Architecture de YOLOv3 [32].

Le réseau de base : "backbone"

Ce réseau est responsable de l'extraction des caractéristiques sur l'image d'entrée. Dans YOLOv3, le réseau DarkNet53 est utilisé. Son architecture est constituée des couches de convolutions 1×1, des connexions résiduelles et des couches de convolutions 3×3 (figure 4.2) pour créer un extracteur de caractéristiques très puissant. Il s'agit de la principale amélioration apportée à YOLOv3. YOLOX utilise également le DarkNet53 comme réseau de base et aussi une autre version du DarkNet53 dénommée CSP-DarkNet53 [28]. On peut retrouver la différence entre ces deux réseaux dans la section 3.4.2, page 57 de ce document. Dans YOLOX, le CSPDarkNet53 est constitué du CSPBlock et de l'activation SiLU [148]. La fonction d'activation SiLU est une version améliorée des fonctions Sigmoïde et ReLU.

Le réseau intermédiaire : "neck"

Ce réseau est utilisé pour la fusion des caractéristiques en utilisant le réseau pyramidal des caractéristiques (FPN) [16] et le réseau d'agrégation des chemins (PAN) [140]. Tout d'abord, le FPN est utilisé pour transférer et fusionner les informations sur les caractéristiques de haut niveau par une réduction d'échelle, puis la carte des caractéristiques prédites est obtenue par une augmentation d'échelle et une fusion par le biais du PAN. Le FPN améliore la capacité de détection des petits objets et le PAN transmet mieux les informations de la couche inférieure à la couche supérieure.

Le réseau de prédiction : "head"

La prédiction dans YOLOX est constituée d'une classification et d'une régression. Trois cartes de caractéristiques à différentes échelles sont obtenues par le réseau intermédiaire et utilisées respectivement

pour identifier les objets de grande, moyenne et petite taille. Chaque carte de caractéristiques peut être considérée comme une collection de points avec un paramètre de position (centre de l'objet) et le nombre de filtres. Par rapport aux méthodes basées sur les boîtes d'ancrage, les détecteurs sans ancre ont deux tiers de paramètres en moins et s'exécutent plus rapidement tout en étant plus performants. Dans la détection d'objets, le conflit entre les tâches de classification et de régression est un problème bien connu [149], [150]. La tête découplée pour la classification et la localisation est donc largement utilisée dans la plupart des détecteurs à une ou à deux étapes. Ainsi, les auteurs de YOLOX remplace la tête de détection de YOLOv3 par une tête découplée, comme on peut l'observer sur la figure 2.21 du chapitre 2, qui améliore considérablement la vitesse de convergence. La sortie de YOLOX correspond donc à 3 tenseurs contenant chacun des informations différentes, au lieu d'un seul tenseur contenant toutes les informations comme dans YOLOv3 :

— la classe de chaque boîte englobante ;

— les coordonnées (x, y, w, h) de chaque boîte englobante ;

— le score de confiance du réseau sur la présence d'un objet dans la boîte englobante.

Chaque sortie de YOLOX a sa propre fonction de coût puisqu'elle doit être optimisée différemment. Pour l'optimisation de la classe, le modèle prédit un vecteur de taille H x W x C, où C représente le nombre de classes. Chaque élément de ce vecteur représente donc la probabilité d'une classe, ou le degré de confiance avec lequel le modèle pense que la classe prédite est celle qui se trouve dans la boîte englobante. L'encodage one-hot [1] est utilisé pour encoder la classe de la boîte englobante de la vérité terrain pour chaque prédiction. La fonction de coût de l'entropie croisée binaire ("BCEWithLogits Loss") est appliquée sur les deux vecteurs (vérité terrain et prédiction) afin d'optimiser la fonction objectif de la classification.

Pour la régression des coordonnées de la boîte englobante, YOLOX utilise le score IoU qui évite le problème de surapprentissage de la régression connue avec la fonction de coût quadratique (MSE) utilisée dans YOLOv3. Le score IoU est calculé entre la boîte englobante prédite et la boîte de vérité terrain. La fonction de coût est donc calculée en faisant $1 - IoU$. La fonction de coût GIou Loss peut être également utilisée pour la régression.

Pour la dernière sortie, qui correspond au degré de confiance sur la présence d'un objet, la fonction de coût de l'entropie croisée binaire (BCE) est utilisée comme pour la classification. Comme chaque exemple positif est associé à une boîte de vérité terrain, on peut calculer l'IoU entre la boîte englobante prédite et la boîte englobante de vérité terrain pour obtenir la valeur que nous voulons que le modèle prédise comme score de confiance. Ensuite, on introduit le score de confiance prédite et l'IoU dans la fonction de coût BCE afin d'obtenir l'erreur pour cette prédiction. La fonction objectif finale pour l'entraînement de YOLOX est obtenue avec l'équation suivante :

$$L = \frac{1}{N_{pos}} L_{cls} + reg_{weight} * \frac{1}{N_{pos}} L_{reg} + \frac{1}{N_{pos}} L_{obj} \tag{4.1}$$

1. https ://fr.wikipedia.org/wiki/Encodage_one-hot

avec N_{pos} le nombre d'exemples positifs, L_{cls}, L_{reg} et L_{obj} les fonctions de coût respectives de la classification, la regression et la confiance ; reg_{weight} le poids de la régression.

Points de caractéristiques ou points d'ancre

Dans la détection d'objets, une boîte d'ancrage est une boîte englobante prédéfinie qui aide le réseau de détection dans sa prédiction. Les boîtes d'ancrage sont donc des paramètres supplémentaires. Combien d'ancres le modèle doit-il utiliser ? Quelle doit être la taille des ancres ? Ces questions conduisent à plus de réglages d'hyperparamètres et à moins de diversité dans le modèle. Dans YOLOX, le modèle prédit directement les dimensions de la boîte englobante, plutôt que de prédire un décalage par rapport à une boîte d'ancrage. Pour faire cela, YOLOX utilise les pas (stride en anglais) pour diviser les cartes de caractéristiques en plusieurs cellules et l'intersection de chaque cellule est utilisée comme point d'intérêt ou point d'ancre (figure 4.3). À l'aide de ces cellules, nous pouvons attribuer chaque prédiction à chacun des points d'intersection de la cellule, puis utiliser ce point comme décalage pour mettre à l'échelle la boîte englobante prédite. Un point d'ancre est donc un décalage permettant de déplacer l'emplacement (x,y) d'une prédiction. Pour obtenir la localisation (x, y) du point d'ancre, par rapport à l'image réelle, on utilise la formule suivante :

$$\begin{cases} x &= s/2 + s * i \\ y &= s/2 + s * j \end{cases} \tag{4.2}$$

avec s le "stride", i et j représentent les coordonnées du point d'intersection sur la cellule. Ainsi, lorsque le modèle nous prédit les coordonnées x_p, y_p, w_p, h_p, on calcule les coordonnées réels de la boîte sur l'image originale avec la formule suivante :

$$\begin{cases} x_b &= x_p * s + x \\ y_b &= y_p * s + y \\ w_b &= s * e^{w_p} \\ h_b &= s * e^{h_p} \end{cases} \tag{4.3}$$

Avec cette équation, on déplace le point prédit en l'ajoutant au point d'ancre et on le met à l'échelle de l'image grâce à s. Ensuite, on dénormalise la largeur w_p et la hauteur h_p en s'assurant qu'elles ne sont pas négatives à l'aide d'une fonction exponentielle et on les met à l'échelle de l'image grâce à s.

Affectation de labels : SimOTA

Dans les modèles de détection d'objets, toutes les prédictions ne sont pas correctes. Pour faire la différence entre les bonnes et les mauvaises prédictions, YOLOX utilise une technique d'affectation SimOTA, qui permet d'attribuer des étiquettes, de manière dynamique, aux prédictions. Grâce à cette affectation, on attribue aux mauvaises prédictions des étiquettes négatives (exemples négatifs ou arrière plan) et aux bonnes prédictions, des étiquettes positives (exemples positifs ou vrais objets). SimOTA a besoin d'objets de vérité terrain pour attribuer les étiquettes. Il n'est donc utilisé que pendant l'entraînement du modèle

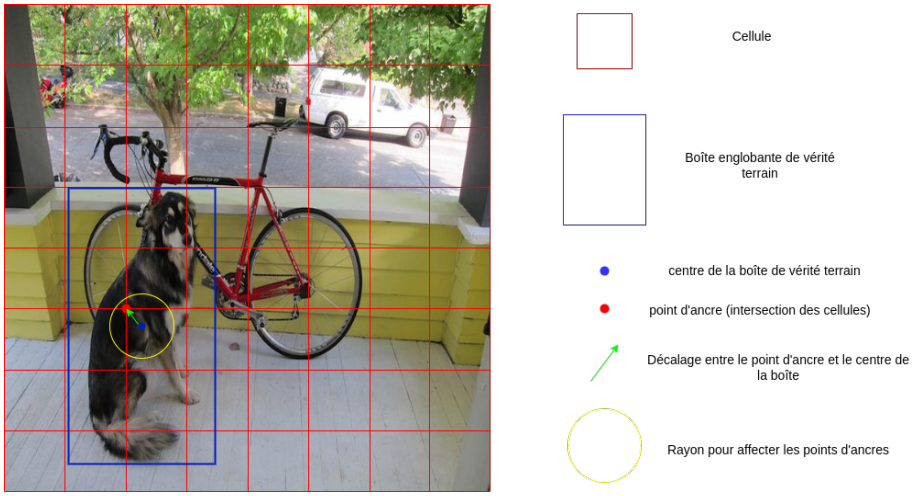


FIGURE 4.3 – Exemple d'affectation de point d'ancre.

et non pendant l'inférence. L'affectation de labels aide le modèle à être plus stable durant l'entraînement. Pour une image de taille 256 x 256 pixels en entrée, nous aurons 1344 prédictions de YOLOX (3 échelles de prédiction avec les strides 32, 16 et 8) et au lieu d'optimiser toutes ces prédictions, on utilise l'affectation des labels pour extraire les meilleures prédictions afin d'optimiser que ces dernières. La raison pour laquelle l'entraînement est plus stable est que le modèle doit faire face à moins de mises à jour du gradient.

L'affectation des étiquettes peut se faire de plusieurs manières. L'une des méthodes les plus courantes consiste à trouver l'IoU le plus élevé entre une boîte englobante de vérité terrain et toutes les autres boîtes englobantes prédites. L'affectation indépendante d'étiquettes positives/négatives pour chaque vérité de terrain sans contexte pourrait être sous-optimale et peut conduire à une mauvaise prédiction. Pour rendre l'affectation plus optimale, les auteurs de YOLOX suggèrent d'utiliser le contexte global de l'image pour attribuer les étiquettes plutôt que le contexte local. Ils proposent donc la méthode OTA [147] où l'affectation des étiquettes est analysée d'un point de vue global et formulée comme un problème de transport optimal (OT). Cependant, dans la pratique, il fut constaté que la résolution du problème OT via l'algorithme de Sinkhorn-Knopp nécessite 25% de temps d'apprentissage supplémentaire, ce qui est assez coûteux pour l'apprentissage sur plusieurs itérations. La méthode d'affectation SimOTA permet d'obtenir une approximation de l'affectation en une seule itération. SimOTA calcule d'abord le degré du couplage par paire, représenté par le coût c_{ij} de chaque paire de prédiction (p_j - vérité terrain g_i). Le coût est calculé comme suit :

$$c_{ij} = L_{ij}^{cls} + \lambda L_{ij}^{reg} \tag{4.4}$$

avec λ un coefficient de pondération, L_{ij}^{cls} et L_{ij}^{reg} sont respectivement l'erreur de classification et l'erreur de régression entre la prédiction p_j et la vérité terrain g_i. Ensuite, pour g_i, on sélectionne les k prédictions les moins coûteuses dont les boîtes englobantes renferment les points d'ancre positifs dans une région centrale fixe (rayon de la figure 4.3) en tant que prédictions positives. Enfin, les points correspondants à

ces prédictions positives sont considérés comme exemples positifs, tandis que les autres points sont des exemples négatifs. La figure 4.3 nous montre un exemple de sélection des points d'ancre positifs (points d'intersection des cellules), c'est-à-dire les points d'ancre qui sont affectés à la boîte de vérité terrain de l'objet.

Augmentation des données

L'augmentation des données est un moyen d'aider un modèle à se généraliser. YOLOX utilise certaines méthodes d'augmentations de données, vu précédemment dans YOLOv4, pour aider le réseau à mieux généraliser :

— mosaïque : permet au modèle de reconnaître des objets à différents endroits, ce qui élimine le besoin d'un contexte spécifique unique. Elle combine 4 images aléatoires en une seule image. Les 4 images sont redimensionnées et réparties dans les 4 coins de l'image finale. Les boites englobantes des objets de ces images sont également redimensionnées et placées sur l'image finale ;

— mixup : génère une combinaison pondérée de paires d'images aléatoires à partir des données d'apprentissage. Étant donné deux images et leurs étiquettes de vérité terrain, (x_i, y_i) et (x_j, y_j), un exemple synthétique $(\hat{x}, \hat{y})$ d'apprentissage est généré comme suit :

$$\begin{cases} \hat{x} &= \lambda x_i + (1-\lambda) x_j \\ \hat{y} &= \lambda y_i + (1-\lambda) y_j \end{cases} \tag{4.5}$$

avec λ tiré de la distribution $Beta(\alpha = 0.2)$.

4.4 Test de YOLOX original

Notre objectif principal ici est d'améliorer la détection des objets en évitant les échecs de détection sur certaines images intermédiaires des vidéos. Pour cela, nous avons construit une autre base de données composée de séquences d'images comme dans le jeu de données MOT20 [2]. En effet, pour tester notre approche, il nous faut une vérité terrain avec des paires d'images consécutives. Le jeu de données 2 ne contient aucune paire, et nous n'avons pas les moyens techniques d'accéder aux vidéos dont sont issues les images du jeu de données 2. Il est donc nécessaire de créer un nouveau jeu de données pour entraîner et tester notre nouvelle approche.

Pour construire notre base d'images, nous avons annoté 27 séquences vidéos de 25 FPS, d'une durée entre 30 secondes et 1 minute chacune, provenant des 3 fermes d'observation (Nohant, Avord et Lusignan). Parmi ces séquences, 24 sont utilisées pour l'entraînement (3462 images) et 3 pour le test (1777 images). On a environ entre 30 et 1500 images, pour chaque séquence, annotées manuellement avec l'outil d'annotation CVAT. Le format de vérité terrain généré à travers cette annotation est le même que ce-

2. https ://motchallenge.net/data/MOT20/

lui du MOT20. Nous avons toujours les deux classes ("chevre_debout" et "chevre_couche") pour cette nouvelle base d'images que nous allons nommer **jeu de données 3**.

Nous avons choisi la version YOLOX-S pour effectuer les tests. Nous avons d'abord entraîné et testé l'architecture original de YOLOX-S sur le jeu de données 2. Comme on peut l'observer dans le tableau 4.2, YOLOX-S est légèrement plus performant que YOLOv4 et Faster R-CNN testés dans le chapitre précédent.

Le tableau 4.3 nous montre les résultats de la précision moyenne de de YOLOX-S entraîné et testé sur le jeu de données 3. On observe que les performances, ici, ont baissé par rapport au jeu de données 2. Cela peut s'expliquer par le fait qu'on n'a pas beaucoup d'images variées dans la base d'entraînement du jeu de données 3, à cause des images successives que se répètent, par rapport au jeu de données 2 qui contient 5330 images variées pour l'entraînement.

TABLE 4.2 – Évaluation de YOLOX-S entraîné et testé sur le jeu de données 2

Jeu de données 2 : test sur 592 images	
Métrique	Valeur
mAP@.5	98.64%
AP@.5 chevre_couche	99.25%
AP@.5 chevre_debout	98.03%
FPS	131

TABLE 4.3 – Évaluation de YOLOX-S entraîné et testé sur le jeu de données 3

Jeu de données 3 : test sur 1777 images	
Métrique	Valeur
mAP@.5	81.75%
AP@.5 chevre_couche	83.28%
AP@.5 chevre_debout	80.22%
FPS	131

4.5 Modification proposée

Afin d'améliorer la détection des objets dans les vidéos, nous proposons une approche basée sur l'utilisation d'une carte de chaleur ("heatmap") des détections précédentes dans une architecture de détection à une étape. La "heatmap" est une carte de chaleur où les pixels, se trouvant dans un rayon central des objets, ont une valeur 1, et les autres pixels ont une valeur 0 (figure 4.4). Cette carte met en valeur les régions de l'image où se trouve le centre d'un objet. Dans "CenterTrack" [23], les auteurs utilisent l'image courante, l'image précédente et les détections précédentes pour effectuer le suivi des objets. Cela permet au réseau d'estimer l'évolution de la scène et de retrouver éventuellement des objets occultés à l'instant t à partir de preuves visuelles à l'instant $t-1$. Leur architecture est basée sur le détecteur d'objets "CenterNet" [96] qui identifie chaque objet par son point central. Afin d'intégrer facilement les détections précédentes dans leur réseau, les auteurs de "CenterTrack" proposent d'utiliser une carte de chaleur, dénommée "heatmap" qui va mettre en valeur les centres des détections précédentes.

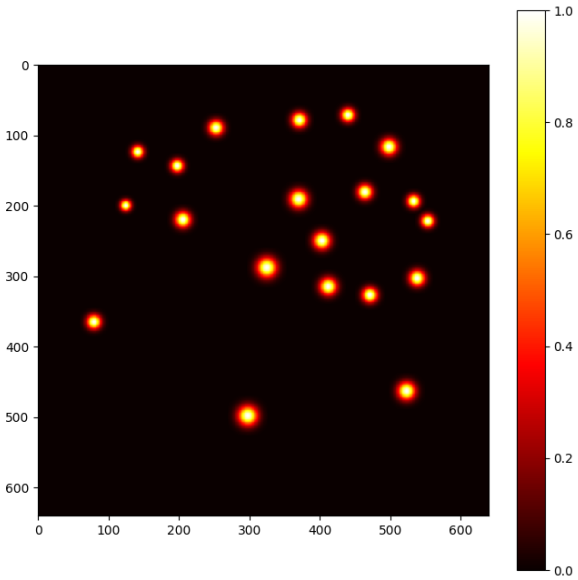


FIGURE 4.4 – Heatmap des centres des objets

Étant donné que chaque objet détecté est représenté par un seul point, il est possible de représenter toutes les détections précédentes sous la forme d'une carte de chaleur H^{t-1} à un seul canal, indépendante de la classe : $H^{t-1} = R(\{p_0^{t-1}, p_1^{t-1}, ...\})$ avec p_i^{t-1} la détection à l'instant précédent et R la fonction gaussienne utilisée dans [95] pour entraîner les détecteurs à base de points. La fonction R, à la position $q \in \mathbb{R}^2$ de l'image, est calculée avec l'équation suivante :

$$R_q(\{p_0, p_1, ...\}) = \max_i \ exp(-\frac{(p_i - q)^2}{2\sigma_i^2}) \tag{4.6}$$

avec σ un paramètre basé sur la taille de la boîte englobante de l'objet [95].

Pour mettre en place notre approche, nous utilisons l'architecture YOLOX qui utilise également les points comme ancres pour détecter les objets. L'idée ici est d'aider le réseau à se rappeler des détections précédentes tout en détectant les nouveaux objets aussi. Pour cela, nous proposons de combiner les caractéristiques de l'image courante avec la "heatmap" des détections précédentes afin de mettre en valeur toutes les zones pouvant contenir un objet potentiel. Cela va permettre au réseau d'apprendre à détecter les objets en prenant en compte les détections précédentes. Sur la figure 4.5, nous pouvons observer l'architecture générale de notre approche. Dans cette architecture, nous prenons en entrée deux images : l'image actuelle de la vidéo et l'image de la carte de chaleur contenant les centres des objets détectés précédemment. Ensuite on extrait les caractéristiques des deux images en utilisant le réseau "backbone" CSPDarknet53. Les deux cartes de caractéristiques provenant des deux images sont ensuite fusionnées dans les premières couches du CSPDarknet53, par une méthode de fusion de caractéristiques, avant

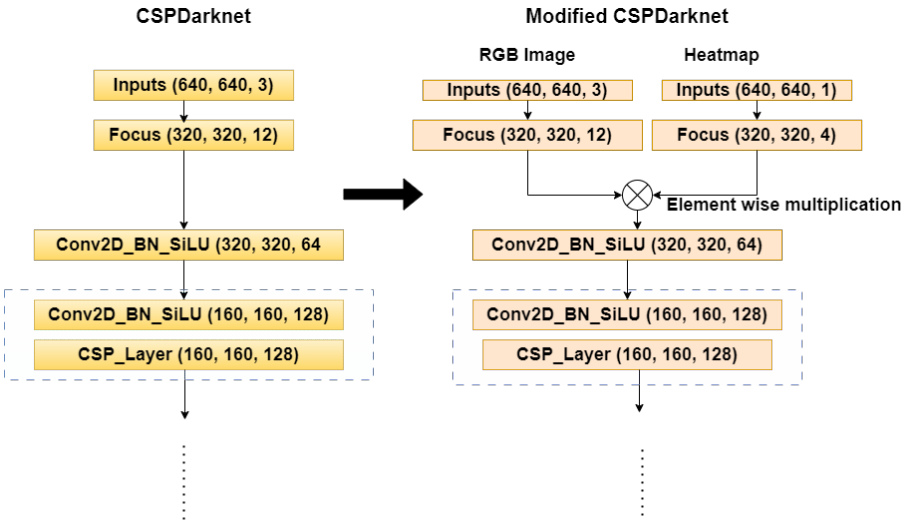


FIGURE 4.5 – Architecture YOLOX modifiée

d'être passées à la suite du réseau YOLOX pour effectuer les prédictions. Il existe plusieurs techniques pour fusionner les caractéristiques provenant de deux sources différentes afin de générer une carte de caratéristiques plus représentative de l'image traitée et améliorer l'efficacité et la précision de la détection. Parmis les techniques de fusion de caractéristiques les plus connues, nous avons :

- **L'addition** : il s'agit d'une opération d'addition élément par élément. C'est une méthode simple mais efficace de fusion des caractéristiques. Cette approche convient lorsqu'on souhaite combiner des caractéristiques provenant de deux branches d'un même réseau ou de sources d'information différentes mais peut entraîner la perte de quelques informations [151], [152]. Par exemple, lorsque nous avons deux cartes de caractéristiques A et B, la fusion par addition est donnée par la formule : $F_p(A, B) = \alpha * A + \beta * B$, où α et β sont des poids qu'on peut apprendre du réseau ou qu'on peut fixer manuellement. Si les poids sont appris par le réseau, ils doivent passer par une fonction sigmoïde ou softmax pour être entre [0, 1].
- **La concaténation** : permet de fusionner plusieurs cartes de caractéristiques le long d'une dimension donnée. La concaténation préserve toutes les informations mais peut augmenter la dimensionnalité des cartes de caractéristiques et consomme plus de mémoire [152]. Cela peut s'avérer utile lorsqu'on souhaite conserver des informations distinctes provenant de différentes sources dans les cartes de caractéristiques. Par exemple une sortie z de fusion des cartes de caractéristiques A et B, va correspondre à $z = [AB]$ avec une dimension $dim(z) = dim(A) + dim(B)$.
- **La multiplication** : il s'agit de la multiplication élément par élément. Cela peut entraîner une certaine perte d'informations originales, mais permet d'obtenir des styles de caractéristiques plus riches. La multiplication élément par élément a pour effet potentiel d'améliorer l'information locale, ce qui pourrait être utile pour détecter les petits objets [152]. La fusion par produit entre les cartes A et B est donnée par : $F_z(A, B) = A \odot B$.

Une approche encore plus sophistiquée consisterait à disposer d'un petit sous-réseau avec deux entrées

et une sortie, afin d'apprendre à combiner au mieux A et B de manière non linéaire. En effet, il n'est pas possible de déterminer quelle opération de fusion est absolument la plus performante dans les différents réseaux existants. Le choix de l'opération dépendra des résultats expérimentaux sur chacune de ces opérations. Afin de trouver la meilleure technique pour fusionner la carte de caractéristiques de l'image et celle de la "heatmap" des détections précédentes, nous avons expérimenté deux méthodes de fusion : l'addition et la multiplication. La figure 4.6 montre l'image des chèvres de Lusignan avec leur "heatmap" générée à partir des centres des boîtes de la vérité terrain.

Nous avons expérimenté également plusieurs distances entre l'image courante et la "heatmap" afin d'évaluer l'impact de la fréquence d'images sur l'architecture modifiée. Pour cela, nous avons entraîné plusieurs modèles en faisant varier les distances d_h ($d_h \in \{1, 5, 10, 15, 20, 25, 30\}$) entre les images. Une distance $d_h = 1$ correspond aux détections qui précèdent directement l'image courante, une distance $d_h = 5$ correspond aux détections de la cinquième image en arrière.

Les performances de cette nouvelle architecture sont testées et évaluées sur la nouvelle base (jeu de données 3).

Pour l'entraînement de la nouvelle architecture, nous avons utilisé les paramètres suivants :

— Taille des entrées : 640x640 pixels

— Taille de batch : 16

— Optimiseur : SGD

— Taux d'apprentissage : entre $[10^{-7}, 10^{-2}]$

— Itération : 150

Nous avons gardé les mêmes couches et les mêmes fonctions de coûts que l'architecture de base de YOLOX. Les poids ont été initialisés à partir des poids du modèle pré-entraîné YOLOX-S avec un apprentissage par transfert. YOLOX a plusieurs sous versions (6 au total) qui varient selon la largeur et la profondeur de l'architecture. Notre architecture se base sur la version YOLOX-S, plus rapide avec moins de paramètres. Il s'agit d'une version moins large et peu profonde (moins de couches) parmi les architectures standards de YOLOX que nous avons voulu tester dans un premier temps. Les autres versions seront également testées par la suite.

4.6 Résultats expérimentaux

Dans cette section, nous allons présenter les résultats obtenus avec l'architecture modifiée de YOLOX et évaluer ces performances par rapport à l'architecture de base. Nous allons d'abord présenter les résultats quantitatifs et ensuite les résultats qualitatifs. Nous avons effectué les entraînements et les tests sur un serveur Ubuntu avec une machine intel(R) Core(TM) i9-9900K CPU 3.60GHz, RAM 64GB avec une carte graphique Nvidia Titan RTX GPU de 24GB de mémoire vidéo.

FIGURE 4.6 – Image avec le heatmap des centres des objets

TABLE 4.4 – Évaluation du mAP@.5 du modèle sur plusieurs distances.

modèle \ d_h	1	5	10	15	20	25	30
modèle_1	92.90%	92.85%	92.35%	92.45%	91.98%	91.94%	92.35%
modèle_5	93.80%	93.45%	93.40%	93%	92.97%	92.90%	93.15%
modèle_10	94.90%	94.80%	94.85%	94.42%	94.39%	94.36%	94.65%
modèle_15	93.60%	93.50%	93.50%	92.44%	92.36%	92.33%	93.65%
modèle_20	93.65%	93.60%	93.55%	92.78%	92.72%	92.68%	93.35%
modèle_25	91.80%	91.75%	91.70%	91.30%	91.24%	91.20%	91.50%
modèle_30	**94.93%**	**94.88%**	**94.38%**	**95.15%**	**95.10%**	**95%**	**95%**

4.6.1 Résultats quantitatifs

Tous les résultats sont obtenus avec CSPDarknet53 comme "backbone". Nous avons évalué les performances de l'architecture modifiée en utilisant les métriques de précision moyenne (AP), la moyenne de la précision (mAP) et également le temps d'inférence sur une image. L'architecture modifiée est entraînée et testée sur plusieurs distances d_h entre l'image courante et les détections précédentes. On peut observer qu'il n'y a pas beaucoup d'écart entre les résultats (tableau 4.4) vu que les images sont assez proches.

TABLE 4.5 – Résultats sur le jeu de données 3 (1777 images de test ; entraînement sur $d_h = 30$ et test sur $d_h = 15$)

Modèle	*mAP@.5*	*chevre_debout AP@.5*	*chevre_couche AP@.5*
YOLOX-S oiginal	81.75%	80.22%	83.28%
YOLOX-S + Heatmap	93.85%	93.69%	94.57%
YOLOX-S * Heatmap	**95.15%**	**94.5%**	**95.8 %**

Le tableau 4.5 nous montre les performances, en terme de précision moyenne, de l'architecture modifiée par rapport à l'architecture originale de YOLOX-S. On peut y observer que la fusion entre la "heatmap"

TABLE 4.6 – Évaluation du mAP@.5 avec et sans suppression des objets du "heatmap"

test \ train	hm	s_hm
hm	95.15%	79.72%
s_hm	85.36%	79.77%

des détections précédentes et l'image courante améliore considérablement les résultats de YOLOX-S. Parmi les deux méthodes de fusion testées, la multiplication est celle qui donne une meilleure performance sur nos données. Ces résultats sont obtenus avec l'architecture entraînée sur la distance $d_h = 30$. Avec cette distance, nous avons plus de variations entre les objets d'une séquence, ce qui améliore les performances du réseau par rapport aux distances plus petites. Il est possible, en phase d'inférence, d'utiliser des distances variées ($d_h < 30$) sans perte importante de précision comme on peut l'observer dans le tabeau 4.4. On peut par exemple accélérer le traitement, lors de l'inférence, en utilisant des fréquences d'images élévées, au lieu de détecter sur chaque image de la séquence.

Nous avons également comparé les performances de l'architecture en l'entraînant avec des "heatmap" où quelques objets sont supprimés. On observe que les performances du réseau, dans ce cas, diminuent. Par contre, lorsqu'on teste l'architecture entraînée avec tous les "heatmap" des objets précédents, sur des "heatmap" où on supprime quelques objets, le réseau arrive quand même à mieux s'en sortir que dans le cas précédent. Le tableau 4.6 nous présente les résultats de cette expérimentation où **s_hm** représente les "heatmap" où plusieurs objets sont supprimés et **hm** représente les "heatmap" sans suppression d'objets. Ces résultats mettent en évidence l'importance des "heatmap" dans la détection. L'intégration des "heatmap" dans l'architecture YOLOX-S rend donc la détection plus stable à travers les séquences d'images.

Le tableau 4.7 nous montre le temps d'inférence pour une image de l'architecture originale de YOLOX-S par rapport à la modification effectuée en intégrant lest "heatmap". On peut observer que le temps d'inférence n'augmente pas beaucoup par rapport à l'architecture originale, ce qui rend toujours possible la détection en temps réel.

TABLE 4.7 – Temps d'inférence

Modèle	*Temps inférence*
YOLOX-S original	7.5 ms
YOLOX-S + Heatmap	7.90 ms
YOLOX-S * Heatmap	7.89 ms

4.6.2 Résultats qualitatifs

Nous avons effectué également des expériences qualitatives sur quelques images successives d'une vidéo, afin d'illustrer les résultats de l'intégration du "heatmap" dans l'architecture de YOLOX-S. Sur la figure 4.7, on peut observer quelques images d'une vidéo de la gauche vers la droite et les résultats de la détection avec respectivement, du haut vers le bas, YOLOX-S original et YOLOX-S * heatmap.

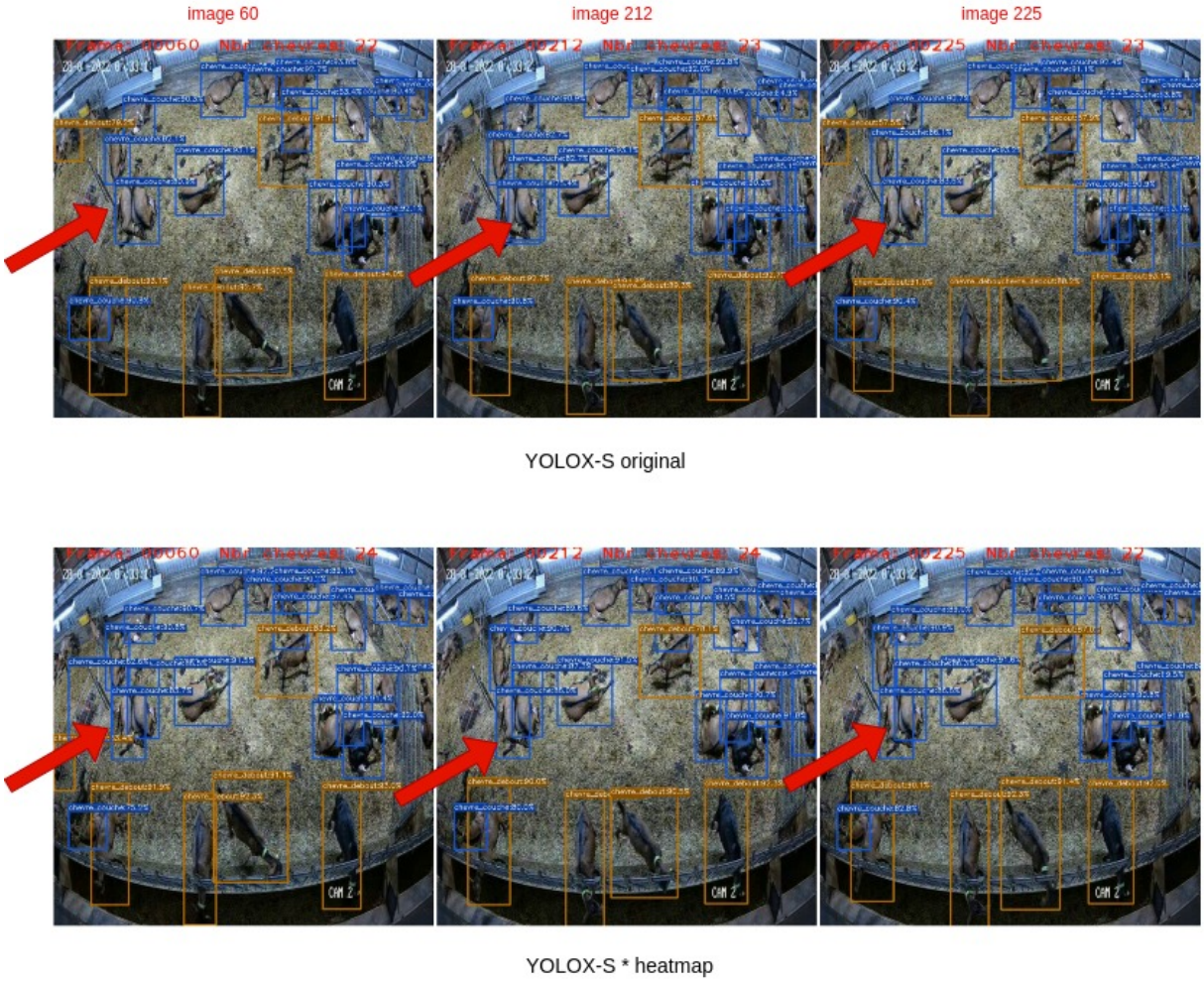


FIGURE 4.7 – Illustration de la détection avec YOLOX sur quelques images d'une vidéo

Sur les résultats de YOLOX-S original, entraîné sur les séquences du jeu de données 3, on peut observer quelques chèvres couchées qui sont mal détectées (indiquées par les flèches rouges). En fusionnant la "heatmap" des détections précédentes avec l'image courante dans l'architecture YOLOX-S, on remarque que ces détections sont maintenues sur les images suivantes.

4.7 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle approche basée sur le détecteur YOLOX dans le but d'améliorer les résultats de détection en réduisant les échecs sur certaines images intermédiaires des vidéos. En effet, les pertes de détection causent des échecs lors du suivi. Une détection plus stable est préférable dans notre système pour un meilleur suivi des animaux et pour détecter des comportements individuels

plus avancés. Ainsi nous avons combiné les détections sur l'image précédente et les caractéristiques de l'image courante afin d'aider le réseau YOLOX à mieux détecter les anciens et les nouveaux objets dans une séquence d'images. Les résultats obtenus avec cette modification sont assez encourageants pour la détection des objets et pour la suite de ce travail.

Chapitre 5

Analyse du comportement chez les chèvres

5.1 Introduction

Dans ce chapitre nous allons présenter l'application du système de détection et de suivi mis en place pour l'analyse du comportement chez les chèvres. L'objectif principal de ce système est d'aider à l'analyse du comportement chez les chèvres et fournir aux éleveurs des indicateurs, qu'ils ne possédaient pas jusqu'à maintenant, pour mieux exploiter leur ferme. Nous allons également présenter les approches utilisées pour identifier quelques comportements recherchés.

5.2 Analyse du comportement des chèvres

L'objectif principal de notre travail est l'analyse automatique du comportement chez les chèvres à travers les vidéos afin de fournir aux éleveurs des indicateurs pour mieux exploiter leur ferme. Pour réaliser cet objetif, nous avons besoin de mettre en place un système de reconnaissance d'actions qui se base sur la détection et le suivi. Dans le chapitre 3, nous avons expérimenté plusieurs méthodes de détection et de suivi afin de mettre en place notre système de reconnaissance d'actions chez les chèvres. A partir de ces expérimentations, la méthode SORT (YOLOv4 combinée avec le filtre de Kalman et l'algorithme Hongrois) est celle retenue, dans un premier temps, pour effectuer la reconnaissance des activités chez les chèvres. Nous avons d'abord défini une liste de comportements recherchés par les éleveurs. Puis nous avons sélectionné les comportements identifiables par notre approche.

5.2.1 Types de comportement

L'établissement des comportements à détecter chez les chèvres a été réalisé conjointement avec le laboratoire INRAe de Bourges et de Tours. Parmi les comportements recherchés nous avons :

— s'alimenter : qui va nous permettre de savoir si l'animal est en train de s'alimenter ou pas. On peut également extraire le nombre total d'animaux en train de s'alimenter sur une période donnée. Cela va fournir des indicateurs sur l'alimentation générale des animaux, qui seront comparés à une norme et créer des alertes lorque les indicateurs ne correspondent pas à la norme ;

— s'abreuver : permet de detecter un animal qui est en train de s'abreuver. On peut aussi extraire des indicateurs sur l'abreuvement au niveau du lot, comme pour l'alimentation, et générer des alertes en cas d'anomalie ;

— ruminer : permet d'identifier si l'animal rumine ou non. Une chèvre qui ne rumine pas durant un certain temps est peut être une chèvre malade ;

— se reproduire : permet de détecter l'accouplement entre un bouc et une chèvre et savoir quelle chèvre a été saillie.

— la posture (debout ou couchée) : permet de savoir si l'animal est debout ou couché. Un animal couché trop longtemps peut être considéré comme un signe d'anomalie. De même une forte densité d'animaux debout pendant la nuit peut être synonyme d'anomalie ou d'un comportement atypique ;

— la mise-bas : permet de détecter la mise-bas d'une chèvre.

Notre système est capable d'analyser actuellement que les comportements sur l'alimentation et l'abreuvement de l'essemble du lot, et également la posture des animaux (debout ou couché).

Une vérité terrain sur ces comportements a été créée par les chercheurs de INRAe en analysant manuellement plusieurs heures de vidéos. Cette vérité terrain est construite à base de 3 vidéos d'élevage de chèvres (2 vidéos de Nohant et une vidéo de Bourges) et contient : un éthogramme, un scan sur l'activité générale du lot et une observation focale sur l'activité individuelle. L'éthogramme contient la liste des comportements à analyser avec les codes correspondants. Dans le scan sur l'activité générale, nous avons le nom des fichiers analysés, l'effectif des animaux effectuant une action donnée (s'alimenter, s'abreuver, debout et couché) et l'heure de l'action. Quant à l'observation focale, elle contient une liste des comportements recherchés, le fichier vidéo dans laquelle le comportement est détecté ainsi que son heure et sa durée. Cette dernière donnée a pour objectif de dresser une liste de comportements dit "atypiques" qui vont permettre de détecter les anomalies au sein du troupeau. Ces informations sont exploitées dans l'analyse de l'activité générale des chèvres par notre système.

Afin de réaliser l'analyse du comportement chez les chèvres en utilisant le système de détection et de suivi mis en place, nous avons dressé une liste de comportements (parmi les types de comportements cités plus haut) que nous pouvons détecter avec notre système. Il s'agit de : s'alimenter, s'abreuver, debout et couché. Ces activités sont détectées pour chaque animal individuellement et regroupées pour effectuer une analyse de l'activité générale du lot. Dans la section suivante, nous allons présenter comment nous avons détecté ces activités.

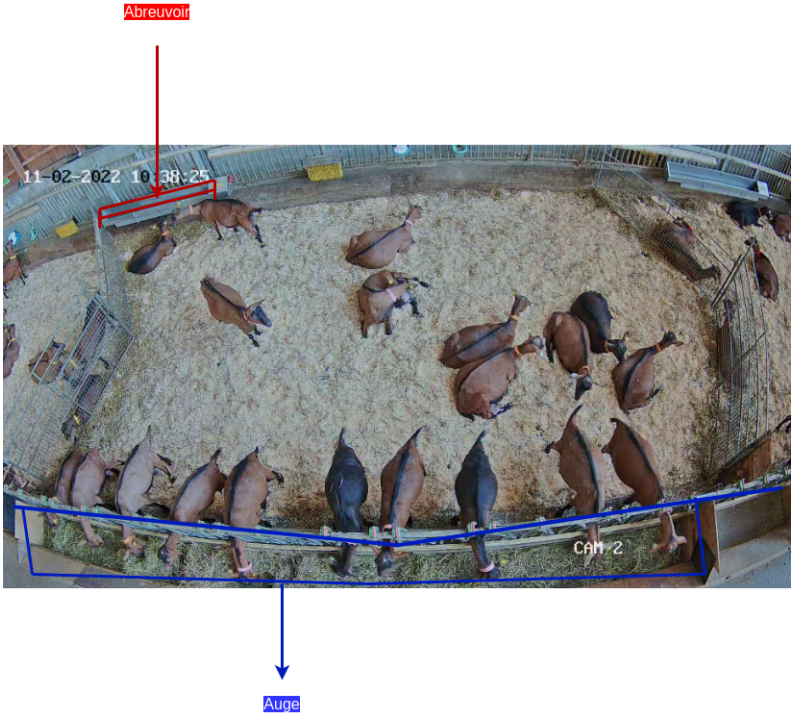


FIGURE 5.1 – Annotation de l'abreuvoir et l'auge

5.2.2 Analyse de l'activité générale du lot

L'analyse de l'activité générale du lot va nous permettre d'avoir des indicateurs sur le comportement générale des animaux présent dans l'enclos, sur une durée donnée. Par exemple nous pouvons avoir le nombre d'animaux qui s'alimentent ou qui s'abreuvent dans la journée en détectant leur présence à l'auge et à l'abreuvoir. L'éleveur peut exploiter ces informations pour détecter un comportement d'alimentation atypique ou anormal chez son troupeau (en comparant par exemple le pourcentage d'animaux qui sont à l'auge en fonction du temps). Il peut également observer, surtout la nuit, le comportement de repos de ses animaux grâce aux effectifs debout et couchés (la nuit par exemple, un effectif trop élevé d'animaux debout peut être source d'anomalie, de stresse ou un comportement atypique). Pour réaliser ce type d'analyse, nous avons effectué la détection sur des vidéos. Pour chaque image de la vidéo, l'activité (debout, couché, à l'auge, à l'abreuvoir) de chaque chèvre est détectée et comptabilisée ensuite pour avoir l'effectif total de chaque activité par image. La détection des activités pour chaque animal est réalisée comme suit :

— **debout et couché** : ces activités sont détectées directement au niveau de la détection des chèvres. En effet, la différence entre les postures (debout et couchée) des chèvres dans les vidéos est assez visible (figure 3.17). Ce qui nous a donné l'idée de créer 2 classes, lors de la phase de détection des chèvres, qui représentent en même temps la posture de la chèvre détectée. Il s'agit d'une classification inter-classe qui nous permet de classer en même temps la posture de la chèvre ;

— **s'alimenter et s'abreuver** : ici, nous avons annoté, pour chaque caméra, les positions des mangoires (auge) et abreuvoirs (figure 5.1). Ainsi, pour détecter si l'animal s'alimente ou s'abreuve, nous

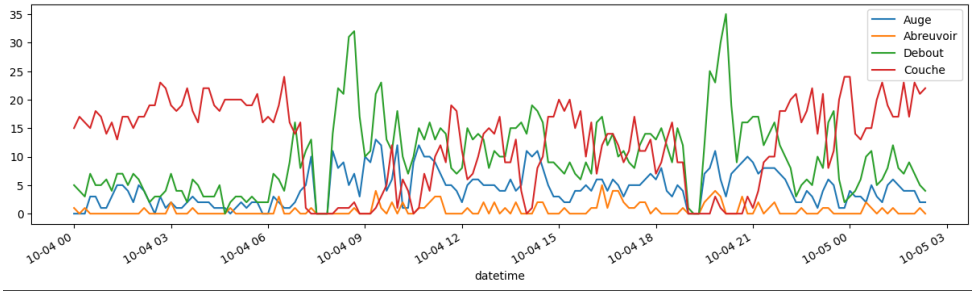


FIGURE 5.2 – Analyse de l'activité générale du lot.

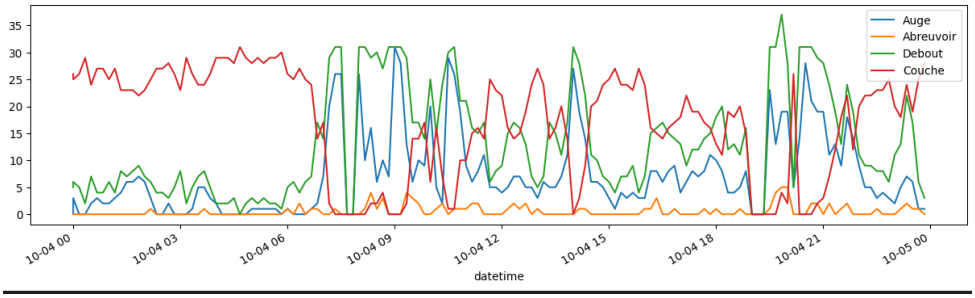


FIGURE 5.3 – Annotation manuelle de l'activité générale du lot.

cherchons s'il est présent dans l'une des zones d'intérêts (l'auge et l'abreuvoir). Pour détecter sa présence, nous comparons les coordonnées de la boîte de détection qui la délimite aux positions des zones d'intérêt et lorsque ces points se chevauchent, ont considère que l'animal est présent dans la zone correspondante.

Pour tester notre système de suivi, nous avons effectué l'analyse de l'activité générale, sur une journée (du 04/10/2020 à 00h00 au 05/10/2020 à 03h), d'une vidéo d'un lot de 35 chèvres de Nohant. Sur la figure 5.2, nous pouvons observer le graphe, produit par notre système, qui résume les activités des chèvres (à l'auge, à l'abreuvoir, debout et couché) sur une journée. Nous avons également l'annotation mannuelle correspondante à cette activité générale du lot de la même vidéo (figure 5.3), mais effectuée par les chercheurs en se basant sur les données du scan de l'activité générale de la vérité terrain. Le traitement, par notre système, est réalisé toutes les 10 images de la vidéo et enrégistré dans un fichier csv. Le fichier csv est ensuite analysé avec la librairie pandas pour générer les graphes des figures 5.2 et 5.4 montrant l'effectif des chèvres effectuant une activité donnée par rapport au temps. Sur la figure 5.4, qui correspond à l'analyse détaillée généreé par notre système, on peut observer pour chaque heure, l'effectif des animaux réalisant une activité donnée. Toujours sur le même graphe, on peut voir que de 00h jusqu'à environ 07h l'effectif d'animaux couchés est élevé et la variabilité est faible. Ensuite, à partir de 07h10, on observe une augmentation de l'effectif à l'auge et des chèvres debout ce qui s'explique par la venue de l'éleveur, qui rapproche les fourrages dispersés à l'auge, qu'on observe sur la védéo traitée. À partir de 07h30 jusqu'à 08h, on observe une baisse des effectifs des animaux qui s'explique par le départ des animaux à la traite. Au retour de la traite vers 08h, on assiste à un regain de l'effectif à l'auge jusqu'à environ 09h30. Ensuite l'effectif des animaux couché redevient important, ce qui peut s'expliquer par le

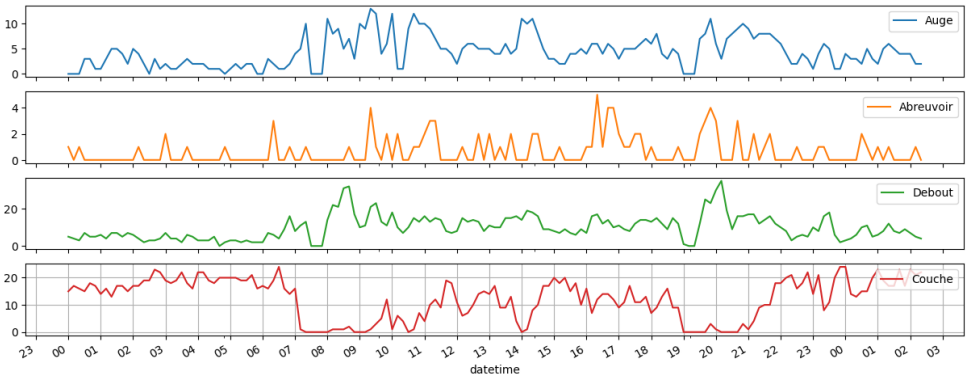


FIGURE 5.4 – Analyse détaillée de l'activité du lot.

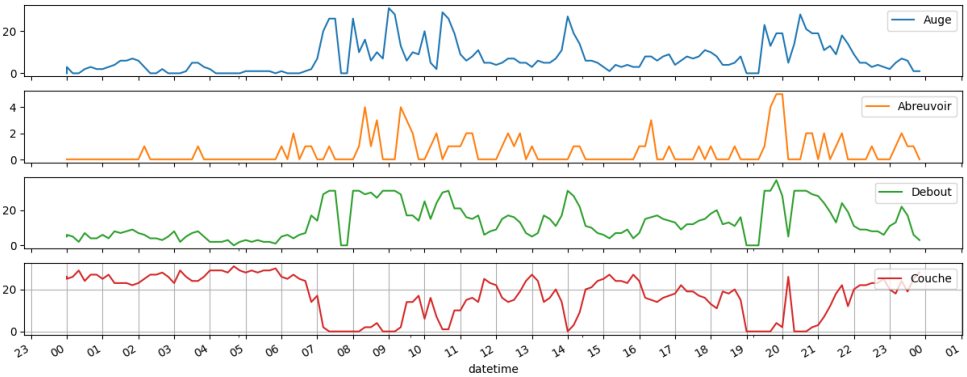


FIGURE 5.5 – Annotation manuelle détaillée de l'activité du lot.

manque de fourrage à l'auge jusqu'à 10h. A partir de 10h, on observe un regain de l'effectif debout, puis à l'auge qui correspond au moment de la distribution de fourrage. De 11h jusqu'à 14h, la plupart des chèvres sont couchées et ruminent généralement. On peut voir que l'activité des animaux est stable avec quelques variations entre debout et couché, et pour l'effectif à l'auge. De 19h à 19h22 on remarque encore une baisse des effectifs des animaux qui sont à la traite en ce moment là. Ensuite, vers 22h, l'effectif des chèvres couchées recommence par augmenter avec une baisse des effectifs debout et à l'auge. En comparant Ces résultats de notre système avec l'annotation manuelle de la figure 5.5 (vérité terrain), on observe qu'ils sont assez proches et reflètent la réalité. Cependant, nous avons également quelques échecs de détections qui se traduisent par l'écart entre l'effectif totale d'une activité dans notre système par rapport à l'effectif réel annoté correspondant. C'est pour corriger cela que nous proposons l'approche dans le chapitre 4.

5.3 Conclusion

Dans ce chapitre, nous avons présenté l'analyse de l'activité générale des chèvres en utilisant le système de détection et de suivi mis en place. Nous avons comparé les résultats de cette analyse aux résultats de l'analyse effectuée manuellement par les chercheurs de l'INRAe (vérité terrain). De façon générale, on arrive à fournir des indicateurs, assez proche de la réalité (vérité terrain), sur l'activité générale du lot que

l'éleveur peut utiliser pour suivre le comportement de ces animaux, de jour comme de nuit. Cependant, quelques améliorations peuvent être encore effectuées en intégrant par exemple l'architecture modifiée de YOLOX (dans le chapitre 4) pour améliorer la détection et également le suivi.

Chapitre 6

Conclusion

6.1 Synthèse de mes contributions

Nos travaux de recherches présentés dans ce manuscrit, portent sur l'analyse automatique du comportement chez les chèvres à base de capteurs vidéos. Afin de réaliser ce travail, nous avons appliqué les techniques de reconnaissances d'actions qui consistent à détecter et à suivre les objets dans les vidéos et faire une analyse de leur activité. Les données vidéos traitées ici ont des propriétés différentes de celles souvent utilisées dans d'autres cas d'études comme la reconnaissance d'actions chez l'homme. Pour cela, nous avons construit manuellement une nouvelle base d'images pour la détection des chèvres dans un environnement d'élevage clos. Afin d'extraire des données assez représentatives des conditions réelles d'élevages des chèvres, notre base d'images est composée de plusieurs vidéos, de jour comme de nuit, qui ont été capturées dans 3 fermes différentes (Nohant, Avord et Lusignan) avec différents angles de vue pour avoir différentes postures des animaux. En parallele à la construction de la base d'images, nous avons fait une étude bibliographique sur les méthodes existantes pour le suivi automatique des animaux en situation d'élevages dans des enclos à base de vidéos. Dans la plupart de ces travaux, le nombre d'animaux etudié est compris entre 3 et 10, ce qui est très inférieur à notre cas où nous avons entre 20 et 60 chèvres par enclos. Ainsi, la complexité de détection et les occultations sont plus élevées dans notre cas. Nous avons aussi fait l'étude bibliographique des méthodes de détection et de suivi souvent utilisées dans l'analyse automatique du comportement ou la reconnaissance d'actions. Pour la détection, nous avons testé et comparé les méthodes les plus populaires de la littérature sur nos données. Nous avons fait de même pour les méthodes de suivi. Ces expérimentations nous ont permis d'évaluer l'efficacité et aussi de relever les points faibles des méthodes existantes et de savoir lesquelles sont les plus adaptées à notre problématique. Ainsi, pour effectuer l'analyse automatique des comportement chez les chèvres, nous avons proposé, dans un premier temps, une approche basée sur la méthode de détection et de suivi dénommée SORT. Nous avons effectué l'analyse de quelques comportements avec cette approche et fournit des indicateurs sur l'activité générale du troupeau que l'éleveur pourra exploiter dans la gestion de ses animaux. Les tests du système d'analyse automatique nous ont permis de relever quelques points faibles dans notre approche

qui sont principalment les pertes de détection sur des images intermédiaires de la vidéo et également les échecs ou changements d'identifiant dans le suivi. Pour améliorer ces points faibles, nous avons proposé une seconde approche permettant d'augmenter la précision de détection et ainsi réduire les pertes sur les images intermédiaires des vidéos. Cette approche pourra être utilisée plus tard pour améliorer le suivi et avoir un système de bout en bout.

Pour la première approche, la détection des chèvres est effectuée dans un premier temps par la méthode YOLOv4 sur chaque image de la vidéo, ensuite chaque chèvre détectée est suivi par un identifiant unique en utilisant le filtre de Kalman et l'algorithme Hongrois. Le filtre de Kalman est utilisé pour estimer la position de la chèvre dans l'image suivante en se basant sur les informations de la détection par YOLOv4 et le mouvement de l'objet dans l'image courante. L'algorithme Hongrois permet de coupler les prédictions du filtre de Kalman avec les prédictions de YOLOv4 afin de leur attribuer le même identifiant et effectuer le suivi.

Pour la deuxième approche, nous avons utilisé une nouvelle architecture de détection d'objets dénommée YOLOX, qui est une version améliorée des versions précédentes de YOLO. Nous avons modifié cette architecture afin de pouvoir intégrer les détections précédentes dans le réseau sous forme de carte de chaleur ("heatmap"). Cela permet au réseau de se rappeler des détections dans l'image précédente et d'éviter leur perte. Pour cela, nous avons testé et comparé deux méthodes de fusion de caractéristiques pour agréger les informations des détections précédentes avec celles de l'image courante. Dans cette approche, l'entraînement est effectué sur une nouvelle base d'images constituée de paires d'images consécutives de vidéos afin d'avoir les détections des images précédentes. Nous avons comparé les résultats de cette nouvelle architecture à l'architecture originale de YOLOX sur la nouvelle base d'images.

6.2 Perspectives

Dans ce travail, nous avons réalisé une partie des objectifs sur l'analyse du comportement chez les chèvres pour le projet ANIMOV. Avec la deuxième approche proposée dans ce travail, de nombreuses perspectives s'ouvrent pour les futurs travaux à venir afin de finaliser les objectifs de ce projet. Parmi ces perpectives nous avons :

— Amélioration du jeu de données 3 sur la détection : en effet l'entraînement d'un modèle de détection, sans boîtes d'ancrage prédéfinies, nécessite beaucoup de données. Une amélioration de notre base avec plus de vidéos variées, avec peu d'images similaires, pourraient améliorer les résultats de notre approche ;

— Application de la suppression non-maximale séquentielle : dans notre approche pour l'amélioration de la détection dans les vidéos, nous utilisons toujours la suppression non-maximal simple pour filtrer les détections lors de l'inférence. On peut tester également la suppression non-maximal séquentielle pour voir si cela améliore la précision de la détection ;

— Tester notre approche sur des bases publiques : les données utilisées pour tester et évaluer les per-

formances de l'architecture modifiée de YOLOX-S proviennent d'une base que nous avons construit manuellement dans le cadre du projet ANIMOV. Il nous semble interressant de tester cette approche sur des données publiques de détection dans les vidéos et comparer les résultats aux méthodes existantes sur ce sujet. Ces données sont en général très volumineuses et peu accessibles, ce qui ne nous a pas permis, à ce jour, de réaliser ces tests ;

— Intégrer la nouvelle approche dans la méthode de suivi d'objets SORT : remplacer la méthode YOLOv4 utilisée dans le chapitre 3 pour le suivi par la nouvelle approche basée sur YOLOX-S et comparer les nouveaux résultats de suivi par rapport aux anciens.

— Le suivi de bout en bout par l'architecture YOLOX : proposée une architecture à base de YOLOX afin d'effectuer la détection et le suivi dans une seule architecture. En effet, le fonctionnement de l'architecture YOLOX peut permettre de propager les détections précédentes vers leur nouvelle position dans l'image courante en se basant sur la propagation de "heatmap" dans la régression ;

— La détection d'autres comportements : effectuer la détection d'autres comportements (mettre-bas, se reproduire, ruminer, etc.) qui n'ont pas été traités dans ce document, en utilisant par exemple une approche à base de l'estimation de pose ;

— Faire des tests dans des environnements extérieurs : notre travail est centré sur des environnements clos d'élevage. Il pourrait être intéressant d'évaluer l'efficacité de notre approche dans un autre environnement.

Bibliographie

[1] Abozar Nasirahmadi, Barbara Sturm, Sandra Edwards, Knut-Håkan Jeppsson, Anne-Charlotte Olsson, Simone Müller, and Oliver Hensel. Deep learning and machine vision approaches for posture detection of individual pigs. *Sensors*, 19(17), 2019.

[2] Mohammadamin Kashiha, Claudia Bahr, Sanne Ott, Christel P.H. Moons, Theo A. Niewold, F.O. Ödberg, and Daniel Berckmans. Automatic identification of marked pigs in a pen using image pattern recognition. *Computers and Electronics in Agriculture*, 93 :111–120, 2013.

[3] Abozar Nasirahmadi, Uwe Richter, Oliver Hensel, Sandra Edwards, and Barbara Sturm. Using machine vision for investigation of changes in pig group lying patterns. *Computers and Electronics in Agriculture*, 119 :184–190, 2015.

[4] Peter Ahrendt, Torben Gregersen, and Henrik Karstoft. Development of a real-time computer vision system for tracking loose-housed pigs. *Computers and Electronics in Agriculture*, 76(2) :169–174, 2011.

[5] M. Nilsson, A. H. Herlin, H. Ardö, O. Guzhva, K. Åström, and C. Bergsten. Development of automatic surveillance of animal behaviour and welfare using image analysis and machine learned segmentation technique. *animal*, 9(11) :1859–1865, 2015.

[6] Lei Zhang, Helen Gray, Xujiong Ye, Lisa Collins, and Nigel Allinson. Automatic individual pig detection and tracking in pig farms. *Sensors*, 19(5), 2019.

[7] Yuan Rao, Min Jiang, Wen Wang, Wu Zhang, and Ruchuan Wang. On-farm welfare monitoring system for goats based on internet of things and machine learning. *International Journal of Distributed Sensor Networks*, 16(7) :1550147720944030, 2020.

[8] Patrizia Tassinari, Marco Bovo, Stefano Benni, Simone Franzoni, Matteo Poggi, Ludovica Maria Eugenia Mammi, Stefano Mattoccia, Luigi Di Stefano, Filippo Bonora, Alberto Barbaresi, Enrica Santolini, and Daniele Torreggiani. A computer vision approach based on deep learning for the detection of dairy cows in free stall barn. *Computers and Electronics in Agriculture*, 182 :106030, 2021.

[9] Mateusz Mittek, Eric Psota, Lance C. Pérez, and Benny E Mote. Health monitoring of group-housed pigs using depth-enabled multi-object tracking. 2016.

[10] Haar Cascade Classifiers. https://medium.datadriveninvestor.com/haar-cascade-classifiers-237c9193746b. Accessed : 2023-08-22.

[11] Massimo Bertozzi, Alberto Broggi, Michael S. Del Rose, Mirko Felisa, Alain Rakotomamonjy, and Frédéric Suard. A pedestrian detector using histograms of oriented gradients and a support vector machine classifier. *2007 IEEE Intelligent Transportation Systems Conference*, pages 143–148, 2007.

[12] Bing Wang, Kap Chan, Gang Wang, and Haijian Zhang. Pedestrian detection in highly crowded scenes using "online" dictionary learning for occlusion handling. 10 2014.

[13] Binary Image classifier CNN using TensorFlow. https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697. Accessed : 2023-08-28.

[14] Convolutional Neural Networks, Explained. https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939. Accessed : 2023-08-28.

[15] An Intuitive Explanation of Convolutional Neural Networks. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets. Accessed : 2023-08-28.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.

[17] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN : towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[18] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX : exceeding YOLO series in 2021. *CoRR*, abs/2107.08430, 2021.

[19] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years : A survey. *CoRR*, abs/1905.05055, 2019.

[20] COCO, Common Objects in Context. https://cocodataset.org/#detection-eval. Accessed : 2023-08-25.

[21] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking : A survey. *ACM Comput. Surv.*, 38(4) :13–es, dec 2006.

[22] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. *CoRR*, abs/1710.03958, 2017.

[23] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. *CoRR*, abs/2004.01177, 2020.

[24] Jinlong Peng, Changan Wang, Fangbin Wan, Yang Wu, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yanwei Fu. Chained-tracker : Chaining paired attentive regression results for end-to-end joint multiple-object detection and tracking. *CoRR*, abs/2007.14557, 2020.

[25] Peize Sun, Yi Jiang, Rufeng Zhang, Enze Xie, Jinkun Cao, Xinting Hu, Tao Kong, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack : Multiple-object tracking with transformer. *CoRR*, abs/2012.15460, 2020.

[26] Understanding and Implementing Faster R-CNN. https://towardsdatascience.com/understanding-and-implementing-faster-r-cnn-a-step-by-step-guide-11acfff216b0. Accessed : 2023-09-21.

[27] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4 : Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.

[28] Yijing Guo, Yixin Zeng, Fengqiang Gao, Yi Qiu, Xuqiang Zhou, Linwei Zhong, and Choujun Zhan. Improved yolov4-csp algorithm for detection of bamboo surface sliver defects with extreme aspect ratio. *IEEE Access*, 10 :1–1, 01 2022.

[29] C_5.02 PANet (Path Aggregation Network) - BoS - EN. https://wikidocs.net/167842. Accessed : 2023-08-30.

[30] The beginner's guide to implementing YOLOv3 in TensorFlow 2.0 (part-1). https://machinelearningspace.com/yolov3-tensorflow-2-part-1/#nms-unique. Accessed : 2023-10-10.

[31] Joseph Redmon and Ali Farhadi. Yolov3 : An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[32] YOLOX Explanation — How Does YOLOX Work. https://medium.com/mlearning-ai/yolox-explanation-how-does-yolox-work-3e5c89f2bf78. Accessed : 2023-09-27.

[33] Prithwish Jana and Partha Pratim Mohanta. Recent trends in 2d object detection and applications in video event recognition. *CoRR*, abs/2202.03206, 2022.

[34] Kalman Filter Explained Simply. https://thekalmanfilter.com/kalman-filter-explained-simply/. Accessed : 2023-08-30.

[35] Nicolai Wojke and Alex Bewley. Deep cosine metric learning for person re-identification. *CoRR*, abs/1812.00442, 2018.

[36] Qiongfeng Shi, Bowei Dong, Tianyiyi He, Zhongda Sun, Jianxiong Zhu, Zixuan Zhang, and Chengkuo Lee. Progress in wearable electronics/photonics—moving toward the era of artificial intelligence and internet of things. *InfoMat*, 2(6) :1131–1162, 2020.

[37] Jinyang Jiao, Ming Zhao, Jing Lin, and Kaixuan Liang. A comprehensive review on convolutional neural network in machine fault diagnosis. *Neurocomputing*, 417 :36–63, 2020.

[38] Dario Augusto Borges Oliveira, Luiz Gustavo Ribeiro Pereira, Tiago Bresolin, Rafael Ehrich Pontes Ferreira, and Joao Ricardo Reboucas Dorea. A review of deep learning algorithms for computer vision systems in livestock. *Livestock Science*, 253 :104700, 2021.

[39] Md Sultan Mahmud, Azlan Zahid, Anup Kumar Das, Muhammad Muzammil, and Muhammad Usman Khan. A systematic literature review on deep learning applications for precision cattle farming. *Computers and Electronics in Agriculture*, 187 :106313, 2021.

[40] Matthew Tscharke and Thomas M Banhazi. A brief review of the application of machine vision in livestock behaviour analysis. *Agrárinformatika/Journal of Agricultural Informatics*, 7(1) :23–42, 2016.

[41] Priscilla Cristina Cabral Ribeiro, Annibal Jose Scavarda, and Mario Otavio Batalha. Rfid in the international cattle supply chain : context, consumer privacy and legislation. *International Journal of Services and Operations Management*, 6(2) :149, 2010.

[42] Vitor M. T. Aleluia, Vasco N. G. J. Soares, João M. L. P. Caldeira, and António M. Rodrigues. Livestock monitoring : Approaches, challenges and opportunities. *International Journal of Engineering and Advanced Technology*, 11(4) :67-76, apr 2022.

[43] Chen Yujuan, He Dongjian, Fu Yinxi, and Song Huaibo. Intelligent monitoring method of cow ruminant behavior based on video analysis technology. *International Journal of Agricultural and Biological Engineering*, 10 :194–202, 2017.

[44] D. Comaniciu and P. Meer. Mean shift : a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5) :603–619, 2002.

[45] Dongju Liu and Jian Yu. Otsu method and k-means. In *2009 Ninth International Conference on Hybrid Intelligent Systems*, volume 1, pages 344–349, 2009.

[46] Amine Bohi. *Descripteurs de Fourier inspirés de la structure du cortex visuel primaire humain : Application à la reconnaissance de navires dans le cadre de la surveillance maritime.* Theses, Université de Toulon, May 2017.

[47] K. Kollis, C.S. Phang, Thomas Banhazi, and S.J. Searle. Weight estimation using image analysis and statistical modelling : A preliminary study. *Applied Engineering in Agriculture*, 23 :91–96, 01 2007.

[48] Hai Ho Dac, Claudia Gonzalez Viejo, Nir Lipovetzky, Eden Tongson, Frank R. Dunshea, and Sigfredo Fuentes. Livestock identification using deep learning for traceability. *Sensors*, 22(21), 2022.

[49] Alvaro Fuentes, Shujie Han, Muhammad Fahad Nasir, Jongbin Park, Sook Yoon, and Dong Sun Park. Multiview monitoring of individual cattle behavior based on action recognition in closed barns using deep learning. *Animals*, 13(12), 2023.

[50] Glenn Jocher, Ayush Chaurasia, Alex Stoken, Jirka Borovec, NanoCode012, Yonghye Kwon, TaoXie, Kalen Michael, Jiacong Fang, imyhxy, Lorna, Colin Wong, (Zeng Yifu), Abhiram V, Diego Montes, Zhiqiang Wang, Cristi Fati, Jebastin Nadar, Laughing, UnglvKitDe, tkianai, yxNONG, Piotr Skalski, Adam Hogan, Max Strobel, Mrinal Jain, Lorenzo Mammana, and xylieong. ultralytics/yolov5 : v6.2 - YOLOv5 Classification Models, Apple M1, Reproducibility, ClearML and Deci.ai integrations, August 2022.

[51] Yangyang Guo, Ziru Zhang, Dongjian He, Jinyu Niu, and Yi Tan. Detection of cow mounting behavior using region geometry and optical flow characteristics. *Computers and Electronics in Agriculture*, 163 :104828, 2019.

[52] Dihua Wu, Mengxuan Han, Huaibo Song, Lei Song, and Yuanchao Duan. Monitoring the respiratory behavior of multiple cows based on computer vision and deep learning. *Journal of Dairy Science*, 106(4) :2963–2979, 2023.

[53] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact : Real-time instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[54] Souhaieb Aouayeb, Xavier Desquesnes, Bruno Emile, Baptiste Mulot, and Sylvie Treuillet. Intelligent video surveillance fornbsp ;animal behavior monitoring. In *Image Analysis and Processing. ICIAP 2022 Workshops : ICIAP International Workshops, Lecce, Italy, May 23–27, 2022, Revised Selected Papers, Part II*, page 361–371, Berlin, Heidelberg, 2022. Springer-Verlag.

[55] Su Myat Noe, Thi Thi Zin, Pyke Tin, and Ikuo Kobayashi. Comparing state-of-the-art deep learning algorithms for the automated detection and tracking of black cattle. *Sensors*, 23(1), 2023.

[56] Victor A. Kulikov, Nikita V. Khotskin, Sergey V. Nikitin, Vasily S. Lankin, Alexander V. Kulikov, and Oleg V. Trapezov. Application of 3-d imaging sensor for tracking minipigs in the open field test. *Journal of Neuroscience Methods*, 235 :219–225, 2014.

[57] Dario Augusto Borges Oliveira, Luiz Gustavo Ribeiro Pereira, Tiago Bresolin, Rafael Ehrich Pontes Ferreira, and Joao Ricardo Reboucas Dorea. A review of deep learning algorithms for computer vision systems in livestock. *Livestock Science*, 253 :104700, 2021.

[58] Zhangqiang Ming, Min Zhu, Xiangkun Wang, Jiamin Zhu, Junlong Cheng, Chengrui Gao, Yong Yang, and Xiaoyong Wei. Deep learning-based person re-identification methods : A survey and outlook of recent works. *Image and Vision Computing*, 119 :104394, 2022.

[59] Object detection. https://en.wikipedia.org/wiki/Object_detection. Accessed : 2023-08-19.

[60] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[61] J. Whitehill and C.W. Omlin. Haar features for facs au recognition. In *7th International Conference on Automatic Face and Gesture Recognition (FGR06)*, pages 5 pp.–101, 2006.

[62] Robert E. Schapire. *Explaining AdaBoost*, pages 37–52. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[63] 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)*, volume 1, pages 886–893 vol. 1, 2005.

[64] E.N. Mortensen, Hongli Deng, and L. Shapiro. A sift descriptor with global context. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 184–190 vol. 1, 2005.

[65] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer. Knn model-based approach in classification. In Robert Meersman, Zahir Tari, and Douglas C. Schmidt, editors, *On The Move to Meaningful Internet Systems 2003 : CoopIS, DOA, and ODBASE*, pages 986–996, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[66] Allam Shehata Hassanein, Sherien Mohammad, Mohamed Sameer, and Mohammad Ehab Ragab. A survey on hough transform, theory, techniques and applications. *CoRR*, abs/1502.02160, 2015.

[67] Yong Shi, Zhao Lv, Ning Bi, and Chao Zhang. An improved sift algorithm for robust emotion recognition under various face poses and illuminations. *Neural Computing and Applications*, 32(13) :9267–9281, Aug 2019.

[68] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. Edinburgh research explorer the pascal visual object classes (voc) challenge.

[69] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[70] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *CoRR*, abs/1511.08458, 2015.

[71] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[72] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[73] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[74] Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, 2015.

[75] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[76] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets : Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[77] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[78] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2) :154–171, 2013.

[79] M.A. Hearst, S.T. Dumais, E. Osuna, J. Platt, and B. Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4) :18–28, 1998.

[80] Mengyu Huang. Theory and implementation of linear regression. In *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)*, pages 210–217, 2020.

[81] Mohammad Amin Sadeghi and David Forsyth. 30hz object detection with dpm v5. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 65–79, Cham, 2014. Springer International Publishing.

[82] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[83] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901, 2013.

[84] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN : object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016.

[85] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Light-head R-CNN : in defense of two-stage object detector. *CoRR*, abs/1711.07264, 2017.

[86] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[87] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once : Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[88] Xin Jin and Jiawei Han. *K-Means Clustering*, pages 563–564. Springer US, Boston, MA, 2010.

[89] Joseph Redmon and Ali Farhadi. YOLO9000 : better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[90] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-yolov4 : Scaling cross stage partial network. *CoRR*, abs/2011.08036, 2020.

[91] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7 : Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *ArXiv*, abs/2207.02696, 2022.

[92] Ultralytics YOLOv5. https://docs.ultralytics.com/. Accessed : 2023-08-13.

[93] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD : single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[94] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[95] Hei Law and Jia Deng. Cornernet : Detecting objects as paired keypoints. *CoRR*, abs/1808.01244, 2018.

[96] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet : Keypoint triplets for object detection. *CoRR*, abs/1904.08189, 2019.

[97] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *CoRR*, abs/2005.12872, 2020.

[98] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR : deformable transformers for end-to-end object detection. *CoRR*, abs/2010.04159, 2020.

[99] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO : common objects in context. *CoRR*, abs/1405.0312, 2014.

[100] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper R. R. Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, and Vittorio Ferrari. The open images dataset V4 : unified image classification, object detection, and visual relationship detection at scale. *CoRR*, abs/1811.00982, 2018.

[101] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242, 2020.

[102] Ido Leichter, Michael Lindenbaum, and Ehud Rivlin. Mean shift tracking with multiple reference color histograms. *Computer Vision and Image Understanding*, 114(3) :400–408, 2010.

[103] John G Allen, Richard YD Xu, Jesse S Jin, et al. Object tracking using camshift algorithm and multiple quantized feature spaces. In *ACM international conference proceeding series*, volume 100, pages 3–7. Citeseer, 2004.

[104] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D) :35–45, 1960.

[105] Xi Chen, Xiao Wang, and Jianhua Xuan. Tracking multiple moving objects using unscented kalman filtering techniques. *CoRR*, abs/1802.01235, 2018.

[106] Hitesh A. Patel and Darshak G. Thakore. Moving object tracking using kalman filter. 2013.

[107] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. *CoRR*, abs/1602.00763, 2016.

[108] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. *CoRR*, abs/1703.07402, 2017.

[109] Maria Isabel Ribeiro. Kalman and extended kalman filters : Concept, derivation and properties. *Institute for Systems and Robotics*, 43(46) :3736–3741, 2004.

[110] M. Jaward, L. Mihaylova, N. Canagarajah, and D. Bull. Multiple object tracking using particle filters. In *2006 IEEE Aerospace Conference*, pages 8 pp.–, 2006.

[111] Fasheng Wang. Particle filters for visual tracking. In Gang Shen and Xiong Huang, editors, *Advanced Research on Computer Science and Information Engineering*, pages 107–112, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[112] Shide Du and Shiping Wang. An overview of correlation-filter-based object tracking. *IEEE Transactions on Computational Social Systems*, 9(1) :18–31, 2022.

[113] David S. Bolme, J. Ross Beveridge, Bruce A. Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550, 2010.

[114] Alan Lukezic, Tomás Vojír, Luka Cehovin, Jiri Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. *CoRR*, abs/1611.08461, 2016.

[115] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3) :583–596, 2014.

[116] Tianzhu Zhang, Changsheng Xu, and Ming-Hsuan Yang. Multi-task correlation particle filter for robust object tracking. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4819–4827, 2017.

[117] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52, 1955.

[118] Jinkun Cao, Xinshuo Weng, Rawal Khirodkar, Jiangmiao Pang, and Kris Kitani. Observation-centric sort : Rethinking sort for robust multi-object tracking. *ArXiv*, abs/2203.14360, 2022.

[119] Yunhao Du, Yang Song, Bo Yang, and Yanyun Zhao. Strongsort : Make deepsort great again. *ArXiv*, abs/2202.13514, 2022.

[120] Yongyi Lu, Cewu Lu, and Chi-Keung Tang. Online video object detection using association lstm. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2363–2371, 2017.

[121] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8) :1735–1780, 11 1997.

[122] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. A simple baseline for multi-object tracking. *CoRR*, abs/2004.01888, 2020.

[123] Aljosa Osep, Wolfgang Mehner, Paul Voigtlaender, and Bastian Leibe. Track, then decide : Category-agnostic vision-based multi-object tracking. *CoRR*, abs/1712.07920, 2017.

[124] Matthieu Paul, Martin Danelljan, Christoph Mayer, and Luc Van Gool. Robust visual tracking by segmentation. *ArXiv*, abs/2203.11191, 2022.

[125] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixé. Tracking without bells and whistles. *CoRR*, abs/1903.05625, 2019.

[126] Qi Chu, Wanli Ouyang, Hongsheng Li, Xiaogang Wang, Bin Liu, and Nenghai Yu. Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism. *CoRR*, abs/1708.02843, 2017.

[127] Ji Zhu, Hua Yang, Nian Liu, Minyoung Kim, Wenjun Zhang, and Ming-Hsuan Yang. Online multi-object tracking with dual matching attention networks. *CoRR*, abs/1902.00749, 2019.

[128] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixé, and Christoph Feichtenhofer. Trackformer : Multi-object tracking with transformers. *CoRR*, abs/2101.02702, 2021.

[129] Fangao Zeng, Bin Dong, Tiancai Wang, Cheng Chen, Xiangyu Zhang, and Yichen Wei. MOTR : end-to-end multiple-object tracking with transformer. *CoRR*, abs/2105.03247, 2021.

[130] Patrick Dendorfer, Aljosa Osep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian D. Reid, Stefan Roth, and Laura Leal-Taixé. Motchallenge : A benchmark for single-camera multiple target tracking. *CoRR*, abs/2010.07548, 2020.

[131] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack : Multi-object tracking by associating every detection box. *CoRR*, abs/2110.06864, 2021.

[132] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup : Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017.

[133] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix : Regularization strategy to train strong classifiers with localizable features. *CoRR*, abs/1905.04899, 2019.

[134] Terrance Devries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017.

[135] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. Dropblock : A regularization method for convolutional networks. *CoRR*, abs/1810.12890, 2018.

[136] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet : Ultra-deep neural networks without residuals. *CoRR*, abs/1605.07648, 2016.

[137] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss : Faster and better learning for bounding box regression. In *AAAI Conference on Artificial Intelligence*, 2019.

[138] Diganta Misra. Mish : A self regularized non-monotonic activation function. In *British Machine Vision Conference*, 2020.

[139] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. CBAM : convolutional block attention module. *CoRR*, abs/1807.06521, 2018.

[140] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8759–8768, 2018.

[141] G. Mclachlan. Mahalanobis distance. *Resonance*, 4 :20–26, 06 1999.

[142] Anirut Suebsing and Nualsawat Hiransakolwong. Feature selection using euclidean distance and cosine similarity for intrusion detection model. In *2009 First Asian Conference on Intelligent Information and Database Systems*, pages 86–91, 2009.

[143] Springer. *MARS : A Video Benchmark for Large-Scale Person Re-identification*, 2016.

[144] Gregory R. Koch. Siamese neural networks for one-shot image recognition. 2015.

[145] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip H. S. Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. HOTA : A higher order metric for evaluating multi-object tracking. *CoRR*, abs/2009.07736, 2020.

[146] Gedas Bertasius, Lorenzo Torresani, and Jianbo Shi. Object detection in video with spatiotemporal sampling networks. *CoRR*, abs/1803.05549, 2018.

[147] Zheng Ge, Songtao Liu, Zeming Li, Osamu Yoshie, and Jian Sun. OTA : optimal transport assignment for object detection. *CoRR*, abs/2103.14259, 2021.

[148] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *CoRR*, abs/1702.03118, 2017.

[149] Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. *CoRR*, abs/2003.07540, 2020.

[150] Yue Wu, Yinpeng Chen, Lu Yuan, Zicheng Liu, Lijuan Wang, Hongzhi Li, and Yun Fu. Rethinking classification and localization in R-CNN. *CoRR*, abs/1904.06493, 2019.

[151] Feng He, Jiubin Tan, Weibo Wang, Shutian Liu, Yuemin Zhu, and Zhengjun Liu. EFFNet : Element-wise feature fusion network for defect detection of display panels. *Signal Processing : Image Communication*, 119 :117043, 2023.

[152] Jiang Deng, Sun Bei, Su Shaojing, and Zuo Zhen. Feature fusion methods in deep-learning generic object detection : A survey. In *2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, volume 9, pages 431–437, 2020.

Djahlin NIKUE AMASSAH

Analyse vidéo pour la détection, le suivi et la reconnaissance du comportement pour l'animal en situation d'élevage

Résumé :
La reconnaissance d'activités, également connue sous le nom de reconnaissance d'actions, est un domaine de recherche en vision par ordinateur et en apprentissage automatique, avec diverses applications. L'application la plus courante est l'identification et la compréhension des activités humaines à partir de données visuelles, telles que des images ou des vidéos. Les techniques de reconnaissance d'actions peuvent être appliquées également à la surveillance du bétail, où elles contribuent à améliorer le bien-être des animaux, la productivité et les pratiques de gestion agricole. Ainsi, les travaux réalisés dans ce document se situent dans le cadre de l'analyse vidéo pour la détection, le suivi et la reconnaissance du comportement animal en situation d'élevage. Ces travaux sont réalisés au sein de ANIMOV « Animal Movements Observation from Videos », un projet de recherche pluridisciplinaire mis en œuvre sur la période 2019-2023 par un consortium régional en Centre-Val-de-Loire. Ce projet porte principalement sur deux espèces animales : les éléphants et les chèvres. Dans ce mémoire, nos recherches portent sur l'analyse des activités chez les chèvres. Afin de construire notre système d'analyse du comportement, nous avons mis en place un système de détection et de suivi d'objets. Pour la détection nous avons testé et comparé deux méthodes populaires de la littérature : YOLOv4 et Faster R-CNN, sur des bases de données créées par nos soins. Parmi les deux méthodes de détection, YOLOv4 présente de meilleures performances en terme de précision moyenne et est 2.5 fois plus rapide que le Faster R-CNN. Pour le suivi des chèvres, nous avons testé et comparé également deux méthodes populaires de la littérature : SORT et Deep SORT. L'évaluation des deux méthodes de suivi sur les vidéos de test montre une légère amélioration de Deep SORT par rapport à SORT en terme d'association des données. Cependant, SORT reste plus rapide et plus adapté à un système temps réel. Le système de détection et de suivi mis en place, nous permet de réaliser, en temps réel, l'analyse de l'activité générale du troupeau, avec des indicateurs assez proches de la réalité. La principale faiblesse dans notre système est la perte de détection sur certaines images de la vidéo, qui entraîne des échecs dans le suivi. Ainsi, pour améliorer les performances, nous avons proposé une approche qui fusionne les informations des détections précédentes et de l'image courante, dans une nouvelle architecture de détection (YOLOX), afin de mieux détecter tous les objets sans perdre les anciens.

Mots clés : reconnaissance d'actions, détection d'objets, suivi d'objets, comportement animal, apprentissage profond

Video analysis for detection, tracking and recognition of animal behavior in breeding situations

Abstract :
Activity recognition, also known as action recognition, is a field of research in computer vision and machine learning, with a variety of applications. One of the most common applications is the identification and understanding of human activities from visual data, such as images or videos. Action recognition techniques can also be applied to livestock monitoring, where they can help improve animal welfare, productivity, and farm management practices. Thus, the work conducted in this document falls within the context of video analysis for the detection, monitoring, and recognition of animal behavior in livestock situations. This work is being achieved within ANIMOV "Animal Movements Observation from Videos", a multidisciplinary research project being implemented over the period 2019-2023 by a regional consortium in Centre-Val-de-Loire. This project concerns two main animal species : elephants and goats. In this thesis, our research focuses on activity analysis for goats. To implement our behavior analysis system, we have built an object detection and tracking system. For detection, we tested and compared two popular methods from the literature : YOLOv4 and Faster R-CNN, on self-created datasets. Of the two detection methods, YOLOv4 performs better in terms of average accuracy and is 2.5 times faster than Faster R-CNN. For goat tracking, we also tested and compared two popular methods from the literature : SORT and Deep SORT. Evaluation of both tracking methods on test videos shows a slight improvement of Deep SORT over SORT in terms of data association. However, SORT is faster and better suited to a real-time system. The detection and tracking system we have set up enables us to analyze the general activity of the livestock in real time, with indicators that are fairly close to reality. The main limitation with our system is the loss of detection on certain video images, which leads to tracking failures. So, to improve the performance, we proposed an approach that merges information from previous detections and the current image, in a new detection architecture (YOLOX), to better detect all objects without losing the old ones.

Keywords : action recognition, object detection, object tracking, animal behavior, deep learning

PRISME
Laboratoire Pluridisciplinaire de Recherche Ingénierie des Systèmes, Mécanique, Énergétique

Laboratoire Pluridisciplinaire de Recherche Ingénierie des Systèmes, Mécanique, Énergétique, 8 Rue Léonard de Vinci, 45100 Orléans

Centre-Val de Loire
www.regioncentre-valdeloire.fr