PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Détection et segmentation des usagers de la route dans des images et vidéos
Auteur: Author:	Hughes Perreault
Date:	2021
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Perreault, H. (2021). Détection et segmentation des usagers de la route dans des images et vidéos [Ph.D. thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/9471/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/9471/
Directeurs de recherche: Advisors:	Guillaume-Alexandre Bilodeau, & Nicolas Saunier
Programme: Program:	Génie informatique

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Détection et Segmentation des Usagers de la Route dans des Images et Vidéos

HUGHES PERREAULT
Département de génie informatique et génie logiciel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Génie informatique

Octobre 2021

© Hughes Perreault, 2021.

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Détection et Segmentation des Usagers de la Route dans des Images et Vidéos

présentée par **Hughes PERREAULT**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Samuel KADOURY, président
Guillaume-Alexandre BILODEAU, membre et directeur de recherche
Nicolas SAUNIER, membre et codirecteur de recherche
Lama SEOUD, membre
Jean-François LALONDE, membre externe

DÉDICACE

À ma fille Agathe,
Tu as été une grande source de motivation.
Tu remets la grandeur des choses en perspective.

REMERCIEMENTS

Tout d'abord, je tiens à remercier de tout cœur mon directeur Guillaume-Alexandre Bilodeau pour sa présence incroyable, ses bons conseils techniques, sa direction de ma recherche ainsi que ses révisions de mes textes. Je crois sincèrement qu'un étudiant ne peut pas mieux être dirigé que par Guillaume-Alexandre, je suis fier d'être son étudiant. Un grand merci également à mon codirecteur Nicolas Saunier pour les discussions inspirantes lors des rencontres, ses révisions des articles ainsi que ses conseils généreux et son temps tout au long de mon doctorat.

Mon doctorat ne se serait pas fait sans l'aide du groupe de recherche en image processing de Genetec. De ce fait, je tiens à remercier premièrement Paule Brodeur, qui a été l'instigatrice du projet de collaboration entre Genetec et Polytechnique Montréal. Un grand merci d'avoir cru en moi Paule, cela me fait chaud au cœur. Ensuite, mes plus sincères remerciements à Pierre Gravel, que je considère comme un de mes mentors académiques. Pierre, merci pour ta curiosité et ta rigueur scientifique. Puis, un grand merci à Maguelonne Héritier pour ses excellentes idées créatives et son ambition, elle a été inspirante. Enfin, un grand merci à Genetec pour son support financier sans qui ce projet n'aurait pas pu voir le jour.

Je voudrais également remercier ma famille pour leur encouragement durant toutes ces années. En particulier ma conjointe Arielle qui m'a le plus souvent remonté le moral lorsque la motivation n'était pas au rendez-vous. Également, un merci tout spécial à ma fille Agathe. Tu as été une grande source de motivation vers la fin.

Finalement, je voudrais remercier tous mes collègues du LITIV. Votre présence a rendu l'expérience beaucoup plus agréable, et vous allez me manquer.

RÉSUMÉ

Les méthodes de détection d'objets sont largement dominées par l'apprentissage profond depuis quelques années, apportant de grandes améliorations. Cette révolution a servi à un ensemble d'applications dans le domaine du transport. On pense par exemple aux applications d'analyse de trafic et aux divers systèmes d'aide à la conduite. Certaines applications sont en devenir et vont requérir de plus amples améliorations, par exemple un système de conduite complètement automatisé.

Cette thèse présente un ensemble de méthodes permettant de détecter et segmenter des usagers de la route dans des images et des vidéos. Les méthodes présentées s'attaquent aux défis propres au domaine du trafic routier, c'est-à-dire des conditions lumineuses et météorologiques difficiles, une haute densité de petits objets ainsi que de nombreuses occlusions partielles.

La première méthode que nous présentons est une méthode de détection d'objets dans les images. Cette méthode tire profit d'annotations semi-supervisées créées par flux optique et soustraction d'arrière-plan. Ces annotations sont utilisées pour entraîner le réseau à générer une carte de saillance qui servira à deux objectifs. Premièrement, un processus d'attention est effectué afin de diriger le réseau vers les zones d'intérêts sur les cartes d'attributs. Deuxièmement, une binarisation de la carte de saillance est effectuée pour obtenir une segmentation des objets d'intérêts dans l'image. Cette méthode est implémentée dans le détecteur CenterNet [1]. En raison de la carte d'attention qui illumine les zones d'intérêts dans l'image, nous nommons notre méthode SpotNet [2].

La deuxième méthode présentée propose une architecture de détection d'objets sur vidéo en fusionnant des cartes d'attributs de trames temporellement proches. Afin de fusionner les cartes d'attributs, nous introduisons un module composé de concaténation de canaux, convolutions 1×1, suivi de réarrangement des canaux. Cette architecture ainsi que le module de fusion sont assez génériques pour être intégrés dans plusieurs détecteurs. En raison de sa rapidité et de ses bonnes performances, nous utilisons RetinaNet [3] comme détecteur de base, et pour cette raison, nous nommons cette méthode RN-VID [4].

Troisièmement, nous présentons une extension ainsi qu'une combinaison des deux travaux précédents. Dans ce travail, nommé FFAVOD, nous implémentons l'architecture de RN-VID dans deux nouveaux détecteurs modernes, CenterNet et SpotNet. Nous démontrons que nous pouvons améliorer les résultats avec un ensemble de détecteurs de bases différents. De plus, nous améliorons le module d'attention de SpotNet et de ce fait, améliorons les performances

de SpotNet. La combinaison de SpotNet amélioré avec l'architecture de fusion des cartes d'attributs de FFAVOD atteint l'état de l'art sur deux jeux de données de trafic routier. De plus, nous comparons le module de fusion avec plusieurs stratégies de fusion différentes et démontrons son utilité.

Le dernier travail présenté consiste en une méthode de segmentation d'instances par polygones englobants fonctionnant en temps réel. Dans ce travail, nous développons une façon d'adapter des annotations de masques en des annotations de polygones pouvant facilement être modélisés et appris par un réseau. Nous entraînons un réseau à produire un polygone par position dans l'image, qui correspond au polygone englobant de l'objet dont le centre se trouve à cette position. Parallèlement, nous détectons le centre des objets présents dans l'image, ainsi que le décalage de chaque objet. Aussi, afin de pallier le problème du grand nombre d'usagers de la route se chevauchant dans les images, par exemple une série de voitures stationnées et une foule de piétons, nous intégrons une tête au réseau qui calcule la profondeur relative des objets. Grâce à cette profondeur relative, nous pouvons placer les objets les dans le bon ordre et améliorer le résultat final.

Finalement, cette thèse présente une série de méthodes couvrant un grand spectre allant de la détection d'objets dans les images, couvrant les vidéos et touchant la segmentation d'instances par polygones englobants. L'ensemble des méthodes présentées dans cette thèse offre une solution flexible et prometteuse au problème de localisation des usagers de la route dans diverses applications. Afin de participer à l'écosystème de la recherche ainsi que pour promouvoir la reproductibilité des résultats, le code pour chaque méthode a été publié en ligne.

ABSTRACT

Visual object detection methods have been largely dominated by deep learning in recent years, bringing great improvements to this field. This revolution has benefited a set of applications in the field of transportation, for example traffic analysis applications and various driving assistance systems. Other applications are in the making, and will require further improvements, for instance a fully automated driving system.

This thesis presents a set of methods for detecting and segmenting road users in images and videos. The presented methods tackle the challenges specific to the field of road traffic, i.e. difficult light and weather conditions, a high density of small objects as well as numerous partial occlusions.

The first method is an object detection method. This method takes advantage of semi-supervised annotations created by optical flow and background subtraction. These annotations are used to train the network to generate a saliency map that will serve two purposes. First, an attention process is used to focus the network on the areas of interest on the feature maps. Second, a binarization of the saliency map is performed to obtain a segmentation of the objects of interest in the image. This method is implemented in the CenterNet detector [1]. Due to the attention map that illuminates areas of interest in the image, we name our method SpotNet [2].

The second method presented proposes an architecture for video object detection by merging feature maps of temporally close frames. In order to merge the feature maps, we introduce a module composed of channel concatenation, 1×1 convolutions followed by channel reordering. This architecture as well as the fusion module are generic enough to be integrated into several detectors. Due to its speed and good performance, we use RetinaNet [3] as our base detector, and for this reason we name this method RN-VID [4].

Third, we present an extension as well as a combination of the two previous works. In this work, named FFAVOD, we implement the architecture of RN-VID in two modern detectors, CenterNet and SpotNet. We show that we can improve the results with multiple different base detectors. In addition, we improve the attention module of SpotNet and therefore improve the performance of the base SpotNet. Combining enhanced SpotNet with FFAVOD fusion module achieves state of the art on two road traffic datasets. Additionally, we compare the fusion module with different fusion strategies and demonstrate its usefulness.

The last work presented consists of an instance segmentation method by bounding polygons

running in real time. In this work, we develop a way to adapt mask annotations into polygon annotations that can easily be modelled and learned by a network. We train a network to produce one polygon per position in the image, which is the bounding polygon of the object whose center is at that position. At the same time, we detect the center of the objects present in the image, as well as the offset of each object. Also, in order to overcome the problem of the large number of overlapping road users in the images, for example a series of parked cars or a crowd of pedestrians, we integrate a network head which calculates the relative depth of the objects. Thanks to this relative depth, we can place the objects in the right order and improve the end result.

Finally, this thesis presents a series of methods covering a large spectrum ranging from the detection of objects in images and videos, to instance segmentation by bounding polygons. The set of methods presented in this thesis offers a flexible and promising solution to the problem of locating road users in various applications. In order to participate in the research ecosystem as well as to promote the reproducibility of the results, the code for each method has been published online.

TABLE DES MATIÈRES

DÉDICACE iii

REMERCIEMENTS iv

RÉSUMÉ v

ABSTRACT vii

TABLE DES MATIÈRES ix

LISTE DES TABLEAUX xiii

LISTE DES FIGURES xv

LISTE DES SIGLES ET ABRÉVIATIONS xix

CHAPITRE 1 INTRODUCTION 1
1.1 Définitions et concepts de base 2
1.2 Éléments de la problématique 4
1.2.1 Détection d'objets dans les images et les vidéos 4
1.2.2 Détection d'objets dans les vidéos 5
1.2.3 Segmentation d'instances 7
1.3 Objectifs de recherche 8
1.4 Contributions 9
1.5 Plan de la thèse 10

CHAPITRE 2 REVUE DE LITTÉRATURE 11
2.1 Réseaux de neurones convolutifs 11
2.2 Détection d'objets dans les images 12
2.2.1 Méthodes en deux phases 13
2.2.2 Méthodes en une phase 14
2.2.3 Méthode par détection de points clés 16
2.2.4 Évaluation de la performance 18
2.3 Détection d'objets dans les vidéos 19
2.3.1 Évaluation de la performance 22
2.4 Segmentation d'instances 22

2.4.1 Évaluation de la performance 25
2.5 Jeux de données 25

CHAPITRE 3 SURVOL DES ARTICLES 27

CHAPITRE 4 ARTICLE 1 : SPOTNET : SELF-ATTENTION MULTI-TASK NETWORK FOR OBJECT DETECTION 30
4.1 Introduction 30
4.2 Related Work 32
4.3 Proposed Method 34
4.3.1 Base network 34
4.3.2 Multi-task Learning 34
4.3.3 Self-Attention Mechanism 35
4.3.4 Semi-Supervised annotations 36
4.3.5 Training for multiple tasks 36
4.4 Experiments 37
4.4.1 Datasets 37
4.4.2 Implementation Details 38
4.4.3 Object detection results 38
4.4.4 Foreground/Background segmentation results 39
4.5 Discussion 41
4.5.1 Ablation study 41
4.5.2 Limitations of our Model 42
4.6 Conclusion 43

CHAPITRE 5 ARTICLE 2 : RN-VID : A FEATURE FUSION ARCHITECTURE FOR VIDEO OBJECT DETECTION 45
5.1 Introduction 45
5.2 Related Work 46
5.2.1 Object Detection 46
5.2.2 Video Object Detection 48
5.2.3 Optical flow by CNNs 48
5.3 Proposed Method 49
5.3.1 Baseline : RetinaNet 49
5.3.2 Model Architecture 50
5.3.3 Fusion Module 50
5.4 Experiments 52

5.4.1 Datasets . . . 52
5.4.2 Implementations Details . . . 52
5.4.3 Performance Evaluation . . . 53
5.4.4 Results . . . 54
5.5 Discussion . . . 55
5.5.1 Analysis . . . 55
5.5.2 Ablation Study . . . 56
5.5.3 Limitations of our Model . . . 57
5.6 Conclusion . . . 57

CHAPITRE 6 ARTICLE 3 : FFAVOD : FEATURE FUSION ARCHITECTURE FOR VIDEO OBJECT DETECTION . . . 59
6.1 Introduction . . . 59
6.2 Related Work . . . 60
6.2.1 Object Detection . . . 60
6.2.2 Video Object Detection . . . 62
6.2.3 Feature Fusion Strategies . . . 63
6.3 Proposed Method . . . 64
6.3.1 Frame Fusion Architecture . . . 64
6.3.2 Fusion Module . . . 65
6.3.3 SpotNet improvement . . . 66
6.4 Experiments . . . 66
6.4.1 Base object detectors . . . 66
6.4.2 Datasets . . . 67
6.4.3 Implementations Details . . . 67
6.4.4 Performance Evaluation . . . 68
6.5 Results and Discussion . . . 68
6.5.1 Ablation Study . . . 71
6.5.2 Limitations . . . 71
6.6 Conclusion . . . 72
6.7 Acknowledgment . . . 72

CHAPITRE 7 ARTICLE 4 : CENTERPOLY : REAL-TIME INSTANCE SEGMENTATION USING BOUNDING POLYGONS . . . 75
7.1 Introduction . . . 75
7.2 Related Work . . . 77
7.2.1 Object detection . . . 78

7.2.2 Instance Segmentation 79
7.3 Proposed Method 79
7.3.1 Modified Heatmaps 80
7.3.2 Polygon Regression Head 81
7.3.3 Vertex Selection Policy 81
7.3.4 Relative Depth Head 83
7.3.5 Training 84
7.4 Experiments 85
7.4.1 Evaluation datasets 85
7.4.2 Implementation Details 85
7.4.3 Results 86
7.5 Discussion 88
7.5.1 Ablation Study 88
7.5.2 Limitations 89
7.6 Conclusion 90

CHAPITRE 8 DISCUSSION GÉNÉRALE 91
8.1 Méthodes par apprentissage versus méthodes classiques 91
8.2 Détection d'objets dans les images 92
8.3 Détection d'objets dans les vidéos 93
8.4 Segmentation d'instances 94
8.5 Utilité des différents éléments 95

CHAPITRE 9 CONCLUSION 97
9.1 Synthèse des travaux 97
9.2 Recommandations pour travaux futurs 98

RÉFÉRENCES 99

LISTE DES TABLEAUX

Tableau 4.1 Results on the UA-DETRAC dataset [5]. 3D-DETNet results are from [6], and others results are reported as in the results section of the UA-DETRAC website (**Boldface** : best result, *Italic* : indicates our baseline). 40
Tableau 4.2 Results on the UAVDT [7] dataset (**Boldface** : best result, *Italic* : indicates our baseline). 41
Tableau 4.3 Results on the changedetection.net [8] dataset. Results are averaged for sequences “highway”, “traffic” and “boulevard” (**Boldface** : best result). 41
Tableau 4.4 Ablation study on the UA-DETRAC [5] dataset. 42
Tableau 5.1 mAP reported on the UA-DETRAC test set compared to our baseline as well as classic state-of-the-art detectors. Results for “Ours” and “RN-VGG16” are generated using the evaluation server on the UA-DETRAC website, 3D-DETNet [6] is reported as in their paper, and others are as reported in the results section of the UA-DETRAC website. **Boldface** : best result, *Italic* : baseline. 55
Tableau 5.2 mAP reported on the UAVDT test set compared to our baseline as well as classic state-of-the-art detectors. Results for "Ours" and "RN-VGG16" are generated using the official Matlab toolbox provided by the authors, others are reported as in their paper. **Boldface** : best result, *Italic* : baseline. 56
Tableau 5.3 mAP reported on the UAVDT test set for different variations of our model to conduct an ablation study. Results are generated using the official Matlab toolbox provided by the authors. Number of frames is the number of frames used for each detection. 57
Tableau 6.1 mAP of FFAVOD applied to base detectors on the UA-DETRAC test set compared their respective base detectors, as well as classic state-of-the-art detectors. FFAVOD uses $n = 2$. Results for FFAVOD and their base detectors are generated using the official toolkit from the UA-DETRAC website. **Boldface** indicates the best result overall, Underlineindicates the best result within a section, while *Italic* indicates the baseline and *indicates the use of multiple frames. 69

Tableau 6.2 mAP of our FFAVOD applied to detectors on the UAVDT test set compared their respective base detectors. FFAVOD uses $n = 2$. Results for our FFAVOD and their base detectors are generated using the official Matlab toolkit provided by the authors. The other results are taken from their respective papers. **Boldface** indicates the best result overall, Underlineindicates the best result within a section, while *Italic* indicates the baseline and *indicates the use of multiple frames. . . . 70

Tableau 6.3 Different fusion strategies to conduct an ablation study. Results are generated using the official Matlab toolbox provided by the authors. **Boldface** indicates the best result overall 72

Tableau 7.1 Results on the cityscapes, KITTI and IDD test sets, as shown on their respective public benchmark, divided between faster (bottom) and slower (top) methods. When not available, runtimes were estimated. Mask type : Full : based on pixel-wise labels, polygon : based on a bounding polygon. Boldface : best results. Results for PANet and Mask R-CNN on IDD were taken from the original IDD paper [9] 87

Tableau 7.2 Ablation study of different parts of CenterPoly as well as performance with various backbones. AP and AP50% on the validation set of cityscapes. Depth refer to the use of our relative depth map. Elliptical refers to the use of elliptical GT. Center of gravity refer to the center heatmaps being at the center of gravity instead of the center of the bounding boxes. 89

LISTE DES FIGURES

Figure 1.1 Un résultat de détection d'objets sur le jeu de données MS COCO. Cette image provient de MTheiler (nom d'usager Wikimédia) et est distribuée sous licence CC BY-SA [10]. 3
Figure 1.2 Index de Jaccard selon divers alignements de rectangles. [11]. 3
Figure 1.3 Un résultat de segmentation d'instances sur le jeu de données cityscapes [12] 4
Figure 1.4 Exemples de difficultés en détection d'objets sur le jeu de données UA-DETRAC [5] 6
Figure 1.5 Plusieurs images capturées par drone montrant l'évolution de l'arrière-plan sur le jeu de données UAVDT [7] 6
Figure 1.6 Une image contenant beaucoup de piétons se chevauchant dans la vue de la caméra sur le jeu de données cityscapes [12] 7
Figure 2.1 L'architecture d'un bloc résiduel [13]. © 2016 IEEE 12
Figure 2.2 L'architecture d'un module sablier du réseau Hourglass [14]. © 2016 Springer International Publishing AG 12
Figure 2.3 Fonctionnement des boîtes ancres [15]. © 2017 IEEE 14
Figure 2.4 Architecture de Faster R-CNN [15]. © 2017 IEEE 15
Figure 2.5 Architecture de YOLO [16]. © 2016 IEEE 16
Figure 2.6 Architecture de RetinaNet [3]. © 2017 IEEE 17
Figure 2.7 Architecture de Keypoints Triplets [17]. © 2019 IEEE 18
Figure 2.8 Un exemple de courbe représentant le mAP avec quelques méthodes [2]. © 2020 IEEE 19
Figure 2.9 Architecture de FGFA [18]. © 2017 IEEE 20
Figure 2.10 Architecture de LSTM-SSD [19]. © 2018 IEEE 21
Figure 2.11 Architecture de Mask R-CNN [20]. © 2017 IEEE 23
Figure 2.12 Architecture de PolarMask [21]. © 2020 IEEE 24
Figure 2.13 (a) Exemple d'image de UA-DETRAC [5] et ses annotations. (b) Exemple d'image de UAVDT [7] et ses annotations. 25
Figure 2.14 Deux exemples d'annotations sur le jeu de données cityscapes [12]. . . 26
Figure 3.1 Un réseau de détection de base, un prenant deux images consécutives concaténées et un prenant une image et une image de flux optique concaténées. [22] 28

Figure 4.1 A visualisation of the attention map produced by SpotNet on top of its corresponding image, from the UAVDT [7] dataset. 32
Figure 4.2 Overview of SpotNet : the input image first passes through a double-stacked hourglass network ; the segmentation head then produces an attention map that multiplies the final feature map of the backbone network ; the final center keypoint heatmap is then produced as well as the size and coordinate offset regressions for each object. 35
Figure 4.3 Example of semi-supervised annotations on UA-DETRAC [5] produced by PAWCS [23] and the intersection with the ground-truth bounding boxes. 37
Figure 4.4 Sample from UA-DETRAC with the ground-truth bounding boxes in yellow. 38
Figure 4.5 Sample from UAVDT with the ground-truth bounding boxes in yellow. 39
Figure 4.6 Example of foreground/background segmentation maps obtained with several segmentation methods. First row : frame 1015 of “highway”, second row : frame 967 of “traffic”, third row : frame 883 of “boulevard”. 42
Figure 4.7 Precision/Recall curve of our model compared with a variant and other methods. 43
Figure 5.1 Qualitative examples where our model (blue) performs better than the RetinaNet baseline (red). (a) the two cars in the back are heavily occluded by the green truck, (b) the car in the bottom center is being occluded by the frame boundary, (c) the green truck is blurry due to motion blur, (d) as cars become smaller, they become harder to detect, like the white one at the top. 47
Figure 5.2 A representation of our architecture with $n = 2$. Each frame is passed through a pre-trained VGG-16, and the outputs of block 3, block 4 and block 5 are collected for fusion. B1 to B5 are the standard VGG-16 [24] blocks, and P3 to P7 are the feature pyramid levels. In the dotted frame is an overview of our baseline, a RetinaNet [3] with VGG-16 as a backbone. 51
Figure 5.3 Our fusion module consists of channel re-ordering, concatenation, 1×1 convolution, and a final concatenation (better seen in color). 52
Figure 5.4 (a) An example frame of UA-DETRAC and its ground-truth annotations. (b) An example frame of UAVDT and its ground-truth annotations. 53

Figure 5.5 Precision-Recall curves on UA-DETRAC [5] (a) and UAVDT [7] (b) for RN-VID (Ours), RN-VGG16 (Baseline) and a few other state-of-the-art methods. 57

Figure 6.1 In this image from the UA-DETRAC dataset [5], one can see multiple examples where the proposed architecture applied on SpotNet [2] (blue) outperforms its baseline (yellow). Detections by both models are shown in green (considering a minimum IOU score of 0.8 to match them). Examples of improved performance include cases of occlusion and of smaller objects (top center and both top corners). The blurred rectangles represent regions excluded for the evaluation, but where the proposed architecture nonetheless can make better detections. 61

Figure 6.2 A visual representation of FFAVOD with a window of 5 frames ($n = 2$). Frames are passed through the backbone network of the base object detection network, and the fusion module takes their outputs as input. Finally, the fusion module outputs a fused feature map compatible with the base object detection network, and the base object detection heads are applied to the fused feature map to classify the object categories and regress the bounding boxes. 65

Figure 6.3 The fusion module. Channels are represented by colors. The fusion module is composed of channel grouping, concatenation followed by 1×1 convolution and a final re-ordering of channels. 73

Figure 6.4 mAP reported on the UA-DETRAC test set for different values of n of FFAVOD 74

Figure 7.1 CenterPoly produces : (a) an accurate instance segmentation by detecting objects and regressing a bounding polygon for each and (b) a relative depth value for each object. In this example, polygons have 16 vertices. 76

Figure 7.2 An overview of the CenterPoly architecture. The image first passes through a CNN backbone, displayed here as an Hourglass network. The feature map is then shared between four network heads, the polygon regression head, the center heatmaps head used for detections, the object offset head and the relative depth map head. The sizes displayed are for illustration purposes only, please refer to the code for the detailed architecture. 80

Figure 7.3 The vertex selection strategy : we trace lines at regular intervals in the bounding box, starting at the top left corner and going clockwise. The selected vertex is the first point on the line within the instance mask when going from the bounding box toward the center. The interval changes depending on the number of vertices used in the method. . . 82
Figure 7.4 Qualitative relative depth predictions of CenterPoly on the cityscapes dataset. Darker is closer, and lighter is further. 84
Figure 7.5 Qualitative instance segmentation results of CenterPoly on the cityscapes dataset. 88

LISTE DES SIGLES ET ABRÉVIATIONS

AP	Average Precision
CNN	Convolutional Neural Network
FPN	Feature Pyramid Network
IOU	Intersection Over Union
mAP	mean Average Precision
ReLU	Rectified Linear Unit
RPN	Region Proposal Network
SVM	Support Vector Machine

CHAPITRE 1 INTRODUCTION

Avec les voitures et les villes qui possèdent de plus en plus de capteurs, les données disponibles ne cessent de croître, et il faut pouvoir les traiter et les comprendre si on veut qu'elles soient utiles. Comme traiter un tel volume de données manuellement n'est pas envisageable, il faut des méthodes informatiques pour pouvoir les traiter automatiquement. L'interprétation automatique d'une image ou d'une vidéo est loin d'être simple, et c'est une tâche sur laquelle travaillent les chercheurs du domaine de la vision par ordinateur depuis plusieurs décennies. La détection et la classification des usagers de la route dans des images et des vidéos couvrent un grand nombre d'applications. Certaines existent actuellement alors d'autres sont en devenir. Pensons premièrement à l'analyse du trafic routier. Nous pouvons considérer des applications comme l'analyse de la sécurité d'une nouvelle infrastructure routière par exemple ou l'estimation du trafic à l'aide de caméras installées sur les infrastructures routières. Deuxièmement, nous pouvons considérer l'ensemble des systèmes d'aide à la conduite. Que ce soit l'aide au freinage, une sécurité accrue des angles morts, l'aide au stationnement ou tout ce qui touche à l'automatisation de la conduite, ces systèmes peuvent tous bénéficier d'une détection automatique des autres usagers de la route autour d'eux. Un système de conduite complètement automatisé est difficilement imaginable sans un moyen de détecter les piétons, les cyclistes et les automobiles. Notons aussi que différencier ces usagers est crucial, car le comportement du système de conduite ne répondra pas de la même manière selon le type de l'usager de la route.

Les applications mentionnées ci-haut sont composées d'un pipeline d'étapes assez long, allant des interfaces usagers jusqu'aux optimisations matérielles. Dans le cadre cette thèse, nous nous intéressons aux tâches relativement bas niveau de traitement des images afin d'en retirer la position et le type des objets d'intérêts s'y trouvant. Ces informations pourraient par la suite être utilisées pour une analyse des relations entre les objets de la scène, un suivi des objets sur plusieurs trames, une réidentification des objets sur des images prises par d'autres caméras, etc.

Il est à noter qu'en 2012, AlexNet [25] déclencha une révolution dans le domaine de la vision par ordinateur en remportant la première place lors de la fameuse compétition de classification ImageNet [26] en utilisant un réseau de neurones convolutif. Depuis, la vision par ordinateur est plutôt dominée par des méthodes qui utilisent l'apprentissage profond au détriment des méthodes dites "classiques" qui n'en utilisent pas. Cette thèse portera sur les méthodes par apprentissage profond et omettra les méthodes classiques qui sont généralement dépassées

pour ce qui est du sujet de la thèse.

1.1 Définitions et concepts de base

Cette thèse porte sur trois tâches similaires, mais légèrement différentes. Nous allons formellement les définir et les différencier ici.

La détection d'objets dans les images consiste à détecter et à classifier chaque objet appartenant à un ensemble de catégories prédéfinies dans une image. Pour détecter, il s'agit de donner la taille et la position d'un rectangle englobant l'objet. Un rectangle peut être défini par quatre scalaires, typiquement le coin supérieur gauche (x_0, y_0) et le coin inférieur droit (x_1, y_1). Certaines conventions vont plutôt utiliser le coin supérieur gauche ainsi que la largeur et la hauteur. Pour classifier, il s'agit de donner la probabilité qu'un objet appartienne à une classe donnée, pour chaque classe, selon un ensemble préétabli de classes. L'ensemble préétabli est défini pour chaque jeu de données ou application visée. La figure 1.1 montre un exemple de détection sur le jeu de données MS COCO [27]. Pour évaluer la qualité d'une détection, l'indice de Jaccard est utilisé si la classe est bien identifiée. Si la classe n'est pas bien identifiée, la détection sera considérée comme incorrecte. Cet indice, aussi appelé l'intersection sur l'union (IOU), peut être compris comme l'intersection entre deux ensembles sur l'union entre ces deux mêmes ensembles, formellement définis par l'équation 1.1.

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{1.1}$$

Dans le cas de la détection d'objets sur des images, les ensembles qui sont comparés sont des rectangles. L'indice de Jaccard est une mesure assez sévère, un léger désalignement peut avoir un gros impact sur le score final, comme on peut le voir sur la figure 1.2. Typiquement, on considérera un index de Jaccard minimum pour considérer une détection comme valide. La valeur dépendra des jeux de données, mais un index minimal de 0.5 ou 0.7 est assez typique.

La détection d'objets dans les vidéos est similaire à celle dans les images. La différence se trouve dans le fait que l'information temporelle peut être utilisée pour améliorer les résultats. Cette information peut être utilisée de plusieurs façons. D'abord, plusieurs trames consécutives peuvent être combinées pour en enrichir les caractéristiques qui décrivent les objets. Ensuite, l'information de mouvement peut être utilisée, par exemple avec des techniques de flux optique. Finalement, une cohérence temporelle peut être appliquée, par exemple en suivant les objets sur plusieurs trames et en éliminant sur des images ceux qui ne sont pas cohérents ou qui ne correspondent pas avec les autres trames de la vidéo. Par exemple, on pourrait

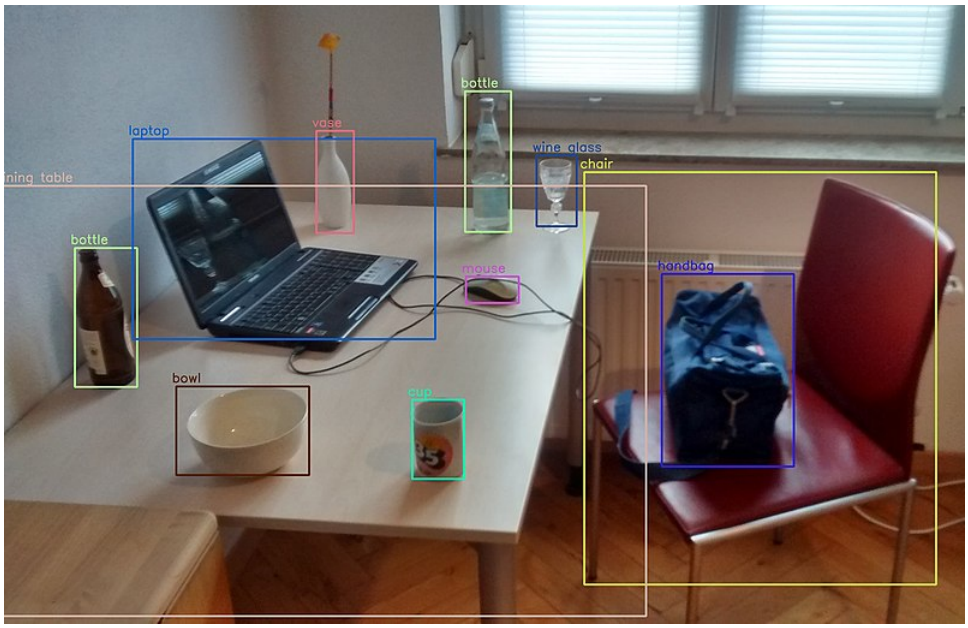


Figure 1.1 Un résultat de détection d'objets sur le jeu de données MS COCO. Cette image provient de MTheiler (nom d'usager Wikimédia) et est distribuée sous licence CC BY-SA [10].

vouloir éliminer une détection qui apparaît sur une seule trame et n'a aucune cohérence temporelle avec ses trames voisines. Une limitation de ces techniques est bien évidemment qu'elles ont besoin de séquences vidéo pour fonctionner.

La dernière tâche qui est explorée dans cette thèse est la segmentation d'instances. Se trouvant à l'intersection entre la segmentation sémantique et la détection d'objets, la segmentation d'instances consiste à segmenter et classifier tous les objets d'intérêt dans une scène. Segmenter est défini ici comme identifier chaque pixel faisant partie d'un objet. Il n'est pas requis que la segmentation soit en seul segment connecté. Les catégories des objets d'intérêts

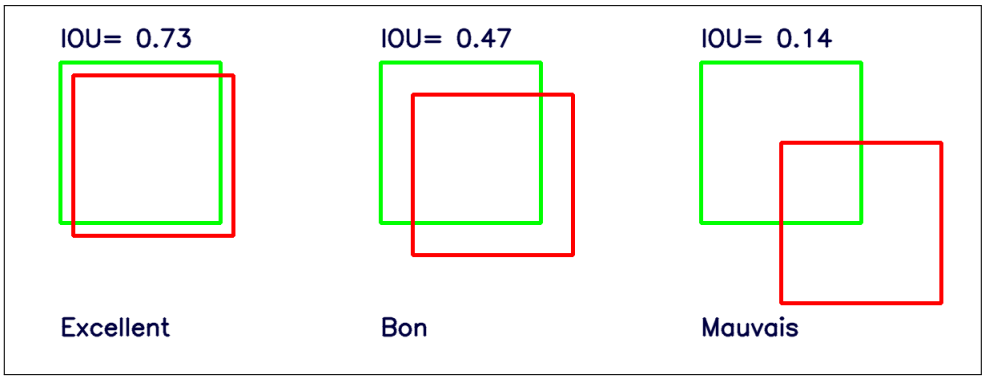


Figure 1.2 Index de Jaccard selon divers alignements de rectangles. [11].

sont encore une fois définies par le jeu de données ou l'application. Un exemple de résultat de segmentation d'instances sur le jeu de données cityscapes [12] peut être vu dans la figure 1.3. Il est important de noter que les différentes instances doivent être distinguées, il faut être capable d'isoler chaque piéton par exemple. Pour évaluer la qualité d'une segmentation, l'indice de Jaccard est aussi utilisé. À la place d'utiliser des rectangles, ce sont l'union et l'intersection entre les pixels détectés et les pixels de la vérité terrain qui sont calculés.



Figure 1.3 Un résultat de segmentation d'instances sur le jeu de données cityscapes [12]

1.2 Éléments de la problématique

1.2.1 Détection d'objets dans les images et les vidéos

Même si la détection d'objets a atteint un stade assez mature, c'est loin d'être un problème résolu et il existe toujours un bon nombre de défis à surmonter.

L'occlusion partielle survient lorsqu'un objet que l'on veut détecter est partiellement caché par un objet ou autre élément de l'image. C'est un problème difficile à résoudre pour les méthodes de vision par ordinateur, car il n'y a pas moyen de récupérer les pixels ainsi perdus. Cette difficulté est particulièrement présente dans les scènes de trafic routier où plusieurs véhicules de tailles différentes vont être très rapprochés les uns des autres (voir figure 1.4(a)). L'occlusion partielle survient aussi très régulièrement pour les objets coupés par les bordures de l'image. Ces objets peuvent être coupés de la moitié ou même du deux tiers, mais nous devons tout de même bien les localiser et les classifier. Notons aussi que l'occlusion totale n'est pas un problème dans la tâche de détection d'objets, contrairement à la tâche de suivi visuel, car lorsqu'un objet est totalement occlus nous ne devons pas le détecter ni le récupérer

dans des trames suivantes. L'occlusion partielle est une difficulté pour plusieurs raisons. Premièrement, comme de l'information sur l'objet est perdue, celui-ci devient plus difficile à classifier. Par exemple, il peut être difficile de distinguer un petit camion d'une voiture si nous n'avons accès qu'au-devant du véhicule. Ensuite, deux objets superposés peuvent être difficilement séparables, surtout si leur couleur est similaire. Ils pourraient ainsi être détectés comme un seul objet au lieu de deux. Finalement, un objet occlus est déformé de sa forme naturelle, il peut donc être plus difficile à localiser précisément, surtout s'il est coupé en deux. Par exemple si un autobus est coupé en deux par un poteau, il pourra être difficilement identifiable comme un seul objet.

Les conditions lumineuses et météorologiques sont d'autres difficultés auxquelles on doit faire face pour bien détecter les usagers de la route. Un bon modèle de détection dans ce contexte doit être capable de fonctionner de jour comme de nuit, avec toutes les conditions lumineuses entre les deux. Dans le cas de scènes de trafic routier, les scénarios peuvent être particulièrement différents si les véhicules ont leurs phares allumés la nuit, ce qui crée des zones très lumineuses potentiellement saturées (voir figure 1.4(b)). Ensuite, certaines conditions météorologiques diminuent la visibilité des objets d'intérêt, par exemple la pluie, le brouillard ou la neige. Ces conditions augmentent la difficulté, car elles ajoutent du bruit dans les images, cachant ou perturbant de l'information à laquelle on aimerait avoir accès.

Dans le domaine routier que nous étudions, la similarité de certaines catégories est une difficulté à laquelle nous devons faire face. Par exemple, plusieurs jeux de données vont distinguer un piéton d'un cycliste, ainsi que le cycliste de la bicyclette, alors qu'un piéton et un cycliste sont tous deux une personne. Dans ce cas, c'est seulement le contexte qui nous permet de distinguer les deux. Aussi, plusieurs types de véhicules sont similaires, par exemple une bicyclette, une mobylette et une motocyclette, ou une voiture et une camionnette.

Une dernière difficulté notable est la présence de structures créant de faux positifs. Pensons par exemple à des panneaux publicitaires rectangulaires, des abris d'autobus ou des terre-pleins ayant des formes et des couleurs ressemblant à des véhicules. Les détecteurs doivent se munir de puissantes capacités de classification afin de surmonter cette difficulté.

1.2.2 Détection d'objets dans les vidéos

Les défis de la détection d'objets sur des images sont aussi valides sur des vidéos. Cependant, certains défis sont propres à la détection d'objets sur vidéo.

Dans un contexte où l'information temporelle doit être intégrée d'une certaine façon, les mouvements de caméra peuvent causer un problème aux détecteurs. En effet, si un détecteur



(a) Une image contenant beaucoup de véhicules occlus (b) Une image contenant une scène de trafic de nuit

Figure 1.4 Exemples de difficultés en détection d'objets sur le jeu de données UA-DETRAC [5]

essaie de combiner l'information sur plusieurs trames consécutives, il serait plutôt avantageux de conserver un champ de vue stable, puisque nous essayons de détecter des objets de l'avant-plan qui ont leurs propres mouvements. Ce problème survient alors que la caméra est en mouvement (sur une voiture, un drone, une personne...), comme montré dans la figure 1.5.



Figure 1.5 Plusieurs images capturées par drone montrant l'évolution de l'arrière-plan sur le jeu de données UAVDT [7]

Le flou de mouvement causé par la vitesse des usagers de la route peut nuire à la détection de deux façons. Premièrement, il peut rendre flou un usager ce qui le rend plus difficile à classifier et détecter. En effet, comme il est flou, il se fond un peu plus avec l'arrière-plan et devient moins discernable. De plus, sa texture est modifiée, ce qui le rend plus difficile à associer à sa catégorie. Deuxièmement, le flou de mouvement peut modifier l'apparence du même usager s'il change significativement sa vitesse durant la séquence, ce qui peut le rendre plus difficile à associer entre les trames.

1.2.3 Segmentation d'instances

La segmentation d'instances est une tâche relativement jeune par rapport à la détection d'objets. C'est une tâche plus complexe qui peut être vue comme une progression naturelle de la détection. Il est donc normal que les problématiques touchant la détection soient aussi pertinentes pour la segmentation d'instances. De plus, il existe des problématiques propres à la segmentation d'instances.

L'occlusion partielle comme présentée ci-haut est bien sûr toujours une difficulté, mais cette difficulté est complexifiée lorsqu'il s'agit de segmentation. En effet, pour un objet dont une petite partie serait partiellement occluse, le rectangle englobant ne changera pas, ou très peu. Cela est dû au fait que les quatre coins les plus éloignés de l'objet ne changeront probablement pas même si une partie de celui-ci est cachée. Dans le cas d'une segmentation, les pixels cachés devront alors être identifiés comme ne faisant pas partie de l'objet, créant une difficulté supplémentaire. Cela est particulièrement difficile dans une scène de trafic dense où presque chaque véhicule en cache un autre (selon le point de vue de la caméra), ou dans une image contenant beaucoup de piétons où leurs membres s'entrecroisent (voir figure 1.6).



Figure 1.6 Une image contenant beaucoup de piétons se chevauchant dans la vue de la caméra sur le jeu de données cityscapes [12]

La déformabilité de certains objets pose une réelle difficulté. Par exemple pour un piéton, il est bien plus difficile de généraliser une segmentation fine qu'une forme rectangulaire générale. Cela est dû au fait que lorsqu'un piéton marche, ses bras et ses jambes peuvent se retrouver dans toutes les positions possibles, qui peuvent toutes être détectées plus ou moins par le même rectangle englobant, mais qui ont toutes des segmentations fines différentes. Cela cause une complexification de la tâche, et par le fait même un besoin d'expressivité plus grand pour

les modèles.

1.3 Objectifs de recherche

L'objectif du travail de recherche présenté dans cette thèse est de concevoir de nouvelles méthodes pour résoudre la tâche de localisation, classification et segmentation des usagers de la route. Les méthodes que nous allons proposer tenteront de tirer profit des caractéristiques propres au domaine des usagers de la route, et tenteront de résoudre certaines difficultés dans ce domaine et énoncées précédemment. Ces détecteurs auront pour but de localiser, classifier et potentiellement segmenter les usagers de la route dans un contexte de trafic routier. Plus spécifiquement, nos objectifs sont de :

1. Développer une méthode de détection des usagers de la route dans des images. Cette méthode doit prendre en entrée une image, et retourner un rectangle englobant ainsi que la catégorie et un score de confiance pour chaque usager de la route présent dans l'image. Les difficultés abordées dans ce travail sont causées par les cas d'occlusions partielles, de visibilité et conditions météorologiques difficiles, de grande densité d'objets ainsi que la présence d'éléments créant de faux positifs. Ici, on s'intéresse aux catégories d'usagers de la route définies dans les jeux de données annotés.
2. Développer une méthode de détection des usagers de la route dans des vidéos. Cette méthode doit prendre en entrée une séquence d'images consécutives temporellement proches, et retourner un rectangle englobant ainsi que la catégorie et un score de confiance pour chaque usager de la route présent pour chaque image dans la séquence. Cette méthode doit tirer profit de l'information temporelle afin d'avoir un avantage sur une méthode fonctionnant avec une seule image. Les difficultés abordées dans ce travail sont les sont les mêmes que pour les images, auxquels s'ajoutent le mouvement de la caméra.
3. Développer une méthode de segmentation d'instances rapide sur des images pour les usagers de la route. Cette méthode doit prendre en entrée une image et doit retourner une segmentation, une catégorie ainsi qu'un score de confiance pour chaque usager de la route présent dans l'image. Cette méthode doit fonctionner avec une vitesse le plus près possible du temps réel sur un ordinateur possédant une carte graphique moderne. Les difficultés abordées dans ce travail sont les cas d'occlusions partielles, de visibilité et conditions météorologiques difficiles, de grande densité d'objets, la présence d'éléments créant de faux positifs ainsi que la présence d'objets déformables tels des piétons. Ce qu'est un usager la route est toujours défini par le jeu de données sur lequel la méthode est entraînée.

1.4 Contributions

Les travaux composant cette thèse peuvent être séparés en trois tâches pour aborder les objectifs énoncés. Voici un résumé des articles et contributions composant le corps de la thèse en fonction de leur objectif associé.

Détection d'objets dans des images : pour cet objectif, nous avons développé et publié SpotNet : Self-attention multi-task network for object detection [2] (article 1). Dans ce travail [2], nous améliorons une méthode de détection d'objets en intégrant un processus d'auto-attention entraîné avec des annotations partielles. Grâce à cette information, un module produit une carte de saillance servant à diriger l'attention du réseau vers les zones pertinentes de l'image afin d'améliorer la performance. La carte de saillance sert à la fois à réduire l'attention sur les zones moins pertinentes et à réduire la présence de faux positifs.

Détection d'objets sur des vidéos : pour cet objectif, nous avons développé et publié RN-VID : A Feature Fusion Architecture for Video Object Detection [4] (article 2) ainsi que soumis une extension de cet article FFAVOD : Feature Fusion Architecture for Video Object Detection (article 3). RN-VID présente une méthode de détection d'objets sur vidéo présentant deux contributions principales. Premièrement, nous présentons une architecture permettant de combiner les cartes d'attributs de plusieurs trames temporellement proches à coût presque nul, étant assez générique pour pouvoir être intégré à plusieurs détecteurs d'objets sur image. Deuxièmement, nous présentons un module de fusion permettant de fusionner les cartes d'attributs de plusieurs trames temporellement proches afin d'en enrichir les attributs des exemples d'objets difficiles et ainsi la performance générale de la méthode.

FFAVOD présente une extension de RN-VID. Il comporte plusieurs contributions. Premièrement, la méthode est implémentée dans deux nouveaux détecteurs de base afin d'en confirmer l'efficacité et d'atteindre des résultats dépassant l'état de l'art. Deuxièmement, une amélioration au module de fusion de SpotNet est présentée afin de pousser encore plus loin la précision de la méthode. Enfin, une meilleure étude d'ablation est effectuée.

Segmentation d'instances : pour cet objectif, nous avons développé et publié CenterPoly : real-time instance segmentation using bounding polygons (article 4). Dans ce dernier travail, nous présentons une méthode de segmentation d'instances fonctionnant avec des polygones englobants. L'avantage de CenterPoly par rapport à d'autres méthodes est sa vitesse, qu'il peut atteindre parce qu'il utilise des polygones à la place des masques. CenterPoly produit aussi une carte de profondeur relative afin de placer les instances les unes devant les autres. Notre méthode atteint la meilleure performance des méthodes fonctionnant en temps réels sur trois jeux de données avec des usagers de la route.

1.5 Plan de la thèse

Nous présentons ici un plan de la structure de la thèse. Dans le chapitre 2, nous présenterons une revue de littérature des principaux travaux en détection d'objets, détection d'objets sur vidéo ainsi que segmentation d'instances. Dans le chapitre 3, nous présenterons un survol des articles composant le corps de la thèse. Dans le chapitre 4, nous présenterons une méthode de détection d'objets dans les images (article 1). Dans le chapitre 5 et 6, nous présenterons une méthode de détection d'objets dans les vidéos (article 2 et 3). Dans le chapitre 7, nous présenterons une méthode de segmentation d'instances (article 4). Dans le chapitre 8, nous discuterons de façon globale des travaux et des améliorations possibles. Finalement, le chapitre 9 présentera une conclusion de la thèse ainsi qu'une discussion de potentielles futures recherches.

CHAPITRE 2 REVUE DE LITTÉRATURE

2.1 Réseaux de neurones convolutifs

La majorité des méthodes de détection d'objets modernes utilise des réseaux de neurones convolutifs (CNN) afin de produire des cartes d'attributs. Pour cette raison, nous ferons une revue des principaux réseaux utilisés ici.

AlexNet [25] est le premier réseau de neurones à avoir remporté la prestigieuse compétition de reconnaissance d'images ImageNet [26] en 2012. Ce réseau a intégré plusieurs innovations dans le domaine de l'apprentissage profond, entre autres l'utilisation de "Rectified Linear Unit" ($ReLU$) comme fonction d'activation ainsi que la stratégie de "*dropout*" afin d'éviter le surapprentissage. Le réseau est composé de cinq couches de convolutions ainsi que trois couches pleinement connectées. Ce réseau a été utilisé pour la détection dans [28].

VGG [24] est une famille de réseaux composés de blocs. Les plus fameux réseaux de cette famille sont VGG-16 et VGG-19, le nombre représente le nombre de couches du réseau. Chaque bloc est composé de trois convolutions avec un filtre de 3×3 suivi d'une réduction de la résolution spatiale par une opération de sous-échantillonnage par le maximum. Les réseaux sont composés des blocs empilés les uns après les autres. Ce réseau a bien survécu dans le temps par sa robustesse et sa bonne performance et fut utilisé entre autre pour les détecteurs dans [29, 30]. L'élégance et la simplicité de toujours utiliser des filtres de même taille pour les convolutions de VGG ont grandement simplifié la conception des CNNs.

ResNet [13] aborde le problème de la dégradation du gradient lors de l'entraînement d'un réseau très profond. Ce travail introduit le concept de connexion raccourcie et de bloc résiduel. Une connexion raccourcie est une connexion qui saute une couche du réseau, ou plus. Ce type de connexion permet de faciliter l'apprentissage, car chaque couche peut utiliser des informations qui n'auraient pas été disponibles sinon. Un bloc résiduel (voir figure 2.1) est composé de deux couches de convolutions suivies de ReLU, ainsi que d'une connexion raccourcie allant du début de la première couche jusqu'à la fin de la deuxième. Les blocs résiduels sont utilisés comme des blocs de construction qui sont empilés les uns à la suite des autres afin de créer et d'entraîner des réseaux très profonds, de 100 à 150 couches. Ce réseau fonctionne aussi très bien dans des versions moins profondes : par exemple ResNet-34 est plus précis que VGG-16 sur ImageNet tout en étant plus rapide. Les réseaux de la famille ResNet sont très précis et performants, en faisant un choix très approprié pour plusieurs tâches. Ils ont démontré une excellente longévité et sont utilisés dans plusieurs méthodes de détection

d'objets [3, 15].

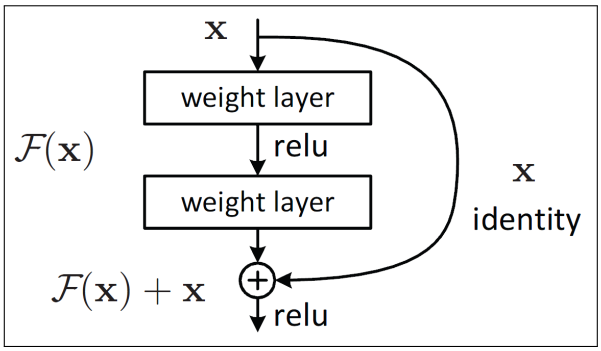


Figure 2.1 L'architecture d'un bloc résiduel [13]. © 2016 IEEE

Le réseau Hourglass introduit dans [14] utilise une architecture d'encodeur-décodeur à répétition, ce qui force le réseau à apprendre des représentations intermédiaires dans l'encodeur et donc à modéliser l'information nécessaire. Le réseau est construit en empilant plusieurs modules en forme de sablier (voir figure 2.2) les uns après les autres. Des connexions raccourcies sont ajoutées entre les blocs correspondants de chaque côté du sablier. Le réseau exécute une rétroaction intermédiaire entre chaque sablier afin de raffiner les représentations. Une implémentation particulière d'un réseau sablier pourrait par exemple empiler deux ou trois sabliers de suite. Il est particulièrement performant pour détecter les points clés, par exemple estimer la pose des humains ou dans les réseaux de détection d'objets par points clés [1].

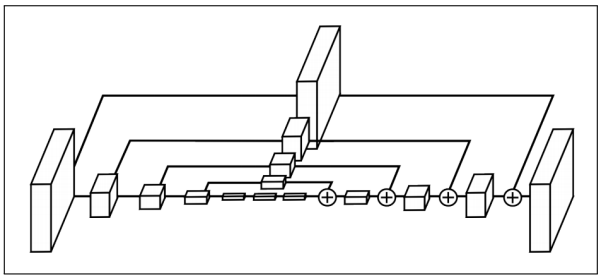


Figure 2.2 L'architecture d'un module sablier du réseau Hourglass [14]. © 2016 Springer International Publishing AG

2.2 Détection d'objets dans les images

Cette section fera un survol des principales méthodes de détections d'objets par apprentissage profond. Les méthodes classiques n'utilisant pas d'apprentissage profond n'ont pas été dominantes depuis 2014, nous les omettrons donc ici. Les méthodes de détection d'objets

sont communément divisées en deux catégories, les méthodes en deux phases et les méthodes en une phase. Les méthodes en deux phases utilisent une première phase de propositions d'objets candidats, et une seconde phase de raffinement où les meilleures propositions sont choisies ainsi que mieux positionnées et classifiées. Les méthodes en une phase quant à elle vont classifier et positionner les objets parmi toutes les possibilités directement sans utiliser de candidats. Plus récemment, une troisième catégorie utilisant les points clés a gagné en popularité. Ces méthodes sont techniquement considérées comme une phase, mais elles sont assez différentes des autres méthodes une phase pour être incluses dans leur propre sous-catégorie.

2.2.1 Méthodes en deux phases

La première méthode de détection d'objets utilisant l'apprentissage profond est R-CNN [28]. Cette méthode a eu un impact énorme, et a engendré deux travaux subséquents, Fast R-CNN [29] et Faster R-CNN [15]. R-CNN fonctionne en utilisant un proposeur de régions candidates externe, Selective Search [31]. Selective Search forme des régions candidates de façon récursive avec un algorithme glouton, premièrement en formant un ensemble de régions dans l'image, puis en combinant les deux régions les plus similaires, et en répétant ces étapes un certain nombre d'itérations. R-CNN va découper ces régions et va les déformer pour les mettre dans une taille standard, avant de les passer dans un réseau de neurones convolutif (CNN) pour en extraire un vecteur de caractéristiques. Ce vecteur de caractéristiques est ensuite classifié par une machine à vecteurs de support (SVM) pour en savoir la classe ainsi que la probabilité que ce soit un objet. La position exacte du rectangle englobant est aussi raffinée par un modèle de régression linéaire.

Fast R-CNN bâtit sur R-CNN en abordant son problème principal, sa lenteur. Plutôt que de passer chaque proposition d'objet dans un CNN, Fast R-CNN va plutôt passer l'image au complet une seule fois dans le CNN. Les régions d'intérêts candidates sont ensuite découpées directement dans la carte d'attributs et déformées pour devenir carrées. Chaque région carrée est ensuite passée dans la suite du CNN qui va classifier les régions et régresser leur taille et position précise.

Faster R-CNN améliore Fast R-CNN en supprimant le besoin d'utiliser un proposeur de régions candidates externe. En effet, Faster R-CNN introduit le Region Proposal Network (RPN), qui est un réseau de neurones convolutifs qui propose des régions d'intérêt ressemblant à des objets. Ce réseau utilise ce qu'on appelle les boîtes ancres, c'est-à-dire un ensemble de boîtes à différentes tailles et ratios d'aspect (voir figure 2.3). Ces boîtes sont appliquées sur une carte d'attributs à chaque position, et chaque combinaison de boîte et de position

sont évaluées selon leur probabilité d'englober un objet. La force de Faster R-CNN est que les paramètres du RPN sont partagés en grande partie par le réseau de classification et de régression (voir figure 2.4). Donc après avoir sélectionné les meilleurs candidats par le RPN, la suite est similaire à Fast R-CNN. Les candidats sont découpés dans la carte d'attributs, et chaque candidat est classifié et leur taille et position précise sont raffinées par un régresseur.

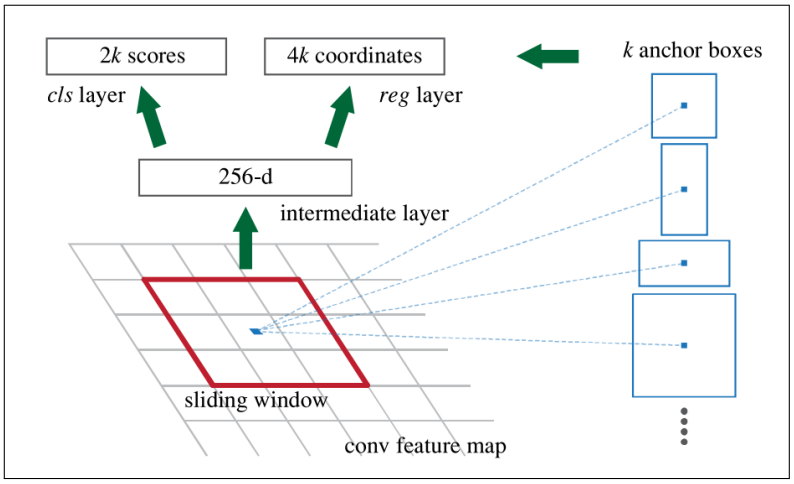


Figure 2.3 Fonctionnement des boîtes ancres [15]. © 2017 IEEE

R-FCN [32] propose une variante de Faster R-CNN pour en accélérer le traitement. Après avoir découpé les candidats, Faster R-CNN doit faire un traitement candidat par candidat afin de les classifier et d'en raffiner la position. R-FCN introduit une technique pour faire ce traitement une seule fois pour tous les candidats en produisant une carte de score sensible à la position pour chaque cellule d'une grille dont la taille est un hyperparamètre. Les candidats sont découpés dans cette carte de score et chaque cellule doit voter pour la catégorie de l'objet, ce qui produit une catégorie finale et un score. Les calculs faits pour chaque candidat de Faster R-CNN sont enlevés, car les scores sensibles à la position sont produits par le réseau pour toute l'image en même temps. De ce fait, R-FCN est aussi précis que Faster R-CNN tout en étant plus de deux fois plus rapide.

2.2.2 Méthodes en une phase

YOLO [16] est la première méthode réellement une phase et son objectif principal était clair, conserver de bonnes performances de détection tout en atteignant des vitesses de fonctionnement en temps réel. Cette méthode a engendré plusieurs successeurs [33–35]. YOLO redéfinit le problème de détection d'objets en une tâche de régression. Ce détecteur fonctionne en divisant l'image en une grille de $S \times S$ (voir figure 2.5). Chaque cellule de la grille est responsable

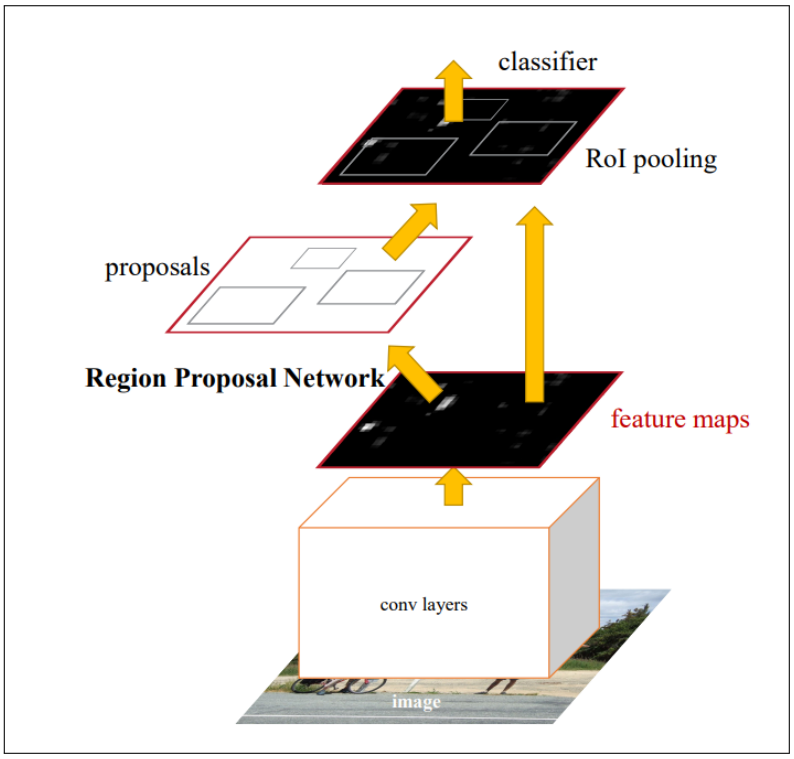


Figure 2.4 Architecture de Faster R-CNN [15]. © 2017 IEEE

de détecter un certain nombre d'objets proche d'elle. À cause de ce choix, YOLO a du mal à détecter une grande densité de petits objets. L'image est donnée en entrée à un CNN et le réseau est entraîné par descente stochastique du gradient. La sortie du réseau est $S \times S$ cellules $\times$ K objets par cellule $+x+y+largeur+hauteur+score$ $+C$ probabilité de classes. YOLO9000 et YOLOv3 améliorent YOLO en ajoutant l'utilisation de boîtes ancres comme dans Faster R-CNN et en améliorant le CNN utilisé. YOLOv4 [35] propose une version encore plus optimisée en intégrant plusieurs stratégies d'entraînement de l'état-de-l'art.

SSD [30] est une méthode similaire à YOLO lui apportant quelques améliorations. Un des points faibles de YOLO est la détection des petits objets, en particulier les petits objets présents en grand nombre. Pour pallier cette difficulté, SSD introduit des têtes de classification et localisation à plusieurs résolutions dans le CNN. Plus précisément, des cartes d'attributs de différentes couches du réseau à différentes profondeurs sont utilisées comme entrées de sous-réseaux de classification et de régression. Ceci facilite la détection des objets à plusieurs échelles différentes. Tout comme Faster R-CNN, SSD utilise les boîtes ancres afin de détecter les objets. RetinaNet [3] est une méthode similaire à SSD qui aborde deux de ses problèmes principaux. Premièrement, RetinaNet introduit la fonction de perte focale, qui ré-équilibre les exemples faciles et les exemples difficiles durant l'entraînement. Comme les méthodes

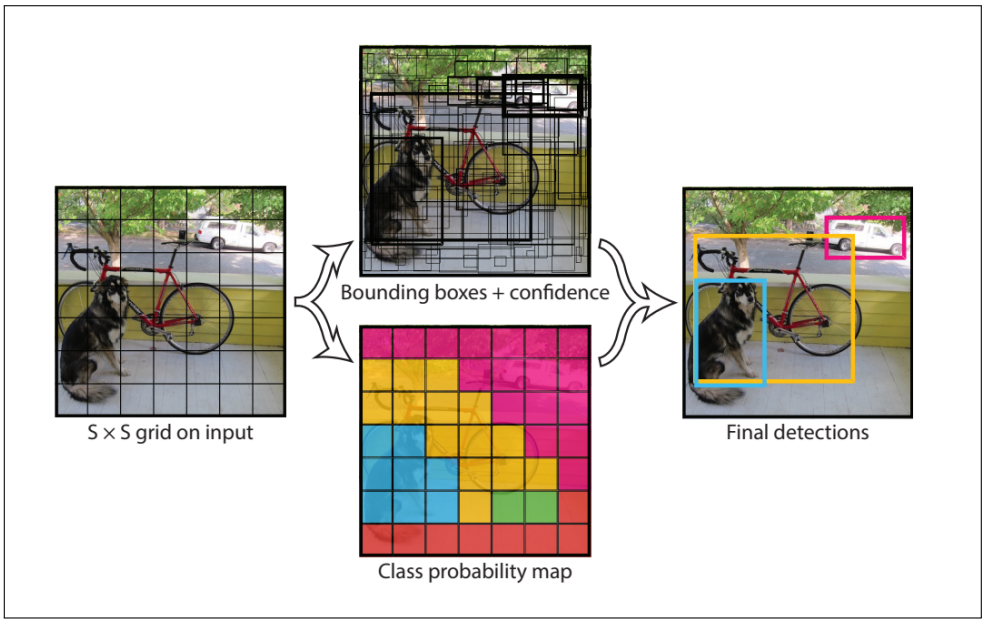


Figure 2.5 Architecture de YOLO [16]. © 2016 IEEE

en une phase n'utilisent pas de proposition d'objets, la grande majorité des boîtes testées sont des négatifs faciles, ce qui laisse moins d'importance pour les exemples positifs dans la fonction de perte. La perte focale met plus d'importance sur les exemples difficiles, l'entraînement peut donc faire le focus sur ce qui est important, d'où le nom. L'autre contribution de RetinaNet est l'utilisation d'un réseau en pyramide (FPN) afin de faire de la détection d'objets à plusieurs échelles. Un FPN fonctionne en combinant l'information sémantique riche des couches profondes d'un CNN avec l'information spatiale des couches moins profondes. Afin de créer la pyramide, les couches profondes sont une à une combinée avec les couches moins profondes à l'aide d'une interpolation de la carte d'attributs suivi d'une addition et d'une convolution. Ces nouvelles cartes d'attributs créées du haut vers le bas sont donc à la fois riches en informations sémantiques et spatiales. Des sous-réseaux de classification et de localisation sont ajoutés à chaque nouvelle couche ainsi créée afin de détecter les objets à diverses échelles (voir figure 2.6).

2.2.3 Méthode par détection de points clés

Bien que techniquement des méthodes en une phase, les méthodes par détection de points clés sont assez différentes pour être dans leur propre catégorie. Plutôt que d'utiliser des boîtes ancres comme la plupart des méthodes une phase et deux phases, ces méthodes vont plutôt détecter la présence d'objets en détectant des points clés présents caractéristiques aux objets. CornerNet [36], comme son nom l'indique, détecte les objets en détectant leurs coins supérieur

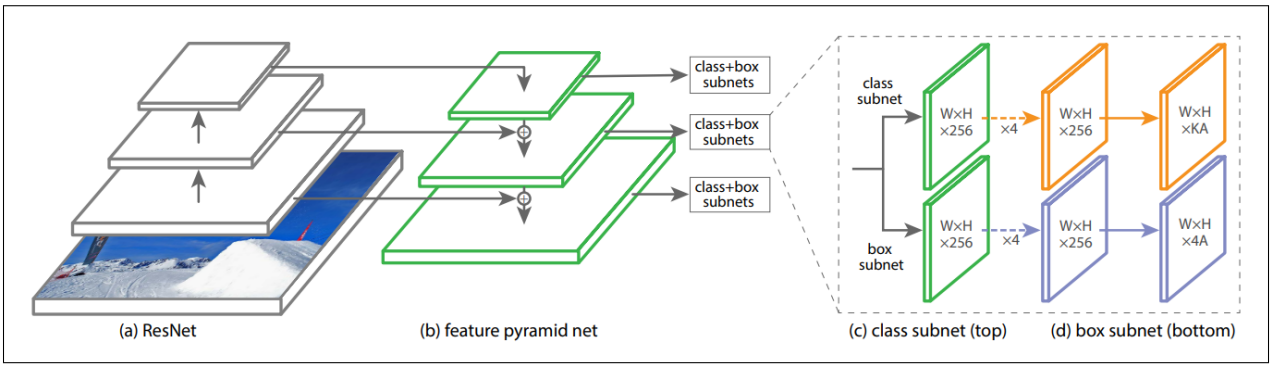


Figure 2.6 Architecture de RetinaNet [3]. © 2017 IEEE

gauche et inférieur droit. La méthode va créer quatre cartes de score, une pour la probabilité de chaque pixel de faire partie d'une bordure du haut, une pour la bordure de gauche, etc. On va ensuite combiner ces scores de façon à créer un score de probabilité d'être un coin supérieur gauche et un autre pour un coin inférieur droit. En parallèle, la méthode va aussi entraîner un vecteur de réidentification de façon qu'il soit similaire si deux coins font partis du même objet et différent sinon. Les coins sont donc regroupés à l'aide de ce vecteur et les objets sont ainsi détectés. Keypoint triplets [17] va améliorer CornerNet en ajoutant une carte de probabilité d'être le centre d'un objet. Ils vont aussi améliorer la façon de créer les cartes de coins. Le score de centre est utilisé afin de supprimer les faux positifs, par exemple le visage de la fillette dans la figure 2.7. C'est-à-dire que si un objet détecté n'a pas de haute probabilité d'avoir le centre d'un objet en son centre, il sera rejeté.

Avec une approche plus simple, CenterNet [1] fonctionne en détectant directement le centre des objets en créant des cartes de probabilité pour chaque catégorie d'objets. Les valeurs conservées sont celles qui sont plus grandes que leurs huit voisins connectés. Une tête de régression du réseau retourne la largeur et la hauteur des objets à chaque position dans l'image, si cette position est le centre d'un objet. Récemment, DETR [37] effectue la détection d'objets à l'aide d'un réseau de neurones à auto-attention. Étant donné un ensemble de requêtes apprises de taille fixe, DETR apprend à produire directement un ensemble de détections en parallèle, avec une seule passe dans le réseau. Dans ce travail, l'utilisation d'une fonction de coût considérant l'ensemble des détections force le réseau à produire des détections différentes les unes des autres, entre autres en faisant un appariement optimal entre les boîtes détectées et la vérité terrain. DETR est la première méthode utilisant un réseau de neurones à auto-attention pour produire des détections d'objets, et son architecture est particulièrement simple et dépourvue d'algorithmes faits à la main. En ce sens, c'est en quelque sorte une preuve de concept.

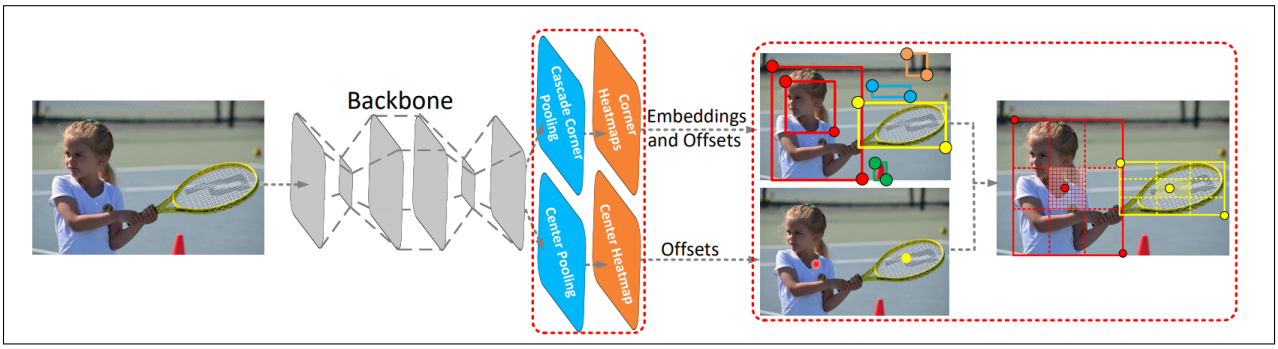


Figure 2.7 Architecture de Keypoints Triplets [17]. © 2019 IEEE

2.2.4 Évaluation de la performance

Afin d'évaluer la performance d'un détecteur d'objets, la *mean average precision* (mAP) est utilisée. Ce concept est particulièrement utile, car il combine la précision et le rappel. Afin de calculer le mAP, il faut choisir un IOU minimum afin de déclarer une détection valide. Typiquement, les jeux de données vont utiliser 0.7 ou 0.5. Pour calculer le mAP, il faut faire la moyenne de la précision en faisant varier le rappel de 0 à 1, par intervalle régulier, comme on peut le voir sur la figure 2.8. Cela est équivalent à calculer l'aire sous la courbe de la fonction précision sur rappel. Le mAP est calculé en utilisant tous les objets annotés de toutes les images sur l'ensemble de test. Nous utiliserons alors la notation mAP si le IOU minimum est préalablement défini, ou AP[minimum IOU] (pour *average precision*), par exemple AP[0.5] si nous avons plusieurs IOU minimum dans le même tableau, afin de les distinguer. Une autre notation utilisée est AP[IOU départ :IOU fin], par exemple AP[0.5 :0.95]. Cette notation signifie une moyenne des mAPs à différents IOU minimum, avec un intervalle défini, par exemple 0.05. Le AP[0.5 :0.95] serait donc la moyenne des AP[0.5], AP[0.55], ..., AP[0.9], AP[0.95].

Afin de s'assurer une évaluation équitable, les jeux de données sont toujours divisés en trois sous-ensembles, l'ensemble d'entraînement, de validation et de test. L'ensemble d'entraînement est utilisé pour entraîner, et l'ensemble de validation est utilisé pour monitorer la performance durant l'entraînement afin de ne pas surentraîner ou sous-entraîner un modèle. Durant cette phase, l'ensemble de validation est utilisé pour optimiser les hyperparamètres également. Une fois que l'entraînement est terminé, la performance peut être évaluée sur l'ensemble de test. Afin d'éviter la tricherie ou le surentraînement, plusieurs jeux de données ne permettent pas d'évaluer l'ensemble de test par soi-même, il faut envoyer nos données sur le serveur du jeu de données pour le faire.

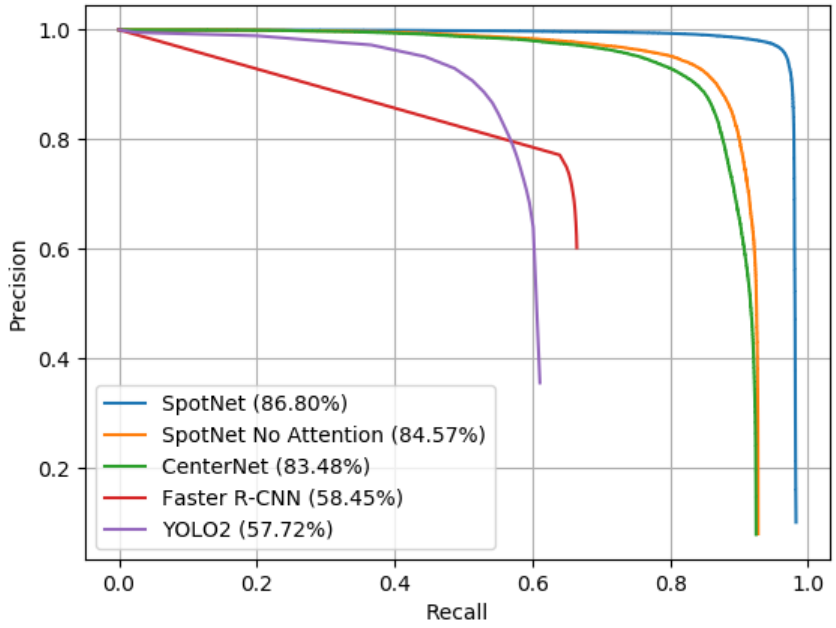


Figure 2.8 Un exemple de courbe représentant le mAP avec quelques méthodes [2]. © 2020 IEEE

2.3 Détection d'objets dans les vidéos

Les méthodes de détection d'objets sur vidéos sont basées sur les méthodes de détection d'objets sur les images, en y ajoutant des aspects de l'information temporelle.

Une première façon d'effectuer la détection d'objets vidéo consiste à combiner les cartes d'attributs de plusieurs images. L'agrégation de caractéristiques guidées par flux optique (FGFA) [18] utilise la déformation de cartes d'attributs par le flux optique pour fusionner des images proches dans le temps afin d'améliorer la précision (voir figure 2.9). FGFA utilise FlowNet [38] pour estimer le flux optique entre les trames. L'utilité de cette approche est de pouvoir agréger des cartes d'attributs temporellement proches, réalignées à l'aide du flux optique, dans des cas où l'objet sur l'image cible serait partiellement occlus ou flou comme dans la figure 2.9. Dans MANet [39], une estimation de flux optique est également effectuée et deux réseaux sont entraînés, un pour faire un calibrage au niveau des pixels (ajustements de mouvement détaillés) et un autre pour un calibrage au niveau des instances (ajustements de mouvement global). Les trames calibrées sont par la suite combinées en utilisant un module de combinaison de cartes d'attributs.

Certains travaux tirent parti des réseaux de neurones récurrents, par exemple STMN [40]

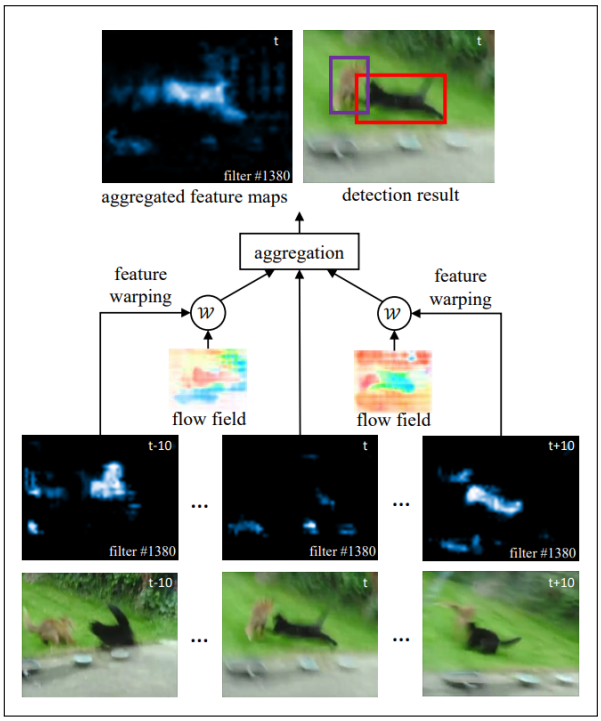


Figure 2.9 Architecture de FGFA [18]. © 2017 IEEE

qui modélise le mouvement et l'apparence d'un objet dans une séquence vidéo. Dans cette méthode, afin de détecter dans une trame cible, une fenêtre de K trames autour sont utilisées. Chaque trame est donnée en entrée à un CNN, et les cartes d'attributs sont liées via un module de mémoire spatio-temporelle. Le module de mémoire spatio-temporelle est un réseau de neurones récurrent qui lie les cartes d'attributs une par une. Plus précisément, les cartes d'attributs sont produites pour toutes les trames, puis elles sont données en entrée à un réseau récurrent. Il fonctionne dans deux directions différentes pour les trames passées et futures, c'est-à-dire que l'information se dirige vers le centre, qui est la trame sur laquelle on veut détecter. Une fois au centre, la carte d'attributs résultante est utilisée pour faire la détection de façon similaire à Faster R-CNN, qui est leur réseau de base. Dans LSTM-SSD [19], les réseaux de neurones récurrents à longue mémoire à court terme (LSTM) sont utilisés pour interpoler les cartes d'attributs (voir figure 2.10), ce qui augmente considérablement la vitesse d'inférence. Multi-Frame SSD [41] s'appuie sur SSD en ajoutant une information temporelle avec un module convolutif récurrent. Plus précisément, un ensemble de K trames précédant une trame cible sont utilisées et données en entrée à un CNN. Les cartes d'attributs résultantes sont passées dans un réseau de neurones récurrent à portes, de la plus ancienne jusqu'à la trame cible. La carte d'attributs résultante enrichie par l'information temporelle est utilisée afin d'effectuer la détection sur la trame cible. Bertasius *et al.* [42] utilise des convolutions

déformables pour calculer les décalages entre des trames. Ils vont, eux aussi, utiliser une fenêtre de K trames temporellement proche à une trame cible. La carte d'attributs de la trame cible est concaténée avec celle des trames de support et est donnée en entrée à un sous réseau composé de plusieurs couches de convolutions déformables. Ce sous-réseau va calculer un décalage entre les pixels de la trame de support et la trame cible et va appliquer ce décalage afin d'aligner la trame de support avec la trame cible. Les cartes d'attributs ainsi alignées vont être combinées pixel par pixel par une combinaison linéaire et le résultat sera utilisé pour effectuer la détection. 3D-DETNet [6] utilise des convolutions 3D sur des cadres temporellement proches qui sont concaténés afin de produire de meilleures cartes d'attributs.

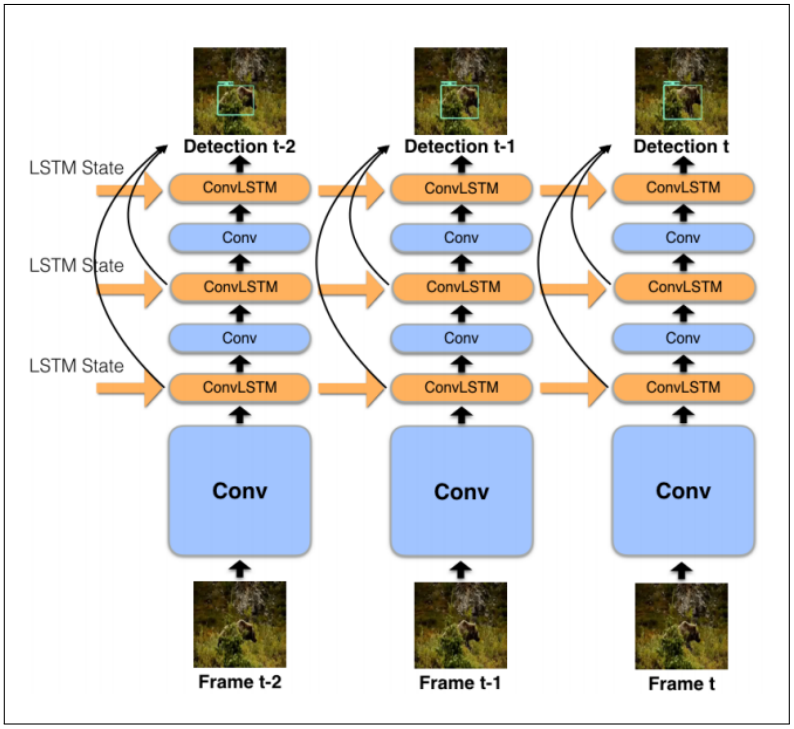


Figure 2.10 Architecture de LSTM-SSD [19]. © 2018 IEEE

Une autre piste possible est de combiner détection et suivi. TrackNet [43] étend la méthode Faster R-CNN en détectant directement un cube 3D délimitant un objet en mouvement. Un réseau à deux branches est premièrement utilisé et une séquence vidéo y est donnée en entrée pour produire des attributs visuels d'un côté avec un CNN standard et des attributs spatio-temporels de l'autre en utilisant des convolutions 3D. Les attributs des deux branches sont concaténés et sont donnés en entrée dans un réseau de neurones à auto-attention afin de normaliser l'angle de vue des attributs des trames de la séquence vidéo. Un réseau de proposition de tubes, des rectangles englobants empilés sur plusieurs trames consécutives, est utilisé par la suite et les tubes candidats sont raffinés et classifiés. La sortie du réseau est

donc des tubes englobants pour suivre les objets sur plusieurs trames. Dans l'article "Joint Detection and Tracking in Videos with Identification Features" [44], les auteurs forment un modèle multitâche par optimisation conjointe de la détection, du suivi et de la réidentification. Un réseau de base prend deux trames en entrée et sort des détections avec leur vecteur de réidentification correspondant. Parallèlement, une autre branche calcule une régression des boîtes entre les trames. Le vecteur de réidentification est utilisé pour récupérer les détections perdues à cause d'occlusion ou autre raison. L'entraînement multitâche est aussi bénéfique à la performance générale du détecteur. Le réseau global de corrélation [45] entraîne également conjointement la détection et la tâche de suivi en entraînant d'abord le module de détection avant de raffiner l'ensemble du réseau. Le module de suivi prend en entrée la carte d'attributs de la trame courante, les détections de la trame courante ainsi que les attributs des trajectoires jusqu'à l'instant présent, et sort en résultat une détection et un score pour chaque trajectoire.

Enfin, les informations de mouvement peuvent être intégrées au réseau. Wang *et al.* [46] intègre le mouvement dans un réseau afin de mettre l'accent sur les véhicules et de supprimer les faux positifs. Pour ne pas affecter la détection des véhicules statiques, ils proposent un réseau avec une branche standard qui prend en entrée l'image et une branche de détection du mouvement qui prend en entrée des attributs de l'image ainsi que des attributs d'un masque binaire de mouvement. Les branches sont finalement agrégées pour en combiner les attributs. Pour mieux détecter les petits objets, MMA [47] propose un réseau à double flux, un flux d'apparence et un flux de mouvement. Ils intègrent également un module d'attention de la mémoire pour aider à sélectionner des attributs discriminants à l'aide des informations temporelles. MFMNet [48] utilise un mouvement du module de mémoire pour encoder le contexte temporel. Ce module contient une mémoire dynamique distincte pour chaque séquence d'entrée et produit des fonctions de mouvement pour chaque image.

2.3.1 Évaluation de la performance

L'évaluation de la performance des détecteurs d'objets sur vidéos est exactement la même que celle des détecteurs d'objets sur des images. La seule différence réside dans les jeux de données utilisés, les détecteurs d'objets sur vidéos étant contraint à utiliser des séquences.

2.4 Segmentation d'instances

La progression naturelle de la détection d'un rectangle englobant est une segmentation fine pixel par pixel. Mask R-CNN [20] s'appuie sur Faster R-CNN en ajoutant une branche de masque invariante à la taille dans la deuxième phase (voir figure 2.11), en parallèle de la

branche de détection de rectangle englobant. Pour chaque objet candidat, il produit un masque pour chaque catégorie possible à l'aide d'une série de convolutions. La même amélioration existe pour RetinaNet, nommée RetinaMask [49]. Dans RetinaMask, un sous-réseau de masque est ajouté pour produire une segmentation pour chaque objet candidat. Les niveaux de la pyramide de cartes d'attributs vont être responsables de produire un masque pour les objets de tailles correspondantes à leur niveau. PANet [50] produit des masques précis en ajoutant un chemin du bas vers le haut à un FPN ainsi qu'une sélection adaptative des attributs. Le réseau commence par produire une pyramide des cartes d'attributs similairement à RetinaNet. Par la suite, une deuxième pyramide est produite dans l'autre sens, en débutant par la couche moins profonde et en construisant la pyramide couche par couche. Les niveaux de la deuxième pyramide sont alors mis en commun en utilisant une opération de maximum, et la carte d'attributs résultante est utilisée pour faire la classification, détection et la production de masques.

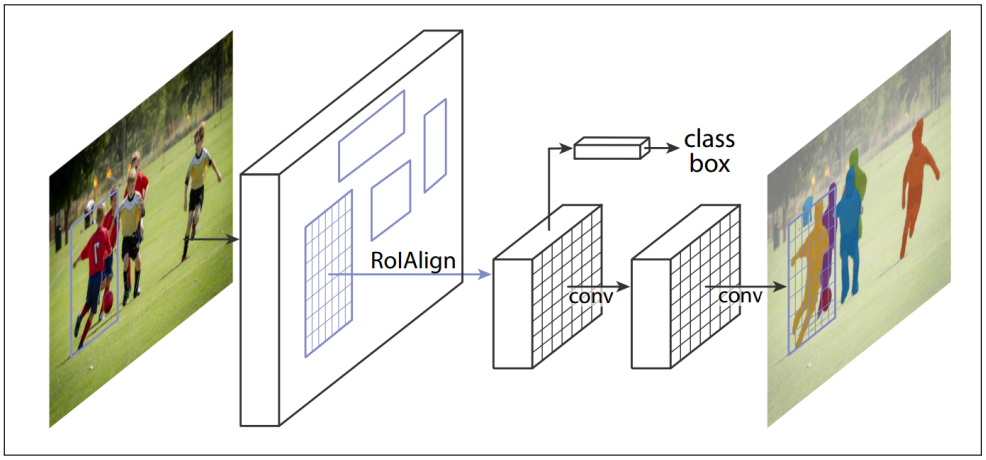


Figure 2.11 Architecture de Mask R-CNN [20]. © 2017 IEEE

Certaines méthodes effectuent une segmentation d'instance en calculant un polygone englobant autour des objets. PolarMask [21] et Poly-YOLO [51] utilisent une grille polaire pour représenter les sommets par leurs angles à partir du centre (voir figure 2.12), et régressent les coordonnées des sommets. La principale différence entre eux est que Poly-YOLO apprend les formes invariantes de taille en utilisant une distance normalisée, alors que PolarMask utilise des distances absolues. Poly-YOLO améliore également l'architecture de YOLO afin de produire de meilleures boîtes englobantes, en plus des polygones. Polygon-RNN [52] présente une architecture CNN et RNN pour annoter rapidement des polygones. À chaque itération, le RNN reçoit la représentation de l'image produite par VGG ainsi que la position des deux sommets précédents et le réseau régresse la position du prochain sommet. Un signal de fin de polygone peut aussi être produit. Polygon-RNN++ [53] introduit plusieurs améliorations,

entre autre une meilleure représentation par le CNN, un entraînement du modèle par apprentissage par renforcement ainsi qu'un mode automatique qui ne requiert pas d'humain, mais qui utilise Faster R-CNN pour obtenir les rectangles englobants.

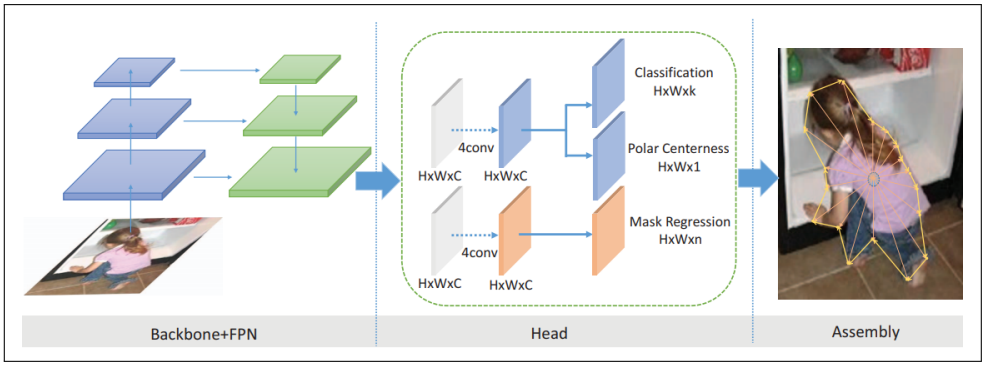


Figure 2.12 Architecture de PolarMask [21]. © 2020 IEEE

Quelques méthodes peuvent produire une segmentation d'instances en temps réel, y compris Poly-YOLO, mais ces travaux restent rares. La plupart des méthodes de segmentation d'instances sont beaucoup plus lentes que leur équivalent de détection de boîtes englobantes. Box2Pix [54] peut produire des résultats sur cityscapes à 10,9 fps en combinant la détection de boîtes englobantes ainsi que la segmentation sémantique et en prédisant le décalage des pixels par rapport aux centres des objets. La segmentation sémantique étant généralement beaucoup plus rapide que la segmentation d'instances, la méthode produit une segmentation sémantique parallèlement aux rectangles englobants. Additionnellement, une carte de décalage de chaque pixel par rapport au centre de l'objet dont il fait partie est entraîné. Grâce à cette carte, on peut grouper les masques et ainsi produire une segmentation d'instances. La méthode de Mazzini *et al.* [55] peut réaliser une segmentation d'instance très rapide sur cityscapes à 113 fps en utilisant un réseau encodeur sans décodeur et en produisant une carte de décalage des objets vers leur centre pour regrouper les instances. La grande vitesse vient du fait comme le réseau fonctionne sans décodeur, la segmentation est faite en très basse résolution. Un petit module produit le suréchantillonnage à la fin. Enfin, Yolact [56] fonctionne en produisant quelques prototypes de segmentation et entraîne un réseau pour apprendre le coefficient approprié pour les combiner. En utilisant un petit réseau nommé Protonet composé de plusieurs couches de convolutions, le réseau produit K masques appelés prototypes. Une autre branche du réseau apprend à produire des coefficients qui serviront à additionner ou soustraire les prototypes entre eux pour produire un masque de segmentation différent pour chaque détection. Ils utilisent ensuite la détection de rectangles englobants pour découper les régions dans les prototypes combinés afin de produire les segmentations d'instances.

2.4.1 Évaluation de la performance

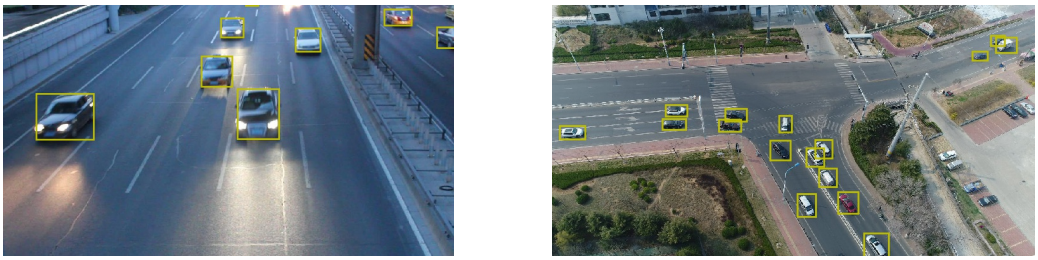
L'évaluation de la performance pour la tâche de segmentation d'instance est la même que pour la détection d'objets, la différence étant que l'IOU est fait en comparant des masques de segmentation pour chaque instance, et non des rectangles englobants. Pour ce qui est de la détection d'objets sur vidéos, l'évaluation est exactement la même que pour la détection d'objets, comme on ne parle pas ici de suivi, la capacité de suivre une instance sur plusieurs trames n'est pas évaluée.

2.5 Jeux de données

Dans cette section, nous parlerons brièvement des jeux de données utilisés dans ce travail.

UA-DETRAC [5] est un jeu de données pour la détection d'objets et le suivi multiobjets. Ce jeu de données est composé de séquences vidéo de scènes de trafics prises par des caméras fixes au-dessus de certaines routes en Chine (voir figure 2.13(a)). La résolution des images (environs 70000) est de 960×540. Les annotations d'UA-DETRAC sont composées de quatre catégories différentes ("bus", "car", "other", "van"), mais leur évaluation ne considère pas les catégories.

UAVDT [7] est un jeu de données pour la détection d'objets et le suivi multiobjets capturé par drones. Ce jeu de données est particulièrement difficile par la petite taille et la densité des objets (voir figure 2.13(b)). En contrepartie, cela fait en sorte qu'il y a moins d'occlusions que dans UA-DETRAC. Comme les images sont filmées par des drones, l'arrière-plan des séquences est mobile. Les annotations comportent trois catégories différentes ("car", 'truck", "bus"). La résolution des images est de 1080×540.



(a) (b)

Figure 2.13 (a) Exemple d'image de UA-DETRAC [5] et ses annotations. (b) Exemple d'image de UAVDT [7] et ses annotations.

Cityscapes [12] est un jeu de données pour la segmentation sémantique, d'instances et panoptique de scènes de trafic capturées dans plusieurs villes d'Allemagne. La résolution

d'image est de 2048 × 1024. Pour la segmentation d'instances, un total de huit catégories d'instance est défini, qui comprend les personnes, les cyclistes, les voitures, les camions, les autobus, les trains, les motocyclettes ainsi que les vélos. Plus de catégories sont définies pour la segmentation sémantique et panoptique (voir figure 2.14).



(a) (b)

Figure 2.14 Deux exemples d'annotations sur le jeu de données cityscapes [12].

KITTI [57] est un regroupement de jeux de données pour diverses tâches. L'ensemble pour la segmentation d'instances est un petit ensemble de données qui se compose de 200 images d'entraînement et 200 images de test de scènes de trafic avec les mêmes catégories que cityscapes. La résolution de l'image est d'environ 1280 × 384, mais peut très légèrement varier.

India Driving [9] est un jeu de données pour la segmentation sémantique et d'instances, composée d'images de scènes de trafic capturées en Inde. Les catégories annotées pour la segmentation d'instances sont "person", "rider", "motorcycle", "bicycle", "autorickshaw", "car", "truck", "bus" et "vehicle fallback", cette dernière catégorie étant pour un véhicule ne correspondant pas aux autres catégories. La résolution des images varie entre 1920 × 1080 et 1280×964. La répartition des ensembles d'entraînement / validation / test est respectivement d'environ 7000/1000/2000 images.

Les jeux de données ont été choisis pour répondre aux objectifs de cette thèse, à savoir détecter et segmenter les usagers de la route. Il est donc normal que tous les jeux de données touchent au domaine routier. De plus, d'autres facteurs justifient leurs choix. UA-DETRAC et UAVDT contiennent des séquences vidéos, ce qui était nécessaire dans ce travail. Leurs différences principales sont que UAVDT contient des objets beaucoup plus petits et d'une plus grande densité que UA-DETRAC, et que le point de vue est mobile pour UAVDT et fixe pour UA-DETRAC. Ces deux facteurs rendent UAVDT plus difficile que UA-DETRAC. Cityscapes, KITTI et IDD sont des jeux de données populaires de segmentation d'instances, le choix de leur utilisation réside dans le fait qu'ils sont dans le domaine des usagers de la route.

CHAPITRE 3 SURVOL DES ARTICLES

Cette thèse suit une structure de thèse par articles afin de présenter les travaux réalisés durant mon doctorat. Chacun des quatre chapitres suivants est un article soit publié ou soumis dans un journal ou un congrès. Les articles répondent aux objectifs fixés, et présentent des solutions adéquates pour plusieurs des applications de la détection et segmentation des usagers de la route dans des images et des vidéos.

J'ai débuté mon doctorat en travaillant sur la combinaison de trames consécutives pour la détection d'objets. Pour ce faire, j'ai entraîné des réseaux à produire des cartes d'attributs prenant en entrée des paires d'images, ainsi que prenant des images de flux optique en entrée (voir figure 3.1). Ce travail, présenté dans un rapport technique [22], démontre que prendre une paire d'images en entrée peut améliorer les performances d'un réseau. Toutefois, prendre une combinaison d'une image de flux optique avec l'image cible n'améliore pas nécessairement. Cela est peut-être dû à la présence d'objets stationnaires. Une des difficultés rencontrées avec ce travail est que comme nous changions l'entrée du CNN, l'utilisation des poids pré-entraînés sur les gros jeux de données était impossible. De ce fait, même si nous montrions une amélioration par rapport à une méthode de base, les résultats par rapport à d'autres méthodes n'étaient pas particulièrement impressionnants. Bien que n'étant pas un chapitre de ma thèse, ce travail a tout de même été important dans mon doctorat, car il est la base du travail suivant.

Le chapitre 4 présente une méthode de détection d'objets sur des images basée sur la détection de points clés. Ce travail présente une adaptation de CenterNet [1] qui ajoute un module d'attention entraîné sur des annotations partielles. CenterNet détecte les objets en produisant des cartes de probabilité d'être le centre des objets pour chaque classe. Parallèlement, une autre tête du réseau produit une largeur, une hauteur ainsi qu'un décalage pour chaque objet. Le module d'attention que nous ajoutons est entraîné par soustraction d'arrière-plan et flux optique et tente de guider le réseau vers les zones saillantes où il a plus de chance de trouver des objets d'intérêts. En raison de cette carte d'attention qui illumine la carte d'attributs, nous nommons notre méthode SpotNet. SpotNet [2] utilise la carte d'attention produite à la fois pour améliorer la détection de rectangles englobants (avec un processus d'attention) et pour produire des segmentations avec une binarisation de la carte. Le processus d'attention que nous utilisons est une multiplication de notre carte de saillance avec la carte d'attributs servant à produire les cartes de probabilités. Ce travail se concentre à enrichir les cartes d'attributs grâce à un processus d'auto-attention, contrairement au chapitre 5 qui améliore

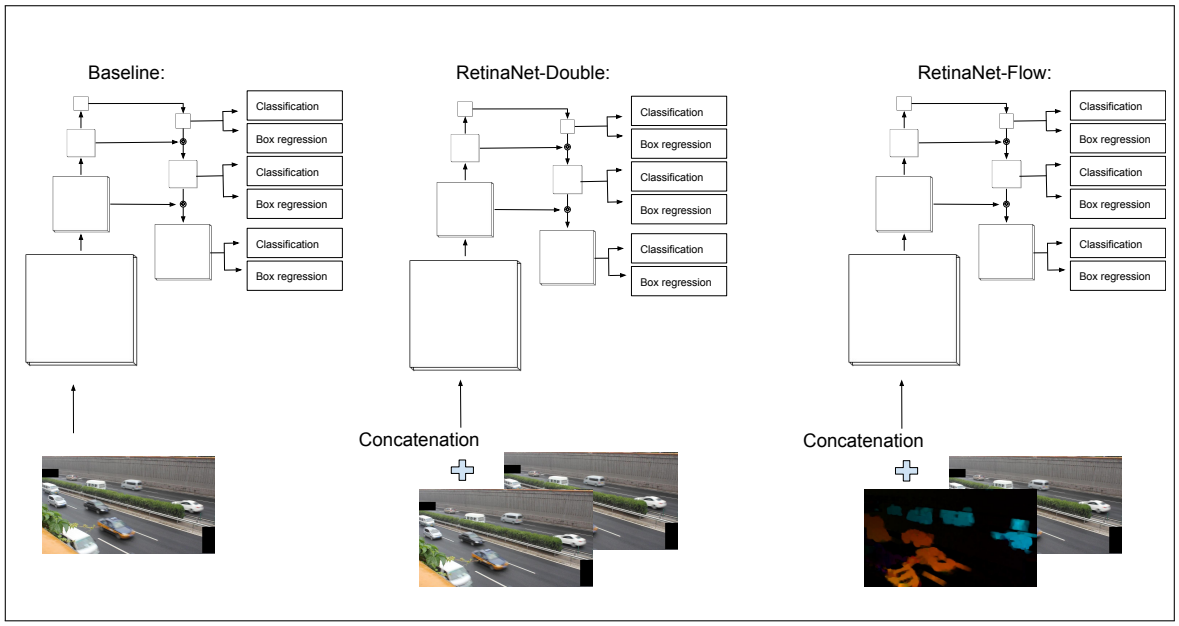


Figure 3.1 Un réseau de détection de base, un prenant deux images consécutives concaténées et un prenant une image et une image de flux optique concaténées. [22]

les cartes d'attributs en combinant l'information de plusieurs trames. Nous verrons dans le prochain chapitre que ces deux façons de faire peuvent être bénéfique l'une à l'autre si on les combine.

Le chapitre 5 présente une méthode de détection d'objets sur vidéo. À la suite de réflexions après avoir complété notre premier rapport technique, nous en sommes venus à la conclusion qu'il fallait combiner les trames consécutives plus profondément dans le réseau afin de tirer profit des poids pré-entraînés. Notre approche pour cette méthode est de combiner les cartes d'attributs de plusieurs trames consécutives autour d'une trame cible. De cette façon, au fur et à mesure que l'on détecte dans la vidéo, la fenêtre des trames utilisées se déplace et nous n'avons pas besoins de recalculer les cartes d'attributs pour les trames restantes dans la fenêtre autour de la trame cible. Nous avons créé un module de fusion afin de combiner les cartes d'attributs en concaténant les canaux correspondant puis en réduisant la résolution à l'aide de convolution 1×1, suivi de réassemblage des canaux dans leur ordre original. Nous avons basé notre méthode sur un très bon détecteur, RetinaNet [3], d'où le nom de notre méthode, RN-VID [4], qui signifie RetinaNet-Vidéo.

FFAVOD, présenté dans le chapitre 6 est une combinaison et extension des deux chapitres précédents. Premièrement, nous intégrons l'architecture de RN-VID et le module de fusion dans deux nouveaux détecteurs modernes et démontrons une amélioration significative. Deuxièmement, nous améliorons l'étude d'ablation et comparons avec d'autres stratégies de fusion.

Finalement, nous introduisons une amélioration au module d'attention de SpotNet qui améliore les résultats du SpotNet de base. Grâce à ces améliorations, nous atteignons l'état de l'art sur deux bases de données de séquences vidéo de trafic routier, UA-DETRAC [5] et UAVDT [7].

Le dernier article présenté dans le chapitre 7 se concentre sur la tâche qui est la progression naturelle de la détection des rectangles englobants, la segmentation d'instances. La segmentation permet d'améliorer la détection de diverses façons. Entre autre, elle permet de capturer l'apparence de l'objet d'intérêt beaucoup plus précisément, ce qui peut aider à le classifier ou à modéliser son apparence pour des applications de suivi. Entraîner la tâche de segmentation parallèlement à la tâche de détection peut aussi aider le réseau à apprendre des attributs plus généraux ou permettant de localiser les objets plus précisément. SpotNet permet d'obtenir une segmentation de l'avant-plan d'une image. Toutefois, une segmentation d'instances est plus utile, car elle permet de différencier l'identité des objets dans un contexte de chevauchement. Dans ce travail, dans l'idée de rester utile à des applications dans le domaine du transport, nous avons décidé d'essayer de produire une méthode rapide, fonctionnant en temps réel. Dans cet esprit, CenterPoly est une méthode de segmentation d'instances par polygones englobants, ce type de méthodes étant généralement plus rapide que leur contrepartie de segmentation par masque. CenterPoly fonctionne en effectuant la détection des points centraux des objets parallèlement à la segmentation par polygone, en produisant un volume en sortie contenant les sommets des polygones pour chaque position dans l'image. Les polygones retenus sont ceux dont un point central se trouve à leur position. En plus des polygones, une tête de régression va estimer une profondeur relative pour chaque objet dans l'image. La profondeur relative signifie la profondeur d'un objet relativement aux objets avec lesquels il est en chevauchement. Cette profondeur sert à placer les polygones les uns par-dessus les autres dans les scènes de trafics denses où plusieurs voitures ou piétons se chevauchent.

Le code pour tous les articles se trouve au lien suivant : https://github.com/hu64.

CHAPITRE 4 ARTICLE 1 : SPOTNET : SELF-ATTENTION MULTI-TASK NETWORK FOR OBJECT DETECTION

Perreault H., Bilodeau GA., Saunier N., Heritier M.
17th Conference on Computer and Robot Vision (CRV), 2020, pp. 230-237
(© 2020 IEEE ; Reprinted with permission.)
doi : 10.1109/CRV50864.2020.00038

Abstract

Humans are very good at directing their visual attention toward relevant areas when they search for different types of objects. For instance, when we search for cars, we will look at the streets, not at the top of buildings. The motivation of this paper is to train a network to do the same via a multi-task learning approach. To train visual attention, we produce foreground/background segmentation labels in a semi-supervised [1] way, using background subtraction or optical flow. Using these labels, we train an object detection model to produce foreground/background segmentation maps as well as bounding boxes while sharing most model parameters. We use those segmentation maps inside the network as a self-attention mechanism to weight the feature map used to produce the bounding boxes, decreasing the signal of non-relevant areas. We show that by using this method, we obtain a significant mAP improvement on two traffic surveillance datasets, with state-of-the-art results on both UA-DETRAC and UAVDT.

4.1 Introduction

There is increasing interest in automatic road user detection for intelligent transportation systems, advanced driver assistance systems, traffic surveillance, etc. Road user detection has its own set of challenges and difficulties, such as the high speed of some road users, the frequent occlusion between them and the small size of road users appearing afar. Despite huge improvements in the last years thanks to advancements in deep learning-based methods, results still need to be improved for reliable practical applications.

Recently, a new family of object detectors were proposed based on keypoint detection rather than based on bounding box classification [1, 17, 36]. This approach presents several advan-

1. Weak annotations rather. This applies to all further mentions of semi-supervised in this paper.

tages, including not having to manually design anchor boxes and having to process fewer candidate boxes. Detecting objects in this way is deceptively simple and elegant, and quite fast. It yields state-of-the-art accuracy results on several datasets. Therefore, in this work, we build upon CenterNet [1] by designing a novel convolutional neural network (CNN) model that directs its attention towards the areas of interest and thus decreases the probability of having false detections in incongruous areas.

Our contributions are : 1) a self-attention mechanism based on multi-task learning (object detection and segmentation) and 2) a semi-supervised training method that capitalizes on automatic foreground/background segmentation annotations.

The idea of attention and self-attention has been around for some time now, most notably in image captioning [58] and natural language processing (NLP) [59]. In those works, neural networks are trained to learn which parts of the input are the most important to solve the task. But they do so progressively, using recurrent neural networks. Can a simple CNN learn which areas it should use to increase its visual attention ? In this work, we show that it is indeed possible and beneficial for object detection by using a semi-supervised training approach and multi-task learning. The network is trained for both object detection and foreground/background segmentation, the latter being also used to weight object detection feature maps. Indeed, the foreground/background segmentation is used in an internal attention mechanism that gives more weight to areas useful for detection. In figure 4.1, we can visualize what the network learns from this approach, that is to concentrate the keypoint search on areas where there are indeed road users, and therefore reducing the response of any other neuron. One can see this process as shining a spotlight on relevant areas and dimming the lights everywhere else. Therefore, we named our method, SpotNet.

This attention approach is particularly beneficial for keypoint-based methods since we are globally looking for keypoints on the whole image at the same time, and not just classifying the object in a cropped bounding box. However, a question remains. How can we train such a self-attention process ? Typically, object detection datasets do not provide the segmentation ground-truth since it is very costly and time-consuming to produce. Instead, we rely on classical computer vision techniques to generate automatic pixel-wise annotation labels and on datasets providing video sequences instead of single frames to train the network. In the case of fixed camera video sequences, we successfully employ a background subtraction method to obtain the automatic annotations, while in the case of moving camera video sequences, we rely on dense optical flow for the same purpose.

Although we use imperfect foreground/background segmentation annotations, we can train a network to produce quality segmentation maps by using multi-task learning. The detection

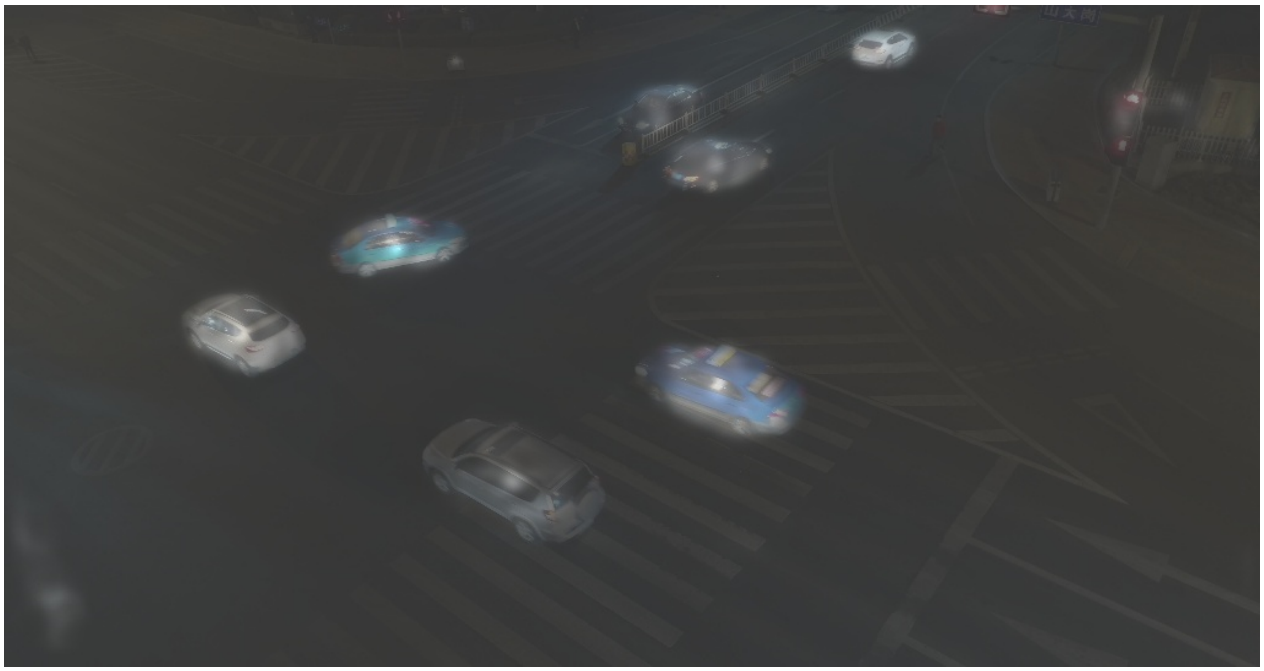


Figure 4.1 A visualisation of the attention map produced by SpotNet on top of its corresponding image, from the UAVDT [7] dataset.

and segmentation tasks are trained jointly by sharing all the parameters of the backbone network. Both tasks are mutually beneficial. Indeed, by producing a better segmentation, the object detection task benefits from a better attention mechanism. And by producing better object detection, the parameters of the backbone network get better at recognizing the features of interest from the images to improve the segmentation maps. We validated our method on two popular traffic scene datasets, and we show that our method is the state-of-art on these datasets by improving significantly the performance of the base network (CenterNet).

4.2 Related Work

Object detection as meant in this paper is the task of drawing a rectangular bounding box around objects of interest in an image, as well as producing a class label for each box. All state-of-the-art object detection methods have been based on deep learning since its rise. They can broadly be split up into two main categories, two-stage and one-stage methods.

Two-stage methods divide the task of object detection into two steps, producing a set of object candidates, and then computing a score, a label and a coordinate offset for each

box. The first deep learning-based method was R-CNN [28], which used an external method to produce box candidates, namely selective search [31]. It then passed each candidate in a CNN to compute features for each box. A classification is done on those features by a SVM afterwards. Fast R-CNN [29] aimed to increase the speed of R-CNN by passing the whole image through a CNN once, and afterwards just cropped the relevant parts of the feature map for each box candidate for classification. Faster R-CNN [15] is a further improvement that introduced the RPN, a region proposal network that shares most of its parameters with the classification and regression parts, making it even faster and more efficient than its predecessors. RFCN [32] further builds upon Faster R-CNN by learning to detect and classify parts of objects and then using a grid of parts to vote on each object. Cascade R-CNN [60] addresses the problems of the mismatch between the minimum IOU (Intersection over union) used to evaluate during inference, and the minimum IOU used to select a positive sample during training. They also address overfitting during training by training while progressively increasing the IOU thresholds.

One-stage methods aim to reduce the processing time of two-stage methods by removing the candidate proposal phase and by detecting objects directly from the feature map. The first one-stage method was YOLO [16] which divides the input image into a regular grid and makes each cell predict two bounding boxes. Further iterations of the method, YOLOv2 [33] and YOLOv3 [34] built upon it by using anchor boxes, a better backbone network and several other tweaks. SSD [30] addresses the multi-scale detection problem by combining feature maps at multiple spatial resolutions and then applying anchor boxes to look for objects. RetinaNet [3] uses an FPN (Feature pyramid network) [61] to produce a multi-scale pyramid of features and applies a set of anchor boxes followed by non-maximal suppression to find objects. CornerNet [36] uses the Hourglass network [14] paired with corner pooling layers to detect a set of top-left corners and bottom-right corners, and combines them with a learned embedding. Keypoint Triplets [17] builds upon CornerNet by improving the corner pooling layers, and by also detecting a center keypoint to validate each object. Objects as Points [1] detects an object as a center keypoint and regresses the size of the object to find the bounding box.

Attention mechanisms in object detection have been around for a while. In Geometric Proposal for Faster R-CNN [62], the authors re-rank the proposals of the region proposal network depending on a geometric estimation of the scene, outperforming the standard Faster R-CNN by a large margin. Their geometric estimation of the scene is mostly based on vehicle scale. The HAT [63] method uses a hierarchical attention mechanism that first trains a part-specific attention model. Then an LSTM models the relations between those parts, making it a part-aware detector. FG-BR Net [64] uses background subtraction methods to produce a

foreground image that is fed as another input to the network. They also introduce a feedback process from the detection outputs to the background subtraction to keep static objects in the foreground image. Compared to these models, our attention mechanism is simple, elegant and fast. Furthermore, we do not need background subtraction at inference, only during the training phase.

4.3 Proposed Method

Figure 4.2 shows a detailed overview of our complete model.

4.3.1 Base network

Our method is based upon CenterNet [1], not to be confused with the homonym method CenterNet, or keypoint triplets [17]. This method trains a backbone network to recognize the center point of objects by assigning the center pixel of a box to be the ground-truth center and gives a reduced loss for other close points. The width and height of the bounding box are regressed, as well as the coordinate offset of the box (to compensate from the error caused by the smaller spatial resolution of the output). The final output is thus a center point heatmap for each possible label, an object size for each point, and an offset for each point, the size and offset being label agnostic.

4.3.2 Multi-task Learning

Our main idea is to train a network to perform multiple tasks, to make it better for at least one of the tasks. In our case, we train a network to perform segmentation of objects of interest while performing bounding box detection, thus making the shared parameters more generic and less prone to overfitting. To do this, we add a two-class (foreground/background) segmentation head to the network and train this head with semi-supervised annotations from training datasets (more details in subsection 4.3.4).

The added segmentation head takes as input a feature map that has been reduced by a factor of four in terms of spatial dimension when compared to the input. It consists of three 3×3 convolutions, with upsampling layers in between. The channel dimension is reduced to 1 in the last convolution, thus resulting in a segmentation map that is the same width and height as the input, with a single channel. The loss, L_{seg}, used to train this head is the binary cross-entropy, given by

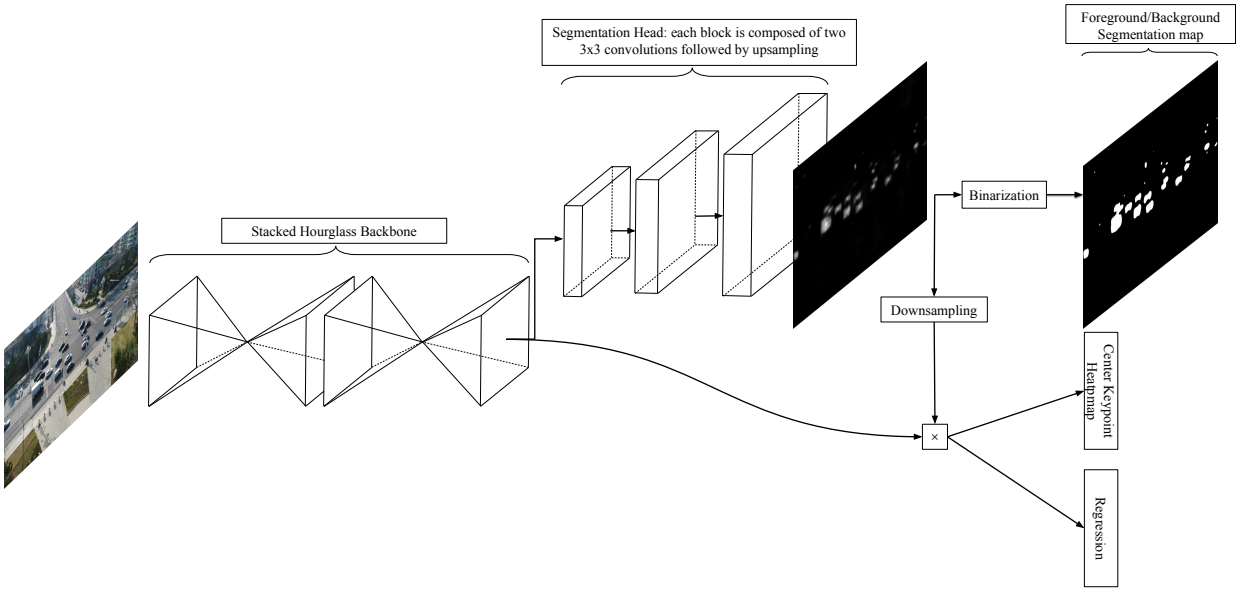


Figure 4.2 Overview of SpotNet : the input image first passes through a double-stacked hourglass network ; the segmentation head then produces an attention map that multiplies the final feature map of the backbone network ; the final center keypoint heatmap is then produced as well as the size and coordinate offset regressions for each object.

$$L_{seg} = -\frac{1}{N}\sum_{i=1}^{N} y_i * log(x_i) + (1 - y_i) * log(1 - x_i), \tag{4.1}$$

where y_i is the annotation label for sample i, x_i its predicted label by the network and N the number of samples. We found out during our experiments that it works better than the initial mean squared error loss that we had tried initially.

4.3.3 Self-Attention Mechanism

To further benefit from our learned segmentation map, we implement a simple yet effective self-attention mechanism within the network. Once we obtain our segmentation map, we downsample it by a factor of 4 to reduce it to the spatial dimension of the original feature map. To attenuate the response at locations unlikely to contain an object of interest, we multiply every channel of the feature map with our segmentation map, thus reducing the probability of false positives in irrelevant areas.

4.3.4 Semi-Supervised annotations

To train our model to produce foreground/background segmentation maps, we had to produce semi-supervised pixel-wise segmentation annotations. To do that, we took advantage of having access to full video sequences, despite training and evaluating on a single frame at a time. For the fixed camera video sequences, we used the background subtraction method PAWCS [23]. Since background subtraction is not designed to work with a moving background, for the moving camera video sequences, we used Farneback optical flow [65] followed by some basic image processing and a threshold on motion magnitude [2]. For both automatic two-class segmentation results, we then do an intersection with the ground-truth bounding boxes for each frame to reduce noise and to obtain pixel-wise segmentation annotations only for the object categories to detect. All other object categories, not inside ground-truth training bounding boxes, are therefore labelled as background. This results in fairly good foreground/background segmentation maps, with sometimes squared corners at one or more sides, due to the intersection with bounding boxes, as can be seen in Figure 4.3. In our experiments, we find that not only are these non-perfect segmentation annotations good enough to train good attention maps, it also allows our segmentation head to produce segmentation maps comparable to good unsupervised foreground/background segmentation methods.

It should be noted that although our method requires videos for training to obtain the semi-supervised segmentation annotations, once trained, it can be applied to single images.

4.3.5 Training for multiple tasks

To adapt the training loss of the whole network, we added the binary cross-entropy loss of our segmentation head (equation 4.1) to the original CenterNet loss. The center point heatmap loss L_{heat} is calculated with the focal loss [3], and the losses for the regressions for the offset L_{off} and width/heigth L_{WH} are formulated as L1 losses as in the original paper [1]. The total loss L_{tot} is given by

$$L_{tot} = L_{heat} + L_{off} + L_{seg} + 0.1 * L_{WH}. \tag{4.2}$$

The total loss is thus the sum of all losses, with the width and height regression having less weight than the others, 0.1 compared to 1.

2. The exact parameters were determined empirically. The basic image processing includes noise cancelling and filling holes.



Figure 4.3 Example of semi-supervised annotations on UA-DETRAC [5] produced by PAWCS [23] and the intersection with the ground-truth bounding boxes.

4.4 Experiments

4.4.1 Datasets

To validate the effectiveness of our method, we trained and evaluated it against other state-of-the-art methods on two datasets of traffic scenes, namely UA-DETRAC [5] and UAVDT [7]. Figure 4.4 and Figure 4.5 show example frames of UA-DETRAC and UAVDT respectively, with their ground-truth. These two datasets were captured with very different settings, UA-DETRAC being filmed with a fixed camera for every scene, and UAVDT with a moving camera. Both datasets have pre-determined test sets, and we used a subset of the training data to do the validation.

Evaluation is done using the Matlab code provided by the authors of both datasets. A strict training and validation protocol was followed and the testing data was never seen by the network before the final evaluation. The performance measure used for evaluation is the mAP, the mean Average Precision, with a minimum IOU of 0.7 between inferred and ground-truth bounding boxes. The minimum IOU is the minimum overlap of a bounding box with the ground-truth to be considered a true detection. The IOU is computed as the intersection

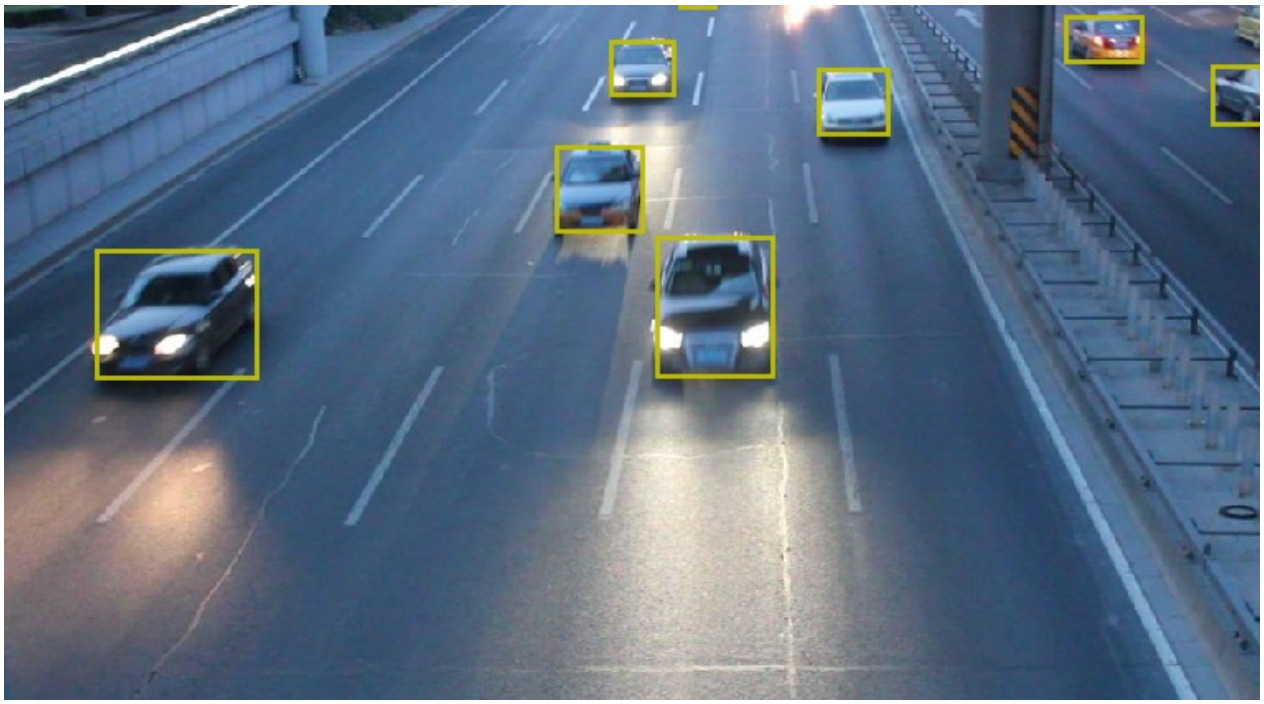


Figure 4.4 Sample from UA-DETRAC with the ground-truth bounding boxes in yellow.

of the boxes divided by the union of the boxes. The mean average precision is the mean of the average precisions for all classes for multiple values of recall, ranging from 0 to 1 with small steps, typically of 0.1.

4.4.2 Implementation Details

We used the stacked hourglass network as our backbone because it shows the best performance for keypoint estimation. This network is composed of modules of downsampling and convolutions followed by upsampling and convolutions with skip connections in an encoder-decoder fashion. For our experiments, we use the Hourglass-104 version as in [36] which stacks two encoder-decoder modules. We implemented the model in PyTorch 0.4.1 using Cuda 10.0. Experiments were run on a workstation with 32 GB of RAM and a NVIDIA GTX 1080Ti GPU. The Github repository for this project is `https://github.com/hu64/SpotNet`.

4.4.3 Object detection results

The experimental results are shown in Table 4.1 for UA-DETRAC and in Table 4.2 for UAVDT. We outperform our baseline, CenterNet, by a very significant margin on both datasets while being the state-of-art results on both datasets as well. The results are very

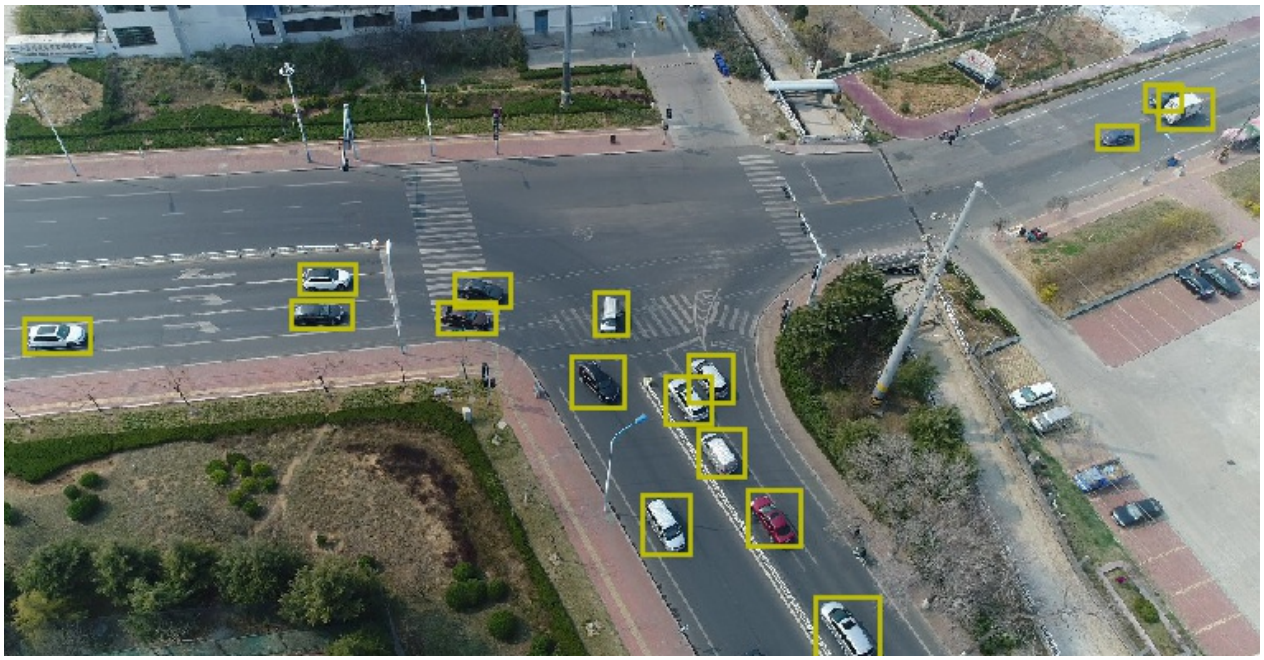


Figure 4.5 Sample from UAVDT with the ground-truth bounding boxes in yellow.

coherent, showing approximately the same percentage of improvement over CenterNet on both datasets, the absolute value on UAVDT being smaller.

For UA-DETRAC, not only do we outperform all previously published results, we do so in every category, showing the benefit of our self-attention mechanism based on multi-task learning. Moreover, the improvements are particularly impressive for the category hard and cloudy, meaning that our model is particularly good for hard examples. It is interesting to note that the improvement for the easy category is very small, due to the mAP values being already very high. Nonetheless, improvement is consistent across all categories. At the moment of writing, our model outperforms every published result on this dataset, including ensemble models from challenges [66].

The UAVDT dataset is more difficult than UA-DETRAC due to its high density of small vehicles and aerial point of view, but the percentage of improvement remains consistent. Our model also outperforms every published result on this dataset by a very significant margin.

4.4.4 Foreground/Background segmentation results

Although that was not our principal objective, it is nonetheless interesting to see how we do on specialized foreground/background benchmarks. To produce results, we used our best

Tableau 4.1 Results on the UA-DETRAC dataset [5]. 3D-DETNet results are from [6], and others results are reported as in the results section of the UA-DETRAC website (**Boldface** : best result, *Italic* : indicates our baseline).

Model	Overall	Easy	Medium	Hard	Cloudy	Night	Rainy	Sunny
SpotNet (ours)	**86.80**%	**97.58**%	**92.57**%	**76.58**%	**89.38**%	**89.53**%	**80.93**%	**91.42**%
CenterNet [17]	83.48%	96.50%	90.15%	71.46%	85.01%	88.82%	77.78%	88.73%
FG-BR_Net [64]	79.96%	93.49%	83.60%	70.78%	87.36%	78.42%	70.50%	89.8%
HAT [63]	78.64%	93.44%	83.09%	68.04%	86.27%	78.00%	67.97%	88.78%
GP-FRCNNm [62]	77.96%	92.74%	82.39%	67.22%	83.23%	77.75%	70.17%	86.56%
R-FCN [32]	69.87%	93.32%	75.67%	54.31%	74.38%	75.09%	56.21%	84.08%
EB [67]	67.96%	89.65%	73.12%	53.64%	72.42%	73.93%	53.40%	83.73%
Faster R-CNN [15]	58.45%	82.75%	63.05%	44.25%	66.29%	69.85%	45.16%	62.34%
YOLOv2 [33]	57.72%	83.28%	62.25%	42.44%	57.97%	64.53%	47.84%	69.75%
RN-D [22]	54.69%	80.98%	59.13%	39.23%	59.88%	54.62%	41.11%	77.53%
3D-DETnet [6]	53.30%	66.66%	59.26%	43.22%	63.30%	52.90%	44.27%	71.26%

model trained on UA-DETRAC and ran it on three sequences of the changedetection.net dataset [8] containing only vehicles (because UA-DETRAC includes only annotations for vehicles). To obtain the foreground, we took the attention maps produced by our network, applied a binary threshold and then masked the resulting image with the bounding boxes detected by our network to remove noise. We can see in table 4.3 that our method produces competitive results, although we do not quite reach state-of-the-art foreground/background performance [3]. Figure 4.6 shows qualitative results on a few frames. Our method does not always fit the object boundaries very well. This is expected since the training annotations are imperfect. Nevertheless, we outperform several classical methods, at no additional cost when producing bounding boxes. It is important to note that a limitation of our model is that it must be trained on the objects we want to segment [4].

3. The F-Measure is the metric used by the dataset, which is therefore used instead of mAP to compare with the other methods.

4. Meaning that the annotated objects must correspond to the ones in the foreground/background annotations

Tableau 4.2 Results on the UAVDT [7] dataset (**Boldface** : best result, *Italic* : indicates our baseline).

Model	Overall
SpotNet (Ours)	**52.80**%
CenterNet [17]	51.18%
Wang *et al.* [68]	37.81%
R-FCN [32]	34.35%
SSD [30]	33.62%
Faster-RCNN [15]	22.32%
RON [69]	21.59%

Tableau 4.3 Results on the changedetection.net [8] dataset. Results are averaged for sequences "highway", "traffic" and "boulevard" (**Boldface** : best result).

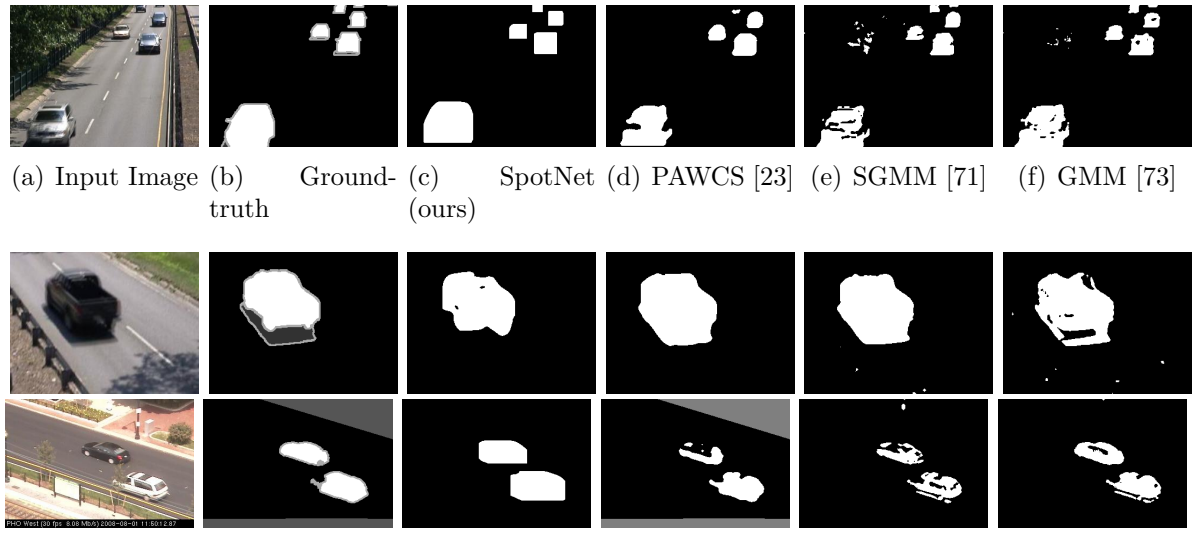
Model	Average F-Measure
PAWCS [23]	**0.872**
SuBSENSE [70]	0.831
SpotNet (Ours)	0.806
SGMM [71]	0.766
KNN [72]	0.731
GMM [73]	0.709

4.5 Discussion

4.5.1 Ablation study

To detail the contribution of each part of our model, we conducted an ablation study on UA-DETRAC. Table 4.4 shows that even though multi-task learning helps, the biggest contribution comes from combining our attention process with it. To further understand the contribution of each part, we draw the precision/recall curve (Figure 4.7) compared to several other methods on UA-DETRAC. On this curve, we can note that the multi-task learning by itself (SpotNet No Attention) helps to be more precise, but does not help to detect more objects, i.e. to reach improved values of recall. On the other hand, the attention mechanism does both, it helps to be even more precise for the same values of recall (fewer false positives), and it also allows the model to detect more and reach significantly higher values of recall.

Since the network is looking for keypoints on the whole image, it is natural that concentrating the search on learned foreground pixels will increase the probability that the keypoints found



(a) Input Image (b) Ground-truth (c) SpotNet (ours) (d) PAWCS [23] (e) SGMM [71] (f) GMM [73]

Figure 4.6 Example of foreground/background segmentation maps obtained with several segmentation methods. First row : frame 1015 of "highway", second row : frame 967 of "traffic", third row : frame 883 of "boulevard".

Tableau 4.4 Ablation study on the UA-DETRAC [5] dataset.

Attention	Multi-Task	Overall	Easy	Medium	Hard	Cloudy	Night	Rainy	Sunny
✓	✓	**86.80**%	**97.58**%	**92.57**%	**76.58**%	**89.38**%	**89.53**%	**80.93**%	**91.42**%
	✓	84.57%	96.72%	90.85%	73.16%	86.53%	88.76%	78.84%	90.10%
		83.48%	96.50%	90.15%	71.46%	85.01%	88.82%	77.78%	88.73%

belong to the objects of interest, thus reducing the rate of false positives. Furthermore, the experiments show that this increases recall because the network can concentrate on useful information.

It is expected that learning the segmentation task jointly with the object detection task can be mutually beneficial since both tasks have a large overlap in what needs to be learned. The main difference is that object detection needs to separate instances, while segmentation needs a more precise border around the objects. We show that semi-supervised annotations are good enough for our purpose, and multi-task learning by itself, based on those annotations, improves precision.

4.5.2 Limitations of our Model

One of the limitations of our model is the fact that it needs semi-supervised annotations to be trained properly. However, we believe that in most real-world applications, video sequences are available and we can thus run background subtraction or optical flow to generate them. In

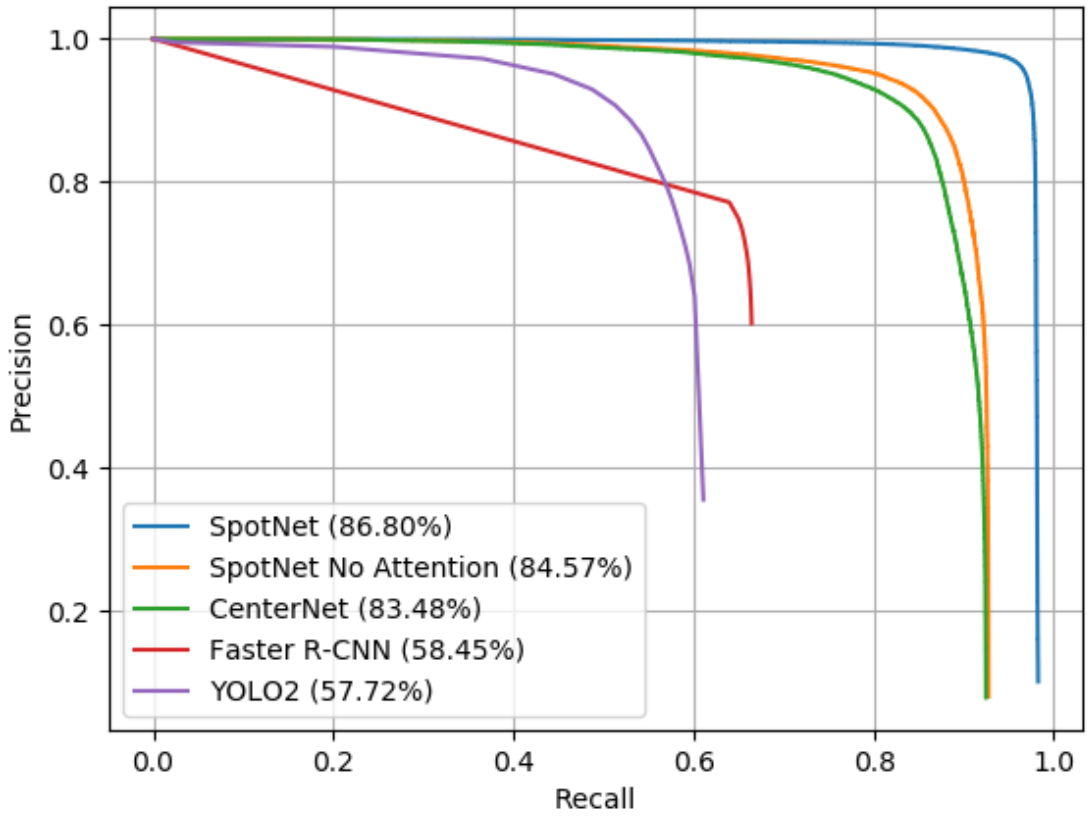


Figure 4.7 Precision/Recall curve of our model compared with a variant and other methods.

other cases, pre-trained semantic segmentation methods could be used to obtain the desired annotations.

4.6 Conclusion

In this paper, we presented a novel multi-task model equipped with a self-attention process, and we trained it with semi-supervised annotations. We show that these improvements allow us to reach state-of-the-art performance on two traffic scenes datasets with different settings. We argue that not only does this improve accuracy by a large margin, it also provides instance segmentations of the road users almost at no cost [5].

5. The cost meant here is the additional time and memory required.

Acknowledgment

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [RDCPJ 508883 - 17], and the support of Genetec.

CHAPITRE 5 ARTICLE 2 : RN-VID : A FEATURE FUSION ARCHITECTURE FOR VIDEO OBJECT DETECTION

Perreault H., Heritier M., Gravel P., Bilodeau GA., Saunier N.
Image Analysis and Recognition. ICIAR 2020. Lecture Notes in Computer Science, vol 12131. Springer, Cham.
https ://doi.org/10.1007/978-3-030-50347-5_12

Abstract

Consecutive frames in a video are highly redundant. Therefore, to perform the task of video object detection, executing single frame detectors on every frame without reusing any information is quite wasteful. It is with this idea in mind that we propose RN-VID (standing for RetinaNet-VIDeo), a novel approach to video object detection. Our contributions are twofold. First, we propose a new architecture that allows the usage of information from nearby frames to enhance feature maps. Second, we propose a novel module to merge feature maps of same dimensions using re-ordering of channels and 1×1 convolutions. We then demonstrate that RN-VID achieves better mean average precision (mAP) than corresponding single frame detectors with little additional cost during inference.

5.1 Introduction

Convolutional neural network (CNN) approaches have been dominant in the last few years for solving the task of object detection, and there has been plenty of research in that field. On the other hand, research on video object detection has received a lot less attention. To detect objects in videos, some approaches try to speed up inference by interpolating feature maps [19], while others try to combine feature maps using optical flow warping [18]. In this work, we present an end-to-end architecture that learns to combine consecutive frames without prior knowledge of motion or temporal relations.

Even though research on video object detection has been less popular than its single frame counterpart, the applications are not lacking. To name a few : autonomous driving, intelligent traffic systems (ITS), video surveillance, robotics, aeronautics, etc. In today's world, there is a pressing need to build reliable and fast video object detection systems. The number of possible applications will only grow over time.

Using multiple frames to detect the objects on a frame presents clear advantages, if used correctly. It can help solve problems like occlusion, motion blur, compression artifacts and small objects (see in figure 6.1). When occluded, an object might be difficult or nearly impossible to detect and classify. When moving, or when the camera is moving, motion blur can occur in the image making it more challenging to locate and recognize objects because it changes their appearance. In digital videos, compression artifacts can alter the image quality and make some parts of the frame more difficult to analyze. Small objects can be difficult to locate and recognize, and having multiple frames allows us to use motion information (implicitly or explicitly) as a way to help us find them. Implicitly by letting the network learn how to do it, explicitly by feeding the network optical flow or frame differences.

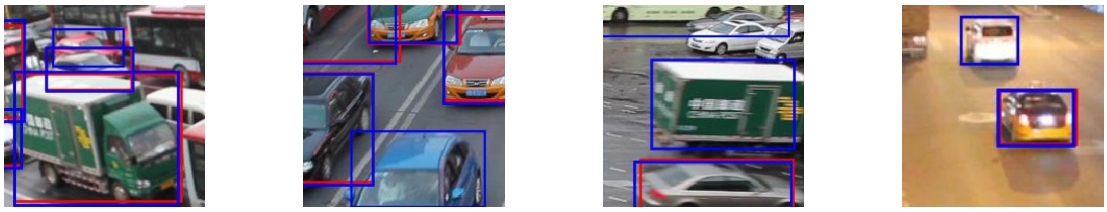
Our model relies on the assumption that a neural network can learn to make use of the information in successive frames to address these challenges, and this paper demonstrates the advantages of such a model. Frame after frame, the object instances are repeated several times under slightly different angles, occlusion levels and illuminations, in a way that could be thought as similar to data augmentation techniques. We seek to make the network learn what is the best fusion operation for each feature map channel originating from several frames. Our proposed method contains two main contributions : an object detection architecture based on RetinaNet [3] that merges feature maps of consecutive frames, and a fusion module that merges feature maps without any prior knowledge or handcrafted features. Combined together, these two contributions form an end-to-end trainable framework for video object detection and classification.

Since this domain contains a lot of interesting challenges and applications, our evaluation is concentrated on traffic surveillance scenes. The effectiveness of our method is evaluated on two popular object detection datasets composed of video sequences, namely UA-DETRAC [5] and UAVDT [7]. We compare both with the RetinaNet baseline from which we build upon, and state-of-the-art methods from the public benchmarks of those datasets. Results show that our method outperforms both the baseline and the state-of-the art methods.

5.2 Related Work

5.2.1 Object Detection

Over the last few years, the research focus for object detection has been on single frame detectors. Deep learning-based methods have been dominant on all benchmarks. The two main categories are two-stage detectors, which use a region proposal network, and single-stage detectors, which do not. R-CNN [28], a two-stage detector, was the first dominant



(a) (b) (c) (d)

Figure 5.1 Qualitative examples where our model (blue) performs better than the RetinaNet baseline (red). (a) the two cars in the back are heavily occluded by the green truck, (b) the car in the bottom center is being occluded by the frame boundary, (c) the green truck is blurry due to motion blur, (d) as cars become smaller, they become harder to detect, like the white one at the top.

object detector to use a CNN. It used an external handcrafted object proposal method called selective search [31] to produce bounding boxes. It would then extract features for each bounding box using a CNN and would classify those features using SVM. Fast R-CNN [29] builds upon this idea by addressing the bottleneck (passing each bounding box in a CNN). The way it solves this problem is by computing deep features for the whole image only once and cropping these corresponding features for each bounding box proposals. Faster R-CNN [15] improves furthermore by making the architecture completely trainable end-to-end by using a CNN to produce bounding box proposals, and by performing a classification and regression to refine the proposals. R-FCN [32] improves Faster R-CNN by introducing position sensitivity of objects, and by doing so can localize them more precisely. It divides each proposal into a regular grid and classifies each cell separately. In Evolving Boxes [67], the authors build an architecture specialized for fast vehicle detection that is composed of a proposal and an early discard sub-network to generate candidates under different feature representation, as well as a fine-tuning sub-network to refine those boxes.

Single-stage object detectors aim to speed up the inference by removing the object proposal phase. That makes them particularly well suited for real-time applications. The first notable single-stage network to appear was YOLO [16], which divides the image into a regular grid and makes each grid cell predict two bounding boxes. The main weakness of YOLO is thus large numbers of small objects, due to the fact that each grid cell can only predict two objects. A high density of small objects is often found in the traffic surveillance context. Two improved versions of YOLO later came out, YOLOv2 [33] and YOLOv3 [34]. SSD [30] tackles the problem of multi-scale detection by combining feature maps at multiple levels and applying a sliding window with anchor boxes at multiple aspect ratio and scale. RetinaNet [3] works similarly to SSD, and introduces a new loss function, called focal loss that addresses

the imbalance between foreground and background examples during training. RetinaNet also uses the state-of-the-art way of tackling multi-scale detection, Feature Pyramid Network (FPN) [61]. FPN builds a feature pyramid at multiple levels with the help of lateral and top-down connections and performs classification and regression on each of these levels.

5.2.2 Video Object Detection

Here we present an overview of some of the most notable work on video object detection. In Flow Guided Feature Aggregation (FGFA) [18], the authors use optical flow warping in order to integrate feature maps from temporally close frames, which allows them to increase detection accuracy. In MANet [39], the authors use a flow estimation and train two networks to perform pixel-level and instance-level calibration. Some works incorporate the temporal aspect explicitly, for example, STMM [40] uses a recurrent neural network to model the motion and the appearance change of an object of interest over time. Other works focus on increasing processing speed by interpolating feature maps of intermediate frames, for instance in [19] where convolutional Long Short-Term Memories (LSTMs) are used. These previous works use some kind of handcrafted features (temporal or motion), while our work aims to train a fusion module completely end-to-end. Kim *et al.* [42] trained a model by using deformable convolutions that could compute an offset between frames. Doing so allowed them to sample features from close frames to better detect objects in a current frame. This helps them in cases of occlusion or blurriness. In 3D-DETNet [6], to combine several frames, the authors focus on using 3D convolutions on concatenated features maps, generated from consecutive frames, to improve them. MF-SSD [41], standing for Recurrent Multi-frame Single Shot Detector, extends the SSD [30] architecture to merge features of multiple sequential frames with a recurrent convolutional module. Perreault *et al.* [22] trained a network on concatenated image pairs for object detection but could not benefit from pre-trained weights and therefore had to train the network from scratch to outperform the detection on a single frame.

5.2.3 Optical flow by CNNs

Works on optical flow by CNNs showed that we can train a network to learn motion from a pair of images. Therefore, similar to our goal, these works put together information from consecutive frames. FlowNet [74] is the most notorious work in this field, being the first to present an end-to-end trainable network for estimating optical flow. In the paper, two models are presented, FlowNetSimple and FlowNetCorr. Both models are trained on an artificial dataset of 3D models of chairs. FlowNetSimple consists of a network that takes as input

a pair of concatenated images, while FlowNetCorr used a correlation map between higher level representation of each image of the pair. The authors later released an improved version named FlowNet 2.0 [75] that works by stacking several slightly different versions of FlowNet on top of each other to gradually refine the flow.

5.3 Proposed Method

Formally, the problem we want to solve is as follows : given a target image, a window of n preceding and n future frames and predetermined types of objects, place a bounding box around and classify every object of the predetermined types in the target image.

To address this problem, we propose two main contributions, a novel architecture for object detection and a fusion module to merge feature maps of the same dimensions. We crafted this architecture to allow the usage of pre-trained weights from ImageNet [26] in order to build over methods from the state-of-the-art.

5.3.1 Baseline : RetinaNet

We chose to use the RetinaNet [3] as a baseline upon which to build our model, due to its high speed and good performance. To perform detection at various scales, RetinaNet uses an FPN, which is a pyramid of feature maps at multiple scales (see figure 6.2). The pyramid is created with top-down and side connections from the deepest layers in the network, and going back towards the input, thus growing in spatial dimension. A sliding window with boxes created with multiple scales and aspect ratios is then applied at each pyramid level. Afterwards, every box is passed through a classification and a regression sub-network. Finally, non maximal suppression is performed to remove duplicates. The detections with the highest confidence scores are the ones that are kept. As a backbone extractor, we used VGG-16 [24] for the good trade-off between speed and size that it offers. RetinaNet uses the focal loss for classification :

$$FL(p') = -\alpha_t(1-p')^{\gamma} log(p') \tag{5.1}$$

where γ is a factor that diminish the loss contributed by easy examples. α_t is the inverse class frequency, and its purpose is to give more representation to underrepresented classes during training. p' is the probability that the predicted label corresponds to the ground-truth label.

So, if the network predicts with a high probability and is correct, or a low probability and is incorrect, the loss will be marginally affected due to those examples being easy. For the cases where the network is confident (high probability) and incorrect at the same time, the

examples will be considered hard and the loss will be affected more.

5.3.2 Model Architecture

The main idea of the proposed architecture is to be able to compute features for every frame of a sequence only once, and to be able to use these pre-computed features to enhance the features for a target frame t. The fusion module thus comes somewhat late in the network.

Our network uses multiple input streams that eventually merge into a single output stream, as shown in figure 6.2. For computing the feature pyramid for a frame at time t, we will use n preceding frames and n future frames. All the $2n+1$ frames are passed through the VGG-16 network, and we keep the outputs of blocks B3, B4 and B5 for each frame. In RetinaNet, these outputs are used to create the feature pyramid. We then use our fusion module to merge the corresponding feature maps of each frame (block B3 outputs together, block B4 outputs together, etc.) in order to enhance them, before building the feature pyramid. This allows us to have higher quality features and to better localize objects. We then use the enhanced maps as in the original RetinaNet to build the feature pyramid.

During the training process, we have to use multiple frames for one ground-truth example, thus slowing down the training process. However, for inference on video, the features computed for each frame are used multiple times making the processing time almost identical [1] to the single frame baseline.

5.3.3 Fusion Module

In order to combine equivalent feature maps of consecutive frames, we designed a lightweight and trainable feature fusion module (see figure 6.3). The inspiration for this module is the various possible way a human would do the task. Let us say you wanted to combine feature map channels of multiple consecutive frames. Maybe you would look for the strongest responses and only keep those, making the merge operation an element-wise maximum. Maybe you would want to average the responses over all the frames. This 'merge operation' might not be the same for all channels. The idea is to have a network learn the best way to merge feature maps for each channel separately, with 1×1 convolutions over the channels.

In our fusion module, we use 1×1 convolutions in order to reduce the dimension of tensors. In the Inception module [76] of the GoogLeNet, the 1×1 convolution is actually used as a way to reduce the dimensions which inspired our work. The inception module allowed them to build a deeper and wider network while staying computationally efficient. In contrast, in

1. The additional inference time is only the time spent by the fusion module.

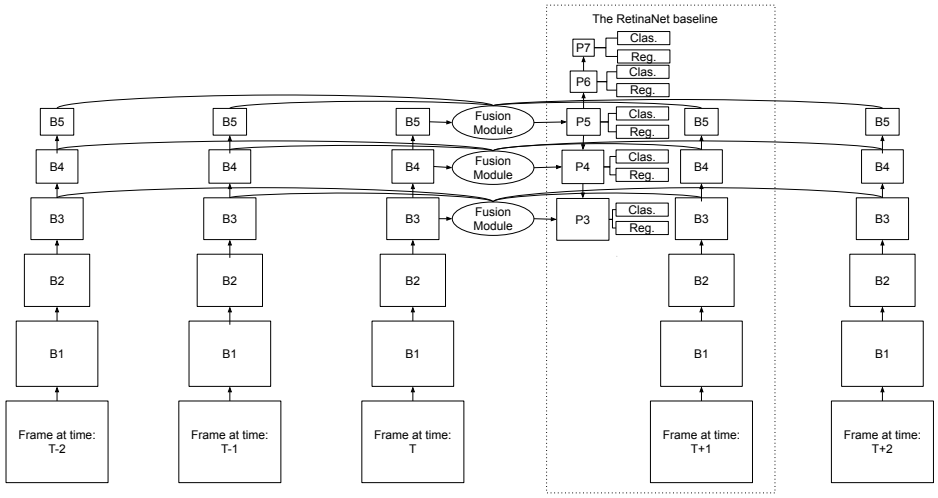


Figure 5.2 A representation of our architecture with $n = 2$. Each frame is passed through a pre-trained VGG-16, and the outputs of block 3, block 4 and block 5 are collected for fusion. B1 to B5 are the standard VGG-16 [24] blocks, and P3 to P7 are the feature pyramid levels. In the dotted frame is an overview of our baseline, a RetinaNet [3] with VGG-16 as a backbone.

our work, we use 1×1 convolutions for learning to merge feature maps.

The module takes as input $2n + 1$ feature maps of dimension $w * h * c$ (for width, height and channels respectively), and outputs a single enhanced feature maps of dimension $w*h*c$. The feature maps that we are combining come from corresponding pre-trained VGG-16 layers, so it is reasonable to think that corresponding channels are responses from corresponding 'filters' [2]. The idea is to take all the corresponding channels from the consecutive frames, and combine them to end up with only one channel, and thus re-build the wanted feature map, as shown in figure 6.3.

Formally, for $2n + 1$ feature maps of dimension $w * h * c$, we extract each c channels one by one and concatenate them, ending up with c tensors of dimension $w * h * (2n + 1)$. We then perform a 2D convolution with a 1×1 convolution kernel $(1 * 1 * (2n + 1))$ on the c tensors, getting c times $w*h*1$ as an output. The final step is to concatenate the tensors channel-wise to finally get the $w * h * c$ tensor that we need. The module is entirely learned, so we can interpret this module as the network learning the operation that best combines feature maps, for each channel specifically, without any prior knowledge or handcrafted features.

2. Meaning that the same channels for the different feature maps represent the fame feature. The response to the same filter.

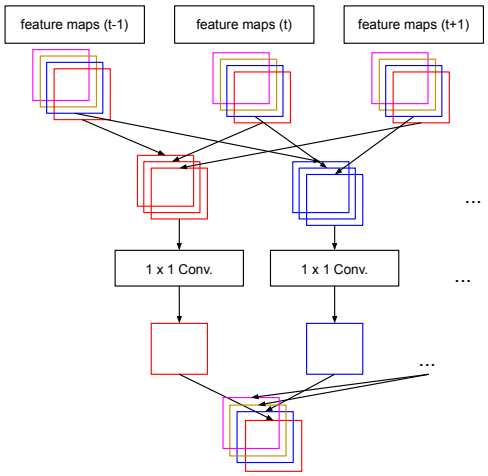


Figure 5.3 Our fusion module consists of channel re-ordering, concatenation, 1×1 convolution, and a final concatenation (better seen in color).

5.4 Experiments

5.4.1 Datasets

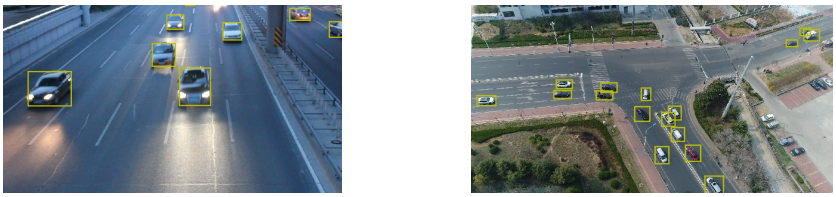
The training process of our method requires consecutive images from videos. We chose two datasets containing sequences of moving road users : UA-DETRAC [5] (fixed camera, 960x540, 70000 images, 4 possible labels, see figure 5.4(a)) and the Unmanned Aerial Vehicle Benchmark (UAVDT) [7] (mobile camera, 3 possible labels, 80000 images, high density of small objects, see figure 5.4(b)).

5.4.2 Implementations Details

We implemented the proposed model in Keras [77] using TensorFlow [78] as the backend. We used a standard RetinaNet as our baseline, without any bells and whistles or post-processing. We want to keep the models simple in order to properly show the contributions of our architecture and fusion module.

We built a feature pyramid with five different levels, called P3, P4, P5, P6, P7, with the outputs of block 3, 4, 5 of VGG-16. P3 to P5 are the pyramid levels corresponding to block 3 to 5. P6 and P7 are obtained via convolution and down-sampling of P5, and their size is reduced in half at each level : P6 is the half the size of P5, and P7 is the half the size of P6. This is standard for RetinaNet.

For UAVDT and UA-DETRAC, we adapted the scales used for the anchor boxes by reducing



(a) (b)

Figure 5.4 (a) An example frame of UA-DETRAC and its ground-truth annotations. (b) An example frame of UAVDT and its ground-truth annotations.

them, due to the high number of small objects in the tested datasets. Instead of using the classic $2^0, 2^{(1.0/3.0)}, 2^{(2.0/3.0)}$ scale ratios, we used 2^0, $1/(2^{(1.0/3.0)})$, $1/(2^{(2.0/3.0)})$. This modification did not affect the results on UA-DETRAC, but improved them on UAVDT, causing a bigger gap with the reported state-of-the-art results in the paper. Since we use the same scales for our baseline, this has no effect on our conclusions. The focal loss parameter γ is 2 and we used an initial learning rate of 1e-5.

To train both the model and the baseline, we used the adam optimizer [79]. In order to fit the model into memory, we had to freeze the first four convolutional blocks of the VGG-16 model during training, and only retrained the other weights, with a batch size of one. For a fair comparison, we used the same training setting for our baseline. Despite this limitation, we still achieve state-of-the-art results when compared to single frame object detectors. Even though the first four convolutional blocks are frozen, they are still initialized with fine-tuned weights for each dataset. Note that the weights used to initialize the backbone are the same for the baseline and the model.

To select the hyperparameter n of our method (the number of frames used before and after), we used a validation set and tried a few values. $n = 2$ was the value that worked best for us, so that is the value we use for the final results. We show the results of different values of n in an ablation study in table 6.3.

5.4.3 Performance Evaluation

For the two datasets, the test set is predetermined and cannot be used for training or to fix hyperparameters. We split the training data into training and validation by choosing a few whole sequences for validation, and the others for training. We did this to prevent overfitting on the validation data that would likely happen if we would split randomly between all frames. We trained the models until the validation loss started to increase, meaning the model was overfitting.

The performance measure used for evaluation is the mAP, meaning Mean Average Precision. The mAP is the mean AP for every class. The AP is the average precision considering the recall and precision curves ; thus, it is the area under the precision-recall curve. The minimum intersection over union (IOU) between the ground-truth and the prediction bounding box, to consider a detection valid, is 0.7 for UA-DETRAC and UAVDT, as defined by the dataset's protocols. The IOU, or the Jaccard index, is the intersection area between two rectangles divided by the union area between them.

5.4.4 Results

Results on UA-DETRAC

Results on the UA-DETRAC dataset are reported in table 6.1. We drew the ROC curves for our model, the baseline and few other state-of-the-art models in figure 5.5(a). Our detector outperforms all classic state-of-the-art models evaluated on UA-DETRAC as well as the baseline by a significant margin.

Something interesting to notice is that our model outperforms R-FCN for the categories labeled "hard" and "cloudy", confirming our hypothesis that the features are indeed enhanced for hard cases like occlusion and blur (from motion or from clouds). As a result, it raised the mAP for "overall" above R-FCN's "overall". We have to keep in mind that most VGG-16 layers are frozen during training, and that the final score would probably be much higher if this was not case. Nonetheless, our model convincingly surpasses the baseline in all categories, showing that features are enhanced not only for hard cases, but at all times. We outperform other video object detection for which we found results on UA-DETRAC, that is, 3D-DETNet [6] and RN-D-from-scratch [22]. The other video object detectors mentioned in the related works section did not produce results on this dataset.

Results on UAVDT

The results on UAVDT dataset are reported in table 6.2. We drew the ROC curves for our model, the baseline and few other state-of-the-art models in figure 5.5(b). Our detector outperforms all classic state-of-the-art models evaluated on UAVDT as well as the baseline by a significant margin. The mAP scores on this dataset are quite low compared to UA-DETRAC due to its very challenging lighting conditions, weather conditions and smaller vehicles. We show that by adapting the scales used for the anchor boxes on each dataset, we can greatly improve results. Also, our model shows results on UAVDT that are consistent with UA-DETRAC's results, having an improvement of $\sim$1.2 mAP points against the $\sim$1.4

Tableau 5.1 mAP reported on the UA-DETRAC test set compared to our baseline as well as classic state-of-the-art detectors. Results for "Ours" and "RN-VGG16" are generated using the evaluation server on the UA-DETRAC website, 3D-DETNet [6] is reported as in their paper, and others are as reported in the results section of the UA-DETRAC website. **Boldface** : best result, *Italic* : baseline.

Model	Overall	Easy	Medium	Hard	Cloudy	Night	Rainy	Sunny
RN-VID (Ours)	**70.57**%	87.50%	75.53%	**58.04**%	**80.69**%	69.56%	56.15%	83.60%
R-FCN [32]	69.87%	**93.32**%	**75.67**%	54.31%	74.38%	**75.09**%	**56.21**%	**84.08**%
RN-VGG16	69.14%	86.82%	73.70%	56.74%	79.88%	66.57%	55.21%	82.09%
EB [67]	67.96%	89.65%	73.12%	53.64%	72.42%	73.93%	53.40%	83.73%
Faster R-CNN [15]	58.45%	82.75%	63.05%	44.25%	66.29%	69.85%	45.16%	62.34%
YOLOv2 [33]	57.72%	83.28%	62.25%	42.44%	57.97%	64.53%	47.84%	69.75%
RN-D [22]	54.69%	80.98%	59.13%	39.23%	59.88%	54.62%	41.11%	77.53%
3D-DETnet [6]	53.30%	66.66%	59.26%	43.22%	63.30%	52.90%	44.27%	71.26%

on UA-DETRAC.

5.5 Discussion

To explain the gains we get from our model, we now discuss a few reasons why aggregating features from adjacent frames is beneficial.

5.5.1 Analysis

Small objects : The smaller the object, the harder it will be to detect and classify, as a general rule. There is a large number of small objects in the evaluated datasets, as there is in traffic surveillance scenes in general. Having multiple frames allows RN-VID to see the object from slightly different angles and lighting conditions, and a trained network can combine these frames to obtain richer features.

Blur : Blur is omnipresent in traffic surveillance datasets due to road users' constant motion (motion blur), and weather/lighting conditions. A blurred object can be harder to classify and detect. Since its appearance is changed, the network could recognize it as none of the predetermined labels, and not considering it as a relevant object. Having multiple slightly different instances of theses objects allows the network to refine the features and output finer features to the classification sub-network, finer than each single frame separately. It could also simply choose the best frame is that seems useful. A convincing example of our model performing better in blurry conditions is the "Cloudy" category in which it got the best

Tableau 5.2 mAP reported on the UAVDT test set compared to our baseline as well as classic state-of-the-art detectors. Results for "Ours" and "RN-VGG16" are generated using the official Matlab toolbox provided by the authors, others are reported as in their paper. **Boldface** : best result, *Italic* : baseline.

Model	Overall
RN-VID (Ours)	**39.43**%
RN-VGG16	38.26%
R-FCN [32]	34.35%
SSD [30]	33.62%
Faster-RCNN [15]	22.32%
RON [69]	21.59%

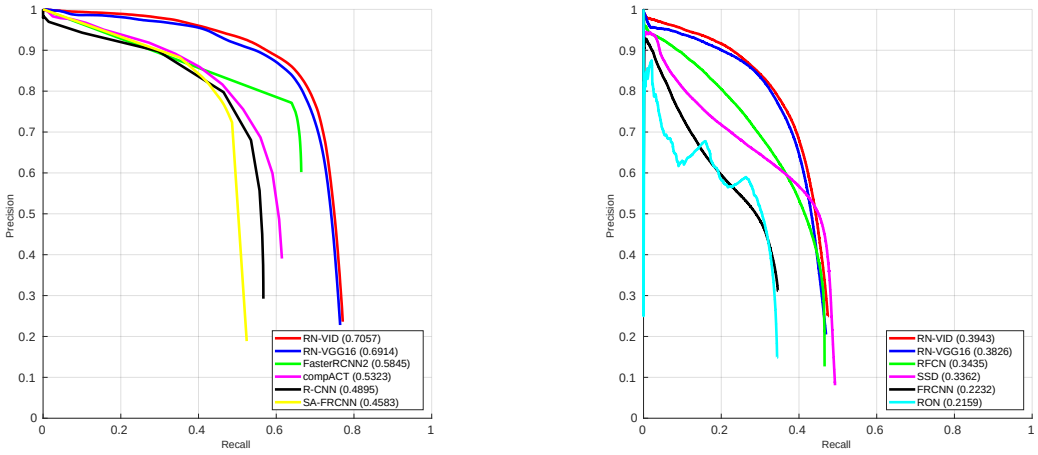
result.

Occlusion : Occlusion from other road users or from various road structures is very frequent in traffic surveillance scenes. Having access to adjacent frames gives our model a strong advantage against temporary occlusions, allowing it to select features from less occluded previous or future frames, making the detections more temporally stable. Figure 6.1 shows a qualitative example of our model performing better than the baseline in a case of occlusion.

5.5.2 Ablation Study

To properly assess the contribution of each part of our model, we performed an ablation study. We tried to isolate, as best as we could, our two contributions and looked at the impact of each of them. We justify the choice of using five consecutive frames with an experiment in which we varied this parameter on the UAVDT dataset. We tried several combinations and reported results in table 6.3. We can see than using five frames is better than using three, and that using three is better than using only one. We did not test with seven frames due to GPU memory issues.

To remove the contribution of the fusion module, we trained a model where instead of merging the feature maps, we would simply concatenate them and continue to build the feature pyramid as usual, by adjusting the kernel size of the convolutions to adapt to the new input size. Doing this actually degrades the performance a lot as shown by the RN-VID-NO-FUSION model in table 6.3. This is easily understandable by the fact that combining feature maps like this is noisy, and we might need much more data and parameters in order to make this work. We can conclude from this that the fusion module is an essential part of our model.



(a) (b)

Figure 5.5 Precision-Recall curves on UA-DETRAC [5] (a) and UAVDT [7] (b) for RN-VID (Ours), RN-VGG16 (Baseline) and a few other state-of-the-art methods.

Tableau 5.3 mAP reported on the UAVDT test set for different variations of our model to conduct an ablation study. Results are generated using the official Matlab toolbox provided by the authors. Number of frames is the number of frames used for each detection.

Model	num. frames	Overall
RN-VID (Ours)	5	**39.43**%
RN-VID	3	39.05%
RN-VGG16 (Baseline)	1	38.26%
RN-VID-NO-FUSION	5	26.95%

5.5.3 Limitations of our Model

A limitation of our model is for border situations, the first and last frames of a sequence where we cannot use our architecture to its full potential. However, this is not a problem since we can do a padding by repeating the first and last frame the number of times needed to without a real loss of performance. Also, it takes more memory to train the model then its single frame counterpart, due to the fact that we need multiple frames to train one single ground-truth example.

5.6 Conclusion

A novel approach for video object detection named RN-VID was introduced, composed of an object detection architecture and a fusion module for merging feature maps. This model

was trained and evaluated on two different traffic surveillance datasets, and compared with a baseline RetinaNet model and several classic state-of-the-art object detectors. We show that by using adjacent frames, we can increase mAP by a significant margin by addressing challenges in the traffic surveillance domain like occlusion, motion and general blur, small objects and difficult weather conditions.

Acknowledgments. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [RDCPJ 508883 - 17], and the support of Genetec. The authors would like to thank Paule Brodeur for insightful discussions.

CHAPITRE 6 ARTICLE 3 : FFAVOD : FEATURE FUSION ARCHITECTURE FOR VIDEO OBJECT DETECTION

Perreault H., Bilodeau GA., Saunier N., Heritier M.

Article submitted to Pattern Recognition Letters in april 2021

Abstract

A significant amount of redundancy exists between consecutive frames of a video. Object detectors typically produce detections for one image at a time, without any capabilities for taking advantage of this redundancy. Meanwhile, many applications for object detection work with videos, including intelligent transportation systems, advanced driver assistance systems and video surveillance. Our work aims at taking advantage of the similarity between video frames to produce better detections. We propose FFAVOD, standing for feature fusion architecture for video object detection. We first introduce a novel video object detection architecture that allows a network to share feature maps between nearby frames. Second, we propose a feature fusion module that learns to merge feature maps to enhance them. We show that using the proposed architecture and the fusion module can improve the performance of three base object detectors on two object detection benchmarks containing sequences of moving road users. Additionally, to further increase performance, we propose an improvement to the SpotNet attention module. Using our architecture on the improved SpotNet detector, we obtain the state-of-the-art performance on the UA-DETRAC public benchmark as well as on the UAVDT dataset. Code is available at `https://github.com/hu64/FFAVOD`.

6.1 Introduction

Object detection as meant in this paper is the task of finding rectangular bounding boxes that tightly bound objects of interest in an image. Video object detection uses the temporal information of a video in order to gain an edge over single frame detection. Object detection in general has been largely dominated by deep learning approaches in the last few years. It is quite a vibrant subject that has seen a lot of research. Its cousin task, video object detection, has seen a lot less activity on the other hand. To capitalize on video information, some methods aim to speed up the inference process by estimating feature maps [19], and some aim to merge feature maps by optical flow warping [18]. In contrast, in this paper,

we develop an end-to-end architecture and train it to merge feature maps without external methods or explicit knowledge of temporal relations.

The applications for video object detection are certainly not lacking, for instance in intelligent transportation systems, such as advanced driver assistance systems and video surveillance. In a world where robotics and automation are growing, the number of these applications can only increase.

Using multiple frames can improve results in situations that include occlusions, blur and smaller objects (see in figure 6.1). Indeed, there are several ways in which using multiple frames can help to detect objects on a target frame. In a video sequence, the objects of interest appear over and over again under different lighting, angles and occlusion conditions. Over a number of consecutive frames, one of them will contain the best view of an object of interest. One can even go even further, for each pixel location in our target frame, one of the temporally close frames will contain the best object features for this location. Of course, the features cannot be too far temporally because the object of interest might have moved too much in that case. FFAVOD aims to learn an operation to best merge the feature maps of several frames, at each image location.

We propose three main contributions : an adaptable video object detection architecture that can be inserted into multiple object detection methods, a module for the fusion of feature maps in order to enrich them, and an improvement to the SpotNet [2] attention module. The result of these contributions is a framework called FFAVOD, which is trainable end-to-end for video object detection and classification. This work generalizes the work published in [4] by extending it and testing it with several baseline object detector architectures.

The evaluation of our method is focused on traffic surveillance scenes, since they contain most of the challenges we aim to solve with our method and they are our target application. The method is evaluated on two datasets, UA-DETRAC [5] and UAVDT [7]. We tested our method by incorporating it into several base networks and comparing them to their corresponding baselines and other state-of-the-art methods on each dataset. A consistent and significant improvement over the base networks is demonstrated, as well as strong overall results on both datasets.

6.2 Related Work

6.2.1 Object Detection

The object detection benchmarks have been systematically dominated by deep learning-based methods in the last few years. They can be broadly divided in three categories : two-stage,

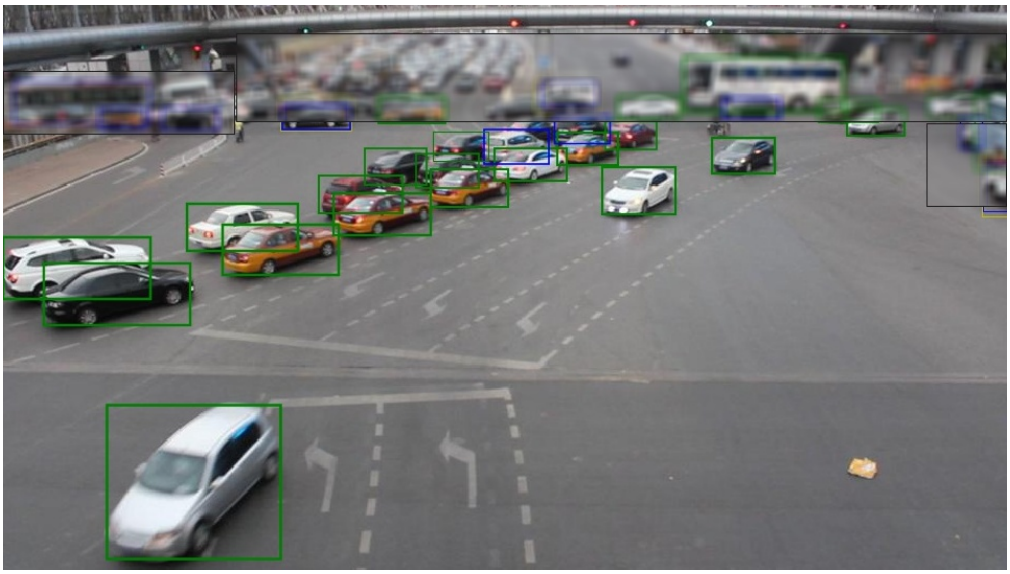


Figure 6.1 In this image from the UA-DETRAC dataset [5], one can see multiple examples where the proposed architecture applied on SpotNet [2] (blue) outperforms its baseline (yellow). Detections by both models are shown in green (considering a minimum IOU score of 0.8 to match them). Examples of improved performance include cases of occlusion and of smaller objects (top center and both top corners). The blurred rectangles represent regions excluded for the evaluation, but where the proposed architecture nonetheless can make better detections.

single-stage and anchor-free.

The two-stage category contains methods that use an object proposal phase where object candidates are proposed, and then refined into final predictions. R-CNN [28] is the first dominant detector to make use of a CNN. However, the object proposal method it uses is an external one, selective search. The CNN is used to extract features for every object proposal. Those features are then classified using an SVM. Fast R-CNN [29] improves upon it by passing the image into a CNN only once, and then cropping the features from the resulting feature map for the corresponding region of the image for each object proposal. The third iteration of the method, Faster R-CNN [15], removes the external object proposal method. It does so by using two CNNs, one that proposes object candidates called the region proposal network (RPN), and another that classifies and refines the bounding box for each candidate. The two CNNs share most parameters, making Faster R-CNN very efficient with accurate results. R-FCN [32] is a variant of Faster R-CNN, where the objects are detected as a grid of parts of objects where each cell of the grid votes, making it better for accurately positioning objects. The method Evolving Boxes [67] is an efficient vehicle detection network that includes a proposal sub-network and an early discard sub-network. It generates candidates with multiple representations and later refines and classifies those candidates.

The single-stage category improves over the two-stage methods by speeding up the inference process to eventually arrive to real-time detection. Single-stage refers to the removal of the object proposal phase. The original method YOLO [16] is the first CNN-based method to reach real-time speed. It divides the image into a grid, and makes each cell predict two bounding boxes using regression. Two subsequent versions of YOLO were proposed, YOLOv2 [33] and YOLOv3 [34], with various improvements. SSD [30] addresses the challenge of detecting objects at multiple scales by using feature maps at multiple levels in the CNN. A sliding window approach is then used to perform classification and regression with anchor boxes at fixed aspect ratios and scales. RetinaNet [3] is an improvement upon SSD that uses a different loss function, the focal loss, that aims to fix the asymmetry between positive and negative examples during training. RetinaNet also incorporates a feature pyramid network (FPN) [61], a network that builds a pyramid of features at different scales by using lateral and vertical connections on feature maps of the CNN. Using a sliding window, RetinaNet then classifies and regresses on these pyramid levels using anchor boxes.

More recently, a different approach to object detection was proposed. It is called anchor-free since it replaces the use for anchor boxes by detecting objects as keypoints. CornerNet [36] first introduced this approach by detecting objects as a pair of keypoints, the top-left and the bottom-right corners. A learned embedding allows the method to later pair the corresponding corners using the similarity between the embedding vectors. CenterNet (keypoint triplets) [17] builds upon this idea by adding a third learned keypoint, the center of the object, which is used to remove false positives, since two corners without a center are not likely to be part of an object. CenterNet (objects as points) [1] uses a different approach, and instead trains a network to detect objects as a single center keypoint, using center heatmaps for each label, as well as regressions for the width and height and the offset. A variant of CenterNet (objects as points), SpotNet [2], makes use of semi-supervised segmentation annotations to train a self-attention mechanism within the network and thus increase its performance.

6.2.2 Video Object Detection

A first way to perform video object detection is to combine the features of several frames. Flow Guided Feature Aggregation (FGFA) [18] makes use of optical flow warping to merge temporally close frames to improve accuracy. In MANet [39], an optical flow estimation is done and two networks are trained, one to do pixel-level calibration (detailed motion adjustments) and another one for instance-level calibration (global motion adjustments). Some works take advantage of recurrent neural networks, for example STMM [40] that models the motion and appearance of an object within a video sequence. In [19], Long Short-Term Me-

mories (LSTMs) are used to interpolate feature maps, which increases the inference speed greatly. Multi-frame Single Shot Detector [41] builds upon SSD by adding a temporal information with a recurrent convolutional module. [42] used deformable convolutions to compute offsets between temporally close frames. Using these offsets they can share some features from neighboring frames to better perform detection. 3D-DETNet [6] makes use of 3D convolutions on temporally close frames that are concatenated in order to produce better feature maps. [22] experimented with training networks on image pairs. Since no pre-trained weights were available, they could only outperform the single frame baseline when training from scratch. The same problem is faced when using 3D convolutions. In contrast to these previous methods, we also merge feature maps, but instead of aligning the feature maps or using 3D convolutions, we train a network to fuse directly the raw features maps from several frames.

Another possible avenue is to combine detection and tracking. TrackNet [43] extends the Faster R-CNN framework by directly detecting a 3D cube bounding a moving object. In Joint detection and tracking in videos with identification features [44], the authors train a multi-task model by joint optimization of detection, tracking and re-identification. The Global Correlation Network [45] also jointly trains the detection and the tracking task by first training the detection module before fine-tuning the whole network. Finally, motion information can be integrated to the network. Illuminating Vehicles With Motion Priors For Surveillance Vehicle Detection [46] integrates motion in a network in order the illuminate the vehicles and suppress false positives. To better detect tiny objects, MMA [47] proposes a dual stream network, an appearance stream and a motion stream. They also integrate a memory attention module to help select discriminating features using the temporal information. MFMNet [48] uses a motion from memory module to encode the temporal context. This module contains a separate dynamic memory for every input sequence, and produces motion features for every frame.

6.2.3 Feature Fusion Strategies

Since we consider combining the features of several frames, we also briefly review common feature fusion strategies. Most methods consider the concatenation of features or their sum. In the inception network [76], convolutions with kernels of various sizes are used to produce a set of feature maps which are then combined using concatenation. In the feature pyramid network [61], pyramid levels are combined using upsampling and addition, using a top-down approach. In FSSD [30], a variant of the feature pyramid is proposed where feature maps from all levels are first concatenated together, and later used to create all the pyramid levels. In ExFuse [80], low-level and high-level semantic information are merged after introducing

spatial information into the high-level features and semantic information into the low-level features. The feature maps are combined using additions.

6.3 Proposed Method

The task we aim to solve can be described as placing a bounding box and label on every object of interest in a target image, using the target image as well as n frames both before and after the target frame. To do so, two main contributions are presented. First, we design an architecture for object detection that allows the use of temporally close frames and that is adaptable to multiple base networks. Second, we propose a fusion module to merge feature maps of the same dimension from temporally close frames. In order to achieve high accuracy with less training, we specifically designed this architecture to use pre-trained weights from state-of-the-art single frame methods. In order to achieve state-of-the-art detection results, we also propose an improvement to the SpotNet attention module.

6.3.1 Frame Fusion Architecture

The idea behind our architecture is to compute feature maps for frames in a video only once for each frame and, when performing detection for a target frame, use the feature maps that are already computed for the temporally close frames to enhance the target frame feature map. Thanks to this idea of computing once and reusing, we also propose a novel fusion module that is used deep in the base network. We do not have to merge early in order the save computation time since we reuse the feature maps for detecting in the next frames. It is inserted between the backbone and the regression and classification heads of a base object detection network.

Our frame fusion architecture for video object detection (FFAVOD) takes multiple images as input, and they are merged as one feature map deeper in the network, as shown in figure 6.2. For a target frame at time t, we use a window of $2n + 1$ frames, that is n frames before and after the frame t. We cannot use a n that is too high because the positions of objects of interest at the different frames will become too different and it will not be possible to share the information. When facing "boundary conditions", meaning that we are too close to the beginning or end of a sequence to take frame $t - n$ or $t + n$ for instance, we simply duplicate the first or last frame. For example, not having access to $t - 2$ but to $t - 1$, with $n = 2$, we would use the five frames $t - 1$, $t - 1$, t, $t + 1$ and $t + 2$.

The way feature maps are merged is adapted to each base detector. For example, for a network like RetinaNet, the three outputs used to create the feature pyramid network are merged.

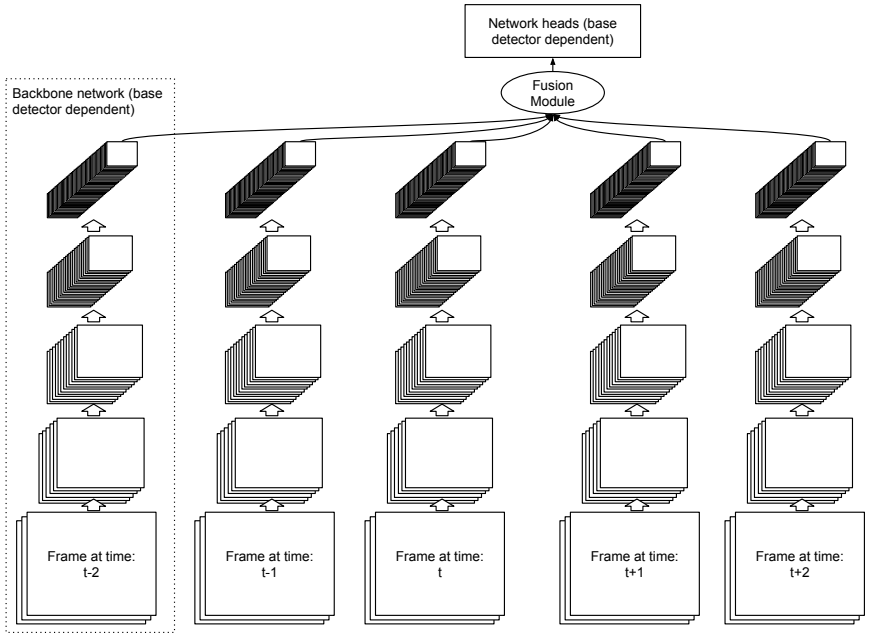


Figure 6.2 A visual representation of FFAVOD with a window of 5 frames ($n = 2$). Frames are passed through the backbone network of the base object detection network, and the fusion module takes their outputs as input. Finally, the fusion module outputs a fused feature map compatible with the base object detection network, and the base object detection heads are applied to the fused feature map to classify the object categories and regress the bounding boxes.

For networks like CenterNet, the outputs of the double stacked hourglass network used as backbone are merged. The merged feature map is used to create the center keypoint heatmaps. Details about which layers are merged are provided for three networks in section 6.4.1.

The fusion process is done with our custom fusion module described below, that enhances the target frame feature map. During the inference process, we can reuse feature maps already computed as we progress sequentially in the video, thus saving time for computing detections at every frame. However, during the training process, multiple images must e used for every ground-truth example, thus requiring more memory and time to train. We do believe that the extra time required to train the model is worth the better detection performance.

6.3.2 Fusion Module

A small trainable module is implemented to merge feature maps of temporally close frames (see figure 6.3). The idea for combining feature maps is the intuitive way a human would approach the task. For instance, when you look at one location for a given channel, you might want to average responses over the feature maps from the different frames, or look for the

maximum response and only keep this one (like in a max pooling). That would mean the feature would come from the frame where it is best seen. Taking the average responses or the maximum response for merging the feature maps depends on the situation. Therefore, we let the neural network learn the merging operation by itself, using 1×1 convolutions over the channels.

1×1 convolutions are often used to adapt the dimension of feature maps. In GoogLeNet, 1×1 convolutions are used to reduce the dimension of tensors, allowing the network to be deeper and remaining efficient. Recently, the network BorderDet [81] uses 1×1 convolutions to learn to produce border sensitive feature maps. In contrast to these previous works, we use 1×1 convolutions to combine features from several frames.

The fusion module receives $2n + 1$ feature maps, each of dimension $w * h * c$, as input (see figure 6.3). Its output is one merged feature map of dimension $w * h * c$. The feature maps taken as input come from the same backbone networks that share parameters. For $2n + 1$ feature maps of dimension $w * h * c$, we slice every c channels and concatenate them. The result is c tensors of shape $w * h * (2n + 1)$. A two dimensional convolution is performed on these tensors with a kernel of shape 1×1 $(1 * 1 * (2n + 1))$ and an output depth of 1, resulting in c tensors of shape $w * h$. We finally concatenate the resulting tensors channel-wise to obtain the $w * h * c$ feature map that is our output. This module thus learns the optimal operation to combine feature maps for the domain on which it is fine tuned.

6.3.3 SpotNet improvement

In order to further push the detection accuracy, we propose an improvement to the SpotNet [2] attention module. Instead of using three 3×3 convolutions to produce the saliency map, we designed and trained a small U-Net segmentation network. The U-Net has four levels and thus reduces the spatial resolution by half four times while doubling the channel resolution four times also, before reversing these changes and returning to the original resolution. This allows the network to produce finer saliency maps and improves detection accuracy.

6.4 Experiments

6.4.1 Base object detectors

The proposed architecture is implemented with several base object detector networks. It was first tested using RetinaNet [3] for its speed and accuracy, along with two other state-of-the-art object detection methods, CenterNet (objects as points) [1] and SpotNet [2]. For the

RetinaNet base model, the backbone is a VGG-16 network [24]. For the CenterNet and SpotNet base models, the backbone is a stacked hourglass network [14].

For RetinaNet, we merged the three outputs that are used to create the feature pyramid network. The network is the same otherwise. For CenterNet and SpotNet, we merged the outputs of the double stacked hourglass network used as backbone for these detectors. The merged feature map is used to create the center keypoint heatmaps, but the other heads use the target frame feature map. Experiments showed that this worked better than if all the heads used the merged feature maps, maybe due to the fact that the center heatmaps use general spatial features more, and the other heads might be more specialized in some specific semantic features in the feature map that do not answer well to merging between frames. The fusion process is done with our custom fusion module as described above, that enhances the target feature map by merging. Illustrations of the three evaluated models can be found at `https://github.com/hu64/FFAVOD`.

6.4.2 Datasets

Since our method relies on temporally close frames, it must be assessed on video datasets. The chosen evaluation domain is traffic surveillance, since it is of great interest to us and there are many applications for video object detection. Two video datasets in the traffic surveillance domain were selected : UA-DETRAC [5] (recorded with a fixed camera) and the Unmanned Aerial Vehicle Benchmark (UAVDT) [7] (recorded with a mobile camera). The versatility of our architecture is demonstrated by using videos from both fixed and mobile cameras. These two datasets are largely different, UA-DETRAC contains sequences taken by cameras fixed above highways and intersections, with medium sized objects. UAVDT, on the other hand, contain sequences taken by drones that hover over roads at different altitudes, but generally with much higher viewpoints than UA-DETRAC. Therefore the objects are significantly smaller and denser in that dataset, and also the background is changing across the sequence, making it more challenging to merge feature maps.

6.4.3 Implementations Details

The neural networks are implemented in Keras [77] using the TensorFlow [78] backend for our RetinaNet base detector. Our CenterNet and SpotNet base detectors are implemented using Pytorch [82].

The same training protocol was used for the all the base detectors to demonstrate the contribution of our approach. The training process is done in two steps. First, the base detector

is fine-tuned on each dataset starting from pre-trained weights on MS COCO [27]. Second, the shallower layers of the backbone are frozen and the fusion module as well as the network heads are trained. The reason for freezing the shallower layers is that the network seems to have difficulty training the fusion module while also training every other layers. Doing it in two steps seems to facilitate the learning.

A VGG-16 backbone with a feature pyramid of five levels is used for RetinaNet. RetinaNet takes the outputs of the last three blocks of VGG-16 to build the five-level feature pyramid, three levels of the same dimension of the three VGG-16 blocks, and two smaller. The fusion module is inserted between the VGG-16 and the feature pyramid. As a result, our fusion module is duplicated three times in the network.

A double stacked hourglass network is used as the backbone network for CenterNet and SpotNet. The fusion module is inserted after the end of the second hourglass. During training, the five frames are therefore passed through the same double stacked hourglass (all the parameters are shared), then passed through the fusion module, and the network continues as usual after that.

To determine the number of frames n used by our model, an ablation study is performed and the results are shown in figure 6.4. We end up using $n = 2$, i.e. a window of five frames in total, which showed the best performance.

6.4.4 Performance Evaluation

For the evaluation process, the test set is predetermined on each dataset. The training data is split into training and validation. The split is done by video sequence and not by frame to prevent overfitting on the validation data. The same split of three sets is employed for all of our experiments. We trained by monitoring the validation loss every epoch and select the best model according by the validation loss. Results were then computed on the test set.

The results are evaluated following the dataset protocols, using the Mean Average Precision (mAP). The mAP is the mean of the average precisions for every class. The average precision is the average precision under different recall values, which can also be described as the area under the precision-recall curve.

6.5 Results and Discussion

The results on the UA-DETRAC dataset are reported in table 6.1. The proposed FFAVOD applied to the three base detectors consistently outperforms them. The FFAVOD applied on

Tableau 6.1 mAP of FFAVOD applied to base detectors on the UA-DETRAC test set compared their respective base detectors, as well as classic state-of-the-art detectors. FFAVOD uses $n = 2$. Results for FFAVOD and their base detectors are generated using the official toolkit from the UA-DETRAC website. **Boldface** indicates the best result overall, Underline indicates the best result within a section, while *Italic* indicates the baseline and *indicates the use of multiple frames.

	Detector	Overall	Easy	Medium	Hard	Cloudy	Night	Rainy	Sunny
SpotNet	*FFAVOD-SpotNet with U-Net	**88.10%**	**97.82%**	**92.84%**	**79.14%**	**91.25%**	**89.55%**	**82.85%**	91.72%
	SpotNet [2] with U-Net	87.76%	97.78%	92.57%	78.59%	90.88%	89.28%	82.47%	**91.83%**
	SpotNet [2]	86.80%	97.58%	92.57%	76.58%	89.38%	89.53%	80.93%	91.42%
CenterNet	*FFAVOD-CenterNet	86.85%	97.47%	92.58%	76.51%	89.76%	89.52%	80.80%	90.91%
	CenterNet [17]	83.48%	96.50%	90.15%	71.46%	85.01%	88.82%	77.78%	88.73%
RetinaNet	*FFAVOD-RetinaNet	70.57%	87.50%	75.53%	58.04%	80.69%	69.56%	56.15%	83.60%
	RetinaNet [3]	69.14%	86.82%	73.70%	56.74%	79.88%	66.57%	55.21%	82.09%
SOTA methods (multiple frames)	*Joint [44]	83.80%	-	-	-	-	-	-	-
	*Illuminating [46]	80.76%	94.56%	85.90%	69.72%	87.19%	80.68%	71.06%	89.74%
	*MMA [47]	74.88%	-	-	-	-	-	-	-
	*Global [45]	74.04%	91.57%	81.45%	59.43%	-	78.50%	65.38%	83.53%
	*Perceiving Motion [48]	69.10%	90.49%	75.21%	53.53%	83.66%	73.97%	56.11%	72.15%
	*RN-D [22]	54.69%	80.98%	59.13%	39.23%	59.88%	54.62%	41.11%	77.53%
	*3D-DETnet [6]	53.30%	66.66%	59.26%	43.22%	63.30%	52.90%	44.27%	71.26%
SOTA methods (single frame)	FG-BR_Net [64]	79.96%	93.49%	83.60%	70.78%	87.36%	78.42%	70.50%	89.8%
	HAT [63]	78.64%	93.44%	83.09%	68.04%	86.27%	78.00%	67.97%	88.78%
	GP-FRCNNm [62]	77.96%	92.74%	82.39%	67.22%	83.23%	77.75%	70.17%	86.56%
	R-FCN [32]	69.87%	93.32%	75.67%	54.31%	74.38%	75.09%	56.21%	84.08%
	EB [67]	67.96%	89.65%	73.12%	53.64%	72.42%	73.93%	53.40%	83.73%
	Faster R-CNN [15]	58.45%	82.75%	63.05%	44.25%	66.29%	69.85%	45.16%	62.34%
	YOLOv2 [33]	57.72%	83.28%	62.25%	42.44%	57.97%	64.53%	47.84%	69.75%

SpotNet achieves the state-of-the-art (SOTA) result on this dataset. Our approach improves the performance of the base detectors particularly well on harder categories like “hard” and “rainy”, hinting that indeed our FFAVOD allows the network to overcome challenges present on the harder examples, but has less impact on the easier ones. There is a significant improvement of the detection results when using our FFAVOD with the RetinaNet detector as well as on the CenterNet detector. It is also true for SpotNet, but the improvement is smaller. This shows that our proposed feature fusion architecture can capitalize on multiple frames to improve object detection, and that our approach is applicable to several networks. In general, the performance is never reduced when using FFAVOD, except in the “Sunny” case for SpotNet where the mAP is virtually identical, due to the examples being easy. Even though they achieve good results, no other detector using multiple frames can outperform the results that we obtained with modern detectors (FFAVOD-CenterNet and FFAVOD-SpotNet). This shows that our approach capitalizes better on the multiple frames since it can be integrated with SOTA single frame detectors to improve them. Also, our SpotNet with U-Net outperforms the original SpotNet by around 0.5 to 1% in both datasets, showing

Tableau 6.2 mAP of our FFAVOD applied to detectors on the UAVDT test set compared their respective base detectors. FFAVOD uses $n = 2$. Results for our FFAVOD and their base detectors are generated using the official Matlab toolkit provided by the authors. The other results are taken from their respective papers. **Boldface** indicates the best result overall, Underline indicates the best result within a section, while *Italic* indicates the baseline and *indicates the use of multiple frames.

	Detector	Overall
SpotNet	*FFAVOD-SpotNet with U-Net	**53.76%**
	SpotNet [2] with U-Net	53.38%
	SpotNet [2]	52.80%
CenterNet	*FFAVOD-CenterNet	52.07%
	CenterNet [1]	51.18%
RetinaNet	*FFAVOD-RetinaNet	39.43%
	RetinaNet [3]	38.26%
SOTA methods	LRF-NET [68]	37.81%
	R-FCN [32]	34.35%
	SSD [30]	33.62%
	Faster-RCNN [15]	22.32%
	RON [69]	21.59%

the significance of using a better attention module.

The results on the UAVDT dataset are reported in table 6.2. Our FFAVOD detectors consistently outperform their respective base detectors. On the SpotNet architecture, the FFAVOD achieves also the state-of-the-art result on this dataset.

The improvement is smaller on SpotNet. Our hypothesis for why that is is that the attention module of SpotNet might help detect many of the same objects (small or occluded) that our fusion module does. Nevertheless, the improvement is still significant. On UA-DETRAC the improvement is almost zero for the “Easy” category, but close 1% for the “Hard” category, which is very consistent with the idea that the fusion module helps overcome challenges on hard examples. On UAV, the improvement is consistent with UA-DETRAC. It is also important to note that the better we perform on a dataset, the harder it becomes to improve our results.

6.5.1 Ablation Study

An ablation study was designed to evaluate the contribution of the different parts of the proposed FFAVOD. The two contributions were isolated and analyzed. On the UA-DETRAC dataset using SpotNet, the fusion module was removed and replaced with a simple concatenation followed by a convolution. Instead of channel by channel grouping and convolution then reordering, we simply concatenate the feature maps and reduce the dimension with a convolution. Replacing the fusion module with concatenation decreased the performance by a large margin as shown in table 6.3 (Concatenation). We believe that combining feature maps in this fashion is noisy and the model might require more parameters in order to learn how to combine them. The fusion module on the other hand is a much more direct and efficient way of combining feature maps. As baseline fusion methods, we also tried using a mean, maximum and a median operation, and we show that the results are worse than the single frame method. Additionally, to justify the use of past and future frames, we did an experiment where we retrain our model using $2n$ past frames with the target frame for $n = 1$ and $n = 2$. This decreased performance dramatically, as the fusion module has a hard time focusing on the target frame and the positions of the objects get too different (features are not aligned). Indeed, the distribution of the spatial information is no longer centered around the target frame, and this causes misalignment of the bounding boxes.

In order to select the number of frames, FFAVOD was applied by varying the number of frames used by the network (see figure 6.4). Each model using a particular n is re-trained by freezing the base network weights and training only the fusion module weights. We show that we obtain the highest result by using 5 frames, and thus that is what we used to produce our main results.

6.5.2 Limitations

One obvious limitation of our architecture is the fact that it needs video sequences with temporally close frames in order to work. This limitation is mitigated by the fact that a lot of applications for object detection rely on such data. Another limitation is the memory usage increase in inference, where FFAVOD has to keep feature maps stored in memory for several iterations before releasing them when they are no longer in the target frame window. This makes our architecture not ideal for embedded applications. Finally, without some sort of tracking, the temporal scope is also somewhat limited, making bigger improvements in accuracy difficult.

Tableau 6.3 Different fusion strategies to conduct an ablation study. Results are generated using the official Matlab toolbox provided by the authors. **Boldface** indicates the best result overall

n	Fusion Method	mAP
2	Learned (ours, as proposed)	**88.10%**
0	None (baseline)	87.76%
2	Max	87.09%
2	Mean	87.09%
2	Median	87.08%
2	Concatenation	83.44%
2	Learned (ours) (past frames only)	75.20%
1	Learned (ours) (past frames only)	76.02%

6.6 Conclusion

We introduce FFAVOD, a new method for video object detection which can be applied and used with most standard object detectors. Using the proposed approach, we trained and evaluated our fusion module on two datasets from the traffic surveillance domain. We demonstrate with three different base detectors that performance can be significantly increased by helping solve challenges in the harder examples of the datasets. Additionally, we propose an improvement to the SpotNet attention module that increases the detection accuracy. Several ideas may be tried to further increase performance, for example by adding temporal coherence in the form of re-identification of features across frames, or by integrating tracking in parallel to detection.

6.7 Acknowledgment

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), [RDCPJ 508883 - 17], and the support of Genetec.

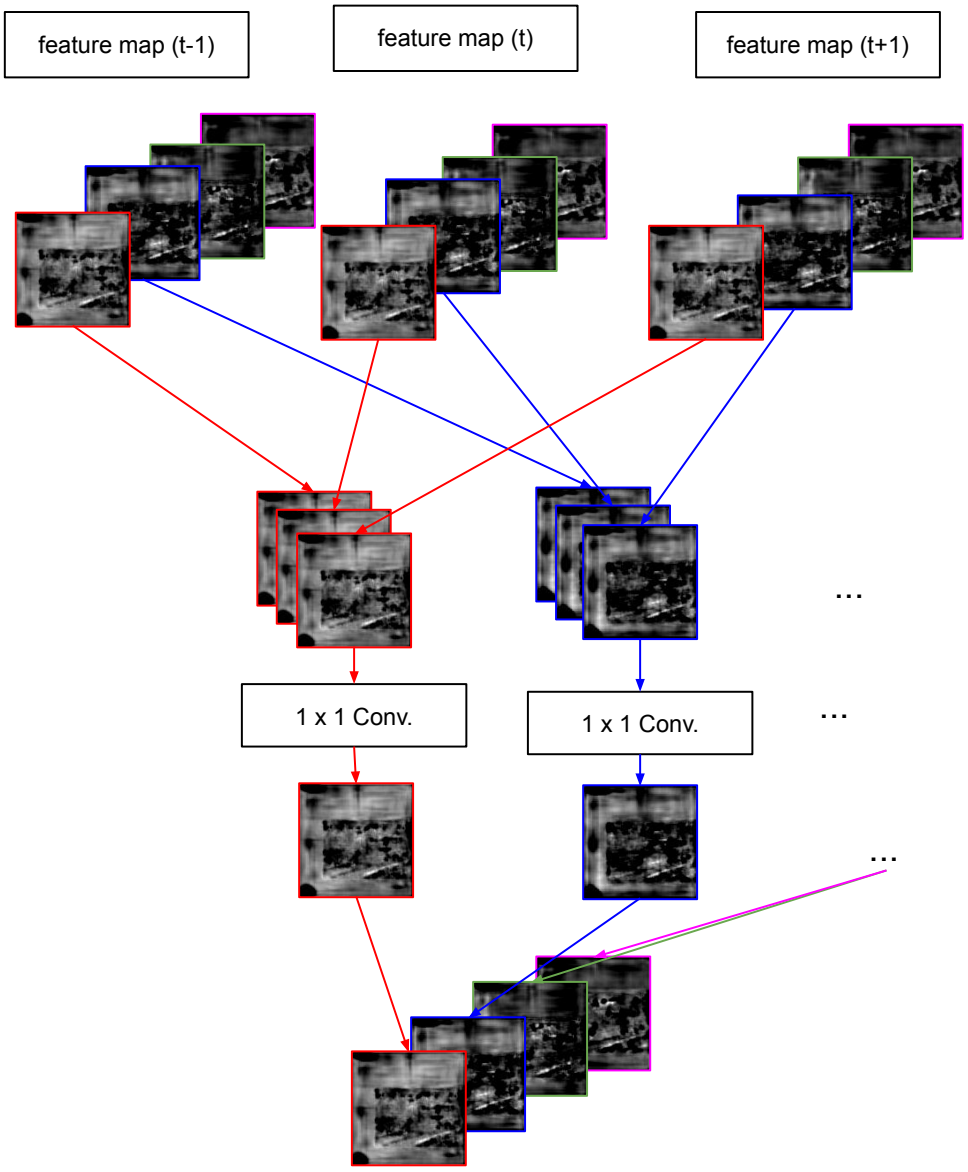


Figure 6.3 The fusion module. Channels are represented by colors. The fusion module is composed of channel grouping, concatenation followed by 1×1 convolution and a final re-ordering of channels.



Figure 6.4 mAP reported on the UA-DETRAC test set for different values of n of FFAVOD

CHAPITRE 7 ARTICLE 4 : CENTERPOLY : REAL-TIME INSTANCE SEGMENTATION USING BOUNDING POLYGONS

Perreault H., Bilodeau GA., Saunier N., Heritier M.

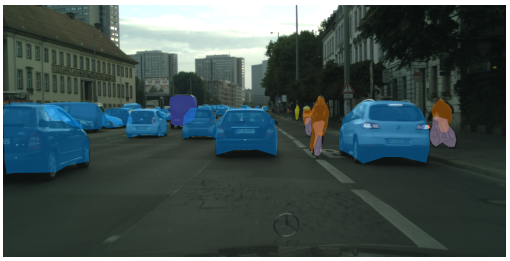
Article submitted to ICCV 2021 in march 2021

Abstract

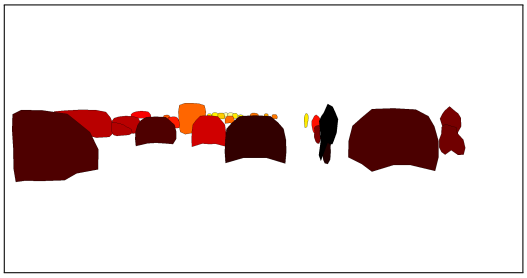
We present a novel method, called CenterPoly, for real-time instance segmentation using bounding polygons. CenterPoly detects objects by their center keypoint while predicting a fixed number of polygon vertices for each object, thus performing detection and segmentation in parallel. Most of the network parameters are shared by the network heads, making it fast and lightweight enough to run at real-time speed. To properly convert mask ground-truth to polygon ground-truth, we designed a vertex selection strategy to facilitate the learning of the polygons. Additionally, to better segment overlapping objects in dense urban scenes, we also train a relative depth branch to determine which instances are closer and which are further, using weak annotations. We propose several models with different backbones to show the possible speed / accuracy trade-offs. We trained and evaluated our method on cityscapes, KITTI and IDD and report our results on their public benchmark, which are state-of-the-art at real-time speeds.

7.1 Introduction

Object detection is typically described as the task of finding a rectangular bounding box around every object of interest in an image, as well as finding a class label for each object. A more complex and demanding task is instance segmentation, where instead of producing a rectangular bounding box, a pixel-wise segmentation is produced for each instance of the objects of interest in the image. As it is more complex, few instance segmentation methods can run as efficiently as object detectors. There is no doubt that real-time instance segmentation (at a frame rate faster than 20 FPS) is a very useful and an important problem to solve. Among the possible applications that would benefit from fast segmentation results are intelligent transportation systems, including traffic surveillance, automated driving and other advanced driving assistance systems. These applications often have to deal with dense overlapping objects such as cars that require a precise segmentation for various tasks such as



(a)



(b)

Figure 7.1 CenterPoly produces : (a) an accurate instance segmentation by detecting objects and regressing a bounding polygon for each and (b) a relative depth value for each object. In this example, polygons have 16 vertices.

counting, tracking and re-identification.

Instance segmentation is a harder problem than object detection because it involves an additional task that requires ideally pixel-level precision. Furthermore, adapting an object detector to perform instance segmentation efficiently is not trivial. Indeed, in YOLOv3 [34], Joseph Redmon famously stated :

> "Boxes are stupid anyway though, I'm probably a true believer in masks except I can't get YOLO to learn them."

In object detection, a box can be represented by four values, regardless of the box size. Therefore, this task can be performed quickly. In contrast, an accurate segmentation mask typically requires several hundred values. For example, Mask R-CNN [20] encodes each instance on COCO [27] by $14 \times 14 \times 80 = 15680$ values. Extracting masks thus requires more computations.

For this reason, for keeping our method fast and lightweight, we decided to use bounding polygons to represent the segmentation masks as a compromise on speed and accuracy between bounding boxes and pixel-wise segmentation masks. Polygons can be represented by a few vertices, a parameter which can be adjusted at will. For example, using 12 vertices to represent an object (i.e. 24 numbers) can produce a much more precise segmentation than a

bounding box at a much lower cost than using masks. But using polygons also brings limitations, among others the fact that a polygon cannot have holes or be fragmented and the fact that the segmentation accuracy is limited by the number of vertices. We mitigate these limitations with our relative depth map with which we can place an object instance polygon in front or behind others.

We approach instance segmentation by modifying a fast anchor-free object detector, CenterNet [1], to produce bounding polygons for each detected object, using a fixed number of vertices for each polygon. The resulting method, called CenterPoly, is almost as fast as CenterNet and the resulting masks are very accurate compared to other real-time instance segmentation methods. In figure 7.1, we show an example of the result of our method.

One of the weaknesses of bounding polygons compared to masks is the object overlapping problem. When two polygons overlap, there is an ambiguity regarding to which polygon the overlapping pixels belong to. To solve this problem, we added a head to our network that learns the relative depth of objects. The relative depth is sometimes available in the annotations, and we used a transfer learning strategy when it is not. We discuss this further in section 7.3.4. This head produces one value per pixel, and this value reflects the relative depth of the object with its center at this location.

In summary, we propose CenterPoly, a novel accurate real-time instance segmentation method that we evaluated on the cityscapes [12], KITTI [57] and IDD [9] datasets. Our contributions are as follows :

— We designed a network head to produce bounding polygons for every detected object in an image.
— To produce accurate ground-truth polygon vertices, we designed a vertex selection policy which proves to be very important for polygon regression.
— To solve the issue of overlapping instances, we trained the network to learn the relative depth of objects.
— We proposed two modifications to the CenterNet center heatmaps to better suit instance segmentation.

7.2 Related Work

As we use a segmentation by detection paradigm, it is important that we highlight the most important work in this field in the last few years.

7.2.1 Object detection

Two-stage detectors have a first phase where they find object candidates, and a second phase where they refine and choose the best of those candidates. It is the slowest object detection paradigm. The most notable two-stage detector is certainly the ground-breaking Faster R-CNN [15], which is the third iteration of the R-CNN [28] family. Faster R-CNN uses a shared backbone between two networks, a region proposal network (RPN) and a refinement network. The RPN proposes the best candidate bounding boxes by using anchor boxes of different predefined sizes and aspect ratios. The refinement network then classifies and refines the bounding box candidates, and keeps the ones with the highest scores.

One-stage detectors remove the object candidate search phase, and rather try the find objects at the same time as classifying them and refining their position and size. As a result, they are much faster than two-stage detectors. YOLO [16] was the first method to use this paradigm. It consists of a network to detect objects based on a division of the image into a regular grid, and having each cell predict a certain number of objects close to them. The two subsequent iterations of the method, YOLO9000 [33] and YOLOv3 [34], improved upon it by switching to anchor boxes and designing a better and deeper backbone network among others. SSD [30] introduced a scheme to merge feature maps at different resolutions before applying anchor boxes, and thus tackles the problem of detecting objects at different scales. RetinaNet [3] uses an architecture that is fairly similar to SSD, although they use a feature pyramid network [61] for multi-scale detection. They also introduce the focal loss, which is a loss designed to help counter the imbalance between positive and negative examples.

Anchor-free detectors refer to the relatively newer paradigm of detection by object keypoints. They can also be classified as one-stage detectors. CornerNet [36] trains a network to recognize top-left and bottom-right corners, using corner pooling. It also has an embedding branch so that if the top-left corner and the bottom-right corner belong to the same object, their embedding will be similar, and different otherwise. They use this embedding to pair corners. Keypoint Triplets [17] improves upon CornerNet by adding a third keypoint, its center. They use this center keypoint and its confidence score to remove false positive by testing each bounding box (paired corners) for a confident center keypoint at its center. They also improve the corner pooling layers. The "objects as points" approach [1] presents a surprisingly simple architecture, detecting objects as their center keypoint, as well as their height and width. SpotNet [2] builds upon objects as points by introducing a self-attention module trained with semi-supervised [1] annotations.

1. weak annotations rather

7.2.2 Instance Segmentation

The natural progression from bounding box detection is a finer pixel by pixel segmentation. Mask R-CNN [20] builds upon Faster R-CNN by adding a size invariant mask branch in the second phase. For each object candidate, it produces one mask for each possible label. The same improvement exists for RetinaNet, named RetinaMask [49]. In RetinaMask, a mask subnetwork is added to produce an instance mask for each object candidate.

Some methods perform instance segmentation by detecting a bounding polygon around objects. PolarMask [21] and Poly-YOLO [51] use a polar grid to represent vertices by their angles from the center. The main difference between them is that Poly-YOLO learns size invariant shapes using a normalized distance. Although our method shares some similarities with them, the use of angles make their network learn shapes, CenterPoly rather learns to detect points on object edges to segment them. Closer to our proposed method, Polygon-RNN++ [53] uses a CNN and RNN architecture to detect one polygon vertex at a time, and is therefore rather slow although accurate. PANet [50] produces accurate masks adding a bottom-up path augmentation to an FPN and adaptive feature pooling.

A few methods can produce instance segmentation in real-time, including Poly-YOLO, but these works remain scarce. Box2Pix [54] can produce results on cityscapes at 10.9 fps by combining bounding box detection as well as semantic segmentation, and predicting pixel offset to the object centers. The spatial sampling network method [55] can achieve very fast instance segmentation on cityscapes at 113 fps using a decoder network and thresholding at inference time. Finally, Yolact [56] works by producing a few segmentation prototypes and trains a network to learn the proper coefficient to combine them. It then uses bounding box detection to crop a region in the combined prototypes.

7.3 Proposed Method

CenterPoly is based on the CenterNet [1] object detector. CenterNet detects objects using three heads, one to produce heatmaps of the center of objects for each label, one to regress their width and height and one to regress the object offsets [2]. We propose to replace the width and height head by a polygon regression head, which will regress a fixed number of points around the object representing a bounding polygon. We modify the heatmaps to account for object proportions and to better deal with instance segmentation. We also add a head to model relative depth to find which objects are further or closer in the scene. This helps

2. The offset represents the displacement needed when upscaling the final output to the original image size.

in segmenting overlapping objects. This information is available implicitly in some dataset annotations. Indeed, the relative depth may be available if the annotators always annotated objects that are behind others before those in front, as it is the case in cityscapes [12] and IDD [9]. To train on datasets where this information is not available, it is possible to use transfer learning. We use this order of appearance of objects in the annotations as relative depth ground-truth. It is by no means an absolute depth or even a "full" annotation, as two objects with the same depth will have different relative depth value. Therefore, this can be considered as a weak annotation. Despite this, it has proven to be sufficient for our purposes as we will see in the results. An overview of our model can be seen in figure 7.2.

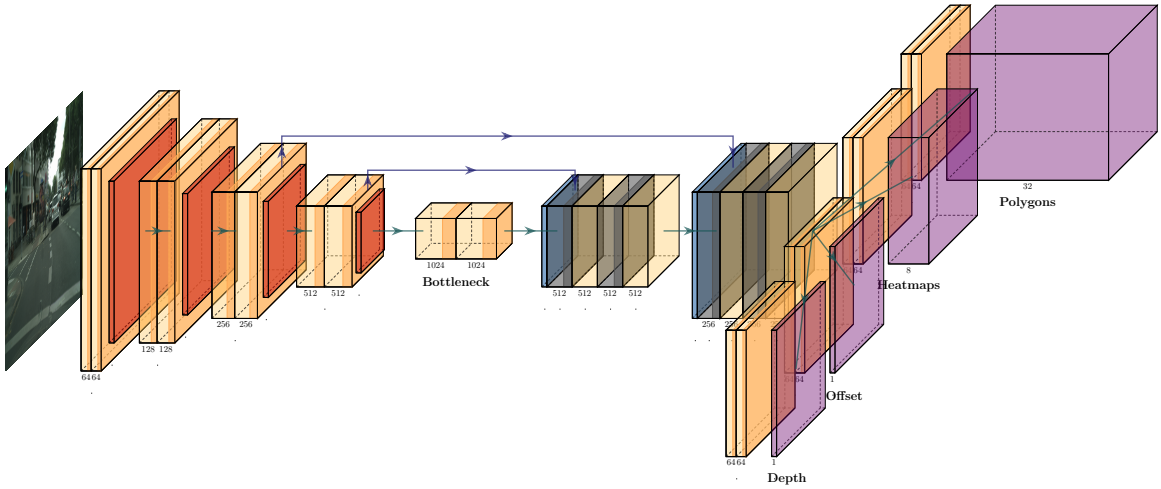


Figure 7.2 An overview of the CenterPoly architecture. The image first passes through a CNN backbone, displayed here as an Hourglass network. The feature map is then shared between four network heads, the polygon regression head, the center heatmaps head used for detections, the object offset head and the relative depth map head. The sizes displayed are for illustration purposes only, please refer to the code for the detailed architecture.

7.3.1 Modified Heatmaps

CenterNet uses ground-truth center heatmaps that are shaped like circular Gaussians to train the network. These kinds of heatmaps will penalize similarly the same error on the vertical axis and on the horizontal axis, even for thin rectangular boxes where an error is more costly on the box short axis. To help overcome this problem, we are instead using elliptical Gaussians that use the ratio of the rectangular bounding box to get the ratio of the small and the large radii of the ellipse. Our small radius is the same as the circular radius from CenterNet, while our large radius captures the object proportions to penalize center

detection errors more accurately.

The use of instance segmentation ground-truth has allowed us to make another tweak to the CenterNet heatmaps. Using the center of the bounding box as the object center can be misleading, as the object might be distributed unevenly in the bounding box (for instance, a pedestrian with an arm up). As an alternative, we use the object center of gravity by computing the mean of the vertex locations on the object contour. This gives us ground-truth centers that are more adapted for computing polygon vertices offsets as we will see in the next section.

7.3.2 Polygon Regression Head

Our polygon regression head is composed of one 3×3 convolution followed by a 1×1 convolution to reduce the dimension, and outputs $N \times 2$ floats at each location on the spatially downsampled feature map, N being the number of vertices used to define each polygon. The values represent offsets from the object center, paired in x and y coordinates. The points are arranged in such a fashion that the first one is always the one closer to the top left corner, and the subsequent ones are the following points of the polygon going clockwise. For instance, if we use a model with N vertices, the output of the polygon head at location (x, y) would be $(i_1, j_1, i_2, j_2, ..., i_N, j_N)$ with $(x + i_n, y + j_n)$ being the n^{th} polygon vertex.

Although this head produces dense polygon predictions, the polygons used for training are only the ones with associated objects. This means that predictions at locations where there are no object centers will not contribute to the loss. To achieve that, we use a maximum number of objects per image and a ground-truth mask to hide irrelevant predictions. We train this head using a standard $L1$ [3] loss, that is :

$$L_{poly} = L1Loss(poly, poly_{GT}), \tag{7.1}$$

where $poly$ represents the regressed polygons with associated objects and $poly_{GT}$ is the ground-truth.

7.3.3 Vertex Selection Policy

During our experiments, we found that the vertex selection was very crucial to the model being able learn the shapes of the polygons. The initial annotations come in two formats, images with label ids for each pixel, as well as bounding polygons (which are not as precise due to overlapping, fragmentation, etc.). Neither of those formats are suitable for us, the

3. L2 loss was also tried but did not produce as good results.

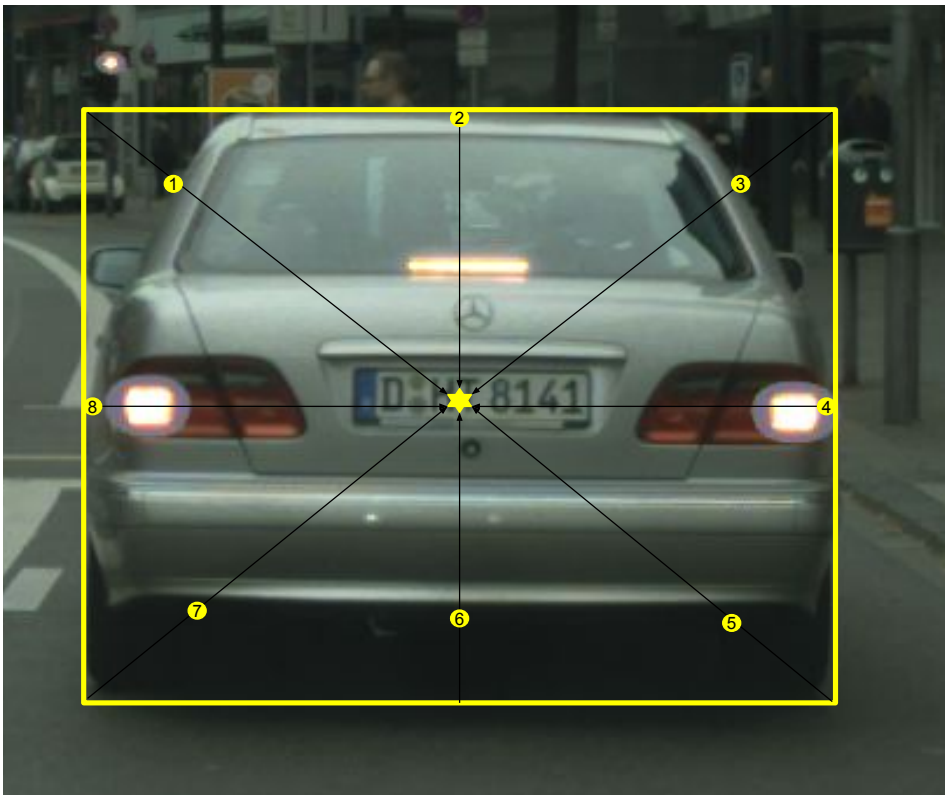


Figure 7.3 The vertex selection strategy : we trace lines at regular intervals in the bounding box, starting at the top left corner and going clockwise. The selected vertex is the first point on the line within the instance mask when going from the bounding box toward the center. The interval changes depending on the number of vertices used in the method.

images for obvious reasons and the polygons because the number of vertices is not fixed for each instance, and the density of points around the objects is far from uniform in most cases, meaning that more points could be on one side of the object.

The first strategy was to create polygons with a fixed number of vertices using the ground-truth bounding polygons, and either adding vertices between distant ones or removing vertices between close ones. However, the network had a really hard time learning these vertices because their positions varied too much on the object boundaries and could not produce good results.

The adopted strategy, shown in figure 7.3, was designed so that the network can better model what each vertex represents, so that each n^{th} vertex represents something similar for each instance. First of all, we calculate the tightest bounding box surrounding the instance mask. We divide a number of points equally between the four sides of the bounding box. Then, at these points on each side of the bounding boxes, we trace a line between the point on the

box and the center of the box. We find the first point that is non-zero on the instance mask, and add it to our ground-truth vertices. We then step at regular intervals along the sides of the bounding box and continue the process, until we reach the starting point. As a result of this process, each n^{th} vertex have approximately the same angle for all object instance, which facilitates the learning.

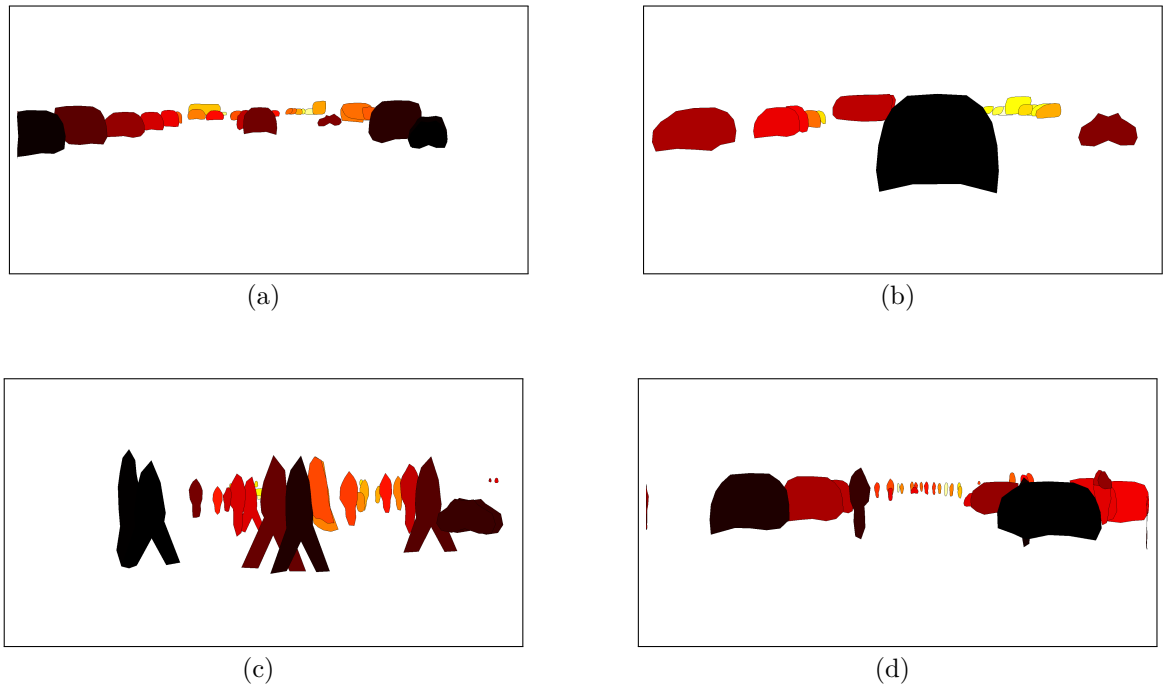
7.3.4 Relative Depth Head

Our method finds a bounding polygon for each instance in the image. However, this is not sufficient to achieve a good performance as a pixel can only be associated with one instance in the ground-truth. Therefore, for overlapping instances, we must decide which instance goes in front of which. In the cityscapes and the IDD annotations, the objects were recorded in order of distance, from furthest to the closest (generally). We use this relative depth as ground-truth to train a dense relative depth head that is trained over the whole image so that an object with the center (x, y) will have depth $D(x, y)$ where D is our two dimensional depth map. To train this branch, we use the same strategy as for the polygon regression branch, where only the values with associated objects will contribute to the loss. The training loss is :

$$L_{depth} = L1Loss(depth, depth_{GT}) \tag{7.2}$$

where $depth$ represents the regressed depth values with associated objects and $depth_{GT}$ is the ground-truth.

The architecture of our relative depth head is the same as the polygon regression head, specifically one 3×3 convolution followed by a 1×1 convolution to reduce the dimension, and it outputs one float at each location on the spatially downsampled feature map, the relative depth value. The smaller the value, the further the object. We use this relative depth value to place instances in front of other instances for high confidence instances only. As a result, an instance with a confidence score lower than a threshold will never hide another object, although the instance is still included in the results for evaluation. We can visualize in figure 7.4 the kind of relative depth map obtained in the form of heatmaps. Please note that the depth is relative from one instance to another and is not an absolute measure of the pixel depth as absolute depth information is not available.



(a) (b) (c) (d)

Figure 7.4 Qualitative relative depth predictions of CenterPoly on the cityscapes dataset. Darker is closer, and lighter is further.

7.3.5 Training

Our ground-truth vertices are produced beforehand, as described in section 7.3.3, and used for training our model. We train our polygon and relative depth head from scratch, and use pre-trained weights from MS-COCO [27] for the center heatmaps and the offset heads. Our heads are integrated in CenterNet and the model is trained end-to-end. Our loss function is as follows :

$$Loss = W_{hm} * L_{hm} + W_{poly} * L_{poly} + W_{depth} * L_{depth} + W_{offset} * L_{offset}, \quad (7.3)$$

where the heatmap loss L_{hm} (focal loss) and the offset loss L_{offset} ($L1$ loss) are the same as the ones in CenterNet. W_{hm}, W_{poly}, W_{depth} and W_{offset} are the respective weights. We talk in depth about the implementation details in section 7.4.2.

7.4 Experiments

We trained and evaluated our model on the cityscapes, KITTI and IDD datasets for the instance segmentation task. We report our results as they appear on their public benchmarks. We also did an ablation study on cityscapes.

7.4.1 Evaluation datasets

Cityscapes [12] is a dataset of street scenes captured in several cities in Germany. The image resolution is 2048 × 1024. A set of 8 image labels is defined which includes person, rider, car, truck, bus, train, motorcycle and bicycle. The other labels are ignored during the instance segmentation evaluation. Three image sets are predefined as train, validation and test. The provided annotations are only for the train and validation sets. The AP (average precision) used by the cityscapes dataset is the MS-COCO AP, defined as AP[50 :95] with steps of 0.05, which is the average of the AP values with minimum IOU (intersection over union) ranging from 0.50 to 0.95. Cityscapes also computes the AP for objects limited to a 50 meter range and a 100 meter range. KITTI Vision [57] for instance segmentation is a small dataset that consists of 200 train and 200 test images of street scenes with the same labels as cityscapes. The image resolution is approximately 1280 × 384, but can very slightly vary. KITTI uses the AP and the AP50 as defined above. IDD [9], India Driving Dataset, consists of street scenes images captured in India. The image resolution varies between 1920× 1080 and 1280× 964. The train/val/test split is approximately 7000/1000/2000 images respectively. This dataset also uses the AP and the AP50 as defined above.

7.4.2 Implementation Details

We implemented CenterPoly with Pytorch [82] and trained it for 240 epochs on a single GTX 1080 Ti using the adam optimizer. Our inference speeds are shown for this GPU, which is not the fastest on the market by far. For our three tested backbones, ResNet18 [13], DLA34 [83] and the hourglass [14], we used CenterNet weights pre-trained on MS-COCO . We then added the Polygon regression and depth map heads and continued the training on cityscapes. For IDD and KITTI, we started the training from cityscapes. The loss weights are $W_{hm} = 1$, $W_{poly} = 1$, $W_{depth} = 0.1$ and $W_{offset} = 0.1$. As we do not have any depth annotations for KITTI, we use fake depth annotations [4] and set $W_{depth} = 0$. We use a batch size of six, a learning rate of 2e-4 and drop the learning rate by a factor of ten at epoch 90 and 120 on each tested dataset. For the evaluation, we only consider objects with a confidence score of

4. We set the depth annotation values to zero.

over 0.5 for the depth map, which means no object with a confidence below that is allowed to hide another one. We trained at a resolution of 1024×512 as our GPU memory is limited. During training, we used standard data augmentation techniques such as random cropping and flipping. For more details on the implementation, please refer to our code.

The backbone used for CenterPoly main results is the Hourglass Network [14], as it proved to be very efficient for CenterNet and keypoint detection in general. The encoder-decoder architecture of the Hourglass has a very high expressivity that is useful for the multi-task training that CenterPoly does. However, in our efforts to reach real-time speeds, we reduced the number of stacks to only one compared to two in CenterNet. Also, we used 16 polygon vertices to segment instances (See section 7.5.1).

7.4.3 Results

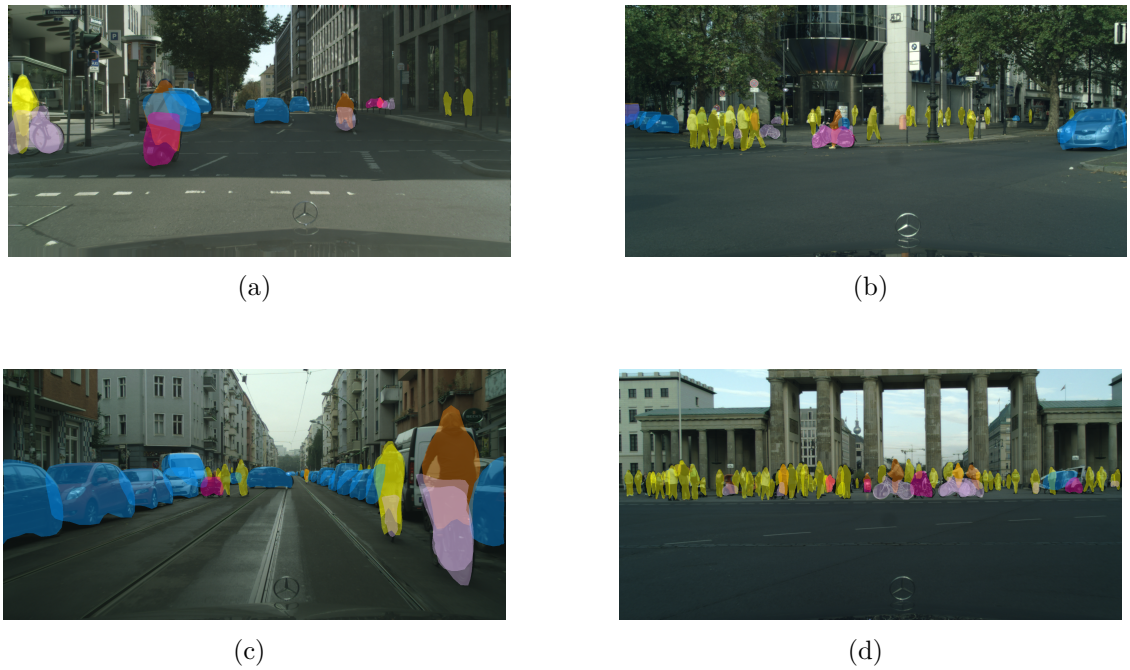
Our detailed results on the three datasets are presented in table 7.1 [5]. We compare CenterPoly to other fast methods that have been tested on each dataset. We also added some slower methods to compare as baselines. Even though we outperform every real-time method with the AP metric, CenterPoly outperforms the other real-time methods by an even larger margin for the AP50 metric. For instance on cityscapes, if for the AP, we outperform LCIS by only 0.44, CenterPoly outperforms it for the AP50 metric by 8.69. This shows that our masks are very accurate for a coarse segmentation, but slightly less for a very fine segmentation, which can be explained by the nature of polygons. On KITTI, although no other real-time instance segmentation method have submitted results, CenterPoly still presents some good results for real-time instance segmentation and presents a baseline for future methods to compare to. On the IDD dataset, we compare CenterPoly to other methods that have produced results on it, and presents state-of-the-art results at real-time speed.

We present some qualitative results in figure 7.5, in which we can notice that CenterPoly can accurately segment legs and arms of pedestrians, even for very small ones. Also, separating dense instances of very small objects can be done, showing that CenterPoly does indeed instance segmentation and not semantic segmentation. However, the segmented bicycles do show some limitations of using bounding polygons, as CenterPoly struggles to produce a very fine masks.

5. Inference time estimations are from the Box2Pix paper.

Tableau 7.1 Results on the cityscapes, KITTI and IDD test sets, as shown on their respective public benchmark, divided between faster (bottom) and slower (top) methods. When not available, runtimes were estimated. Mask type : Full : based on pixel-wise labels, polygon : based on a bounding polygon. Boldface : best results. Results for PANet and Mask R-CNN on IDD were taken from the original IDD paper [9]

Method	Mask type	AP	AP50%	AP100m	AP50m	Runtime(s)
Results on cityscapes						
PANet [50]	Full	**31.80**	**57.10**	**44.20**	**46.00**	-
Mask R-CNN [20]	Full	26.22	49.89	37.63	40.11	**0.2**
LCIS [84]	Full	15.10	30.80	24.20	25.80	> 0.2
InstanceCut [85]	Full	13.00	27.90	22.10	26.10	-
Recurrent Attention [86]	Full	9.50	18.90	16.80	20.90	> 0.33
CenterPoly (Ours)	Polygon	**15.54**	**39.49**	**23.33**	**24.45**	0.046
Box2Pix [54]	Full	13.10	27.20	-	-	0.09
Spatial Sampling Net [55]	Full	9.20	16.80	16.40	21.40	**0.009**
Poly-YOLO [51]	Polygon	8.70	24.00	-	-	0.046
Poly-YOLO lite [51]	Polygon	7.80	21.70	-	-	0.026
Results on KITTI						
UniDet_RVC [87]	Full	**23.19**	**49.13**	-	-	**0.3**
BAMRCNN_ROB	Full	0.68	1.81	-	-	1
CenterPoly (Ours)	Polygon	**7.55**	**22.94**	-	-	**0.046**
Results on IDD						
PANet [50]	Full	**37.60**	**66.10**	-	-	-
Mask R-CNN [20]	Full	26.80	49.90		-	**0.2**
CenterPoly (Ours)	Polygon	**14.40**	**36.90**	-	-	0.046
Poly-YOLO [51]	Polygon	11.50	26.70	-	-	0.049
Poly-YOLO lite [51]	Polygon	10.10	23.90	-	-	**0.027**



(a) (b)

(c) (d)

Figure 7.5 Qualitative instance segmentation results of CenterPoly on the cityscapes dataset.

7.5 Discussion

7.5.1 Ablation Study

In order to evaluate the possible speed / accuracy trade-offs, we trained our method using various, lighter backbones, namely ResNet18 [13] and DLA34 [83]. We also trained with various configurations to assess and demonstrate the benefits of our contributions. We performed our ablation study on the validation set of cityscapes, and show our results in table 7.2. We did not use the test set as evaluation on the server is restricted.

From the ablation study results, we can conclude the following. Firstly, we can notice that using the depth map is very useful for CenterPoly with all the tested backbones, allowing the network to better attribute pixels of overlapping instances. Secondly, the AP seems to plateau when more polygon vertices are used. This shows that a very large number of vertices is not necessary to capture the external contour of objects. This is why we chose to use 16 vertices in our final results. Thirdly, we can observe a decrease in AP when not using our elliptical center heatmaps and our center of gravity locations showing their benefit for localizing objects more accurately. Elliptical center heatmaps help in the case of long rectangular objects, as the loss penalizes less an error over the long axis than the short one. Using the center of gravity helps CenterPoly to regress the polygon vertices as center offsets, contrarily to CenterNet

Tableau 7.2 Ablation study of different parts of CenterPoly as well as performance with various backbones. AP and AP50% on the validation set of cityscapes. Depth refer to the use of our relative depth map. Elliptical refers to the use of elliptical GT. Center of gravity refer to the center heatmaps being at the center of gravity instead of the center of the bounding boxes.

Backbone	Depth	Elliptical	C. of Grav.	Nbr. Vertices	Res.	AP	AP50%	Runtime(s)
Hourglass	✓	✓	✓	32	1024×512	18.4	46.0	0.047
Hourglass	✓	✓	✓	16	1024×512	**18.5**	**46.2**	0.046
Hourglass	✓	✓	✓	8	1024×512	15.6	42.6	0.046
Hourglass	✓	✓	✗	16	1024×512	17.5	44.7	0.046
Hourglass	✓	✗	✓	16	1024×512	17.4	43.0	0.046
Hourglass	✗	✓	✓	16	1024×512	17.8	45.4	0.046
Hourglass	✓	✓	✓	16	512×512	10.8	25.3	**0.026**
DLA34	✓	✓	✓	16	1024×512	**11.3**	**30.1**	0.021
DLA34	✗	✓	✓	16	1024×512	10.9	29.6	0.021
DLA34	✓	✓	✓	16	512×512	7.7	19.9	**0.012**
ResNet18	✓	✓	✓	16	1024×512	**8.9**	**23.3**	0.016
ResNet18	✗	✓	✓	16	1024×512	8.7	23.2	0.016
ResNet18	✓	✓	✓	16	512×512	5.2	13.3	**0.01**

that regresses a width and height. Fourthly, the best speed / accuracy trade-off comes from using a lighter backbone rather than reducing the resolution, as it makes the detection and segmentation of small instances harder. However, instance segmentation is a very complex task and we can see that using lighter backbones reduces the performance a lot. We can conclude that we require a certain level of expressivity in our backbone networks to maintain a good performance. Finally, the number of polygon vertices (from 8 to 32) does not affect significantly the speed of our method.

7.5.2 Limitations

One of the limitations of our method for instance segmentation is of course the fact that we do not obtain very fine masks, the precision of our masks is limited by the number of vertices of the polygons that we use as well as because they cannot include holes. Still, we can see in figure 7.5 that our method has enough precision to segment the arms and legs of pedestrians. Nevertheless, this limitation explains the fact that we rank better in AP50 than in AP, which is a very strict metric.

Another limitation is that due to training and network design, a particular trained model will always output the same number of vertices for every polygon, even if it does not need to. There might be some future work to do in that direction to try to further speed up the method by using fewer vertices when not needed. Poly-YOLO tackles this problem by introducing a confidence score for each vertex, but this is not a direction we wanted to take as it introduces yet another value and is counter productive in our case as we wanted to

reduce the number of values needed to represent the polygons.

7.6 Conclusion

In this work, we presented CenterPoly, a novel real-time instance segmentation method that segments objects using bounding polygons. We designed a policy to select ground-truth vertices which helps the training process. We show the importance of depth map to improve the accuracy of the polygons because of the overlap problem. CenterPoly ranks first for real-time instance segmentation methods on the popular cityscapes, KITTI and IDD datasets.

CHAPITRE 8 DISCUSSION GÉNÉRALE

Dans ce chapitre, nous faisons un retour global sur le travail accompli. Nous discuterons des avantages et des inconvénients des méthodes par apprentissage en comparaison des méthodes de vision par ordinateur classiques. Nous abordons les limitations et les améliorations possibles de chaque méthode du corps de la thèse. Nous discutons aussi des adaptations à faire pour utiliser ces méthodes dans des situations réalistes non contrôlées.

8.1 Méthodes par apprentissage versus méthodes classiques

Les méthodes de détection et segmentation d'objets par apprentissage ont dominées les compétitions sur des données depuis plusieurs années. Toutefois, elles comportent quelques désavantages par rapport aux méthodes dites classiques, par exemple la soustraction d'arrière-plan. Les méthodes par apprentissage profond doivent être entraînées, ce qui requiert des données et des annotations qui sont la plupart du temps assez coûteuses à obtenir. De plus, l'entraînement prend du temps et des ressources matérielles, l'entraînement d'un modèle moderne est plutôt impossible sans une carte graphique d'un certain niveau de performance. Un autre désavantage des méthodes par apprentissage profond est une moins bonne réaction à l'inconnu. Dans une méthode de soustraction d'arrière-plan, tout ce qui n'est pas de l'arrière-plan sera identifié comme étant d'intérêt. Dans une méthode par apprentissage profond, un objet d'un type jamais vu pendant l'entraînement et assez différent des autres catégories ne sera pas détecté, et cela pourrait causer des problèmes. On pense par exemple à un être humain portant un costume de mascotte qui ne serait pas identifié comme un piéton par une voiture autonome. Le dernier désavantage identifié sera la possibilité d'attaques antagonistes dans les méthodes par apprentissage profond, qui consiste à exploiter certaines faiblesses spécifiques d'un réseau de neurones pour le tromper et obtenir une autre sortie que celle attendue.

Cependant, les méthodes par apprentissage jouissent de bien plus grands précision et rappel que leurs contreparties classiques. Cet argument à lui seul peut suffire à justifier leur utilisation. Ils fonctionnent généralement de façon plus universelle, par exemple la soustraction d'arrière-plan requiert quelques exemples d'images d'arrière-plan sans objet d'intérêts afin de bien le modéliser. Le flux optique quant à lui requiert que les objets d'intérêts soient en mouvement. Un détecteur d'objets par apprentissage quant à lui a seulement besoin d'une image. De plus, certaines branches de l'apprentissage machine tentent de palier aux autres inconvénients, par exemple l'entraînement non supervisé ne requiert pas d'annotations [88] et

certaines recherches tentent de rendre les CNNs plus résistants aux attaques antagonistes [89].

8.2 Détection d'objets dans les images

La méthode de détection proposée, SpotNet, a été conçue pour maximiser le mAP sur deux jeux de données. Par ses résultats, elle a en effet surpassée l'état-de-l'art sur UA-DETRAC et UAVDT, et de ce côté est un succès. Cette méthode a toutefois quelques limitations. Premièrement, pour entraîner le module d'attention à produire la carte d'attention, nous nous appuyons sur des annotations partielles produites par soustraction d'arrière-plan et flux optique. Ces deux méthodes requièrent des séquences vidéo, ou du moins des paires d'images pour le flux optique. Ceci fait en sorte qu'il peut être problématique d'entraîner SpotNet sur n'importe quel jeu de données, bien que ce ne soit pas un problème si on veut utiliser le réseau en mode inférence seulement. Des stratégies d'entraînement par transfert pourraient aussi être utilisées si nous devions absolument entraîner ce réseau sur un jeu de données ne contenant que des images qui ne forment pas de séquences.

Ensuite, au niveau des segmentations produites par le réseau, nous pouvons ajouter quelques détails. La binarisation de la carte d'attention produit une segmentation avant-plan et arrière-plan de la scène. Si nous effectuons une intersection entre cette segmentation et les rectangles englobants dans l'image, nous obtenons une segmentation qui est presque une segmentation d'instances. En effet, pour chaque rectangle englobant, nous pouvons obtenir un masque à l'intérieur de ce rectangle. Cependant, quelques problèmes peuvent survenir. Premièrement, nous n'avons aucune garantie d'obtenir quoi que ce soit d'autre que des pixels noirs à l'intérieur d'un rectangle, ce qui voudrait dire que la carte d'attention n'a rien détectée alors que le reste du réseau a détecté un objet à cet endroit. C'est tout à fait possible. Ensuite, il est aussi très courant d'avoir des rectangles englobants qui se chevauchent avec les types d'images avec lesquelles on travaille. Cela fera en sorte que nous pourrions avoir des parties de plusieurs objets différents dans le même rectangle sur la carte d'attention. Il sera alors impossible de différencier quels pixels du masque font parties de l'objet dont nous utilisons le rectangle et lesquels non. Pour cette raison, nous appelons ce masque une segmentation d'avant-plan et d'arrière-plan, et non une segmentation d'instances ou sémantique. Notre méthode CenterPoly quant à elle garantie un masque de segmentation pour chaque détection.

Ensuite, nous pouvons noter que cette méthode ayant été conçue pour maximiser le mAP n'est pas particulièrement rapide. En effet, SpotNet serait impossible à utiliser en temps réel, ce qui pourrait être un problème pour certaines applications. Pour l'adapter, le meilleur moyen serait de changer le CNN extracteur d'attributs. Au lieu d'utiliser le réseau Hourglass, nous pourrions utiliser un réseau ResNet plus rapide, par exemple ResNet-50 ou ResNet-34. Ceci

affaiblirait certainement les performances, mais la méthode resterait sans doute compétitive avec les autres méthodes fonctionnant à cette vitesse.

Les défis propres au domaine du transport à laquelle cette méthode s'attaque sont la présence d'éléments distracteurs et les conditions lumineuses et météorologiques difficiles. En effet, une tête du réseau entraînée de façon différente permet au réseau de mieux apprendre ce qui n'est pas un usager de la route, par exemple des panneaux publicitaires et diverses structures routières. Inversement, la carte d'attention permet au réseau de plus se concentrer sur les zones contenant des usagers de la route, aidant dans des situations où la visibilité pourrait être amoindrie.

8.3 Détection d'objets dans les vidéos

Les chapitres 5 et 6 présentent une méthode de détection d'objets dans les vidéos. Nous avons démontré que cette méthode peut être adaptée sans problème à plusieurs détecteurs de base et est assez universelle à ce niveau. Une des plus grandes limitations de cette méthode est la petite marge possible d'amélioration. En effet, cette méthode permet d'améliorer la détection seulement dans les cas où le détecteur de base ne pourrait pas détecter parfaitement par lui-même. Cette méthode fonctionne donc particulièrement bien sur des séquences difficiles et est moins utile sur des séquences faciles, car dans le cas des séquences faciles, le détecteur de base peut déjà faire le travail par lui-même. Et même sur des séquences difficiles, beaucoup d'objets seront bien détectés par le détecteur de base et ne pourront donc pas être améliorés. En d'autres mots, notre méthode peut seulement améliorer ce qui est raté par le détecteur de base, ce qui crée une limite quant à l'amélioration que l'on peut apporter avec cette méthode.

Une autre limitation de la méthode est la fenêtre temporelle à laquelle elle a accès, qui est sommes toute relativement petite. Cette limitation est due au compromis entre avoir un décalage utile entre les trames et avoir un décalage trop grand, ce qui fait en sorte que les cartes d'attributs deviennent trop différentes. Pour s'attaquer à ce problème, nous aurions pu utiliser une technique de déformation des cartes d'attributs par flux optique par exemple. Cela aurait permis d'allonger la fenêtre temporelle, et d'utiliser un plus grand nombre de trames potentiellement utiles. Toutefois, cela aurait ajouté de la complexité et des éléments externes au réseau, et nous avons opté pour la simplicité. D'autant plus que la conception de notre méthode est basée sur la possibilité de réutiliser les cartes d'attributs trame après trame et de rester rapide.

Comme mentionné plus haut, cette méthode est particulièrement utile pour détecter les exemples difficiles. Comme le réseau a accès une fenêtre temporelle de quelques trames, il

peut sélectionner les attributs sur certaines trames s'ils sont occlus dans une autre. Pour cette raison, cette méthode peut aider pour les cas d'occlusions partiels qui sont très présents dans le domaine du transport. De façon générale, cette méthode sert à enrichir les cartes d'attributs, elle peut donc aider aussi à éliminer la présence de distracteurs et dans les cas de visibilité réduite.

8.4 Segmentation d'instances

Comme mentionné ci-haut, bien que SpotNet apporte une certaine segmentation de l'avant-plan, ce n'est pas une vraie segmentation d'instances. CenterPoly quant à elle, est une méthode qui apporte une vraie segmentation d'instances via des polygones englobants. On pourrait s'interroger sur l'utilité d'utiliser des polygones englobants plutôt que d'utiliser des masques. Utiliser des polygones nous permet d'atteindre des vitesses en temps réel et donc ouvre une porte à un grand nombre d'applications. Toutefois, nous perdons un peu en précision. En effet, si une application requiert une segmentation fine pixel par pixel, alors une méthode par polygone ne serait pas appropriée. Si cependant une application avait plutôt besoin d'une plus grande précision par rapport aux rectangles englobants tout en restant rapide (par exemple le suivi multiobjets ou la réidentification), alors une méthode de segmentation par polygones serait tout à fait appropriée. Notons qu'une méthode par polygones englobants est assez précise pour segmenter les membres et la tête des piétons, les rétroviseurs des voitures, etc.

La carte de profondeur relative générée par ce réseau est particulièrement utile afin de mieux segmenter une grande densité de petits objets, par exemple une foule de piétons ou un groupe de voitures. En effet, dans ces cas, nous aurons un grand chevauchement des usagers de la route les uns avec les autres, et nous devrons décider quel polygone placer devant les autres. Ce genre de situations peut survenir assez fréquemment dans des images du domaine routier, pensons notamment à une ligne de voitures stationnées, un boulevard achalandé, une foule de piétons, etc. On peut aussi penser que la profondeur relative des objets serait utile comme attribut supplémentaire pour du suivi multiobjets, notamment pour éviter les transferts d'identités lors d'occlusions totales.

Une des plus grandes limitations de cette méthode est la nécessité d'annotations coûteuses à produire. En effet, cela limite beaucoup les jeux de données sur lesquels on peut s'entraîner et s'évaluer. Toutefois, cela n'empêche en rien d'utiliser un réseau pré-entraîné si on veut l'utiliser en inférence sur une application en particulier. Pour pallier cette difficulté, il serait très intéressant de développer une méthode d'entraînement semi-supervisé. Semi-supervisé dans le sens où nous aurions toujours besoin des annotations de rectangles englobants, mais

nous pourrions entraîner les polygones sans annotations de segmentation. Une façon de faire serait d'utiliser un bon réseau de segmentation d'instances pré-entraîné pour produire des annotations estimées, c'est-à-dire des pseudo-étiquettes. Avec cette méthode, nous pourrions probablement obtenir des résultats similaires au réseau pré-entraîné, mais nous pourrions difficilement le surpasser.

En ce qui a trait aux difficultés propres au domaine routier, cette méthode traite particulièrement la grande densité des usagers de la route grâce à la carte de profondeur relative. Les occlusions partielles sont aussi mieux gérées entre autres grâce à la profondeur. Aussi, le fait d'avoir une segmentation plus fine permet de mieux distinguer les objets proches. Par exemple, si deux rectangles englobants de deux objets différents sont trop près, un des deux pourraient être supprimé par la suppression non maximale. Dans ce cas, dépendant de l'angle des objets, les rectangles englobants pourraient être en fort recouvrement alors que la segmentation l'est beaucoup moins.

8.5 Utilité des différents éléments

Nous pouvons évaluer et recommander l'utilité des éléments des différentes méthodes proposées pour certains types d'applications.

L'utilisation de plusieurs trames avec le module de fusion peut améliorer les résultats dans la plupart des situations, lorsque ces trames sont disponibles. Cette méthode fonctionnant avec plusieurs détecteurs, elle s'adapte assez bien à un grand nombre d'applications. La vitesse d'inférence n'est pas impactée grandement par le module de fusion, cependant, la mémoire utilisée oui, car les cartes d'attributs de la fenêtre autour de la trame cible doivent être gardées en mémoire. Donc pour des applications embarquées avec une mémoire réduite, par exemple roulant sur une caméra ou un téléphone, cette méthode ne serait pas recommandée.

La module d'attention de SpotNet améliore considérablement la précision. Son utilisation est recommandable dans la plupart des applications, excepté pour des applications temps réels. En effet, l'utilisation du module d'attention ralentit quand même considérablement le réseau, et ne serait donc pas une priorité dans le design d'un détecteur rapide. Le choix du bon CNN de base d'un détecteur serait à prioriser avant le module d'attention, pour les mêmes vitesses atteintes. Notons que ce processus d'attention est adaptable dans plusieurs cas d'entraînements multitâches, par exemple détection et réidentification, détection et segmentation, etc. Pour clarifier, afin de maximiser le temps de calcul, l'application devrait pouvoir se servir du résultat du module d'attention, tel que la segmentation, sinon un autre détecteur serait à prioriser.

La segmentation des instances peut s'avérer très utile dans un grand nombre d'applications. Par exemple, pour effectuer le suivi multiobjets, la segmentation peut servir à mieux construire le modèle d'apparence d'un objet et soustraire l'arrière-plan. Une application de réidentification peut en bénéficier pour les mêmes raisons. En effet, avec moins d'arrière-plan dans le modèle d'apparence, la qualité des attributs s'en verra améliorée. De plus, l'utilisation des polygones n'a pas un impact énorme sur la vitesse et la mémoire, elle est donc appropriée pour des applications rapides et embarquées.

CHAPITRE 9 CONCLUSION

9.1 Synthèse des travaux

Dans le cadre de cette thèse, nous avons présenté des méthodes originales pour classifier, détecter et segmenter les usagers de la route dans des images et des vidéos. Nous avons travaillé avec des techniques d'apprentissage supervisé et semi-supervisé. Pour toutes nos contributions, nous avons publié le code en ligne ainsi que des modèles pré-entraînés pour une utilisation immédiate.

La première méthode proposée est une méthode de détection d'objets. Elle utilise des techniques de vision par ordinateur classique pour produire des annotations partielles de saillance sur des jeux de données de surveillance routière. Ces annotations sont utilisées pour entraîner un module d'auto-attention qui sert à produire des cartes de saillances, qui serviront à la fois comme processus d'attention sur les cartes d'attributs à l'intérieur du réseau ainsi que pour produire une segmentation d'avant-plan en plus des rectangles englobants.

La deuxième méthode proposée est une méthode de détection d'objets sur vidéo. Elle introduit un module de fusion qui apprend à combiner des cartes d'attributs de trames temporellement proches afin d'en produire des attributs enrichis. Ce module de fusion fonctionne grâce à une concaténation des canaux, de convolutions 1×1 suivies d'un réarrangement des canaux. Grâce à une stratégie de fusion tard dans le réseau ainsi que la réutilisation des cartes d'attributs lorsqu'on fait la détection trame par trame dans la séquence, le coût de calcul est assez bas.

La troisième méthode proposée est une combinaison des deux méthodes précédentes. Premièrement, nous généralisons l'architecture de la première méthode et nous l'introduisons dans deux nouveaux détecteurs modernes. Ensuite, nous proposons une amélioration au module d'attention de la deuxième méthode en utilisant un réseau de type U-Net. Une étude d'ablation de plus grande qualité est aussi réalisée afin de comparer le bénéfice du module de fusion à des opérations de fusion classiques telles que le maximum, la moyenne, etc.

Finalement, la quatrième méthode proposée est une méthode de segmentation d'instances rapide par polygones englobants. Elle fonctionne en introduisant une façon de normaliser les sommets des polygones afin d'en faciliter l'apprentissage. Les objets sont détectés par leur point central et le réseau est entraîné afin de produire un polygone pour chaque objet détecté. Parallèlement, nous entraînons une tête du réseau à produire une profondeur relative qui sert à placer les objets plus ou moins profondément les uns par rapport aux autres. Cette méthode est particulièrement rapide et fonctionne en temps réel.

La contribution au domaine scientifique qu'a amenée la réalisation de ces différentes méthodes s'exprime par de nombreuses publications dans des actes de congrès ainsi qu'un article de journal.

9.2 Recommandations pour travaux futurs

La détection d'objets par rectangles englobants pleinement supervisée a atteint une certaine maturité. Bien qu'il reste toujours une marge d'amélioration, dans le futur, il sera de plus en plus difficile d'y apporter des améliorations. Cela est un peu moins vrai pour la segmentation d'instances, comme c'est un sujet plus jeune et plus complexe. Notamment, il existe très peu de méthodes de segmentation d'instances en temps réel, c'est une avenue très prometteuse. Dans la perspective où nous utiliserons de plus en plus de vision par ordinateur dans des applications de la vie courante, il sera aussi très intéressant de rendre les méthodes de détection et segmentation très efficaces en termes de coût de calcul ainsi que de l'utilisation de la mémoire.

La grande disponibilité des données et le coût des annotations favorisent de plus en plus une approche d'apprentissage auto-supervisé ou semi-supervisé. Produire des annotations de segmentation est particulièrement coûteux, et il sera très intéressant de trouver des moyens d'utiliser les données sans annotations. Cela sera particulièrement vrai pour une application avec un domaine spécifique, par exemple une application de vision embarquée sur une caméra fixe avec un point de vue sur une route. On peut imaginer dans ce contexte que des dizaines de caméras exécuteraient la même application, mais avec un domaine de données différent. Il serait très fastidieux de produire des annotations pour chaque point de vue afin de raffiner chaque modèle : automatiser ce processus de raffinement à partir d'un modèle pré-entraîné serait très bénéfique.

Nous pouvons observer un certain rapprochement entre certaines tâches autrefois distinctes. L'apparition de la segmentation panoptique en est un bon exemple, combinant la segmentation sémantique et segmentation d'instances. Parallèlement à ce phénomène, nous voyons l'apparition de méthodes multitâches étant capables de produire des résultats pour plusieurs tâches en même temps. Ces deux phénomènes s'encouragent mutuellement. Nous pourrions facilement imaginer que cette combinaison de tâches va continuer et nous verrons l'apparition de plus en plus de "super" tâches. Ultimement, ces tâches pourraient être par exemple le suivi et la segmentation panoptique sur vidéo, multicaméras.

RÉFÉRENCES

[1] X. Zhou, D. Wang et P. Krähenbühl, "Objects as points," *arXiv preprint arXiv :1904.07850*, 2019.

[2] H. Perreault, G.-A. Bilodeau, N. Saunier et M. Héritier, "Spotnet : Self-attention multi-task network for object detection," dans *2020 17th Conference on Computer and Robot Vision (CRV)*. IEEE, 2020, p. 230–237.

[3] T.-Y. Lin, P. Goyal, R. Girshick, K. He et P. Dollár, "Focal loss for dense object detection," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[4] H. Perreault, M. Heritier, P. Gravel, G.-A. Bilodeau et N. Saunier, "Rn-vid : A feature fusion architecture for video object detection," dans *International Conference on Image Analysis and Recognition*. Springer, 2020, p. 125–138.

[5] L. Wen, D. Du, Z. Cai, Z. Lei, M.-C. Chang, H. Qi, J. Lim, M.-H. Yang et S. Lyu, " UA-DETRAC : A New Benchmark and Protocol for Multi-Object Detection and Tracking," *arXiv CoRR*, vol. abs/1511.04136, 2015.

[6] S. Li et F. Chen, "3d-detnet : a single stage video-based vehicle detector," dans *Third International Workshop on Pattern Recognition*, vol. 10828. International Society for Optics and Photonics, 2018, p. 108280A.

[7] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang et Q. Tian, "The unmanned aerial vehicle benchmark : Object detection and tracking," dans *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, p. 370–386.

[8] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad et P. Ishwar, "Changedetection. net : A new change detection benchmark dataset," dans *2012 IEEE computer society conference on computer vision and pattern recognition workshops*. IEEE, 2012, p. 1–8.

[9] G. Varma, A. Subramanian, A. Namboodiri, M. Chandraker et C. Jawahar, "Idd : A dataset for exploring problems of autonomous navigation in unconstrained environments," dans *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, p. 1743–1751.

[10] MTheiler. (2019) Detected-with-YOLO–Schreibtisch-mit-Objekten. File : Detected-with-YOLO–Schreibtisch-mit-Objekten.jpg. [En ligne]. Disponible : https://commons.wikimedia.org/wiki/File:Detected-with-YOLO--Schreibtisch-mit-Objekten.jpg

[11] Adrian Rosebrock. (2016) Intersection_over_Union_-_poor,_good_and_excellent_score. File : Intersection_over_Union_-_poor,_good_and_excellent_score.png. [En ligne]. Disponible : https:

//commons.wikimedia.org/wiki/File:Intersection_over_Union_-_poor,_good_and_excellent_score.png

[12] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth et B. Schiele, "The cityscapes dataset for semantic urban scene understanding," dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, p. 3213–3223.

[13] K. He, X. Zhang, S. Ren et J. Sun, "Deep residual learning for image recognition," dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, p. 770–778.

[14] A. Newell, K. Yang et J. Deng, "Stacked hourglass networks for human pose estimation," dans *European conference on computer vision*. Springer, 2016, p. 483–499.

[15] S. Ren, K. He, R. Girshick et J. Sun, "Faster r-cnn : Towards real-time object detection with region proposal networks," dans *Advances in neural information processing systems*, 2015, p. 91–99.

[16] J. Redmon, S. Divvala, R. Girshick et A. Farhadi, "You only look once : Unified, real-time object detection," dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, p. 779–788.

[17] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang et Q. Tian, "Centernet : Keypoint triplets for object detection," dans *Proceedings of the IEEE International Conference on Computer Vision*, 2019, p. 6569–6578.

[18] X. Zhu, Y. Wang, J. Dai, L. Yuan et Y. Wei, "Flow-guided feature aggregation for video object detection," dans *Proceedings of the IEEE International Conference on Computer Vision*, 2017, p. 408–417.

[19] M. Liu et M. Zhu, "Mobile video object detection with temporally-aware feature maps," dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, p. 5686–5695.

[20] K. He, G. Gkioxari, P. Dollár et R. Girshick, "Mask r-cnn," dans *Proceedings of the IEEE international conference on computer vision*, 2017, p. 2961–2969.

[21] E. Xie, P. Sun, X. Song, W. Wang, X. Liu, D. Liang, C. Shen et P. Luo, "Polarmask : Single shot instance segmentation with polar representation," dans *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, p. 12 193–12 202.

[22] H. Perreault, G.-A. Bilodeau, N. Saunier et P. Gravel, "Road user detection in videos," *arXiv preprint arXiv :1903.12049*, 2019.

[23] P.-L. St-Charles, G.-A. Bilodeau et R. Bergevin, "A self-adjusting approach to change detection based on background word consensus," dans *2015 IEEE winter conference on applications of computer vision*. IEEE, 2015, p. 990–997.

[24] K. Simonyan et A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv :1409.1556*, 2014.

[25] A. Krizhevsky, I. Sutskever et G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, p. 1097–1105, 2012.

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li et L. Fei-Fei, "ImageNet : A Large-Scale Hierarchical Image Database," dans *CVPR09*, 2009.

[27] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár et C. L. Zitnick, "Microsoft coco : Common objects in context," dans *European conference on computer vision*. Springer, 2014, p. 740–755.

[28] R. Girshick, J. Donahue, T. Darrell et J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, p. 580–587.

[29] R. Girshick, "Fast r-cnn," dans *Proceedings of the IEEE international conference on computer vision*, 2015, p. 1440–1448.

[30] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu et A. C. Berg, "Ssd : Single shot multibox detector," dans *European conference on computer vision*. Springer, 2016, p. 21–37.

[31] J. R. Uijlings, K. E. Van De Sande, T. Gevers et A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, n°. 2, p. 154–171, 2013.

[32] J. Dai, Y. Li, K. He et J. Sun, "R-fcn : object detection via region-based fully convolutional networks," dans *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, p. 379–387.

[33] J. Redmon et A. Farhadi, "Yolo9000 : better, faster, stronger," dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, p. 7263–7271.

[34] ——, "Yolov3 : An incremental improvement," *arXiv preprint arXiv :1804.02767*, 2018.

[35] A. Bochkovskiy, C.-Y. Wang et H.-Y. M. Liao, "Yolov4 : Optimal speed and accuracy of object detection," *arXiv preprint arXiv :2004.10934*, 2020.

[36] H. Law et J. Deng, "Cornernet : Detecting objects as paired keypoints," dans *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, p. 734–750.

[37] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov et S. Zagoruyko, “End-to-end object detection with transformers,” dans *European Conference on Computer Vision*. Springer, 2020, p. 213–229.

[38] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers et T. Brox, “Flownet : Learning optical flow with convolutional networks,” dans *Proceedings of the IEEE international conference on computer vision*, 2015, p. 2758–2766.

[39] S. Wang, Y. Zhou, J. Yan et Z. Deng, “Fully motion-aware network for video object detection,” dans *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, p. 542–557.

[40] F. Xiao et Y. Jae Lee, “Video object detection with an aligned spatial-temporal memory,” dans *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, p. 485–501.

[41] A. Broad, M. Jones et T.-Y. Lee, “Recurrent multi-frame single shot detector for video object detection.” dans *BMVC*, 2018, p. 94.

[42] G. Bertasius, L. Torresani et J. Shi, “Object detection in video with spatiotemporal sampling networks,” dans *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, p. 331–346.

[43] C. Li, G. Dobler, X. Feng et Y. Wang, “Tracknet : Simultaneous object detection and tracking and its application in traffic video analysis,” *arXiv preprint arXiv :1902.01466*, 2019.

[44] B. Munjal, A. R. Aftab, S. Amin, M. D. Brandlmaier, F. Tombari et F. Galasso, “Joint detection and tracking in videos with identification features,” *Image and Vision Computing*, vol. 100, p. 103932, 2020.

[45] X. Lin, Y.-a. Guo et J. Wang, “Global correlation network : End-to-end joint multi-object detection and tracking,” *arXiv preprint arXiv :2103.12511*, 2021.

[46] X. Wang, X. Hu, C. Chen, Z. Fan et S. Peng, “Illuminating vehicles with motion priors for surveillance vehicle detection,” dans *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, p. 2021–2025.

[47] H. Hu, W. Wang, A. Zheng et B. Luo, “Mma : motion memory attention network for video object detection,” dans *International conference on image and graphics*. Springer, 2019, p. 167–178.

[48] W. Liu, S. Liao et W. Hu, “Perceiving motion from dynamic memory for vehicle detection in surveillance videos,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, n°. 12, p. 3558–3567, 2019.

[49] C.-Y. Fu, M. Shvets et A. C. Berg, “Retinamask : Learning to predict masks improves state-of-the-art single-shot detection for free,” *arXiv preprint arXiv :1901.03353*, 2019.

[50] S. Liu, L. Qi, H. Qin, J. Shi et J. Jia, “Path aggregation network for instance segmentation,” dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, p. 8759–8768.

[51] P. Hurtik, V. Molek, J. Hula, M. Vajgl, P. Vlasanek et T. Nejezchleba, “Poly-yolo : higher speed, more precise detection and instance segmentation for yolov3,” *arXiv preprint arXiv :2005.13243*, 2020.

[52] L. Castrejon, K. Kundu, R. Urtasun et S. Fidler, “Annotating object instances with a polygon-rnn,” dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, p. 5230–5238.

[53] D. Acuna, H. Ling, A. Kar et S. Fidler, “Efficient interactive annotation of segmentation datasets with polygon-rnn++,” dans *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, p. 859–868.

[54] J. Uhrig, E. Rehder, B. Fröhlich, U. Franke et T. Brox, “Box2pix : Single-shot instance segmentation by assigning pixels to object boxes,” dans *IEEE Intelligent Vehicles Symposium (IV)*, 2018. [En ligne]. Disponible : http://lmb.informatik.uni-freiburg.de/Publications/2018/UB18

[55] D. Mazzini et R. Schettini, “Spatial sampling network for fast scene understanding,” dans *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[56] D. Bolya, C. Zhou, F. Xiao et Y. J. Lee, “Yolact : Real-time instance segmentation,” dans *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, p. 9157–9166.

[57] H. Alhaija, S. Mustikovela, L. Mescheder, A. Geiger et C. Rother, “Augmented reality meets computer vision : Efficient data generation for urban driving scenes,” *International Journal of Computer Vision (IJCV)*, 2018.

[58] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel et Y. Bengio, “Show, attend and tell : Neural image caption generation with visual attention,” dans *International conference on machine learning*, 2015, p. 2048–2057.

[59] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser et I. Polosukhin, “Attention is all you need,” dans *Advances in neural information processing systems*, 2017, p. 5998–6008.

[60] Z. Cai et N. Vasconcelos, “Cascade r-cnn : Delving into high quality object detection,” dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, p. 6154–6162.

[61] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan et S. Belongie, “Feature pyramid networks for object detection,” dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, p. 2117–2125.

[62] S. Amin et F. Galasso, “Geometric proposals for faster r-cnn,” dans *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2017, p. 1–6.

[63] S. Wu, M. Kan, S. Shan et X. Chen, “Hierarchical attention for part-aware face detection,” *International Journal of Computer Vision*, vol. 127, nº. 6-7, p. 560–578, 2019.

[64] Z. Fu, Y. Chen, H. Yong, R. Jiang, L. Zhang et X.-S. Hua, “Foreground gating and background refining network for surveillance object detection,” *IEEE Transactions on Image Processing*, vol. 28, nº. 12, p. 6077–6090, 2019.

[65] G. Farnebäck, “Two-frame motion estimation based on polynomial expansion,” dans *Scandinavian conference on Image analysis*. Springer, 2003, p. 363–370.

[66] S. Lyu, M.-C. Chang, D. Du, W. Li, Y. Wei, M. Del Coco, P. Carcagnì, A. Schumann, B. Munjal, D.-H. Choi *et al.*, “Ua-detrac 2018 : Report of avss2018 & iwt4s challenge on advanced traffic monitoring,” dans *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE, 2018, p. 1–6.

[67] L. Wang, Y. Lu, H. Wang, Y. Zheng, H. Ye et X. Xue, “Evolving boxes for fast vehicle detection,” dans *2017 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2017, p. 1135–1140.

[68] T. Wang, R. M. Anwer, H. Cholakkal, F. S. Khan, Y. Pang et L. Shao, “Learning rich features at high-speed for single-shot object detection,” dans *Proceedings of the IEEE International Conference on Computer Vision*, 2019, p. 1971–1980.

[69] T. Kong, F. Sun, A. Yao, H. Liu, M. Lu et Y. Chen, “Ron : Reverse connection with objectness prior networks for object detection,” dans *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2017, p. 2.

[70] P.-L. St-Charles, G.-A. Bilodeau et R. Bergevin, “Subsense : A universal change detection method with local adaptive sensitivity,” *IEEE Transactions on Image Processing*, vol. 24, nº. 1, p. 359–373, 2014.

[71] R. H. Evangelio, M. Pätzold et T. Sikora, “Splitting gaussians in mixture models,” dans *2012 IEEE Ninth international conference on advanced video and signal-based surveillance*. IEEE, 2012, p. 300–305.

[72] Z. Zivkovic et F. Van Der Heijden, “Efficient adaptive density estimation per image pixel for the task of background subtraction,” *Pattern recognition letters*, vol. 27, n°. 7, p. 773–780, 2006.

[73] C. Stauffer et W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” dans *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 2. IEEE, 1999, p. 246–252.

[74] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers et T. Brox, “Flownet : Learning optical flow with convolutional networks,” dans *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[75] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy et T. Brox, “Flownet 2.0 : Evolution of optical flow estimation with deep networks,” dans *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[76] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke et A. Rabinovich, “Going deeper with convolutions,” dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, p. 1–9.

[77] F. Chollet *et al.*, “Keras,” https://keras.io, 2015.

[78] M. Abadi *et al.*, “TensorFlow : Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [En ligne]. Disponible : http://tensorflow.org/

[79] D. P. Kingma et J. Ba, “Adam : A method for stochastic optimization,” *arXiv preprint arXiv :1412.6980*, 2014.

[80] Z. Zhang, X. Zhang, C. Peng, X. Xue et J. Sun, “Exfuse : Enhancing feature fusion for semantic segmentation,” dans *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, p. 269–284.

[81] H. Qiu, Y. Ma, Z. Li, S. Liu et J. Sun, “Borderdet : Border feature for dense object detection,” *arXiv preprint arXiv :2007.11056*, 2020.

[82] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai et S. Chintala, “Pytorch : An imperative style, high-performance deep learning library,” dans *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox et R. Garnett, édit. Curran Associates, Inc., 2019, p. 8024–8035. [En ligne]. Disponible : http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[83] F. Yu, D. Wang, E. Shelhamer et T. Darrell, “Deep layer aggregation,” dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, p. 2403–2412.

[84] Y.-C. Hsu, Z. Xu, Z. Kira et J. Huang, “Learning to cluster for proposal-free instance segmentation,” dans *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, p. 1–8.

[85] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy et C. Rother, “Instancecut : from edges to instances with multicut,” dans *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, p. 5008–5017.

[86] M. Ren et R. S. Zemel, “End-to-end instance segmentation with recurrent attention,” dans *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, p. 6656–6664.

[87] X. Zhou, V. Koltun et P. Krähenbühl, “Simple multi-dataset detection,” *arXiv preprint arXiv :2102.13086*, 2021.

[88] L. Schmarje, M. Santarossa, S.-M. Schröder et R. Koch, “A survey on semi-, self-and unsupervised learning for image classification,” *IEEE Access*, 2021.

[89] F. O. Olowononi, D. B. Rawat et C. Liu, “Resilient machine learning for networked cyber physical systems : A survey for machine learning security to securing machine learning for cps,” *IEEE Communications Surveys & Tutorials*, vol. 23, n°. 1, p. 524–552, 2020.