



الجمهورية الجزائرية الديمقراطية الشعبية  
وزارة التعليم العالي والبحث العلمي  
جامعة وهران للعلوم والتكنولوجيا محمد بوضياف  
كلية الرياضيات و الاعلام الالي

République Algérienne Démocratique et Populaire  
Ministère de l'Enseignement Supérieur Et de la Recherche Scientifique  
Université des Sciences et de la Technologie d'Oran Mohamed BOUDIAF  
Faculté des Mathématiques et Informatique

*Département : Informatique*

Mémoire de fin d'études

## Détection des véhicules en temps réel en utilisant les réseaux de neurones convolutifs

Pour l'obtention du diplôme  
De **Master**

Domaine : **Mathématiques – Informatique**

Filière : **Informatique**

Spécialité : **Intelligence artificielle et ses applications (IAA)**

Présenté le :  
Par :

-Akram DOUDOU  
-Amine BENABDALLAH

Jury	Nom et Prénom	Grade	Université
Présidente :	Hadria FIZAZI	Professeur	USTO-MB
Encadrante :	Hadjer MOUSSA	Maître de Conférence B	USTO-MB
Co-encadrant :	M.Anis BENALLAL	Maître de Conférence B	USTO-MB
Examineur :	Karima KIES	Maître de Conférence B	USTO-MB
Examineur :	Ibtisem BEKKOUCHE	Maître de Conférence B	USTO-MB

2018/2019

# *Remerciement*

En préambule à ce mémoire on tient à remercier notre dieu ALLAH le tout puissant, de nous avoir donné la force et le courage d'achever ce projet.

Ce travail n'aurait pu aboutir sans l'aide de plusieurs personnes, que nous ne remercierons jamais assez.

Nos remerciements vont en tout premier lieu à notre encadreur et notre co-encadreur :

Mme Hadjer MOUSSA et Mr Anis BENALLAL, pour nous avoir aidé et dirigé durant notre modeste travail, pour le temps que vous nous avez consacré et votre patience, et plus particulièrement pour avoir partagé avec nous votre savoir. Nous vous présentons, notre plus grand respect pour ce que vous avez fait pour nous.

A Mme Hadria Fizazi :

Nous tenons à vous témoigner nos plus sincères remerciements pour avoir accepté de présider notre soutenance, nous vous sommes très reconnaissants pour la qualité d'enseignement que vous nous avez prodigué durant ces cinq dernières années toujours sous la joie et la bonne humeur.

A Mme Karima KIES :

Nous sommes comblés de l'immense honneur que vous nous faites en acceptant de faire partie des membres du Jury de notre soutenance de Master, malgré vos nombreuses occupations. Vous resterez, pour nous, l'un de nos enseignants les plus marquants durant tout notre cursus universitaire.

A Mme Ibtisem BEKKOUCHE :

Nous vous adressons nos sincères remerciements pour votre présence en tant qu'examinatrice. Nous Avons eu le privilège de bénéficier de vos enseignements précis et clairs. Vos rigueurs scientifiques, et votre amour du travail bien fait.

# Dédicace

Je dédie ce travail :

A ma très chère mère

Quoi que je fasse ou que je dise, je ne saurai point te remercier comme il se doit, ton affection me couvre, ta bienveillance me guide et ta présence à mes côtés a toujours été ma source de force pour affronter les différents obstacles.

A mon très cher père

Qui n'a jamais cessé de me soutenir et de m'épauler pour que je puisse atteindre mes objectifs, et qui a toujours été un mentor pour moi.

A mes frères et sœurs

Pour leurs présences et leurs aides dans les moments difficiles, j'espère vous avoir près de moi pour le reste de ma vie.

A ma très chère tante Fadela :

Que je considère comme ma deuxième maman.

A mes amis

Je ne saurai comment remercier mes chers amis en particulier notre défunt Bilel qu'il repose en paix Madjid, El Safi, Houari, Moukhtar, El Hadj, Djilali, Znati, Laid, Slimane, Bouazza, abdelatif et Souhil qui ont été là pour moi et qui m'ont toujours soutenue et tendue la main quand j'en avais besoin , vous êtes comme des frères pour moi.

A mon cher binôme

Pour m'avoir supporté tout au long de ce travail, son enthousiasme et sa patience nous a permis d'arriver jusque là

A mes chers Collègues

Et amis Ali, Nadir, Farid et Khalil pour avoir passé avec moi deux années inoubliables nous avons vu le pire et le meilleur ensemble.

**Amine**

**BENABDALLAH**

# Dédicace

Je dédie ce projet :

À mes chers parents, qui n'ont jamais cessé de formuler des prières à mon égard, de me soutenir et de m'épauler pour que je puisse atteindre mes objectifs, je leurs remerciais jamais assez pour tous leurs efforts fournis afin que j'arrive au niveau dont lequel je suis maintenant.

À ma chère sœur, pour le soutien moral tout au long de mes études.

À ma grand-mère, à qui je souhaite une très bonne santé et longue vie.

À mon cher binôme, pour son enthousiasme et sa sympathie et qui m'a énormément aidé durant tout le long de notre projet.

À mon oncle Bourayou Fethi, qui m'a soutenu moralement dans les situations difficiles et qui m'a guidé vers le droit chemin.

À mon oncle Bourayou AbdelKarim, qui m'a énormément soutenu, et qui m'a beaucoup conseillé dans mon projet et qui m'a suggéré plusieurs idées à effectuer à l'avenir.

À mon ami Billel Allouche, qui m'a beaucoup soutenu moralement lorsque je me sentais dépressif et démotivé, et qui a toujours été à mes côtés dans mes études et en dehors des études, j'ai de la chance d'avoir un ami tel que lui.

À tous mes collègues, en particulier Boudjenene Ali et Kadri Nadir, qui m'ont beaucoup aidé durant notre cursus d'études et qui m'ont

À tous ma famille que je ne cesse jamais d'aimer, qui m'a toujours donné de la motivation pour concrétiser ce projet et le mener à bien jusqu'à la fin.

À tous mes autres amis.

Akram

DOUDOU

## Table des matières :

Chapitre 1 : .....	9
1.1 Définition de la vidéosurveillance : .....	14
	4

1.2	Historique de la vidéosurveillance : .....	14
1.3	Approche sociologique et politique : .....	14
1.3.1	En Grande-Bretagne : .....	14
1.3.2	En France : .....	15
1.3.3	En Suisse : .....	15
1.3.4	Autres pays : .....	15
<b>Chapitre 2 :</b>	.....	17
2.1	Détection de véhicules : .....	18
2.2	Les réseaux de neurones : .....	19
2.3	L'algorithme YOLO : .....	23
2.4	Les différentes approches basées sur la détection des véhicules .....	24
2.4.1	Les véhicules autonomes : .....	24
2.4.2	SINet, un réseau de neurones à convolution insensible à l'échelle pour la détection rapide de véhicules : .....	27
2.4.3	Cadres évolutifs pour la détection rapide de véhicules : .....	28
2.4.4	Formation d'un détecteur d'objet rapide pour les images de la gamme LiDAR à l'aide de données étiquetées provenant de capteurs à résolution supérieur : .....	29
2.4.5	LiStereo : Générer des cartes de profondeur dense à partir d'images LIDAR et stéréo : .....	31
2.4.6	Détection simultanée d'objets et segmentation sémantique : .....	32
2.4.7	Détection améliorée de l'espace libre dans plusieurs voies basée sur un seul CNN avec identification de la scène : .....	32
2.4.8	RRPN : Réseau de propositions de régions radar pour la détection d'objets dans les véhicules autonomes : .....	33
2.4.9	Classification des modèles de véhicules par apprentissage en profondeur : .....	34
<b>Chapitre 3 :</b>	.....	36
3.1	Introduction .....	37
3.2	Programmes utilisés : .....	37
3.1.1	Python .....	37
3.1.2	Numpy .....	38
3.1.3	Matplotlib .....	38
3.1.4	OpenCV .....	38
➤	OpenCV et la détection des véhicules : .....	39
3.1.5	TensorFlow .....	40
➤	Qu'est-ce que TensorFlow ? .....	40
➤	Les avantages de Tensorflow .....	41
➤	L'API de Tensorflow .....	42
3.1.6	Keras .....	42
3.1.7	Yolo ( You Only Look Once ) .....	43
➤	Comment fonctionne YOLO ? .....	43

➤	Les avantages de YOLO dans la détection des véhicules .....	45
3.3	Outils d'étiquetage : .....	45
	LabelImg .....	45
➤	Les fichiers générés par LabelImg : .....	45
➤	L'avantage de labelImg : .....	46
3.4	La base de données utilisée .....	47
	<i>Cars</i> .....	47
3.5	Le matériel utilisé : .....	47
	Conclusion : .....	47
	<b>Chapitre 4 :</b> .....	48
	Introduction .....	49
4.1	Préparation de la base de données : .....	50
➤	Collecte de données : .....	50
➤	Etiquetage des images : .....	50
➤	Redimensionnement des images: .....	51
4.2	Traitement sur le modèle: .....	51
➤	Choix du modèle: .....	51
➤	Training du modèle : .....	52
4.3	<i>Essaie du modèle sur l'algorithme :</i> .....	55
➤	Taux de reconnaissance des images par le modèle choisi : .....	55
4.3.1	YoloV2 : .....	55
4.3.2	YoloV3 (Original) : .....	56
4.3.3	YoloV3 (entraîné sur notre base de données) : .....	57
	<b>Conclusion :</b> .....	58
	<b>Perspectives :</b> .....	58
	Conclusion Générale : .....	59
	Bibliographie : .....	60

## Table des figures :

Figure 1 : framework du modèle proposé .....	22
Figure 2 : exemple d'utilisation de l'algorithme YOLO .....	23
Figure 3: L'algorithme de détection des véhicules utilisé par les voitures autonomes.....	25
Figure 4: véhicule sur lequel les travaux on était effectués .....	26
Figure 5: statistiques et comparaison des différents modèles par rapport au modèle SINet....	27
Figure 6 Architecture du framwork proposé .....	29
Figure 7 : capteurs utilisés.....	30
Figure 8: Résultats obtenus .....	30

Figure 9: tableau comparatif des méthodes utilisées.....	31
Figure 10: fonctionnement général du système de détection améliorée .....	33
Figure 11: résultats obtenus avec l'algorithme proposé .....	34
Figure 12: Exemple de réseau d'illustration de réseaux de neurones avec tensorflow .....	41
Figure 13: Exemple d'application de l'API de Tensorflow .....	42
Figure 14: Etapes de traitement de l'algorithme YOLO.....	43
Figure 15: Comparaison de la performance des différents modèles actuels .....	44
Figure 16: Outil d'étiquetage et les deux types de fichiers qu'il peut générer.....	45
Figure 17: Fichier sous format YOLO généré par LabelImg .....	46
Figure 18: Fichier sous format YOLO généré par LabelImg .....	46
Figure 19: Exemple d'utilisation de LabelImg .....	47
Figure 20 : schéma représentatif de l'ensemble du travail .....	49
Figure 21: base de données .....	50
Figure 22 : choix du nom de l'étiquetage.....	50
Figure 23: : sélection du cadre représentatif d'une voiture.....	50
Figure 24: changement de la résolution des images.....	51
Figure 25: graphe représentatif des taux de reconnaissance .....	52
Figure 26: exemple de détection Car.....	53
Figure 27: exemple de détection non-Car .....	53
Figure 28 exemples de fausse prédiction non-Car .....	54
Figure 29: test des modèles sur nos propres images .....	55
Figure 30: application de l'algorithme YOLOv2 sur nos vidéos.....	55
Figure 31: application de l'algorithme YOLOv3 sur nos vidéos.....	56
Figure 32: application de l'algorithme YOLOv3 pré-entraîné sur nos vidéos .....	56
Figure 33: application de l'algorithme YOLOv3 pré-entraîné sur une autre de nos vidéos.....	57
Figure 34 : comparaison de l'application des modèles sur nos vidéos .....	58

## Introduction générale

De nos jours, la vidéosurveillance est omniprésente que ce soit dans les secteurs d'activité tel que les banque, les transports ou l'industrie, Ou lieux de vie (villes, immeubles de bureau, équipements collectifs, etc.), mais contenu du nombre impressionnant de caméra de surveillance actuel et plus précisément les caméras de surveillance du trafic routier, il est devenu impossible de pouvoir surveiller tous les écrans en même temps, la détection de véhicules est donc apparue, et a permis de surveiller tous les écrans simultanément.

Dans ce projet nous allons voir les différentes avancées en matière de détection de véhicules. Nous allons, par la suite, créer notre application de détection des véhicules en se basant sur les modèles et les algorithmes de détection les plus sophistiqués pour pouvoir atteindre une vitesse et précision de détection équivalente à ce qui se fait de mieux de nos jours dans le domaine de la détection des véhicule.



# **Chapitre 1 :**

# **Généralités**

# Introduction :

L'importance de la technique de surveillance vidéo en tant que composant technique d'un concept de sécurité global et homogène ne cesse d'augmenter. Les systèmes modernes permettent une détection ainsi qu'une analyse ciblée des événements uniquement par l'observation et la documentation des processus relatifs à la sécurité. Toutes les images peuvent être visualisées dans un ou plusieurs postes de surveillance. Combinée à des techniques de sécurité et de signalisation de danger, la technique vidéo permet une évaluation globale de la situation sans avoir à se rendre sur le lieu de l'événement.

Au cours des dernières années, la technique vidéo s'est considérablement développée. Qu'une installation soit basée sur une technique analogique ou IP, le choix des caméras, des objectifs, des boîtiers de protection et des consoles sont identiques. Les différences entre la technique analogique et la technique IP ne sont perceptibles que lors de la transmission des signaux d'images.

Le fonctionnement d'une caméra est comparable à celui d'un œil humain. Elle perçoit un objet ou une personne via un objectif et enregistre la prise de vue sur une puce. Les installations de sécurité actuelles sont de plus en plus nombreuses et complexes et présentent également des exigences de sécurité variables. Grâce à un système de surveillance vidéo, le service de sécurité est constamment informé, 24 heures sur 24, 7 jours sur 7. Les solutions actuelles vont de différents types de caméras aux systèmes d'analyse d'images, en passant par les postes de gestion vidéo qui évaluent les images enregistrées et avertissent automatiquement le personnel compétent lorsqu'un événement survient.



### 1.1 Définition de la vidéosurveillance :

La vidéosurveillance est un système de caméras et de transmission d'images, disposé dans un espace public ou privé pour le surveiller à distance ; il s'agit donc d'un type de télésurveillance. Les images obtenues avec ce système, peuvent être traitées automatiquement et/ou visionnées puis archivées ou détruites. La surveillance a pour but de contrôler les conditions de respect de la sécurité, de la sûreté ou de l'exécution d'une procédure particulière.

Le principe de filmer et de contrôler un phénomène à distance a été utilisé pendant la Seconde Guerre mondiale par les Allemands pour des raisons de sûreté : pour observer le lancement de leurs missiles. Mais on ne parle de télésurveillance stricto sensu qu'à partir du moment où le système est industrialisé et permet à un opérateur (surveillant) de contrôler simultanément plusieurs lieux sur une batterie d'écrans, ceci à des fins civiles.



Durant les années 1980, le Royaume-Uni a été le premier pays au monde à généraliser ce système (suite aux attentats de l'IRA). Il reste actuellement le pays d'Europe le plus « télé-surveillé », Londres étant réputée comme la ville où la vidéosurveillance tant publique que privée est la plus importante. Des politiques de vidéosurveillance ont été mises en place dans plusieurs villes européennes durant les années 1990.

Selon ses partisans, la vidéosurveillance permet de prévenir le terrorisme et plus généralement la criminalité (hold ups, cambriolages, agressions sur voirie, etc.) et d'opérer un contrôle social (mouvements de foule, etc.)

À l'opposé, ses détracteurs lui reprochent fondamentalement son atteinte à la vie privée mais aussi son coût et son inefficacité. Ainsi plusieurs rapports, notamment aux États-Unis et au Royaume-Uni, esquissent-ils un bilan critique. Un représentant de Scotland Yard à la Security Document World Conférence d'avril 2008 parle ainsi de « utter fiasco » (« échec complet »), parce que les officiers de police ne sont pas assez formés, souvent ils ne veulent pas chercher les images vidéo « parce que c'est beaucoup de travail ».

Étant donné que les solutions durables en terme de surveillance de la circulation autoroutière pour toutes les catégories de routes n'ont pas été identifiées, en particulier les routes rurales et éloignées à faible trafic, une hiérarchie de contrôle devrait être appliquée, de même que les classifications utilisées pour améliorer la sécurité et la santé au travail. Au plus haut niveau, il est possible de prévenir durablement les blessures graves et les accidents mortels, et il est nécessaire de tenir compte de tous les domaines de résultats clés. Au deuxième niveau, la réduction des risques en temps réel, qui consiste à fournir aux utilisateurs un avertissement spécifique pour leur permettre de prendre des mesures d'atténuation. Le troisième niveau consiste à réduire le risque de collision impliquant l'application des normes et des lignes directrices sur la conception des routes (par exemple, de la Ligue contre la violence routière ou de American Association of State Highway and Transportation Officials), ce qui améliore le comportement du conducteur et son application.

### 1.2 Historique de la vidéosurveillance :

Le premier système de vidéosurveillance fut installé par **Siemens AG** en 1942 en Allemagne pour observer le lancement des fusées V-2. Un système est commercialisé en 1949 (**Vericon**) aux US sans avoir besoin d'une autorisation du gouvernement. En 1949, l'écrivain anglais George Orwell décrit dans son roman d'anticipation 1984 un univers entièrement sous la coupe d'un personnage appelé Big Brother (en réalité une incarnation de l'État totalitaire) capable d'identifier les faits et gestes de toute une population grâce à un immense parc de caméras disséminées partout dans la ville.

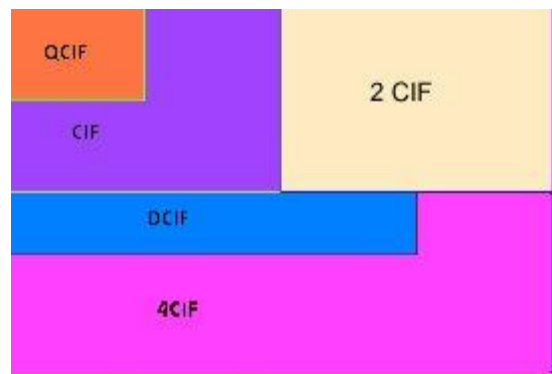


**Marie Van Brittan Brown** Brevette en 1969 un système de sécurité domestique, avec une caméra qui peut être dirigée vers quatre points d'observation avec un moniteur pour voir une personne qui se présente devant la porte d'entrée. Comme les premiers systèmes n'enregistrent pas, ils nécessitent une surveillance humaine permanente. Le développement de la vidéo et des cassettes permet d'enregistrer et de détruire les images, rendant l'usage beaucoup plus fréquent dans les années 1970, et sa popularisation dans des séries policières comme Colombo est constante. En 1968, la ville d'Olean, dans l'État de New York, est la première aux USA à surveiller ses rues, dans un objectif de lutte contre la criminalité. En 1973, Times Square est équipée. En Grande-Bretagne en 1985, la ville de Bournemouth s'équipe, puis les autorités locales de King's Lynn en 1987. Les premières statistiques semblent ne pas montrer d'effet. Le développement de la vidéo-surveillance des espaces publics et des entreprises dans les années 1980 et 90, s'étend à des bâtiments divers (écoles, banques, parking, parcs) publics et privés. En 1998, 3000 systèmes de vidéo-surveillances fonctionnent à New York



Initialement, en France, on parlait de circuit fermé, puisque la diffusion des images était interne au bâtiment, par opposition à une diffusion vers l'extérieur de la télévision. L'évolution technologique passe par l'amélioration des caméras. La première caméra vidéo portable au monde date de la fin des années 1970 (premier caméscope en 1983). La mise en place est liée aux régimes d'autorisations juridiques, en particulier sur l'espace public. Les caméras actuelles les plus performantes sont de type numérique, en couleur, et permettent des zooms et une

bonne mise au point. En France, la définition des caméras est réglementairement définie. La définition requise est dite 4 **CIF**, soit  $704 \times 576$  pixels. Cette définition est très rarement atteinte par les anciennes caméras ou même certaines toujours sur le marché, en général en CIF soit  $352 \times 288$  pixels, ou VGA, soit  $640 \times 480$  pixels. Il est possible d'avoir une caméra de résolution plus faible si elle permet de prendre une « vignette de visage » pour identification de  $90 \times 60$  pixels. Sur les anciennes caméras, cela signifie que le visage doit représenter 5 % environ de la superficie de l'image (1 % en 4 CIF). Par ailleurs, le nombre d'images par seconde requis est de 6 ou 12, selon la situation, lente ou rapide, à surveiller. Les nouvelles installations doivent se conformer à la loi. C'est l'utilisateur du système qui est responsable de sa conformité à la loi.



### 1.3 Approche sociologique et politique :

#### 1.3.1 En Grande-Bretagne :

La vidéosurveillance fut introduite à Londres pour la première fois suite aux attaques de l'IRA. Au Royaume-Uni, une politique de vidéosurveillance de grande ampleur a été entreprise dès le début des années 1990. Aujourd'hui, les caméras au Royaume-Uni couvrent la plupart des centres-villes, et de nombreuses gares et parkings. On avance des chiffres approximatifs allant de 65 000 à 500 000 caméras à Londres et plus de 4 millions au Royaume-Uni au total. Les critiques pointant la totale inefficacité de ce déploiement n'ont pas permis une modification de la politique de vidéosurveillance de ce pays. Un rapport d'Eric Heilmann publié en 2003 pour l'Institut des hautes études de la sécurité intérieure de Grande-Bretagne indique qu'il est impossible d'affirmer que ces caméras ont eu une influence sur la criminalité ou la délinquance. Par contre un simple citoyen faisant ses courses dans les rues de Londres est filmé plus de trois cents fois.

### **1.3.2 En France :**

Une politique de vidéosurveillance a également été mise en place en France. Depuis l'initiative de Patrick Balkany dans les années 1990 à Levallois-Perret, la vidéosurveillance s'est généralisée : les professionnels reconnaissent installer chaque année entre 25 et 30 000 nouveaux systèmes de vidéosurveillance. En 2007, le nombre de caméras « autorisées » (donc dans l'espace public) était estimé à 340 000 mais pourrait atteindre un million d'ici peu. Ces caméras sont présentes dans les aéroports et les gares, autour des routes, dans les transports publics. Dans la circulaire réglementaire de février 2009 « relative aux objectifs en matière de sécurité intérieure », Michèle Alliot-Marie disait fixer comme objectif de « parvenir à 60 000 caméras sur la voie publique d'ici 2010.

Le site d'information **OWNI** a publié en décembre 2011 un "palmarès des villes sous surveillance", étudiant les politiques municipales en matière de vidéosurveillance. Ce palmarès (qui classe Nice en tête et souligne le peu de villes sans caméras) constate que le sujet est une ligne (très générale) de clivage gauche droite, les villes tenues par la droite étant équipées, en moyenne, de trois fois plus de caméras par habitant.

### **1.3.3 En Suisse :**

Un système de vidéosurveillance est présent dans tous les trains Suisses ainsi que presque tous les bus et tram. Le record est pour le RER de Zurich, avec plus de 6 000 caméras que dans les trains et dans les gares. D'ici décembre 2012, toutes les gares du canton de Zurich ainsi que 300 autres gares auront un système de surveillance vidéo, ce sont ainsi plus de 2 300 caméras qui seront installées dès 2011. Des caméras sont aussi installées dans les quartiers fréquentés par les prostituées et ce pour garantir leur sécurité. C'est le cas à Olten dans le cadre d'un projet pilote. La ville de Zurich possède 2 500 caméras de surveillance pour la sécurité des habitants, Genève en possède 1 000, Lausanne 700 et Berne 800.

Les autoroutes suisses sont toutes sous surveillance vidéo, on en compte plus de 9 000 sur le réseau routier suisse. Le Conseil fédéral suisse a défendu l'installation de caméras de vidéosurveillance dans les trains, en invoquant la nécessité de « renforcer la sécurité dans les gares et les trains ».

### **1.3.4 Autres pays :**

Certains pays ont mis en place des lois pour réglementer la mise en place des caméras de vidéosurveillance. C'est le cas de la Nouvelle-Zélande, où des lois relatives à l'installation de caméras ont été instaurées. Ces dernières prévoient notamment la consultation des personnes touchées par la mise en place de surveillance électronique. En Allemagne, la législation ne permet la vidéosurveillance que pour des lieux publics dans lesquels a été constaté un taux de délinquance élevé.

Après avoir présenté de façon générale la vidéo-surveillance, l'intérêt majeur de notre sujet de recherche est la sécurité des automobilistes, la réduction le nombre d'accidents mortels, ainsi que la gestion du trafic autoroutier dans ses différents aspects, à savoir : la détection des accidents, la détection des conduites dangereuses et la gestion générale du trafic (régulation, monitoring) tout en minimisant le coût du personnel, Grace au nombre de caméra installé ces derniers temps, ce projet pourrais prendre forme et aboutir à une sécurité totale des routes algériennes.

Nous allons entamer notre projet en présentant l'avancée technologique concernant le domaine de la vidéo-surveillance dédiée à la circulation autoroutière dans le chapitre suivant.



# **Chapitre 2 :**

## **Etat de l'art**

Dans ce chapitre, nous présentons l'état de l'art sur les recherches actuelles effectuées sur la vidéo-surveillance, en se focalisant plus sur la détection des véhicules.

## **2.1 Détection de véhicules :**

La détection de véhicules sur des images aériennes est une tâche importante et difficile. Traditionnellement, de nombreux modèles de détection de cible basés sur la mode à fenêtre coulissante ont été développés et ont obtenu des performances acceptables, mais ces modèles prennent beaucoup de temps dans la phase de détection. Récemment, avec le grand succès des réseaux de neurones à convolution (CNN) en vision par ordinateur, de nombreux détecteurs à la pointe de la technologie ont été conçus à partir de CNN profonds. Cependant, ces détecteurs basés sur CNN sont inefficaces lorsqu'ils sont appliqués dans des données d'image aérienne en raison du fait que les modèles existants basés sur CNN rencontrent des difficultés avec la détection d'objet de petite taille et la localisation précise. Pour améliorer la précision de la détection sans diminuer la vitesse, les chercheurs ont proposé un modèle de détection basé sur CNN combinant deux réseaux de neurones à convolution indépendants. Où le premier réseau est appliqué pour générer un ensemble de régions semblables à des véhicules à partir de cartes à caractéristiques multiples de différentes hiérarchies et échelles. Étant donné que les cartes à caractéristiques multiples combinent les avantages de la couche de convolution profonde et peu profonde, le premier réseau se comporte bien pour localiser les petites cibles dans les données d'image aérienne. Ensuite, les régions candidates générées sont introduites dans le deuxième réseau pour l'extraction de caractéristiques et la prise de décision. Des expériences exhaustives sont menées sur l'ensemble de données VEDAI (détection de véhicules en imagerie aérienne) et sur les véhicules à Munich. Le modèle de détection en cascade proposé offre des performances élevées, non seulement en précision de détection, mais également en vitesse de détection. Le premier réseau réussit bien à localiser les petites cibles dans les données d'images aériennes.

La détection de véhicules est une tâche pertinente dans divers domaines, tels que : la télédétection, le transport intelligent et la reconnaissance militaire. Avec le grand développement des technologies de véhicule aérien sans pilote (UAV), les images aériennes sont capturées de manière pratique et flexible de cette manière. Pour les données d'imagerie aérienne en pleine croissance, la détection de véhicules est devenue un défi qui a récemment attiré l'attention. En tant que tâche fondamentale en vision par ordinateur, la détection de

véhicules est largement étudiée dans certaines applications pratiques, telles que la surveillance du trafic et la conduite d'assistants de sécurité, mais pour les images aériennes, le problème reste difficile en raison de l'obscurité, de la taille relativement petite des cibles et des arrière-plans encombrés. En outre, d'autres objets, tels que les grands conteneurs et les marques de route, présentent toujours un aspect similaire à celui des véhicules, ce qui entraînera une détection erronée ou une perte de précision. En outre, dans un modèle de détection, non seulement l'exactitude de détection est requise, mais également une bonne vitesse de détection.

## **2.2 Les réseaux de neurones :**

Au cours de la dernière décennie, la technologie de détection de cible a beaucoup évolué et peut être divisée en trois étapes. Au cours de la première étape, la combinaison de fonctionnalités conçues à la main et de classificateurs discriminants a été utilisée pour détecter des cibles. D'une part, certaines méthodes classiques telles que l'Histogramme de dégradé orienté (HOG) et la transformation d'instruments à l'échelle variable (SIFT) ont été conçues pour l'extraction d'entités. Par ailleurs, les classificateurs discriminants tels que Support Vecteur Machine (SVM) et Ada-Boost ont été adoptés pour la classification. Felzenszwalb et al (2010) a proposé un modèle de pièces déformables (DPM), qui utilise divers composants entraînés pour détecter des cibles à partir d'une pyramide d'images en mode fenêtre glissante. Bien que DPM soit un excellent détecteur, la stratégie de la fenêtre glissante prend beaucoup de temps dans la phase de détection. Dans la deuxième étape, la méthode de la fenêtre coulissante a été remplacée par une méthode de proposition de région. Cela signifie que les détecteurs n'ont pas besoin de détecter les cibles à partir de la pyramide d'images, mais à partir de milliers de régions candidates ressemblant à des cibles. C'est un moyen très efficace de réduire le temps de détection. Par exemple, les régions candidates d'une image (de taille  $400 \times 500$ ) représentent environ  $10^3$ , ce qui est beaucoup moins que l'espace de recherche (environ  $10^4 \sim 10^5$ ) de la pyramide de l'image par une fenêtre coulissante. La troisième étape a débuté en 2012, lorsque Krizhevsky et al (2012) ont appliqué la méthode des réseaux de neurones à convolution (CNN) à un défi de classification d'images (ILSVRC2012) et ont obtenu des résultats saisissants, qui ont propulsé les méthodes basées sur CNN dans le champ de la vision par ordinateur. Récemment, Girshick et al (2014) et Sermanet et al (2013) ont proposé des modèles de détection efficaces basés sur les CNN. En particulier, la méthode décrite appelée Régions avec CNN (R-CNN), est devenue la base de référence du cadre de détection. Le flux de travail de R-CNN est

principalement divisé en deux étapes : (a) il utilise la méthode de proposition de région décrite dans pour générer un ensemble de régions candidates, puis (b) ces régions sont gendonnées en une taille fixe et introduites dans un CNN pour en extraire les caractéristiques profondes. D'après les nombreux résultats expérimentaux, les caractéristiques de CNN montrent une capacité de discrimination plus grande que les caractéristiques traditionnelles conçues à la main. Il est à noter que la méthode de proposition de région prend toujours plusieurs secondes sur une image de taille moyenne ( $500 \times 300$  pixels, par exemple) et que les entités CNN des différentes régions seraient extraites à plusieurs reprises. Ensuite, des méthodes améliorées nommées SPP-Net et Fast R-CNN ont été proposés pour accélérer la vitesse de détection. Dans Fast R-CNN, une stratégie de région d'intérêt (ROI) a été utilisée pour traiter le problème de l'extraction répétée d'entités CNN, ce qui accélère considérablement la procédure d'extraction d'entités CNN. Un autre goulot d'étranglement majeur de R-CNN est les coûts de calcul de la procédure de proposition de région. Ren et al (2017) ont proposé une architecture basée sur CNN appelée Région Proposition Réseau (RPN) pour remplacer la méthode décrite. Ils ont combiné RPN avec Fast R-CNN et ont formé un modèle de détection unifié, qui permet d'obtenir des performances de pointe sur les bases de données **PASCAL 2007/2012**<sup>1</sup> et **MS COCO**<sup>2</sup>. La vitesse de détection a atteint 5 fps avec un réseau VGG-16.

Bien que les travaux montrent des résultats prometteurs en matière de détection de cible, ils ne conviennent pas aux images aériennes. La première raison est que les véhicules de l'image aérienne sont de taille relativement petite (la taille moyenne d'un véhicule est de  $40 \times 20$  pixels) et en raison de la grosseur de la carte des caractéristiques (sortie de la couche de convolution profonde du CNN). RPN a des performances de localisation médiocres pour les petites cibles. De plus, les modèles de détection sont conçus pour la détection multi-catégories, mais pour la catégorie spécifique «véhicule», ils fonctionnent mal à cause des faux positifs. La deuxième raison est que les véhicules apparaissent toujours comme des toits de véhicule sur des images aériennes, ce qui présente des similitudes avec d'autres cibles d'arrière-plan. Cela entraînerait une perte de précision sans formation spécifique. De plus, contrairement aux jeux de données publics à grande échelle (tels que **ImageNet**<sup>3</sup> et **MS COCO**) comprenant des millions d'images, les données de formation des jeux de données d'images aériennes annotées disponibles (pour les véhicules) sont insuffisantes.

Dans cet article, un modèle CNN en cascade a été proposé permettant de détecter des véhicules dans des données d'imagerie aérienne, ce qui permet de maintenir une précision de détection élevée et une vitesse élevée. Le cadre de notre modèle, illustré à la [figure 1], comprend deux

**PASCAL 2007/2012**<sup>1</sup> : base de données utilisée pour reconnaître des objets d'un certain nombre de classes d'objets visuels dans des scènes réalistes. 20

**MS COCO**<sup>2</sup> : ou bien Microsoft COCO, est une base de données ayant pour but de faire progresser la reconnaissance des objets.

**ImageNet**<sup>3</sup> : est une base de données d'images annotées produit par l'organisation du même nom, à destination des travaux de recherche en vision par ordinateur.

réseaux basés sur CNN. Le premier réseau est appelé réseau de propositions de régions de véhicule (VPN), qui vise à générer les régions analogues à des véhicules. La deuxième partie est le réseau de détection de véhicule (VDN) qui prend des décisions pour les régions générées par le premier réseau. Le flux de travail de la phase de détection est divisé en trois étapes : (1) une image d'entrée est insérée dans le VPN pour générer des régions candidates, (2) les régions générées incorporant l'image d'entrée sont introduites dans le VDN afin d'extraire les caractéristiques de chaque région et de prédire le résultat. Pour score de confiance, (3) les régions avec un score élevé (supérieur à un seuil) sont sorties en tant que détections. Par rapport au travail, le modèle présente trois différences principales : (1) contrairement au travail qui forme un réseau unifié, les chercheurs ont formé deux réseaux indépendants. Cela signifie que les couches de convolution de deux réseaux ne sont pas partagées, ce qui évite de refaire la formation des couches non partagées de deux réseaux; (2) les cartes de caractéristiques générées par les couches à convolution profonde (de CNN) peuvent détecter la cible avec un rappel élevé mais une performance de localisation médiocre, tandis que la carte de caractéristiques des couches peu profondes offre une meilleure performance de localisation mais permet d'obtenir un rappel réduit. Pour tirer parti des deux, les chercheurs ont combiné les cartes de caractéristiques des couches peu profondes et des couches profondes afin de générer les régions semblables à des véhicules à différentes échelles et hiérarchies. De cette manière, la méthode des chercheurs permet d'obtenir des régions plus fines et plus précises analogues à des véhicules que les RPN ; (3) le VDN est formé en tant que détecteur de catégorie spécifique appliqué à la détection de véhicules de type multiple.

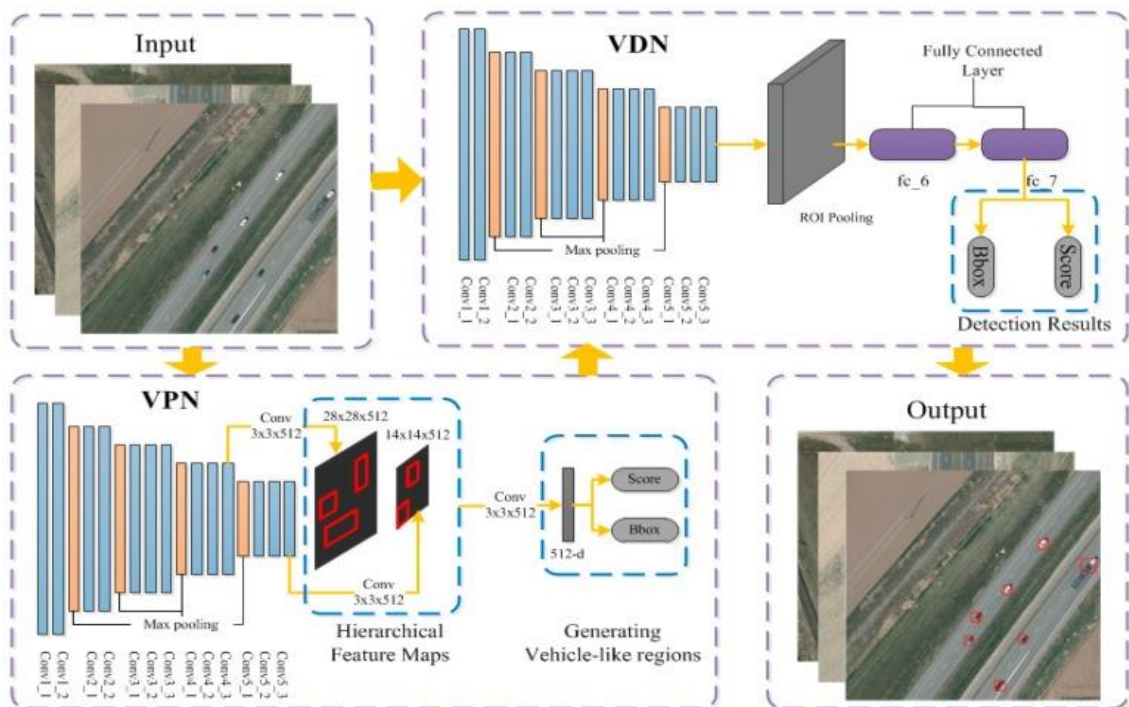


Figure 1 : framework du modèle proposé

Les approches basées sur CNN ont montré un pouvoir de représentation riche et des résultats prometteurs. R-CNN utilise la proposition d'objet générée par la recherche sélective pour former CNN aux tâches de détection. Dans le cadre R-CNN, SPP-Net et Fast R-CNN accélèrent en générant une proposition de région sur une carte de caractéristiques ; ces approches n'ont besoin que d'un ordinateur. Plus rapide, R-CNN utilise un réseau de propositions de région au lieu de la recherche sélective. Il peut ensuite s'entraîner de bout en bout et la vitesse et la précision s'améliorent également. R-FCN tente de réduire le temps de calcul avec des cartes de scores sensibles à la position.



## 2.3 L'algorithme YOLO :

Compte tenu de la grande efficacité, l'approche en une étape attire beaucoup plus l'attention récemment. YOLO utilise un seul réseau de neurones à convolution à feed-forward pour prédire directement les classes d'objets et les emplacements, ce qui est extrêmement rapide.

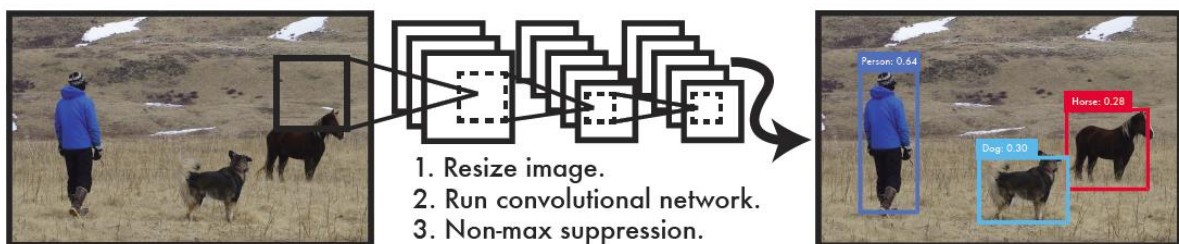


Figure 2 : exemple d'utilisation de l'algorithme YOLO

Comme le montre la figure précédente, on remarque que le modèle YOLO se compose uniquement de couches à convolution et des couches de MaxPooling. La taille du filtre utilisé dans chaque couche varie (7x7, 3x3, 1x1 et autre).

## **2.4 Les différentes approches basées sur la détection des véhicules**

### **2.4.1 Les véhicules autonomes :**

Le véhicule autonome est le véhicule du futur. Les perspectives de ce dernier sont considérables. En effet, énormément d'acteurs travaillent sur des projets concernant la voiture autonome que ce soit des constructeurs automobiles comme par exemple, Renault, Audi, Mercedes Benz, Tesla, ainsi que des entreprises spécialisées dans un tout autre domaine, comme Bosch, Google ou encore Uber. Tous annoncent vouloir commercialiser la voiture autonome pour 2020.

Google travaille sur la mise au point des technologies utilisées pour la Google Car. Le géant de la High-tech refuse d'ajouter des fonctionnalités semi-autonome aux voitures, et se donne l'objectif de produire une voiture complètement autonome, c'est à dire sans frein ni volant d'ici 2020.



Après avoir lancé des voitures Ford dotées de la technologie auto-conduite à Pittsburgh, ville qui compte plus de 2,6 millions d'habitants, le PDG de la start-up californienne promet près de 500 000 Tesla-Uber autonomes en 2020. Cette dernière a aussi accepté une alliance avec le constructeur automobile Volvo de 300 millions de dollars pour développer des voitures sans conducteur.





Figure 3: L'algorithme de détection des véhicules utilisé par les voitures autonomes

Le véhicule autonome Tesla, pour sa part prend de l'avance vis à vis de ses concurrents et annonce une voiture auto conduite prête pour 2018. En effet, le modèle S, le modèle X actuellement en production seront dotées d'une nouvelle technologie leur permettant de circuler totalement sans conducteur.

Les voitures ne seront pas les seules à être autonomes. En effet, les bus automatisés représentent un véritable défi dans le cadre de la mobilité urbaine. Le bus autonome est une solution prometteuse pour les problèmes environnementaux, d'embouteillages et de mégapoles. De plus, plus d'une dizaine de villes dans le monde expérimentent ces bus dans les rues, citons en premier lieu l'état d'Ohio qui met en place un système de bus autonome dans la ville et en deuxième lieu Lyon qui teste pendant un an (09.2016 à 09.2017) un service de navette avec des minibus électriques et autonomes.

En s'appuyant sur la figure en dessus, on peut dire que l'utilité de la détection des véhicules par les voitures autonomes est primordiale car pour que les voitures autonomes arrivent à s'adapter à l'environnement soumis, elles ont besoin de d'observer et détecter ce qu'il y a autour d'elles

pour pouvoir prendre une décision sûre (par exemple, détecter s'il y a des voitures en avant pour décider d'avancer ou pas).



*Figure 4: véhicule sur lequel les travaux on était effectués*

Suite à un partenariat entre Transdev (Caisse Des Dépôts, Veolia) et Renault Nissan, Rouen accueillera en 2018 un service de voitures autonomes en complément des transports en commun. Ces voitures autonomes électriques de modèle Zoé se situeront au Technopôle du Madrillet, à Saint Etienne Du Rouvray. Ils circuleront avec les autres véhicules (bus et automobiles) et franchiront des ronds-points et passages piétons, sur lesquels auront été installés des technologies communicantes. Il s'agit d'un système de transport complet qui permettra au client de réserver son trajet, et aux opérateurs de gérer et d'exploiter une multitude de voitures autonomes en un même temps.

## 2.4.2 SINet, un réseau de neurones à convolution insensible à l'échelle pour la détection rapide de véhicules :

Les approches de détection de véhicules basées sur la vision rencontrent un succès incroyable ces dernières années avec le développement du réseau de neurones à convolution profonde (CNN). Cependant, les algorithmes existants basés sur les CNN souffrent du problème suivant: les paramètres de convolution sont sensibles à l'échelle dans la tâche de détection d'objet, mais il est fréquent que les images et les vidéos de trafic contiennent des véhicules avec une grande variance d'échelles. Dans l'article de TOTS (Transactions On Intelligent Transportation Systems), les auteurs explorent la source de la sensibilité d'échelle et révèlent deux problèmes clés :

- ✓ Le pooling existant de RoI détruit la structure d'objets de petite taille.
- ✓ La grande distance intra-classe pour une grande variance d'échelles dépasse la capacité de représentation d'un seul réseau. Sur la base de ces résultats, ils présentent un réseau de neurones à convolution insensible à l'échelle (SINet) pour la détection rapide de véhicules présentant une grande variance d'échelles. Premièrement, ils présentent un pooling de RoI sensible au contexte pour maintenir les informations contextuelles et la

Model	Time/Image	Strategy 1							Strategy 2			
		Mean	Sparse			Crowded			Sparse			
			Car	Bus	Van	Car	Bus	Van	Mean	Car	Bus	Van
SINet_VGG (ours)	0.20s	<b>70.17</b>	<b>81.82</b>	<b>85.60</b>	<b>78.65</b>	<b>56.80</b>	<b>55.78</b>	62.38	<b>78.71</b>	74.51	<b>84.34</b>	<b>77.27</b>
SINet_PVA (ours)	<b>0.08s</b>	70.04	81.40	84.39	77.39	53.76	54.06	<b>69.23</b>	77.69	<b>74.79</b>	81.35	76.92
MS-CNN [28]	0.23s	63.23	79.94	83.71	76.79	51.74	32.95	54.26	72.66	71.13	74.34	72.50
Faster RCNN [27]	0.31s	46.44	60.93	66.68	60.14	26.08	24.55	40.24	40.22	36.19	42.79	41.69
YOLOv2 [45]	<b>0.03s</b>	43.82	59.71	65.51	58.35	17.39	21.55	40.42	54.00	53.16	53.88	54.96
YOLO [44]	<b>0.03s</b>	16.53	23.06	31.13	22.44	3.87	8.35	10.32	23.78	22.97	24.52	23.85



Figure 5: statistiques et comparaison des différents modèles par rapport au modèle SINet

structure originale des objets de petite taille. Deuxièmement, ils présentent un réseau décisionnel multi-branches afin de minimiser la distance intra-classe des caractéristiques. Ces techniques légères n'entraînent aucune complexité en temps supplémentaire, mais apportent une amélioration notable de la précision de détection. Les techniques proposées peuvent être équipées de n'importe quelles architectures de réseau profond et les maintenir entraînés de bout en bout.

Leurs SINet arrive à réaliser des performances de pointe en termes de précision et de rapidité (jusqu'à 37 images par seconde) avec la référence KITTI et un nouveau corpus concernant l'autoroute, qui contient une grande variance d'échelles et des objets extrêmement petits.

Le tableau en haut montre la comparaison de différents modèles (SINet\_VGG, SINet\_PVA, Microsoft-CNN, Faster RCNN, YOLO version 1 et 2) en terme de performance de détection de différents types de véhicules (Voiture, Bus,...), en utilisant deux stratégies différentes.

### **2.4.3 Cadres évolutifs pour la détection rapide de véhicules :**

Dans l'article depuis Shanghai Key Lab of Intelligent Information Processing, School of Computer Science, Fudan University, Shanghai, China et Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai, China et School of Computer Science and Engineering, University of Washington, Seattle, USA, les chercheurs effectuent une détection rapide des véhicules à partir de caméras de surveillance du trafic. Un nouveau Framework de deep learning, à savoir les cadres évolutifs, est développé. Il propose et raffine les cadres d'objets sous différentes représentations caractéristiques. Plus précisément, leur Framework est intégré à un réseau de propositions léger afin de générer des cadres d'ancrage initiales et ainsi supprimer des régions improbables. Un réseau qui fonctionne correctement produit des caractéristiques détaillées pour ces cadres candidats. Le but est de montrer avec intrigue qu'en appliquant différentes techniques de fusion de caractéristiques, les zones initiales peuvent être affinées à la fois pour la localisation et la reconnaissance.

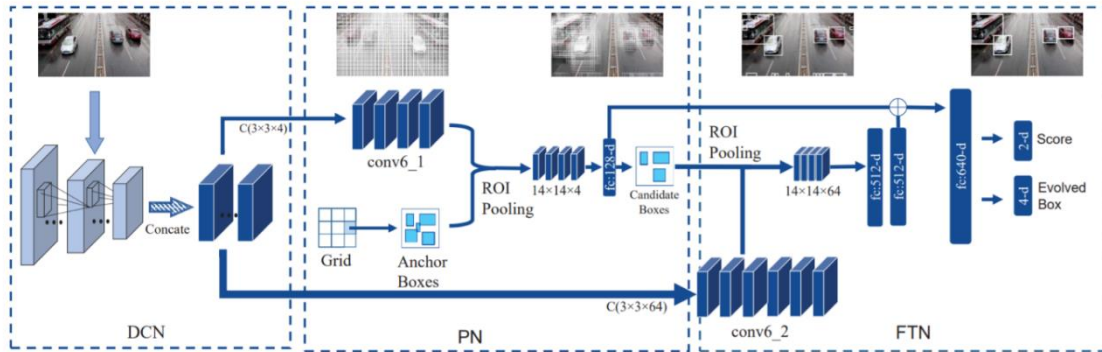


Figure 6 Architecture du framework proposé

La figure ci-dessus représente l'architecture en évolution de leur Framework. Trois réseaux sont impliqués, y compris le réseau de neurones à convolution profonde (DCN), le réseau de proposition (PN) et le réseau de qui fonctionne bien (FTN). DCN est chargé de générer de riches caractéristiques à partir des images. PN produit des propositions d'ancrage et filtre ceux qui ne sont probablement pas des véhicules. FTN peaufine les cadres candidats et génère des résultats de localisation et de reconnaissance raffinés. Concatenate : concaténation de caractéristiques provenant de différentes couches de convolutions. Le symbole + dans la figure précédente : Concaténation de caractéristiques des différents réseaux. L'évaluation du réseau a été faite sur la référence récente DETRAC et obtiennent une amélioration significative par rapport à la technologie de pointe Faster RCNN de 9,5%. De plus, leur réseau atteint une vitesse de détection de 9-13 FPS sur un GPU commercial modéré.

#### 2.4.4 Formation d'un détecteur d'objet rapide pour les images de la gamme LiDAR à l'aide de données étiquetées provenant de capteurs à résolution supérieur :

Cet article de Manuel Herzog et Klaus Dietmayer présente un modèle efficace pour la détection d'objets à partir de capteurs LiDAR (Light Detection And Ranging) dans les voitures autonomes ainsi que la stratégie pour former le modèle à l'aide de données provenant d'un différent type de capteurs LiDAR . Actuellement, les algorithmes les plus performants pour la détection d'objets à partir de mesures LiDAR sont basés sur des réseaux de neurones.

La formation de ces réseaux à l'aide d'un apprentissage supervisé nécessite de vastes corpus annotés. Cela a conduit à la situation suivante : la plupart des recherches utilisant des réseaux



de neurones pour la détection d'objets à partir de nuages de points LiDAR sont effectuées sur très peu de corpus disponibles au public et très petit peu de types de capteurs.

Cet article utilise un corpus annoté existant pour former un réseau de neurones pouvant être utilisé avec un capteur LiDAR ayant une résolution inférieure à celle utilisée pour enregistrer le corpus annoté. Ceci est réalisé en simulant des données provenant du capteur LiDAR de résolution inférieure en se basant sur le corpus de résolution supérieure. En outre, des améliorations sont présentées aux modèles qui utilisent des images de la plage LiDAR pour la détection d'objets. Les résultats sont validés à la fois sur des données de capteur simulées et sur des données provenant d'un capteur actuel de résolution inférieure monté sur un véhicule de recherche. Il est montré que le modèle peut prédire des objets à partir d'images à 360 ° en temps réel.



*Figure 7 : capteurs utilisés*



*Figure 8: Résultats obtenus*

### 2.4.5 LiStereo : Générer des cartes de profondeur dense à partir d'images LIDAR et stéréo :

Une carte de profondeur précise de l'environnement est essentielle à la sécurité des robots et des véhicules autonomes. Actuellement, soit des algorithmes de détection et de télémétrie par la lumière (LIDAR) ou d'appariement stéréo sont utilisés pour acquérir ces informations de profondeur. Cependant, un LIDAR de haute résolution coûte cher et produit une carte de profondeur épaisse à grande distance ; Les algorithmes d'appariement stéréo sont capables de générer des cartes de profondeur plus denses, mais sont généralement moins précis que LIDAR à longue distance. Cet article combine ces approches pour générer des cartes de profondeur dense de haute qualité.

Contrairement aux approches précédentes qui sont formées à l'aide d'étiquettes de vérité sur le terrain, le modèle proposé adopte un processus de formation auto-supervisé. Les expériences montrent que la méthode proposée est capable de générer des cartes de profondeur denses de haute qualité et fonctionne de manière robuste, même avec des entrées à faible résolution.

Method	iRMSE(1/km)	iMAE(1/km)	RMSE(mm)	MAE(mm)
LiStereo (sparse conv) + w/o GT	4.12	1.56	1379.31	355.42
LiStereo (conv) + w/o GT	3.83	1.32	1278.87	326.10
LiStereo (sparse conv) + with GT	2.22	<b>1.02</b>	894.56	<b>268.69</b>
LiStereo (conv) + with GT	<b>2.19</b>	1.10	<b>832.16</b>	283.91

Figure 9: tableau comparatif des méthodes utilisées

Les résultats de cette recherche montre qu'en combinant les deux méthodes ils arrivent à obtenir les avantages des deux pour générer des données précises, ils ont pu utiliser le LIDAR avec une résolution inférieure tout en produisant des résultats comparables. Sachant que le LIDAR à haute résolution (Velodyne 128) est dix fois supérieur en coût que le LIDAR à basse résolution (Velodyne 16) et stéréo. La méthode proposée ici offrirait une réduction massive des coûts pour une estimation en profondeur de résolution élevée.

### **2.4.6 Détection simultanée d'objets et segmentation sémantique :**

La détection d'objets et la segmentation sémantique des images de caméra sont des tâches importantes pour les véhicules automatisés. La détection d'objets est nécessaire pour que les modules de planification et de comportement puissent raisonner à propos des autres usagers de la route. La segmentation sémantique fournit par exemple des informations sur l'espace libre et des informations sur les parties statiques et dynamiques de l'environnement. De nombreuses recherches ont été menées pour résoudre ces deux tâches à l'aide de réseaux de neurones à convolution. Ces approches donnent de bons résultats mais sont exigeantes en calcul. En pratique, un compromis doit être trouvé entre les performances de détection, la qualité de la détection et le nombre de tâches. Sinon, il n'est pas possible de répondre aux besoins en temps réel des véhicules automatisés. Dans ce travail, les chercheurs proposent une architecture de réseau de neurones pour résoudre les deux tâches simultanément. Cette architecture a été conçue pour fonctionner avec environ 10 Hz sur des images de 1 MP sur le matériel actuel. Leur approche permet d'obtenir une moyenne de 61,2% de l'IoU (Intersection over Union) pour la tâche de segmentation sémantique sur le repère difficile de Cityscapes. Il atteint également une précision moyenne de 69,3% pour les voitures et de 67,7% sur le niveau de difficulté moyen du repère KITTI<sup>1</sup>.

### **2.4.7 Détection améliorée de l'espace libre dans plusieurs voies basée sur un seul CNN avec identification de la scène :**

De nombreux systèmes de navigation pour véhicules autonomes reposent sur la détection de voie. Généralement, les algorithmes traditionnels estiment uniquement la position des voies sur la route, mais un système de contrôle autonome peut également avoir besoin de savoir si le marquage d'une voie peut être traversé ou non, et quelle partie de l'espace à l'intérieur de la voie est exempte d'obstacles pour faire des décisions sûres. D'autre part, les algorithmes de détection d'espace libre ne détectent que les zones navigables, sans information sur les voies.

Des algorithmes à la pointe de la technologie utilisent des CNN pour les deux tâches, avec une consommation importante de ressources informatiques. Dans cet article, les chercheurs proposent une nouvelle approche qui estime l'espace libre à l'intérieur de chaque voie avec un seul CNN. De plus, en ajoutant seulement une petite exigence concernant la RAM et GPU, ils ont pu déduire le type de route, ce qui sera utile pour la planification du chemin.



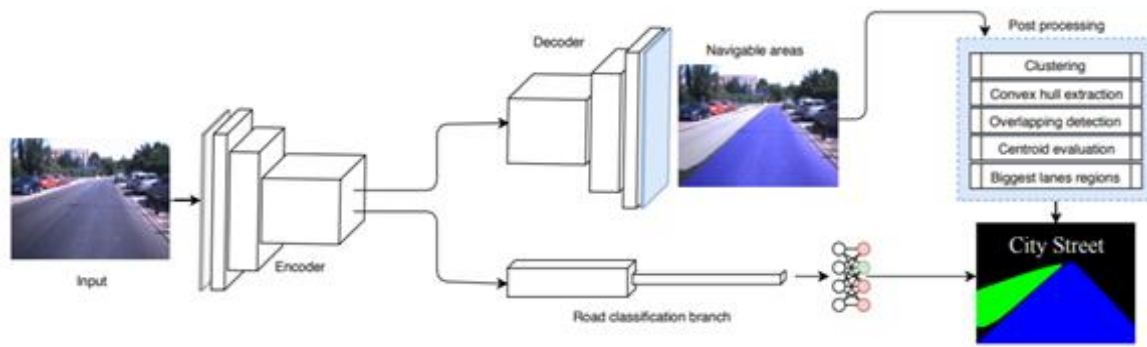


Figure 10: fonctionnement général du système de détection améliorée

Pour atteindre ce résultat, les chercheurs ont formé un CNN multitâches. Ensuite, les chercheurs ont élaboré la sortie du réseau pour extraire des polygones pouvant être utilisés efficacement dans le contrôle de la navigation. Enfin, ils fournissent une implémentation efficace en termes de calcul, basée sur les ROS, qui peut être exécutée en temps réel.

#### 2.4.8 RRPN : Réseau de propositions de régions radar pour la détection d'objets dans les véhicules autonomes :

Les algorithmes de proposition de régions jouent un rôle important dans la plupart des réseaux à la pointe de la technologie de détection d'objets à deux étages en supposant des emplacements d'objets dans chaque image. Néanmoins, on sait que les algorithmes de propositions de régions constituent le goulet d'étranglement dans ces réseaux de détection d'objets à deux étages, ce qui les rend lents et ne conviennent pas aux applications en temps réel telles que les véhicules autonomes. Dans cet article de Ramin Nabati et Hairong Qi depuis Department of Electrical Engineering and Computer Science The University of Tennessee Knoxville, USA, ils présentent un algorithme RRPN en temps réel basé sur un radar pour la détection d'objets dans les véhicules autonomes. Les régions d'intérêt (RoI) proposées sont générées en mappant les détections de radar par le système de coordonnées d'image et en générant des cadres d'ancrage prédéfinies sous forme de propositions d'objet à chaque point radar mappé.

Ils effectuent ensuite des opérations de transformation et de mise à l'échelle sur les cadres ancrés générées en fonction de la distance des objets afin de mieux les adapter aux objets détectés. Ils ont évalué leur méthode sur le nouveau corpus « NuScenes » récemment publié en utilisant le réseau de détection d'objets Fast R-CNN. Comparé à l'algorithme de proposition d'objets de recherche sélective, leur modèle fonctionne plus de 100 fois plus vite tout en offrant une précision de détection et un rappel plus élevés.



Figure 11: résultats obtenus avec l'algorithme proposé

#### **2.4.9 Classification des modèles de véhicules par apprentissage en profondeur :**

Cet article de Burak Satar et Ahmet Emir Dirik depuis Uludag University, Bursa, Turkey Department of Electrical-Electronics Engineering et Department of Computer Engineering, étudie les problèmes de classification des marques et des modèles de véhicules. Certains des principaux défis consistent à atteindre une précision de classification élevée et à réduire le temps d'annotation des images. Pour remédier à ces problèmes, les chercheurs ont créé un corpus détaillé sur les marchés des véhicules en ligne de la Turquie. Un pipeline est proposé pour combiner un modèle SSD (Single Shot Multibox Detector) avec un modèle CNN (Convolutional Neural Network) pour former en basant sur le corpus. Dans le pipeline, ils détectent d'abord les véhicules en suivant un algorithme qui réduit le temps

d'annotation. Ensuite, ils les introduisent dans le modèle CNN. Le résultat obtenu avec une précision de classification est environ 4% meilleur à celui obtenu avec un modèle CNN. Ensuite, ils proposent d'utiliser les véhicules détectés comme un cadre de délimitation de la vérité au sol (GTBB) des images et de les insérer dans un modèle SSD dans un autre pipeline. À ce stade, il est atteint un résultat de précision de classification raisonnable sans utiliser GTBB de forme parfaite. Dernièrement, une application est implémentée dans un cas d'utilisation en utilisant les pipelines proposés. Il détecte les véhicules non autorisés en comparant leurs numéros de plaque d'immatriculation et leurs marques et modèles. Il est supposé que les plaques d'immatriculation sont lisibles

# **Chapitre 3 :**

# **Outils et algorithmes**

# **utilisés**

## 3.1 Introduction

Après avoir bien mis en évidence l'état de l'art des travaux et recherches faits jusqu'à ce jour, nous allons dans ce chapitre, illustrer les différents outils et algorithmes qu'on a utilisé durant notre travail.

## 3.2 Programmes utilisés :

### 3.1.1 Python

Python est un langage de programmation open source créé par le programmeur Guido van Rossum en 1991, le langage de programmation Python apparu à l'époque comme une façon d'automatiser les éléments les plus ennuyeux de l'écriture de scripts ou de réaliser rapidement des prototypes d'applications.



Depuis quelques années, toutefois, ce langage de programmation s'est hissé parmi les plus utilisés dans le domaine du développement de logiciels, de gestion d'infrastructure et d'analyse de données. Il s'agit d'un élément moteur de l'explosion du Big Data. Python est un langage interprété, qui ne nécessite donc pas d'être compilé pour fonctionner. Un programme « interpréteur » permet d'exécuter le code Python sur n'importe quel ordinateur. Ceci permet de voir rapidement les résultats d'un changement dans le code. En tant que langage de programmation de haut niveau, Python permet aux programmeurs de se focaliser sur ce qu'ils font plutôt que sur la façon dont ils le font.

Le langage Python doit sa popularité à plusieurs avantages qui profitent aussi bien aux débutants qu'aux experts. Tout d'abord, il est facile à apprendre et à utiliser. Ses caractéristiques sont peu nombreuses, ce qui permet de créer des programmes rapidement et avec peu d'efforts. De plus, sa syntaxe est conçue pour être lisible et directe.

Un autre avantage du Python est sa popularité. Ce langage fonctionne sur tous les principaux systèmes d'exploitation et plateformes informatiques. De plus, même s'il ne s'agit clairement pas du langage le plus rapide, il compense sa lenteur par sa versatilité.

Enfin, même s'il est principalement utilisé pour le *Scripting* et l'automatisation, ce langage est aussi utilisé pour créer des logiciels de qualité professionnelle. Qu'il s'agisse d'applications ou de services Web, le Python est utilisé par un grand nombre de développeurs pour créer des logiciels.

### 3.1.2 Numpy

NumPy est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux.



Plus précisément, cette bibliothèque logicielle libre et open source fournit de multiples fonctions permettant notamment de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes. NumPy est la base de SciPy, regroupement de bibliothèques Python autour du calcul scientifique.

Le module NumPy est la boîte à outils indispensable pour faire du calcul scientifique avec Python. La bibliothèque NumPy est notamment utile pour gérer les problèmes d'algèbre linéaire : les vecteurs, matrices, et plus généralement les tableaux à  $n$  dimensions ce qui est parfait pour pouvoir faire les calculs nécessaires aux traitements des frames de notre séquence vidéo.

### 3.1.3 Matplotlib

Matplotlib est une bibliothèque permettant de créer des tracés 2D de tableaux en Python . Même s'il ait pour origine l'émulation des commandes graphiques MATLAB, il est indépendant de MATLAB et peut être utilisé de manière Pythonic orientée objet. Bien que Matplotlib soit écrit principalement en Python pur, il utilise beaucoup NumPy et d'autres codes d'extension pour offrir de bonnes performances, même pour les tableaux de grande taille.



Matplotlib est conçu avec la philosophie selon laquelle on doit être capable de créer des tracés simples avec seulement quelques commandes ou une seule ! Si besoin est de voir un histogramme de données, l'instanciation des objets n'est pas nécessaire, des méthodes d'appel, des propriétés, etc. ça devrait marcher.

### 3.1.4 OpenCV

OpenCV (Open Source Computer Vision Library) est une bibliothèque de logiciels open source de vision informatique et d'apprentissage automatique. OpenCV a été conçu pour fournir une infrastructure commune aux applications de vision par ordinateur et pour accélérer l'utilisation de la perception de la machine dans les produits commerciaux. OpenCV étant un produit sous licence BSD, il est facile pour les entreprises d'utiliser et de modifier le code.



La bibliothèque contient plus de 2500 algorithmes optimisés, qui incluent un ensemble complet d'algorithmes classiques et avancés de vision assistée par ordinateur et d'apprentissage automatique. Ces algorithmes peuvent être utilisés pour détecter et reconnaître des visages, identifier des objets, classer les actions humaines dans des vidéos, suivre les mouvements d'une caméra, suivre des objets en mouvement, extraire des modèles 3D d'objets, produire des nuages de points 3D à partir de caméras stéréo, assembler des images pour obtenir une haute résolution.

L'image d'une scène entière, trouver des images similaires dans une base de données d'images, supprimer les yeux rouges des images prises au flash, suivre les mouvements des yeux, reconnaître les paysages et établir des repères pour les superposer à la réalité augmentée, etc.

### ➤ OpenCV et la détection des véhicules :

Le principal atout qu'offre OpenCV est que c'est un programme basé sur le traitement d'images, pour cela OpenCV procure un éventail d'outils pour traiter les images et faciliter la reconnaissance des véhicules, mais aussi permet de faire la liaison entre la caméra de surveillance et notre algorithme de détection

Les avantages de l'utilisation d'OpenCV pour YOLO sont :

- Intégration facile avec une application OpenCV : si l'application utilise déjà OpenCV et que le but est tout simplement utiliser YOLOv3, on a pas à se soucier de la compilation et de la construction du code supplémentaire Darknet.
- L'exécution d'OpenCV sur CPU est 9 fois plus rapide : la mise en œuvre du module DNN par OpenCV sur CPU est incroyablement rapide. Par exemple, lorsqu'il est utilisé avec Darknet demande environ 2 secondes à un processeur pour obtenir une inférence sur une seule image. En revanche, la mise en œuvre d'OpenCV ne prend que 0,22 seconde !


## Python et l'IA

Les projets IA diffèrent des projets logiciels traditionnels. Les différences résident dans la pile technologique, les compétences requises pour un projet basé sur l'IA et la nécessité de recherches approfondies. Pour mettre en œuvre des aspirations d'intelligence artificielle, on doit utiliser un langage de programmation stable, flexible et disposant d'outils. Python offre tout cela, c'est pourquoi nous avons opté pour ce langage.

Du développement au déploiement et à la maintenance, Python aide les développeurs à être productifs et confiants dans les logiciels qu'ils développent. Les avantages qui font de Python le meilleur choix pour l'apprentissage automatique et les projets basés sur l'IA incluent la simplicité et la cohérence, l'accès à de superbes bibliothèques et infrastructures pour l'IA et l'apprentissage automatique, la flexibilité, l'indépendance de la plate-forme et une large communauté.

### 3.1.5 TensorFlow

#### ➤ Qu'est-ce que TensorFlow ?

Créé par l'équipe Google Brain en 2011, sous la forme d'un système propriétaire dédié aux réseaux de neurones de Deep Learning, TensorFlow s'appelait à l'origine DistBelief. Par la suite, le code source de DistBelief a été modifié et cet outil est devenu une bibliothèque basée application. En 2015, il a été renommé TensorFlow et Google l'a rendu open source. Depuis lors,  il a subi plus de 21000 modifications par la communauté et est passé en version 1.0 en février 2017.

Pour faire simple, TensorFlow est une bibliothèque de Machine Learning, il s'agit d'une boîte à outils permettant de résoudre des problèmes mathématiques extrêmement complexes avec aisance. Elle permet aux chercheurs de développer des architectures d'apprentissage expérimental et de les transformer en logiciels.

On peut le concevoir comme un système de programmation dans lequel les calculs sont représentés sous forme de graphes. Les nœuds du graphe représentent les opérations mathématiques, et les bordures représentent des flèches de données multidimensionnelles communiquées entre elles : les tensors.

TensorFlow regroupe un grand nombre de modèles et d'algorithmes de Machine Learning et de Deep Learning. Son API front-end de développement d'applications repose sur le langage de programmation Python, tandis que l'exécution de ces applications s'effectue en C++ haute-performance.

Cette bibliothèque permet notamment d'entraîner et d'exécuter des réseaux de neurones pour la classification de chiffres écrits à la main, la reconnaissance d'image, les plongements de mots, les réseaux de neurones récurrents, les modèles sequence-to-sequence pour la traduction automatique, ou encore le traitement naturel du langage.

TensorFlow permet aux développeurs de créer des graphes de dataflow (dataflow graphs), à savoir des structures qui décrivent la façon dont les données se déplacent sur un graphe ou une série de nœuds de traitement comme le montre la figure ci-dessous.



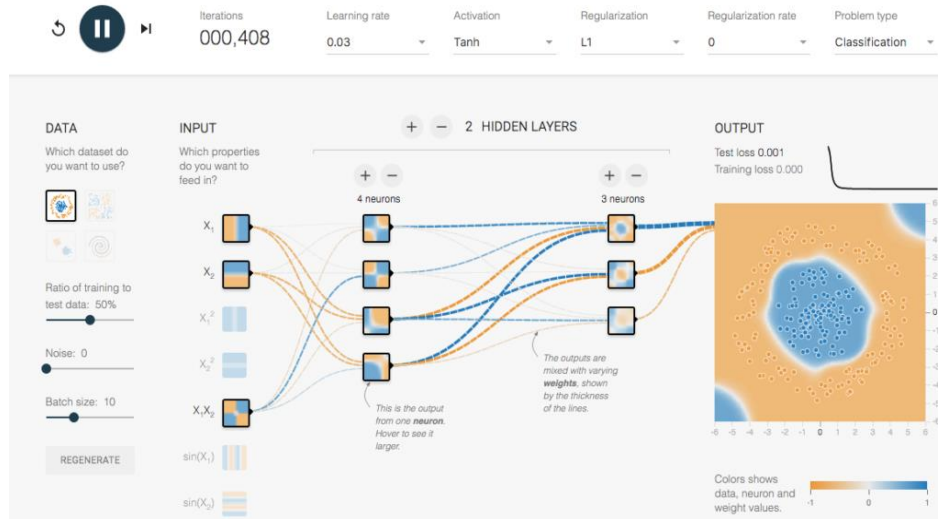


Figure 12: Exemple de réseau d'illustration de réseaux de neurones avec tensorflow

### ➤ Les avantages de Tensorflow

Les avantages de TensorFlow sont multiples. Tout d'abord, grâce à l'abstraction, ce framework facilite l'implémentation d'algorithmes et permet au développeur de se focaliser sur la logique générale d'une application.

Le mode **eager execution** permet d'évaluer et de modifier toutes les opérations de graphe de façon séparée et transparente plutôt que d'avoir à construire l'intégralité du graphe en tant qu'objet opaque unique et de tout évaluer en une fois. De même, la suite de visualisation TensorBoard permet d'inspecter la façon dont les graphes fonctionnent par le biais d'un tableau de bord interactif basé web.

En outre, grâce au soutien de Google, le déploiement et l'utilisation de TensorFlow sont facilités. En plus du TPU proposé sur Google Cloud pour accélérer les performances, le géant californien propose aussi un hub online permettant de partager les modèles créés avec TensorFlow ainsi que des versions mobiles et web du framework. Un autre avantage est son caractère open source, personnalisable, et modulaire.

Les applications TensorFlow peuvent être exécutées sur une machine locale, un cluster sur le Cloud, des appareils mobiles iOS ou Android, ou même des CPU et des GPU. Sur la Google Cloud Platform, il est possible d'exécuter TensorFlow sur la TPU (TensorFlow Processing Unit) pour profiter d'une importante accélération.

### ➤ L'API de Tensorflow

La création de modèles d'apprentissage automatique précis, capables de localiser et d'identifier plusieurs objets dans une seule image comme le montre la figure 2, reste un défi majeur en vision par ordinateur. L'API de détection d'objet TensorFlow est une infrastructure open source conçue pour faciliter la construction, la formation et le déploiement de modèles de détection d'objet. Chez Google, nous avons certainement trouvé cette base de code utile pour nos besoins en vision assistée par ordinateur.

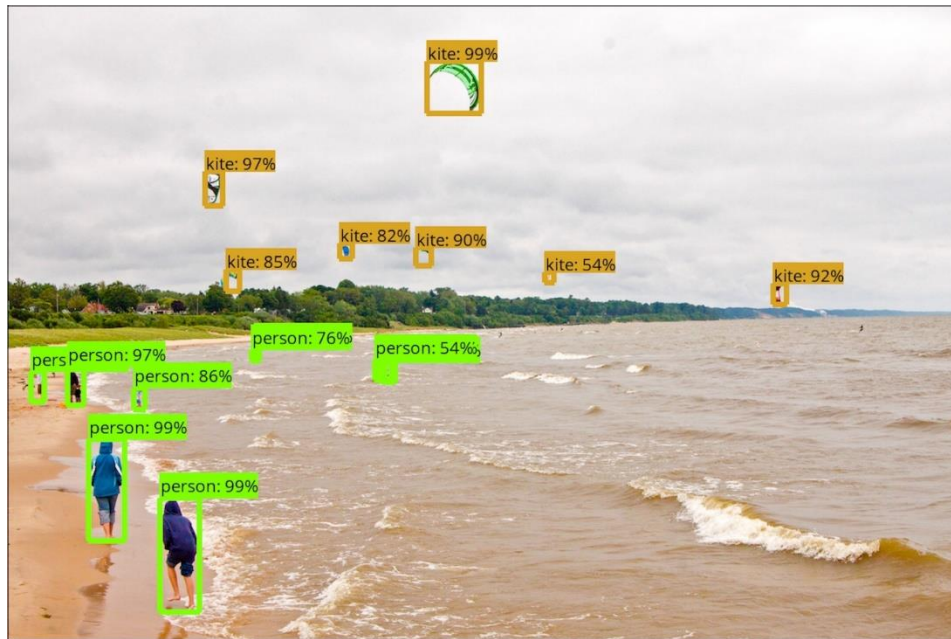


Figure 13: Exemple d'application de l'API de Tensorflow

### 3.1.6 Keras

Keras est un API de réseaux de neurones, capable de fonctionner au-dessus de tensorflow, Théano, et CNTK. Il permet une expérimentation rapide grâce à un API de haut niveau, conviviale, modulaire et extensible. Keras peut également être exécuté à la fois sur le processeur (CPU) et sur le processeur graphique (GPU). Développé et maintenu par François Chollet.



Keras a été développé pour les humains, pas pour les machines. Il met l'expérience utilisateur au centre de tout. Keras suit les meilleures pratiques afin de réduire la charge cognitive : il propose un API simple et cohérent, minimisant le nombre d'actions requises pour les cas les plus communs, et les messages d'erreur sont suffisamment explicites pour permettre une résolution simple des problèmes.

Cela rend Keras facile à apprendre et à utiliser. L'utilisation de Keras offre la possibilité de tester plus d'idées et être plus productif, ce qui permet de s'adapter à l'environnement du Machine Learning plus rapidement. Cette simplicité d'utilisation n'impacte pas la flexibilité : Keras s'intègre avec les outils de Deep Learning (en particulier Tensorflow) à bas niveau, ce

qui permet d'implémenter toutes les fonctionnalités souhaitées. Il s'intègre parfaitement avec Tensorflow par le biais d'une instruction simple (`tf.keras`). Keras est très utilisé par le monde industriel tout comme la communauté de chercheurs.

### 3.1.7 Yolo ( You Only Look Once )

YOLO utilise l'apprentissage approfondi et les réseaux de neurones convolutionnels (CNN) pour la détection d'objet. Il se distingue de ses "concurrents" car, comme son nom l'indique, il ne doit "voir" chaque image qu'une seule fois. Cela permet à YOLO d'être l'un des algorithmes de détection les plus rapides (sacrifiant naturellement une partie de la précision). Grâce à cette rapidité, YOLO peut détecter des objets en temps réel (jusqu'à 30 FPS).



Les premiers algorithmes de détection d'objets basés sur Deep Learning, tels que R-CNN et Fast R-CNN, utilisaient une méthode appelée recherche sélective pour réduire le nombre de cadres englobants que l'algorithme devait tester. Une autre approche appelée Overfeat consiste à numériser l'image à plusieurs échelles à l'aide de mécanismes glissants ressemblant à des fenêtres et fonctionnant par convolution.

Cela a été suivi par Faster R-CNN qui a utilisé un réseau de propositions régionales (RPN) pour identifier les cadres englobants devant être testés. De par leur conception intelligente, les caractéristiques extraites pour la reconnaissance des objets ont également été utilisées par le RPN pour proposer des cadres englobants potentiels, permettant ainsi d'économiser beaucoup de calculs.

YOLO, par contre, aborde le problème de la détection d'objet d'une manière complètement différente. Il ne transmet l'image qu'une seule fois par le réseau. Sa dernière version nommée YOLOv3 est considérée comme l'un des algorithmes les plus rapides qui offre une précision optimale, YOLOv3 donne plus rapidement des résultats en temps réel sur un GPU M40, TitanX ou 1080 Ti.

➤ Comment fonctionne YOLO ?

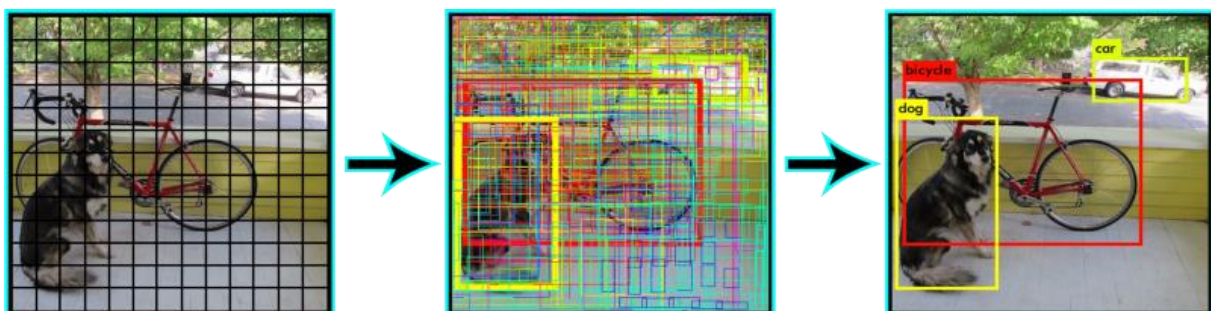


Figure 14: Etapes de traitement de l'algorithme YOLO

Pour effectuer la détection, l'image est divisée en une grille de  $S \times S$  cellules (image de gauche). Chacune des cellules prédit  $N$  «cadres» possibles et le niveau de certitude (ou probabilité) de chacune d'entre elles (image au centre), ce qui signifie que les cadres  $S \times S \times N$  sont calculées. La grande majorité de ces cadres auront une très faible probabilité, c'est pourquoi l'algorithme procède à la suppression des cadres situés en dessous d'un certain seuil de probabilité minimale. Les cadres restants sont soumis à une «suppression non-max» qui élimine les objets en double possibles et ne laisse donc que le plus exact d'entre eux (image à droite).

Les auteurs de YOLOv3, Joseph Redmon et Ali Farhadi, ont rendu YOLOv3 plus rapide et plus précis que leur précédent travail, YOLOv2. YOLOv3 gère mieux plusieurs échelles. Ils ont également amélioré le réseau en le rendant plus grand (augmentation du nombre de couches cachées) et en l'ajustant aux réseaux résiduels en ajoutant des connexions raccourcies qui permettent d'alléger le traitement effectué par le réseau.

Model	Train	Test	mAP	FLOPS	FPS
SSD300	COCO trainval	test-dev	41.2	-	46
SSD500	COCO trainval	test-dev	46.5	-	19
YOLOv2 608x608	COCO trainval	test-dev	48.1	62.94 Bn	40
Tiny YOLO	COCO trainval	test-dev	23.7	5.41 Bn	244
SSD321	COCO trainval	test-dev	45.4	-	16
DSSD321	COCO trainval	test-dev	46.1	-	12
R-FCN	COCO trainval	test-dev	51.9	-	12
SSD513	COCO trainval	test-dev	50.4	-	8
DSSD513	COCO trainval	test-dev	53.3	-	6
FPN FRCN	COCO trainval	test-dev	59.1	-	6
Retinanet-50-500	COCO trainval	test-dev	50.9	-	14
Retinanet-101-500	COCO trainval	test-dev	53.1	-	11
Retinanet-101-800	COCO trainval	test-dev	57.5	-	5
YOLOv3-320	COCO trainval	test-dev	51.5	38.97 Bn	45
YOLOv3-416	COCO trainval	test-dev	55.3	65.86 Bn	35
YOLOv3-608	COCO trainval	test-dev	57.9	140.69 Bn	20
YOLOv3-tiny	COCO trainval	test-dev	33.1	5.56 Bn	220
YOLOv3-spp	COCO trainval	test-dev	60.6	141.45 Bn	20

Figure 15: Comparaison de la performance des différents modèles actuels

La figure ci-dessus représente un tableau de comparaison entre les différents modèles réalisés jusqu'à présent dans le domaine de la détection d'objet en termes de rapidité d'exécution et de nombre d'images par seconde (FPS). Ces modèles sont formés sur le jeu de données COCO.

➤ Les avantages de YOLO dans la détection des véhicules

Contrairement aux approches basées sur les classificateurs, YOLO est formé sur une fonction de perte qui correspond directement à la performance de détection. YOLO est le détecteur d'objets polyvalent le plus rapide de la littérature. Il repousse les limites de la technologie de pointe de détection d'objet en temps réel. Ce qui fait de lui l'algorithme idéal pour les applications qui reposent sur détection de véhicules.

### 3.3 Outils d'étiquetage :

Un outil d'étiquetage est un outil qui permet de créer des annotations pour chaque image qui lui est parvenu, et qui par la suite génère un fichier sous format PascalVOC en XML soit sous format YOLO en TXT, selon le besoin du développeur, et dans le cas de notre travail, on utilise le fichier généré sous format YOLO en TXT.

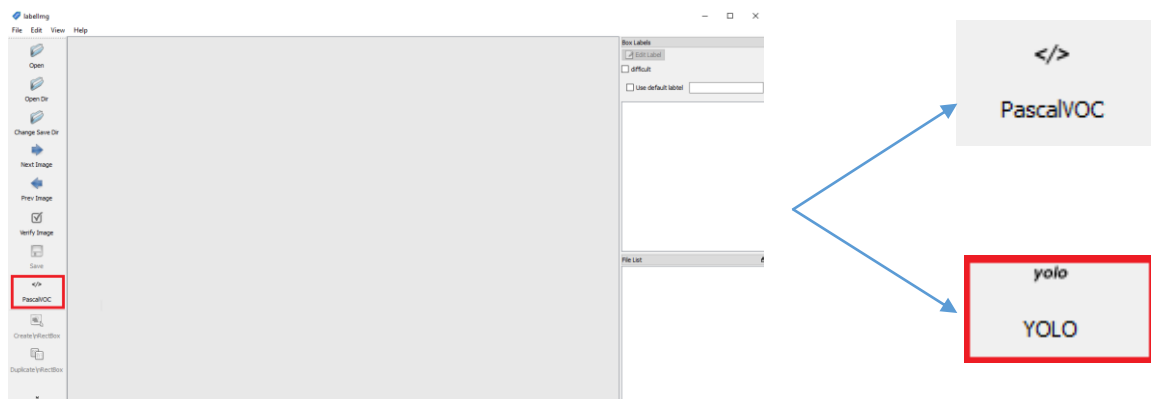


Figure 16: Outil d'étiquetage et les deux types de fichiers qu'il peut générer

### LabelImg :

LabelImg est un outil d'annotation d'image graphique .Il est écrit en Python et utilise Qt pour son interface graphique.Les annotations sont enregistrées sous forme de fichiers XML au format PASCAL VOC, format utilisé par ImageNet . En outre, il prend également en charge le format YOLO.

➤ Les fichiers générés par LabelImg :

LabelImg possède deux de génération de fichiers, un fichier sous format PascalVOC ou bien un fichier sous format YOLO.

Le fichier sous format PascalVOC est comme suit :

```
<annotation>
  <folder>Cars</folder>
  <filename>00001.jpg</filename>
  <path>C:\Users\Akram\Desktop\CNN\Cars\00001.jpg</path>
  <source>
    <database>Unknown</database>
  </source>
  <size>
    <width>600</width>
    <height>400</height>
    <depth>3</depth>
  </size>
  <segmented>0</segmented>
  <object>
    <name>car</name>
    <pose>Unspecified</pose>
    <truncated>0</truncated>
    <difficult>0</difficult>
    <bndbox>
      <xmin>41</xmin>
      <ymin>119</ymin>
      <xmax>568</xmax>
      <ymax>377</ymax>
    </bndbox>
  </object>
</annotation>
```

Figure 17: Fichier sous format YOLO généré par LabelImg

Ce format est structuré en XML, qui illustre toutes les caractéristiques concernant chaque image.

Et le fichier sous format YOLO, qui est le domaine d'intérêt de notre travail, qui est comme suit dans la figure :

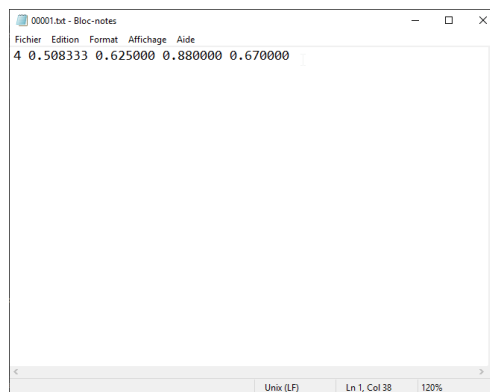


Figure 18: Fichier sous format YOLO généré par LabelImg

Ce format est simple et facile à comprendre pour l'algorithme YOLO.

### ➤ L'avantage de labelImg :

LabelImg est un outil qui rend très facile l'annotation d'images. Le choix entre les outils d'étiquetage est énorme, mais LabelImg semble être l'un des plus populaires et tout simplement parce qu'une fois dedans, c'est assez simple à utiliser. Il suffit d'ouvrir simplement un répertoire des images que nous souhaitons annoter, sélectionner un répertoire de sauvegarde et commencer à annoter.



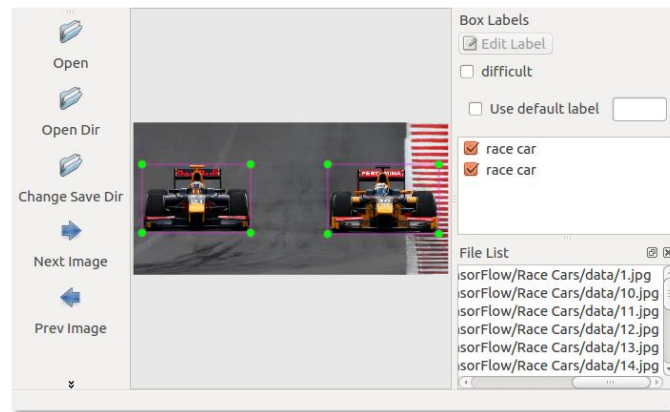


Figure 19: Exemple d'utilisation de LabelImg

### 3.4 La base de données utilisée

#### *Cars*

Pour pouvoir faire la détection des véhicules, il nous fallait une base de donnée assez remplie, le jeu de données *Cars* contient 16 185 images de 196 classes de voitures sous différent angle, ce qui fait de lui la base de données parfaite pour l'étape d'apprentissage et nous permettra de généraliser notre algorithme de manière à reconnaître le plus de modèles de voitures sur la caméra.

### 3.5 Le matériel utilisé :

Le matériel utilisé durant notre travail est : un **PC DELL XPS 13** doté d'un processeur I7-7560u cadencé à 2.40 GHZ de 7<sup>ème</sup> génération, et une ram de 16GB DDR3 ainsi qu'un disque dur SSD de 512GB, et un GPU Intel HD 620 qui est suffisamment puissant pour certaines tâches dans notre projet. Aussi un **PC LENOVO** doté d'un processeur I3-6006u cadencé à 2.00 GHZ de 6<sup>ème</sup> génération, et d'une ram de 4 GB DDR4 ainsi qu'un disque dur de 1TB, et un GPU Intel HD 520 qui a servi seulement à la programmation de l'algorithme sans l'exécution. Et aussi, pour finir on a utilisé l'espace de travail offert par Google, qui est la machine **Google Colab**, doté d'un processeur Xeon de dernière génération, et une ram de 13GB DDR4, ainsi qu'un disque dur SSD de 350GB, et d'un GPU très puissant Tesla K80 de 13GB, qui nous est très utile en termes d'exécution des tâches les plus lourdes, tel que le training des images pour le modèle, exécution de l'algorithme de détection des véhicules.

### Conclusion :

Nous avons vu durant ce chapitre, les différents outils qui nous ont facilité beaucoup de tâches de notre travail, comme l'étiquetage des images, et l'algorithme YOLO qui nous a fait gagner du temps, et surtout l'espace de travail offert par Google, Google Colab, qui nous a permis d'exécuter tâches les plus lourdes, tel que le training des images qui, en utilisant un matériel normal, prend des heures, voire plusieurs jours mais avec Google Colab, ça nous a pris quelques secondes, voire quelques minutes.

# **Chapitre 4 : Implémentation et expérience**



## Introduction

Le but de ce travail est de combiner toutes les informations et méthodes acquises dans les chapitres précédents pour réussir à mettre en place un algorithme de détection des véhicules optimal.

Ce chapitre englobe les étapes allant de la collecte de données jusqu'au résultat final en passant d'abord par le prétraitement et ensuite le traitement.

Nous allons présenter le cheminement de notre travail sous la forme d'un schéma hiérarchique.

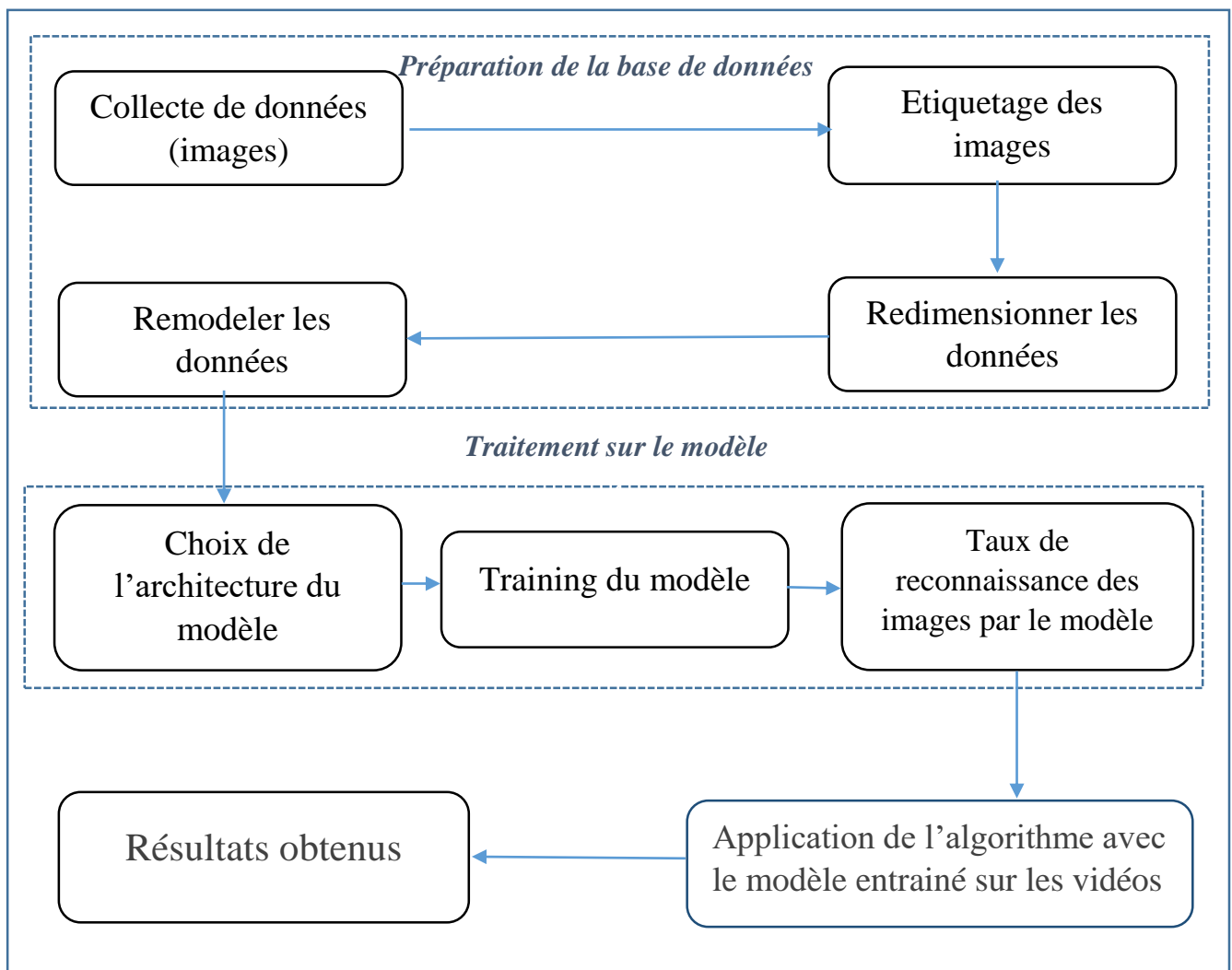


Figure 20 : schéma représentatif de l'ensemble du travail

### 4.1 Préparation de la base de données :

Cette étape consiste à préparer les données collectées en les étiquetant puis les redimensionner :

#### ➤ Collecte de données :

La base de données utilisée comporte 2359 images de voitures :

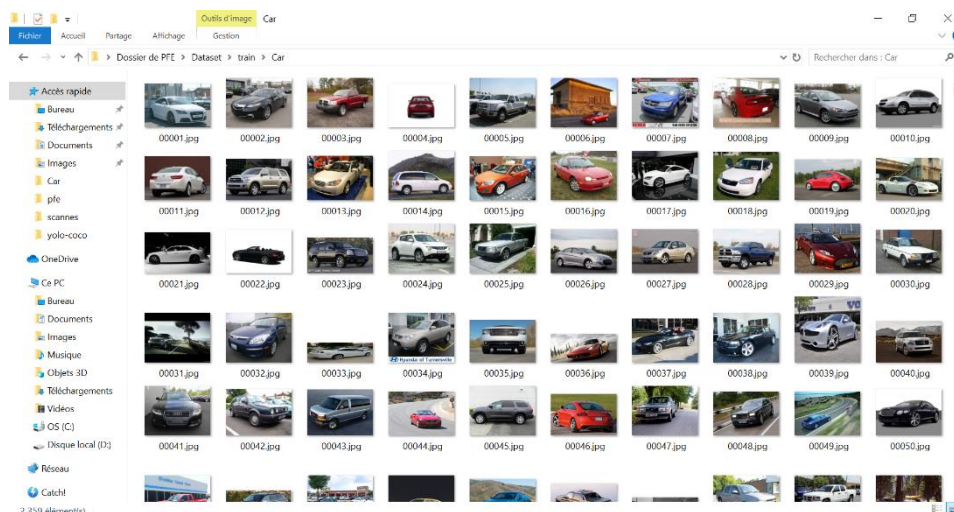


Figure 21: base de données

#### ➤ Etiquetage des images :

Chacune des images récupérées a été étiqueter manuellement avec le logiciel LabellImg :



Figure 23: : sélection du cadre représentatif d'une voiture

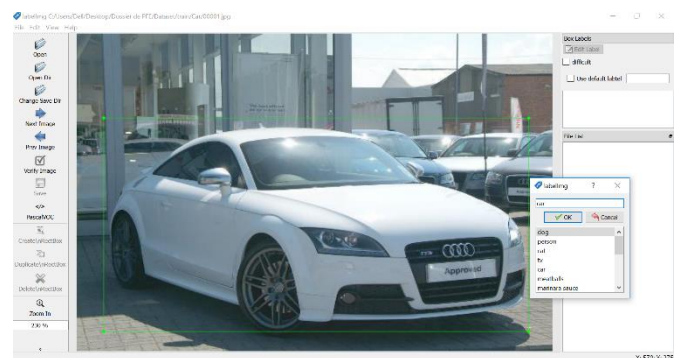


Figure 22 : choix du nom de l'étiquetage

➤ Redimensionnement des images:

Le traitement sur les images ne peut être effectué qu'après le redimensionnement des images, car il est plus facile pour le modèle de traiter les images de même dimension.



*Figure 24: changement de la résolution des images*

## 4.2 Traitement sur le modèle:

Cette étape consiste à choisir l'architecture du modèle la plus approprié à la détection des véhicules, puis faire l'apprentissage des données au modèle et pour finir, procéder à l'évaluation des performances du modèle en terme de rapidité et de taux de reconnaissance :

➤ Choix du modèle:

Le choix du modèle fut difficile car la détection des véhicules requière une architecture de modèle précise et compliquée, nous avons commencé notre projet par le développement de notre propre modèle avec une architecture simple, mais les résultats furent infructueux, lorsque nous testions le modèle sur les images, le pourcentage de reconnaissance était moyen.

Nous nous sommes ensuite orientés vers les modèles les plus avancés en matière de rapidité et de précision, pour commencer nous avons opté pour un modèle avec une précision très élevée mais lent.

Par la suite nous avons basculé vers une architecture de modèle aillant un précision moins élevée que le modèle précédent, mais meilleur en terme de temps d'exécution.

Après plusieurs recherches, nous nous sommes finalement penchés vers le modèle darknet19 qui représente le modèle le plus adéquat à notre travail, aillant un équilibre entre la précision et la rapidité de traitement.

### ➤ Training du modèle :

Le but à présent est d'utiliser la base de données préparée sur un modèle.

Nous avons, tout d'abord, compilé préalablement, avec le changement de certains de ses paramètres d'origine à commencer par l'optimiseur, dans notre cas l'optimiseur **Adam**<sup>1</sup>, la fonction de calcul d'erreurs nommée **sparse\_categorical\_crossentropy**<sup>2</sup> et pour finir la métrique qui s'orientera plus vers la précision « accuracy », puis, nous avons entraîné notre modèle sur 10 itérations.

Par la suite, nous avons abordé la méthode du **transfert learning**<sup>3</sup>, qui consiste à modifier les paramètres d'un modèle existant, puis l'entraîner sur une base de données réduite dans notre cas ce fût les modèles VGG16 et Mobilenet et pour finir sur Darknet19-53.

Le training effectué dans le cas du modèle Darknet19-53 s'est déroulé suivant des critères appropriés à la détection des véhicules dont, le nombre de classes qui, dans notre cas, est de 1 car nous ne nous sommes focalisés que sur la détection des voitures de manière générale. Le nombre d'epochs utilisé 50 epochs pour éviter le sur-apprentissage.

Taux de reconnaissance :

Après que les modèles s'étaient auto-évalués, on obtient les résultats suivants :

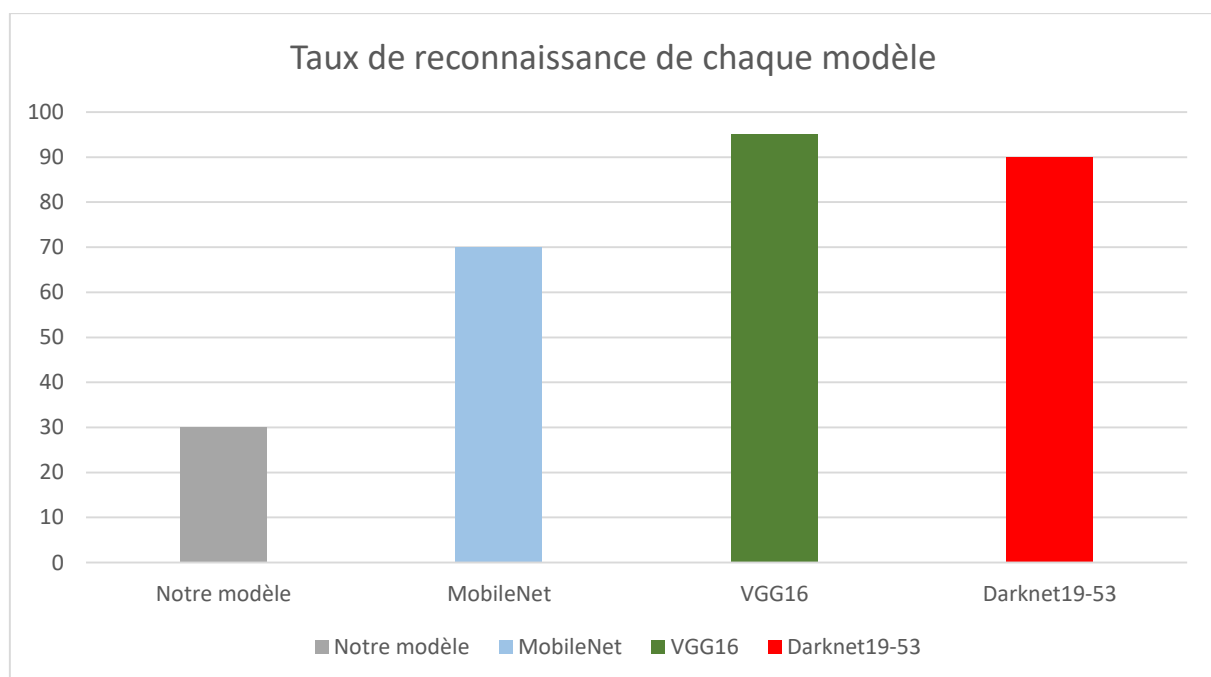


Figure 25: graphe représentatif des taux de reconnaissance

**Adam**<sup>1</sup> : est une méthode de calcul de taux d'apprentissage adaptative

**Sparse-categorical-crossentropy**<sup>2</sup> : chaque donnée ne peut appartenir qu'à une seule classe

**Transfert learning**<sup>3</sup> : est un transfert d'apprentissage d'un modèle vers un autre en changeant quelques paramètres

Test des modèles entraînés sur les images :

Sur plusieurs essais, le modèle qu'on a conçu a eu des difficultés à correctement détecter les véhicules, nous avons remarqués que sur l'ensemble des tests effectués le modèle n'arrivait globalement pas à faire la différence entre ce qui est un véhicule et ce qui ne l'était pas

### Cas de détection correcte :

Voici ci-dessous deux exemples de détection correcte avec leurs taux de prédiction :

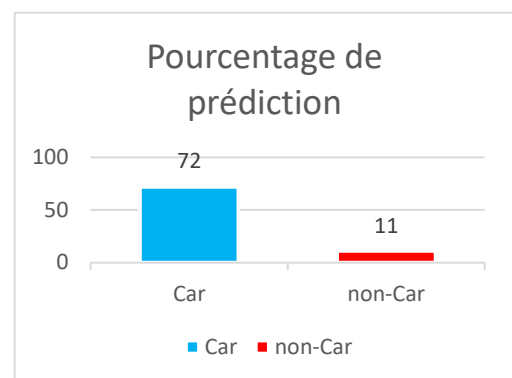


Figure 26: exemple de détection Car

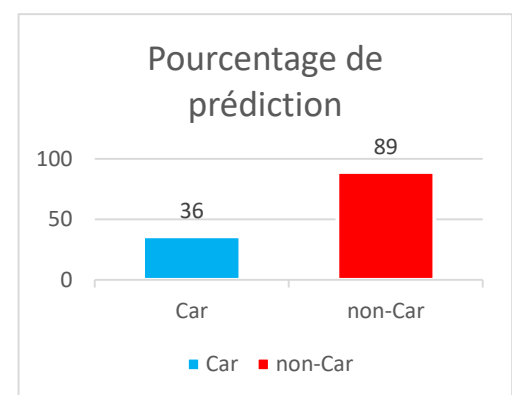


Figure 27: exemple de détection non-Car



Comme on peut le constater dans les figure précédentes «26 et 27», les modèles sont arrivés à détecter certains images correctement en tant que véhicules ou non véhicules, on observe que les pourcentages de prédiction reflètent la nature de chaque image donnée.

- **Cas de fausse détection :**

Voici ci-dessous deux exemples de fausse détection avec leurs taux de prédiction

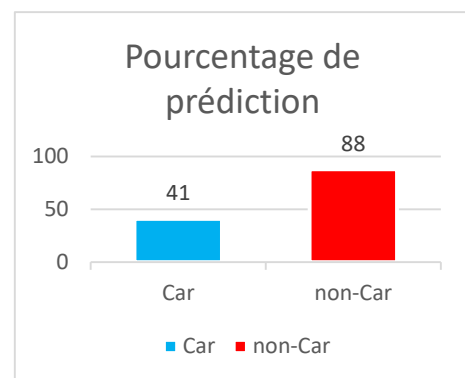


Figure 28: exemple de fausse prédiction Car

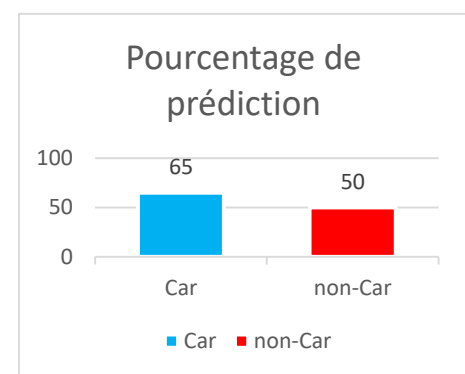


Figure 28 exemples de fausse prédiction non-Car

Nous remarquons dans les figures 29 et 30, certains des modèles peuvent aussi se confronter au problème de fausse détection, comme on peut le voir dans le tableau qui suit « voir tableau.... » avec les modèles qu'on a utilisés.

Nos tests ont été faits à partir de 20 photos de voitures :

Les modèles utilisés	Détection correcte	Détection incorrecte
Notre modèle	3	17
MobileNet	13	7
VGG16	17	3
Darknet19-53	16	4

Figure 29: test des modèles sur nos propres images

### 4.3 Essai du modèle sur l'algorithme :

Cette étape permet, après avoir choisi le modèle et l'avoir évalué, de mettre en pratique les connaissances acquises par le modèle sur des données réelles (images, vidéos) prises sur le terrain, ceci inclut 2 tâches :

#### ➤ Taux de reconnaissance des images par le modèle choisi :

Après avoir fait les prédictions concernant ce qui est véhicule et non véhicule, à ce niveau, on a testé le taux de reconnaissance des véhicules par le modèle choisi.

#### 4.3.1 YoloV2 :

Tout d'abord, on a effectué le test sur la version 2 du modèle et ça a donné les résultats suivants :

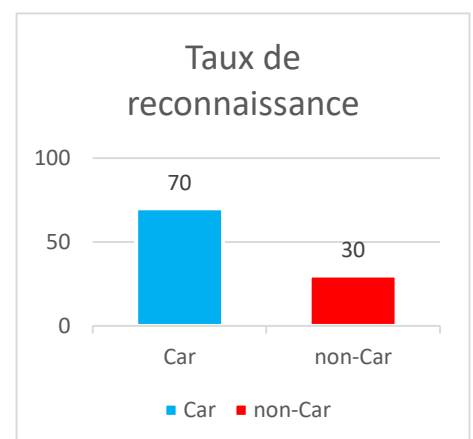
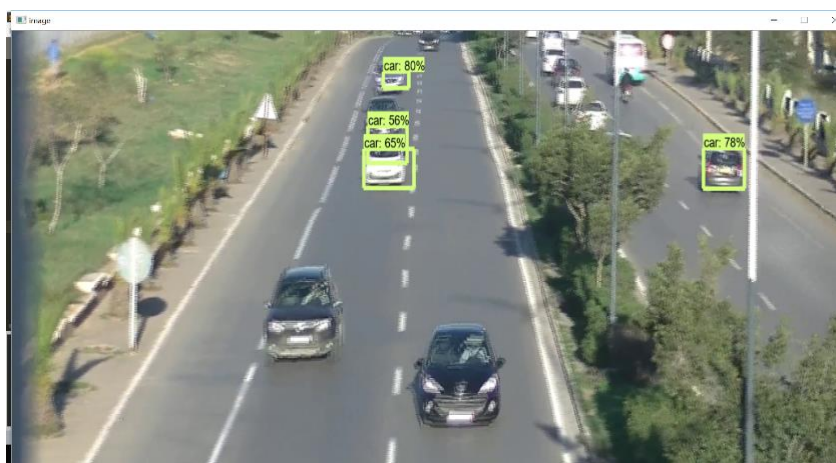


Figure 30: application de l'algorithme YOLOv2 sur nos vidéos

Comme on peut le remarquer ici [voir figure 30], l'algorithme YOLOv2 n'est pas assez précis, il arrive même à détecter quelques voitures mais le taux de reconnaissance est relativement faible par rapport à la nouvelle version YOLOv3 avec un pourcentage moyen de 70%, ce qui plutôt acceptable.

### 4.3.2 YoloV3 (Original) :

Ensuite, on a continué les tests sur la version récente du modèle, la version 3 et on a obtenu les résultats suivants :



Figure 31: application de l'algorithme YOLOv3 sur nos vidéos

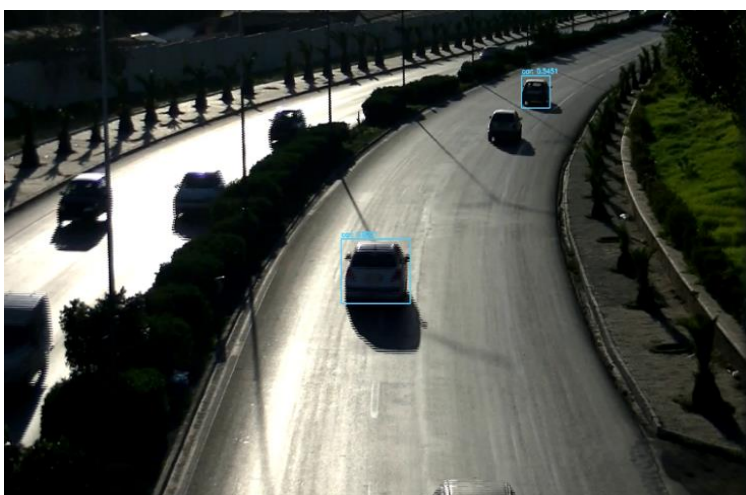
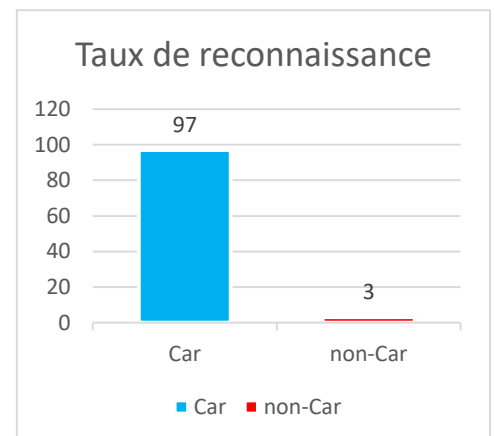
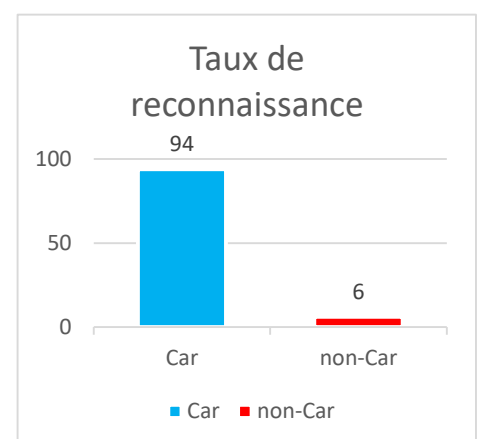


Figure 32: application de l'algorithme YOLOv3 pré-entraîné sur nos vidéos





Malgré le fait que la dernière version YOLOv3 soit plus performante que la version précédente, néanmoins elle reste assez limitée en termes de détections de véhicules comme l'illustre la figure ci-dessus [31 et 32], avec un taux de reconnaissance assez pertinent de 94%, obtenu grâce à plusieurs tests effectués.

### 4.3.3 YoloV3 (entraîné sur notre base de données) :

Ici, on prit le modèle qu'utilise l'algorithme YOLOv3 et on l'a entraîné sur notre base de données pour enfin aboutir résultats suivants :

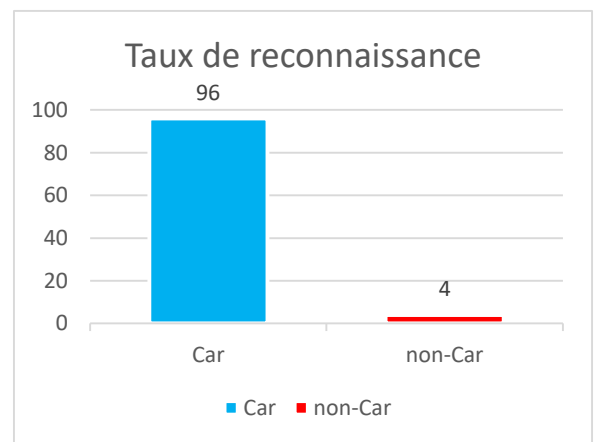


Figure 33: application de l'algorithme YOLOv3 pré-entraîné sur une autre de nos vidéos

Finalement, le dernier résultat auquel on est arrivé, est le plus précis et le plus satisfaisant [voir la figure 33], car l'algorithme final arrive à détecter 98% des véhicules qui passent, avec un taux de reconnaissance assez marquant, estimé à 96%.

Les vidéos acquises ont été prises sur les routes oranaises, nous avons fait exprès de choisir une vidéo avec un contraste non favorable à la détection pour pousser les performances de l'algorithme au maximum et pour obtenir le résultat le plus optimal possible.

• **Tableau comparatif des différents modèles qu'on a utilisé :**

Modèle	Nombre de couches cachées	Nombre d'images par secondes (i/s)	Matériel utilisé (GPU)	Taux de reconnaissance (%)
Notre modèle	16	X	Intel HD 620	30%
VGG16	15	10 i/s	Intel HD 620	71%
MOBILNET	15	20 i/s	Intel HD 620	76%
<b>DARKNET19-53 (Dédié à YOLOv3)</b>	<b>51</b>	<b>30-35 i/s</b>	<b>Tesla K80</b>	<b>96%</b>

Figure 34 : comparaison de l'application des modèles sur nos vidéos

Parmi les modèles situés ci-dessus, le plus adéquat pour la tâche de détection des véhicules est bien le modèle « Darknet19-53 », avec son architecture plutôt large et optimisé de façon à ce qu'il soit utilisé pour la détection des véhicules dans une image comme dans une vidéo en temps réel, et son taux de reconnaissance est bien plus élevé que tous les autres modèles dont on est passé par. Le GPU « Tesla K80 » est un processeur graphique très puissant, qui nous énormément aidé pour la fluidité du résultat et aussi aidé à avoir le plus d'images par secondes possibles, comparé à l'autre GPU « Intel HD 620 » qui très faible, et donc nous a donné un résultat qui n'est pas optimal mais moyen.

## Conclusion :

Nous avons vu durant ce chapitre, les différentes étapes et manœuvres qu'illustre notre travail, et qui montre les résultats finaux qu'on a obtenu après beaucoup de tests, recherches et plusieurs difficultés qu'on a rencontré durant le cheminement des étapes citées de la préparation de la base de données, à l'utilisation de l'algorithme YOLO en passant par le choix et le training du modèle qui fût particulièrement difficile due à la recherche du meilleur équilibre entre précision, rapidité.

## Perspectives :

Après avoir terminé notre travail, on peut citer quelques perspectives, qu'on peut éventuellement mettre en place à l'avenir :

- ✓ Concevoir un système complet qui sera dédié à la sécurité et surveillance routière afin de surveiller les autoroutes pour éviter les accidents et, en cas d'accidents graves, automatiquement intervenir en envoyant des ambulances sur les lieux de l'accident.
- ✓ Mettre en évidence, la gestion de la circulation afin pallier aux problèmes de bouchons.
- ✓ Procéder au tracking des voitures potentiellement suspectes en termes de conduites dangereuses pour remédier au problème de conduite délinquante.
- ✓ Créer un dispositif dédié pour les personnes aveugles afin de les assister lors de leurs parcours dans les rues, ainsi les prévenir en cas de passage sur une route, s'il y a un véhicule qui approche vers eux

## Conclusion Générale :

Suite aux recherches effectuées durant notre projet, nous sommes arrivés à la conclusion que le domaine de la détection et plus précisément la détection des véhicules est un domaine très intéressant, vaste et qui ne cesse d'accroître, durant notre travail nous avons fait en sorte de suivre ce qui se fait de mieux en matière de détection pour aboutir à des résultats satisfaisants.

Chacune des étapes de la conception de notre application est essentielles, en commençant par le prétraitement qui fût l'étape la plus délicate due à la difficulté du choix de l'architecture adéquate du modèle et de la base de données.

Nous avons commencé notre travail, tout d'abord, par la recherche d'une base de données pour la détection des véhicules, notre but dans cette étape était de trouver une base de données assez généralisée qui permettrait au modèle de reconnaître le maximum de genre de voiture présente sur nos routes, puis nous sommes passés à la création du modèle et cela par la conception de notre propre modèle, malheureusement, la détection résultant n'était pas très satisfaisante, puis nous nous sommes penchés vers les architectures les plus efficaces en terme de détection, et pour finir l'étape de prétraitement, nous avons regroupé tout cela en pré-entraînant la base de données sur l'architecture que nous avons choisis nommée darknet19.

Après obtention du modèle pré-entraîné, notre but suivant fût d'introduire ce modèle à un algorithme permettant de faire la détection des voitures à partir d'une vidéo, pour cela nous avons choisis d'incorporer notre modèle à l'algorithme YOLOv2, qui nous procurera des résultats assez prometteurs, mais durant l'année YOLOv3 a été lancé et nous nous devons de suivre le cours des événements, nous avons donc réincorporés le modèle pré-entraîné sur la version la plus récente de YOLO et nous avons obtenus au final des résultats très satisfaisants avec un taux de reconnaissance très élevée .

Après l'obtention de ses résultats, nos perspectives sont nombreuses et nous prévoyons de continuer d'améliorer notre projet à l'avenir.

# Bibliographie :

- Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)* (pp. 265-283).
- Bowcott, O. (2008). CCTV boom has failed to slash crime, say police. *The Guardian*, 6, 1.
- Bradski, G., & Kaehler, A. (2008). *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc."
- Brown, M. V. B., & Brown, A. L. (1969). *U.S. Patent No. 3,482,037*. Washington, DC: U.S. Patent and Trademark Office.
- Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248-255). Ieee.
- Felzenszwalb, P. F., McAllester, D. A., & Ramanan, D. (2008, June). A discriminatively trained, multiscale, deformable part model. In *Cvpr* (Vol. 2, No. 6, p. 7).
- Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).
- Herzog, M., & Dietmayer, K. (2019). Training a Fast Object Detector for LiDAR Range Images Using Labeled Data from Sensors with Higher Resolution. *arXiv preprint arXiv:1905.03066*.
- Hu, Q., Wang, H., Li, T., & Shen, C. (2017). Deep CNNs with spatially weighted pooling for fine-grained car recognition. *IEEE Transactions on Intelligent Transportation Systems*, 18(11), 3147-3156.
- Hu, X., Xu, X., Xiao, Y., Chen, H., He, S., Qin, J., & Heng, P. A. (2018). SINet: A scale-insensitive convolutional neural network for fast vehicle detection. *IEEE Transactions on Intelligent Transportation Systems*, 20(3), 1010-1019.
- Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in science & engineering*, 9(3), 90.
- Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. (2014, November). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia* (pp. 675-678). ACM.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).
- Li, R., & Yang, J. (2018, May). Improved YOLOv2 Object Detection Model. In *2018 6th International Conference on Multimedia Computing and Systems (ICMCS)* (pp. 1-6). IEEE.

- Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.
- Nabati, R., & Qi, H. (2019). RRPN: Radar Region Proposal Network for Object Detection in Autonomous Vehicles. *arXiv preprint arXiv:1905.00526*.
- Pizzati, F., & García, F. (2019). Enhanced free space detection in multiple lanes based on single CNN with scene identification. *arXiv preprint arXiv:1905.00941*.
- Purkait, P., Zhao, C., & Zach, C. (2017). Spp-net: Deep absolute pose regression with synthetic views. *arXiv preprint arXiv:1712.03452*.
- Rätsch, G., Onoda, T., & Müller, K. R. (2001). Soft margins for AdaBoost. *Machine learning*, 42(3), 287-320.
- Razakarivony, S., & Jurie, F. (2015). Vehicle detection in aerial imagery (vedai): a benchmark. *Tech. Rep.*
- Redmon, J., & Farhadi, A. (2018). YoloV3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*(pp. 91-99).
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*(pp. 91-99).
- Satar, B., & Dirik, A. E. (2018, October). Deep Learning Based Vehicle Make-Model Classification. In *International Conference on Artificial Neural Networks* (pp. 544-553). Springer, Cham.
- Soltani, A., Boukens, A., & Chebout, S. (2018). Un System de détection des piétons.
- Suykens, J. A., & Vandewalle, J. (1999). Least squares support vector machine classifiers. *Neural processing letters*, 9(3), 293-300.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).
- Van Der Walt, S., Colbert, S. C., & Varoquaux, G. (2011). The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2), 22.
- Wang, L., Lu, Y., Wang, H., Zheng, Y., Ye, H., & Xue, X. (2017, July). Evolving boxes for fast vehicle detection. In *2017 IEEE International Conference on Multimedia and Expo (ICME)* (pp. 1135-1140). IEEE.

Zhang, J., Ramanagopal, M. S., Vasudevan, R., & Johnson-Roberson, M. (2019). LiStereo: Generate Dense Depth Maps from LIDAR and Stereo Imagery. *arXiv preprint arXiv:1905.02744*.

Zhong, J., Lei, T., & Yao, G. (2017). Robust vehicle detection in aerial images based on cascaded convolutional neural networks. *Sensors*, 17(12), 2720.