REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE YAHIA FARES DE MEDEA

Faculté de Technologie
Département de Génie Electrique



Projet de fin d'études de Master

Filière : Electronique

Spécialité : Instrumentation

Réalisation d'un système intelligent pour la collecte des données au niveau d'une Intersection

---

Préparé et présenté par,

Mr. Diaoui Faycel El Tahar.

Proposé et dirigé par,

Dr. BOUHEDDA Mounir

Dr. BENYEZZA Hamza

Année universitaire 2020-2021

Remerciements

En premier lieu, je veux remercier le Bon Dieu de m'avoir donné la force et le courage pour accomplir ce travail et qui m'a procuré ce succès.

Je tiens à remercier en premier lieu ***Dr. BENYEZZA hamza*** *et je le félicite pour l'obtention du diplôme de Doctorat et* ***Dr. BOUHEDDA Mounir*** *pour leurs encadrement et encouragement et pour l'intérêt apporté pour l'accomplissement de ce travail*

J'adresse aussi mes très sincères remerciements à ***Dr. BACHEN Mourad*** *de m'avoir aidé à la réalisation de ce mémoire.*

Un grand merci à mes enseignants, mes amis, ainsi que mes collègues, un chaleureux merci pour ***M. GUEDMANI Mohamed***, ***Dr. HENTABLI Hamza, Pr. GUEMANA Mouloud*** *et* ***Pr. ABUDURA Salam****pour leurs encouragements et pour leurs aides.*

Dédicaces

Je dédie cette mémoire à :

Mes très chers parents pour les sacrifices et les encouragements qu'ils n'ont cessé de me conférer. Que dieu les garder éternellement heureux.

Toutes Mes frères.

Toute la famille Ferkous chacun de son nom

Toutes Mes collègues d'université de Médéa.et le groupe M2 Instrumentation en spéciale

Nos enseignants du département génie électrique.

Je me souviens du Dr.BENREKIA Fayçal Que Dieu le garde dans son vaste paradis.

ملخص

الهدف من مشروع التخرج هو تطوير نظام ذكي لجمع المعلومات الطرق المرورية. من خلال استخدام الكاميرا وخوارزمية YOLO. يمكن للنظام المقترح اكتشاف وتصنيف وحساب عدد الاجسام في التقاطع. يسمح استخدام البيانات التي تم جمعها لاحقًا بتحليل والمعالجة وإدارة التقاطع. تظهر نتائج المحاكاة نتائج مرضية للغاية ، حيث تمكنا من حساب عدد مستخدمي الطريق بطريقة دقيقة ، وتحديد أنواع المستخدمين (الحافلات ، والدراجات النارية ، والشاحنات ، وما إلى ذلك) ، وحساب التدفق حركة المرور على الطرق وتسجيلها في الفيديو المعالج.

الكلمات المفتاحية:
YOLO، جمع معلومات الطريق ، تدفق حركة المرور ، تصنيف الأشياء

Résumé

L’objectif de ce projet de fin d’études est le développement d’un système intelligent pour la collection des informations routière. Grâce à l’utilisation des caméras et l’algorithme YOLO, le système proposé peut détecter, classer et calculer le nombre des objets au niveau d’une intersection. Les données collectées peuvent être exploitées ultérieurement pour faire des analyses, traitement et gestion d'une l’intersection. Les résultats de simulation montrent des résultats très satisfaisants, où nous avons pu calculer le nombre d’usagers de la route d’une manière précise, identifié les types d’usagers (car, moto, camion, etc.), calculé le débit lié à lacirculation routière et enregistré la vidéo traitée.

Les mots clé : collection des informations routières, YOLO, débit de circulations, classification des objets.

Summary

The aim of this project is to develop an intelligent system for collecting traffic information. Through the use of cameras and the YOLO algorithm, the proposed system can detect, classify and calculate the number of objects at an intersection. The collected data can be used later for analysis, processing and to manage the intersection. The simulation results show very satisfactory results, where we were able to calculate the number of road users in a precise way, identified the types of users (bus, motorcycle, truck, etc.), calculated the relatedflow. to road traffic and recorded the processed video.

Key words: collection of road information, YOLO, traffic flow, classification of objects.

Table des matières

Introduction générale……...…….…….....1
Chapitre I : Etat de l'art 3
Etat de l'Art 3
I.1 Introduction 4
I.2 Feux De Circulation intelligent 4
I.2.1 Définition 4
I.2.2 Différents types de feux de circulation intelligente au monde..... 4
I.2.3 Avantages et inconvénients du feu de circulation intelligent..... 9
I.3 Systèmes de collection des données..... 9
I.3.1 Système ATLYST à l'Australie (Covid 2021)..... 10
I.3.2 TMA-122 11
I.3.3 Collecte de données de circulation avec KiTraffic Statistiques..... 13
I.3.4 Feux de circulation ont bas radar 14
I.3.5 Détecteur MC2224..... 15
I.3.6 Détecteur MC3224..... 16
I.3.7 Détecteur de trafic TDC1 17
I.3.8 GECKO TRAFIC / CLASSE..... 19
I.4 Conclusion..... 21
CHAPITRE II : Généralité sur Deep Learning, modèles pré-entrainés et YOLO25
Généralités sur Deep Learning, modèles pré-entrainés et YOLO..... 25
II.1 Introduction 26
II.2 Apprentissage profond (Deep Learning) 26
II.2.1 Définition..... 26
II.2.2 Utilisation du Deep Learning 27
II.2.3 Réseaux Neurones Convolutionnels (CNN)..... 28
II.2.3.1 Définition..... 28

II.2.3.2 Description des CNNs 29
II.3 Modèles pré entrainé 31
II.3.1 Introduction 31
II.3.2 Definition 32
II.3.3 Comment utiliser les modèles pré-entraînés? 32
II.3.4 Façons d'affiner un modèle 33
II.3.5 Présentation des models pré-entraînés 34
II.3.5.1 Détection d'objets 34
II.3.5.2 Reconnaissance faciale et régénération 35
II.3.5.3 Segmentation 35
II.3.5.4 Divers 36
II.4 Algorithme YOLO (You Only Look Once) 37
II.4.1 Introduction 37
II.4.2 Architecture 39
II.4.3 Apprentissage 40
II.4.4 Detection 40
II.4.5 Avantages et inconvinients de YOLO 44
II.4.6 Implementation de l'algorithme YOLO 45
II.4.7 Exemples de travail par YOLO sur Darknet 46
II.5 Conclusion 52
Chapitre III : Implimentation 53
Implémentation 53
III.1 Introduction 54
III.2 Cahier des charges et système proposé 54
III.3 Organigramme de fonctionnement 56
III.4 Implémentation software 56
III.4.1 Environnement de travail 56

❖ Hardware utilisé 56

III.4.2 Programmation 61

III.4.2.1 Vidéo utilisée 61

III.4.2.2 Initialisation 61

III.4.2.3 Initialisation de l'analyse et le traitement 65

III.4.2.4 Application des fonctions de comptage, de traitement et l'affichage des information extrait 69

III.4.3 Discussions 78

III.4.3.1 Pour quoi YOLO 78

III.4.3.2 Comparaison avec d'autres systèmes de collection des données ... 79

III.5 Conclusion 80

Conclusion générale……..……82

Bibiographie……...……..……...83

Liste des Figures

Figure I- 1: feu de circulation intelligent en chine 6

Figure I- 2: Feux de circulation intelligents au Texas (Etat Unis). 7

Figure I- 3 : feu de circulation lié par des ondes Wi-Fi..... 8

Figure I- 4 : Données de trafic en temps réel à South Perth..... 10

Figure I- 5 : l'intersection de Mends Street et Mill Point Road à l'aide d'ATLYST .11

Figure I- 6 : le détecteur TMA 122. 12

Figure I- 7 : architecture d'implémentation de TMA-122 13

Figure I- 8 : Détecteur MC2224..... 15

Figure I- 9 : Détecteur MC3224..... 16

Figure I- 10 : Détecteur De Trafic TDC1 17

Figure I- 11 : Champ De Vision 19

Figure I- 12 : GECKO TRAFIC / CLASSE 19

Figure I- 13: GECKO TRAFIC / CLASSE SWARCO-WEB..... 21

Figure II- 1 : Déférence entre Deep-Learning et réseaux de neurone simple..... 25

Figure II- 2: Architecture d'un réseau CNN..... 27

Figure II- 3 : schéma du parcours de la fenêtre de filtre sur l'image..... 28

Figure II- 4: exemple d'une convolution 29

Figure II- 5: reconstruction de visage en 3d 33

Figure II- 6 : semantic image segmentation..... 34

Figure II- 7 : chronologique des méthodes de détection d'objets. 34

Figure II- 8 : illustration de la structure de la méthode YOLO 36

Figure II- 9: Architecteur générale de l'algorithme YOLO. 37

Figure II- 10 : Image YOLO (divisée en grille S*S) 38

Figure II- 11: YOLO boîte de délimitation à grille unique. 39

Figure II- 12 : Carte de probabilité conditionnelle YOLO 39

Figure II- 13 : Combinaison de boîtes de délimitation YOLO 40

Figure II- 14 : Carte des fonctions de sortie YOLO. 41

Figure II- 15 : Résultat du test YOLO. 41

Figure II- 16 : commande d'exécuté pour une seule image..... 44

Figure II- 17 : la sortie de programme de premier essai..... 44

Figure II- 18: l'image obtenu en le premier essai avec les détections. 45

Figure II- 19: la commande pour traiter des images multiples. 45

Figure II- 20: image expérimentale des classes type Person et Moto..... 46

Figure II- 21: image expérimentale des classes type oiseau et Ball 46

Figure II- 22: image expérimentale des classes type Person et Moto..... 46

Figure II- 23: image expérimentale de class type Person. 46

Figure II- 24: image expérimentale de class type voiture..... 47

Figure II- 25: image expérimentale des classes type Person et cheval 47

Figure II- 26: l'image obtenu contient des détections par l'algorithme des classes Moto et Person. 48

Figure II- 27: l'image obtenu contient de détection par l'algorithme d'une la classe Person. 48

Figure II- 28: l'image obtenu contient des détections par l'algorithme des classes oiseau et Ball. 48

Figure II- 29: l'image obtenu contient des détections par l'algorithme des classes Person et Moto. 48

Figure II- 30: l'image obtenu contient de détection par l'algorithme d'une la classe voiture..... 49

Figure II- 31: l'image obtenu contient des détections par l'algorithme des classes Person et cheval 49

Figure II- 32: la certitude des prédictions de l'algorithme 49

Figure III- 1: Architecture interne du système proposé. 55

Figure III- 2: Architecture générale du système proposé. 55

Figure III- 3: Organigramme de fonctionnement 57

Figure III- 4: Caractéristiques de la Vidéo. 61

Figure III- 5: Capture de la vidéo utilisée. 61

Figure III- 6: Les Commande de L'importation des bibliothèques. 62

Figure III- 7 : Les Commandes d'argumentation. 62

Figure III- 8: Emplacement D'Argument. 63

Figure III- 9: Exemple De Déclaration De Notre Argument. 63

Figure III- 10 : Block de Pointage et la lecture de la Vidéo. 63

Figure III- 11: Block de pré-détection. 64

Figure III- 12 : le vidéo résultant sans aucun traitement 64

Figure III- 13: chargement des classes de model COCO. 65

Figure III- 14: Boucle d'extraction des Images (Frames). 65

Figure III- 15: Arrêt d'attraction des Images et le dimensionnement. 65

Figure III- 16: Initialisation des cadres pour les objets Classés 65

Figure III- 17 : Filtrage Et Extraction Des Cordonnées Des Identifiant. 66

Figure III- 18: La Commande De Suppression Non Maximale. 66

Figure III- 19 : Cadrage Des Identifient. 67

Figure III- 20:cadrages des véhicules 68

Figure III- 21: affichage de classement des véhicules 68

Figure III- 22: un programme pour l'essai 69

Figure III- 23 : expérience des fonction cv2[format]. 69

Figure III- 24 : déclaration des constants des point référence. 70

Figure III- 25: la déclaration des commandes cv2line 70

Figure III- 26 : capture sur le vidéo résultant contient les lignes imprimés 70

Figure III- 27: fonction de calculs des nombres des frames..... 71

Figure III- 28 : une fonction pour estimer le duré nécessaire a complété le traitement71

Figure III- 29 : la compilation avec le nombre des frames..... 71

Figure III- 30: affichage des résultats au niveau du terminal linux. 71

Figure III- 31 : la fonction du comptage des véhicules de chaque classe 72

Figure III- 32 : affichage des résultats de comptage 72

Figure III- 33 : capture sur le vidéo résultant contient le nombre des véhicules de la route. 73

Figure III- 34 : une capture sur la compilation de notre programme sur terminal 73

Figure III- 35 : organigramme de calcul le débit 74

Figure III- 36: comptage des objets entrant et sortant de la zone surveillée 75

Figure III- 37 : fonction de calcul le débit. 75

Figure III- 38 : fonction d'incrémentation les Constans. 76

Figure III- 39 : la première expérience de calcul de débit. 76

Figure III- 40: fonction traqueuse..... 76

Figure III- 41:une des résultats finals de notre system par la Fonction traqueuse. ... 77

Figure III- 42 : les résultats de la fonction traqueuse. 77

Figure III- 43 : cibler une seule route 78

Liste des tableaux

Tableau III - 1 : comparaissent des performances a d’autres systèmes 80

Introduction Générale

Les embouteillages sont devenus très fréquents dans la société actuelle, ce qui amène plusieurs agences à adopter de nouveaux projets de gestion et de collection des donnés routières pour éradiquer ces problèmes de Traffic routier. En Allemagne. Ils ont créé un système communiqué avec la ville ! Plus exactement avec les feux rouges. Le système testé à Ingolstadt, fief d'Audi en Allemagne, vise à améliorer la fluidité du trafic en évitant ou en réduisant les arrêts inutiles. Donc la solution c'est système Un logiciel algorithmique contrôle les feux de circulation de la ville allemande. Les véhicules et le système de gestion des feux sont liés par ondes Wi-Fi et UMTS (3G). Les Canadian aussi inventé deux system un de bassedes fibres optiques qui permettra de gérer à distance les caméras vidéo et les feux de circulationet un autre à Boucherville. La Ville s'est dotée d'un système qui permet qu'un feu de circulation passe au vert à l'approche d'un camion en urgence.et nombreux d'exemple dans quelque pays Malheureusement. Malgré toutes ces tentatives, ces systèmes restent lourds et peu efficaces. C'est ce qui nous a amenés à chercher une solution parfaite et simple pour surmonter ce problème.

La reconnaissance et la classification des objets par la vision artificielle est un thème de recherche qui a été abordé depuis plusieurs années, grâce aux récentes avancées technologiques dans le domaine du développement des systèmes calculateurs et les microprocesseurs. Ce thème est redevenu très utilisé par les chercheurs et scientifiques modernes, cette technologie permet de recueillir des informations précieuse sur le Traffic ainsique de réduire le travail manuel fastidieux et long ,et qui continue de susciter les travaux de la communauté scientifique jusqu'à ce jour , tout en renouvelant de nouvelles méthodes et techniques de travail pour objectif de rendre ces systèmes meilleur et encore plus sophistiqués tous les systèmes travaillant sur ce problème ont quelques inconvénient majeur. La cherté, l'implémentation et l'adaptation du système et le rendement de ce dernier, ainsi que la rapiditédu système en cause.[1][2]

Un système de reconnaissance et classification des objets typique est divisé en quatrephases comme indiqué ci-dessous (Figure 1) :

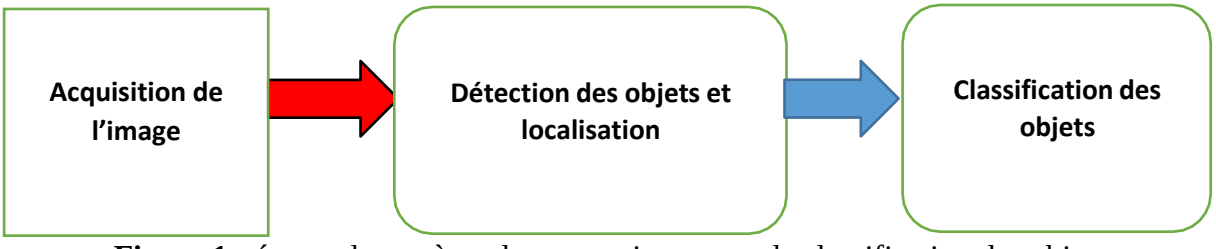


Figure 1 : étapes du système de reconnaissance et de classification des objets.

1) **Acquisition de l'image à** partir d'une séquence vidéo et son envoi vers le système**.**
2) **Détection et localisation des objets :** c'est l'opération la plus importante et la plus difficile, elle détermine la rapidité du système. Plusieurs études et recherches sont consacrées à cette phase depuis plusieurs années et cela donne naissance à différentes méthodes proposées. On peut citer dans ce cas les caractéristiques d'une image numérique comme la résolution, la taille de l'image, les couleurs, les contours et l'emplacement des objets.
3) **Classification des objets :** une fois les objets détecté et localisé cette phase permet de classifier et identifier ces derniers à travers des techniques récentes.

Notre objectif consiste à proposer un système intelligent qui permet de détecter, classer, et calculer le nombre des objets au niveau d'une intersection par classe (moto, car, camion, etc.), et aussi respecter les contraintes suivantes : la précision, le temps d'exécution, et le coût d'installation. Les données collectées exploiter pour faire des analyses, statistiques et surtout pour prise la bonne décision pour contrôler les feux de circulation au niveau de l'intersection.

Le présent manuscrit est devisé en trois chapitres :

- **Le 1Er Chapitre :** présente les feux de circulation moderne et intelligent tout en énumérant quelques systèmes intelligents de gestion routière et collection des données.

- **Le 2EMe Chapitre :** présente des concepts fondamentaux de l'intelligence artificielle, et le Deep Learning, les réseaux de neurones convolutifs, ensuite on a donné une idée sur les modèles des algorithmes pré entrainé et on s'est focalisé sur l'algorithme YOLO.

- **Le 3EMChapitre :** décrit en détail l'implémentation de l'algorithme YOLO et la conception de notre système pour atteindre notre objectif de collection des données routière utilisent langage python 3 et les bibliothèques OpenCv et keras.

- Finalement on va présenter les résultats obtenus avec une démonstration détaillé et comparée notre système a des autres pour une évaluation.

Chapitre I :

Etat de l'Art

I.1 Introduction

Les transports routiers sont essentiels au développement. En facilitant la circulation des personnes et des marchandises, ils améliorent l'accès à l'éducation, aux soins de santé, à l'emploi et aux marchés économiques.

La congestion urbaine est un problème de plus en plus présent dans de nombreuses villes à travers le monde. Ce phénomène a des effets sur les déplacements quotidiens des citadins et a de nombreux impacts négatifs en termes de qualité de vie : durée des déplacements allongée, pollution atmosphérique, pollution visuelle et sonore, coût financier, etc.

Les informations sur le volume et la charge de circulation jouent un rôle important dans la planification des infrastructures à long terme, gestion de trafic, les améliorations pour la sécurité de circulation et l'optimisation de l'entretien des routes.

Dans ce chapitre, on va présenter dans la première partie quelques systèmes intelligents de gestion de feux de circulation à travers le monde. Dans la deuxième partie, on va présenter quelques systèmes de gestion de feux de circulation à travers le monde.

I.2 Feux De Circulation intelligent

I.2.1 Définition

L'intelligence est la capacité d'analyser, de planifier, et de résoudre des problèmes à une situation, et choisir des moyens d'action en fonction des circonstances, et de tirer des conclusions et d'agir rapidement. Un feu de circulation intelligent permet de prendre des décisions intelligentes de régulation du trafic routier entre les usagers de la route, les véhicules et les piétons à partir des informations (nombre et débit de véhicules) issues de plusieurs sources (caméras, capteurs, GPS, etc.).[1]

I.2.2 Différents types de feux de circulation intelligente au monde

Certains modèles de gestion des feux de circulation ont été commercialisés. Ceux-ci utilisent soit des capteurs qui déterminent la position de véhicules en temps réel, soit une méthode qui permet de simuler le déplacement des véhicules ou bien les deux. Dans cette section, on va présenter quelques exemples récents :

1) Feux de circulation intelligents à Montréal (Canada)

Les feux Les feux de circulation de Montréal, qui ont parfois l'air d'avoir été volontairement désynchronisés, sont plutôt réglés au quart de tour pour favoriser la fluidité du trafic, comme l'a découvert Jean-Sébastien Cloutier en discutant avec Daniel Beaulieu, ingénieur en transport et circulation à la Ville de Montréal.

Optimisation des feux de circulation afin de sécuriser et améliorer la mobilité de tous les usagers dans les villes. Le comité exécutif de Montréal a approuvé la mise en place d'un réseau principal de fibres optiques qui permettra de gérer à distance les caméras vidéo et les feux de circulation [2].

2) Feux de circulation intelligents à Boucherville(canada)

Les pompiers pourront circuler de façon plus sécuritaire sur les artères principales de Boucherville. La Ville s'est dotée d'un système qui permet qu'un feu de circulation passe au vert à l'approche d'un camion en urgence.

Ce système améliore la sécurité de tout le monde en plus d'accélérer le temps de réponse des pompiers fonctionne par signal infrarouge ou GPS, qui est envoyé du camion de pompier vers un détecteur installé sur le feu de circulation. La technologie fait passer le feu du rouge au vert lorsqu'un camion de pompier avec les gyrophares allumés arrive tout près. La circulation est alors stoppée dans l'autre sens et permet aux véhicules devant le camion de pompier de circuler et de libérer la route plus facilement.

Le Service de sécurité incendie a demandé aux municipalités de son territoire d'inclure ce système dans leurs futurs travaux de réfection d'intersection. Une vingtaine de villes au Québec utilisent ce système. Il est notamment utilisé à Montréal et Longueuil pour la flotte d'autobus [2].

3) Feux de circulation intelligents en Chine

En Chine, il existe de nombreux types de feux tricolores. Celui-ci regroupe sur un même groupe trois ensembles de feux différents comme indique sur la figure I.1 (pour les automobilistes désirant aller tout droit, à gauche, et pour les cyclistes allant à gauche). Ce type de feux tricolores est placé après le carrefour. Chacune des lampes de ce feu change de couleur de façon indépendante. Les lumières soi-disant trafic plein intelligent, en fait, un feu de signalisation communs, plus un affichage électronique, le contenu de l'écran peut varier en fonction de la situation du trafic dans les options du programme préétablis, tels

que la nécessité de contrôler l'intersection de la circulation, tourner à droite, l'écran invites affichera "en face de l'intersection pour permettre seul bus grand tourisme, taxi, tourner à droite." A cette époque, la voie dans les mêmes cartes de direction sera également affichée simultanément. "En plus de l'autobus de passagers grande, taxi, tourner à droite à interdire" l'écran de message texte invite à deux participants sur les routes suggèrent que tandis que les lumières sont synchronisées en conséquence l'ajustement. Par exemple, dans la période la fête nationale, le conducteur a conduit dans l'intersection, nous pouvons voir le mot « interdire les véhicules à moteur droite" ou "En plus du taxi de bus de bus externe interdite virage à gauche" signes sur la route, nous pouvons suivre les instructions sur le panneau d'affichage conduite [2].



Figure I- 1: feu de circulation intelligent en chine

4) Feux de circulation intelligents au Texas (Etat Unis)

Siemens Siemens trafic Intelligent Solutions (Figure I- 2) développe une technologie de transport intelligent pour l'évacuation rapide et ordonnée des citoyens. Dans ce projet, les systèmes de chronométrage de feu n'estiment que le nombre de véhicules en enregistrant les signaux.

Bluetooth émis par les téléphones cellulaires des conducteurs et ajuster les phases de la lumière rouge et vert en conséquence. Le logiciel associé est fourni par l'unité de recherche mondiale de Siemens.Les adresses anonymes des appareils Bluetooth à proximité sont détectées et transmis de chaque intersection au centre de gestion du trafic, qui calcule le temps de voyage entre les intersections.

Les données de densité de trafic collectées par les systèmes sont également incorporées dans les cartes routières numériques, les pilotes peuvent télécharger sur leurs smartphones ou des systèmes de navigation afin de trouver les voies les plus rapides hors de la ville. En cas d'urgence, le système peut contrôler de manière centralisée tous les feux de circulation.

Avec Bluetooth, les installations comprennent GPS et de communication à courte portée (DSRC) antennes dédiées. Véhicules d'urgence à bord l'équipement envoie la localisation GPS via DSRC à un logiciel dans le contrôleur indiquant que le véhicule approche devrait être une priorité. L'application est capable de lire direction d'approche et la vitesse permettant la synchronisation du signal d'initier un vert indépendamment de la vitesse en sens inverse du véhicule d'urgence approche. Cela signifie que la priorité peut être appliquée spécifiquement à la route à suivre, que ce soit tout droit à l'intersection ou tourner à gauche [2].



Figure I- 2: Feux de circulation intelligents au Texas (Etat Unis).

5) Feux de circulation intelligents en Bruxelles (Belgique)

La Région bruxelloise investit dans une centrale de gestion des feux de circulation. Les feux rouges pourront bientôt être commandés à distance par des êtres intelligents en fonction de la densité du trafic. Prolonger la durée du feu vert pour fluidifier le trafic durant les heures de pointe, voilà une des possibilités de ces feux dit intelligents. En fonction de la densité du trafic, des accidents ou des chantiers, ils s'adapteront. Grace

aux caméras de surveillance déjà installées aux principaux carrefours de la capitale, les experts pourront intervenir en temps réel sur les feux de signalisation [2].

6) Des feux de circulation intelligents en Allemagne

Problème de feux Problème de feux de circulation en Allemagne c'est : Quoi de plus pénible en ville que de devoir sans cesse s'arrêter au feu rouge, puis redémarrer pour s'arrêter à nouveau quelques dizaines de mètres plus loin ?

L'objectif est de communiquer avec la ville ! Plus exactement avec les feux rouges. Testé à Ingolstadt, fief d'Audi en Allemagne, le système vise à optimiser le flux de circulation en évitant ou en réduisant les temps d'arrêts inutiles. Donc la solution pour faire un feu de circulation intelligent c'est système présenter par la figure I-3, Un logiciel algorithmique contrôle les feux de circulation de la ville allemande. Les véhicules et le système de gestion des feux sont liés par ondes Wi-Fi et UMTS (3G).Actuellement une vingtaine de voitures et près de 50 feux sont équipés de cette technologie expérimentale. Concrètement la voiture identifie en temps réelle l'état actuel et futur des feux tricolores de la ville d'Ingolstadt. Le véhicule peut alors conseiller une vitesse pour passer tous les feux au vert.

Pour un meilleur confort, le dispositif est couplé au régulateur de vitesse et permet d'adapter automatiquement la vitesse requise à l'approche du feu (si désiré). Si l'arrêt est inévitable le conducteur est informé à l'approche d'un feu rouge via son ordinateur de bord et le Start & Stop coupe le moteur automatiquement, ou réduit sa puissance sur un court laps de temps. Le conducteur voit sur son écran le temps restant avant le passage au vert [2]



Figure I- 3 : feu de circulation lié par des ondes Wi-Fi.

I.2.3 Avantages et inconvénients du feu de circulation intelligent

1) Avantages

Les feux de circulation intelligents présentent beaucoup d'avantages :

- Amélioration des conditions de déplacement.
- Augmentation de la capacité des carrefours.
- Il améliore également les flux de circulation de tous les jours.
- En imposant l'arrêt total aux usagers susceptibles de croiser leur trajectoire les véhicules ayant rarement à y effectuer un arrêt complet pour céder le passage.
- Les feux tricolores sont particulièrement adaptés à la gestion du trafic dense et rapide engendré par les véhicules motorisés sur des axes importants en agglomération
- Tendances de navette simplifiées avec des réductions probables de la quantité de temps nécessaire pour obtenir des points spécifiques à destination.

2) Inconvénients

Voici les divers inconvénients des feux de circulation :

- Une défaillance du signal due à une panne de courant électrique ou de tout autre défaut peut causer de la confusion.
- La collision arrière peut augmenter.
- Le cout élevé.

I.3 Systèmes de collection des données

La collection des données est la base de tous les systèmes de recherche et de gestion quelle que soit le domaine d'études, y compris le physique, le science sociales et humaines et même l'économie, bien que les méthodes variant selon la discipline, l'accent mis sur la garantie d'une collecte précise et honnête reste la même.

L'objectif de toute collecte de données est de recueillir des preuves et de mesures de qualité qui permettent à l'analyse de conduire à la formulation de réaction convaincantes et précise et crédibles aux circonstance qui ont été posées sur le volume et la charge de circulation des véhicules et les piétons jouent un rôle important dans la planification des infrastructures à long terme, les améliorations pour la sécurité de circulation et l'optimisation de l'entretien des routes et la gestion de trafic .

Dans le passé, cela impliquait d'avoir des gens debout au bord des routes et d'enregistrer leurs observations sur des blocs de papier. Ces dernières années, cette approche a été largement remplacée par des capteurs de trafic ,Le dispositif est généralement déployé à proximité immédiate de la chaussée et utilise un support routier tel que des tubes routiers pneumatiques posés en travers de la chaussée des capteurs piézo-électriques intégrés et des boucles inductives découpées ou une combinaison de ceux-ci pour détecter les véhicules qui passent , ou à l'aide de caméras vidéo de surveillance et traitement et même par les satellites ,le dispositif relier a un registre et calculateur électronique pour stocke les donne et analysée .Encore plus récemment, avec l'arrivée de vitesses Internet plus rapides, certaines sociétés de collecte des données sur le trafic ont décidé de faire compter (ou dénombrer) leurs enquêtes de trafic vidéo par des application téléphonique de suivi comme Google Maps [3].

Il existe plusieurs systèmes sur le marché en vas parles sur quelque mode les plus utilisait dans les pays et les plus précis :

I.3.1 Système ATLYST à l'Australie (Covid 2021).

Le confinement COVID à Perth, en Australie, a souligné la valeur de la collecte de données sur le trafic en temps réel pour surveiller les mouvements de véhicules pendant les périodes critiques. Collecter des données précises à partir de chaque véhicule et présenter les informations en temps réel (Figure I- 4) a' jusqu'à présent, été tout un défi. Cependant, après de nombreuses années de recherche et de tests, MetroCount a développé un système innovant qui fait exactement cela de manière fiable [4].



Figure I- 4 : Données de trafic en temps réel à South Perth.

La technologie des magnétomètres à énergie solaire intégrés avec des clous au milieu d'une voie de circulation. Ce système horodate chaque véhicule et enregistre sa longueur, sa vitesse et sa direction. Ces informations sont ensuite envoyées en temps réel au tableau de bord d'analyse en ligne ATLYST de MetroCount où les données sont clairement affichées sur une carte, dans des graphiques automatisés ou des rapports personnalisables. Dans le cadre d'un projet conjoint entre Main Roads Western Australia, la ville de South Perth et MetroCount, cette nouvelle technologie a été installée en janvier pour surveiller l'intersection de Mends Street et de Mill Point Road.À l'insu de tout le monde, l'installation a capturé les mouvements de véhicules extrêmement inhabituels causés par l'annonce de dernière minute d'un confinement complet dû au COVID.

la figure I-5 illustre Vue des points de collecte de données à l'intersection de Mends Street et Mill Point Road à l'aide d'ATLYSTLes données ont brossé un tableau (Figure I-5) en temps réel de la ruée de dernière minute pour les approvisionnements dimanche soir, suivie de cinq jours d'un débit extrêmement réduits qui révèlent des informations intéressantes sur les mouvements de la circulation locale.[4]

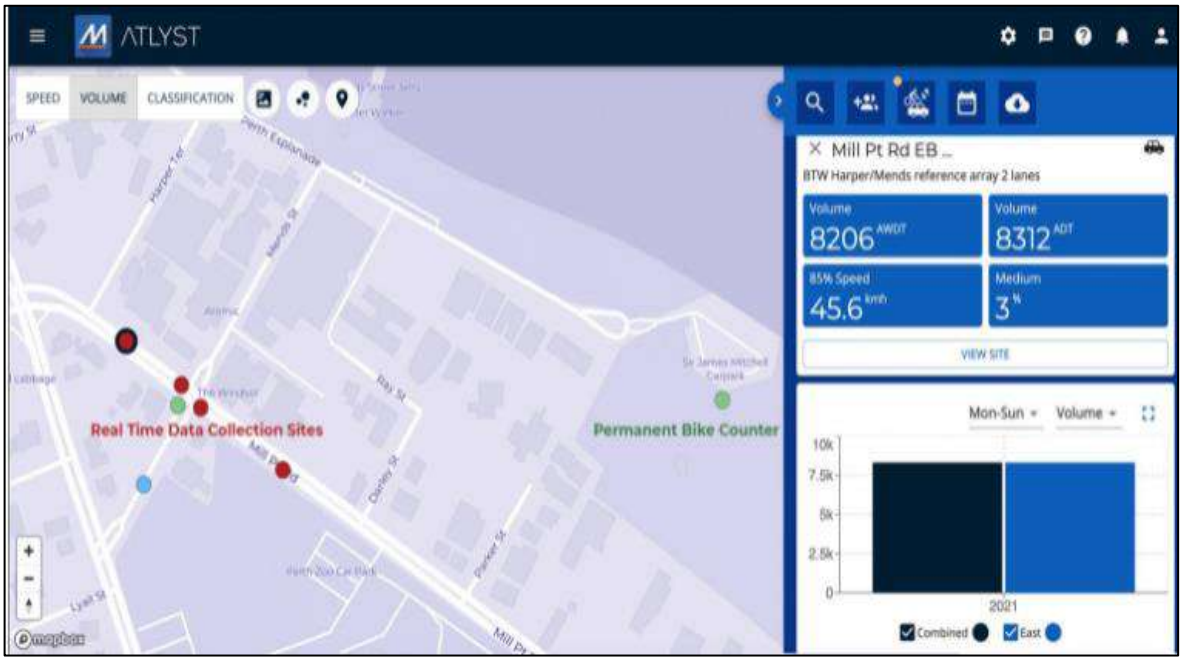


Figure I- 5 : l'intersection de Mends Street et Mill Point Road à l'aide d'ATLYST

I.3.2 TMA-122

Le TMA-122 est un détecteur hyperfréquences "tout-en-un" destiné à la gestion des feux tricolores. Il remplace la boucle d'approche et la boucle de présence en pied de feu [5]. Il compte les véhicules franchissant la ligne de feu, détecte leur approche ET leur présence en pied de feu. Ce système offre une réduction des coûts sur :

Votre bénéfice ? Une réduction des coûts sur :

- Le budget génie civil.
- Le budget sécurité.
- Le budget personnel d'intervention.

❖ **Fonctionnement et points forts**

Le TMA-122 (Figure I-6) comme indique la Figure I-7 détecte les véhicules approchant de la ligne d'arrêt, sur une distance de 20 ou 60 m. Deux relais sont activés : l'un lorsqu'un véhicule est en mouvement dans une zone de 20 ou 60 m devant la ligne d'arrêt, l'autre lorsqu'un véhicule est en mouvement ou s'arrête dans une zone (réglable) de 10 m par défaut devant la ligne d'arrêt (Figure I- 7). Le relais « présence pied de feu » est maintenu, dans la limite du temps d'oubli, jusqu'au redémarrage



Figure I- 6 : le détecteur TMA 122.

du véhicule. Ce relais bascule pour chaque véhicule franchissant la ligne d'arrêt. L'information ainsi transmise à l'automate peut donc être facilement convertie en comptage. Si un véhicule ne démarre pas lorsque le feu passe au vert, l'automate de contrôle est ainsi informé d'un éventuel blocage du carrefour.[5]

- Détection des véhicules en mouvement et à l'arrêt
- Portée : jusqu'à 60 m pour le mouvement, 10 ou 15 m pour la présence en pied de feu
- Réglages sans ouverture du boîtier
- Auto surveillance

❖ **Confort d'installation et d'utilisation**

- Installation hors sol
- Fonctionne par tous les temps
- Sans entretien
- Fiable

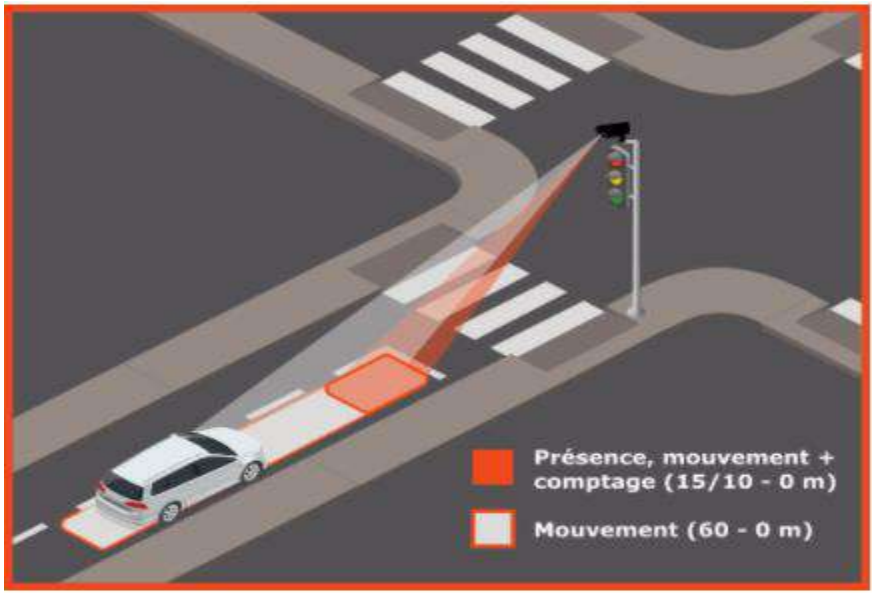


Figure I- 7 : architecture d'implémentation de TMA-122

I.3.3 Collecte de données de circulation avec KiTraffic Statistiques

KiTraffic Statistiques de Kistler permet la collecte automatique des données de circulation en temps réel. Ce système compact est constitué d'un journal de données de pesage dynamique, de capteurs et de composants pré câblés pouvant être installés sur tous les revêtements, rapidement et facilement. Les capteurs à quartz enterrés Linas Compact sont installés à 25 mm sous le revêtement, ce qui donne leur donne l'avantage d'une longue durée de vie et une collecte de données de trafic précise, indépendamment des conditions météo et de circulation.[6]

Ces systèmes permettent d'améliorer la planification de l'infrastructure, d'optimiser l'entretien des routes et d'uniformiser davantage le flux de circulation. De plus, les capteurs de haute précision au cristal de quartz de Kistler supportent sans problèmes des conditions climatiques difficiles et la circulation rapide [6].

- **Les caractéristiques de KiTraffic**

- Des mesures fiables et précises des charges de véhicules
- Une plus grande durée d'utilisation grâce au capteur enterré Lineas Compact unique en son genre, installé sous le revêtement
- Installation facile sous tout type de route
- Sans entretien, même pour les utilisations long-terme
- Facile et rapide à installer grâce aux composants précâblés
- Intégration simple dans les solutions ITS existantes
- Conformes aux normes COST 323 et ASTM E1218-09
- Le contrôle automatisé de la circulation simplifie le processus de planification des maintenances et réduit significativement les coûts.

I.3.4 Feux de circulation ont bas radar

- **Feu Comportemental**

WP Signalisation est un fabricant de feu comportemental permettant changer le comportement des usagers et lutter contre les excès de vitesse. Les véhicules son détecté par un radar qui permet aussi de calcules la vitesse.[7][8]

❖ **Le système a deux modes de travaille**

1) Feu tricolore mode Rouge

Le feu tricolore est maintenu par défaut au rouge.

En cas d'approche d'un usager, le feu reste au rouge si l'automobiliste dépasse la vitesse autorisée, et le feu passe au vert si le véhicule est détecté en dessous de la vitesse limite autorisée.

2) Feu tricolore mode Vert

Le feu tricolore est maintenu par défaut au vert.

En cas de dépassement de la vitesse limite autorisée, le feu bascule à l'orange, puis au rouge. Ce mode de fonctionnement sanctionne les comportements de vitesse excessive.

❖ **Fonctionnement**

- Feu tricolore Multiple
- Fonctionnement déclenché par la détection d'une vitesse excessive
- Détection par un radar doppler déporté placé au niveau du feu tricolore

❖ **Détection de la vitesse**

- Utilisation d'un système de radar doppler
- Distance de détection des véhicules paramétrable de 50 à 150 mètres.
- Plage de vitesse mesurable : 5 à 255 km/h
- Précision de mesure : ± 1 km/h
- Mesure en continu

I.3.5 Détecteur MC2224

Le MC2224 (Figure I-8) fonctionne avec deux boucles d'induction par voie, conformément à la spécification TLS. Basé sur le MC2024/CD9054 éprouvé et testé, le détecteur utilise la technologie de pointe. [9]



Figure I- 8 : Détecteur MC2224

❖ **Fonctionnalités**

Acquisition des données de trafic selon TLS1 / BASt2 pour deux voies avec mesure de vitesse et de longueur, détection de sens de circulation et de mauvais sens avec systèmes à double boucle, taux d'occupation en liaison avec un contrôleur

Le programme d'exploitation Loop Master, disponible gratuitement, offre des fonctions étendues d'analyse des boucles et des détecteurs. La personnalisation avant-gardiste permet une configuration et un réglage individuels des éléments de commande et des fonctions.

I.3.6 Détecteur MC3224

Le détecteur de boucle MC3224 (Figure I-9) a été spécialement conçu pour la détection de véhicules avec classification et mesure de vitesse pour l'acquisition des données de trafic et la gestion du trafic. Le détecteur est conçu pour un montage sur rail DIN et comprend un module complet de protection contre les surtensions pour les boucles inductives. [9]

❖ **Descriptif fonctionnel**

Le détecteur offre désormais les fonctions et les caractéristiques exceptionnelles des détecteurs de classification SWARCO TRAFFIC SYSTEMS en technologie enfichable 19" également en version à montage sur rail DIN. Basé sur le MC2224 éprouvé et testé et

le CD9054 certifié, il comprend un module complet de protection contre les surtensions. Cette intégration minimise l'effort de câblage et réduit considérablement l'espace requis.

Le MC3224 classe les véhicules en (8+1), (5+1) ou deux classes accès. À TLS*. La classification n'est pas influencée par ex. conditions météorologiques. Lors de l'activation de la logique directionnelle, des rapports de mauvais sens de conduite peuvent être générés. Le type de véhicule est déterminé au moyen de courbes passantes qui présentent des caractéristiques dépendant des différentes classes et des types de boucles utilisées.



Figure I- 9 : Détecteur MC3224

I.3.7 Détecteur de trafic TDC1

Détecteurs de trafic Pire non intrusifs à faible puissance pour l'acquisition de données de trafic à voie unique

La série TDC1 (Figure I-10) sont des détecteurs de trafic avancés utilisant la technologie infrarouge passive. Données de trafic complètes, y compris la classe de véhicule, la vitesse, la longueur, le temps d'occupation et les intervalles de temps sont fournis via le bus de données RS 485. [9]

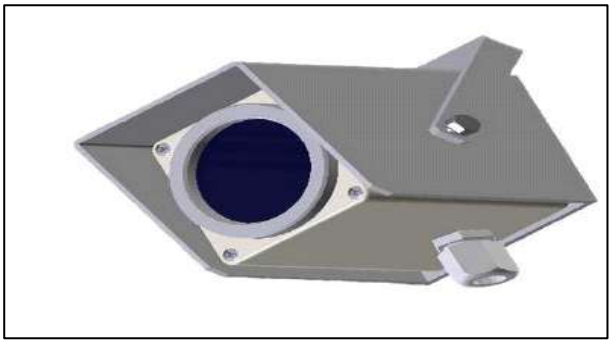


Figure I- 10 : Détecteur De Trafic TDC1

❖ Applications typiques

Les détecteurs de la série TDC1 sont spécialement conçus pour une variété d'applications de collecte de données de trafic et de contrôle du trafic où

Une faible consommation d'énergie est requise :

- Comptage de véhicules (volume)
- Vitesse du véhicule individuel
- Classement du véhicule
- Détection de présence

❖ Principe de fonctionnement

- Les détecteurs de trafic TDC1 utilisent plusieurs zones de détection PIR.
- Une combinaison de canaux de détection statiques et dynamiques
- Un total de cinq zones de détection. Le contraste du rayonnement thermique

D'un véhicule entrant ou traversant les zones de détection contre le rayonnement de fond de la surface de la route est corrélé au passage ou à la présence d'un véhicule. Le traitement sophistiqué du signal transforme les données analogiques des capteurs en informations numériques pour chaque événement sans avoir besoin d'équipement informatique externe.

❖ Montage

Les points de montage recommandés sont des portiques, des viaducs ou des ponts ou encore sur un poteau au bord de la route. Des performances et une fiabilité nettement supérieure sont le résultat de :

- Détection PIR multicanale.
- Compensation complète de la température sur toute la plage de température.
- Algorithmes sophistiqués éliminant l'impact des changements environnementaux.

❖ Accessoires

- Interface RS 485 & Logiciel

Pour la communication entre les détecteurs et un PC pendant la mise en service et la maintenance, un module d'interface en combinaison avec le logiciel dédié est nécessaire. Le module d'interface et le logiciel doivent être commandés séparément.

- Accessoires de montage

Un support pour le montage du détecteur sur un poteau rond est disponible comme accessoire séparé (non inclus dans la livraison standard).

- **Outil d'alignement**

Un outil d'alignement mécanique pour une installation rapide et précise est disponible. L'outil peut être facilement placé sur l'appareil et permet à l'installateur d'aligner le détecteur à l'aide du niveau et du viseur intégrés. L'installation du système et typiquement que la figure I- 11 indique

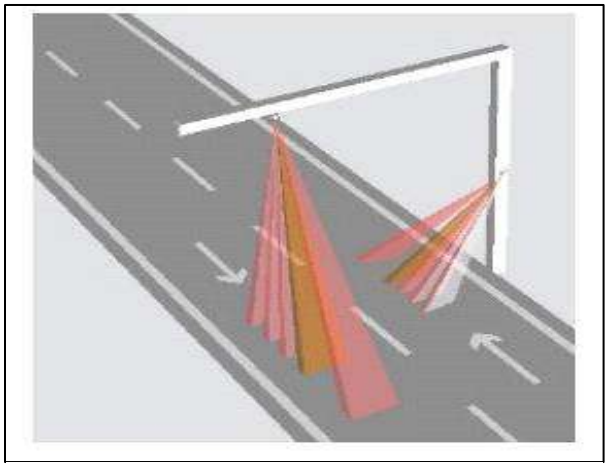


Figure I- 11 : Champ De Vision

I.3.8 GECKO TRAFIC / CLASSE.

GECKO GECKO TRAFFIC / CLASS (Figure I-12) fait partie de la solution système Swarco Cloud, qui suit systématiquement l'approche IoT (Internet des objets) et connecte les appareils de terrain avec une intelligence centrale. Les appareils de terrain envoient leurs données directement à notre service cloud, qui prend en charge le traitement individuel des données, leur transmission et leur intégration dans des systèmes tiers. [9]



Figure I- 12 : GECKO TRAFIC / CLASSE

L'avantage de la solution est que l'installation et la configuration sont aussi simples que la mise en service d'un smartphone moderne, que les interfaces standardisées entre l'appareil de terrain et le cloud garantissent une communication sûre et stable et que la sortie est extrêmement flexible.

Les appareils de terrain Gecko transmettent leurs données au Gecko Dashboard, un module intégré de MyCity Monitoring. Bien entendu, la séparation technique des systèmes pour assurer la sécurité du système ainsi que la protection des données et un service d'exploitation logicielle certifié ISO27001 sont une évidence.

Pour la connexion de systèmes tiers et de systèmes clients individuels, nous disposons d'interfaces cloud standard (API) et mettons en œuvre des services d'exportation individuels. L'effort est ici nettement moindre qu'avec les systèmes classiques et toujours transparent, sûr et stable grâce aux processus de communication surveillés des appareils de terrain via le cloud vers le système cible.Partagez vos données de comptage avec le public et augmentez par ex. la prise de conscience des charges de trafic ou documentez les progrès en montrant vos projets de construction de routes.

Le système Gecko a été conçu pour l'acquisition autonome de données de trafic avec alimentation par batterie et transmission de données via un réseau de communication mobile. Presque tous les domaines de l'acquisition de données de trafic sont couverts, du comptage de base des véhicules avec Gecko Traffic jusqu'à la classification des véhicules, par ex. acc. À TLS1 ou ASTRA2 avec la classe Gecko.

Complété par la plate-forme SWARCO-WEB basée sur le cloud et les interfaces WEB, le système Gecko fait déjà partie de l'IoT (Internet des objets)

- **Communication et évaluation**

Transmission automatique de données sans fil via un modem de réseau de communication mobile vers la plate-forme SWARCO-WEB (cloud SWARCO).

Système basé sur WEB pour la visualisation, l'évaluation, la documentation et l'exportation des valeurs de comptage.

Agrégation des données : données d'intervalle séparées par voie de circulation pour compter les valeurs de toutes les classes de véhicules configurées et de la vitesse moyenne.

❖ **Fonctionnement**

Gecko fournit les fonctions et les excellentes caractéristiques du Détecteur à boucle inductive Swarco dans un système autonome pour acquisition de données de trafic. Le cœur du système sont les détecteurs.Pour le montage sur rail DIN, qui ont été étendus pour inclure la transmission de données via un réseau de communication mobile.

En raison de la réduction considérable de la consommation d'énergie, il est désormais possible de mettre en œuvre une acquisition de données autonome jusqu'à 6 mois à l'aide d'un système de détection facile à installer. Le raccordement alternatif à la tension secteur ou l'installation photovoltaïque disponible en option permet un fonctionnement permanent avec une autonomie de deux semaines en cas de panne de courant (batterie rechargeable intégrée).

La Figure I-13 ci-dessus détermine la façon que Les données de trafic sont transmises sous forme agrégée, les périodes d'intervalle pour l'acquisition et la transmission des données peuvent être ajustées. Les données de mesure et de diagnostic (par exemple, ID du lieu de mesure, état de la batterie, intensité du réseau de communication mobile, défaillance de la boucle, fréquence de mesure) sont accessibles à tout moment via la plateforme.

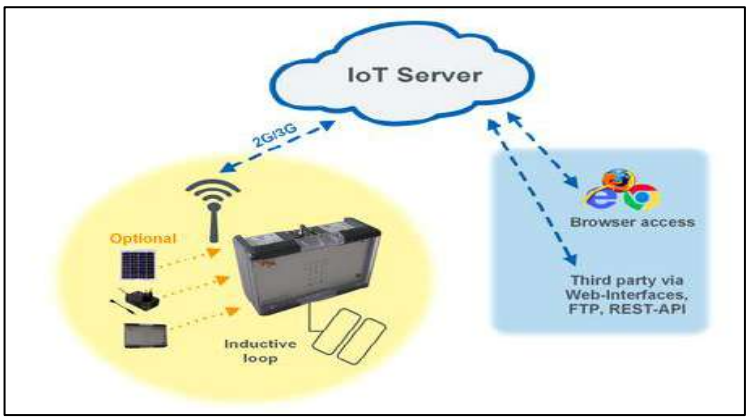


Figure I- 13: GECKO TRAFIC / CLASSE SWARCO-WEB.

I.4 Conclusion

Les systèmes intelligents de gestion des routes et de collection des données sont un sujet important et nécessaire pour de nombreux chercheurs afin de résoudre le problème des embouteillages qui devenu une obsession qui gêne de nombreux peuples.

Dans ce chapitre, nous avons présenté quelques feux de circulation intelligents à travers le monde. Ensuite, nous avons présenté quelques systèmes utilisés pour la collection des données routières.

CHAPITRE II

Généralités sur Deep Learning, Modèles pré-entrainés et YOLO

II.1 Introduction

Les humains regardent une image et savent instantanément quels objets s'y trouvent, où ils sont et comment ils interagissent. Le système visuel humain est rapide et précis, ce qui nous permet d'effectuer des tâches complexes comme la conduite automobile sans trop de réflexion consciente. Des algorithmes rapides et précis pour la détection des objets permettraient aux ordinateurs de conduire des voitures sans capteurs spécifiques, permettraient aux appareils fonctionnels de transmettre des informations en temps réel aux utilisateurs humains et débloqueraient le potentiel ou les systèmes robotiques réactifs et polyvalents.

Dans ce chapitre on va donner une description générale sur Deep Learning et les modèles pré-entrainés dans sa première partie. Ensuite, on va détaillée l'algorithme YOLO utilisé dans notre projet.

II.2 Apprentissage profond (Deep Learning)

II.2.1 Définition

L'apprentissage profond (Deep Learning en anglais) est un sous-domaine de l'apprentissage automatique qui a connu un véritable renouveau depuis 2006. Ces modèles sont construits à partir d'éléments de calcul de base appelés neurones, disposés en général en couches successives. Ces neurones sont reliés entre eux par des connexions dont l'importance est ajustée durant la phase d'apprentissage. Chaque neurone transforme ses entrées en une sortie en appliquant une fonction de transfert à l'ensemble des informations qui lui parviennent par ces connexions. Les réseaux de neurones sont des systèmes schématiquement inspirés du fonctionnement du cerveau. Toutefois, il ne s'agit que d'une analogie grossière. Après apprentissage, on constate que le réseau de neurones a décomposé les données d'entrée en différentes couches d'abstraction hiérarchisées. Par exemple, pour identifier des visages sur une photo, la première couche identifie uniquement les formes élémentaires constituant les visages (lignes, cercles, etc..), la couche suivante les assemble en gros traits du visage (nez, lèvres, front etc..) et la description est progressivement affinée (taille du nez, largeur des narines etc..) jusqu'à recomposer complètement les visages. Les réseaux de

neurones sont particulièrement efficaces dans le traitement de données assimilables à des signaux : son, image, langage, vidéos car ils sont capables d'en capturer les structures complexes sur les plans spatial et temporel. On peut noter particulièrement deux types de modèles que sont les réseaux de neurones Convolutionnels (standard dans le domaine de l'image) et les réseaux de neurones récurrentsdans lesquels la notion de couche disparaît. On les utilise surtout pour les signaux temporels et le langage. La figure II.1 démontre la déférence entre le simple réseaux neurone et un réseau neurone profonde [10].

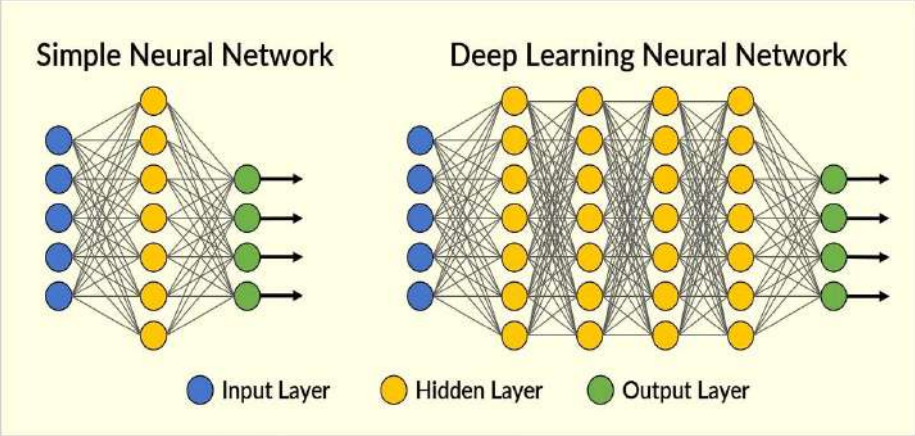


Figure II- 1 : Déférence entre Deep-Learning et réseaux de neurone simple

II.2.2 Utilisation du Deep Learning

Le Deep_Learning peut être utilisé dans de nombreux domaines et application comme :

- Traduction automatique,
- Voiture autonome,
- Diagnostic médical,
- Recommandations personnalisées,
- Modération automatique des réseaux sociaux,

- Prédiction financière et trading automatisé,
- Identification de pièces défectueuses,
- Détection de malwares ou de fraudes,
- Chatbots (agents conversationnels),
- Exploration spatiale,
- Robots intelligents.[10]

II.2.3 Réseaux Neurones Convolutionnels (CNN)

II.2.3.1 Définition

Un réseau de neurone convolutif (CNN : Convolutional Neural Network) est un réseau de neurones avec une ou plusieurs couches de convolution. C'est un cas particulier de perception multicouches adapté aux images. Ce type de réseaux a été inspiré par les travaux effectués sur le fonctionnement biologique du cortex visuel chez les mammifères. Il est basé sur trois idées architecturales :

- Des champs récepteurs locaux associés à des convolutions qui permettent de détecter des caractéristiques élémentaires sur l'image, formant ainsi une carte de caractéristiques.
- Un principe de partage des poids qui consiste à apprendre les mêmes paramètres (ou poids) d'une convolution et par conséquent extraire les mêmes caractéristiques pour toutes les positions sur l'image. C'est l'idée clé des CNNs permettant de réduire considérablement la complexité en diminuant le nombre de paramètres à apprendre, et d'avoir ainsi des architectures multicouches qui opèrent sur des entrées de grande dimension tout en étant de taille raisonnable (ce qui n'était pas réalisable avec les MLPs). De plus, le partage des poids permet d'améliorer la capacité de généralisation du réseau.
- Des opérations de sous-échantillonnage ou de pooling qui permettent de réduire la sensibilité aux translations, ainsi que le coût du traitement.

La structure du CNN a été proposée initialement par Fukushima en 1980 grâce à ses travaux sur le Néocognitron. Cependant, il n'a pas été largement popularisé à cause de

la complexité de l'algorithme d'apprentissage. Ce n'est que dans les années 1990, que Lecun a popularisé les réseaux de neurones convolutifs en appliquant son premier réseau développé LeNet-1 à la reconnaissance des caractères. [10]

II.2.3.2 Description des CNNs

L'architecture d'un CNN peut être divisée en deux parties (Figure II-2). La première a pour but d'extraire des caractéristiques. Elle est composée d'une succession de couches de convolution et de couches de sous-échantillonnage ou de pooling. La deuxième partie est formée d'une part, d'un ensemble de couches totalement connectées destinées à construire des caractéristiques de haut niveau en combinant celles de bas niveau issu de l'étage précédent, et d'autre part, d'une couche de sortie qui classe les vecteurs de caractéristiques des données d'entrée.

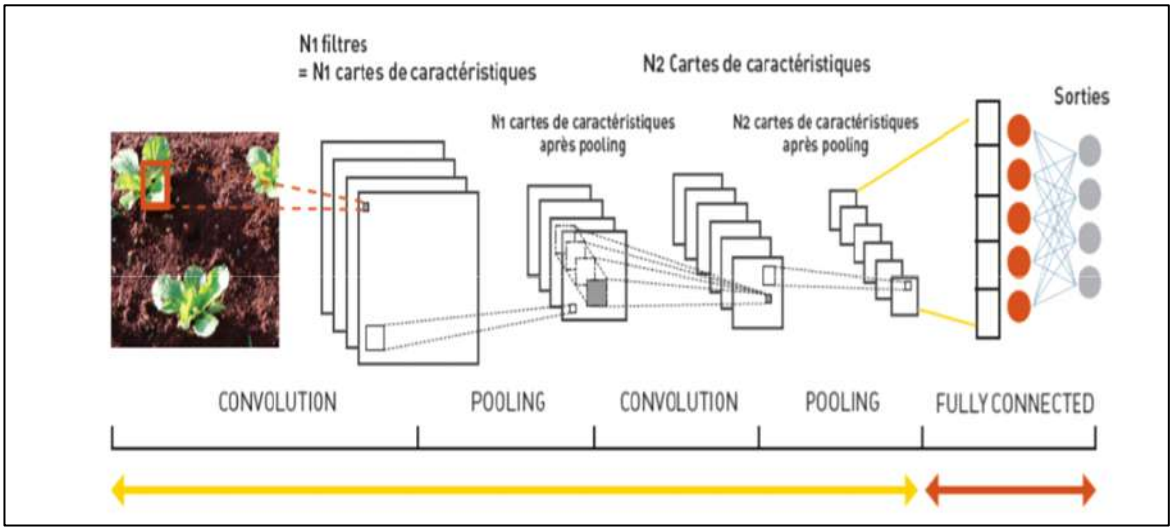


Figure II- 2: Architecture d'un réseau CNN

L'alternance entre convolution et sous-échantillonnage donne une structure pyramidale au réseau CNN. Ainsi, les couches supérieures représentent des caractéristiques de plus en plus globales de l'entrée, car leur champ perceptif correspond à une plus grande partie de l'entrée [10].

- **Convolution**

La convolution est un outil mathématique simple qui est très largement utilisé pour le traitement d'image, ce qui explique que les réseaux de neurones à convolution soient particulièrement bien adaptés à la reconnaissance d'image.

La convolution agit comme un filtrage. On définit une taille de fenêtre qui va se balader à travers toute l'image. Au tout début de la convolution, la fenêtre sera positionnée tout en haut à gauche de l'image puis elle va se décaler d'un certain nombre de cases (c'est ce que l'on appelle le pas) vers la droite et lorsqu'elle arrivera au bout de l'image, elle se décalera d'un pas vers le bas ainsi de-suite jusqu'à ce que le filtre soit parcourue la totalité de l'image.

Le but est de se servir des valeurs présentes dans le filtre à chaque pas. La convolution va effectuer une opération avec ces pixels. Il peut s'agir de n'importe quelle opération, par exemple (Figure II- 3) : on extrait la valeur la plus grande (soit le pixel avec la plus grande valeur).[10]

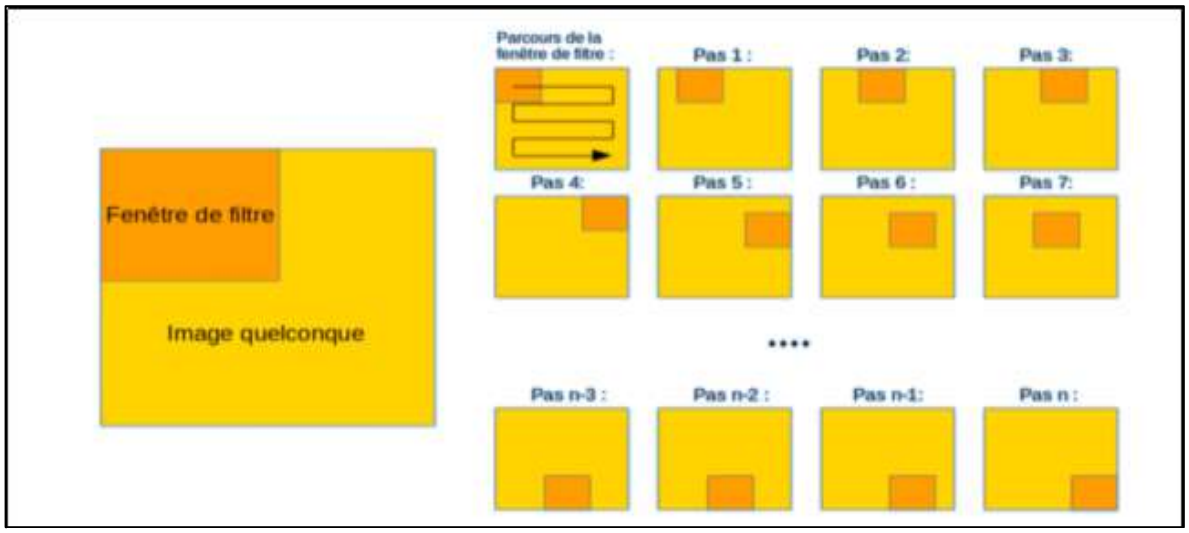


Figure II- 3 : schéma du parcours de la fenêtre de filtre sur l'image.

❖ Exemple de convolution

Figure II.4 illustre un exemple de convolution dont la configuration est comme suite : Opération = Argument maximale, Pas horizontale = 1 pixel, Pas vertical = 1 pixel.

On fait glisser la fenêtre en orange (Figure II- 4) et à chaque pas on récupère la valeur la plus grande parmi les 9 valeurs de pixels. On remarque que la sortie de la convolution, que l'on peut appeler « carte de caractéristiques », à des dimensions plus petites que celle de l'image en entrée. [10]

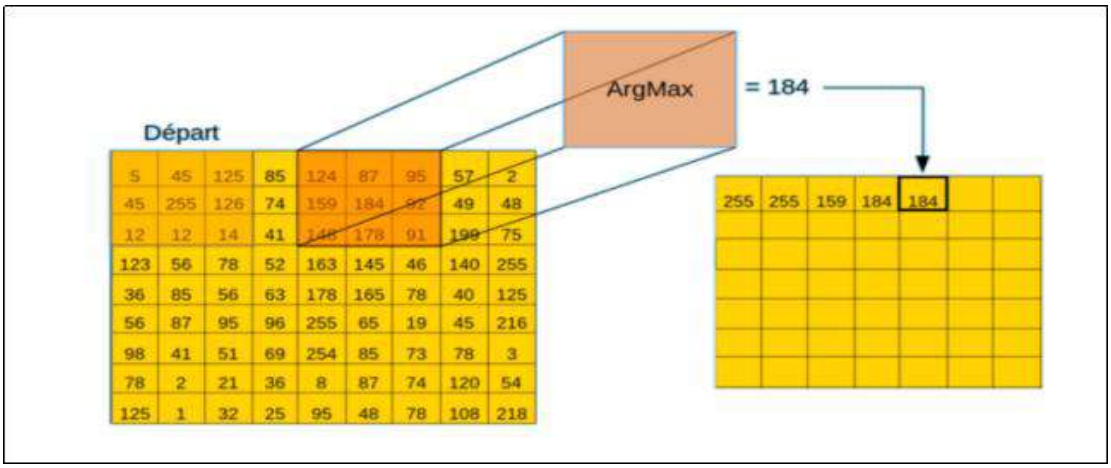


Figure II- 4: exemple d'une convolution

II.3 Modèles pré entrainé

II.3.1 Introduction

Dans le monde d'aujourd'hui, la RAM sur une machine est bon marché et est disponible en abondance. Nous avons besoin de centaines de Go de RAM pour exécuter un problème d'apprentissage automatique supervisé super complexe - cela peut être le vôtre pour un petit investissement / loyer. D'un autre côté, l'accès aux GPU n'est pas si bon marché. Nous avons besoin d'accéder à une centaine de Go de VRAM sur les GPU - ce ne sera pas simple et impliquerait des coûts importants.

Maintenant, cela peut changer à l'avenir. Mais pour l'instant, cela signifie que nous devons être plus intelligents dans la façon dont nous utilisons nos ressources pour résoudre les problèmes de Deep Learning. Surtout lorsque nous essayons de résoudre des problèmes complexes de la vie réelle dans des domaines tels que la reconnaissance d'images et la reconnaissance vocale. Une fois que Nous avons quelques couches cachées dans votre modèle, l'ajout d'une autre couche de couche cachée nécessiterait d'immenses ressources.

Heureusement, il existe quelque chose appelé « Apprentissage par transfert » qui nous permet d'utiliser des modèles préformés d'autres personnes en apportant de petits changements. Dans ce chapitre, on va expliquer comment nous pouvons utiliser des modèles pré-entraînés pour accélérer nos solutions.

II.3.2 Definition

En terme simple, un modèle pré-entraîné est un modèle créé par quelqu'un d'autre pour résoudre un problème similaire. Au lieu de créer un modèle à partir de zéro pour résoudre un problème similaire, nous utilisez le modèle formé sur un autre problème comme point de départ.

Par exemple, si nous voulons construire une voiture autodidacte. nous pouvons passer des années à créer un algorithme de reconnaissance d'images décent à partir de zéro ou nous pouvons prendre un modèle de création (un modèle pré-entraîné) de Google qui a été construit sur les données ImageNet pour identifier les images dans ces images.

Un modèle pré-entraîné peut ne pas être précis à 100 % dans votre application, mais il permet d'économiser d'énormes efforts pour réinventer la roue. Permettez-moi de nous montrer cela avec un exemple récent [11].

II.3.3 Comment utiliser les modèles pré-entraînés?

Quel est notre objectif lorsque nous formons un réseau de neurones ? Nous souhaitons identifier les poids corrects pour le réseau par plusieurs itérations avant et arrière. En utilisant des modèles pré-entraînés qui ont été préalablement entraînés sur de grands ensembles de données, nous pouvons directement utiliser les poids et l'architecture obtenus et appliquer l'apprentissage sur notre énoncé de problème. C'est ce qu'on appelle l'apprentissage par transfert. Nous « transférons l'apprentissage » du modèle pré-entraîné à notre énoncé de problème spécifique.

nous devons être très prudent lorsque nous choisissons le modèle pré-entraîné que nous devons utiliser dans votre cas. Si l'énoncé du problème que nous avons sous la main est très différent de celui sur lequel le modèle pré-entraîné a été formé, la prédiction que nous obtiendrions serait très inexacte. Par exemple, un modèle préalablement formé pour la reconnaissance vocale fonctionnerait horriblement si nous essayions de l'utiliser pour identifier les objets qui l'utilisent.

Nous avons la chance que de nombreuses architectures pré-entraînées soient directement disponibles pour nous dans la bibliothèque Keras. L'ensemble de données ImageNet a été largement utilisé pour construire diverses architectures car il est suffisamment grand (1,2 million d'images) pour créer un modèle généralisé. L'énoncé du problème consiste à former un modèle capable de classer correctement les images en 1 000 catégories d'objets distinctes. Ces 1 000 catégories d'images représentent des classes d'objets que nous rencontrons dans notre vie de tous les jours, telles que des espèces de chiens, de chats, divers objets ménagers, des types de véhicules, etc.

Ces réseaux pré-entraînés démontrent une forte capacité à généraliser à des images en dehors de l'ensemble de données ImageNet via l'apprentissage par transfert. Nous apportons des modifications au modèle préexistant en ajustant le modèle. Puisque nous supposons que le réseau préformé a été assez bien formé, nous ne voudrions pas modifier les poids trop tôt et trop. Lors de la modification, nous utilisons généralement un taux d'apprentissage inférieur à celui utilisé pour l'entraînement initial du modèle [11].

II.3.4 Façons d'affiner un modèle

- **Extraction de caractéristiques**

Nous pouvons utiliser un modèle pré-entraîné comme mécanisme d'extraction de caractéristiques. Ce que nous pouvons faire, c'est supprimer la couche de sortie (celle qui donne les probabilités d'être dans chacune des 1000 classes), puis utiliser l'ensemble du réseau comme extracteur de caractéristiques fixes pour le nouvel ensemble de données [11].

- **Utiliser l'architecture du modèle pré-entraîné**

Ce que nous pouvons faire, c'est utiliser l'architecture du modèle pendant que nous initialisons tous les poids de manière aléatoire et que nous entraînons à nouveau le modèle en fonction de notre ensemble de données.

- **Entraînez certaines couches tout en gelant d'autres**

Une autre façon d'utiliser un modèle pré-entraîné consiste à s'entraîner partiellement. Ce que nous pouvons faire, c'est conserver les poids des couches initiales du modèle gelés pendant que nous ne recyclons que les couches supérieures. Nous pouvons essayer de déterminer combien de couches doivent être gelées et combien de couches doivent être formées [11].

II.3.5 Présentation des models pré-entraînés

Dans cette section on va présenter quelques modèles pré-entrainés et disponible à utiliser dans plusieurs domaines [11]

II.3.5.1 Détection d'objets

- Mask R-CNN : Mask R-CNN est un framework flexible développé dans le but de segmenter les instances d'objets

- YOLOv (2/3/4) : YOLO est un framework de détection d'objets ultra populaire pour les applications d'apprentissage en profondeur.

- Mobile Net : Comme son nom l'indique, MobileNet est une architecture conçue pour les appareils mobiles Ce modèle particulier est livré avec des poids pré-entraînés sur la base de données populaire ImageNet

- Ripe/unripe tomato classification : Dans ce modèle de classification nous devons identifier si la tomate dans l'image donnée est grandie ou non à l'aide d'un modèle Keras VGG16 pré-entraîné.

- Car classification : Le modèle a été entraîné à l'aide des modèles pré-entraînés VGG16, VGG19 et InceptionV3. Le réseau VGG se caractérise par sa simplicité, n'utilisant que 3x3 couches convolutives empilées les unes sur les autres avec une profondeur croissante.

II.3.5.2 Reconnaissance faciale et régénération

◆ VGG-Face Model : Ses applications couvrent un large éventail de tâches – déverrouillage du téléphone, détection de foule, analyse des sentiments en analysant le visage, entre autres. Créer un modèle de reconnaissance faciale à partir de zéro est une tâche ardue, VGG-Face est un ensemble de données qui contient 2 622 identités uniques avec plus de deux millions de visages. Ce modèle pré-entraîné a été conçu selon la méthode suivante :

- Vgg-face-keras : convertissez directement le modèle vgg-face en modèle keras
- Vgg-face-keras-fc : convertissez d'abord le modèle vgg-face Caffe en modèle mxnet, puis convertissez-le en modèle keras

◆ 3D Face Reconstruction from a Single Image (FigureII-5) : C'est une implémentation vraiment cool de l'apprentissage en profondeur. Nous pouvons déduire de l'image ci-dessus comment fonctionne ce modèle afin de reconstruire les traits du visage dans un espace en 3 dimensions [11].

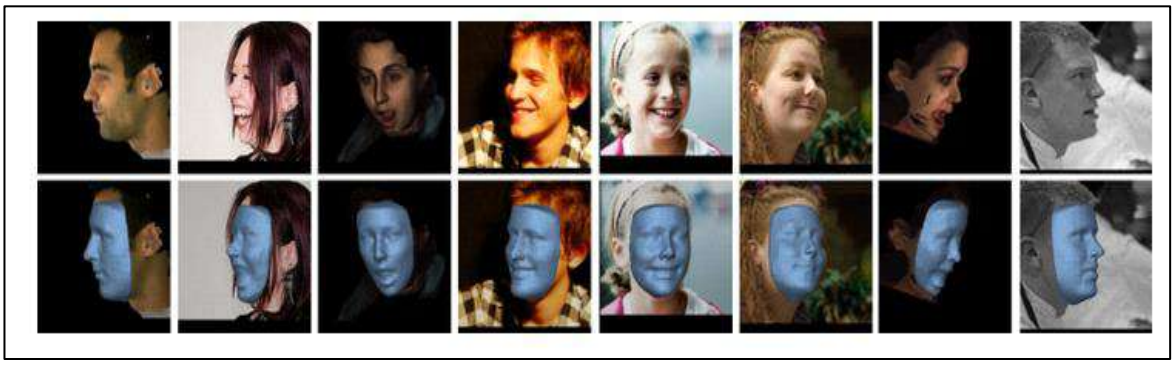


Figure II- 5: reconstruction de visage en 3d

II.3.5.3 Segmentation

Semantic Image Segmentation – Deeplabv3 (Figure II- 6) : La segmentation sémantique des images consiste à attribuer une étiquette sémantique à chaque pixel d'une image. Ces étiquettes peuvent être « ciel », « voiture », « route », « girafe », etc. Cette technique permet de trouver les contours des objets et impose ainsi des restrictions sur les exigences de précision. Deeplabv3 est le dernier modèle de segmentation d'images sémantique de Google

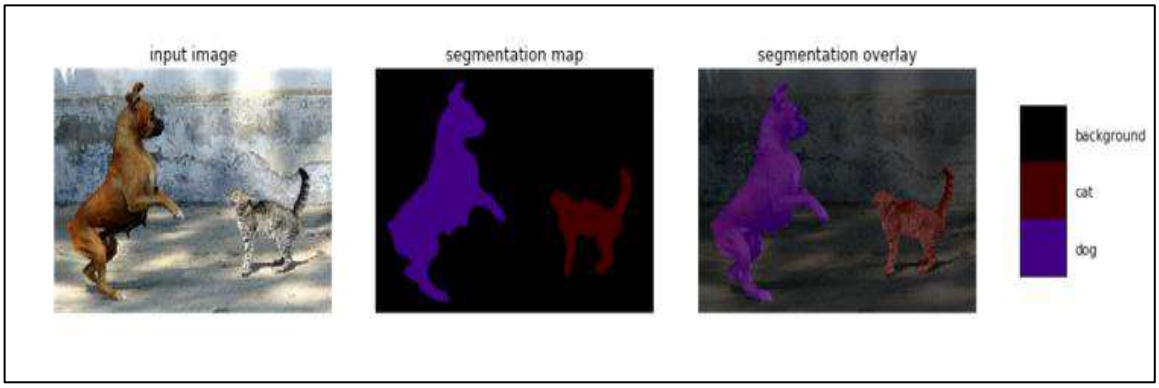


Figure II- 6 : semantic image segmentation

◆ Robot Surgery Segmentation : Ce modèle tente de résoudre le problème de la segmentation d'images des instruments chirurgicaux dans un scénario de chirurgie assistée par robot.

II.3.5.4 Divers

◆ Image Captioning : nous souvenons de ces jeux où l'on nous donnait des images et deviez créer des légendes ? C'est essentiellement ce qu'est le sous-titrage d'images. Il utilise une combinaison de PNL et de vision par ordinateur pour produire les légendes. Cette tâche a été un défi pendant longtemps car elle nécessite d'énormes ensembles de données avec des images et des scénarios impartiaux.

Figure II- 7 démontre un chronogramme de l'évaluation et l'invention des programmes pré entraîné depuis année 2013 jusque a maintenant.Les algorithmes situent en bas sont de premier étage de détection et les autres en bas de deux étages a analyse et détection des objets.

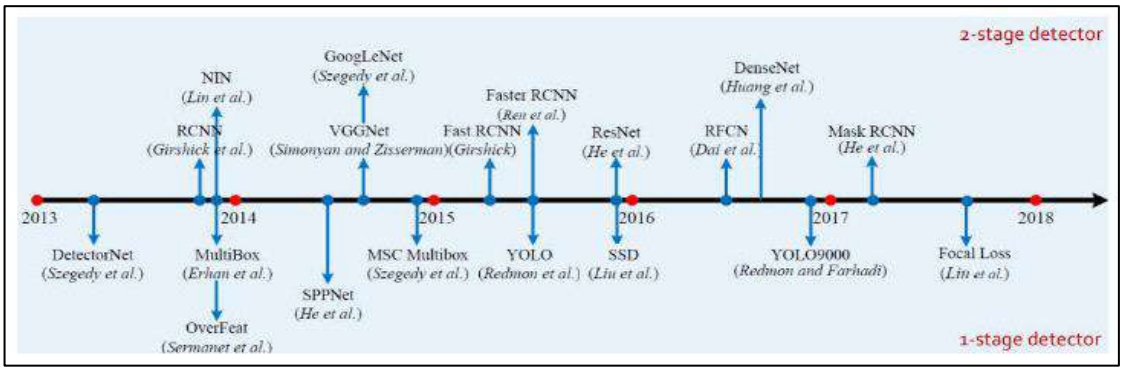


Figure II- 7 : chronologique des méthodes de détection d'objets.

II.4 Algorithme YOLO (You Only Look Once)

II.4.1 Introduction

Pour acquérir une compréhension complète de l'image, nous ne devons pas nous concentrer uniquement sur la classification des différentes images, mais également essayer d'estimer avec précision les concepts et les emplacements des objets contenus dans chaque image. Cette tâche est appelée détection d'objet, Entant que l'un des problèmes fondamentaux de la vision informatique, la détection d'objets est capable de fournir des informations précieuses pour la compréhension sémantique des images et des vidéos, Cependant, en raison des grandes variations des points de vue, des poses, des occlusions et des conditions d'éclairage, il est difficile de réaliser parfaitement la détection d'objets avec une tâche de localisation d'objet supplémentaire. Une attention particulière a été portée à ce domaine ces dernières années. La définition du problème de la détection d'objet est de déterminer où se trouvent les objets dans une image donnée (localisation d'objet) êta quelle catégorie chaque objet appartient (classification d'objet)

Les détecteurs d'objets récents basés sur l'apprentissage en profondeur peuvent être classés en deux grandes catégories, les détecteurs à un étage et les détecteurs à deux étages. Les modèles de la famille R-CNN, Fast RCNN et Faster RCNN sont des détecteurs à deux étages, ils sont basés sur la structure de la région de proposition. La détection se déroule en deux étapes :

1. Premièrement, le modèle propose un ensemble de régions d'intérêt par recherche sélective ou un réseau de propositions régionales. Parce que les candidats potentiels à la boîte englobante peuvent être infinis.

2. Ensuite, un classificateur ne traite que les candidats de région ; l'autre catégorie saute l'étape de proposition de région et exécute la détection directement sur un échantillonnage dense d'emplacements possibles. Il utilise la structure Régression / Classification, c'est plus rapide et plus simple, mais cela pourrait potentiellement ralentir un peu les performances. Dans cette section, nous expliquons comment fonctionne un algorithme de détection d'objets en une étape. Notre objectif est d'appliquer un détecteur d'objet à un calculateur (Ordinateur) pour la détection de trafic nous avons donc besoin que la détection soit rapide et en temps réel, pour cela. Nous nous concentrons sur la deuxième

catégorie, les détecteurs à un étage, qui utilise une structure connue sous le nom de Régression/Classification, dans cette section nous expliquons en détail le fonctionnement de l'algorithme de Régression/Classification.

Les détecteurs à une étape basés sur la régression/classification globale, mappant directement les pixels de l'image aux coordonnées de la boîte englobante et aux probabilités de classe, peuvent réduire les dépenses de temps. Nous nous concentrons sur le modèles importants, You Only Look Once (YOLO)

YOLO a été publié en 2015 par Joseph Redmon. La 2eme version de YOLO, appelée YOLO9000, également connue sous le nom de YOLOv2 et la troisième version de YOLO, appelée YOLOv3, sortie en mai 2018.en suit la quatrième version de YOLOv4 a été publié l'année 2020, il existe plusieurs sous version développée comme Tiny YOLOv3, L'algorithme « YOLO » est un meilleur algorithme qui s'attaque au problème de la prédiction des boîtes englobantes précises tout en utilisant la technique de la fenêtre glissante à convolution. La structure de la méthode YOLO est illustrée dans la figure II-8. [12][13],

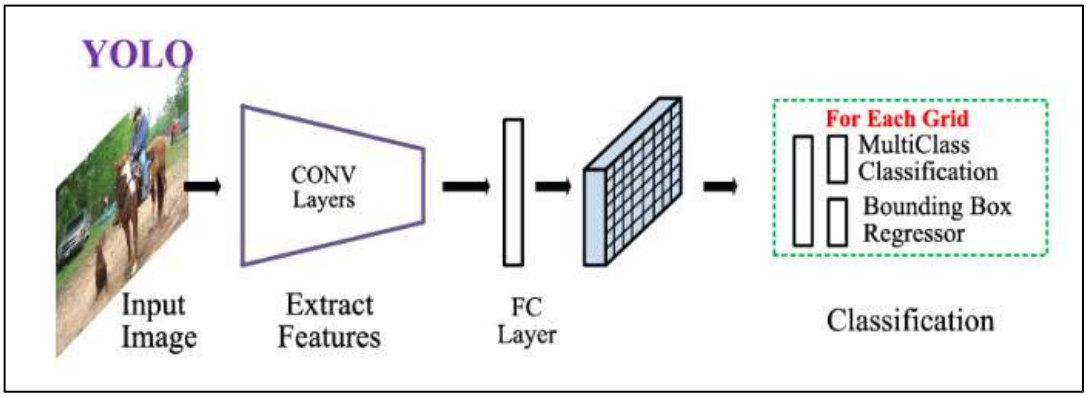


Figure II- 8 : illustration de la structure de la méthode YOLO

II.4.2 Architecture

Figure II.9 illustre l'architecteur générale de l'algorithme YOLO. Cette architecture prend une image en entrée et la redimensionne en 448*448 en gardant le même rapport d'aspect et en effectuant un remplissage. Cette image est ensuite passée dans le réseau CNN .

Ce modèle comporte 24 couches de convolution, 4 couches de max-pooling suivies de 2 couches entièrement connectées. Pour réduire le nombre de couches, une convolution 1*1 a été utilisée, suivie d'une convolution 3*3. Cette architecture utilise « Leaky ReLU » comme fonction d'activation dans toute l'architecture, sauf dans la couche de convolution où elle utilise une fonction d'activation linéaire. [14]

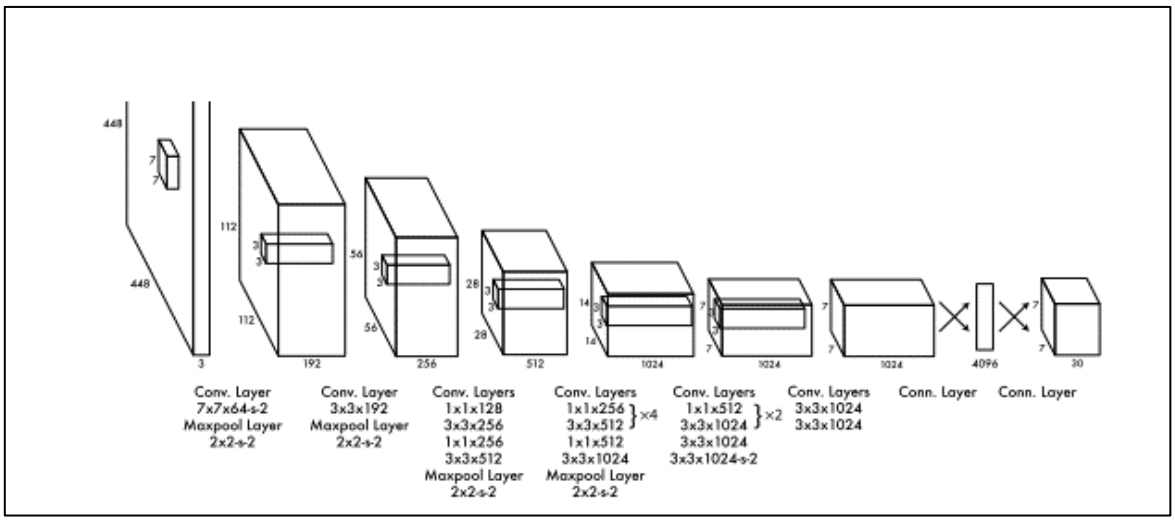


Figure II- 9: Architecteur générale de l'algorithme YOLO.

II.4.3 Apprentissage

Le modèle précèdent est entraîné sur la base données ImageNet-1000 (couvre 1000 classes d'objets et contient 1 281 167 images d'entraînement, 50 000 images de validation et 100 000 images de test) [15]. Le modèle a été entraîné pendant une semaine et a atteint une précision de 88 % lors de la validation d'ImageNet 2012, ce qui est comparable à GoogLeNet (lauréat 2014 de l'ILSVRC), le modèle de pointe à ce moment-là [14]..

II.4.4 Detection

L'architecture de l'algorithme YOLO divise l'image en une grille de taille S*S(.Figure II-10) Si le centre de la boîte de délimitation de l'objet se trouve dans cette grille, alors cette grille est responsable de la détection de cet objet. Chaque grille prévoit des cases limitatives avec leur score de confiance. Chaque score de confiance montre à quel point il est exact que le bounding qui prédit contient un objet et comment précis il prédit les coordonnées du box de délimitation . [16]



Figure II- 10 : Image YOLO (divisée en grille S*S)

Au moment du test, nous multiplions les probabilités de classe conditionnelles et les prédictions de confiance de la case individuelle. Nous définissons notre côté de confiance.

III.1. P_r (objet)* $CIoP_v$

P_r(objet) : les probabilités de classe conditionnelles

$CIoP_v$: prédictions de confiance de la case individuelle

Notez que le score de confiance doit être 0 lorsqu'il n'y a aucun objet dans la grille. S'il y a un objet présent dans l'image, le score de confiance doit être égal à IoU entre la vérité du sol et les boîtes prédites. Chaque case limitative comprend 5 prédictions : (x, y, w, h) et le score de confiance. Les coordonnées (x, y) représentent le centre de la boîte (Figure II- 11)par rapport aux limites de la cellule de la grille. Les coordonnées h, w représentent la hauteur, la largeur de la boîte de délimitation par rapport à (x, y). Le score de confiance représente la présence d'un objet dans la zone limitative.

Il en résulte une combinaison de boîtes limitatives de chaque grille comme ceci (Figure II- 11).

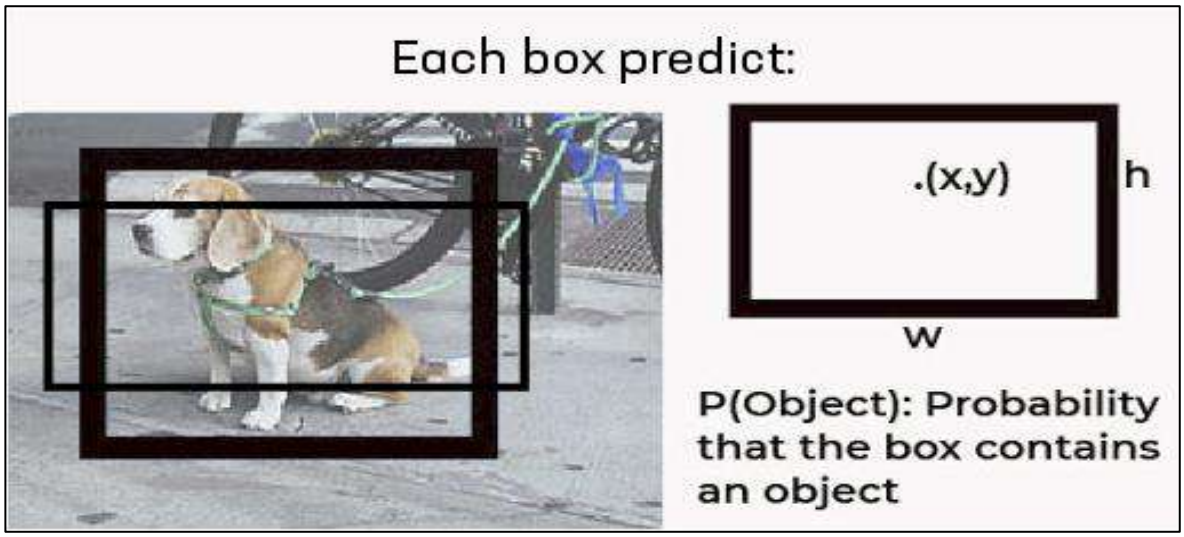


Figure II- 11: YOLO boîte de délimitation à grille unique.

Chaque grille prédit également la probabilité de classe conditionnelle C, Pr (Class $_i$ | Objet) (Figure II- 12).

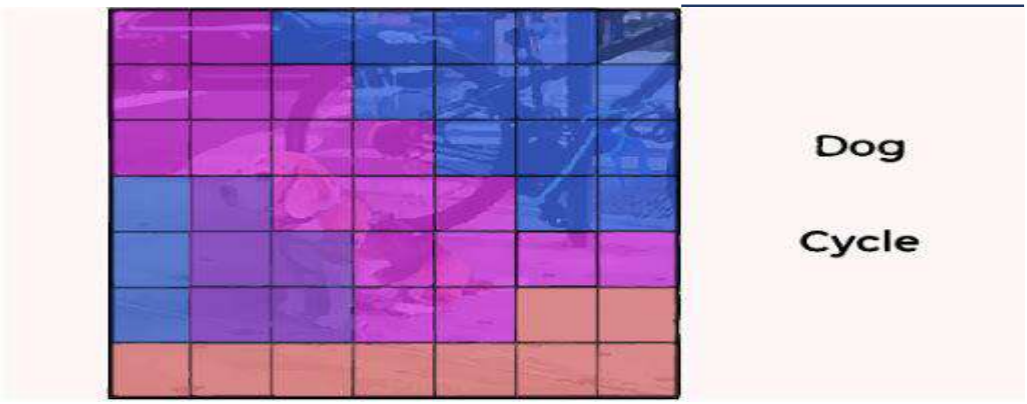


Figure II- 12 : Carte de probabilité conditionnelle YOLO

Cette probabilité était conditionnelle à la présence d'un objet dans la cellule de la grille. Peu importe le nombre de cases, chaque cellule de la grille ne prévoit qu'un seul ensemble de probabilités de classe. Ces prédictions sont encodées dans le tenseur 3D de taille S * S * (5*B +C).[16]

Maintenant, nous multiplions les probabilités de classe conditionnelles (Figure II-13) et les prédictions de confiance de la case individuelle,



Figure II- 13 : Combinaison de boîtes de délimitation YOLO

III.2. P_r (class$_i$ / objet) * P_r(objet)* $CIoP_{rv}$ = P_r (class$_i$)* $CIoP_{rv}$

$CIoP_{rv}$: prédictions de confiance de la case individuelle

P_r (class$_i$) : les prédictions de confiance de la case individuelle,

P_r(objet) : les probabilités de classe conditionnelles

P_r (class$_i$ / objet) : la probabilité de chaque classes

Ce qui nous donne des scores de confiance spécifiques à chaque classe pour chaque cas. Ces scores encodent à la fois la probabilité que cette classe apparaisse dans la boîte et la mesure dans laquelle la boîte prédite correspond à l'objet (Figure II- 14).

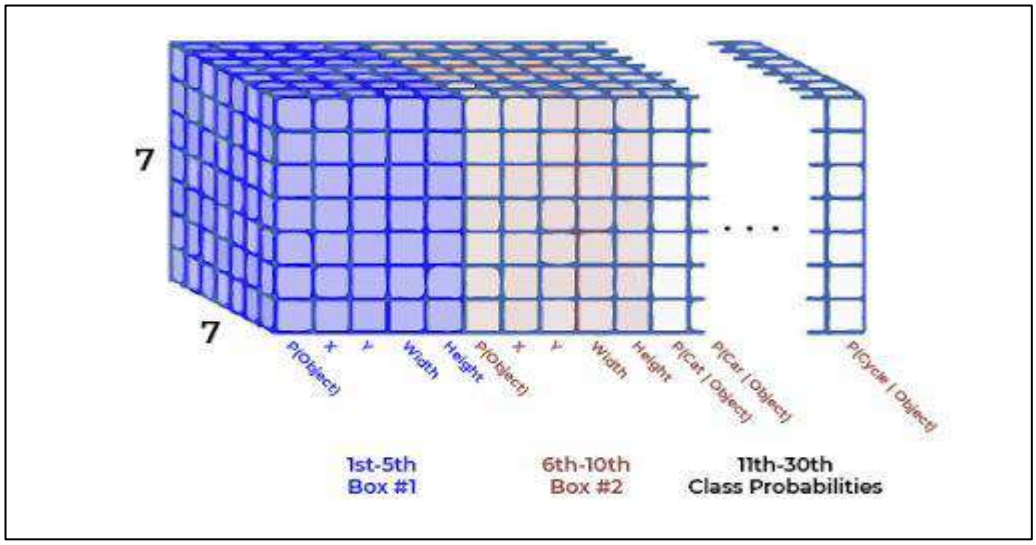


Figure II- 14 : Carte des fonctions de sortie YOLO.

Ensuite, après nous appliquons la suppression non mécanique pour prédire générer le résultat final de l'entrée (Figure II- 15)

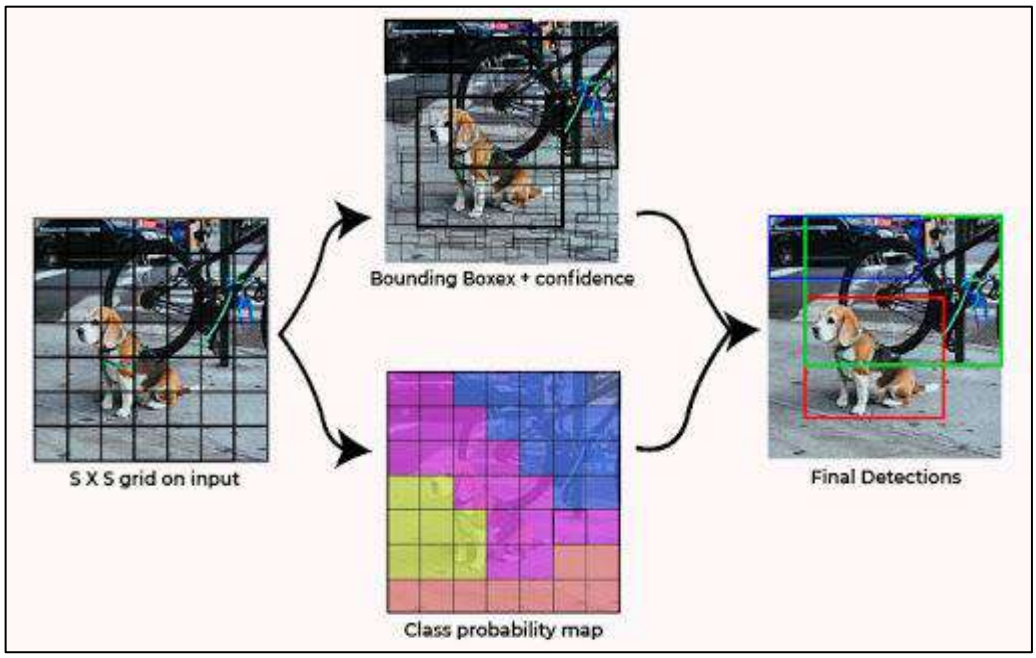


Figure II- 15 : Résultat du test YOLO.

YOLO est très rapide au moment du test car il n'utilise qu'une seule architecture CNN pour prédire les résultats et la classe est définie de telle manière qu'elle traite la classification comme un problème de régression.

Résultats :

Le YOLO simple a mAP de 63,4% lorsqu'il est formé sur VOC 2007 et 2012, le YOLO rapide qui est presque 3x plus rapide dans la génération de résultats a mAP de 52%. Ceci est inférieur au meilleur modèle Fast R-CNN atteint (71% mAP) et également au R-CNN atteint (66% mAP). Cependant, il bat d'autres détecteurs en temps réel tels que (DPMv5 33% mAP) sur la précision.[16]

II.4.5 Avantages et inconvinients de YOLO

- **Avantages**

Le problème avec la méthode R-CNN standard est qu'elle était douloureusement lente et qu'elle n'était pas un détecteur d'objets complet de bout en bout.

Girshick et al. A publier un deuxième article en 2015, intitulé Fast R-CNN. L'algorithme Fast R-CNN a apporté des améliorations considérables au R-CNN d'origine, notamment en augmentant la précision et en réduisant le temps nécessaire pour effectuer une passe avant.

Cependant, le modèle reposait toujours sur un algorithme de proposition de région externe.

Ce n'est que dans l'article de suivi 2015 de Girshick et al., Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, que les R-CNN sont devenus un véritable détecteur d'objets d'apprentissage profond de bout en bout en supprimant l'exigence de recherche sélective et en s'appuyant à la place sur un réseau de proposition de région (RPN) qui est premièrement entièrement convolutif et en deuxième peut prédire les cadres de délimitation des objets et les scores « d'objectivité » (c'est-à-dire un score quantifiant la probabilité qu'il s'agisse d'une région d'une image peut contenir une image). Les sorties des RPN sont ensuite transmises au composant R-CNN pour la classification finale et l'étiquetage.

Alors que les R-CNN ont tendance à être très précis, le plus gros problème avec la famille de réseaux R-CNN est leur vitesse - ils étaient incroyablement lents, n'obtenant que 5 FPS sur un GPU.

YOLO contient des ordres de grandeur plus rapide (fonctionne à 45 FPS sur GPU) que les autres algorithmes de détection d'objets. La limitation de l'algorithme YOLO est qu'il se débat avec de petits objets dans l'image, par exemple il pourrait avoir des difficultés à détecter un troupeau d'oiseaux. Cela est dû aux contraintes de l'algorithme.

L'avantage de l'algorithme YOLO est qu'il est très rapide car au moment de l'exécution, l'image n'a été exécutée qu'une seule fois sur CNN, ce qui rend YOLO beaucoup plus rapide que Faster R-CNN et peut être s'exécuter en temps réel. [12][13] [16]

❖ **Inconvinients**

Yolo à certain limitation [14].

- Un rappel comparativement faible et une erreur de localisation plus importante par rapport au Faster R_CNN .
- Difficulté à détecter les objets proches, car chaque grille ne peut proposer que 2 boîtes englobantes.
- Il a du mal à détecter les petits objets.

II.4.6 Implementation de l'algorithme YOLO

Il existe quelques implémentations différentes de l'algorithme YOLO sur le Web. Darknet est l'un de ces Framework de réseau neuronal open source, Darknet a été écrit en langage C et en technologie CUDA, ce qui le rend très rapide et permet de faire des calculs sur un GPU, ce qui est essentiel pour les prédictions en temps réel ou par un CPU pour l'analyse en temps décalée , Pour l'installation, nous pouvons utiliser la ligne de commande sur l'ordinateur ou en téléchargeant les fichiers nécessaires et en exécutant directement ceux que nous pouvons trouver sur le site officiel de darknet[17].

Après l'installation, nous pouvons utiliser un modèle pré-entraîné ou en construire un nouveau à partir de zéro. Par exemple, dans cette mémoire on va utiliser pour détecter des objets sur image à l'aide d'un modèle pré-entraîné sur le jeu de données COCO [17],

L'ensemble de données COCO se compose de 80 étiquettes (labels) comprenant : Gens, Vélos ; Voitures et camions ; Avions ; Panneaux d'arrêt et bornes d'incendie ; Animaux, y compris les chats, les chiens, les oiseaux, les chevaux, les vaches et les moutons, pour n'en nommer que quelques-uns ; Objets de cuisine et de salle à manger, tels que verres à vin, tasses, fourchettes, couteaux, cuillères, etc.

On pout entraîner YOLO à partir de zéro si on souhaite jouer avec différents régimes d'entraînement, hyper paramètres ou ensembles de données. Pour entraîner YOLO, on a besoin de toutes les données VOC de 2007 à 2012.nous trouve toutes les méthodes de entraînements par VOC sur darknet web officiel. [12][13] [16] [17]

II.4.7 Exemples de travail par YOLO sur Darknet

- **Détection à l'aide d'un model pré entrainé**

Dans cet exemple on va utiliser un système de détection des objets par l'algorithme YOLO a l'aide d'un model pré entrainé de la base Darknet (plateforme open source en C) [18].

La Figure II-16 au-dessus illustre la commande pour lancer le détecteur.

```
./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg
```

Figure II- 16 : commande d'exécuté pour une seule image

La sortie étant comme montre la Figure II-17 suivante :

```
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   416 x 416 x   3   ->   416 x 416 x  32  0.299 BFLOPs
    1 conv     64  3 x 3 / 2   416 x 416 x  32   ->   208 x 208 x  64  1.595 BFLOPs
    -------
  105 conv    255  1 x 1 / 1    52 x  52 x 256   ->    52 x  52 x 255  0.353 BFLOPs
  106 detection
truth_thresh: Using default '1.000000'
Loading weights from yolov3.weights...Done!
data/dog.jpg: Predicted in 0.029329 seconds.
dog: 99%
truck: 93%
bicycle: 99%
```

Figure II- 17 : la sortie de programme de premier essai

Darknet imprime les objets qu'il a détectés, sa confiance et le temps qu'il a fallu pour les trouver. Nous n'avons pas compilé Darknet avec OpenCV, il ne peut donc pas afficher les détections directement. Au lieu de cela, il les enregistre au format predictions.png. Nous pouvons l'ouvrir pour voir les objets détectés (Figure II- 18). Puisque nous utilisons Darknet sur le processeur, cela prend environ 6 à 12 secondes par image. Si nous utilisons la version GPU, ce serait beaucoup plus rapide.

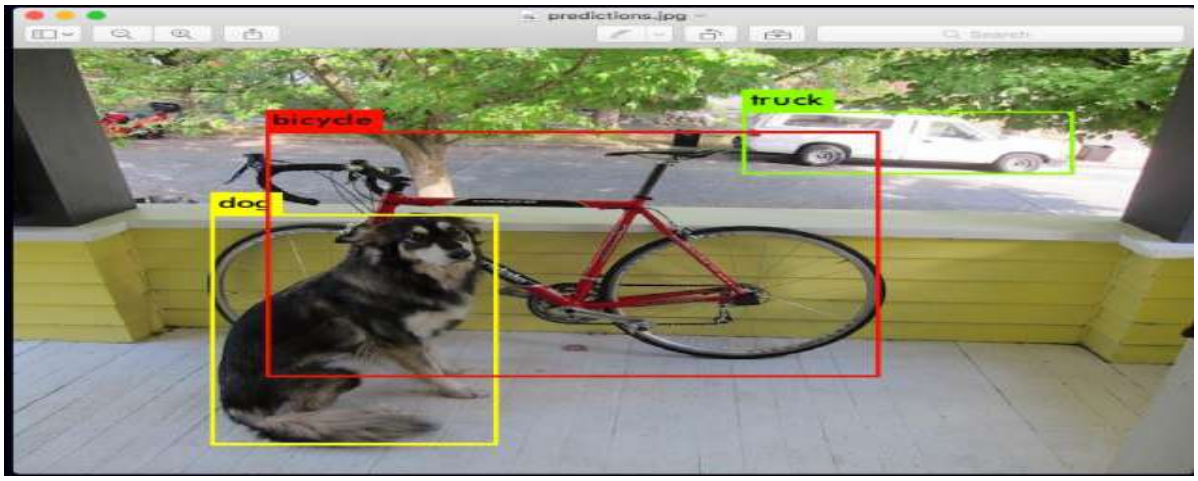


Figure II- 18: l'image obtenu en le premier essai avec les détections.

❖ **Images multiples**

Au lieu de fournir une image sur la ligne de commande, on la laisser vider pour essayer plusieurs images d'affilée. Au lieu de cela, nous verrons une invite lorsque la configuration et les poids seront chargés (Figures II-19) :

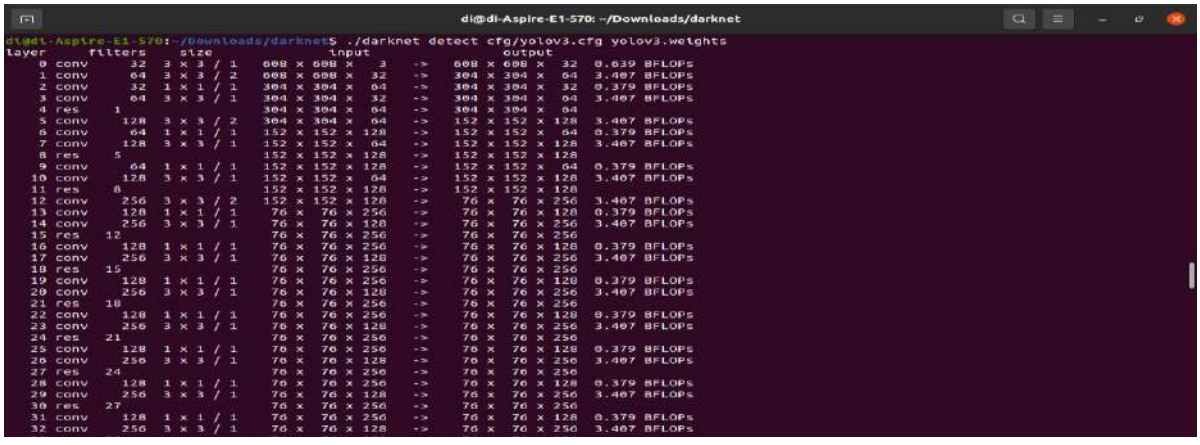


Figure II- 19: la commande pour traiter des images multiples.

On a utilisé quelque image à examiner a des classes différentes comme indique ces Figures (II-20.. II.-25).



Figure II- 23: image expérimentale de class type Person.



Figure II- 22: image expérimentale des classes type Person et Moto



Figure II- 21: image expérimentale des classes type oiseau et Ball



Figure II- 20: image expérimentale des classes type Person et Moto



Figure II- 25: image expérimentale des classes type Person et cheval



Figure II- 24: image expérimentale de class type voiture

On a obtenu des détections et des classifications exactes de tous les objets des images comme indique les figures II-26… II-31. Les sorties de system ont retenu des pourcentages des certitudes des classes obtenu comme illustre la Figure II-32, les certitudes résultantes sont excellentes, pour la class Person la certitude est de 100% pour la classe moto elle ne dépasse que 71 % car cette dernies ne pas bien positionnée a l'image. FigureII-22 pour le oiseux et la Ball il ont détecté d'une façon précise du pourcentage pour le premier de 88% et le deuxième de 95% .la figure II-24 la voiture a été classé d'une certitude de 100%.

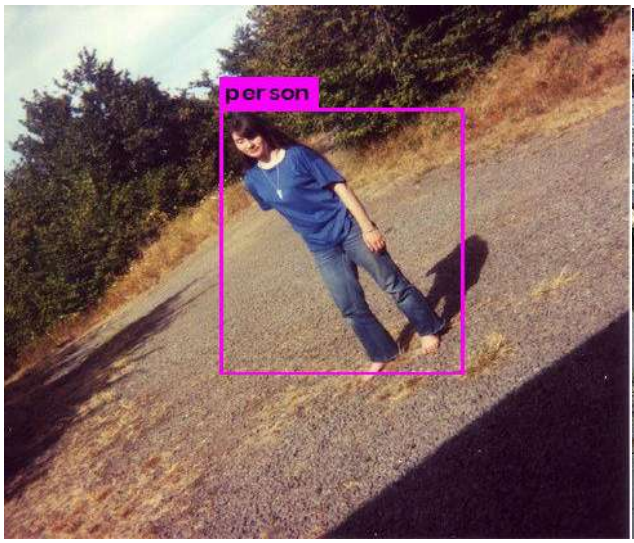


Figure II- 27: l'image obtenu contient de détection par l'algorithme d'une la classe Person.

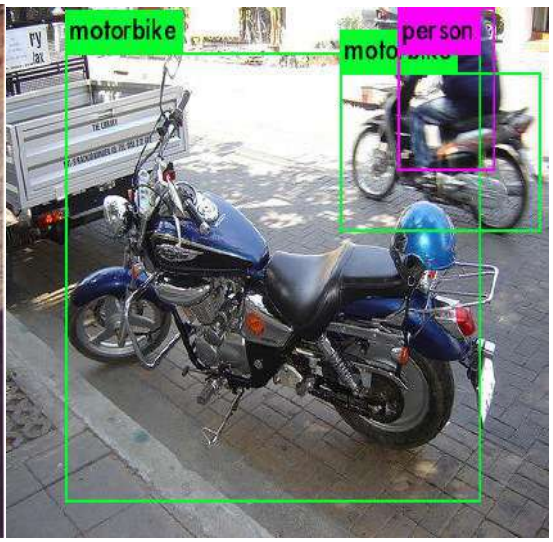


Figure II- 26: l'image obtenu contient des détections par l'algorithme des classes Moto et Person.

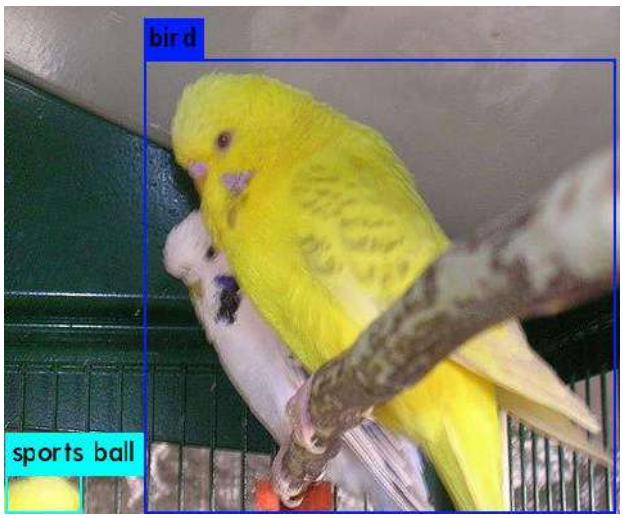


Figure II- 28: l'image obtenu contient des détections par l'algorithme des classes oiseau et Ball.

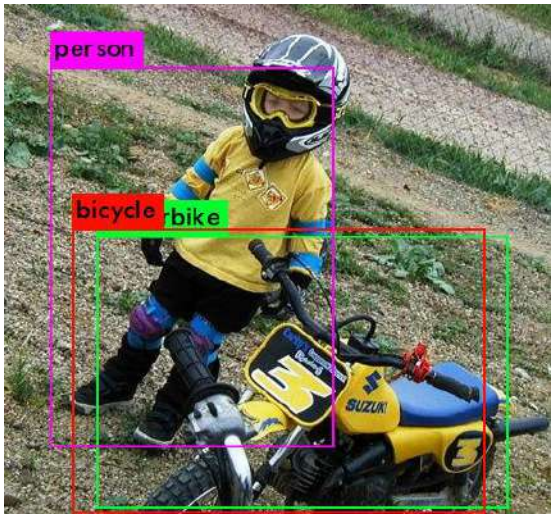


Figure II- 29: l'image obtenu contient des détections par l'algorithme des classes Person et Moto.

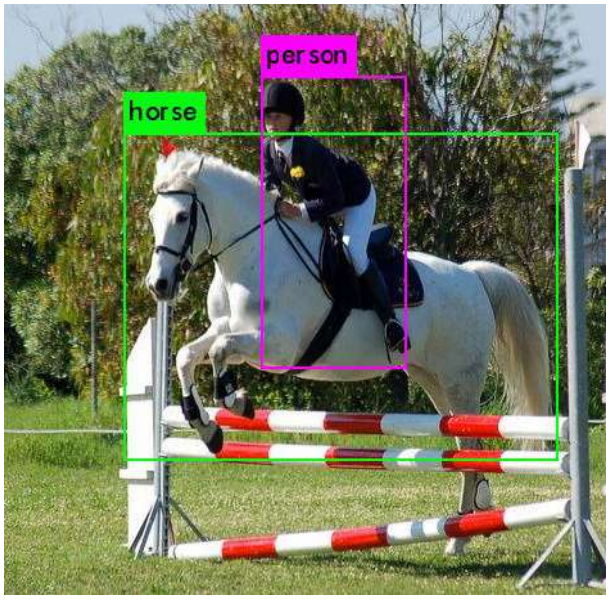


Figure II- 31: l'image obtenu contient des détections par l'algorithme des classes Person et cheval



Figure II- 30: l'image obtenu contient de détection par l'algorithme d'une la classe voiture.

```
Loading weights from yolov3.weights...Done!
Enter Image Path: figureII-20.jpg
figureII-20.jpg: Predicted in 39.940813 seconds.
person: 100%
Enter Image Path: figureII-21.jpg
figureII-21.jpg: Predicted in 39.561753 seconds.
motorbike: 100%
motorbike: 97%
Enter Image Path: figureII-22.jpg
figureII-22.jpg: Predicted in 39.379178 seconds.
bird: 85%
Enter Image Path: figureII-23.jpg
figureII-23.jpg: Predicted in 39.249864 seconds.
motorbike: 57%
bicycle: 71%
person: 100%
Enter Image Path: figureII-24.jpg
figureII-24.jpg: Predicted in 39.231463 seconds.
horse: 96%
person: 100%
Enter Image Path: figureII-25.jpg
figureII-25.jpg: Predicted in 39.321149 seconds.
car: 100%
Enter Image Path: figureII-26.jpg
Cannot load image "figureII-26.jpg"
STB Reason: can't fopen
di@di-Aspire-E1-570:~/Downloads/darknet$
```

Figure II- 32: la certitude des prédictions de l'algorithme

II.5 Conclusion

Dans ce chapitre une description sur l'apprentissage profond (Deep_learning) de la machine a été présenté ainsi que les réseaux neurone convolutif et ces demains d'application par ce dernier, on a ciblé à ces utilités sur notre objectif qui sur le traitement et la reconnaissance des images. À la suite on a défini quelques model pré entrainé traitent ce thème. Que nous utilise grâce à « Apprentissage par transfert » qui nous permet d'utiliser des modèles préformés d'autres personnes en apportant de petits changements.

Par la suite Nous avons choisi l'un de ces modèles après ce que nous avons évalué en termes de rendement, de vitesse et de précision, L'algorithme YOLO est un algorithme exécute la détection directement. L'image n'a été exécutée qu'une seule fois sur CNN. Il utilise la structure Régression / Classification, ce qui rend YOLO beaucoup plus simple et rapide qui peut être exécuté en temps réel.

Chapitre III

Implémentation

Chapitre III : Implémentation

III.1 Introduction

L'apprentissage profond est un algorithme d'apprentissage automatique populaire qui est largement utilisé dans de nombreux domaines de la vie quotidienne actuelle. Ses performances robustes et ses cadres et architectures prêts à l'emploi permettent à de nombreuses personnes de développer divers logiciels ou systèmes basés sur l'apprentissage profond pour soutenir les tâches etactivités humaines. La gestion du trafic est un domaine qui utilise l'apprentissage profond à plusieursfins. L'utilisation de caméras installées à certains endroits sur les routes permet de réaliser de nombreuses tâches telles que : comptage des véhicules, identification des véhicules, surveillance desinfractions au code de la route, surveillance de la vitesse des véhicules, etc.

Dans ce chapitre on va présenter en détail notre système de collection de données routière au niveau d'une intersection.

III.2 Cahier des charges et système proposé

Le but de ce projet est le développement d'un système intelligent qui permet de détecter, classer, et calculer le nombre des objets au niveau d'une intersection par classe (moto, car, camion, etc.), et aussi respecter les contraintes suivantes : la précision, le temps d'exécution, et le coût d'installation. Les données collectées exploiter pour faire des analyses, statistiques et surtout pour prise la bonne décision pour contrôler les feux de circulation au niveau de l'intersection.

Figure III.1 illustre l'architecture général de la solution proposée. L'idée est basée sur l'installation des caméras au niveau d'une intersection pour capturer les informations routières, Les données capturées (image/vidéo) sont transférées vers une unité de traitement. Au niveau de cette dernière, nous avons développé un système de détection et classification des objets basé sur l'algorithme YOLO qui permet de traiter la vidéo capturée, classer, calculer, et afficher les objets qu'il contient. Le système proposé permet de calculer le nombre des objets dans une zone spécifique et le débit dans une période de temps. Dans notre projet, les types des objets qui nous intéressent sont : véhicule, car, camion, moto, et bicycle. La figure III-2 illustre l'architecture interne du système proposé.

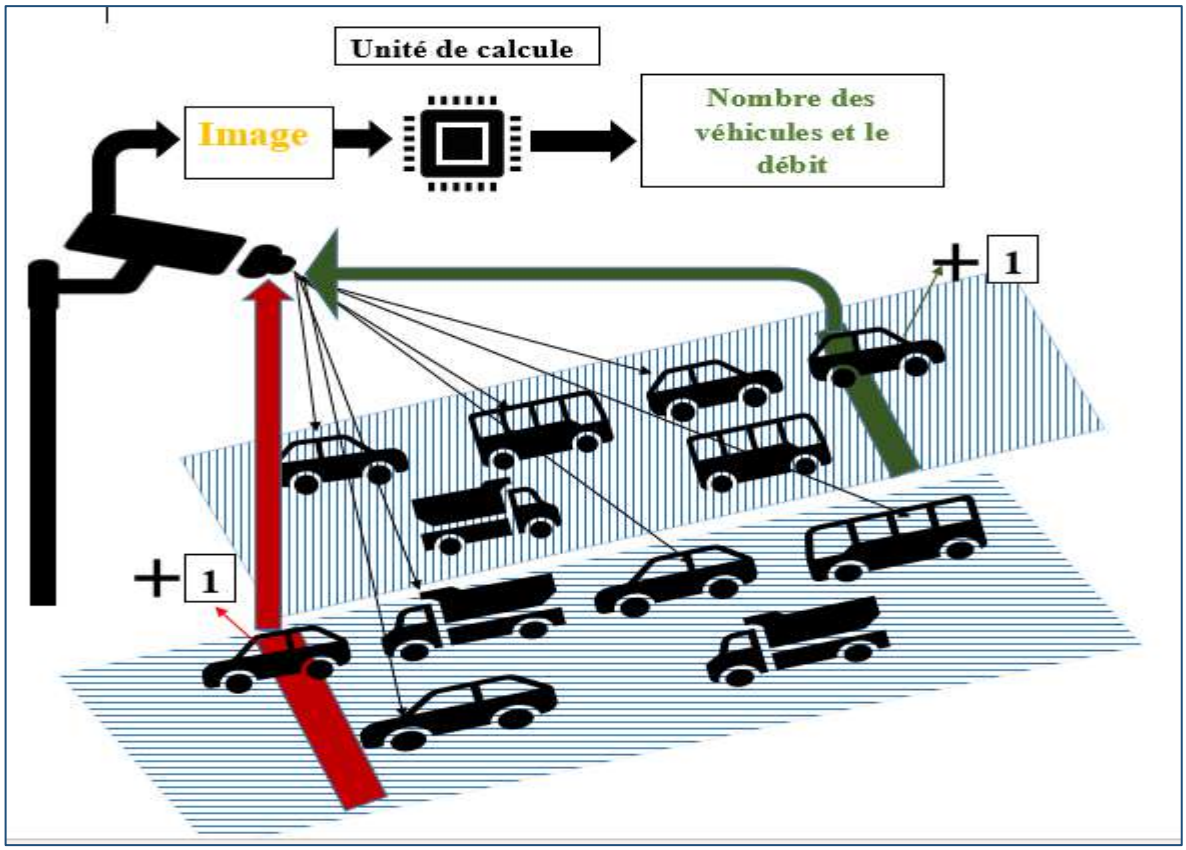


Figure III- 2: Architecture générale du système proposé.

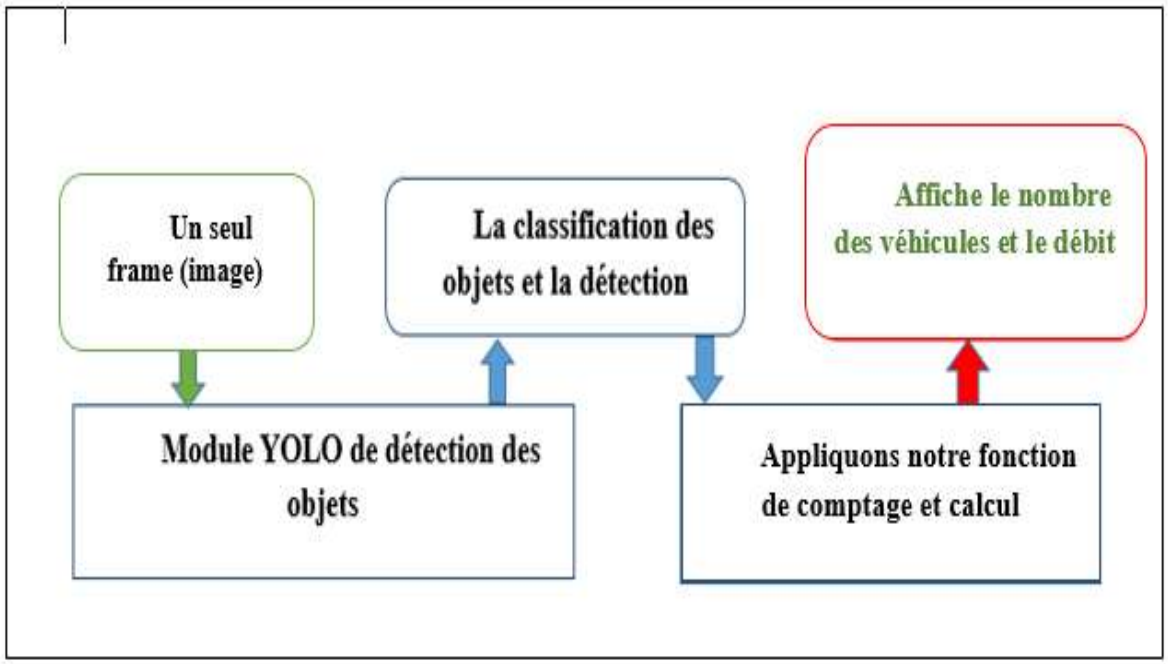


Figure III- 1: Architecture interne du système proposé.

III.3 Organigramme de fonctionnement

Figure III-3 illustre l'organigramme de fonctionnement de notre système proposé. Tout d'abord la caméra de surveillance prend des sections d'images de la route. Une fois l'image reçue, notre système commence la lecture de ces images une par une, avant qu'il calcule le nombre total d'images et le FPS (nombres des images par second). En absence d'images le système affiche un message d'avertissement, si non-il continue de calculer le nombre total des images et estime le temps nécessaire pour continue l'analyse total de la vidéo et afficher ces informations sur une fenêtre de commande. Le système après avoir pris ces images une par une, commence le traitement et l'analyse de chacune. Au début, nous prenons les dimensions de chaque image et initialisons la détection des objets qui nous intéressent avant de les localiser et les classer, si le système ne peut pas les calculer ou le degré de confiance n'est pas compatible, il affiche un message également. Sinon, il continue pour traiter les images à travers des fonctions pour calculer le nombre des véhicules dans une surface bien déterminée et estimer le débit, ensuite imprimer quelques formes sur les images. À la fin, ces informations sont affichées dans notre vidéo résultant et dans la fenêtre d'exécution.

III.4 Implémentation software

III.4.1 Environnement de travail

❖ Hardware utilisé

Dans notre projet, nous avons utilisé un PC avec les caractéristiques suivantes :

- Processeur : Intel ® core ™ I3 27500 2.60 GHz.
- Carte graphique : Intel ® HD Graphics 4000.
- Ram : 4 GB.
- Rom : 320 Gb HDD.

❖ Software utilisé

Ubuntu est le système d'exploitation le plus adapté, il permettra d'éviter les problèmes d'installation du langage python et ses bibliothèques. Pour cette raison il a été utilisé dans notre projet.

La programmation dans notre projet a été basée sur langage python dans un environnement de travail virtuel(virtualenv).

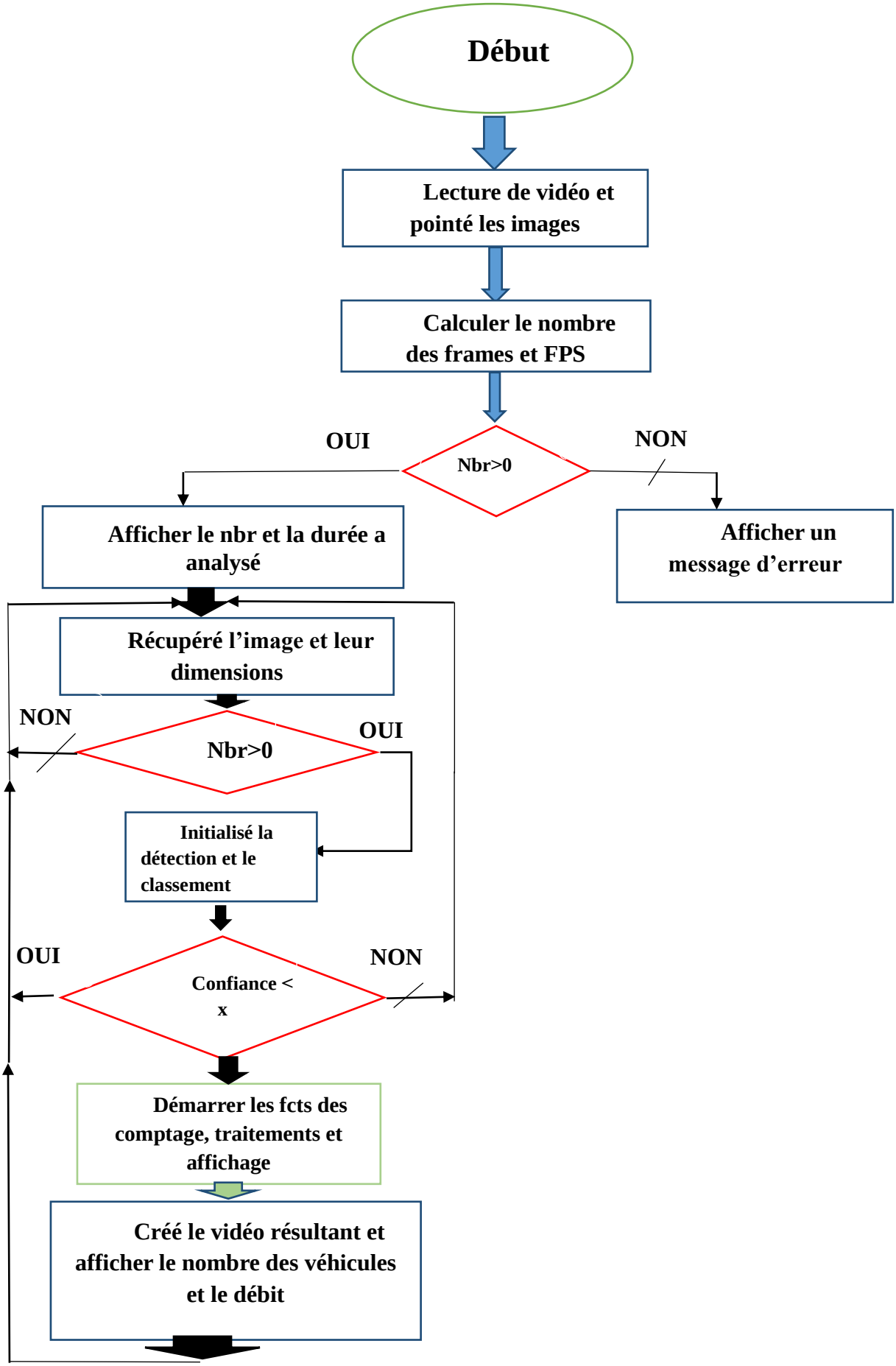


Figure III- 3: Organigramme de fonctionnement

1) Installation du langage Python3

Au début, le langage python3 été installé dans le système d'exploitation par la fenêtre terminale pour permettre l'exécution de toutes les commandes d'implémentation.

Étape 1 : Mettre à jour et actualiser les listes de référentiels par la commande :

« sudo apt update »

Étape 2 : installer le logiciel de support

Le package software-properties-commun nous donne un meilleur contrôle sur votre gestionnaire de packages en nous permettant d'ajouter des référentiels PPA (Personal Package Archive). Installez le logiciel de support avec la commande :

« sudo apt install software-properties-common »

Étape 3 : Ajouter le PPA Deadsnakes

Deadsnakes est un PPA avec des versions plus récentes que les référentiels Ubuntu par défaut. Ajoutez le PPA en saisissant ce qui suit :

« sudo add-apt-repository ppa:deadsnakes/ppa »

Étape 4 : Nous avons installé Python 3, et lancer l'installation de Python 3.8 avec la commande :

« sudo apt install python3.8 »

Pour vérifier la version Python installée la commande « Python —version » peut-être utiliser.

2) Création d'Environnement Virtuel de travail

Après l'installation de langage python3 sur terminal, on a préféré utiliser un environnement virtuel spécial de travail pour installer nos bibliothèques nécessaires pour notre programme. Dans notre cas, on considère plusieurs objets, il est important de gérer plusieurs environnements virtuels. Les environnements virtuels (virtualenv) créent des espaces isolés et cohérents sur le système, dédiés à des projets spécifiques, pour soit : aucun environnement n'interagit avec un autre, chacun fonctionne comme une unité indépendante et sa destruction n'endommage pas les autres projets [19].

La création d'un environnement virtuel doit être effectuée principalement via le terminal du système d'exploitation. Pour créer ces environnements, nous avons installé un module appelé venv, qui est présent dans la bibliothèque Python. Ensuite, procédez à l'installation de venv avec la commande suivante :

« sudo apt-get install -y python3-venv »

Il est maintenant le temps de créer de nouveaux environnements virtuels pour notre projet. Pour commencer, utilisez la commande mkdira été utilisé pour créer un nouveau répertoire à remplir avec notre environnement. Ensuite, déplacez-nous dedans avec la commande cd, comme dans l'exemple suivant :

« mkdir directory_env ».

« cd directory_env ».

Une fois dans le répertoire où créer les nouveaux environnements, créez le premier à l'aide du module venv. Pour ce faire, on a suivi la commande :

« python -m venv ».

La sortie soit :

« environnement1 ».

L'environnement a été créé avec sucée pour le vérifier on peut utiliser la commande :

« ls ». qui permet d'afficher tous le contenu de notre répertoire.

Cette commande doit affichiez toutes les contenue de votre répertoire.

3) Bibliothèques utilisées

Après la création de l'environnement virtuel, nous avons installé quelques bibliothèques qui permet de faire des fonctions spécifiques.

Si on parle de la classification des objets et le traitement d'image ça nous intéresse à la bibliothèque « OpenCV ».

- OpenCV est une énorme bibliothèque open source pour la vision par ordinateur, l'apprentissage automatique et le traitement d'images. OpenCV prend en charge une grande variété de langages de programmation tels que Python, C++, Java, etc. Il peut traiter des images et des vidéos pour identifier des objets, des visages ou

même l'écriture d'un humain. Lorsqu'il est intégré à diversesbibliothèques, comme par exemple Numpy une bibliothèque hautement optimisée pour les opérationsnumériques, le nombre d'armes augmente dans votre Arsenal, c'est-à-dire que toutes les opérations que l'on peut faire dans Numpy peuvent être combinées avec OpenCV. Pour l'installation on a fait par la commande toujours sur terminal : « pip install opencv », âpre on va mis à jour par la commande

: « pip upgrade opencv ». [20]

- Imutils package : Ce package comprend une série de fonctions de commodité OpenCV et qui effectuent des tâches de base telles que la traduction, la rotation, le redimensionnement et la squelettisation. [20]
- NumPy est un package de traitement de tableau à usage général. Il fournit un objet tableau multidimensionnel hautes performances et des outils pour travailler avec ces tableaux. [20]

C'est le package fondamental pour le calcul scientifique avec Python. Il contient diverses fonctionnalités, dont les plus importantes : Un puissant objet de tableau à N dimensions, Fonctions sophistiquées (diffusion), Outils d'intégration de code C/C++ et Fortran, Fonctionnalités utiles d'algèbre linéaire, de transformée de Fourier et de nombres aléatoires.

Outre ses utilisations scientifiques évidentes, NumPy peut également être utilisé comme un conteneur multidimensionnel efficace de données génériques.

Les types de données arbitraires peuvent être définis à l'aide de Numpy, ce qui permet à NumPy de s'intégrer de manière transparente et rapide à une grande variété de bases de données, l'installation de cette bibliothèque est aussi par la commande :pip install numpy.

- Collections : Ce module implémente des types de données de conteneurs spécialisés qui apportent des alternatives aux conteneurs natifs de Python plus généraux dict, list, set et tuple.

La commande d'installation sur terminal : « pip install collections »

- Math : Ce module fournit l'accès aux fonctions mathématiques définies par la norme C.

La commande d'installation sur terminal : « pip install math »

- Os : Ce module fournit une façon portable d'utiliser les fonctionnalités dépendantes du système d'exploitation. Si nous voulons simplement lire ou écrire un fichier.

La commande d'installation sur terminal : « pip install os ». [20]

Il y’a d’autre bibliothèques qu’on va utiliser en va affirmer ces fonctionnalités dans la description détaillée de notre programme.

III.4.2 Programmation

III.4.2.1 Vidéo utilisée

Les caractéristiques de la vidéo choisi pour notre projet sont données par la Figure III.4. On a pri une route à Montréal (Figure III-5) sur un pont de haut débit de trafic.

Vidéo	
Longueur	00:00:24
Largeur de trame	1920
Hauteur de trame	1080
Débit de données	7979 Kbits/s
Débit total (en bits)	8059 Kbits/s
Fréquence d'images	23 trames/s
Audio	
Vitesse de transmission	79 Kbits/s
Chaînes	2 (stéréo)
Taux d'échantillonnage audio	44 kHz

Figure III- 4: Caractéristiques de la Vidéo.



Figure III- 5: Capture de la vidéo utilisée.

III.4.2.2 Initialisation

1) La première étape consiste à importer les bibliothèques et ces instructions de commande (Figure III-6). Ensuite on met un analyseur pour les options de ligne de commande, les arguments et les sous-commandes (argparse).

```
 5 from imutils.video import VideoStream
 6 from centroidtracker import CentroidTracker
 7 import matplotlib.pyplot as plt
 8 import numpy as np
 9 import argparse
10 import imutils
11 import time
12 import cv2
13 import glob
14 import collections
15 import math
16 import os
```

Figure III- 6: Les Commande de L'importation des bibliothèques.

- Le argparse fait partie de la bibliothèque « argparse » facilite l'écriture a interfaces de ligne de commande conviviales. Le programme définit les arguments dont il a besoin et argparse trouvera par les commandes de l'utilisateur a l'interface de système d'exploitation et comment les analyser à partir des commandes. Le argparse module génère également automatiquement des messages d'aide et d'utilisation et émet des erreurs lorsque les utilisateurs donnent au programme des arguments non valides.[21]

Pour initialiser cette commande on va la définir en premier et l'intitulé par exemple :

« ap = argparse.argumentparser () »

L'instruction « argparse.Argumentparser » est nommé « ap ».

Pour les autres lignes on a créé un mode de lecture des arguments et leur ajoutée (linge19, Figure III.7).

```
18 ap = argparse.ArgumentParser()
19 ap.add_argument("-i", "--input", required=True,
20         help="path to input video")
21 ap.add_argument("-o", "--output", required=True,
22         help="path to output video")
23 ap.add_argument("-y", "--yolo", required=True,
24         help="base path to YOLO directory")
25 ap.add_argument("-c", "--confidence", type=float, default=0.5,
26         help="minimum probability to filter weak detections")
27 ap.add_argument("-t", "--threshold", type=float, default=0.3,
28         help="threshold when applyong non-maxima suppression")
29 args = vars(ap.parse_args())
```

Figure III- 7 : Les Commandes d'argumentation.

On a créé un espace pour l'utilisateur de notre algorithme pour qu'il annonce un argument par choix à travers l'interface extérieur (terminal). Le contenu des parentaise décrite Comment allons-nous séparer et déclaré l'argument, aussi le type et des valeurs si nécessaire. Par exemple de notre programme pour déclarer l'emplacement de vidéo d'entré (Figure III-4) ou la caméra on commence par « -i » ou « --input » âpre en décrit

l'emplacement (Figure III-8) .si en décrit la commande « required=true » (ligne 19) si l'argument n'est pas annoncé à la commande d'exécution cela nous mène à une erreur d'exécution. On utilise L'instruction « help="message" » pour qu'il affiche un message voyant si on a une erreur de ce dernier [21].

Pour la commande 25 et 27 (Figure III-9) on a déclaré le type d'argument et la valeur que le programme va le prendra par défaut si nous décidons de ne pas le mettre dans notre commande d'exécution.



Figure III- 8: Emplacement D'Argument.

```
python3 yolo_videogh.py -i /home/di/Downloads/789.mp4
```

Figure III- 9: Exemple De Déclaration De Notre Argument.

2) Ensuite, nous occuperons de certaines tâches spécifiques à la vidéo :

Dans le bloc de la (Figure III-10), nous ouvrons un pointeur de fichier vers le fichier vidéo pour lire les images dans la boucle à venir (ligne 94). Après, on a initialisé notre vidéo écrivain (ligne 96) et les dimensions du cadre (ligne 99).

```
92 # initialize the video stream, pointer to output video file, and
93 # frame dimensions
94 vs = cv2.VideoCapture(args["input"])
95 #object_detector = cv2.createBackgroundSubtractorMOG2()
96 writer = None
97 ct = CentroidTracker()
98 (W, H) = (None, None)
```

Figure III- 10 : Block de Pointage et la lecture de la Vidéo.

(Si la caméra du pc est utilisée, il faut remplacer le contenu des parentaise sur la ligne 94par « 0 »).

Pour la ligne 95 on a préparé une instruction du masquage et filtrage en cas de problème de vision par camera ou la vidéo est capturé la nuit l'instruction « cv2.createbackgroundSubstractorMOG2() » est très utile pour la détection des objets en mouvement dans les endroits sombre.

3) Dans la Figure III- 8, nous avons simplement :
   - Initialiser notre vidéo écrivain si nécessaire (Lignes 240-249). L'écrivain sera initialisé à la première itération de la boucle.

- Pointeurs de nettoyage et de libération (lignes 252 et 253).

```
240         if writer is None:
241                 # initialize our video writer
242                 fourcc = cv2.VideoWriter_fourcc(*"MJPG")
243                 writer = cv2.VideoWriter(args["output"], fourcc, 30,
244                         (frame.shape[1], frame.shape[0]), True)
245
246
247
248         # write the output frame to disk
249         writer.write(frame)
250 # release the file pointers
251 print("[INFO] cleaning up...")
252 writer.release()
253 vs.release()
```

Figure III- 11: Block de pré-détection.

Le vidéo résultant soit le même que le vidéo d'entré au système (Figure III- 12).



Figure III- 12 : le vidéo résultant sans aucun traitement

❖ Chargement des modèles de détection et classification pré entrainé

4) Pour faire classer et détecté nos objets, il faut initialement charger les classes de la base COCO et initialisé les couleurs représentent chaque classe. Ensuite démarrer le détecteur YOLO entrainé seulement sur cette base (Figure III- 13).

```
72 # load the COCO class labels our YOLO model was trained on
73 labelsPath = os.path.sep.join([args["yolo"], "coco.names"])
74 LABELS = open(labelsPath).read().strip().split("\n")
75
76 # initialize a list of colors to represent each possible class label
77 np.random.seed(42)
78 COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
79         dtype="uint8")
80
81 # derive the paths to the YOLO weights and model configuration
82 weightsPath = os.path.sep.join([args["yolo"], "yolov3.weights"])
83 configPath = os.path.sep.join([args["yolo"], "yolov3.cfg"])
84
85 # load our YOLO object detector trained on COCO dataset (80 classes)
86 # and determine only the *output* layer names that we need from YOLO
87 print("[INFO] loading YOLO from disk...")
88 net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)
89 ln = net.getLayerNames()
90 ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
```

Figure III- 13: chargement des classes de model COCO.

III.4.2.3 Initialisation de l'analyse et le traitement

5) Nous sommes maintenant prêts à commencer à traiter les images une par une :
   - Nous définissons une boucle (Figure III- 14), puis nous récupérons notre première image (Ligne 122).

```
119 while True:
120         # read the next frame from the file
121
122         (grabbed,frame) = vs.read()
```

Figure III- 14: Boucle d'extraction des Images (Frames).

   - Ensuite, nous récupérons les dimensions du cadre si elles ne l'ont pas encore été (lignes 136 et 137 (Figure III- 15))

```
129
130         # if the frame was not grabbed, then we have reached the end
131         # of the stream
132         if not grabbed:
133                 break
134
135         # if the frame dimensions are empty, grab them
136         if W is None or H is None:
137                 (H, W) = frame.shape[:2]
```

Figure III- 15: Arrêt d'attraction des Images et le dimensionnement.

6) Après, effectuons une passe avant de YOLO, en utilisant notre Cadre en entrée :

A la Figure III- 16, nous construisons une hypothèse (les détections estimative) et le transmettre à travers le réseau, on obtiendra des prédictions. On a bouclé l'opération de passe avant avec des horodatages afin que nous puissions calculer le temps écoulé pour faire des prédictions sur une image - cela nous aidera à estimer le temps nécessaire pour traiter l'intégralité de la vidéo.

```
144         blob = cv2.dnn.blobFromImage(frame , 1 / 255.0, (416, 416),
145                 swapRB=True, crop=False)
146         net.setInput(blob)
147         start = time.time()
148         layerOutputs = net.forward(ln)
149         end = time.time()
150
151         # initialize our lists of detected bounding boxes, confidences,
152         # and class IDs, respectively
153         boxes = []
154         confidences = []
155         classIDs = []
156         rects = []
```

Figure III- 16: Initialisation des cadres pour les objets Classés

Nous allons ensuite initialiser les quatre mêmes listes (lignes 153 vers 156) que nous utiliser dans la prédiction de la détection : des boites, confidences, et ID de classe, et les rectangles.

7) Extraire les détections (Figure III- 17) :

- Boucle sur les couches de sortie et les détections (lignes 159-161).
- Extraire l'identifiant de classe et filtrer les prédictions de faible confidence pour éliminer (lignes164-170).
- Calculer les coordonnées du cadre de délimitation pour encadrer les objets (Lignes 176-183).
- Mettre à jour nos listes respectives et charger les détection (Lignes 186-189).

```
158         # loop over each of the layer outputs
159         for output in layerOutputs:
160                 # loop over each of the detections
161                 for detection in output:
162                         # extract the class ID and confidence (i.e., probability)
163                         # of the current object detection
164                         scores = detection[5:]
165                         classID = np.argmax(scores)
166                         confidence = scores[classID]
167
168                         # filter out weak predictions by ensuring the detected
169                         # probability is greater than the minimum probability
170                         if confidence > args["confidence"]:
171                                 # scale the bounding box coordinates back relative to
172                                 # the size of the image, keeping in mind that YOLO
173                                 # actually returns the center (x, y)-coordinates of
174                                 # the bounding box followed by the boxes' width and
175                                 # height
176                                 box = detection[0:4] * np.array([W, H, W, H])
177                                 (centerX, centerY, width, height) = box.astype("int")
178                                 rects.append(box.astype("int"))
179
180                                 # use the center (x, y)-coordinates to derive the top
181                                 # and and left corner of the bounding box
182                                 x = int(centerX - (width / 2))
183                                 y = int(centerY - (height / 2))
184                                 # update our list of bounding box coordinates,
185                                 # confidences, and class IDs
186                                 boxes.append([x, y, int(width), int(height)])
187                                 confidences.append(float(confidence))
188                                 classIDs.append(classID)
189                                 classname.append(LABELS[classID])
```

Figure III- 17 : Filtrage Et Extraction Des Cordonnées Des Identifiant.

8) Ensuite (Figure III-18), nous appliquons la suppression non maximale et commencerons à annoter le cadre .

- Appliquer NMS (Suppression non maximale) en utilisant la fonction « cv2.dnn.NMSBoxes » sur la ligne 182 et 183 pour supprimer les cadres de délimitation faibles et qui se chevauchent.

```
180         # apply non-maxima suppression to suppress weak, overlapping
181         # bounding boxes
182         idxs = cv2.dnn.NMSBoxes(boxes, confidences, args["confidence"],
183                 args["threshold"])
184
```

Figure III- 18: La Commande De Suppression Non Maximale.

- Boucle sur le « idx » (Figure III-19) calculé par NMS et dessinez les cadres de délimitation et les étiquettes correspondants (lignes 186-195). La ligne 190 et

191 fait le calcul des dimensions des rectangle encadré les détections, la fonction « color » fait colorer ces cadres a des couleurs correspondante à chaque classe.

```
185         # ensure at least one detection exists
186         if len(idxs) > 0:
187                 # loop over the indexes we are keeping
188                 for i in idxs.flatten():
189                         # extract the bounding box coordinates
190                         (x, y) = (boxes[i][0], boxes[i][1])
191                         (w, h) = (boxes[i][2], boxes[i][3])
192
193                         # draw a bounding box rectangle and label on fthe frame
194                         color = [int(c) for c in COLORS[classIDs[i]]]
195                         cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
```

Figure III- 19 : Cadrage Des Identifient.

On va utiliser deux instructions, une pour imprimer une forme ligne sur notre sortie vidéo.la deuxième pour insérer des formats rectangles sur les objets détectés.

- ❖ Format ligne : Syntaxe : « cv2.line(image, point start, point de fin, couleur, épaisseur). »
- ❖ Format rectangle : syntaxe « cv2rectangle(image, point référence ,longueur et hauteur d'objet, colleur des ligne rectangle , épaisseur) »

Quelques Paramètres :

- Image : C'est l'image sur laquelle la ligne et le rectangle doivent être tracée.
- Point start : Il s'agit des coordonnées de départ de la ligne. Les coordonnées sont représentées par des lignes de deux valeurs (valeur de coordonnées X, valeur de coordonnées Y).
- Point de fin : Il s'agit des coordonnées de fin de ligne. Les coordonnées sont représentées par des lignes de deux valeurs (valeur de coordonnées X, valeur de coordonnées Y).
- Couleur : C'est la couleur des lignes à dessiner. Pour BGR (bleue vert rouge), nous passons un tuple. Par exemples : (255, 0, 0) pour la couleur bleue. (0,255, 0) pour la couleur rouge.
- Épaisseur : C'est l'épaisseur des lignes en px.

On essaye en premier d'appliquer l'instruction pour cadrer les objets dans le but de les détecter. Tous les objets utilisant la route sont bien détectés comme l'illustre la figure III- 20, en va voir l'intitulation de chaque classe pour définir le classement de ces derniers et même la confidence de chaque prédiction.



Figure III- 20:cadrages des véhicules

On utilise la fonction « cv2.putText » pour imprimer des formates texte (string) sur l'image. Notre besoin est de voir le classement de chaque objet dans notre vidéo résultante.

- ❖ Syntaxe : « cv2.putText(image, texte, position, format, transparence, couleur, épaisseur), »
- ❖ Paramètre :
- Image : c'est l'image sur laquelle le texte doit être imprimer.
- Format : c'est la forme de texte
- Texte : le texte a imprimé, il doit être déclaré avant sous la forme (string).

La résultat est indiqué par la figure III-21.



Figure III- 21: affichage de classement des véhicules

Le classement des objets a été bien définie l'intitulation pour chaque classe au dessus de chaque objet.

III.4.2.4 Application des fonctions de comptage, de traitement et l'affichage des information extrait

❖ **Imprimé des lignes sur les images traitées**

On a utilisé la fonction à dessiner des lignes pour faire afficher des lignes a des dimensions spéciales. On a essayé l'instruction pour dessiner des lignes à l'aide d'un petit programme comme suit (Figure III-22) :

```
import cv2


# define a video capture object
vid = cv2.VideoCapture(0)

while(True):

    # Capture the video frame
    # by frame
    ret, frame = vid.read()
    cv2.line(frame , (400, 100), (760, 600),(255,0,0), 3)
    cv2.line(frame , (200, 300), (1150, 300),(0,0,255), 3)
    cv2.rectangle(frame, (211, 44), (350, 280), (0,255,0), 2)
    text = "diaoui"
    cv2.putText(frame, text, (211, 40),cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0,0,255), 2)
    # Display the resulting frame
    cv2.imshow('frame', frame)

    # the 'q' button is set as the
    # quitting button you may use any
    # desired button of your choice
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# After the loop release the cap object
vid.release()
# Destroy all the windows
cv2.destroyAllWindows()
```

Figure III- 22: un programme pour l'essai

Nous obtenons des lignes a des dimensions désirés (Figure III- 23).

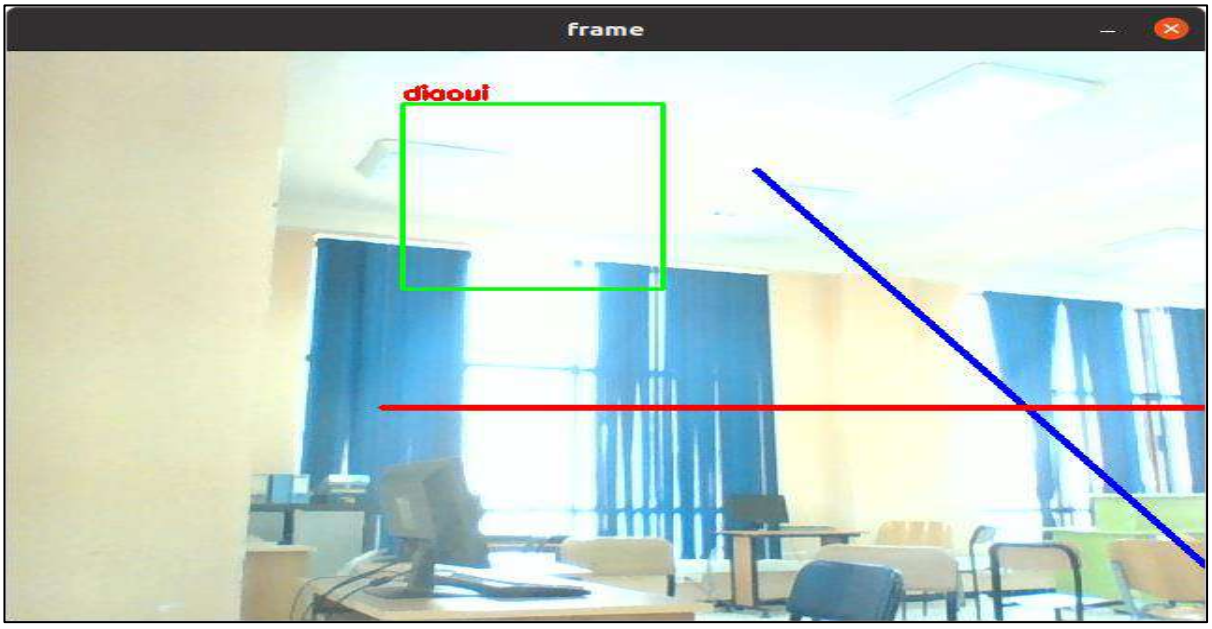


Figure III- 23 : expérience des fonction cv2[format].

On va appliquer les instruction de la ligne 199 jusqu'à la ligne 206 (Figure III-24), on affiche des lignes de dimensions spéciales, et on déclare à l'avance les constantes des points (Figure III-25).

```
30 EntranceCounter = 0
31 ExitCounter = 0
32 lineCount = 700
33 linecount2=300
34 pc1=(600,300)
35 pc2=(790,300)
36 p1=(480, lineCount)
37 p2=(770,lineCount)
38 p3=(950,linecount2)
39 p4=(1120,linecount2)
40 line1=(p1,p2)
41 linecolor=(pc1,p1)
42 linecolor2=(pc2,p2)
```

Figure III- 24 : déclaration des constants des point référence.

```
199                     Cx=int(x + w / 2 )
200                     Cy=int( y + h / 2 )
201                     ObjectCentroid = (Cx,Cy)
202                     cv2.circle(frame,ObjectCentroid,1,(0,0,255),-1)
203                     cv2.line(frame,p1,p2,(0,255,255), 2)
204                     cv2.line(frame,p3, p4,(255,255,0), 2)
205                     cv2.line(frame,pc1, p1,(255,255,0), 2)
206                     cv2.line(frame,pc2, p2,(255,255,0), 2)
```

Figure III- 25: la déclaration des commandes cv2line

Les résultats sont illustrés par la figure III-26. Les lignes sont bien dessinées on va l'utiliser prochainement pour le comptage des objets coupant ces dernies.



Figure III- 26 : capture sur le vidéo résultant contient les lignes imprimés

L'objectif de notre système est de compté le nombre des véhicules. Pour cela, on a essayé de déterminer le nombre total des frames totales dans le fichier vidéo (ligne 102 vers la ligne 106 de la Figure III- 27) et préparé un message en cas de non-détection. Afin que nous puissions estimer la durée de l'analyse de l'ensemble de la vidéo.

```
101 # try to determine the total number of frames in the video file
102 try:
103         prop = cv2.cv.CV_CAP_PROP_FRAME_COUNT if imutils.is_cv2() \
104                 else cv2.CAP_PROP_FRAME_COUNT
105         total = int(vs.get(prop))
106         print("[INFO] {} total frames in video".format(total))
107
108 # an error occurred while trying to determine the total
109 # number of frames in the video file
110 except:
111         print("[INFO] could not determine # of frames in video")
112         print("[INFO] no approx. completion time can be provided")
113         total = -1
```

Figure III- 27: fonction de calculs des nombres des frames

Le résultat de calcul est donné par la figure III.28.

```
(cv) di@di-Aspire-E1-570:~/Downloads$ python3 yolo_videogkh.py -i /home/di/Downl
oads/789.mp4 -o/home/di/Downloads/888.avi  -y 1 -c 0.6 -t 0.6
[INFO] loading YOLO from disk...
[INFO] 576 total frames in video
[INFO] cleaning up...
(cv) di@di-Aspire-E1-570:~/Downloads$
```

Figure III- 29 : la compilation avec le nombre des frames

Le nombre total des frames dans la vidéo est de 576. Maintenant, on va appliquer la fonction pour estimer le durée a finalisé l'analyse d'une seule frame et tous les frames et les afficher dans la fenêtre terminale (Figure III- 29).

```
224             # some information on processing single frame
225             if total > 0:
226                     elap = (end - start)
227                     print("[INFO] single frame took {:.4f} seconds".format(elap))
228                     print("[INFO] estimated total time to finish: {:.4f}".format(
229                             elap * total))
230
```

Figure III- 28 : une fonction pour estimer le duré nécessaire a complété le traitement

La Figure III- 30 illustre le résultat obtenu:

```
(cv) di@di-Aspire-E1-570:~/Downloads$ python3 yolo_videogkh.py -i /home/di/Downloads/789.mp4 -o/  -y 1 -c 0.6 -t 0.6
[INFO] loading YOLO from disk...
[INFO] 576 total frames in video
[INFO] single frame took 2.5373 seconds
[INFO] estimated total time to finish: 1461.4566
[INFO] single frame took 2.1004 seconds
[INFO] estimated total time to finish: 1209.8265
[INFO] single frame took 2.0956 seconds
[INFO] estimated total time to finish: 1207.0868
[INFO] single frame took 2.3114 seconds
[INFO] estimated total time to finish: 1331.3441
[INFO] single frame took 2.1098 seconds
[INFO] estimated total time to finish: 1215.2395
```

Figure III- 30: affichage des résultats au niveau du terminal linux.

❖ Comptage de nombres des véhicules

La partie comptage est la partie la plus importante dans le programme, pour déterminer le nombre des véhicules dans une image, il faut initialement cibler les classes qui nous intéresse sur notre étude. A la ligne 59 de la figure III-31 on a ciblé des classes par une liste d'objets sur notre programme (Bus, Véhicules, Motos, Bicyclette, et Camions).

```
59 list_of_vehicles = ["bicycle","car","motorbike","bus","truck"]
60 def get_vehicle_count(boxes, class_names):
61         total_vehicle_count = 0 # total vechiles present in the image
62         dict_vehicle_count = {} # dictionary with count of each distinct vehicles detected
63         for i in range(len(boxes)):
64                 class_name = class_names[i]
65                 # print(i,".",class_name)
66                 if(class_name in list_of_vehicles):
67                         total_vehicle_count += 1
68                         dict_vehicle_count[class_name] = dict_vehicle_count.get(class_name,0) + 1
69
70         return total_vehicle_count, dict_vehicle_count
71
```

Figure III- 31 : la fonction du comptage des véhicules de chaque classe

On a créé à la ligne 60 jusqu'à 70 une fonction pour compter les objets devant cette liste pour les cibler dans notre calcul.

Après on a lancé la fonction (ligne 232) dans notre boucle de détection comme suite (Figure III-32) :

On a utilisé deux méthodes pour afficher le nombre des véhicules :

1) Nous avons imprimé un texte dans notre vidéo résultant par les instructions des lignes (237-238).
2) Nous avons affiché le comptage dans la compilation de programme sur terminal (lignes 234-235)

```
231         # check if the video writer is None
232         total_vehicles, each_vehicle = get_vehicle_count(boxes, classname)
233
234         print("Total vehicles in image", total_vehicles)
235         print("Each vehicles count in image", each_vehicle)
236
237         font = cv2.FONT_HERSHEY_SIMPLEX
238         cv2.putText(frame,'counter='+str(each_vehicle), (20,45),font,1, (255, 0, 0),2)
```

Figure III- 32 : affichage des résultats de comptage

Les résultats obtenus est considéré excellent (Figure III-33/34) l'algorithme a compté tous les véhicules sur les deux voies d'une manière précise et fidèle pour chaque expérience



Figure III- 33 : capture sur le vidéo résultant contient le nombre des véhicules de la route.

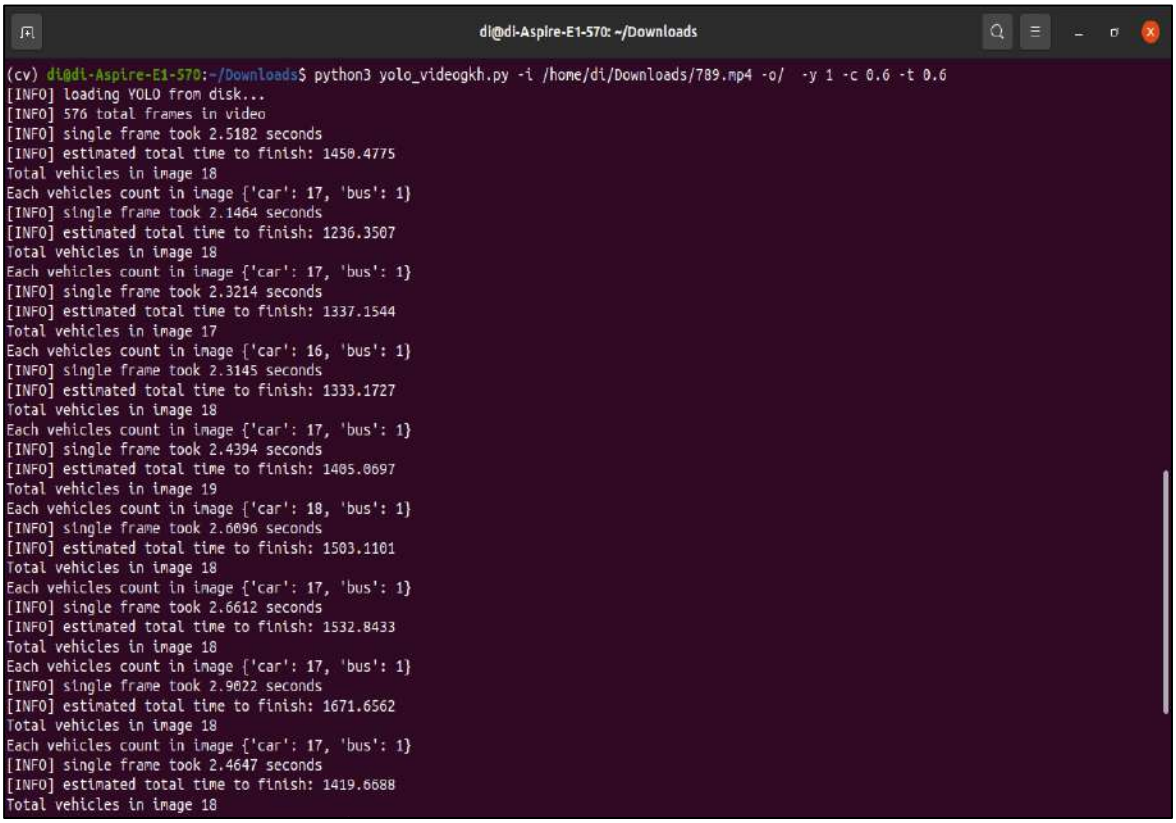


Figure III- 34 : une capture sur la compilation de notre programme sur terminal

❖ Calcul du débit

L'organigramme de la Figure III-35 illustre la méthode pour calculer le débit des véhicules, en commence par préparer une zone de surveillance laquelle on fait les calculs selon ces limites. Le concept de zone surveillée est simple, des lignes limites sont nécessaires, pour déterminer l'entrée ou la sortie des objets. Ces lignes limites sont constituées de lignes virtuelles tracées sur ces images par OpenCV. Il y a deux lignes limites :

Ligne d'entrée (exit) : ligne de référence qui définit la limite d'entrée d'un objet dans la zone surveillée

Ligne de sortie (extra) - ligne de référence qui définit la limite de sortie d'un objet de la zone surveillée.

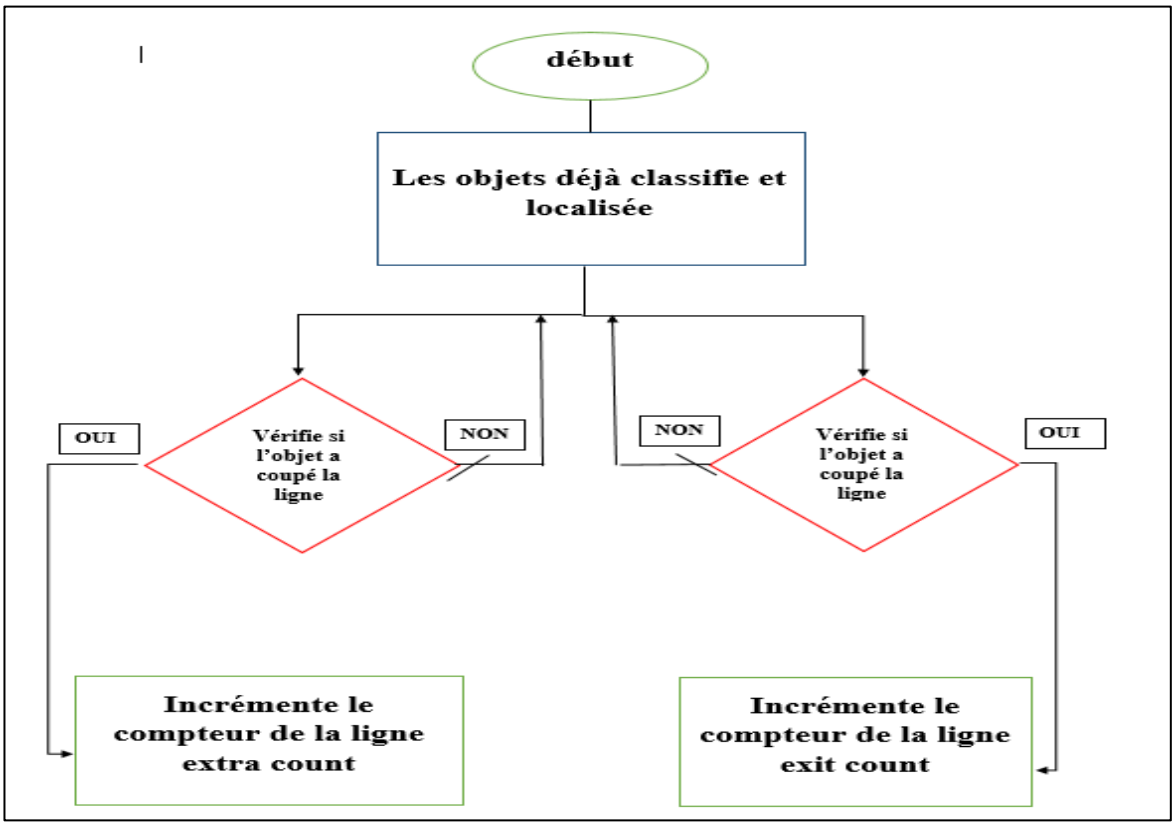


Figure III- 35 : organigramme de calcul le débit

Calculer le nombre total des véhicules coupant la ligne dans une unité de temps représente le flux de cette route. Par conséquent, comme le contrôle d'entrée/sortie est basé sur les lignes d'entrée et de sortie (ces lignes sont tracées sur l'image de la vidéo résultant), elles peuvent être facilement définies et modifiées. L'incrémentation d'un compteur (compteur d'entrée ou de sortie) se produit lorsque le centroïde (centre) de l'objet franchit

les lignes d'entrée ou de sortie. Cette approche rend le compte indépendant de la forme de l'objet. Mais nous appliquons cette fonction seulement pour les objets déjà classés

Le comptage des objets entrant et sortant de la zone surveillée se fait comme de la manière décrite comme suit.

Pour compter les objets entrant dans la zone surveillée : tous les centroïde de l'objet qui croisent la ligne bleue/ligne d'entrée et proviennent de la ligne rouge/ligne de sortie (en d'autres termes, tous les centroïdes de l'objet qui étaient situés entre les deux lignes et se déplacent dans la direction de la ligne bleue) sont comptés comme des objets qui sont entrés dans la zone surveillée. Ceci est montré dans la figure III- 36 /a.

Pour compter les objets sortant dans la zone surveillée : tous les centroïdes de l'objet qui croisent la ligne rouge / ligne de sortie et proviennent de la ligne bleue / ligne d'entrée (en d'autres termes, tous les centroïdes de l'objet qui se trouvaient entre les deux lignes et se déplacent dans la direction de la ligne rouge) sont comptés comme des objets sortant de la zone surveillée. Ceci est illustré par la figure III- 36/ b.

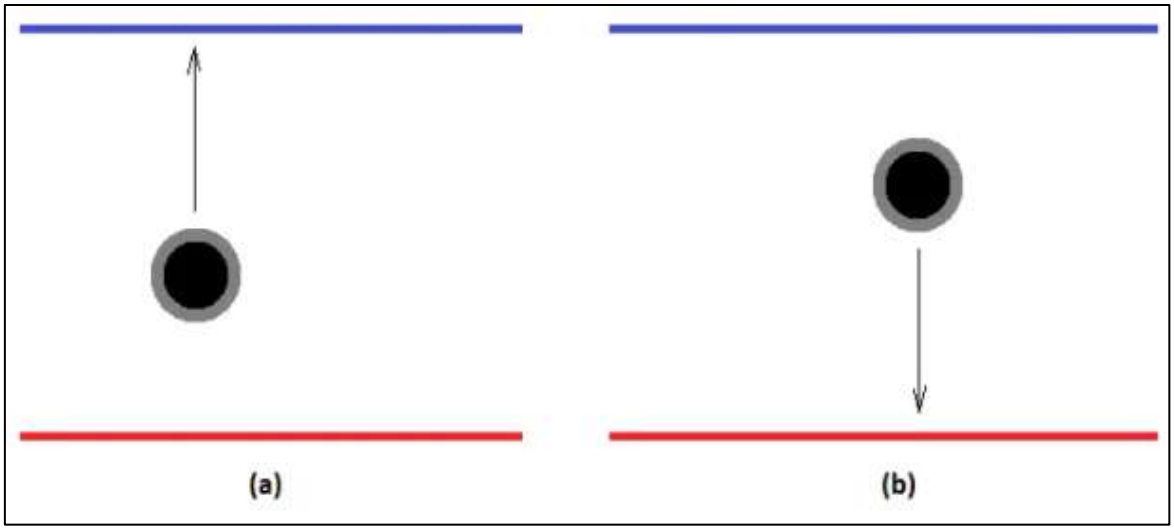


Figure III- 36: comptage des objets entrant et sortant de la zone surveillée

Pour le calcul de débit des véhicules dans notre programme, on a initialement identifié une fonction de calcule par condition en boucle (Figure III-38) si l'objet déjà classé a coupé la ligne de calcul. Des compteurs s'incrémentent (Figure III-37).

```
def CheckEntranceLineCrossing(y, linecount2, lineCount):
  AbsDistance = abs(y - linecount2)

  if ((AbsDistance <= 2) and (y < linecount)) :
        return 1
  else:
        return 0

#Check if an object in exitting from monitored zone
def CheckExitLineCrossing(y, linecount2, lineCount):
    AbsDistance = abs(y - lineCount)
    if ((AbsDistance <= 2) and (y > linecount2)):
        return 1
    else:
        return 0
```

Figure III- 37 : fonction de calcul le débit.

```
207                     if (CheckEntranceLineCrossing(Cy,linecount2,lineCount)):
208                         EntranceCounter += 1
209                     if (CheckExitLineCrossing(Cy,linecount2,lineCount)):
210                         ExitCounter += 1
211                     cv2.putText(frame, "Entrances: {}".format(str(EntranceCounter)), (10, 150),cv2.FONT_HERSHEY_SIMPLEX, 1, (250, 0,
    1), 2)
212                     cv2.putText(frame, "Exits: {}".format(str(ExitCounter)), (10, 250),cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
```

Figure III- 38 : fonction d'incrémentation les Constans.

On a expérimenté la fonction sur notre vidéo. Les résultats n'étant pas exact Figure III- 39 (dépend de l'unité de traitement utilisé). Chaque véhicule coupant de la ligne fait incrémenter le compteur d'affichage de 3 à 4 fois, la raison ce que les cadres ne sont pas bien fixé.



Figure III- 39 : la première expérience de calcul de débit.

Pour résoudre ce problème, on a créé une fonction traqueuse (Figure III- 40). Elle a pour objectif de faire traquer l'objet détecté et fixer une identification sur chacun. Le fichier identifie cette fonction est sur le nom centroïde tracker, nous importé de la ligne 6 de la Figure III-3, ce genre des fonctions nécessite des calculateurs de haute performance. Notre système hardware n'est pas convenable pour cette tâche.

```
215                     objects = ct.update(rects)
216                     # loop over the tracked objects
217                     for (objectID, centroid) in objects.items():
218                     # draw both the ID of the object and the centroid of the
219                     # object on the output frame
220                             text = "ID {}".format(objectID)
221                             cv2.putText(frame, text, (Cx-10,Cy-10),
222                             cv2.FONT_HERSHEY_SIMPLEX, 0.3, (0, 255, 0), 1)
223                             cv2.circle(frame, (Cx,Cy), 4, (0, 255, 0), -1)
224             # some information on processing single frame
```

Figure III- 40: fonction traqueuse

On a essayé sur notre algorithme pour voir les résultats. Les objets sont bien identifiés (Figure III- 41/42) mais malheureusement les centre des cadres ne sont pas fixes.



Figure III- 42 : les résultats de la fonction traqueuse.



Figure III- 41:une des résultats finals de notre system par la Fonction traqueuse.

Pour un développement plus utile, on a préparé une fonction pour cibler l'analyse sur une seule partie de la route ou une seule voie pour sélectionner à travers une commande laquelle on met les dimensions de cette partie ce qui permet que l'analyse soit juste proportionnelle pour cela (Figure III- 43).

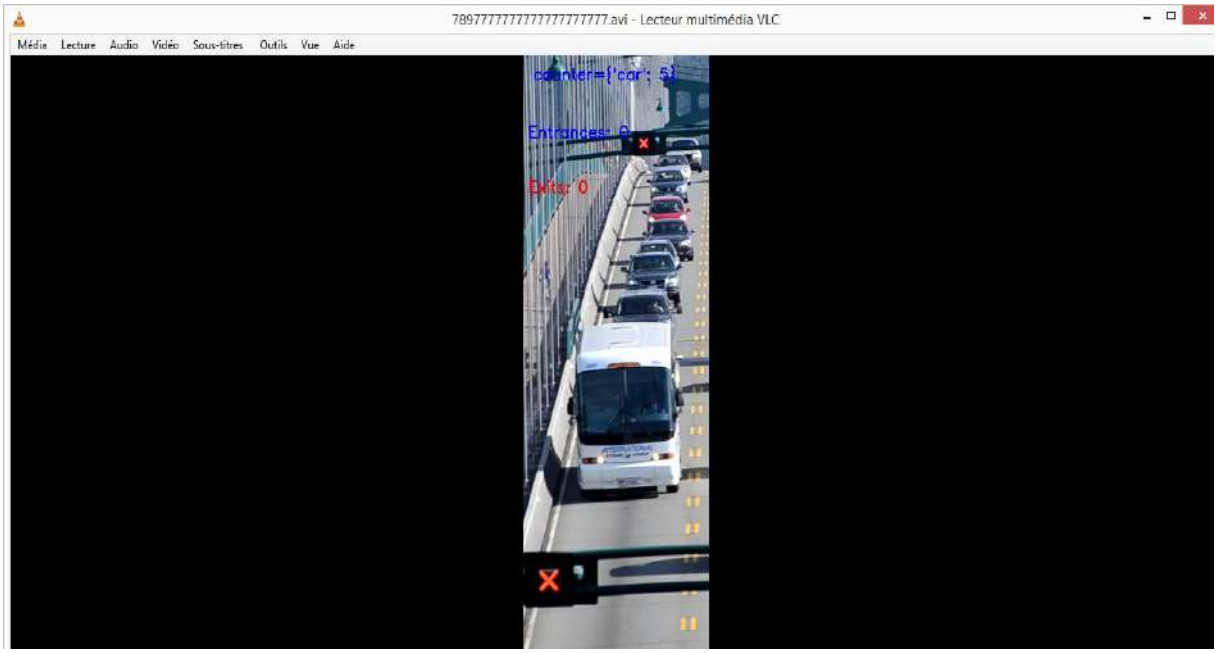


Figure III- 43 : cibler une seule route

Cette instruction est très utile, pour améliorer les performances de notre système.

III.4.3 Discussions

III.4.3.1 Pour quoi YOLO

En ce qui concerne la détection d'objets basée sur l'apprentissage profond, il existe trois principaux détecteurs d'objets

- R-CNN et leurs variantes, y compris le R-CNN original, le Fast R-CNN et le Faster R-CNN
- Détecteur à coup unique (SSD)
- YOLO

Les R-CNN sont l'un des premiers détecteurs d'objets basés sur l'apprentissage profond et sont un exemple de détecteur à deux étages.

Le problème avec la méthode R-CNN standard était qu'elle était douloureusement lente et qu'elle n'était pas un détecteur d'objets complet de bout en bout.

Alors que les R-CNN ont tendance à être très précis, le plus gros problème avec la famille de réseaux R-CNN est leur vitesse - ils étaient incroyablement lents, n'obtenant que 5 FPS sur un GPU.

Pour aider à augmenter la vitesse des détecteurs d'objets basés sur l'apprentissage profond, les détecteurs à un seul coup (SSD) et YOLO utilisent une stratégie de détection à une étape.

Ces algorithmes traitent la détection d'objets comme un problème de régression, prenant une image d'entrée donnée et apprenant simultanément les coordonnées de la boîte englobante et les probabilités d'étiquette de classe correspondantes.

En général, les détecteurs à un étage ont tendance à être moins précis que les détecteurs à deux étages, mais sont nettement plus rapides.

YOLO (You Only Look Once) est un excellent exemple de détecteur à un étage. Un détecteur d'objet capable de détecter des objets en super temps réel, obtenant 45 FPS sur un GPU [12] [14] [16].

III.4.3.2 Comparaison avec d'autres systèmes de collection des données

Pour bien évalué notre système, il faut d'abord comparer à d'autres système du même objectif, pour cela on a choisi trois systèmes à comparer en se basant sur les critères les plus important à la commercialisation. Tableau III.1 indique une comparaison entre notre système et quelques d'autres systèmes de collection des données. [4][7][8]

Les critères	Notre Système	Système radar	Système ATLYSTE
Rendement	✓ Peut analyser plusieurs voies d'une route. ✓ Une large superficie d'analyse	✓ Une seule voie à analyser. ✓ Nécessite deux radars au minimum pour les calculs	✓ Une seule voie à analyser. ✓ Nécessite deux capteurs au minimum pour les calculs
Précision	Haut	Haut	Haut
Adaptation et implémentation	Très facile	Très facile	Difficile : Nécessite des grands travaux.
Coût	Moyenne	Trop chère	Moins Chère
Les fonctions peu s'exprime	✓ Gestion de trafic routiers. ✓ Surveillance des routes en temps réel.	✓ Peut s'exprimer pour les calculs de	✓ Calcul le nombre des véhicules

	✓ Utile dans la sécurité frontière des États.	vitesse des véhicules.	passant.

Tableau III- 1: comparaissent des performances a d'autres systèmes.

L'avantage majeur de ce système basé sur l'algorithme YOLO par rapport à tous les autres systèmes intelligents de type gestion routière, est la vitesse de traitement des images et la capacité de traitement. 80 classes peuvent être identifié à la fois dans une image d'une façon très rapide et précise,

Dans ce chapitre, le traitement cible 4 classes. Le système est très adaptatif de tous les chemins et les intersections des routes, et très facile à implémenter, le système est même capable de réajuster une image et de prendre juste une partie de la route pour étudier une seule voie ou plusieurs ou une partie d'une voie. Il peut même traiter plusieurs voies d'un carrefour, ce qui est un plus de ce système par rapport aux autres.

III.5 Conclusion

Dans ce chapitre nous avons expérimenté l'algorithme YOLO pour la détection de trafic routier et détaillé les étapes de l'implémentation et par la suite définir les besoins du coté hardware et software. Les résultats de simulation sont très satisfaisants. Le système été très facile à utiliser et adaptatif. Cependant comme tous les systèmes basés sur le deep learning, ils nécessitent des calculateurs de haute performance pour le traitement en temps réel.

Pour les calculs du débit du trafic ça nécessite des fonctions traqueuses, malheureusement elles sont trop difficiles à implémenter et de faible précision pour les captures de haute résolution mais quoi qu'il en soit-ce system reste le plus fiable.

Un effort sur l'optimisation reste nécessaire pour réduire le temps de traitement et permettre l'intégration du système proposé dans divers application telle que la sécurité et le contrôle des frontières.

Conclusion Générale

Le travail élaboré dans le cadre de ce mémoire représente le fruit d'une recherche dans le demain de gestion de trafic. Nous intéressons sur la collection des données routiers pour faire analyser et utiliser pour la gestion des feux de circulation, c'est un système intelligent qui propose une solution basée sur une caméra et la technique de l'apprentissage profond (l'algorithme YOLO) implémenté sur un calculateur pour la détection et la classification des objets.

Premièrement on a présenté quelque systèmes intelligent de gestion routier et collection des données. Ensuite, nous avons abordé les techniques d'intelligence artificielle en générale, avant d'examiner les techniques de l'apprentissage profond en détail. On a présenté quelques algorithmes pré-entrainé utilisant ces techniques. Après on cible les algorithmes qui utilisent les technique de l'apprentissage profond qui prouve son efficacité dans le domaine de traitement d'image et l'identification et la localisation des objets. Cela a permis de choisir un algorithme qui contient des bonnes caractéristiques pour l'utiliser dans notre projet.

Grâce à plusieurs avantages que soit sa rapidité ou sa simplicité d'implémentation On choisit d'utiliser l'algorithme YOLO, c'est un modèle pré entrainé fameux dans le domaine de la détection et la classification des objets, YOLO exécute la détection directement sur un échantillonnage dense d'emplacements possibles. Il utilise la structure Régression / Classification. Après nous avons expliqué en détail les étapes de l'implémentation de cet algorithme et le langage python3 et les bibliothèques installées, comme OpenCV et Keras, ensuite on a présenté le matériel utilisé,

Pour validation des tests sont fait sur une vidéo avec des caractéristiques spéciales pour évaluer notre système. De bons résultats sont obtenus, le nombre des véhicules dans la séquence de la vidéo été bien compté et d'une façon précise pour l'indiquer dans les systèmes de gestion routiers

Comme perspective, et pour arriver a améliorer ce travail nous pouvons envisager d'introduire des optimisations sur le software et améliorer quelques fonctions pour les calculs de débit de trafic et permettre l'intégration du système proposé dans divers applications tels que la sécurité et le contrôle des frontières.

Bibliographie

[1] Hadj Rabah Sabrina et Nouari Djahida « Feux Tricolores Intelligents pour les Villes Intelligentes ». Mémoire de master. Université Akli Mohand Oulhadj de Bouira.2019.

[2] CHABANE Meriem. HADJ HACENE Rabab. « Contrôle Intelligent des Feux de Circulation d'une Intersection en Utilisant L'algorithme de Colonie de ».mémoire Master en Génie Électrique. FACULTE DE DES SCIENCES ET DE LA TECHNOLOGIE. UNIVERSITÉ Dr. YAHYA FARES DE MEDEA.

[3] Traffic count - Wikipedia. (n.d.). Retrieved July 11, 2021, Disponible sur Nom de la base de données: https://en.wikipedia.org/wiki/Traffic_count#Counting_methods

[4] Les données de trafic en temps réel montrent l'impact du COVID | MetroCount. (n.d.). Retrieved July 16, 2021, Disponible sur Nom de la base de données : https://metrocount.com/fr/les-donnees-de-trafic-en-temps-reel-montrent-limpact-du-covid/?fbclid=IwAR3YuowpA1tEOAfPF-vSt40hfucbVLfA8KocIdp4TXG8YcMKOd4mZSI3mBc

[5] TMA-122 - Icoms Detections. (n.d.). Retrieved July 16, 2021, Disponible sur Nom de la base de données : https://www.icomsdetections.com/fr/project/tma-122/

[7] Comptage Vehicule Trafic Routier | TagMaster France. (n.d.). Retrieved July 16, 2021, from https://fr.tagmaster.com/comptage-vehicule-trafic-routier

[8]Comptage Vehicule Trafic Routier | TagMaster France. (n.d.). Retrieved July 16, 2021, from https://fr.tagmaster.com/comptage-vehicule-trafic-routier

[9] For Traffic Counting | SWARCO. (n.d.). Retrieved July 11, 2021, Disponible sur Nom de la base de données: https://www.swarco.com/products/detection-sensors/traffic-counting

[10] Boughaba Mohammed et Boukhris Brahim. « L`apprentissage profond (Deep Learning) pour la classification et la recherche d'images par le contenu ». Mémoire Master Professionnel. Faculté des Nouvelles Technologies de l'Information et de la Communication. UNIVERSITE KASDI MERBAH OUARGLA. 2016/2017

[11] PreTrained Deep Learning Models | Computer Vision. (n.d.). Retrieved July 11, 2021, Disponible sur Nom de la base de données: https://www.analyticsvidhya.com/blog/2018/07/top-10-pretrained-models-get-started-deep-learning-part-1-computer-vision/

[12] Adrian Rosebrock. YOLO object detection with OpenCV. [consulté le 2 juin 2021]. Dernière mise à jour le 5 juillet 2021. Disponible sur Nom de la base de données: https://www.pyimagesearch.com/2018/11/12/yolo-object-detection-with-opencv/

[13] Comment détecter les objets plus efficacement avec YOLO • Aqsone. (n.d.). Retrieved July 11, 2021, Disponible sur Nom de la base de données: https://www.aqsone.com/fr/blog/2021/data-science-fr/comment-detecter-les-objets-plus-efficacement-avec-yolo/

[14] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2016-December, pp. 779–788, 2016.

[15] "ImageNet." [Online]. Available: https://image-net.org/download.php. [Accessed: 16-Jul-2021].

[16] YOLO : You Only Look Once - Real Time Object Detection - GeeksforGeeks. (n.d.). Retrieved July 17, 2021, Disponible sur Nom de la base de données: https://www.geeksforgeeks.org/yolo-you-only-look-once-real-time-object-detection/

[17] Darknet in 2021 - Reviews, Features, Pricing, Comparison - PAT RESEARCH: B2B Reviews, Buying Guides & Best Practices. (n.d.). Retrieved July 11, 2021, Disponible sur Nom de la base de données: https://www.predictiveanalyticstoday.com/darknet/

[18] "Darknet: Open Source Neural Networks in C." [Online]. Available: https://pjreddie.com/darknet/. [Accessed: 16-Jul-2021].

[19] Installing Darknet. (n.d.). Retrieved July 11, 2021, Disponible sur Nom de la base de données: https://pjreddie.com/darknet/install/

[20] 3.9.6 Documentation. (n.d.). Retrieved July 11, 2021, Disponible sur Nom de la base de données: https://docs.python.org/3/

[21] argparse — Analyseur pour les options de ligne de commande, les arguments et les sous-commandes — Documentation Python 3.9.6