

Rapport de Projet de Fin d'Etudes

En vue de l'obtention d'un diplôme d'ingénieur d'état en Génie électrique

Filière : Génie électrique, systèmes embarqués et télécommunications

Réalisé au sein de l'entreprise Deweb Technology



Sous le thème :

ÉCLAIRAGE PUBLIC INTELLIGENT SÉLECTIF PAR RAPPORT À UN OBJET MOUVANT

Réalisé par :

- INDJAREN Ayoub

Membres de jury :

- MOHAMMED KHALDOUN : **Président**
- ABDELHADI ENNAJIH : **Rapporteur**
- AHMED ERRAMI : **Encadrant pédagogique**
- ILIASS JOUDAT : **Encadrant industriel**

Soutenu le : 05/07/2022

Promotion 2022

Résumé

Dans le cadre de l'évolution technologique et des innovations qui s'accroissent de jour en jour, et tenons compte de la vision du gouvernement d'appuyer l'économie numérique et la technologie de l'information, il est temps de penser à une infrastructure intelligente dans notre pays.

Cet essai démontre l'intérêt de transformer l'éclairage public existant en éclairage public intelligent en utilisant les nouvelles technologies à savoir l'Internet des Objets et l'artificielle intelligence. Cette transformation peut être le premier pas pour la création des villes intelligentes « SMART CITY » où plusieurs autres services publics et privés seront être pris en considération dans le but d'optimiser, contrôler la consommation d'énergie électrique.

Ce rapport présente le travail effectué dans le cadre de mon projet de fin d'études. Concernant la conception et le développement d'un système d'éclairage public intelligent utilisant Raspberry pi et Intelligence Artificielle. Concevoir un prototype économe en énergie et faire un pas en avant pour rendre nos rues intelligentes.

La réalisation du prototype comprend deux étapes principales : Une étude préliminaire a été menée pour décrire les plateformes matérielles et les bibliothèques logicielles répondant aux spécifications requises. TensorFlow, OpenCV et Numpy ont été sélectionnés en tant que bibliothèques logicielles pour construire le modèle de détection des objets en python basé sur le Deep Learning. Et après avoir construit le modèle de détection des objets formé par des réseaux de neurones convolutifs, celui-ci est implanté dans un système proposé pour rendre le service d'éclairage public intelligent basé sur une carte Raspberry Pi équipé d'un module caméra a vision nocturnal, ainsi que sur d'autre composants matériels indispensables comme les capteurs de mouvement PIR et de lumière LDR.

Le test des performances a conclu que le système proposé répond aux exigences en matière des plates-formes matérielles et logicielles utilisées.

Mots clés : Villes intelligentes, Eclairage public intelligent, Intelligence artificielle, Internet des objets, Raspberry Pi, Tensorflow, OpenCV.

Abstract

In the context of technological evolution and innovations which are accelerating day by day, and consider the government's vision to support the digital economy and information technology, it's time to think about smart infrastructure in our country.

This test demonstrates the interest of transforming existing public lighting into smart public lighting using new technologies, namely the Internet of Things and artificial intelligence. This transformation can be the first step for the creation of smart cities "SMART CITY" where several other public and private services will be taken into consideration in order to optimize, control the consumption of electrical energy.

This report presents the work done as part of my graduation projects. Concerning the design and development of a smart public lighting system using Raspberry pi and Artificial Intelligence. To conceive a prototype that saves energy consumption and takes a step forward to make our streets smart.

The realization of the prototype consists of two main steps: A preliminary study has been carried out to describe the hardware platforms and software libraries that meet the requirements. TensorFlow, OpenCV and Numpy have been selected as software libraries to build the object detection model with python based on Deep Learning. And after having built the object detection model formed by convolutional neural networks, it is implemented in a system proposed to make the public lighting service intelligent based on a Raspberry Pi card equipped with a vision camera module night light, as well as other essential hardware components such as PIR motion and LDR light sensors.

The performance test concluded that the proposed system meets the requirements of the hardware and software platforms used.

Mots clés: Smart City, smart public lighting, Artificial Intelligence, Internet of Things, Raspberry Pi, Tensorflow, OpenCV.

Dédicaces

Je dédie mon modeste travail

A mes chers parents

Pour leur grand soutien, leurs sacrifices que rien au monde ne peut les récompenser et sans lesquels je n'aurais jamais pu parcourir ce chemin.

Que Dieu vous protège.

A mes chers frères et sœurs

Pour leur amour et leur incontestable appui.

A mes chers ami(e)s.

A toutes les personnes chères à mon cœur je dédie ce travail.

Remerciements

Avant de commencer la présentation de ce travail, je profite de l'occasion pour remercier toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce PFE.

Je tiens à présenter mes reconnaissances à mon encadrant industriel **Mr. Iliass Joudat**,

Ainsi que mon encadrant pédagogique **Mr. Ahmed Errami** pour leurs conseils, leurs orientations, et la disponibilité qu'il m'ont accordée pour faire réussir ce travail, tout au long de la période du mémoire.

Je remercie mes professeurs de l'École nationale supérieure d'électricité et de mécanique- **ENSEM** - qui ont fait beaucoup d'efforts pour nous transmettre leurs connaissances. Vos compétences incontestables ainsi que vos qualités humaines vous valent l'admiration et le respect de tous. Je vous adresse mes sincères remerciements pour votre patience et votre encadrement durant toutes ces années.

Merci aux membres du jury pour l'honneur qu'ils me font en jugeant ce travail. Et à vous tous, honorables lecteurs.

Table des matières

Résumé	i
Abstract	ii
Dédicaces	iii
Remerciements	iv
Liste des figures	v
Liste des tableaux	vi
Liste des acronymes	vii
Introduction générale	12
Chapitre 1 : Présentation générale	
1.1 Présentation de l'organisme d'accueil	13
1.2 Etude préliminaire	15
1.3 Contexte du projet	19
1.3.1 Problématique.....	19
1.3.2 Description du projet	19
1.3.3 Cahier des charges.....	19
1.3.4 Principe de fonctionnement.....	20
1.3.5 Les objectifs du projet	21
Chapitre 2	
1) Généralités sur l'apprentissage en profondeur	
1.1 Bref historique de la reconnaissance et de la détection d'objets.....	22
1.2 Bref aperçu de l'apprentissage en profondeur - Deep Learning.....	24
1.3 Réseaux de Neurones Convolutifs CNN	25
1.3.1 Structure des réseaux de neurones convolutionnels	25
1.3.1.1 Couches de convolution	25
1.3.1.2 Couche d'activation.....	27
1.3.1.3 Couche de regroupement	27
1.3.1.4 Couches Entièrement Connectées.....	28
1.3.2 Visualiser les réseaux de neurones convolutionnels	28
1.3.3 Entraînement des réseaux de neurones.....	30
1.3.4 Extracteurs de fonctionnalités.....	31
1.3.5 Détection des Objets avec les CNNs	31
1.3.6 Eviter le sur-Apprentissage.....	32

2) Présentation du modèle MobileNet-SSD

2.1	Détection des Objets avec MobileNets	36
2.1.1	Le principe de fonctionnement de MobileNet	37
2.1.2	L'avantage d'utilisation du modèle MobileNet	38
2.2	Comprendre SSD MultiBox	39
2.2.1	Architecture	42
2.2.2	Pertes MultiBox d'architecture SSD	42
2.2.3	Priors MultiBox et IoU	42
2.3	Le modèle MobileNet-SSD	43
2.4	Importance des jeux de données	45

Chapitre 3 : Préparation de l'environnement logiciel

3.1	Présentation de l'architecture du système de développement utilisé	46
3.2	Le choix des outils utilisés	46
3.3	Mise en œuvre de la connexion entre la carte Raspberry pi et le PC	47
3.4	Déploiement de L'algorithme SSDLite-Mobilenet sur le RPi	55
3.4.1	La mise en œuvre de la carte Raspberry Pi	55
3.4.2	Installation de la librairie TensorFlow	56
3.4.3	Installation de la bibliothèque OpenCV	57
3.4.4	Compilation et installation de Protobuf	57
3.4.5	Configuration de la structure de répertoire TensorFlow et de la variable PYTHONPATH	58
3.5	Test du modèle	60

Chapitre 4

1) Conception matériel du prototype

1.1	L'environnement matériels	62
1.2	Conception de la maquette	62
1.2.1	L'unité d'entrée	62
1.2.2	L'unité de traitement	63
1.2.3	L'unité de contrôle des lampadaires	63
1.3	Mise en œuvre du prototype	64
1.4	Conception électrique du système	64
1.4.1	Schéma de principe de l'unité de commande	65
1.4.2	Conception d'une alimentation électrique autonome du système	65
1.4.2.1	Système jour/nuit	65
1.4.2.1	Basculement charge/alimentation	66
1.4.2.1	Chargement de la batterie	66
1.5	Choix des composants	67

2) Evaluation du prototype et étude technico-économique

2.1	Réalisation du Prototype	71
2.2	Evaluation du système	71
2.3	Etude technico-économique	74

Conclusion générale

78

Bibliographie

Annexe 1 : Les caractéristiques de chaque composant utilisé dans ce projet

Annexe 2 : Code Python pour détecter les objets en temps réel

Liste des figures

1.1	Concept du Smart City	4
1.2	Eclairage public à commande locale par départ	5
1.3	Eclairage public connecté à un système de gestion centralisé.....	6
1.4	Eclairage public connecté à commande locale décentralisé.....	7
1.5	Les situations principales du système.....	9
1.6	Schéma descriptif du fonctionnement du système	9
2.1	L'opération de convolution.....	15
2.2	Entrée complétée de zéro (à gauche), avec un filtre 3×3 (moyen), pour obtenir une sortie ayant la même taille que l'entrée non complétée de zéro (à droite).....	15
2.3	La fonction d'activation ReLU	16
2.4	Un exemple de l'opération de max pooling, utilisant une taille de mise en commun de 2×2	16
2.5	Un exemple de la façon dont les premières couches d'un réseau neuronal détectent des caractéristiques simples, telles que les couleurs et les lignes, tandis que les couches ultérieures les combinent en des caractéristiques de plus en plus complexes.....	17
2.6	Une visualisation de l'entrée préférée de certains filtres dans les 5 premières couches du VGG-16.....	18
2.7	Des images synthétiques générées de ce qu'un CNN considère comme étant le poivron (à gauche), le citron (au milieu) et le husky (à droite)	18
2.8	Une image synthétique générée d'une pie, avec 99,99% de confiance	19
2.9	Un aperçu de l'architecture R-CNN	21
2.10	Un aperçu de l'architecture Fast R-CNN	21
2.11	Vue d'ensemble d'une version de l'architecture SSD (en haut) et de l'architecture YOLOv1 (en bas)	22
2.12	Précision en fonction du temps, avec méta-architecture d'indication de formes de marqueur et extracteur d'indicateur de couleurs	23
2.13	Principe de la méthode de Dropout	24
3.1	Les filtres de convolution standard en (a) sont remplacés par deux couches : convolution en profondeur en (b) et convolution ponctuelle en (c) pour construire un filtre séparable en profondeur.....	27
3.2	filtres convolutionnels en profondeur en (a), et filtres convolutionnels ponctuels en (b).....	27
3.3	Exemple d'approche de fenêtre coulissante.....	28
3.4	Exemple d'une grille de 4×4	29
3.5	Exemple de deux boîtes d'ancrage	30
3.6	La boîte englobante du bâtiment 1 est plus haute tandis que la 2-eme est plus large.....	30
3.7	Architecture d'un réseau de neurones convolutifs avec un détecteur SSD	31
3.8	Diagramme expliquant IoU (de Wikipedia).....	32
4.1	L'interface graphique de système d'exploitation de Raspberry Pi	44
4.2	ajoutez PYTHONPATH pour TensorFlow	48
4.3	Menu de configuration de Raspberry Pi.....	50
4.4	Exemple des objets détectés	50

5.1	Schéma synoptique de la maquette	52
5.2	Plan 3D de la maquette	53
5.3	Schéma électronique du système	54
5.4	Montage globale	55
5.5	Montage LDR	56
5.6	Basculement Charge/alimentation	56
5.7	Chargeur de batterie.....	56
5.8	RPi 3 model B.....	57
5.9	Pi caméra de vision nocturnal.....	57
5.10	Capteur de mouvement PIR.....	58
5.11	Capteur de lumière LDR.....	58
5.12	Caractéristique résistance/éclairage	59
6.1	Prototype final	60
6.2	L'image résultat	61
6.3	Exemple d'une voiture détecté par le système	62
6.4	La distance optimale entre les lampadaires	63
6.5	Comparaison entre le système d'éclairage public classique et le système d'éclairage intelligent proposé en termes de consommation d'énergie électrique	65

Liste des tableaux

1	La fiche technique de l'entreprise	13
2	Comparaison entre l'architecture du modèle SSD et celui du modèle YOLO	34
3.1	Modèles entraînés par la base de données COCO	44
3.2	Les principaux jeux de données disponibles	45
6.1	Le nombre des FPS traités par les modèles YOLO, SSDLite-Mobilenet-V2 et SSDLite-Mobilenet-V2 avec TF Lite	73
6.2	Prix des équipements nécessaires par unité	75

2D 2 dimensions

3D 3 dimensions

ARM Advanced Risk Machine

CNN Convolutional Neural Network

CPU Central Processing Unit

DNN Deep Neural Network

DPM Deformable Parts Model

FPS Frames Par Seconde

GPU Graphics Processing Unit

HOG Histogram of Oriented Gradients

IA Intelligence Artificielle

ITSA Intelligent Transportation Society of America

NIN Network In Network

R-CNN Region-based Convolutional Neural Network

RPi Raspberry Pi

RPN Region Proposal Network

SPD Single Pass Detector

SSD Single Shot multibox Detector

SVM Support Vector Machine

YOLO You Only Look Once

CSI Camera Serial Interface

Introduction

L'éclairage public est en pleine mutation depuis quelques années grâce aux développements de la vie urbaine et sa fonction de base, permettant aux utilisateurs de vivre dans de bonnes conditions de confort. Malgré ça, les systèmes d'éclairage aujourd'hui rencontre un grand défi concernant la consommation de l'énergie électrique et sa mauvaise gestion.

Notre projet est concentré principalement sur la conception et le développement d'un système d'éclairage public intelligent par le biais d'un prototype à base d'une carte Raspberry Pi utilisant une Picamera capable de détecter et de compter les objets empruntant la route pendant la nuit par application des techniques d'apprentissage automatique profond et de traitement d'images en temps réel, et aussi par utilisation des capteurs de lumière et de présence (humain / voitures,...) visant en premier lieu d'optimiser la consommation de l'énergie électrique au niveau des systèmes d'éclairages publics en faisant allumer les lampadaires juste durant des événements prédéfinis et préétablis.

Au-delà du savoir et du savoir-faire dévolu, ce projet nous a donné l'occasion de mettre en œuvre et intégrer les connaissances techniques acquises avant et au cours du projet ainsi que d'acquérir une expérience professionnelle indispensable pour compléter notre formation universitaire par des compétences pratiques.

Ce rapport est devisé en quatre chapitres principaux, commençant par une présentation générale qui est Composé d'une présentation sur l'organisme d'accueil, les systèmes d'éclairages publics intelligents existants puis en fin le contexte général du projet.

Le deuxième chapitre présente l'étude des modèles d'apprentissage profonds les plus utilisés dans l'embarquée pour faire un choix convenable avant le déploiement sur notre carte Raspberry pi.

Enfin le dernier chapitre, sera consacré à la conception matériel et la réalisation du prototype et une étude technico-économique pour la concrétisation et la commercialisation de notre projet au futur.

Nous terminons notre rapport par une conclusion générale mettant en évidence les résultats obtenues après la réalisation du prototype ainsi que d'autre solutions créatives qu'on peut apporter aux systèmes d'éclairages publics pour le contrôle de la consommation d'énergie dans ces derniers.

Chapitre 1

Présentation générale

1.1 Présentation de l'organisme d'accueil

Deweb Technology est une société située à Agadir spécialisée dans l'importation, la distribution et l'intégration de produits et solutions créatives en matière de l'efficacité énergétique et de sécurité professionnelles plus spécifiquement dans les domaines de la domotique.

- **Domaines d'activité :**

Parmi les activités principales de l'entreprise on trouve :

- Les systèmes de vidéosurveillance filaire et sans fil,
- Les systèmes de détection d'incendie,
- Les systèmes de détection d'intrusions,
- Les systèmes de pointage et gestion des temps.
- Les systèmes d'accès et gestion de parking,
- Le contrôle d'accès.
- Intégration des solutions technologiques pour l'optimisation des dépenses énergétiques à domicile.

- **Fiche technique :**

Table 1 – La fiche technique de l'entreprise

Raison sociale	Deweb Technology
Forme juridique	Société à responsabilité limitée (SARL)
Siège social	Lot 124, Rue Essaouira Zone Industriel Tassila Dcheira el jhadia, Agadir
Téléphone	+212528844511
Fax	+212528844921
Effectif	12 personnes
Date de création	22/05/2005
Site Web	www.deweb.ma

- *Principaux clients de Deweb Technology :*



- *Les certifications du Deweb Technology :*

- ❖ **Bosh certificat :**

Définit les exigences en matière des systèmes de contrôle d'accès, la vidéo surveillance, les systèmes de détection d'intrusion et les systèmes de détection d'incendie.

- ❖ **Legrand certificat :**

Définit l'ensemble des exigences pour un système de management de la qualité.



BOSCH

1.2 Etude préliminaire

1.2.1 Les villes intelligentes :

La ville, est considérée comme le plus important rassemblement humain, c'est un organe vital et complexe, c'est un lieu de concertation de plusieurs activités, et un lieu de mobilité intense pour des individus afin d'assurer leurs besoins. L'intelligence des villes est une notion plutôt récente qui représente une nouvelle approche de développement urbain en mettant en avant l'intégration de nouvelles technologies d'information et de communication dans la gestion de la ville dans le but de répondre aux nécessités de celle-ci de façon efficiente. L'idée de l'intelligence d'une ville est donc de mettre en place un développement de ladite ville en utilisant les nouvelles technologies dans le but d'améliorer la qualité, la performance et l'interactivité des services urbains tout en réduisant au maximum les coûts en argent, temps et ressources.

Dans ce contexte, l'éclairage public intelligent est considéré comme les veines qui relient tous les organes vitaux de la ville, permettant aux usagers de la voie publique de circuler de nuit avec une sécurité et un confort aussi élevé que possible tout en réduisant le taux de consommation électrique.

LES SIX PILIERS D'UNE SMART CITY

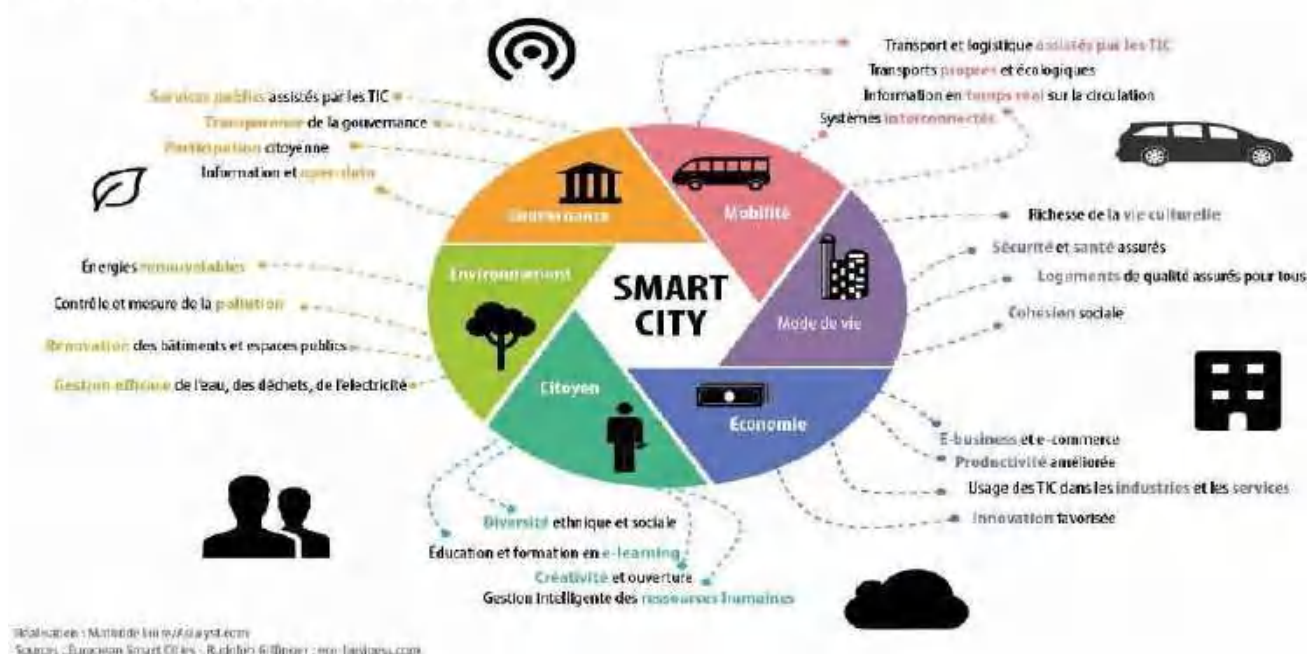


Figure 1.1 – Concept du Smart City

Selon l'expert Rudolf Giffinger, les villes intelligentes peuvent être classées selon six critères principaux liés aux théories régionales et néo-classiques de la croissance et du développement urbain. Ces critères sont respectivement fondés sur les théories de la compétitivité régionale, l'économie des transports et des technologies de l'information et de la communication, les ressources naturelles, les capitaux humains et sociaux, la qualité de vie et la participation des citoyens à la vie démocratique de la ville.

1.2.2 L'éclairage intelligent :

L'éclairage intelligent c'est la capacité pour un système d'éclairage à communiquer avec les utilisateurs par l'intermédiaire d'une interface homme machine et à remonter des informations sur son état et l'état de son environnement, les un système d'éclairage intelligent incorpore toute une panoplie de capteurs qui permettent de comprendre comment lieux utilisés pour réduire la consommation liée à l'éclairage extérieur.



1.2.2.1 Les solutions existantes :

Aujourd'hui, le concept d'éclairage intelligent a le vent en poupe. Il est utilisé, mis en place ou revendiqué par denombreuse entreprise à l'échelle international comme ECOSYS Group, HUWAWEI, PHILIPS LIGHTING... Ensuite on passe à une étude comparative des applications existantes dans le domaine de l'éclairage public avant d'introduire notre solution proposée.

a. Eclairage à commande locale par départ

Les lampadaires sont commandés on/off par un relais placé dans une armoire électrique, les relais sont excités par un dispositif appelé contrôleur d'armoire connecté à un modem qui lui permet d'envoyer et recevoir les informations et les commandes aux lampadaires.

Le point faible de cette solution est qu'on ne peut pas commander les lampadaires chacun tout seul mais seulement par départ ou par groupe.



Figure 1.2 – Eclairage public à commande locale par départ

b. Eclairage public connecté à un système de gestion centralisé :

Cette solution consiste à connecter chaque lampadaire directement à une interface de gestion d'éclairage à distance via un module GSM qui permet de transmettre les informations et les commandes dans les deux sens.

L'inconvénient de cette solution est clairement le cout très élevé pour l'installation d'un module GSM ou une carte Sim pour chaque point lumineux.



Figure 1.3 – Eclairage public connecté à un système de gestion centralisé

c. Eclairage public connecté à commande locale décentralisé

Dans ce mode de contrôle, les points lumineux sont connectés directement à un contrôleur d'armoire (unité de commande) qui connecté à un réseau internet via un réseau sans fil (wifi, Lora, RF) ou un réseau filaire disponible.

Le contrôleur permet de commander on/off chaque point lumineux ou chaque groupe de luminaires, il permet d'envoyer et recevoir dans les deux sens, les informations et les commandes d'éclairages aux point lumineures suivant un ordre de fonctionnement bien déterminé.



Figure 1.4 – Eclairage public connecté à commande locale décentralisé

1.2.2.2 La solution proposée :

Tout d'abord, le système fonctionne seulement pendant la nuit, c'est à dire à partir du coucher du soleil jusqu'à son lever. Pour assurer ce fonctionnement automatique on utilise un capteur de lumière qui consiste à détecter la différence entre le jour et la nuit selon un seuil de luminosité bien déterminé, pour garantir que les lampadaires ne fonctionnent pas au cours de la journée.

Par ailleurs, pendant la nuit et en absences de mouvement des piétons et des véhicules sur la route, c'est inutile de laisser les lampadaires allumés avec leurs intensité maximale et pour cela le système d'éclairage fonctionnera à 15% de la luminosité maximale.

D'une autre part lorsqu'une caméra à vision nocturne capte un mouvement des objets pré identifiés (les piétons ou bien toute type de véhicule) à l'aide d'un algorithme basé sur l'apprentissage en profondeur implémenté dans une carte de commande, le système d'éclairage doit donner l'ordre aux lampadaires d'éclairer avec leurs intensités maximales suivant le mouvement des objets en question, et pour cela il faut installer aussi des détecteurs de mouvement sur chaque lampadaire afin d'assurer le fonctionnement adéquat du système.

Cette solution a pour objectif principale d'optimiser encore plus la consommation d'énergie électrique dans les éclairages publics en commandant les lampadaires d'allumer seulement en cas des évènements bien définis.

1.3 Contexte du projet

1.3.1 Problématique

Dans notre projet on s'intéresse au secteur de l'éclairage public, un service clé fourni par les autorités publiques aux niveaux local et municipal. Un bon éclairage est essentiel pour la sécurité routière, la sécurité personnelle et l'ambiance urbaine. Cependant, de nombreuses installations d'éclairage public sont obsolètes et donc extrêmement inefficaces. Cela conduit à des exigences énergétiques et des niveaux de maintenance plus élevés. L'éclairage public au Maroc représente entre 25 et 30% de la consommation d'énergie selon les estimations de l'ONE, Ce qui constitue, pour les communes, près de «25% de la facture globale d'énergie et 30% à 40% de la facture d'électricité», une proportion deux fois plus importante qu'en Europe.

Ces chiffres nous imposent aujourd'hui à penser qu'il est nécessaire d'utiliser l'éclairage public de façon raisonnable pour aller vers un éclairage plus efficace tout en optimisant la consommation de l'énergie électrique. **L'éclairage public intelligent** est une solution très importante, car il permet de gérer le fonctionnement exact de nos services d'éclairage.

1.3.2 Description du projet

Ce projet porte principalement sur la réalisation d'un modèle d'éclairage public intelligent visant l'optimisation de la consommation d'énergie électrique en commandant les lampadaires de s'allumer à 100% de leurs intensité d'éclairage seulement en cas de circulation des objets bien précis en exploitant les techniques du Deep Learning et de reconnaissance d'images pour détecter et identifier ces objets circulant sur la route pendant la nuit, et cela bien évidemment à l'aide d'une carte de commande (unité de traitement), ainsi que par l'utilisation des capteurs de mouvements attachée à chaque lampadaire et un capteur de lumière pour que le système fonctionne seulement dans la nuit.

1.3.3 Cahier des charges

Le travail demandé par l'entreprise consiste à concevoir et réaliser un prototype d'un système d'éclairage public intelligent à l'aide d'une carte Raspberry pi et par l'implémentation d'un algorithme d'apprentissage automatique de type Deep Learning.

Pour ce faire nous avons élaboré le cahier des charges suivant :

- ✓ Ce prototype permet d'éclairer en fonction des besoins, via des lampadaires et en utilisant un module caméra infrarouge (Picamera) interfacé avec une carte Raspberry pi. Cette caméra est capable de prendre des séquences vidéo pendant la nuit puis traiter ces derniers avec un algorithme basé sur l'apprentissage profond pour identifier les objets qui passent sur la route. Ensuite, et en cas de détection de présence de l'un des objets d'intérêt précis (piéton, voiture, vélo, moto, . . .), la carte Raspberry pi dirige l'action vers les capteurs de mouvements associée à chaque lampadaire pour être prêt à détecter les objets qui passent

devant, puis commander chaque point lumineux à allumer à 100% en sélectionnant les zones à éclairer suivant le mouvement d'objet dans la route, d'autre part si les objets détectés sont pas dans notre classe d'objets d'intérêt (chat, chien, oiseau, . . .), l'éclairage sera réduit à 15% d'intensité maximale.

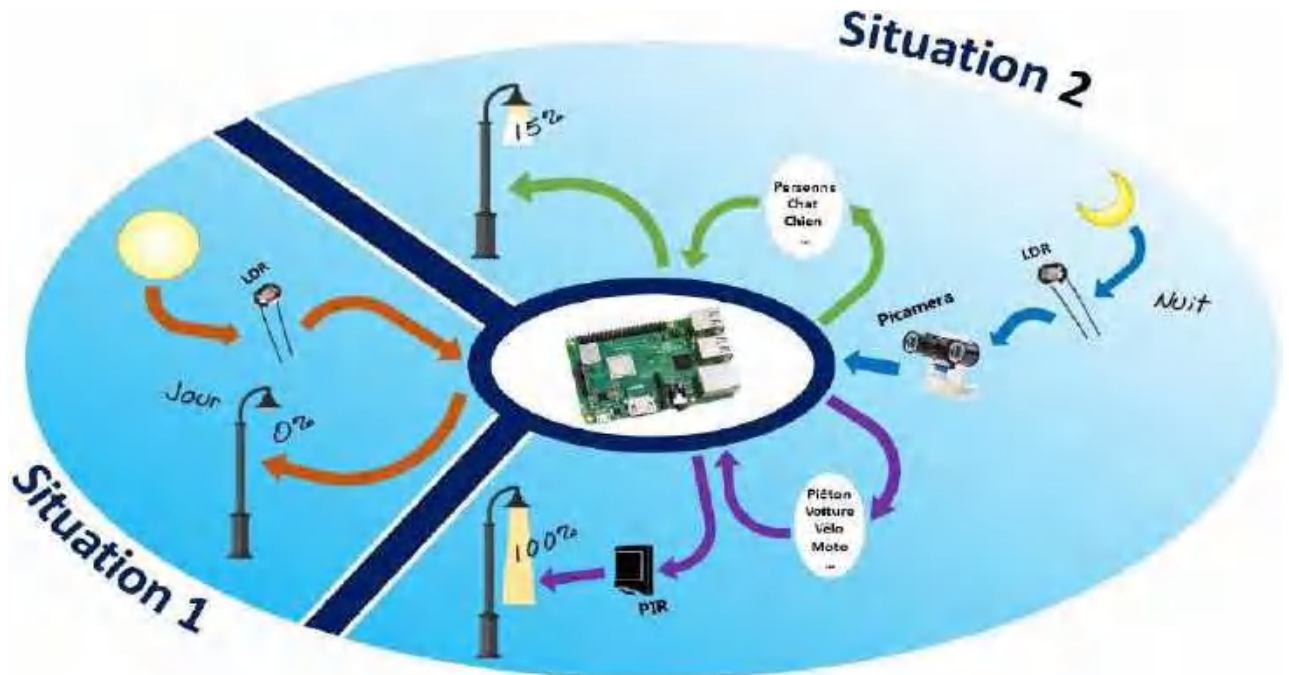


Figure 1.6 – les situations principales du système

1.3.4 Principe de fonctionnement

Pour mieux comprendre le fonctionnement du prototype ainsi que l'utilité de chaque composant voici un schéma descriptif dans la figure 1.7 ci-dessous.



Figure 1.7 – Schéma descriptif du fonctionnement du système

1.3.5 Les objectifs du projet

- Optimiser l'éclairage des rues en fonction de l'éclairage ambiant.
- Commuter vers des solutions technologiques modernes plus intelligentes sous le concept de « SmartCity » qui vise le développement urbain.
- Régler l'éclairage en réponse à des événements prédéfinis, tels que lors de la circulation des piétons, voiture etc...
- Améliorer l'efficacité énergétique en utilisant un éclairage optimal pour réduire les dépenses énergétiques, le cout élevé de la maintenance ainsi que les nuances lumineuses grâce à un éclairage sélectif.

Chapitre 2

1) Généralités sur l'apprentissage en profondeur

En raison de sa relation étroite avec l'analyse vidéo et la reconnaissance d'images, la détection d'objets a attiré l'attention des chercheurs au cours de ces dernières années. Les méthodes traditionnelles de détection d'objets sont construites à partir de caractéristiques artisanales et d'architectures peu profondes pouvant être entraînées. Leur performance stagne facilement en construisant des ensembles complexes qui combinent de multiples caractéristiques d'images de bas niveau avec le contexte de haut niveau issu de détecteurs d'objets et de classificateurs de scènes. Avec le développement rapide de l'apprentissage profond automatisé (en anglais Deep Learning), des outils plus puissants, capables d'apprendre des fonctionnalités sémantiques, de haut niveau et plus profondes, sont introduits pour traiter les problèmes existant dans les architectures traditionnelles. Ces modèles se comportent différemment dans l'architecture de réseau, la stratégie de formation et la fonction d'optimisation, etc.

Ce chapitre présente un bref historique de la reconnaissance et de la détection d'objets, de même pour l'apprentissage profond, suivi d'une explication détaillée du fonctionnement des CNNs et des extracteurs de caractéristiques.

1.1 Bref historique de la reconnaissance et de la détection d'objets

La reconnaissance et la détection d'objets constituent depuis longtemps un défi pour la vision par ordinateur, de nombreuses approches différentes ont été adoptées pour tenter de le surmonter. Certaines de ces approches incluent la mise en correspondance d'aspects visuels d'un objet, tels que les bords, les contours et la couleur, avec des occurrences similaires dans une image, ou l'utilisation de fonctionnalités plus spécifiques pour faire de même.

Pour acquérir une compréhension complète de l'image, nous devons non seulement de concentrer sur la classification des 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 [6]. Elle consiste généralement en différentes sous-tâches telles que la détection de visage [7], la détection de piétons [8] et la détection de squelette [9].

L'un des problèmes fondamentaux de la vision par ordinateur, la détection d'objet qu'est capable de fournir des informations précieuses pour la compréhension sémantique des images et des vidéos. Elle est liée à de nombreuses applications, notamment la classification des images [10], [11], l'analyse du comportement humain [12], reconnaissance du visage [13] et conduite autonome [14], [15]. Parallèlement, héritant des réseaux de neurones et des systèmes d'apprentissage associés, les progrès réalisés dans ces domaines développeront des algorithmes de réseaux de neurones, et auront également un impact considérable sur les techniques de détection d'objets pouvant être considérées comme des systèmes d'apprentissage [16], [17]. Cependant, en raison de fortes variations des points de vue, des occlusions et des conditions d'éclairage, il

est difficile d'effectuer une détection parfaite des objets avec une tâche de localisation d'objets supplémentaire. Beaucoup d'attention a été attirée sur ce domaine ces dernières années [18, [19].

La définition du problème de la détection d'objet consiste à déterminer où se trouvent les objets dans une image donnée (localisation d'objet) et à quelle catégorie appartient chaque objet (classification d'objet). Ainsi, le pipeline de modèles de détection d'objets traditionnels peut être principalement divisé en trois étapes : sélection de la région informative, extraction des caractéristiques et classification.

- **Sélection informative de la région.** Étant donné que différents objets peuvent apparaître à n'importe quelle position de l'image et avoir différents formats d'image ou tailles, il est donc naturel de numériser l'ensemble de l'image à l'aide d'une fenêtre coulissante multi-échelle. Bien que cette stratégie exhaustive puisse identifier toutes les positions possibles des objets, ses lacunes sont également évidentes. En raison du grand nombre de fenêtres candidates, le calcul est coûteux et produit trop de fenêtres redondantes. Cependant, si seul un nombre fixe de modèles de fenêtre glissante est appliqué, des régions non satisfaisantes peuvent être produites ;
- **Extraction de caractéristiques.** Pour reconnaître différents objets, nous devons extraire des caractéristiques visuelles pouvant fournir une représentation sémantique et robuste. Les fonctions SIFT [20], HOG [21] et Haar-like [22] sont représentatives. Cela est dû au fait que ces caractéristiques peuvent produire des représentations associées à des cellules complexes dans le cerveau humain [20]. Cependant, en raison de la diversité des apparences, des conditions d'éclairage et des arrière-plans, il est difficile de concevoir manuellement un descripteur de caractéristiques robuste pour décrire parfaitement plusieurs types d'objets ;
- **La classification.** En outre, un classificateur est nécessaire pour distinguer un objet cible de toutes les autres catégories et pour rendre les représentations plus hiérarchiques, sémantiques et informatives pour la reconnaissance visuelle. Habituellement, la machine à vecteurs supportée (SVM) [23], AdaBoost [24] et le modèle à base de pièce déformable (DPM) [25] sont de bons choix. Parmi ces classificateurs, le DPM est un modèle souple combinant des parties d'objet avec un coût de déformation permettant de traiter des déformations sévères. Dans DPM, à l'aide d'un modèle graphique, des fonctions de bas niveau soigneusement conçues et des décompositions de pièces inspirées par la cinématique sont combinées.

Avant 2012, la principale méthode utilisée pour la détection d'objet était basée sur l'utilisation de la correspondance des caractéristiques. Dans de nombreux cas, ces caractéristiques étaient fabriquées à la main. Un exemple de méthode populaire est la transformation de fonction évolutive à l'échelle (SIFT) [27], Il est capable de reconnaître des objets connus dans des images et a résolu de nombreux problèmes d'appariement de caractéristiques avec des variations d'échelle et de rotation. Histogrammes de dégradés orientés (HOG) [28] est une méthode utilisant une approche similaire, bien qu'elle s'intéresse davantage aux contours qu'aux points caractéristiques spécifiques. HOG a été utilisé avec succès pour des défis tels que la détection de piétons, bien qu'il fonctionne moins bien pour les objets déformables et les personnes dans

des situations plus variées. Une solution à cela a été introduite avec le modèle de pièces déformables (DPM) [29]. Au lieu d'utiliser un seul grand modèle pour localiser un objet, comme le fait HOG, plusieurs modèles sont utilisés pour diverses parties d'objet ainsi que pour l'objet de base. Avant 2012, DPM était à la pointe de la performance en matière de référence telles que ImageNet.

Grâce à l'émergence des réseaux de neurones profonds (DNN) [26], un gain plus significatif est obtenu avec l'introduction des régions avec fonctions CNN (R-CNN) [28]. Les DNN, ou les CNN les plus représentatifs, agissent de manière tout à fait différente des approches traditionnelles. Ils ont des architectures plus profondes avec la capacité d'apprendre des caractéristiques plus complexes que celles peu profondes. De plus, l'expressivité et les algorithmes d'apprentissage robustes permettent d'apprendre des représentations d'objets informatives sans avoir à concevoir manuellement des fonctionnalités [32].

Depuis la proposition de R-CNN, de nombreux modèles améliorés ont été proposés, y compris Fast R-CNN qui optimise conjointement les tâches de classification et de régression du cadre de sélection [29], Faster R-CNN qui utilise un sous-réseau supplémentaire pour générer des propositions de région [31], et YOLO qui effectue la détection d'objet via une régression à grille fixe [30]. Tous apportent différents degrés d'amélioration des performances de détection par rapport au réseau principal R-CNN et permettent de réaliser une détection d'objets en temps réel et précise. Puis, en 2012, AlexNet [26], un modèle de reconnaissance d'objets basé sur CNN, a participé au défi annuel d'ImageNet. Il a largement surperformé ses concurrents, atteignant 15,3% d'erreurs dans le top 5, contre 26,2% pour la deuxième meilleure place. C'était à bien des égards l'avènement des CNN, et depuis lors, les CNN sont devenus immensément populaires. Les CNNs, cependant, n'étaient pas vraiment quelque chose de nouveau. Un exemple important en est LeNet de 1998 [30], qui est un modèle utilisé pour lire des caractères manuscrits dans les codes postaux. L'augmentation de la puissance de calcul des ordinateurs et l'augmentation des données disponibles ont été utilisées pour expliquer la résurgence des CNN en 2012.

1.2 Bref aperçu de l'apprentissage en profondeur - Deep Learning

Les modèles profonds peuvent être appelés réseaux de neurones à structures profondes. L'histoire des réseaux de neurones peut remonter aux années 1940 [33]. L'intention initiale était de simuler le système cérébral humain pour résoudre les problèmes d'apprentissage généraux de manière raisonnée. C'était populaire dans les années 1980 et 1990 avec la proposition de l'algorithme de rétropropagation de Hinton et al. [34]. Cependant, en raison du problème de surapprentissage, du manque de données pendant la phase d'entraînement du modèle, de la puissance de calcul limitée et de la faiblesse des performances par rapport à d'autres outils d'apprentissage automatique, les réseaux de neurones sont devenus obsolètes au début des années 2000.

L'apprentissage en profondeur est devenu populaire depuis 2006 [35] avec une percée dans la reconnaissance de la parole [36]. La récupération de l'apprentissage en profondeur peut être attribuée aux facteurs suivants :

- L'émergence de données de formation annotées à grande échelle, telles qu'ImageNet [37], afin de démontrer pleinement sa très grande capacité d'apprentissage.
- Développement rapide des systèmes informatiques de calcul en parallèles hautes performances, tels que des grappes de GPU.
- Avancées significatives dans la conception des structures de réseau et des stratégies de formation. Avec un pré-entraînement non supervisé et par couche guidé par un codeur automatique (AE) [38] ou une machine Boltzmann restreinte (RBM) [39], une bonne initialisation est fournie. Avec l'abandon et l'augmentation des données, le problème de sur-apprentissage en entraînement a été résolu [11], [40]. Avec la normalisation par lots (BN), la formation des réseaux de neurones en profondeurs devient très efficace [41]. Parallèlement, diverses structures de réseau, telles que AlexNet [11], Overfeat [42], GoogLeNet [43], VGG [44] et ResNet [46], ont fait l'objet de nombreuses études pour améliorer les performances.

1.3 Réseaux de Neurones Convolutifs CNN

Cette partie donne une introduction aux CNN. Tout d'abord, une explication du fonctionnement de CNN est donnée, avant de présenter des exemples d'extracteurs de caractéristiques et d'architectures de détection d'objets basés sur CNN.

1.3.1 Structure des réseaux de neurones convolutionnels

Les CNNs sont principalement composés de 4 éléments principaux : les couches de convolution, les fonctions d'activation et couches de regroupement (pooling layers) et finalement la couche entièrement connectée.

1.3.2 Couches de convolution

Dans une couche de convolution, la mission principale est l'extraction des caractéristiques pertinentes dans l'image et pour cela, des convolutions sont effectuées entre les filtres de la couche et la matrice introduite dans la couche. Ceci est similaire à l'utilisation d'un filtre pour détecter les contours. La différence est que, pour la détection des contours, les filtres sont réalisés manuellement, alors que pour les CNN, les filtres sont détectés en résolvant un problème d'optimisation, et que les CNN contiennent généralement de nombreux filtres. Les valeurs dans ces filtres s'appellent des poids.

Les filtres sont plus petits que l'entrée et passent d'une entrée à l'autre via une fenêtre glissante [47]. Un exemple de l'approche de convolution et de fenêtre glissante est présenté à la figure 2.1. Le nombre de filtres dans un calque, la taille des filtres et la foulée avec laquelle les filtres sont déplacés entre les convolutions sont des hyperparamètres définis par l'utilisateur. Dans ce contexte, un hyperparamètre est un paramètre dont la valeur est définie avant l'ajustement d'un modèle aux données, tandis que d'autres paramètres sont dérivés lors de la formation et l'entraînement du modèle [48]. La sortie d'une couche de convolution est souvent appelée carte de caractéristiques (features map).

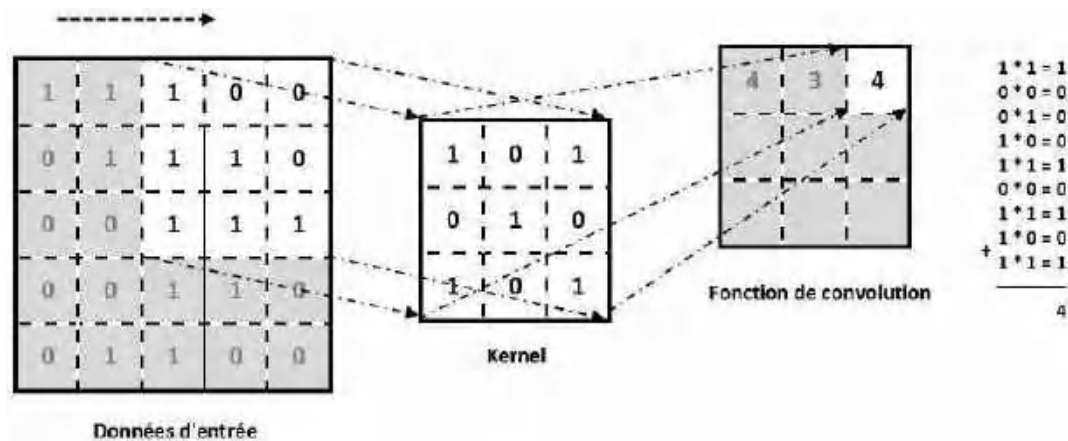


Figure 2.1 – l'opération de convolution

Pour une entrée tridimensionnelle, telle qu'une image RVB avec 3 couches de canaux, la taille du filtre est généralement définie par la hauteur et la largeur, tandis que la profondeur du filtre est implicitement égale à la profondeur de l'entrée. Les filtres ont tendance à être carrés avec une hauteur et une largeur avec un nombre impair, tels que 3×3 , 5×5 ou 7×7 , de sorte qu'il y a un pixel central dans le filtre.

En raison de la manière dont les convolutions entre l'entrée et le filtre sont effectuées, les informations contenues près des bords et des coins de l'entrée ont moins d'impact sur la sortie du calque. Pour atténuer ce problème, un remplissage nul de la représentation binaire de l'image d'entrée est souvent utilisé. Dans de tels cas, souvent appelés le même remplissage, l'entrée est complétée par des zéros afin que la sortie du calque ait la même hauteur et la même largeur que l'entrée [47]. Un exemple de ceci peut être vu dans la figure 2.2 suivante.

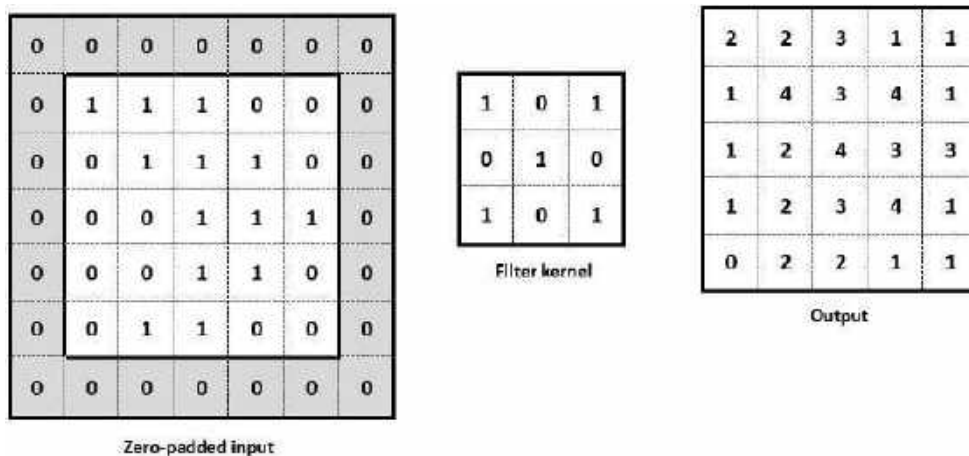


Figure 2.2 – Entrée complétée de zéro (à gauche), avec un filtre 3×3 (moyen), pour obtenir une sortie ayant la même taille que l'entrée non complétée de zéro (à droite)

1.3.2.1 Couche d'activation

En générale il existe diverses fonctions d'activation, mais pour les CNN, une unité linéaire rectifiée (ReLU), représentée à la figure 2.3, est généralement utilisée dans la plupart des applications de réseaux de neurones [47]. Cette fonction d'activation qui est appliquée élément par élément, permet de rendre le réseau non linéaire. Cette non-linéarité est ce qui permet au réseau de neurones de modéliser des problèmes complexes. La fonction d'activation joue également un rôle important dans la formation et l'entraînement efficace du réseau de neurones.

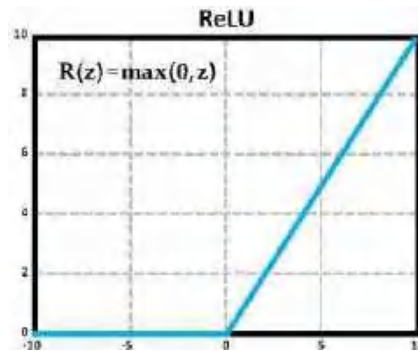


Figure 2.3 – La fonction d'activation ReLU

1.3.2.2 Couche de Pooling

Dans le bloc de construction final, on trouve la couche de pooling. La taille spatiale des données est réduite, souvent avec une taille de filtre de regroupement de 2×2 . Cette opération contribue à rendre le modèle plus robuste et moins affecté par les petites variations des données d'entrée [49]. L'opération de pooling, trouve une statistique résumée des emplacements voisins, en les combinant en une valeur. Généralement le max pooling est le type de pooling le plus utilisé, où la valeur maximale à l'intérieur du filtre de pooling est récupérée. Un exemple de cette opération de mise en commun est présenté à la figure 2.4. Parmi les autres opérations de mise en commun pouvant être utilisées, on peut citer la moyenne (average pooling), la norme L2 ou une moyenne pondérée. Il convient de noter que la mise en commun n'est généralement pas effectuée sur toute la profondeur des données.

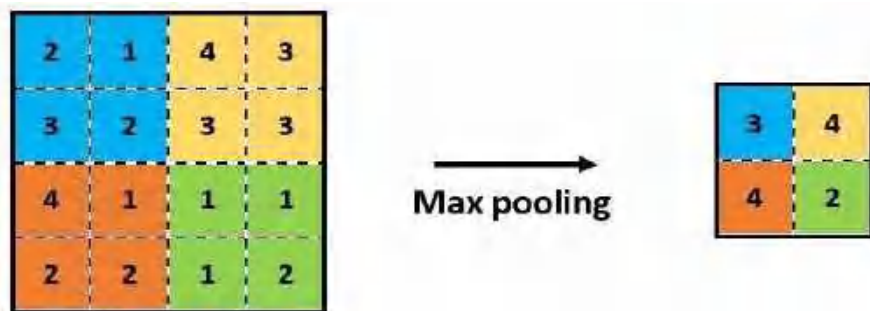


Figure 2.4 – Un exemple de l'opération de max pooling, utilisant une taille de mise en commun de 2×2

1.3.2.3 Couches Entièrement Connectées

L'objectif des couches entièrement connectées est de mapper des combinaisons d'entités de haut niveau avec des probabilités de classe. Ces couches sont souvent ajoutées à la fin des CNN utilisés pour la classification et ont plus ou moins la même structure qu'un réseau neuronal à rétroaction standard. La sortie des parties de convolution et de max pooling du CNN sont modifiées de la 3D à la 2D, ou de la 2D à la représentation unidimensionnelle avant d'être transmise à des couches de neurones entièrement connectées. La fonction d'activation utilisée dans ces neurones sera souvent la même que celle utilisée avec les couches de convolution, à l'exception de la couche de sortie qui est conçue pour la classification. Pour la classification, où l'entrée ne représente qu'une classe, la fonction d'activation softmax est utilisée. La fonction softmax est une fonction logistique, est utilisé pour convertir un score en probabilité dans un contexte de classification multi-classe.

1.3.3 Visualiser les réseaux de neurones convolutionnels

Comprendre les CNNs, cependant, n'est pas si facile. Même en connaissant les fondements mathématiques sur lesquels ils sont basés, ils peuvent apparaître comme des boîtes noires. L'une des explications souvent données sur le fonctionnement des réseaux de neurones est la manière dont les premières couches détectent des caractéristiques simples telles que les couleurs les contours et textures, tandis que les couches ultérieures combinent des fonctionnalités antérieures en des fonctionnalités de plus en plus complexes. Un exemple de ceci peut être vu dans la figure 2.5.

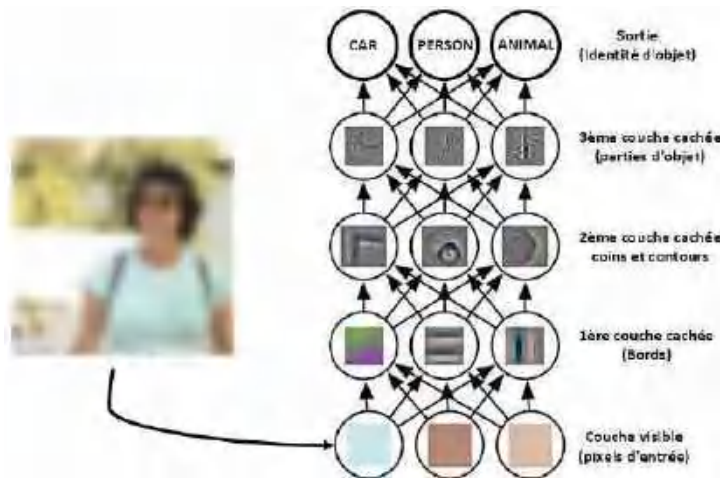


Figure 2.5 – Un exemple de la façon dont les premières couches d'un réseau neuronal détectent des caractéristiques simples, telles que les couleurs et les lignes, tandis que les couches ultérieures les combinent en des caractéristiques de plus en plus complexes

Il est intéressant de voir ce qui déclenche différents filtres dans différentes couches d'un CNN, en calculant le type d'entrée générant une sortie élevée d'un filtre. Un exemple de Chollet [50], illustré à la figure 2.6, montre ceci pour certains filtres des 5 premières couches de VGG-16 (exemple d'une architecture basée sur CNN), un extracteur de caractéristiques décrit plus en détail à la sous-section 2.3.4. Il est donc clair que ces dernières couches sont déclenchées par une combinaison d'entités qui déclenchent les couches précédentes.

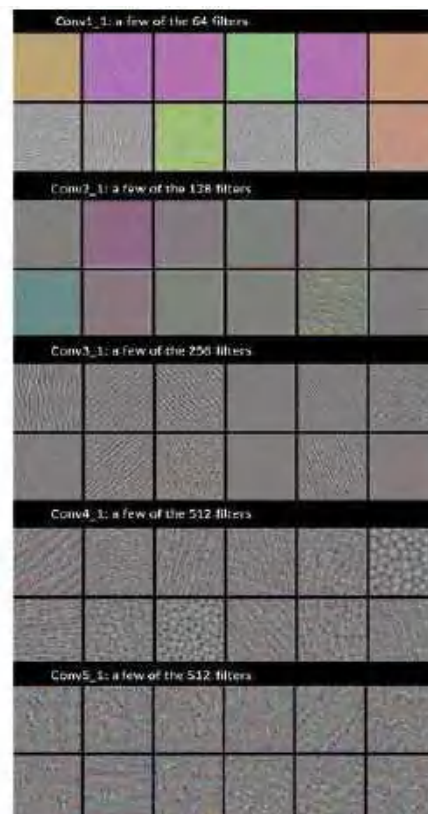


Figure 2.6 – Une visualisation de l'entrée préférée de certains filtres dans les 5 premières couches du VGG-16

Bien que les CNN soient capables de mapper correctement les images sur des classes probables, Ils n'ont pas la même capacité de reconnaissance que dans les réseaux de neurones du système cérébrale humaine. Ils cartographient simplement des combinaisons de caractéristiques diverses, que ce soit la couleur, les textures, les contours ou autres [50]. Cela peut être illustré en générant des images synthétiques de ce qu'un CNN considère comme différentes classes. Un exemple pour le poivron, le citron et le Husky est présenté à la figure 2.7. Dans une certaine mesure, je peux, en tant qu'être humain, convenir qu'il s'agit d'exemples de poivron, de citron et de Husky, mais ce n'est pas ce que je visualiserais. Le fait que l'image de la figure 2.7 soit une pie est cependant plus difficile à avaler.

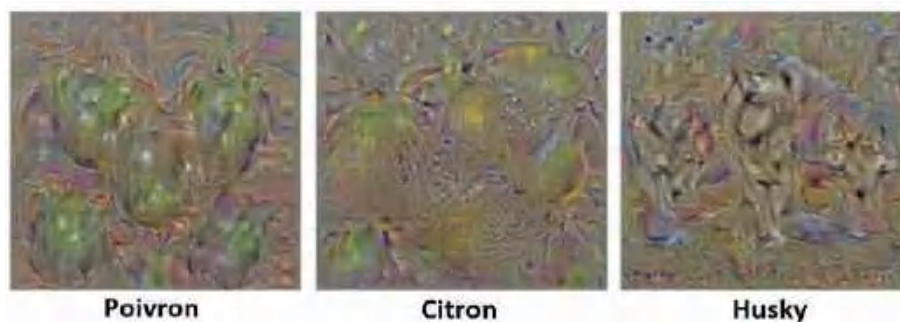


Figure 2.7- Des images synthétiques générées de ce qu'un CNN considère comme étant le poivron (à gauche), le citron (au milieu) et le husky (à droite)

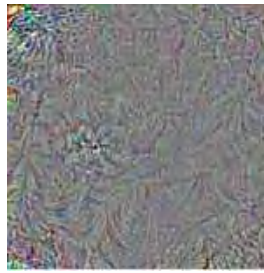


Figure 2.8 – Une image synthétique générée d'une pie, avec 99,99% de confiance

Bien qu'ils ne soient pas parfaits, on ne peut pas nier l'efficacité des CNN pour la reconnaissance d'objets. Les CNN ont des performances de pointe dans les repères de reconnaissance d'objets. Ils constituent actuellement la meilleure option en matière de précision dans les problèmes de reconnaissance d'objets multi-classes. Pour cette raison, un modèle basé sur CNN était préféré dans ce projet.

1.3.4 Entraînement des réseaux de neurones

Lorsqu'il s'agit de former des CNN, l'entraînement consiste à déterminer et à calculer empiriquement la valeur de chacun de ses poids. Le principe est le suivant : le CNN traite une image (de la base de données d'entraînement) et en sortie il fait une prédiction, c'est-à-dire qu'il dit à quelle classe il pense que cette image appartient. Sachant qu'on connaît préalablement la classe de chacune des images d'entraînement, on peut vérifier si ce résultat est correct.

En fonction de la véracité de ce résultat, on met à jour tous les poids du CNN selon un algorithme qui s'appelle la rétropropagation du gradient de l'erreur.

Lors de la phase d'entraînement du modèle, le processus expliqué ci-dessus est répété plusieurs fois et avec la totalité des images de la base de données d'entraînement. Le but étant que le modèle classifie au mieux ces données avec une précision maximale.

Lorsque le modèle a fini de mettre à jour ses poids, on évalue le modèle en lui présentant la base de données de validation. Il classe toutes ces images (qui sont des images que le modèle n'a jamais vues) et on calcule son taux de bonne classification, c'est ce qu'on appelle la précision du modèle.

Ce problème d'optimisation des performances est généralement résolu en utilisant une descente de gradient. Avec la rétropropagation, le gradient descente est propagé par les mots clés à travers les différentes couches du réseau, à l'aide de la règle de chaîne et les pondérations des filtres, des neurones sont mis à jour [47]. Étant donné que les réseaux de neurones sont souvent formés à des jeux de données très volumineux, et même qu'ils requièrent de très grands jeux de données, le calcul du gradient en fonction du jeu de données complet peut prendre du temps. Pour cette raison, des méthodes telles que la descente de gradient stochastique et la descente de gradient en mini-batch sont souvent préférées. Avec la descente de gradient stochastique, le gradient est calculé pour la rétropropagation effectuée pour chaque échantillon de données, tandis que pour la descente de gradient en mini-lot, la même chose est effectuée mais sur la base d'un petit lot d'échantillons de données.

1.3.5 Extracteurs des caractéristiques

Depuis 2012 et par la recherche, de nombreuses architectures d'extracteurs des caractéristiques ont été élaborées, avec des valeurs de précision sans cesse croissantes sur les jeux de données de référence. Deux principaux extracteurs des caractéristiques dans les images, LeNet et AlexNet, ont déjà été mentionnés à la section 1.

Bien que tous les extracteurs de caractéristiques mentionnés soient construits en utilisant les mêmes blocs de construction de base mentionnés dans la sous-section 2.3.1, la taille et le nombre de filtres utilisés, ainsi que le nombre de couches, diffèrent considérablement. Certains de ces extracteurs de caractéristiques introduisent également des structures et des couches uniques.

VGG [53] se distingue par le fait qu'il a commencé à utiliser des couches empilées de filtres 3×3 , plutôt que les filtres 9×9 et 11×11 utilisés dans AlexNet. Dans l'article décrivant VGG, on affirme que cela rend le réseau plus discriminatoire, réduit le nombre de paramètres et impose une certaine régularisation. Cette approche est également utilisée par les extracteurs de fonctionnalités ultérieurs. La version de VGG la plus couramment utilisée est le VGG-16, qui comporte 16 couches.

Avec le NIN (Réseau dans le réseau) [54], il a été suggéré que 1×1 convolutions pourrait être utile en combinant des fonctionnalités de niveau supérieur après la réalisation des convolutions.

GoogLeNet [55] utilise des convolutions de 1×1 pour réduire la complexité de calcul d'opérations qui seraient autrement trop coûteuses, pour cela GoogLeNet nécessite beaucoup moins d'opérations par rapport à VGG.

Avec Inception V2 [57], des couches de normalisation par lots ont été introduites. Dans ces couches, la sortie d'une couche convolutionnelle est normalisée. Lorsque toutes les couches répondent dans le même intervalle de valeurs, ce qui réduit le temps d'apprentissage.

ResNet [56] a introduit l'idée d'utiliser un contournement pour ignorer les couches. Cela sert à deux fins. Premièrement, il permet la combinaison de fonctions de niveau plus basses et plus hautes. Deuxièmement, et peut-être la plus important, cela rend l'apprentissage plus efficace et permet aux réseaux d'être encore plus profonds et complexes et généraliste.

Un dernier extracteur de caractéristiques qui doit être mentionné, est MobileNet [58]. MobileNet a été conçu pour les appareils mobiles avec ressources matériels limités. Il utilise certaines des innovations mentionnées auparavant, telles que les convolutions 1×1 et la normalisation par lots. Bien qu'il ne soit pas aussi précis que les nouveaux extracteurs de fonctions comme Inception V3, ResNet et YOLO V3 il est capable d'obtenir des résultats similaires à GoogLeNet et VGG-16 avec beaucoup moins d'opérations et de paramètres. Cela rend MobileNet le modèle d'apprentissage le plus optimale pour notre projet.

1.3.6 Détection des Objets avec les CNNs

Avec les CNNs, beaucoup de progrès ont été réalisés, non seulement avec la reconnaissance d'objets, mais aussi la reconnaissance vocale, traitement du langage naturel, etc... De multiples architectures (méta-architectures) pour la détection d'objets ont été développées et itérées. Le terme méta-architecture est utilisé pour désigner les architectures de détection d'objets qui utilisent des approches similaires pour la détection [59].

L'une des premières méta-architectures proposées était R-CNN [60]. R-CNN utilise un algorithme de recherche sélective pour trouver les régions où les objets d'intérêt peuvent se localiser. Un encadrage d'images de chaque région est ensuite effectué et passé à travers un CNN pour extraire les caractéristiques. Enfin, une machine à vecteurs de support SVM, est utilisée pour déterminer si la boîte de détection contient un objet et quel type d'objet il s'agit pour le classifier. Cependant on rencontre deux problèmes majeurs avec R-CNN, c'est qu'il est lent, souvent avec de nombreux calculs en double, et que, comme il est constitué de 3 parties distinctes, il peut être difficile de s'entraîner [61]. La figure 2.9 présente un aperçu de l'architecture R-CNN.

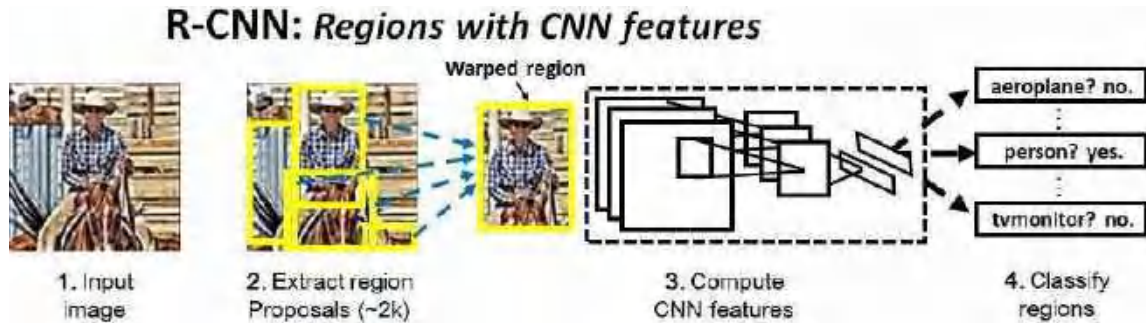


Figure 2.9 – Un aperçu de l'architecture R-CNN

Pour résoudre ces problèmes, Fast R-CNN [62], et plus tard Faster R-CNN [63], ont été développés. Pour Fast R-CNN, le principal changement réside dans le fait que l'ensemble de l'image est transmis une seule fois par CNN, avant que les caractéristiques ne soient extraites de l'espace de l'objet résultant. De cette façon, les entités contenues dans les régions où les objets se chevauchent ne sont calculées qu'une seule fois. De plus, les trois parties de R-CNN ont été réunies et formées en une seule fois. La Figure 2.10. présente une vue d'ensemble de l'architecture Fast R-CNN. Pour Faster R-CNN, le principal changement concerne la manière dont les propositions des régions sont trouvées ou sélectionnées. Au lieu d'utiliser la recherche sélective, un CNN appelé réseau de proposition de région (RPN) est utilisé. Ce réseau prédit les régions d'intérêt en fonction des caractéristiques déterminées par un extracteur de fonctionnalités, des fonctionnalités calculées lors de la classification des objets contenu dans l'image, entraînant une rapidité au niveau de temps d'apprentissage.

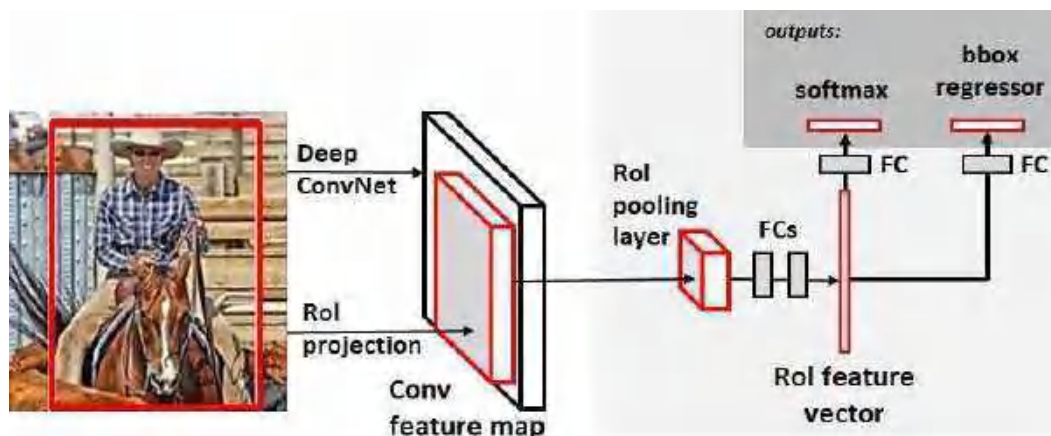


Figure 2.10 – Un aperçu de l'architecture Fast R-CNN

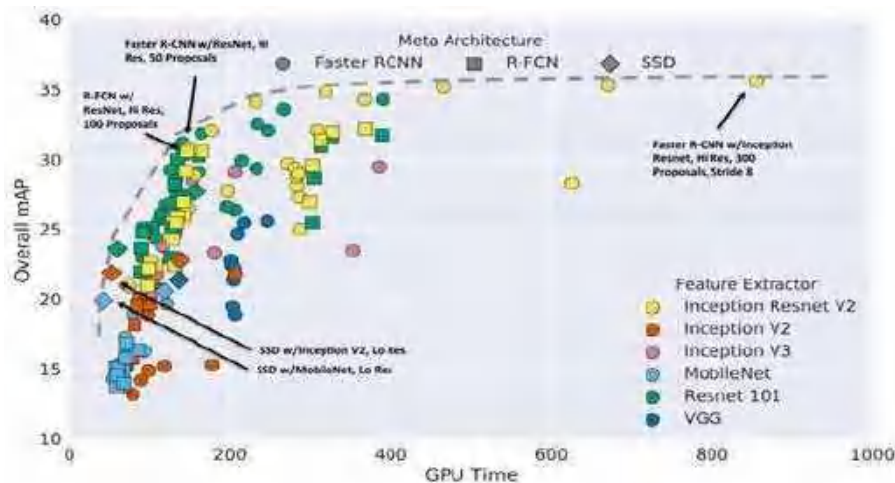


Figure 2.12- Précision en fonction du temps, avec méta-architecture d'indication de formes de marqueur et extracteur d'indicateur de couleurs

Il convient de noter que lorsque Single shot Detector a été publié, YOLO en était à sa première version et a connu une performance inférieure à celle du SSD dans toutes les métriques. Cependant avec YOLOv2 [67] et YOLOv3 [68], YOLO a connu des améliorations significatives et semble DE surpasser les performances du SSD dans de nombreux cas.

Pour une utilisation dans ce projet, cependant, la vitesse et la complexité de calcul est peut-être la métrique la plus importante. Alors que le modèle YOLO le plus léger, Tiny YOLO, affiche un débit d'images impressionnant de 244 FPS, une passe à travers le modèle nécessite 5,41 milliards d'opérations en virgule flottante [69]. Les disques SSD utilisant MobileNet, n'exigent que 1,2 milliard de multiplications et d'ajouts [58]. Cependant, comparé à la faible puissance de calcul d'un Raspberry Pi, même ces modèles légers semblent lourds. Pour cette raison, une approche différente doit être adoptée dans le cadre de détection d'objet implémenté.

Table 2 - Comparaison entre l'architecture du modèle SSD et celui du modèle YOLO

SSD	YOLO
Single Shot Detector	You Only Look Once
Exécute un réseau convolutif sur des images d'entrée en une seule fois et calcule une carte de caractéristiques.	Une technique open source de détection d'objets qui reconnaîtra rapidement les objets dans les images et les vidéos
Le SSD pourrait être un meilleur choix car nous avons tendance à être capables de l'exécuter sur une vidéo et donc le vrai compromis est extrêmement modeste.	YOLO est une meilleure option lorsque l'exactitude n'est pas trop inquiétante mais que vous voulez aller très vite
Lorsque la taille de l'objet est minuscule, les performances diminuent un peu	YOLO pourrait être un choix plus élevé même lorsque la taille de l'objet est petite.
Exécute un réseau convolutif sur l'image d'entrée une seule fois et calcule une carte de caractéristiques	Peuvent être appliqués pour les applications ainsi que pour l'intelligence artificielle, les voitures autonomes et les approches de reconnaissance du cancer.

1.3.7 Eviter le sur-Apprentissage

Comme déjà décrit auparavant, le problème de sur-apprentissage survient lorsqu'on possède un grand nombre du paramètre dans le réseau, et c'est le cas pour un réseau de neurones profonds. Pour éviter, et détecter ce problème de sur-apprentissage, des méthodes ont été proposées :

- La validation croisée :

Pour détecter un sur-apprentissage, on sépare les données en deux sous-ensembles : l'ensemble d'apprentissage et l'ensemble de validation. L'ensemble d'apprentissage comme son nom l'indique permet de faire évoluer les poids du réseau de neurones avec par exemple une rétro propagation. L'ensemble de validation n'est pas utilisé pour l'apprentissage mais permet de vérifier la pertinence du réseau avec des échantillons qu'il ne connaît pas. On peut vraisemblablement parler de sur-apprentissage si l'erreur de prédiction du réseau sur l'ensemble d'apprentissage diminue alors que l'erreur sur la validation augmente de manière significative. Cela signifie que le réseau continue à améliorer ses performances sur les échantillons d'apprentissage mais perd son pouvoir de prédiction sur ceux provenant de la validation.

- La régularisation :

L'une de ces techniques concerne la quantité de données utilisée pour l'entraînement du modèle. Plus on dispose de données uniques, moins il est probable qu'il y ait de sur-apprentissage. Dans de nombreux cas, cependant, une large quantité de données n'est pas disponibles. Dans certains cas, l'augmentation des données peut être satisfaite par exemple pour les images, cette augmentation peut se représenter par le renversement horizontal ou vertical, des rotations à divers degrés, un changement de teinte et de saturation, un flou, un renforcement et un rognage ...

Cependant la méthode de régularisation la plus utilisée est le « Dropout », cette méthode consiste à activer et désactiver les neurones aléatoirement, dans le cadre d'entraînements successifs. Une fois les séries d'entraînements terminées, on rallume tous les neurones et on utilise le réseau comme d'habitude. Cette technique a montré non seulement un gain dans la vitesse d'apprentissage, mais en déconnectant les neurones, on a aussi limité des effets marginaux, rendant le réseau plus robuste et capable de mieux généraliser les concepts appris.

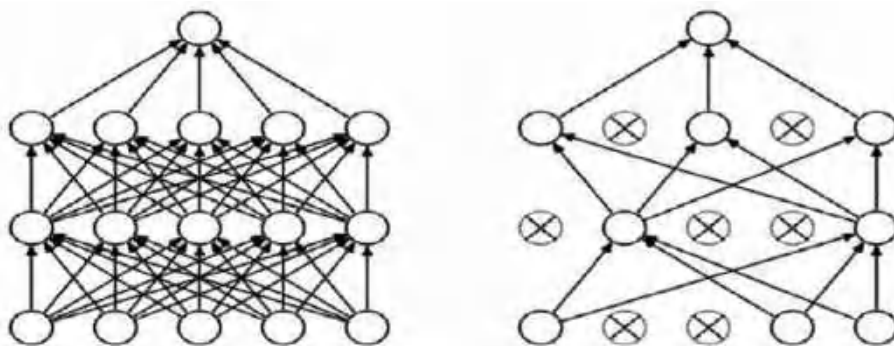


Figure 2.13: Principe de la méthode de Dropout

La technique du dropout est notamment utilisée dans les systèmes de reconnaissance d'images, de voix, le classement de documents et sur des problèmes de calculs en biologie.

2) Présentation du modèle MobileNet-SSD

Il n'y a pas si longtemps, on parlait des MobileNet, de la reconnaissance d'image en temps réel. Son application la plus connue à l'heure actuelle est l'identification de " boxing ", i.e. trouver dans une image tous les objets et toutes les personnes pour les encadrer et les mettre en évidence.

Dans cette partie dédiée aux MobileNet-SSD, nous allons voir comment l'algorithme de classification MobileNet a été combiné avec le framework Single Shot multibox Detector(SSD) pour donner le modèle de Deep Learning MobileNet-SSD. Ce dernier pourra être facilement transformé en une application Android et être embarqué dans n'importe quel smartphone (qu'il soit récent ou non), voire à un Raspberry Pi, pour fournir un outil extrêmement puissant d'analyse d'image en temps réel. Et puisque dans ce projet on va l'implémenter dans un Raspberry Pi, on va essayer de donner une vision générale sur le RPi.

2.1 Détection des Objets avec MobileNets

Comme nous avons déjà vu dans le chapitre précédent, la reconnaissance d'image est l'enjeu majeur du Deep Learning (les algorithmes d'apprentissage à plusieurs niveaux) car les champs de ses applications sont innombrables. Néanmoins, entraîner un algorithme lourd ou nonquantifié et surtout l'utiliser est très coûteux en ressources, car il faut traiter des dizaines de milliers d'images dans des dizaines de milliers de fois . . . Sauf si on utilise une architecture qui est optimisé en termes de complexité de calcul ou de traitement arithmétique comme le MobileNet.

Pour des services ponctuels en ligne, cela ne pose pas de problème dans la plupart des applications. Mais pour de l'analyse des enregistrements vidéo en temps réel, comme par exemple dans une voiture autonome, le fait de pouvoir identifier les objets à savoir les piétons, la route, les voitures, il est critique et indispensable d'adopter un modèle d'apprentissage profonds pour le traitement d'images.

Heureusement, l'architecture MobileNets, des réseaux de neurones convolutionnels (CNN), viennent de faire leur apparition dans la librairie TensorFlow. Et parmi leurs avantages c'est qu'il est :

- Extrêmement légers et petits (en termes de ligne de code et de poids des modèles).
- Assez rapide.
- Précision supérieure à 80%.
- Dédiés à l'embarqué et aux smartphones, pour déporter les calculs.

Ainsi que des utilisations variées, tournant autour de la reconnaissance/classification des images :

- Détection d'objet (personne, bus, avion, voiture, chien, chat, écran TV, etc....).
- Visages (attributs d'un visage notamment).
- Classification " fine-grained ", c'est-à-dire entre des espèces d'une même race (chien, chat, rouge-gorge, ...).
- Reconnaissance de lieux et de paysages.

2.1.1 Le principe de fonctionnement de MobileNet

Tout d'abord il faut voir la section 3 du chapitre 2, pour comprendre le fonctionnement des CNN puisque le modèle MobileNet est basé sur l'architecture des réseaux de neurones convolutifs et plus précisément la convolution séparable en profondeur.

La convolution séparable en profondeur est ainsi nommée parce qu'elle ne traite pas seulement des dimensions spatiales, mais également de la dimension de profondeur - le nombre de canaux. Une image d'entrée peut avoir 3 canaux : RGB. Après quelques convolutions, une image peut avoir plusieurs canaux. Vous pouvez visualiser chaque canal comme une interprétation particulière de cette image ; dans par exemple, le canal « rouge » interprète l'échelle « rouge » de l'image, de même pour le vert et le bleu.

Semblable à la convolution spatiale séparable, une convolution séparable en profondeur divise un noyau en 2 noyaux distincts qui effectuent deux convolutions : la convolution en profondeur et la convolution ponctuelle.

— Une convolution « profonde » (depthwise convolution) de taille 3×3

Elle sert à appliquer un filtre unique à chaque canal d'entrée (un canal d'une image représente un niveau de couleur de RGB, il y en a donc trois dans la majorité des cas). Elle diffère de la convolution classique qui utilise un filtre (kernel) de dimension $3 \times 3 \times 3$, m fois, avec m est le nombre de canal de l'image résultante.

— Une convolution « ponctuelle » (pointwise convolution) de taille 1×1

Cette couche est la deuxième phase d'une convolution séparable en profondeur.

Concrètement, elle applique un filtre (kernel) 1×1 , ou un noyau qui itère sur chaque pixel de l'image original. Ce kernel a une profondeur correspondant au nombre de canaux de l'image d'entrée ; dans notre cas est 3. Par conséquent, nous itérons un kernel $1 \times 1 \times 3$ à travers notre image d'entrée, pour obtenir après une image avec une profondeur de dimension 1.

Puis après on peut multiplier le nombre de kernel utilisé pour avoir une image à la sortie avec le nombre de canal désirés pour modifier le nombre de paramètres du réseau de neurones.

La figure 3.1. Présente une comparaison entre les filtres de convolution standard et les filtres de convolution séparable.

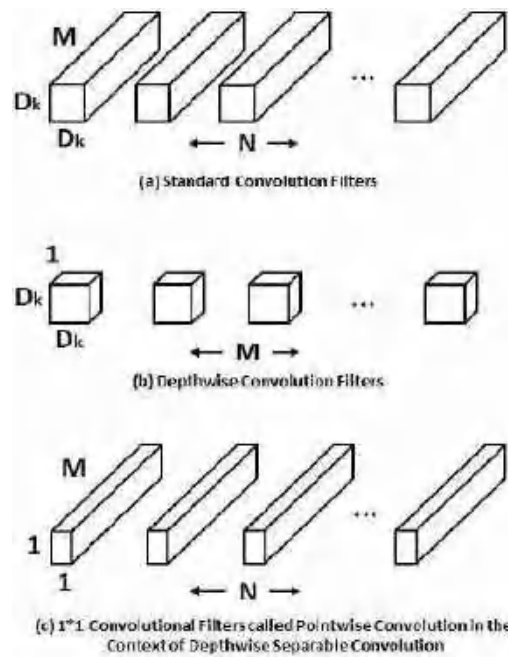


Figure 3.1– Les filtres de convolution standard en (a) sont remplacés par deux couches : convolution en profondeur en (b) et convolution ponctuelle en (c) pour construire un filtre séparable en profondeur

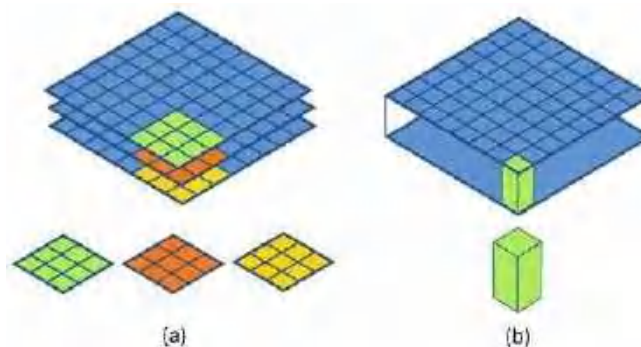


Figure 3.2– filtres convolutionnels en profondeur en (a), et filtres convolutionnels ponctuels en (b)

2.1.2 L'avantage d'utilisation du modèle MobileNet

Maintenant qu'on a dit tout ça, ce n'est pas évident de comprendre en quoi « c'est mieux ». La réponse est toute simple : ce modèle de MobileNets (abrégé en MB) requiert largement moins de calculs que celui des CNN, puisqu'au lieu de combiner M (nombre de channels d'entrée) et N (nombre de channels de sortie), il additionne M et N (ils ne sont plus multipliés en termes de complexité).

Voici les formules, à titre purement académique [70] :

2.1.2.1 Complexité $* D_F * D_F$

CNN : $D_k * D_k * M * N$

2.1.2.2 Complexité $* D_F + M * N * D_F * D_F$

MB : $D_k * D_k * M * D_F$

D_F : est la largeur de la fenêtre de filtre

Ce qui représente un rapport de :

$$\frac{MB}{CNN} = \frac{1}{N} + \frac{1}{D_K^2} \quad (3.1)$$

Cet écart n'est pas anodin, puisqu'au final un algorithme MobileNets entraîné ne pèsera pas plus de 1 mégaoctet (contre au moins une vingtaine pour les concurrents orientés « mobiles » et plusieurs centaines pour les autres), et permettra, par exemple, de classifier des images à 130fps sur une GTX970 pour le MobileNets le plus lourd (avec le plus de couches) et à 450fps pour le plus léger. . . (Contre 15fps maximum pour les autres). Cette rapidité incroyable se paye néanmoins par quelques pourcents d'erreur : 95.5% de succès pour reconnaître une route avec un Mobilenet et 95.9% pour son concurrent Inception [71].

2.2 Comprendre SSD MultiBox

Dans cette section on va essayer de constituer une explication intuitive de la technique de détection d'objet SSD Multibox, à travers les principes de cette architecture, qui comprend l'explication de ce que l'algorithme MultiBox fait.

Pour mieux comprendre le SSD, commençons par expliquer pourquoi l'approche des fenêtres coulissantes ne fonctionnerait-elle pas bien ?

Il est naturel de penser à construire un modèle de détection d'objets au-dessus d'un modèle de classification d'images. Une fois que nous avons un bon classificateur d'image, un moyen simple de détecter des objets consiste à faire glisser une "fenêtre" sur l'image et à classer si l'image dans cette fenêtre (région rognée de l'image) est du type souhaité. Cela semble simple ! Eh bien, il y a au moins deux problèmes :

- ❖ Comment connaître la taille de la fenêtre pour qu'elle contienne toujours l'objet ? Différents types d'objets, même les objets de même type peuvent également être de tailles différentes.
- ❖ Format d'image (rapport hauteur/largeur d'un cadre de délimitation). De nombreux objets peuvent être présents sous différentes formes, comme une empreinte de bâtiment aura un rapport d'aspect différent de celui d'un palmier.

Pour résoudre ces problèmes, nous devrions essayer différentes tailles/formes de fenêtre glissante, ce qui est très gourmand en calcul, en particulier avec un réseau neuronal profond.

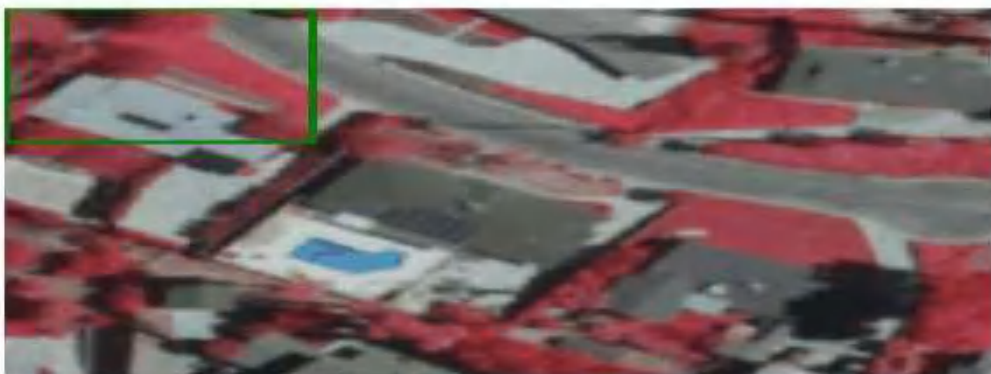


Figure 3.3 : Exemple d'approche de fenêtre coulissante

En pratique, il existe deux types d'algorithmes de détection d'objets courants. Des algorithmes comme R-CNN et Fast(er) R-CNN utilisent une approche en deux étapes - d'abord pour identifier les régions où les objets sont censés être trouvés, puis détecter les objets uniquement dans ces régions à l'aide de convnet. D'autre part, des algorithmes comme YOLO (You Only Look Once) [1] et SSD (Single-Shot Detector) [2] utilisent une approche entièrement convolutive dans laquelle le réseau est capable de trouver tous les objets d'une image en une seule passe (d'où 'single-shot' ou 'look once') à travers le convnet. Les algorithmes de proposition de région ont généralement une précision légèrement meilleure mais plus lents à exécuter, tandis que les algorithmes à un seul coup sont plus efficaces et ont une précision aussi bonne et c'est ce sur quoi nous allons nous concentrer dans cette section.

Ensuite, passons en revue les concepts/paramètres importants du SSD.

Cellule de grille :

Au lieu d'utiliser une fenêtre coulissante, SSD divise l'image à l'aide d'une grille et chaque cellule de la grille est responsable de la détection des objets dans cette région de l'image. La détection d'objets signifie simplement prédire la classe et l'emplacement d'un objet dans cette région. Si aucun objet n'est présent, nous le considérons comme la classe d'arrière-plan et l'emplacement est ignoré. Par exemple, nous pourrions utiliser une grille 4x4 dans l'exemple ci-dessous. Chaque cellule de la grille est capable de sortir la position et la forme de l'objet qu'elle contient.



Figure 3.4 : Exemple d'une grille de 4x4

Maintenant, ce qui se passe s'il y a plusieurs objets dans une cellule de la grille ou si nous devons détecter plusieurs objets de formes différentes ?

C'est là que la boîte d'ancrage (Anchor box) et le champ réceptif entrent en jeu.

Boîte d'ancrage (Anchor Box):

Chaque cellule de grille dans SSD peut être affectée à plusieurs cases d'ancrage/prior. Ces boîtes d'ancrage sont prédéfinies et chacune est responsable d'une taille et d'une forme dans une cellule de la grille. Par exemple, la piscine dans l'image ci-dessous correspond à la boîte d'ancrage la plus haute tandis que le bâtiment correspond à la boîte la plus large.

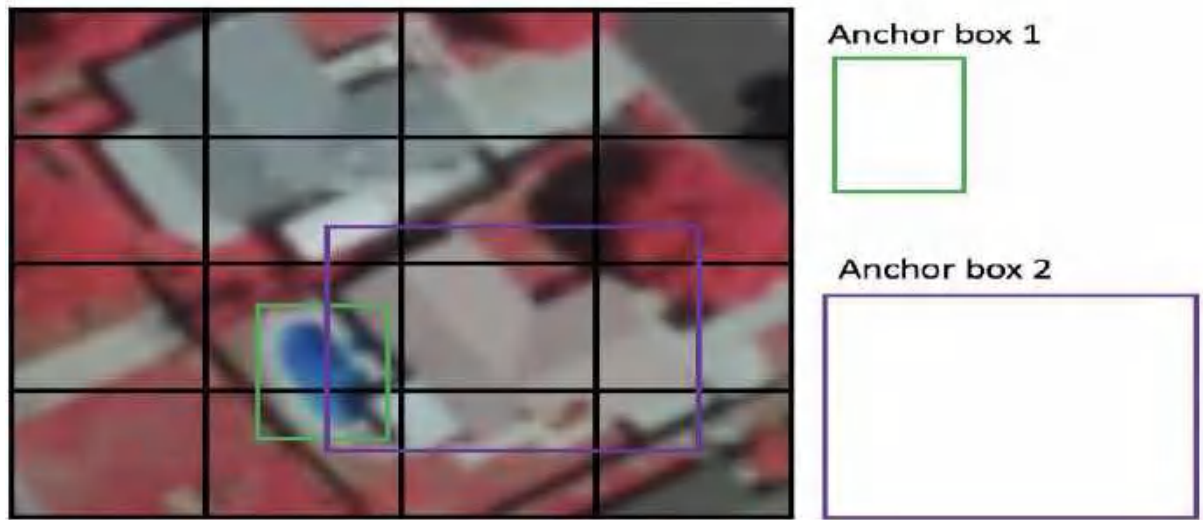


Figure 3.5 : Exemple de deux boîtes d'ancrage

Ratio d'aspect :

En réalité tous les objets ne sont pas de forme carrée. Certains sont plus longs et certains sont plus larges, à des degrés divers. L'architecture SSD permet des rapports d'aspect prédéfinis des boîtes d'ancrage pour en tenir compte. Le paramètre ratios peut être utilisé pour spécifier les différents rapports d'aspect des boîtes d'ancrage associées à chaque cellule de la grille à chaque niveau de zoom/échelle.



Figure 3.6: La boîte englobante du bâtiment 1 est plus haute tandis que la 2-eme est plus large

Le niveau de zoom :

Il n'est pas nécessaire que les boîtes d'ancrage aient la même taille que la cellule de la grille. Nous pourrions être intéressés par la recherche d'objets plus petits ou plus grands dans une cellule de la grille. Le paramètre zooms est utilisé pour spécifier de combien les boîtes d'ancrage doivent être agrandies ou réduites par rapport à chaque cellule de la grille. Tout comme ce que nous avons vu dans l'exemple de la boîte d'ancrage, la taille du bâtiment est généralement plus grande que celle de la piscine.

2.2.1 Architecture

Comme vous pouvez le voir sur la figure 3.7, l'architecture SSD a deux composants : un modèle de dorsale et une entête SSD. Le modèle dorsal est généralement un réseau de classification d'images préformé en tant qu'extracteur de caractéristiques. Il s'agit généralement d'un réseau comme MobileNet, ResNet formé sur ImageNet à partir duquel la couche de classification finale entièrement connectée a été écrasé. Nous nous retrouvons donc avec un réseau de neurones profonds capable d'extraire le sens sémantique de l'image d'entrée tout en préservant la structure spatiale de l'image mais à une résolution inférieure. Pour ResNet34 par exemple, le backbone donne 256 cartes de caractéristiques 7x7 pour une image d'entrée. Nous expliquerons plus tard quelles sont les fonctionnalités et la carte des fonctionnalités ou des caractéristiques (features map). L'entête de SSD est juste une ou plusieurs couches convolutives ajoutées à ce modèle dorsale (MobileNet par exemple) et les sorties sont interprétées comme les boîtes englobantes (detector box) et les classes d'objets dans l'emplacement spatial des activations des couches finales.

Dans la figure ci-dessous, les premières couches (boîtes blanches) sont la colonne vertébrale, les dernières couches (boîtes bleues) représentent la tête du SSD.

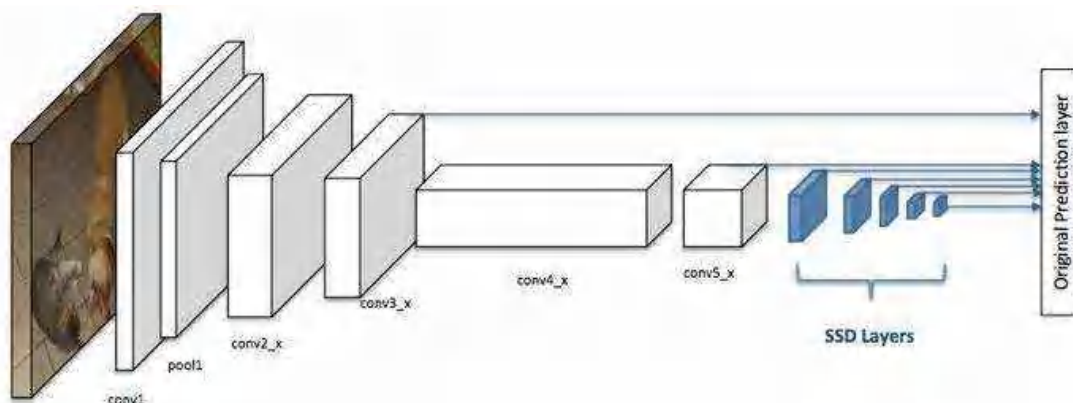


Figure 3.7– Architecture d'un réseau de neurones convolutifs avec un détecteur SSD

2.2.2 Pertes Multibox d'architecture SSD

La fonction de perte de MultiBox combine également deux composants critiques qui sont :

- **Perte de confiance** : cela mesure le degré de confiance du réseau vis-à-vis de l'objectivité du cadre de sélection (bounding box) calculé. La fonction d'entropie croisée catégorique est utilisée pour calculer cette perte.
- **Perte de localisation** : cette option mesure la distance entre les boîtes de détection prédites par le réseau et les boîtes de vérités par rapport au jeu d'entraînement.

L'expression de la perte Multibox total est mesuré par la relation suivante :

$$\text{Perte Multibox} = \text{Perte_Confiance} + \alpha * \text{Perte_Localisation} \quad (3.2)$$

Le terme *alpha* nous aide à équilibrer la contribution de la perte de localisation. Comme d'habitude dans l'apprentissage en profondeur, l'objectif est de trouver les valeurs des paramètres ou des poids qui réduisent au mieux la fonction de perte, en se rapprochant ainsi à un model plus optimale.

2.2.3 Priors MultiBox et IoU

La logique qui entoure la génération du cadre de sélection (Bounding Box) est en réalité plus complexe que ce qu'on a déclaré précédemment.

Dans MultiBox, les chercheurs ont créé ce que nous appelons des priors, qui sont des Bounding Boxes précalculés, de taille fixe qui correspondent étroitement à la distribution des boîtiers de vérité originale. En fait, ces priors sont sélectionnés de telle sorte que leur rapport Intersection sur Union (alias IoU, et parfois appelé indice de Jaccard) soit supérieur à 0,5. Comme vous pouvez le déduire de la figure 3.8, une IoU de 0,5 n'est toujours pas suffisante, mais elle fournit néanmoins un bon point de départ pour l'algorithme de régression du cadre de sélection - il s'agit d'une stratégie bien meilleure que celle de commencer les prédictions avec des coordonnées aléatoires ! Par conséquent, MultiBox commence avec les prédécesseurs sous forme de prédictions et tente de régresser plus près des boîtiers de vérité.

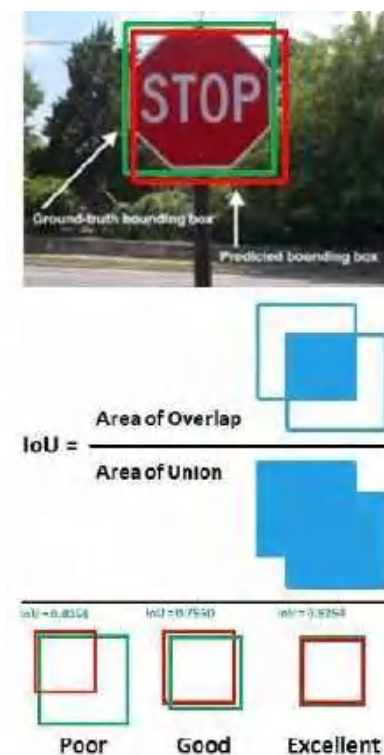


Figure 3.8 – Diagramme expliquant IoU (de Wikipedia)

A la fin, MultiBox conserve uniquement les K principales prévisions ayant minimisé les pertes de localisation (LOC) et de confiance (CONF).

2.3 Le modèle MobileNet-SSD

MobileNet-SSD c'est la combinaison entre les deux algorithmes SSD et Mobilenet, le premier permet de placer des boîtes sur toute l'image puis les classer par le deuxième, ce qui rend l'algorithme très performant et utile.

Le modèle MobileNet-SSD peut considérablement réduire le nombre de paramètres et atteindre une précision supérieure dans des conditions matérielles limitées comme pour le cas de l'implémentation sur une Raspberry Pi. Le modèle complet comprend quatre parties : la couche d'entrée pour importer l'image cible, le réseau de base MobileNet pour extraire les caractéristiques de l'image, le SSD pour la régression de classification et la régression de la boîte bornée (Detector box) et la couche de sortie pour l'exportation des résultats de la détection [73].

Mobilenet-SSD était l'algorithme de choix pour ce projet. C'est parce que le RPi a un processeur faible, nous devons donc utiliser un modèle qui utilise moins de puissance de traitement. Pour ce projet, nous avons adopté la version 2 allégée du modèle qui s'appelle SSDLite-Mobilenet-V2 car c'est le modèle le plus rapide disponible.

Le tableau 3.1. Résume les résultats obtenus que nous avons tiré de l'article [76] pour les différentes versions du modèle `ssd_mobilenet` entraînés par la base de données COCO.

Table 3.1- Modèles entraînés par la base de données COCO

Nom du modèle	Vitesse (ms)	COCO (Précision)	Sorties
<code>ssd_mobilenet_v1_coco</code>	30	80	Cases
<code>ssd_mobilenet_v1_0.75_depth_coco</code>	26	75	Cases
<code>ssd_mobilenet_v1_quantized_coco</code>	29	75	Cases
<code>ssd_mobilenet_v1_0.75_depth_quantized_coco</code>	29	73	Cases
<code>ssd_mobilenet_v1_ppn_coco</code>	26	80	Cases
<code>ssd_mobilenet_v1_fpn_coco</code>	56	88	Cases
<code>ssd_resnet_50_fpn_coco</code>	76	94	Cases
<code>ssd_mobilenet_v2_coco</code>	31	86	Cases
<code>ssd_mobilenet_v2_quantized_coco</code>	29	86	Cases
<code>ssdlite_mobilenet_v2_coco</code>	27	86	Cases
<code>ssd_inception_v2_coco</code>	42	90	Cases

2.4 Importance des jeux de données

Les jeux de données jouent un rôle très important dans la recherche. Chaque fois qu'un nouvel ensemble de données est publié, des articles sont publiés et de nouveaux modèles sont comparés et souvent améliorés, repoussant les limites du possible.

Malheureusement, il n'y a pas assez de jeux de données pour la détection d'objets. Les données sont plus difficiles (et plus coûteuses) à générer, les entreprises n'ont probablement pas l'impression de donner librement leurs investissements et les universités n'ont pas beaucoup de ressources

Vous trouverez dans le tableau 3.2 une liste des principaux jeux de données disponibles.

Table 3.2 – Les principaux jeux de données disponibles

Name	Images (trainval)	Classes
ImageNet	450k	200
COCO	120K	90
Pascal VOC	12k	20
Oxford-IIIT Pet	7K	37
KITTI Vision	7K	3

On a choisi dans un premier temps de travailler avec le modèle `ssdlite_mobilenet_v2`, qui est pré-entraîné avec la base de données des images étiquetées COCO, contenant 90 classes (pour couvrir la totalité des objets qui bougent), en prenant en considération ses performances illustrées au tableau 3.1 ;

Chapitre 3

Préparation de l'environnement logiciel

3.1 Présentation de l'architecture du système de développement utilisé :



Nous avons d'un cote un PC qui va nous servir dans notre développement et d'une autre cote une carte Raspberry pi dont laquelle on va installer les outils nécessaires pour la commande de notre système.

Outils installés sur la carte Raspberry Pi :

- Raspberry pi OS 32 bits
- Tensorflow
- Numpy
- OpenCV

Logiciels utilise le PC :

- PyCharm
- Isis Proteus
- TinkerCad
- Fritzing
- Google Colab

3.2 Le choix des outils utilisés :

Le choix de l'ensemble de ces outils et APIs n'était pas anodin, mais il vient après une étude benchmarking sur les différents outils et solutions existantes appropriées à notre matérielle :

- ✓ Etude 1 : une étude benchmarking sur les outils et les APIs de développement des applications de Deep Learning

Après cette étude, on a décidé de travailler avec la plateforme **Google Colab** pour plusieurs raisons :

- Il est Open source.
- Programmation en langage Python
- Il Supporte l'exécution du traitement sur GPU et CPU en Cloud.

- ✓ Etude 2 : une étude benchmarking sur les outils et les APIs de traitement des images

On a choisi de travailler avec la bibliothèque graphique libre **OpenCV** pour les raisons suivantes :

- Il Supporte la majorité des formats d'images.
- Il est Open source comme il possède une documentation riche.

- **Raspberry pi OS 32 bits** : Est le système d'exploitation spécialement conçu pour le Raspberry Pi. Sa dernière version, Stretch, apporte de nombreuses modifications pour optimiser les performances, et reste la version proposée par défaut par **Raspberry Pi Imager**, le logiciel officiel d'installation de systèmes d'exploitation pour **Raspberry Pi**.
- **PyCharm** comme IDE de développement en Python
- **TensorFlow** : est le deuxième Framework d'apprentissage automatique créé et utilisé par Google pour concevoir, élaborer et former des modèles d'apprentissage en profondeur.
- **OpenCV** : OpenCV (Open Source Computer Vision Library) est une bibliothèque open source de vision par ordinateur et d'apprentissage automatique. OpenCV a été construit pour fournir une infrastructure commune pour les applications de vision par ordinateur et pour accélérer l'utilisation de la perception machine dans les produits commerciaux.
- **Fritzing** : est un outil open source, visant à développer un logiciel de CAO simple a manipulé pour la conception de matériel électronique, destiné à aider les concepteurs, ainsi que les débutants dans l'électronique, à expérimenter des prototypes par la conception des circuits électroniques de façon entièrement graphique.
- **Numpy** : NumPy est la bibliothèque fondamentale du calcul scientifique avec Python.
- **Isis Proteus** : est un logiciel de réalisation des cartes électroniques qui permet aussi la simulation de montages électroniques.
- **Tinkercad** est un outil de modélisation 3D en ligne gratuit qui s'exécute dans un navigateur Web, connu pour sa simplicité et sa facilité d'utilisation.
- **Google Colab** est un service cloud, offert par Google (gratuit), basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud. Sans donc avoir besoin d'installer quoi que ce soit sur notre ordinateur à l'exception d'un navigateur.

3.3 Mise en œuvre de la connexion entre la carte Raspberry pi et le PC

Pour connecter notre carte Raspberry Pi à un ordinateur portable, nous pouvons simplement utiliser le WiFi. L'interface graphique de bureau du Raspberry Pi (interface utilisateur graphique) peut être visualisée via l'écran de l'ordinateur portable à l'aide d'une connexion sans fil entre les deux. Il existe de nombreux logiciels disponibles qui peuvent établir une connexion entre la carte Raspberry Pi et notre ordinateur portable. Pour nous avons opté d'utiliser le logiciel VNC (Virtual Network Computing).

L'installation de VNC sur notre Pi nous permet de voir l'interface bureautique du Raspberry Pi à distance (après installation de système d'exploitation), en utilisant la souris et le clavier de notre PC portable. Cela signifie également que nous pouvons ce connecté à notre carte Raspberry Pi n'importe où ailleurs, il suffit juste l'alimenter.

3.3.1 Téléchargement et installation du Raspberry Pi OS sur notre carte microSD

Avant de passer à connecter notre Raspberry Pi à notre ordinateur portable, nous avons besoin d'une carte SD contenant un système d'exploitation de Raspberry Pi. De nombreux systèmes d'exploitation sont disponibles pour le Raspberry Pi et la plupart sont centrés sur Linux, mais Raspbian reste la version la plus populaire.

Pour cela on va utiliser un mini logiciel qui s'appelle Raspberry pi Imager dans lequel on sélectionne notre MicroSD qui est de taille de 16GB puis installer le raspberry pi OS 32 bits sur elle.

On clique pour choisir le système d'exploitation



Puis on sélectionne le raspberry pi OS 32 bits dans le menu OS



Puis on choisit notre MicroSD card



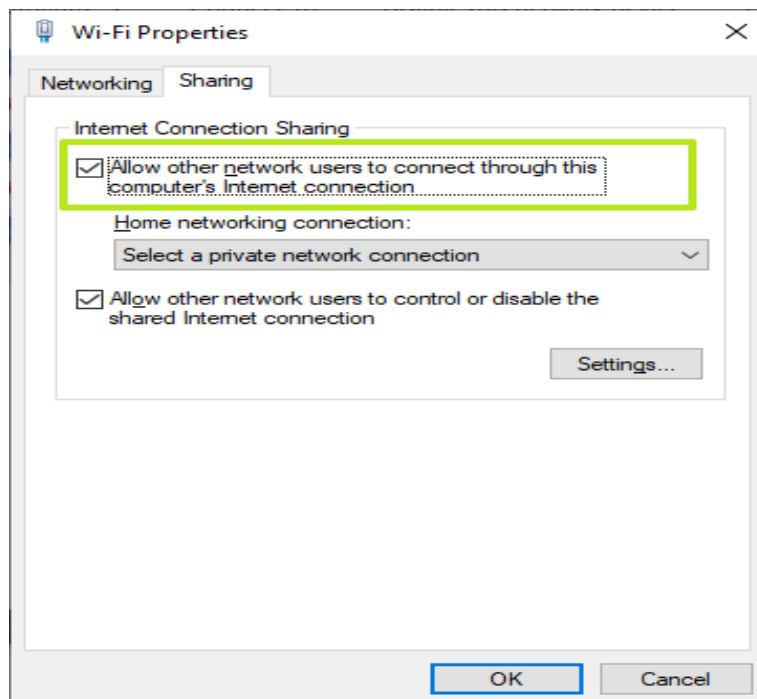
En fin en clique sur Write pour écrire le Raspberry pi OS 32 bits sur notre MicroSD card



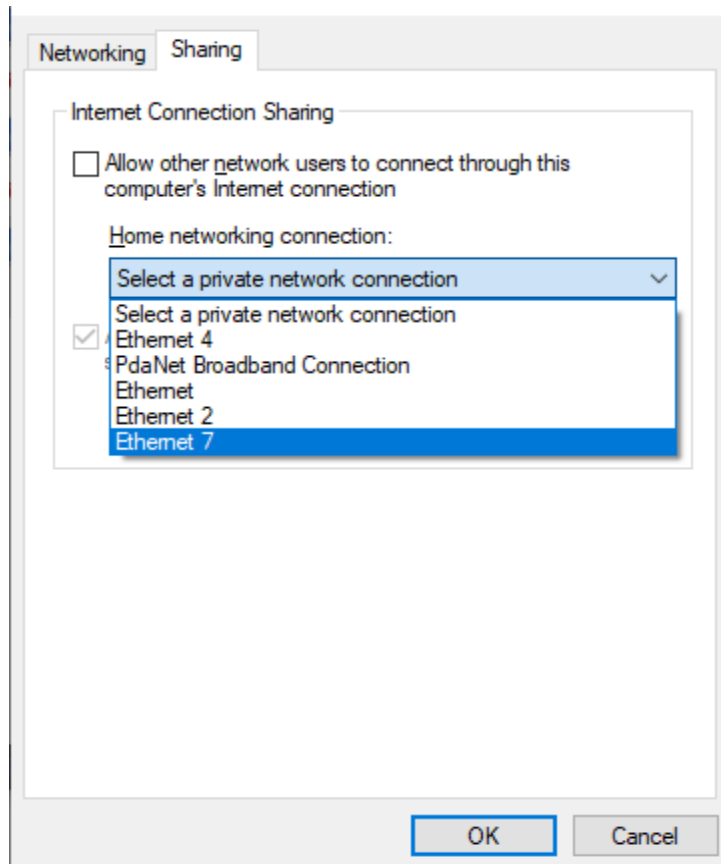
3.2.2 Se connecter à internet via un câble Ethernet branché entre la carte Raspberry pi et le PC

On a accédé au menu Connexions réseau, qui fait partie du Panneau de configuration en allant dans Paramètres->Réseau et Internet->Wi-Fi, puis en cliquant sur "Modifier les paramètres de l'adaptateur" sur le côté droit de l'écran. Cela fonctionne que vous partagiez une connexion Internet qui arrive sur votre PC à partir du Wi-Fi ou d'Ethernet.

- Activons " Allow other network users to connect " dans l'onglet "Sharing".



- On Sélectionne le port Ethernet connecté au Raspberry Pi dans le menu "Home networking connexion" puis on clique Ok

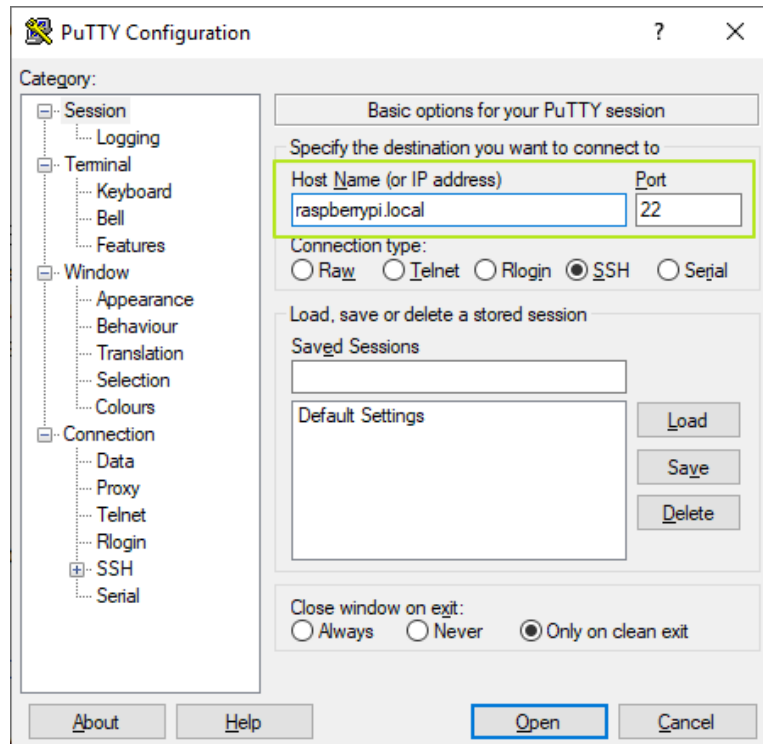


Après avoir connecté le Pi directement au partage WIFI de notre PC, passant à la connexion via SSH

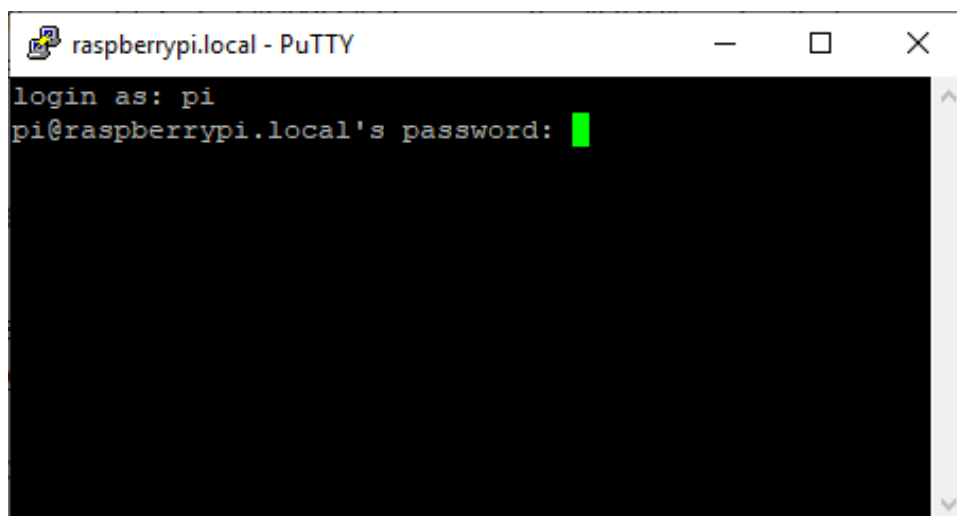
3.2.3 Connexion via SSH

Maintenant pour se connecter au Raspberry pi seulement via le mode terminal de commande on a adopté la connexion avec SSH.

1. Pour cela on a téléchargé et installé le logiciel Putty qui est le principal client SSH client pour Windows
2. On entre raspberrypi.local comme adresse à laquelle on souhaite se connecter dans Putty, puis cliquant sur Ouvrir. On doit généralement ajouter le local puisque la carte Raspberry Pi est directement connectée à notre PC via le câble Ethernet.



Après on clique sur Ok, et une fenêtre de commande va apparaître demandant d'entrer le mot de passe pour accéder au carte Raspberry pi.

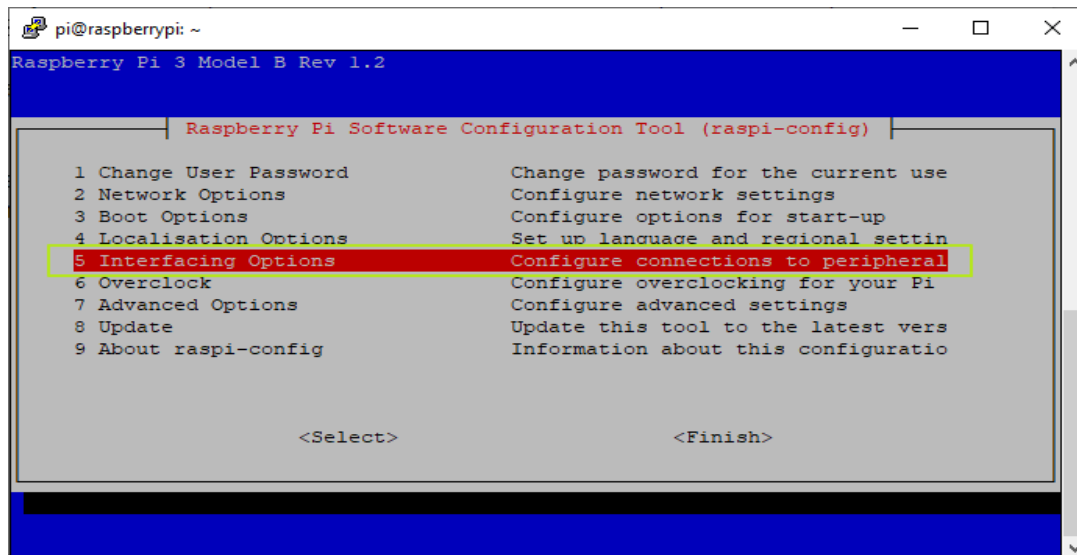


Maintenant on est connecté seulement via l'invite de commande.

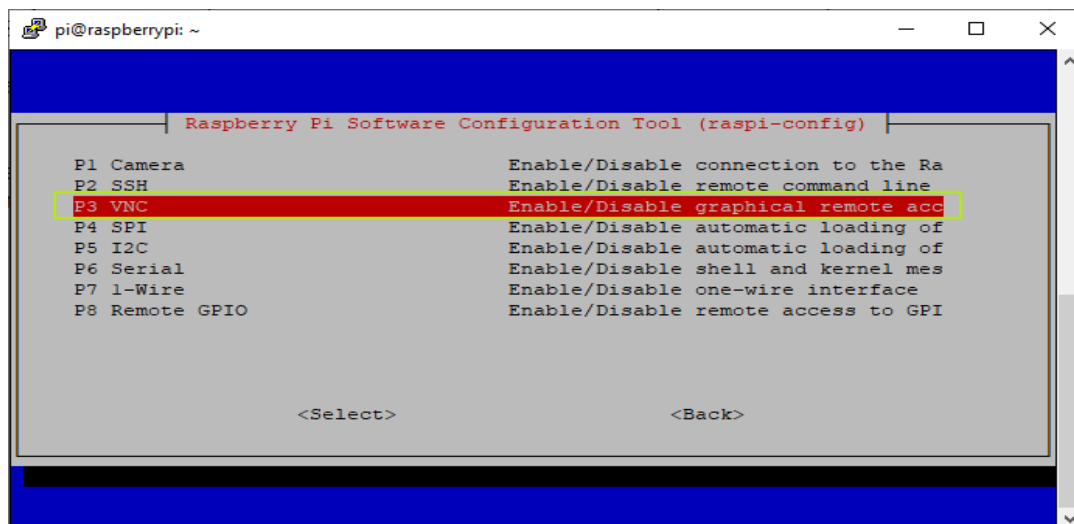
Pour accéder à l'interface graphique, avec un bureau et des fenêtres flottantes, on a utilisé le logiciel de visualisation et de contrôle à distance VNC.

Pour cela on a exécuté la commande `sudo raspi-config`

Puis après la fenêtre suivante va apparaître, et on clique sur "Interfacing Options".



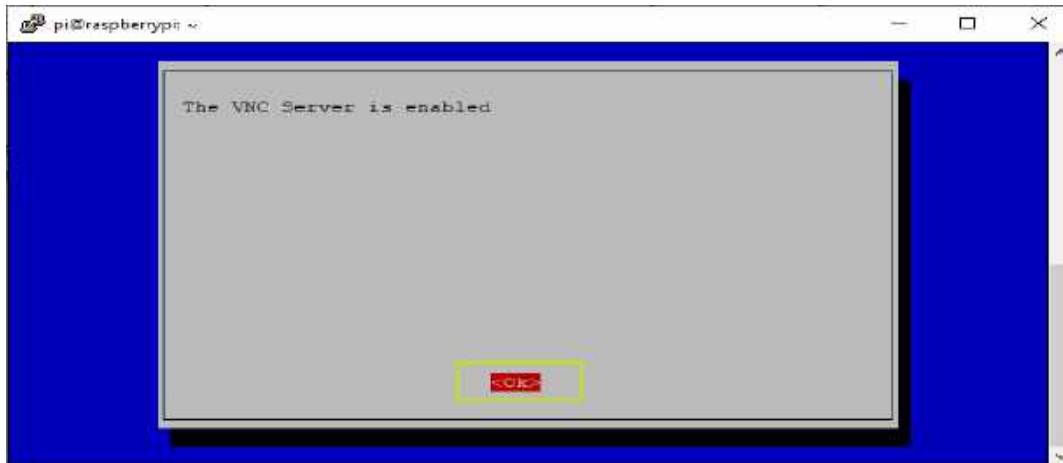
Après, on sélectionne l'outil VNC pour l'activer



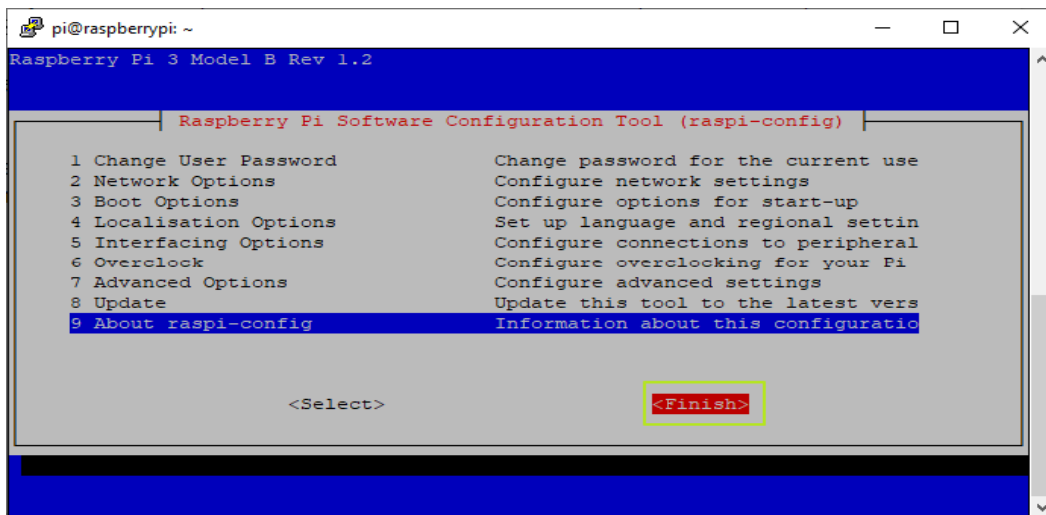
Puis on clique sur YES



On appui sur 'Ok' pour confirmer que le serveur VNC est activé.

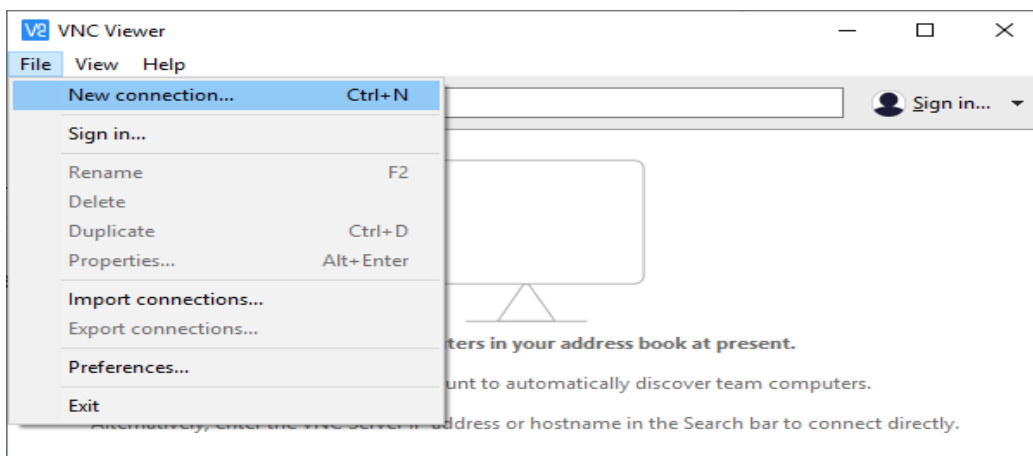


Finalemnt on appui sur 'Finish'

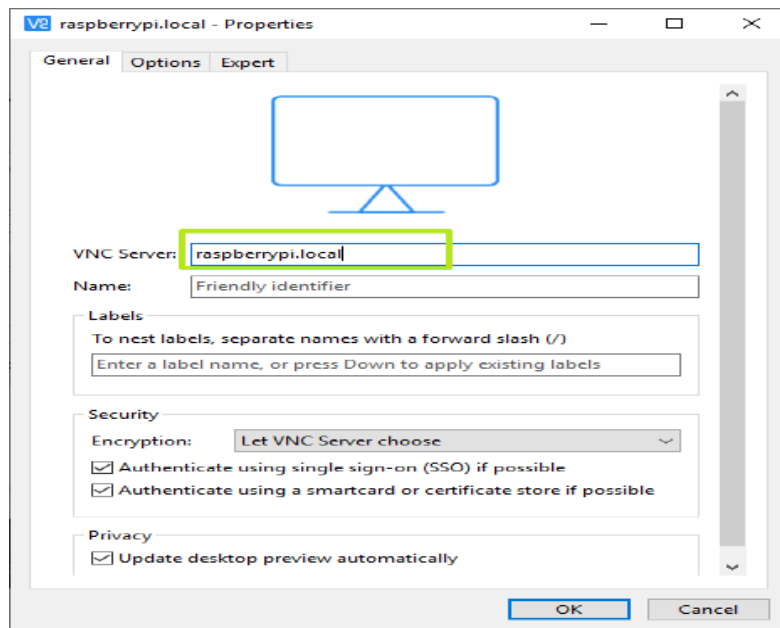


Maintenant, on doit télécharger et installer VNC viewer sur notre PC afin de pouvoir visualiser et contrôler la carte Raspberry Pi depuis notre PC.

Après on lance le programme et on ouvre une nouvelle session de connexion.



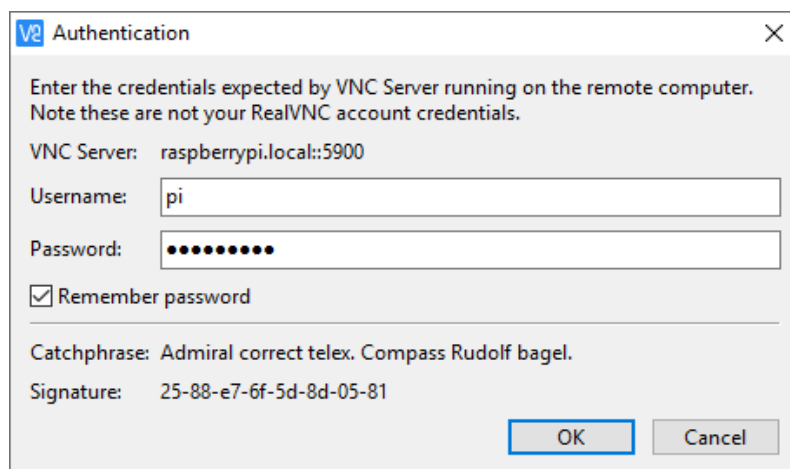
Puis, on entre 'raspberrypi.local' dans le champ "Serveur VNC" et on clique sur 'OK'.



Puis, la fenêtre suivante apparaitre, On clique sur elle



Il est demande de nous après de saisir le nom d'utilisateur et le mot de passe.



On clique sur 'OK' et enfin, le bureau de Raspberry Pi va apparaître comme une fenêtre VNC.

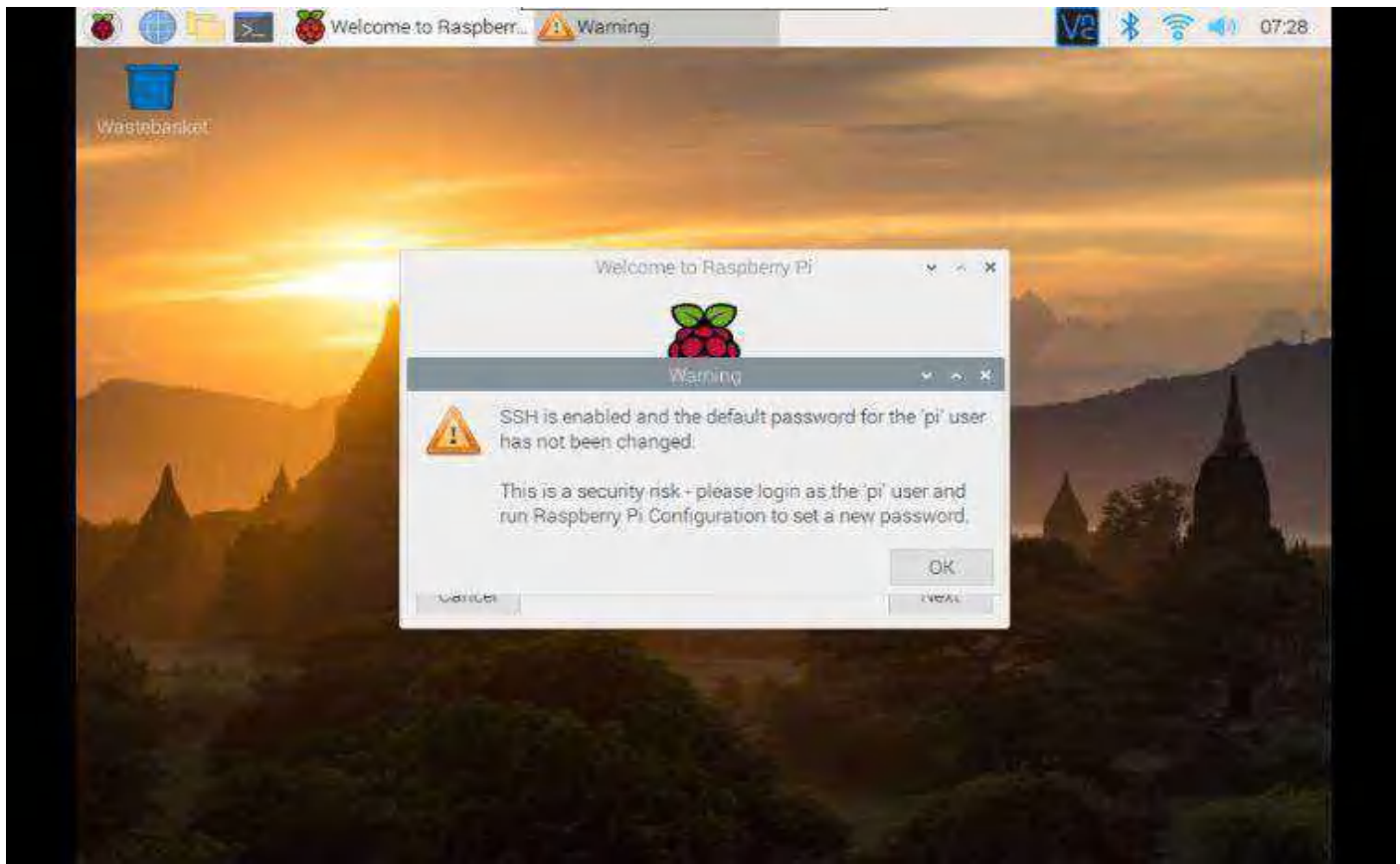


Figure 4.1 – L'interface graphique de système d'exploitation de Raspberry Pi

Et ainsi nous pourrons accéder à l'interface graphique et faire usage des périphériques de notre PC (Clavier, souris et écran)

3.4 Déploiement de L'algorithme SSDLite-Mobilenet sur le RPi

Cette section fournit des instructions détaillées sur la configuration des APIs nécessaires pour la détection d'objets dans des flux vidéo en temps réel à savoir TensorFlow, OpenCv, Protobuf, etc...

3.4.1 La mise en œuvre de la carte Raspberry Pi

Tout d'abord, la Raspberry Pi doit être entièrement actualisé. Pour cela en fait usage a l'invite de Commande et lance les commandes suivantes :

```
sudo apt-get update
```

```
sudo apt-get dist-upgrade
```

Selon le temps écoulé depuis la mise à jour de votre Pi, la mise à jour peut prendre entre une minute et une heure.

3.4.2 Installation de la librairie TensorFlow

Ensuite, nous allons installer TensorFlow. Dans le répertoire */home/pi*, créez un dossier nommé "tf", qui sera utilisé pour contenir tous les fichiers d'installation de TensorFlow et Protobuf :

```
mkdir tf  
cd tf
```

Un fichier préconstruit, Raspberry Pi compatible, pour l'installation de la librairie TensorFlow est disponible dans le référentiel "TensorFlow for ARM" dans Github avec des packages d'installation précompilés.

```
wget https://github.com/lhelontra/tensorflow-on-arm/releases/download/v1.8.0/  
tensorflow-1.8.0-cp35-none-linux_armv7l.whl
```

Maintenant que nous avons le fichier, on installe TensorFlow en lançant la commande:

```
sudo pip3 install /home/pi/tf/tensorflow-1.8.0-cp35-none-linux_armv7l.whl
```

TensorFlow a également besoin du paquet LibAtlas. Installez-le en lançant (si cette commande ne fonctionne pas, lancez "*sudo apt-get update*" puis réessayez) :

```
sudo apt-get install libatlas-base-dev
```

Tant que cela est fait, installons d'autres dépendances qui seront utilisées par l'API de détection d'objets TensorFlow. Celles-ci sont répertoriées dans les instructions d'installation du référentiel GitHub de TensorFlow's Object Detection.

```
sudo pip3 install pillow lxml jupyter matplotlib  
cythonsudo apt-get install python-tk
```

L'installation du tensorflow est faite avec succès, passant maintenant au téléchargement et installation de la bibliothèque OpenCV.

3.4.3 Installation de la bibliothèque OpenCV

Les exemples de détection d'objets de TensorFlow utilisent généralement matplotlib pour afficher des images, mais on préfère d'utiliser OpenCV car il est plus facile à manipuler.

Pour qu'OpenCV fonctionne sur Raspberry Pi, il existe de nombreuses dépendances qui doivent être installées via la commande *apt-get*.

```
sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev libpng12-dev
```

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
```

```
sudo apt-get install libxvidcore-dev libx264-dev
```

```
sudo apt-get install qt4-dev-tools
```

Si l'une des commandes suivantes ne fonctionne pas, lancez "*sudo apt-get update*" puis réessayez.

Maintenant que tous ces dépendances sont téléchargées, nous pouvons lancer le processus d'installation de la bibliothèque OpenCV en utilisant la commande suivante :

```
pip3 install opencv-python
```

3.4.4 Compilation et installation de Protobuf

L'API de détection d'objets TensorFlow utilise Protobuf (Protocole Buffers), un package qui implémente le format de données du tampon de protocole de Google. Malheureusement, il n'existe actuellement aucun moyen facile d'installer Protobuf sur la carte Raspberry Pi. Nous devons le compiler nous-mêmes à partir des sources, puis l'installer. Heureusement, un guide [74] a déjà été écrit sur la compilation et l'installation de Protobuf sur la carte Raspberry pi. Merci à *OSDevLab* d'avoir écrit le guide.

Tout d'abord, il faut obtenir les paquets nécessaires à la compilation de Protobuf à partir des sources.

```
sudo apt-get install autoconf automake libtool curl
```

Ensuite, téléchargez la version de protobuf depuis un dépôt dans GitHub en lançant la commande :

```
wget https://github.com/google/protobuf/releases/download/v3.5.1/protobuf-  
all-3.5.1.tar.gz
```

Si une version plus récente de protobuf est disponible, téléchargez-le à la place.

Décompressez le fichier et le placer dans le dossier :


```
tar -zxvf protobuf-all-3.5.1.tar.gz
cd protobuf-3.5.1
```

Configurez la construction en lançant la commande suivante :

```
./configure
```

Construisez le paquet en lançant :

```
make
```

Le processus de construction a pris 1h approximativement sur notre Raspberry Pi. Quand ce sera fini, lancez :

```
make check
```

Ce processus prend encore plus de temps, avec un pointage à 107 minutes sur notre Pi. Selon d'autres guides que j'ai vus, cette commande peut sortir avec des erreurs, mais Protobuf fonctionnera toujours. Si vous voyez des erreurs, vous pouvez les ignorer pour le moment. Maintenant qu'il est construit, installez-le en lançant :

```
sudo make install
```

Déplacez-vous ensuite dans le répertoire python et exportez le chemin de la bibliothèque :

```
cd python
export LD_LIBRARY_PATH=./src/.libs
```

Ensuite,

```
python3 setup.py build --cpp_implementation
python3 setup.py test --cpp_implementation
sudo python3 setup.py install --cpp_implementation
```

Ensuite, lancez les commandes de chemin d'accès suivantes :

```
export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=cpp
export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION_VERSION=3
```

Enfin,

```
sudo ldconfig
```

C'est tout ! Maintenant, Protobuf est installé sur le RPi. Vérifiez qu'il est installé correctement en lançant la commande ci-dessous et en veillant à afficher le texte d'aide par défaut.

```
protoc
```

Enfin, la carte Raspberry Pi doit être redémarré après ce processus, sinon TensorFlow ne fonctionnera pas. en lançant la commande:

```
sudo reboot now
```

3.4.5 Configuration de la structure de répertoire TensorFlow et de la variable PYTHONPATH

Maintenant que tous les paquets sont installés, nous devons configurer le répertoire TensorFlow. Revenez dans votre répertoire personnel, puis créez un répertoire appelé «tensorflow1».

```
mkdir tensorflow1  
cd tensorflow1
```

Téléchargez le référentiel tensorflow depuis GitHub en lançant :

```
git clone --recurse-submodules https://github.com/tensorflow/models.git
```

Ensuite, nous devons modifier la variable d'environnement PYTHONPATH pour pointer vers certains répertoires du dépôt TensorFlow que nous venons de télécharger. Nous voulons que PYTHONPATH soit défini à chaque fois que nous ouvrons un terminal, nous devons donc modifier le fichier *.bashrc* .

```
sudo nano ~/.bashrc
```

Déplacez-vous à la fin du fichier, et sur la dernière ligne, ajoutez :

```
export PYTHONPATH=$PYTHONPATH:/home/pi/tensorflow1/models/research:/home/  
pi/tensorflow1/models/research/slim
```

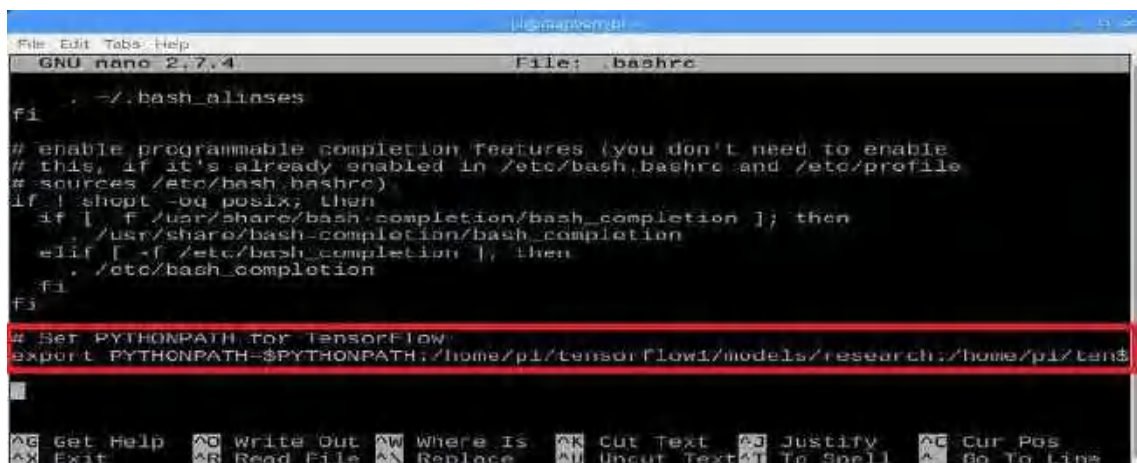


Figure 4.2 – ajoutez PYTHONPATH pour TensorFlow

Ensuite, enregistrez et quittez le fichier. Ainsi, la commande « export PYTHONPATH » est appelée à chaque fois que vous ouvrez un nouveau terminal. La variable PYTHONPATH sera donc toujours définie de manière appropriée. Fermez puis rouvrez le terminal.

Maintenant, nous devons utiliser *Protoc* pour compiler les fichiers de tampon de protocole (.proto) utilisés par l'API de détection d'objets. Les fichiers (.proto) se trouvent dans `/research/object_detection/protos`, mais nous devons exécuter la commande à partir du répertoire `/research`.

```
cd /home/pi/tensorflow1/models/research  
protoc object_detection/protos/*.proto --python_out=.
```

Cette commande convertit tous les fichiers "*name*".proto en fichiers "*name_pb2*".py. Ensuite, allez dans le répertoire `object_detection` :

```
cd /home/pi/tensorflow1/models/research/object_detection
```

Maintenant, nous allons télécharger le modèle SSDLite-MobileNet depuis le **zoo** des modèles de détection d'objets dans le site officiel de TensorFlow. C'est une collection de modèles de détection d'objets pré-entraînés de Google offrant différents niveaux de vitesse et de précision. Bien que le modèle offre une précision moyenne de 80% avec vitesse de détection élevée.

Google publie continuellement des modèles avec des performances améliorées par les experts de temps en temps.

Téléchargez le modèle SSDLite-MobileNet et décompressez-le en lançant la commande :

```
wget http://download.tensorflow.org/models/object_detection/ssdlite_mobilenet_v2_coco_2018_05_09.tar.gz  
tar -xvzf ssdlite_mobilenet_v2_coco_2018_05_09.tar.gz
```

Le modèle est maintenant dans le répertoire **object_detection** et prêt à être utilisé.

3.5 Test du modèle

Maintenant, tout est configuré pour effectuer la détection d'objet sur le RPi. Vous trouverez le code Python de ce référentiel dans l'annexe 2 pour détecter les objets contenus dans les flux du vidéo en direct à partir d'une Picamera. Fondamentalement, le script définit les chemins d'accès au modèle et à la mappe d'étiquettes, chargement du modèle dans la mémoire, initialisation de la Picamera, puis commence à effectuer la détection d'objet sur chaque image/vidéo capturé par Picamera.

Tout d'abord, on doit assurer que le module camera de la carte Raspberry pi est activée dans le menu de configuration de Raspberry Pi, voire la figure 4.3.

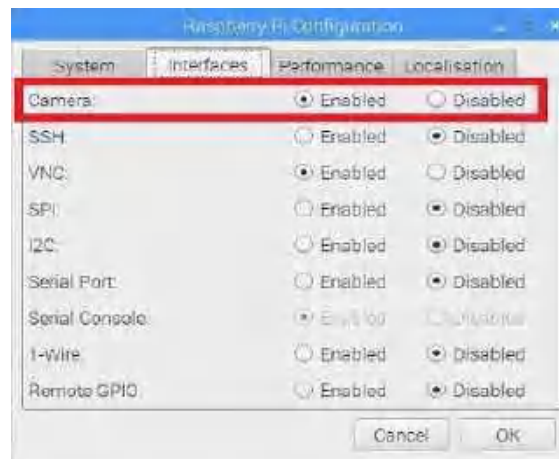


Figure 4.3 – Menu de configuration de Raspberry Pi

Une fois le script initialisé (ce qui peut prendre jusqu'à 30 secondes), une fenêtre affiche une vue en direct de la caméra, et les objets communs à l'intérieur de la vue seront identifiés et entourés avec un bounding box. La figure 4.4 ci-dessous représente les résultats obtenus après la détection des objets existant à l'intérieur de l'image.

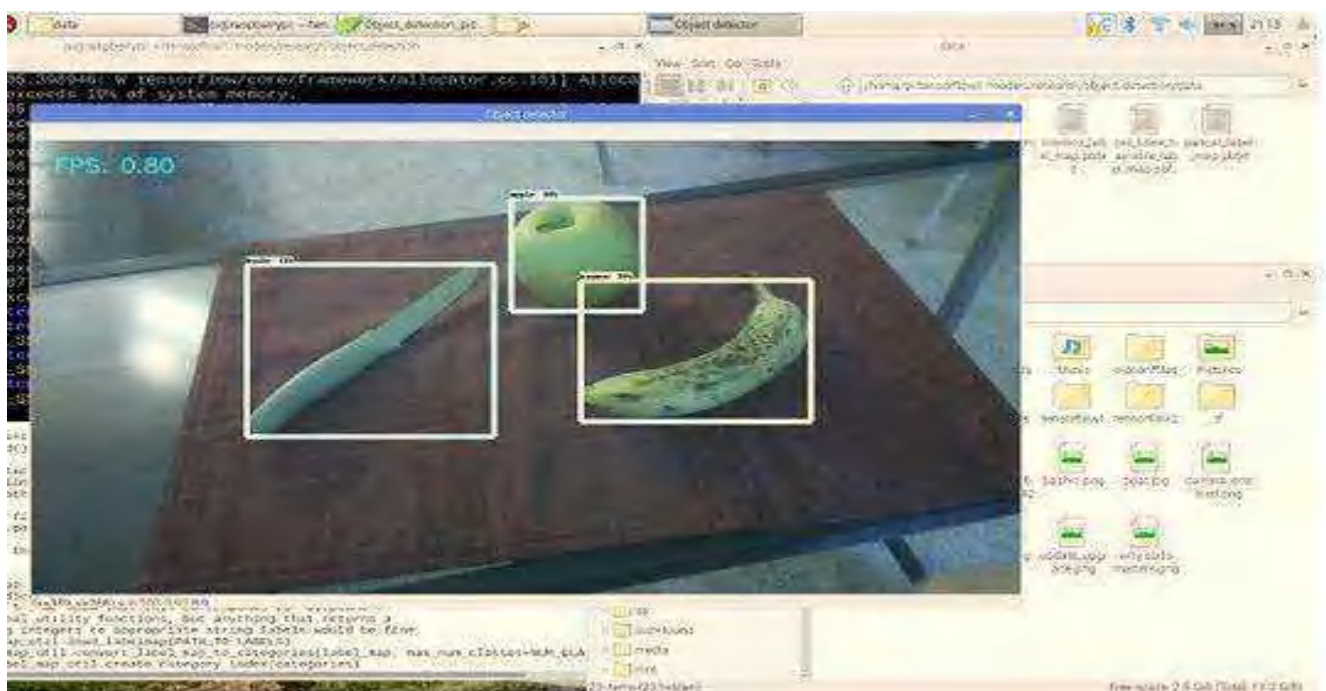


Figure 4.4 – Exemple des objets détectés

Avec le modèle SSDLite-MobileNet, le Raspberry Pi 3 fonctionne assez bien, atteignant un taux d'images par seconde approximativement de 1FPS. C'est assez rapide par rapport à la plupart des autres modèles/architectures de détection d'objets en temps réel.

Chapitre 4

1) Conception matériel du prototype

Le projet repose sur la conception et la réalisation d'un prototype portant sur la gestion intelligente de l'éclairage public, par utilisation de l'intelligence artificielle basé sur application des algorithmes de l'apprentissage en profondeur pour identifier les objets mouvants sur la route, et aussi à l'aide des capteurs de mouvement et de lumière.

Dans ce chapitre, on va présenter les besoins fonctionnels et non-fonctionnels qu'on doit respecter pour la réalisation du projet.

Cette réalisation impose l'utilisation de plusieurs composants bien définis. Le choix de ces derniers doit être soigneusement précis pour que le système fonctionne convenablement.

1.1 L'environnement matériels

L'environnement matériel de notre prototype est composé des éléments suivants :

- Raspberry Pi 3 Model B ;
- Pi Camera Night Vision ;
- Capteur de Mouvement Infrarouge PIR ;
- LDR Photorésistance ;
- Les lampadaires (LEDs..).

1.2 Conception de la maquette

La réalisation de la maquette impose l'utilisation de plusieurs composants bien déterminés. Le choix de ces derniers doit être soigneusement précis pour que le projet fonctionne convenablement.

La figure suivante représente un schéma synoptique de la maquette avec les différents composants et les liaisons entre eux.

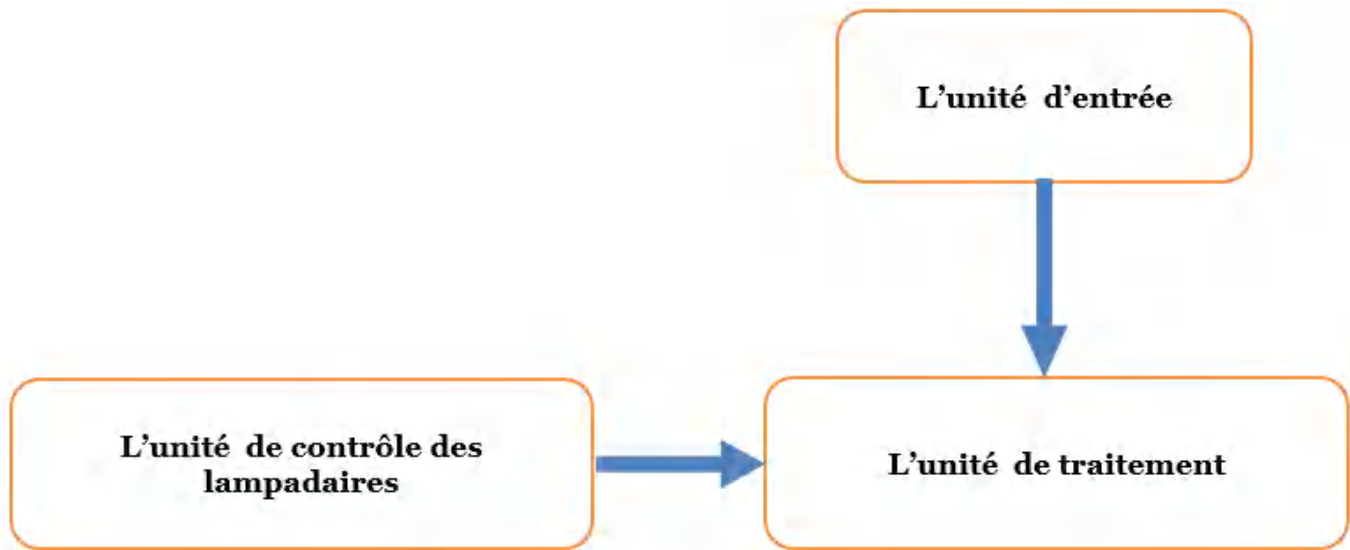


Figure 5.1 – Schéma synoptique de la maquette

Le système comprend trois sous-systèmes : une unité d'entrée (Capteur de lumière LDR, Picamera), une unité de traitement (Carte Raspberry Pi) et l'unité de contrôle des LEDs (les capteurs de détection de mouvement PIR).

1.2.1 L'unité d'entrée

Une carte Raspberry Pi, reliée à un capteur de lumière LDR avec le GPIO 26 et une Picamera nocturnal branchée sur le connecteur CSI (camera serial interface), sont utilisées pour collecter les données d'entrée. Le capteur LDR est utilisé pour différencier entre le jour et la nuit, et la caméra pour enregistrer une vidéo en direct avec une résolution de 1080x720 Pixels.

1.2.2 L'unité de traitement

L'unité de traitement (Raspberry Pi elle-même) effectue de multiples tâches : réception des données par les GPIO d'entrées du RPi et la Picamera, formation des prédictions par un modèle de Deep Learning (pilotage), détection et classification d'objets (piétons, voiture, moto, animaux...), détection des mouvements lorsque des objets spécifiques passent devant les capteurs de mouvement, et l'envoi des commandes aux GPIO de sortie dont lesquelles sont branchées les LEDs. Pour commander l'éclairage de ces derniers.

1.2.3 L'unité de contrôle des lampadaires

Les lampadaires fonctionnent automatiquement dans la nuit avec une intensité de 15%. Lorsque le système détecte l'un des objets qu'on a précisé, la carte Raspberry Pi envoie les commandes aux capteurs PIR à l'aide des GPIOs. Ensuite, si l'objet passe devant les capteurs de mouvement, ceux-ci simulant des actions via des GPIOs qui sont liées aux LEDs, pour fonctionner avec leur intensité maximale.

1.3 Mise en œuvre du prototype

Pour cette tâche, on a choisi le logiciel **Tinkercad** pour faire une conception 3D détaillée du plan de la maquette représenté dans la figure 5.2, avec l'emplacement de chaque composant électronique.

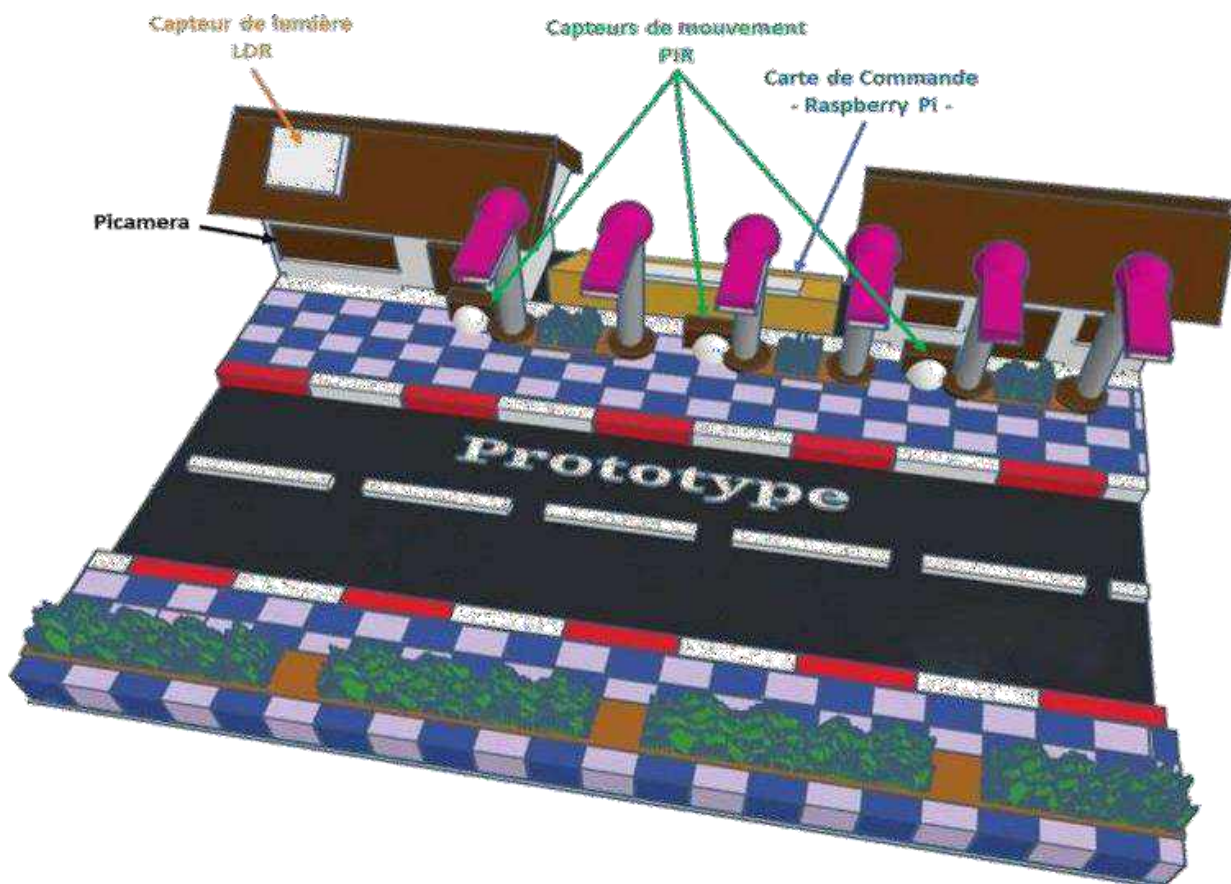


Figure 5.2 – Plan 3D de la maquette

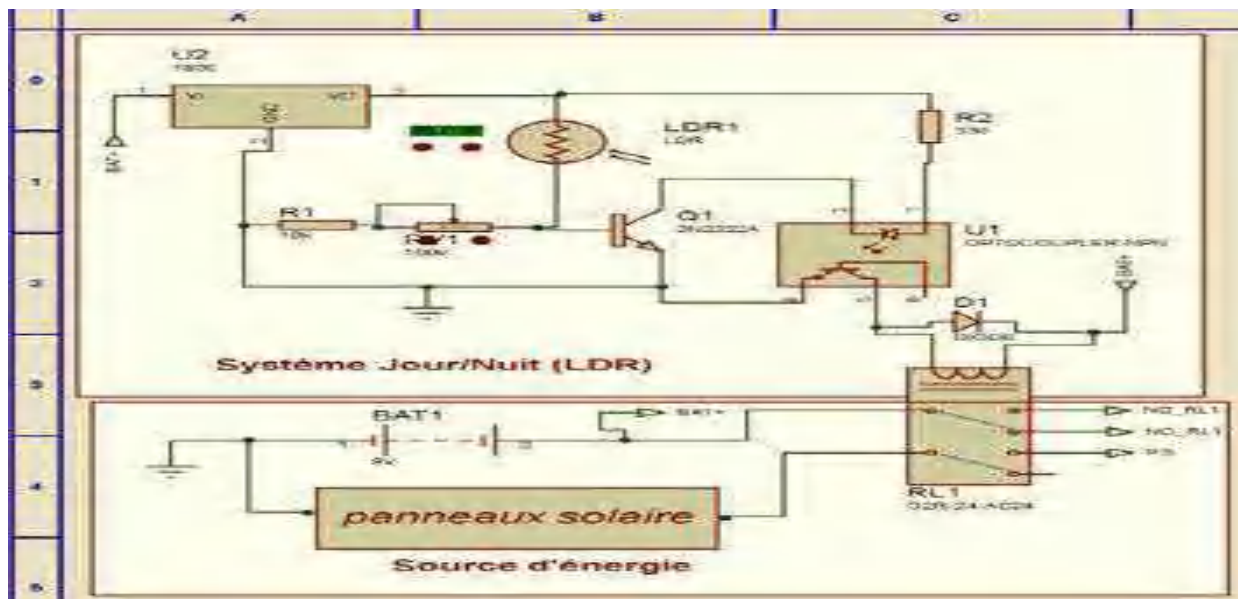


Figure 5.4 : Montage Globale

1.4.2.1 Système jour/nuit :

Ce montage joue le rôle d'un commutateur automatique : Il détecte la lumière à l'aide d'un LDR ce qui sature le transistor Q1 puis la diode de l'optocoupleur qui, de même, sature son transistor pour exciter la bobine du relais puis fermer le contact ouvert.

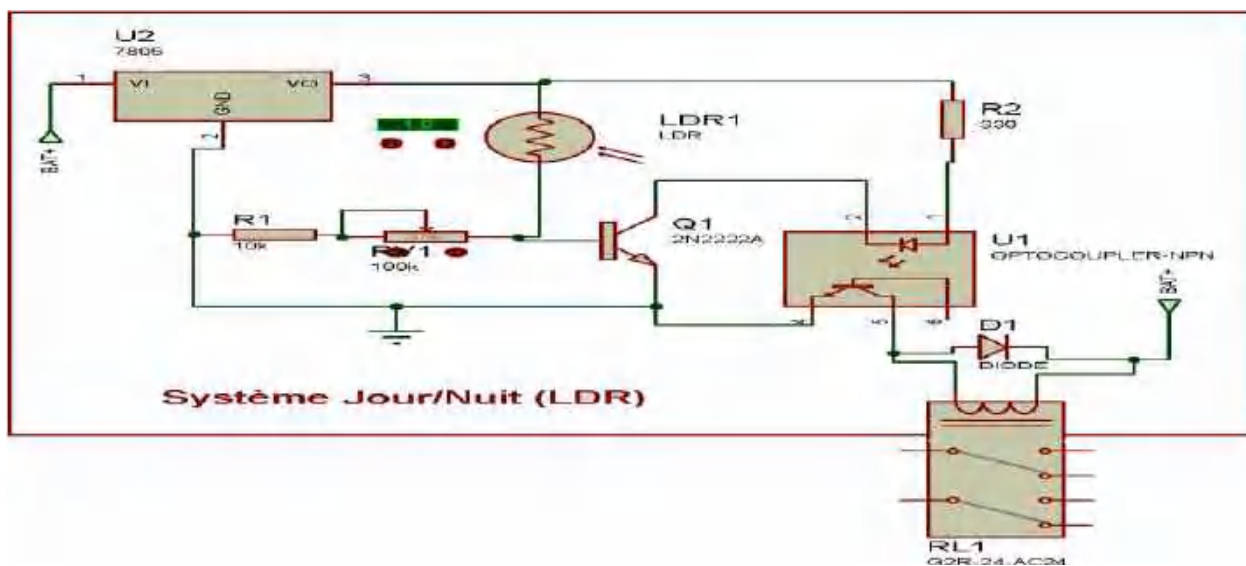


Figure 5.5 : Montage LDR

1.4.2.2 Basculement charge/alimentation :

Pendant la nuit, le système est alimenté par des accumulateurs qui se chargent pendant la journée à l'aide des panneaux solaires. La figure suivante représente le montage de basculement entre la batterie et les panneaux solaires.

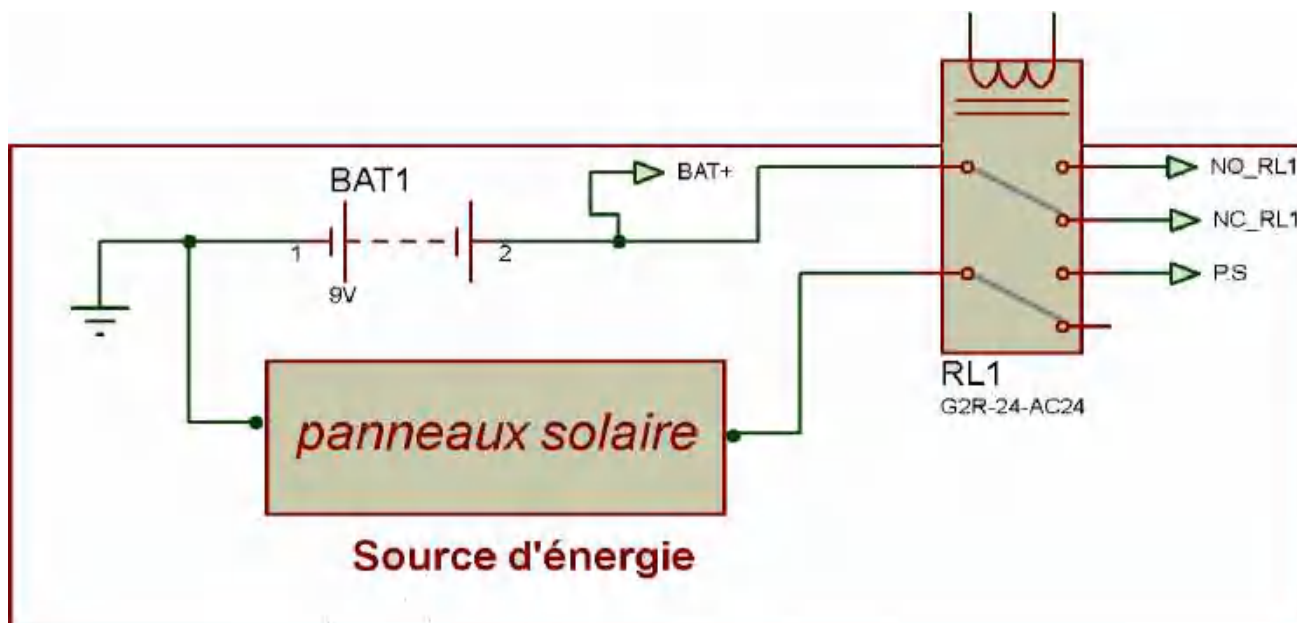


Figure 5.6 : Basculement Charge/alimentation

1.4.2.3 Chargement de la batterie :

Le jour, la batterie se charge à l'aide des panneaux. Une fois chargée, une LED verte s'allume. Cependant, lorsque la batterie est déchargée (en atteignant une tension seuil réglable avec un potentiomètre), une LED rouge s'allume.

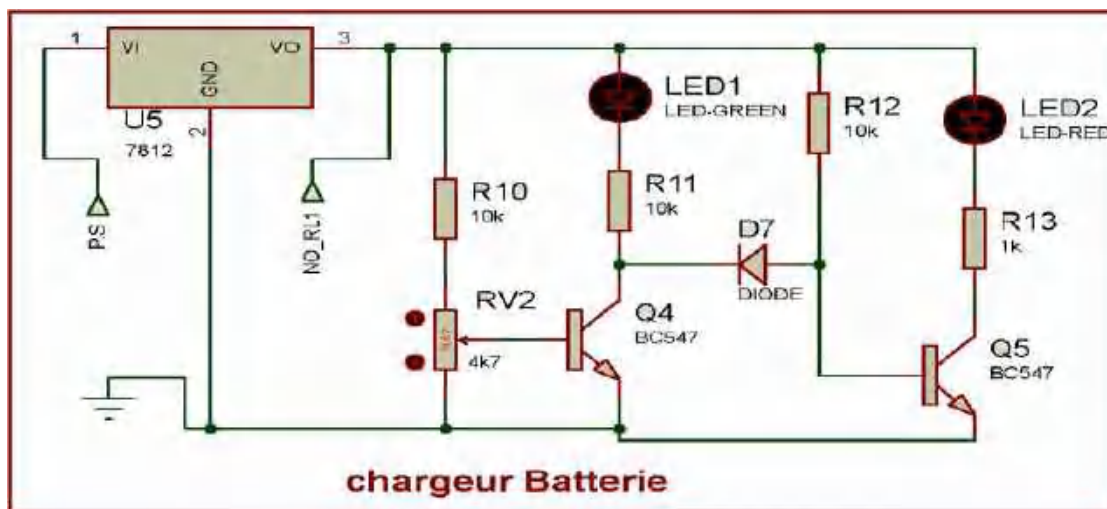


Figure 5.7 : Chargeur de batterie

1.5 Choix des composants

a. La carte Raspberry Pi

Compte tenu des exigences de performance relativement élevées du traitement de l'image en général et de l'équipement actuellement à la disposition à l'entreprise, la carte Raspberry Pi 3 modèle B, qui est une plateforme embarquée relativement peu coûteuse et puissante, reste le choix évident pour réaliser ce projet. Les spécifications matérielles prises en compte pour la RPi est présenté dans l'annexe 1.

Un autre facteur ayant contribué à ce choix est la disponibilité du module de Picamera, capable de capturer des vidéos hautes définitions ainsi que des images fixes. Il suffit simplement de mettre à jour le software de RPi à la dernière version. Il peut aussi facilement être utilisé dans les applications basées sur la bibliothèque OpenCV, comme il est pris en charge par de nombreuses bibliothèques tierces, aussi que toute caméra Web USB peut être utilisée.



Figure 5.8 – RPi 3 model B

b. Caméra Pi de vision nocturne

La Picamera de vision nocturne qu'on a choisie, se branche directement sur le connecteur CSI du Raspberry Pi et dispose de deux projecteurs à LED infrarouges de haute intensité pour l'enregistrement des vidéos dans la nuit ! Les LEDs infrarouges sont alimentées directement par le port CSI et peuvent éclairer une zone d'un porté de 8 m ! Lors des tests, les meilleures images ont été capturées à une distance de 3 à 5 m. La caméra dispose également d'un objectif à focale variable de 3,6 mm et d'un angle de vision de 60 degrés, avec une résolution du capteur de 1080 p.



Figure 5.9 – Pi caméra de vision nocturnal

c. Capteur de mouvement PIR *HC-SR501*

Ce capteur détecte un mouvement jusqu'à 20 pieds de distance. Il est très similaire au capteur de mouvement utilisé dans les systèmes de sécurité à domicile.



Figure 5.10 – Capteur de mouvement PIR

Ceci est un module infrarouge passif sophistiqué de capteur qui fait le « gros » de la détection du mouvement, le filtrage, etc... La puce BIS001 IC à bord gère tout le traitement du signal et émet un signal numérique lorsqu'un mouvement est détecté. La sortie numérique sur ce module est très facile à utiliser avec la Raspberry Pi et autre microprocesseur ou microcontrôleur.

Par ailleurs, le module contient des pots de finition qui vous permettent d'ajuster le délai pour éviter le re déclenchement rapide de la sortie.

d. Capteur de lumière LDR

La photorésistance LDR est un composant dont la valeur en ohms dépend de la lumière à laquelle il est exposé, qui réalise la conversion d'un signal lumineux en signal électrique.



Figure 5.11 – Capteur de lumière LDR

La résistance de LDR varie en fonction de l'intensité de lumière qu'elle reçoit. Si elle reçoit une lumière faible (l'obscurité) sa valeur R est très élevée à environ de $10\text{ M}\Omega$. Si l'intensité de la lumière augmente, sa valeur R descendra jusqu'à quelques dizaines d'ohms. Nous pouvons alors dire que la résistance de la LDR est inversement proportionnelle à la lumière reçue comme le présente la figure 5.12.

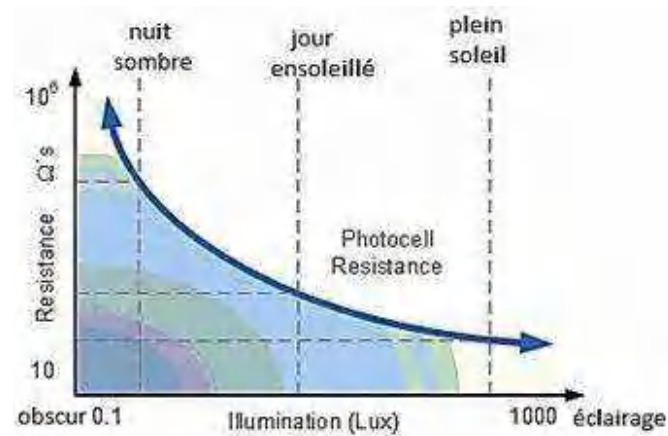


Figure 5.12 – Caractéristique résistance/éclairage

Après le choix des matériels ainsi que la conception électrique et technique du système, on va passer à la partie pratique qui comprend tout ce qui concerne la réalisation du modèle et le test des performances de ce dernier, qui seront présentés dans le chapitre suivant.

2) Evaluation du prototype et étude technico-économique

2.1 Réalisation du Prototype

Pour la réalisation du prototype, on a utilisé comme matière principale du Karton Duplex pour garantir une bonne robustesse, sur laquelle on a fixé les différents composants, après l'avoir décoré avec du cuir, des papiers autocollants et des ruban adhésifs afin qu'il semble plus réel. Voir le prototype final dans la figure 6.1.



Figure 6.1 – Prototype final

2.2 Évaluation du système

Comme nous l'avons déjà mentionné, au coucher du soleil, le système d'éclairage s'active automatiquement à l'aide du capteur de lumière LDR, à ce stade, le module caméra nocturnal - Pi caméra – utilisé est en train de prendre un enregistrement vidéo avec une résolution de 1080p, permettant ainsi notre carte Raspberry pi de traiter les captures vidéo en temps réel à l'aide du modèle SSDLite-Mobilenet-V2 pour détecter les objets qui empruntent la route pendant la nuit.

L'interface de détection est présentée dans la figure 6.2 suivante.



Figure 6.2 – L'image résultat

Le rectangle rouge représente la zone de détection du système, car ce dernier est capable de reconnaître les objets dans l'entier de l'image enregistrée par la Picamera, mais la détection de la classe de l'objet (humain, voiture, moto, chat, chien, ...) se fait à l'intérieur du rectangle rouge qu'on a appelé *Detection Box*.

FPS représente le nombre des images traité par seconde. Avec le modèle de détection SSDLite-Mobilenet-V2 on a obtenu des résultats qui varient entre 0.72 FPS et 1.04 FPS, qu'est assez mieux que le modèle YOLO-Tiny qui atteignant un taux variant entre 0.09 et 0.31 FPS.

Dernièrement l'équipe de recherche de Google publiant une version lite de TensorFlow, la solution officielle pour l'exécution de modèles d'apprentissage automatique sur des appareils mobiles et intégrés avec des ressources matériels limités. Il permet une inférence d'apprentissage automatique sur machine avec une faible latence et une petite taille binaire sous Android, iOS, Raspberry Pi OS, etc. TensorFlow Lite utilise pour cela de nombreuses techniques, telles que les noyaux quantifiés qui permettent des modèles plus légers et plus rapides (calculs à virgule fixe) [75].

D'après [75], le modèle SSDLite-Mobilenet-V2 avec la version du TensorFlow lite donne un résultat de 1.73 FPS, qu'est beaucoup mieux que la version normale mais avec une précision qui est plus petit à cette dernière.

Le tableau 5.1. Résume le nombre des FPS moyenne traité par les trois modèles qu'on a cité précédemment.

Table 6.1 – Le nombre des FPS traités par les modèles YOLO, SSDLite-Mobilenet-V2 et SSDLite-Mobilenet-V2 avec TFLite

Modèle	FPS
YOLO Tiny	0.32
SSDLite-Mobilenet-V2	1.04
SSDLite-Mobilenet-V2 avec TensorFlowLite	1.73

Et pour que le système puisse décider de changer l'état de fonctionnement, il faut détecter l'un d'objets appartient aux classes que nous avons identifiées dans le programme (humain, voiture, moto, ...). Lorsque cette condition est vérifiée, le système cherche les coordonnées centrales de l'objet en consultant les variables de la case [0][0]. Les cases [0][0] contiennent les coordonnées des objets détectés (ymin, xmin, ymax, xmax)

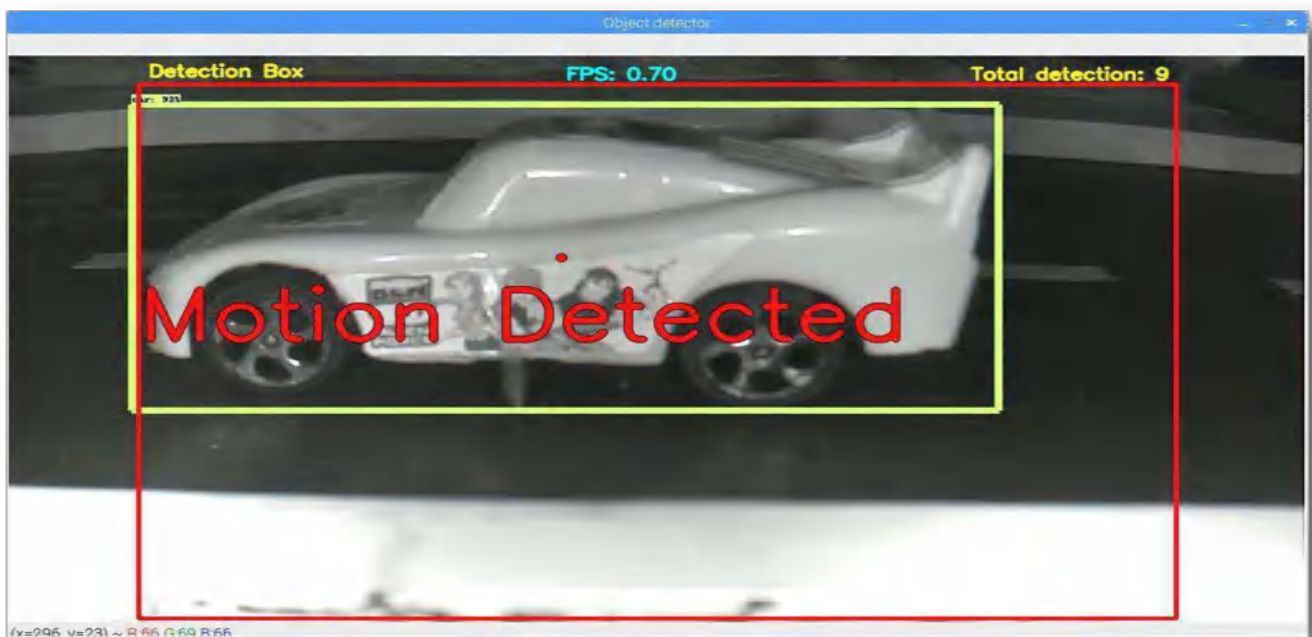


Figure 6.3 – Exemple d'une voiture détecté par le système

Ensuite, lorsque le centre de la boîte de l'objet se trouve à l'intérieur du rectangle (représenté par un point rouge), le système dirige l'action vers les capteurs de mouvement et lorsque l'objet passe devant ces capteurs, les lampadaires seront éclairés avec leur intensité maximale.

Enfin, nous pouvons dire que le taux de 1 FPS obtenu par le modèle de choix pour ce projet est prometteur mais pour permettre un suivi plus rapide et strict de l'objet on peut opter pour des cartes plus puissantes que la Raspberry Pi, telles que la carte Udoo X86, qui 10 fois plus puissante que carte Raspberry Pi, mais bien sûr plus chère que cette dernière.

2.3 Étude technico-économique

Cette partie, bien qu'elle soit la dernière à être évoquée, reste l'une des plus importantes puisqu'elle détermine le coût de projet et par la suite la survie du projet, et aussi de savoir si l'affaire est rentable. Dans le but de maximiser la rentabilité d'une entreprise il faut :

- + Sélectionner les projets d'investissement qui ajoutent de la valeur à l'entreprise, c'est-à-dire qui rapportent plus qu'ils ne coûtent.
- + Optimiser les financements pour réduire, ou stabiliser, le coût du financement des projets.

2.3.1 La range entre les lampadaires :

Chaque poteau a une portée qui représente la portée de détection de son capteur de mouvement, en prenant l'exemple de deux poteaux successifs on peut remarquer que la distance peut jouer un rôle très important dans l'optimisation d'énergie, par exemple dans le cas où la distance entre deux lampadaires successifs est assez petite une quantité considérable d'énergie sera perdue dans une zone de croisement de la lumière provenant de ses deux lampadaires comme le montre la figure ci-dessous, d'où la nécessité de respecter la distance entre les lampadaires en tenant en considération la portée de l'angle de détection des capteurs de mouvement associée à chaque lampadaire ainsi que la hauteur des poteaux installés.

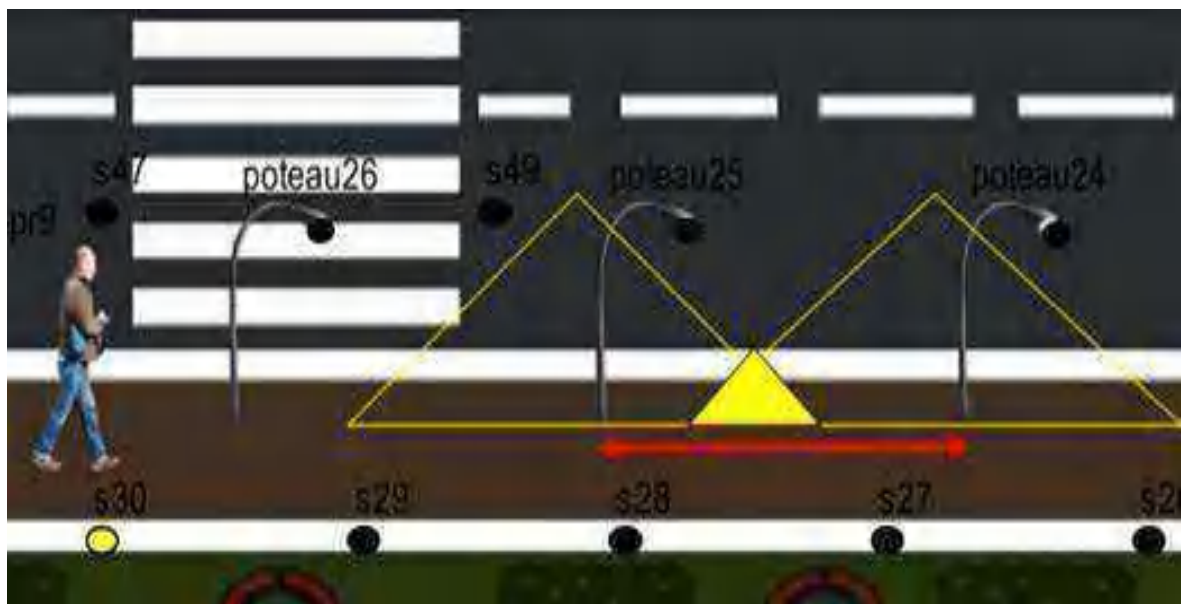


Figure 6.4 – la distance optimale entre les lampadaires




2.3.2 Cout de l'investissement :



On discerne tous ce qui est informations concernant les dépenses, prix de matériels et l'estimation du coût totale des équipements nécessaires pour l'implantation du projet. Cela sera représenté sous forme d'un bilan qui mettra en évidence les dépenses prévisionnelles.

En se référant sur cette fiche prévisionnelle qui détermine les équipements à installer ainsi que pour conduire le projet vers une réalisation palpable.

Le tableau suivant présente une estimation des prix (avec TVA inclus) pour la réalisation du projet :

Table 6.2 – Prix des équipements nécessaires par unité

Équipement	Quantité	Prix unitaire (DH)	Prix total (DH)
Lampadaire a panneau solaire (100W) 	40	360	14.400
Détecteur de mouvement infrarouge (Legrand) 	40	200	8.000
Cellule de mesure de luminosité 	1	200	200

<p>Carte Raspberry Pi</p> 	1	800	800
<p>Caméra infrarouge</p> 	1	300	300
Prix Totale :			23.700

Pour l'implantation du système dans un district ou un quartier routier d'une distance de couverture De 1Km avec une distance de séparation de 25 m entre chaque lampadaire on aura besoin de 40 lampadaires, 40 Détecteurs de mouvements, 1 camera de surveillance pendant la nuit, une cellule de mesure de luminosité pour activer notre système d'éclairage seulement pendant la nuit et en fin la carte Raspberry Pi comme unité de commande et de traitement.

Et pour cela le cout global estimé pour l'implantation initiale de notre système sera de **23.700 DH**

Mais bien évidemment ce cout peut être réduit après jusqu'à 50 % grâce à l'achat en gros du matériel et non pas par unité chez le fournisseur contractuelle de l'entreprise « le groupe industrielle Legrand »

2.3.3 Estimation de la consommation d'énergie électrique :

La consommation d'énergie électrique total par an liée à l'éclairage public au Maroc d'après les derniers statistiques est presque de 1GWh ce qui est extrêmement élevée par rapport à la consommation d'électricité au niveau du modèle d'éclairage public qu'on a proposé.

Ce dernier peut optimiser la consommation d'énergie avec un pourcentage qui peut atteindre jusqu'à 75% selon la densité de circulation des objets d'intérêt sur la route pendant la nuit.

Cette estimation est représentée dans la figure descriptif ci-dessous.

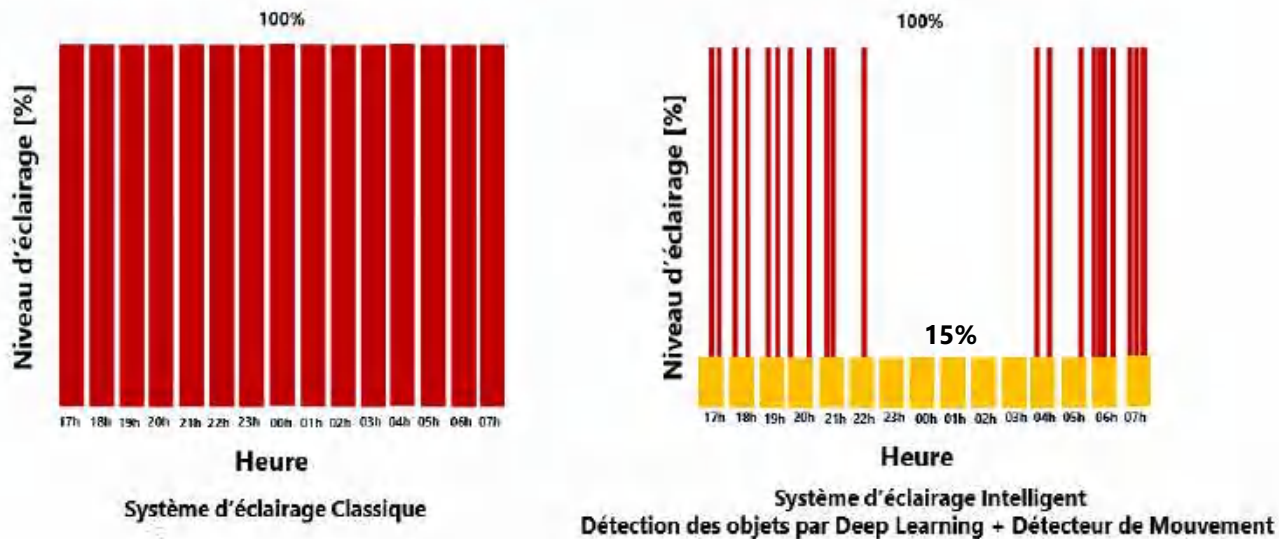


Figure 6.5 – Comparaison entre le système d'éclairage public classique et le système d'éclairage intelligent proposé en termes de consommation d'énergie électrique

2.3.4 Estimation du retour sur investissement :

Etant donnée que le cout d'investissement total est de 23 700 DH et que chaque lampadaire utilisé consomme 100W par jour lorsqu'il est allumé pendant toute la nuit.

Or le modèle d'éclairage public intelligent qu'on a proposé peut optimiser la consommation de l'énergie électrique jusqu'à 75% par jour.

Alors la consommation totale d'énergie électrique en KW/jour dans les conditions parfaites au niveau de notre système est calculée comme suivant :

$$\text{Consommation (en KW/par jour)} = 40 * 100 * 0.25 = 1000 \text{ KW}$$

En comparant avec les systèmes d'éclairages publics classiques, et en utilisant le même type et des lampadaires et la même quantité de 40 lampadaires par un Km tel que chaque unité consomme 100W par jour en termes de puissance.

Alors la consommation d'électricité en cas d'un système d'éclairage classique sera de **4000 KW/jour**.

D'où, on déduit qu'on peut optimiser jusqu'à 3000KW par jour.

Et sachant que le prix moyenne d'électricité au Maroc aujourd'hui pour 1KW est de 1.21 DH, Alors le gain total en termes d'argent par jour est : **3000 * 1.21 = 3750 DH**.

Donc pour compenser le cout total d'investissement qui est de **23 700 DH**, on aura besoin de **23 700 / 3750 = 6.32 Jr**, qui est approximativement une semaine.

Conclusion générale

Ce rapport présente le résultat d'un travail soutenu et assidu qui s'inscrit dans le cadre de la réalisation d'un projet de fin d'étude pour l'obtention du diplôme d'ingénieur d'état en génie électrique, systèmes embarqués et télécommunication.

L'éclairage public est généralement un éclairage de chaussée offrant une amélioration de la visibilité. Il est utilisé lorsqu'il y a fréquemment coexistence de piétons et de véhicules, c'est-à-dire essentiellement à l'intérieur des localités, dans les zones bâties et le long des autoroutes et des voies de circulations rapides.

Dans notre étude nous avons réalisé un prototype de système d'éclairage public intelligent en exploitant des techniques du Deep Learning et du traitement d'images, dont le but est d'économiser la consommation d'énergie électrique avec une estimation initiale d'optimisation énergétique qui peut atteindre jusqu'à 75% et cela pour rendre nos rues plus intelligentes et écologiques, en se référant à de nouvelles technologies.

En général, on peut dire que les résultats du test du système sont bons, puisque la détection des objets se fait avec une précision assez élevée (supérieure à 80%) pour la plupart des objets, avec un taux approximativement d'une image par seconde sur la carte Raspberry pi, ce qui permet une réaction acceptable au système avec les mouvements d'objets tout au long de la route (vitesse de détection).

Ce travail peut se poursuivre pour permettre en plus de faire la supervision du système à distance à l'aide d'un module GSM ou WIFI comme L'ESP32 pour se renseigner de l'état du système et de contrôler la consommation d'énergie, ainsi que la gestion intelligente de l'éclairage ambiante, afin d'assurer à l'utilisateur une traçabilité sur son environnement en termes d'éclairage et de consommation.

En fin nous souhaitons vivement que ce projet puisse servir comme élément de base pour d'autres études plus approfondies pour le faire intégrer sous des systèmes plus complexes.

Bibliographie

- [1] A. Guan, S. H. Bayless and R. Neelakantan. "Trends in Computer Vision". Intelligent Transportation Society of America, May 2012. [2022]. Available : <http://www.itsa.org/knowledgecenter/technologyscan/1301>.
- [2] "What is a Raspberry Pi?". Raspberry Foundation, 2015. [2022]. Available : <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>.
- [3] E. Upton, "Programming the Raspberry Pi". [2022]. Available : https://www.element14.com/community/servlet/JiveServlet/previewBody/44424-102-1-240260/element14_RPi_Webinar_040412_V1.0_FINAL.pdf.
- [4] R. Longbottom, "Roy Longbottom's Raspberry Pi, Pi 2 and Pi 3 Benchmarks," 5 2017. [2022]. Available : <http://www.roylongbottom.org.uk/Raspberry%20Pi%20Benchmarks.htm#anchor24b>.
- [5] M. Larabel, "Raspberry Pi 3 Model B+ Benchmarks," 22 3 2018. [Online]. Available : <https://www.phoronix.com/scan.php?page=article&item=raspberrypi-3-bplus&num=1>
- [6] P. F. Felzenszwalb, R. B. Girshick, D. Mcallester, and D. Ramanan, "Object detection with discriminatively trained part-based models," IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 9, p. 1627, 2010.
- [7] K. K. Sung and T. Poggio, "Example-based learning for view-based human face detection," IEEE Trans. Pattern Anal. Mach. Intell., vol. 20, no. 1, pp. 39–51, 2002.
- [8] C. Wojek, P. Dollar, B. Schiele, and P. Perona, "Pedestrian detection : An evaluation of the state of the art," IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 4, p. 743, 2012.
- [9] H. Kobatake and Y. Yoshinaga, "Detection of spicules on mammogram based on skeleton analysis." IEEE Trans. Med. Imag., vol. 15, no. 3, pp. 235–245, 1996.
- [10] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe : Convolutional architecture for fast feature embedding," in ACM MM, 2014.
- [11] A. krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in NIPS, 2012.
- [12] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2d pose estimation using part affinity fields," in CVPR, 2017.
- [13] Z. Yang and R. Nevatia, "A multi-scale cascade fully convolutional network face detector," in ICPR, 2016.
- [14] C. Chen, A. Seff, A. L. Kornhauser, and J. Xiao, "Deepdriving : Learning affordance for direct perception in autonomous driving," in ICCV, 2015.

- [15] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in CVPR, 2017.
- [16] A. Dundar, J. Jin, B. Martini, and E. Culurciello, "Embedded streaming deep neural networks accelerator with applications" IEEE Trans. Neural Netw. & Learning Syst., vol. 28, no. 7, pp. 1572-1583, 2017.
- [17] A. Stuhlsatz, J. Lippel, and T. Zielke, "Feature extraction with deep neural networks by a generalized discriminant analysis." IEEE Trans. Neural Netw. & Learning Syst., vol. 23, no. 4, pp. 596–608, 2012.
- [18] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in CVPR, 2014.
- [19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn : Towards real-time object detection with region proposal networks," in NIPS, 2015, pp. 91–99.
- [20] D. G. Lowe, "Distinctive image features from scale-invariant key-points," Int. J. of Comput. Vision, vol. 60, no. 2, pp. 91–110, 2004.
- [21] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in CVPR, 2005.
- [22] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in ICIP, 2002.
- [23] C. Cortes and V. Vapnik, "Support vector machine," Machine Learning, vol. 20, no. 3, pp. 273–297, 1995.
- [24] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," J. of Comput. & Sys. Sci., vol. 13, no. 5, pp. 663–671, 1997.
- [25] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, pp. 1627–1645, 2010.
- [26] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," 2012. [2022]. Available <https://www.cs.toronto.edu/~fritz/absps/imagenet.pdf>
- [27] D. Lowe, "Object recognition from local scale-invariant features," in Proceedings of the eventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 1999.
- [28] N. Dalal og B. Triggs, «Histograms of oriented gradients for human detection,» i CVPR'05, San Diego, CA, USA, 2005.
- [29] P. F. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume : 32, Issue : 9), pp. 1627-1645, 9 2010.
- [30] Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE (Volume : 86, Issue : 11), pp. 2278-2324, Nov 1998.
- [31] S. Sabour, N. Frosst and G. E. Hinton, "Dynamic Routing Between Capsules," 7 11 2017. [2022]. Available :<https://arxiv.org/abs/1710.09829>.

- [32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, “Dropout : A Simple Way to Prevent Neural Networks from Overfitting,” *Journal of Machine Learning Research*, pp. 1929-1958, 15 6 2014.
- [33] W. Pitts and W. S. McCulloch, “How we know universals the perception of auditory and visual forms,” *The Bulletin of Mathematical Biophysics*, vol. 9, no. 3, pp. 127–147, 1947.
- [34] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representation by back-propagation of errors,” *Nature*, vol. 323, no. 323, pp. 533–536, 1986.
- [35] G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Sci.*, vol. 313, pp. 504–507, 2006.
- [36] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath et al., “Deep neural networks for acoustic modeling in speech recognition : The shared views of four research groups,” *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.
- [37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet : A large-scale hierarchical image database,” in *CVPR*, 2009.
- [38] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A.-r. Mohamed, and G. Hinton, “Binary coding of speech spectrograms using a deep autoencoder,” in *INTERSPEECH*, 2010.
- [39] G. Dahl, A.-r. Mohamed, G. E. Hinton et al., “Phone recognition with the mean-covariance restricted boltzmann machine,” in *NIPS*, 2010.
- [40] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing coadaptation of feature detectors,” *arXiv :1207.0580*, 2012.
- [41] S. Ioffe and C. Szegedy, “Batch normalization : Accelerating deep network training by reducing internal covariate shift,” in *ICML*, 2015.
- [42] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat : Integrated recognition, localization and detection using convolutional networks,” *arXiv : 1312.6229*, 2013.
- [43] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *CVPR*, 2015.
- [44] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv : 1409.1556*, 2014.
- [45] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016.
- [46] v. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *ICML*, 2010.
- [47] J. Petterson and A. Gibson, *Deep learning : A practitioner’s approach*, O’Reilly Media, Inc, 2017.
- [48] Wikipedia, “Hyperparameter (machine learning),” 23 4 2018. [2022].
Available :[https://en.wikipedia.org/wiki/Hyperparameter_\(machine_learning\)](https://en.wikipedia.org/wiki/Hyperparameter_(machine_learning)).
- [49] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press, 2016.

- [50] F. Chollet, “How convolutional neural networks see the world : An exploration of convnet filters with Keras,” The Keras Blog, 30 1 2016. [2022]. Available : <https://blog.keras.io/how-convolutional-neural-networks-see-the-world.html>.
- [51] L. Taylor and G. Nitschke, “Improving Deep Learning using Generic Data Augmentation,” 20 8 2018. [2022]. Available : <https://arxiv.org/abs/1708.06020>.
- [52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, “Dropout : A Simple Way to Prevent Neural Networks from Overfitting,” Journal of Machine Learning Research, pp. 1929-1958, 15 6 2014.
- [53] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” 10 4 2015. [2022]. Available : <https://arxiv.org/abs/1409.1556>.
- [54] M. Lin, Q. Chen and S. Yan, “Network In Network,” 4 3 2014. [2022]. Available : <https://arxiv.org/abs/1312.4400>.
- [55] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke and A. Rabinovich, “Going deeper with convolutions,” 17 9 2014. [2022]. Available : <https://arxiv.org/abs/1409.4842>.
- [56] K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition,” 10 12 2015. [Online]. Available : <https://arxiv.org/abs/1512.03385>.
- [57] S. Ioffe and C. Szegedy, “Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift,” 2 3 2015. [2022]. Available : <https://arxiv.org/abs/1502.03167>.
- [58] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, A. Weyand, M. Andreetto and H. Adam, “MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications,” 17 4 2017. [2022]. Available : <https://arxiv.org/pdf/1704.04861>.
- [59] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama and K. Murphy, “Speed/accuracy trade-offs for modern convolutional object detectors,” 25 4 2017. [2022]. Available : <https://arxiv.org/abs/1611.10012>.
- [60] R. Girshick, J. Donahue, T. Darrell and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” 22 10 2014. [2022]. Available : <https://arxiv.org/abs/1311.2524>.
- [61] D. Parthasarathy, “A Brief History of CNNs in Image Segmentation : From R-CNN to Mask R-CNN,” 22 4 2017. [2022]. Available : <https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4>.
- [62] R. Girshick, “Fast R-CNN,” 27 9 2015. [2022]. Available : <https://arxiv.org/abs/1504.08083>.
- [63] S. Ren, K. He, R. Girshick and J. Sun, “Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks,” 6 1 2016. [2022]. Available : <https://arxiv.org/abs/1506.01497>.
- [64] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu and A. C. Berg, “SSD :

Single Shot MultiBox Detector,” 29 12 2016. [2022]. Available :
<https://arxiv.org/abs/1512.02325>.

[65] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, “You Only Look Once : Unified, Real-Time Object Detection,” 9 5 2016. [2022]. Available :
<https://arxiv.org/abs/1506.02640>.

[66] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, “Focal Loss for Dense Object Detection,” 7 2 2018. [2022]. Available : <https://arxiv.org/abs/1708.02002>.

[67] J. Redmon and A. Farhadi, “YOLO9000 : Better, Faster, Stronger,” 25 12 2016. [2022]. Available :<https://arxiv.org/abs/1612.08242>.

[68] J. Redmon and A. Farhadi, “YOLOv3 : An Incremental Improvement,” 8 4 2018. [2022]. Available :<https://arxiv.org/abs/1804.02767>.

[69] J. Redmon, “YOLO : Real-Time Object Detection,” [2022]. Available :
<https://pjreddie.com/darknet/yolo/>.

[70] G. Howard and M Zhu, “MobileNets : Efficient Convolutional Neural Networks for Mobile Vision Applications,” 17 4 2017 . [2022]. Available :
<https://arxiv.org/pdf/1704.04861.pdf>.

[71] R. Lambert, “MobileNet, une reconnaissance d’images temps réel et embarquée surpuissante,” 20 1 2018. [2022]. Available :<http://penseeartificielle.fr/mobilenet-reconnaissance-images-temps-reel-embarque/>.

[72] "Understanding SSD MultiBox-Real-Time Object Detection In Deep Learning", [2022]. Available :<https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab>.

[73] Y. Li, H. Huang, " Research on a Surface Defect Detection Algorithm Based on MobileNet-SSD"

[74] "How to Install Google Protocol Buffers on Raspberry Pi and Test with Python", 2016. [2022]. Available :<http://osdevlab.blogspot.com/2016/03/how-to-install-google-protocol-buffers.html>

[75] K. Yamkovyi, "Mobile object detector with TensorFlow Lite". [2022]. Available :https://medium.com/@Quantum_inc/mobile-object-detector-with-tensorflow-lite-9e2c278922d0/

[76] <https://hal.archives-ouvertes.fr/hal-03531390/document> [Realisé par Julia Cohen, Carlos F Crispim-Junior, Jean-Marc Chiappa, Laure Tougne]

Annexe 1

Les caractéristiques de chaque composant utilisé dans ce projet

- Raspberry Pi 3 B :

- Processeur: Broadcom BCM2837B0, Cortex-A53 64-bit SoC @ 1.4GHz ;
- Mémoire : 1GB LPDDR2 SDRAM ;
- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE ;
- Gigabit Ethernet USB 2.0 (300 Mbps) ;
- 4 × USB 2.0 ports ;
- GPIO : 40 PIN GPIO ;
- Vidéo et son ;
- Sortie HDMI ;
- MIDI DSI port écran ;
- MIPI CSI port camera ;
- 4 sorties stéréo Output et port vidéo composite ;
- Multimedia : H.264, MPEG-4 decode (1080p30) ;
H.264 encode (1080p30) ; OpenGL ES 1.1, 2.0 graphics ;
- Support carte SD : Micro SD format pour chargement de la distribution et la sauvegarde de données ;
- Connecteur 5V/2.5A DC via connecteur Micro USB ;
- Alimentation via le Header GPIO ;
- Via le Power Over Ethernet (POE) - Prévoir l'acquisition de la carte POE HAT pour connecter la PI ;
- Température d'utilisation : entre 0° C et 50° C ;

- Pi Camera Night Vision :

- Caméra avec vision de nuit ;
- 5 mégapixels OV5647 capteur ;
- Capteur meilleure résolution : 1080 p ;
- 4 vis trous : Utilisé pour la fixation IR lumières, Et pour 3.3 v électrique conduction ;
- Focal Longueur : 2.1 ;
- Diagonale angle : 130 degrés ;
- Dimension : 25mm x 24mm ;

- Capteur de Mouvement Infrarouge PIR :

- Numéro du modèle : HC-SR501 ;
- Sensor distance : 5-10m ;
- Working Temperature : -10C +50C ;
- Environmental Protective : IP44 ;
- Sensor degree : 140 Degree Cone angle ;
- Environmental illumination : 2-2000LUX (adjustable) ;
- Load power : 5W-100W ;
- Certification : CCC,CE,LVD,RoHS ;

- LDR Photorésistance :

- Tension De Fonctionnement : 3.5 v-5 v ;
- Forme de sortie : Numérique Commutateur spectacle (0 et 1) ;

Annexe 2

Code Python pour détecter les objets en temps réel

```
1 import os
2 import cv2
3 import numpy as np
4 from picamera.array import PiRGBArray
5 from picamera import PiCamera
6 import tensorflow as tf
7 import argparse
8 import sys
9
10 IM_WIDTH = 1280
11 IM_HEIGHT = 720
12
13 camera_type = 'picamera'
14 parser = argparse.ArgumentParser()
15 parser.add_argument('--usbcam', help = 'Use a USB webcam instead of picamera', action = 'store_true')
16 args = parser.parse_args()
17
18 if args.usbcam :
19     camera_type = 'usb'
20
21 sys.path.append('.')
22
23 from utils import label_map_util
24 from utils import visualization_utils as vis_util
25 MODEL_NAME = 'ssdlite_mobilenet_v2_coco'
26 CWD_PATH = os.getcwd()
27 PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME, 'frozen_inference_graph.pb')
28
29 PATH_TO_LABELS = os.path.join(CWD_PATH, 'data', 'mscoco_label_map.pbtxt')
30
31 NUM_CLASSES = 90
32
```

```

label_map = label_map_util.load_labelmap(PATH_TO_LABELS)
categories = label_map_util.convert_label_map_to_categories(label_map, max_num_classes=4096, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(PATH_TO_ODT, 'r') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')
    sess = tf.Session(graph=detection_graph)

image_tensor = detection_graph.get_tensor_by_name('input:0')

detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')

detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')

detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')

num_detections = detection_graph.get_tensor_by_name('num_detections:0')

frame_rate_text = '\n'
time = cv2.getTextSize(frame_rate_text, font, 0.5, 20)[0] + 10
font = cv2.FONT_HERSHEY_SIMPLEX
camera_type = 'webcam'
camera = PiCamera()
camera.resolution = (DWIDTH, DHEIGHT)
camera.framerate = 15
rawCapture = PiRGBWindow(camera, size=(DWIDTH, DHEIGHT))
rawCapture.truncate(0)

```



```
66 for frame1 in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True) :
67     t1 = cv2.getTickCount()
68     frame = np.copy(frame1.array)
69     frame.setflags(write=1)
70     frame_expanded = np.expand_dims(frame, axis=0)
71     (boxes, scores, classes, num) = sess.run([detection_boxes, detection_scores, detection_classes, num_detections],
72     feed_dict={image_tensor : frame_expanded})
73     vis_util.visualize_boxes_and_labels_on_image_array(frame,
74     np.squeeze(boxes),
75     np.squeeze(classes).astype(np.int 32),
76     np.squeeze(scores),
77     category_index,
78     use_normalized_coordinates=True,
79     line_thickness=8,
80     min_score_thresh=0.40)
81     cv2.putText(frame, "FPS: {0:.2f}".format(frame_rate_calc), (30, 50), font, 1, (255, 255, 0), 2, cv2.LINE_AA)
82     cv2.imshow('Object detector', frame)
83     t2 = cv2.getTickCount()
84     time1 = (t2 - t1) / freq
85     frame_rate_calc = 1 / time1
86
87     if cv2.waitKey(1) == ord('q'):
88         break
89
90     rawCapture.truncate(0)
91
92 camera.close()
```