RÉPUBLIQUE DU SÉNÉGAL

Un Peuple – Un But – Une Foi

MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE ET DE L'INNOVATION

UNIVERSITE CHEIKH ANTA DIOP DE DAKAR

****

FACULTE DES SCIENCES ET TECHNIQUES

****

DEPARTEMENT DE MATHEMATIQUES ET INFORMATIQUE

SECTION INFORMATIQUE

MÉMOIRE DE MASTER INFORMATIQUE

Option : Réseaux et Télécommunications

Sujet : Système de détection et reconnaissance d'objets dans le transport routier au SÉNÉGAL

Présenté et soutenu publiquement par :

M. FRANÇOIS NDIÈRE DIOUF

Encadré par Professeur CHERIF DIALLO

Soutenu le : 11/01/2020

Jury

Pr. Cheikh Ahmadou Bamba GUEYE	Maître de conférences Titulaire, UCAD	Président
Dr. Ibrahima DIANE	Maître Assistant Titulaire, UCAD	Membre
Dr. Mohamed Ould DEYE	Maître Assistant Titulaire, UCAD	Membre
Pr. Ibrahima NIANG	Professeur Titulaire, UCAD	Membre

Année académique 2019

REMERCIEMENTS

Je rends louange à ALLAH (SWT), de m'avoir accordé la santé et la chance d'arriver jusque-là.

Je tiens à remercier tous les membres de la section informatique, en particulier ceux de la filière Réseaux et Télécommunications.

Je tiens à adresser chaleureusement mes remerciements à mon encadrant, Monsieur CHERIF DIALLO pour le temps et l'attention qu'il a bien voulu me consacrer me l'encadrement de ce mémoire.

Je ne pourrais pas conclure ces remerciements sans évoquer les précieux encouragements de la famille DIOUF, de ma belle-sœur, des camarades de chambre au campus, des camarades de classe et surtout de ma mère.

Elle a toujours trouvé les mots, les ressources et les prières pour me motiver.

Pour terminer, je remercie mes camarades de classe et toutes les personnes qui ont pu m'aider de près ou de loin pendant mon parcours et dont les noms ne figurent pas sur cette page. Sachez que vous avez toute ma gratitude.

RESUME

Ce mémoire présente la mise en place d'un système de détection et de reconnaissance des objets dans le trafic routier au Sénégal en utilisant le Deep Learning dans le but de réduire le nombre d'accidents de la route pour sauver des vies.

Lorsqu'un objet (piéton, un pneu, un tableau, …) est en approche du véhicule c'est-à-dire du champ de vision de la caméra installé sur la voiture le système fait la détection de l'objet puis la reconnaissance de ce dernier grâce à l'algorithme YOLO. Plusieurs algorithmes de réseaux de neurones artificiels permettent de faire la reconnaissance et la détection des objets, YOLO donne des résultats intéressants grâce à sa rapidité, sa capité de traitement des images (45 images par seconde) et sa précision.

Ce système est composé d'une carte nano Raspberry avec une caméra embarquée et une application web. Le système de détection et de reconnaissance des objets envoie les images capturées à une base de données. Cette dernière sera couplée avec un système de traitement des images qui permettra à l'application web d'afficher les objets détectés sur un itinéraire déjà parcourue.

Finalement nous avons conçu un système capable de détecter et reconnaitre des objets en utilisant un ordinateur pour servir de Raspberry Pi et de caméra afin d'exécuter YOLO qui est un algorithme du Deep Learning.

Mots clés :

Reconnaissance des objets, Deep Learning, Réseaux de neurones, YOLO, Raspberry Pi.

ABSTRACT

This thesis presents the establishment of a system for detecting and recognizing objects in road traffic in Senegal using Deep learning in order to reduce the number of road accidents to save lives.

When an object (passenger, tire, switchboard, etc.) is approaching the vehicle, i.e. the field of vision of the camera installed on the car, the system detects the object and then recognition of the latter thanks to the YOLO algorithm. Several artificial neural network algorithms make it possible to recognize and detect objects, YOLO gives interesting results thanks to its speed, its image processing capacity (45 images per second) and its precision.

This system consists of a Raspberry nano card with an on-board camera and a web application. The object detection and recognition system sends the captured images to a database. The latter will be coupled with an image processing system that will allow the web application to display the objects detected on a route already traveled.

Finally we designed a system capable of detecting and recognizing objects using a computer to serve as a Raspberry Pi and camera in order to execute YOLO which is a Deep learning algorithm.

Keywords:

Object recognition, Deep learning, Neural networks, YOLO, Raspberry Pi.

Table des matières

REMERCIEMENTS II
RESUME III
ABSTRACT IV
Tables des figures VIII
Liste des tableaux IX
Liste des acronymes X
Introduction générale 1
CHAPITRE 1 : la reconnaissance des objets 4
Introduction 5
1. Définition 5
2. Reconnaissance des formes 7
2.1. Introduction à l'Intelligence Artificielle 7
a. Segmentation d'images 9
b. Description d'images 11
c. Deep Learning 13
Conclusion 17
CHAPITRE2 : Etat de l'art sur les algorithmes du Deep Learning 18
Introduction 19
1. Fonctionnement d'un réseau de neurones 19
2. Les différents types de modèles 20
2.1. Les réseaux de neurones convolutifs 20
2.1.1. Inspiration 20
2.1.2. L'opération de convolution 21
2.1.3. La couche convolutif 22
2.1.4. La couche de pooling 23
2.1.5. Le perceptron 24
2.1.6. Quelques réseaux convolutifs célèbres 26
2.2. Réseaux de neurones récurrents 26
2.2.1. C'est quoi un RNN ? 27
2.2.2. Apprentissage 28
2.2.3. Application 30
2.3. Deep generative model 30
3. La fonction d'activation 31

3.1. Les caractéristiques 31
3.2. Quelques fonctions d'activation 31
3.2.1. Rectified Linear Units et leurs généralisations 31
3.2.2. Sigmoïde et tangente hyperbolique 33
4. La famille de modèles R-CNN 33
4.1. Le R-CNN 34
4.2. Le Fast R-CNN 35
4.3. Le Faster R-CNN 36
5. La famille de modèles YOLO 38
5.1. YOLO 38
5.2. YOLOv2 (YOLO900) et YOLOv3 40
6. SSD (**S**ingle **S**hot **D**etector) 41
7. Le software 41
7.1. Theano 42
7.2. TensorFlow 42
7.3. Keras 42
7.4. PyTorch 42
7.5. Caffee 43
7.6. CNTK 43
7.7. Caffe2 43
7.8. MXNet 44
7.9. Torche 44
7.10. Darkne 44
7.11. Comparaison des frameworks de l'IA 44
Conclusion 44
Chapitre 3 : mise en œuvre, analyse et conception 46
1. Logiciels et librairies utilisés dans l'implémentation 47
1.1. Analyse des besoins du système 47
1.1.1. Besoins fonctionnels 47
1.1.2. Besoins non fonctionnels 47
1.2. Matériels et logiciels 47
1.2.1. Matériels 47
1.2.2. Logiciels et plateforme 47
2. Conception 51
2.1. Diagramme du cas d'utilisation 51

2.2. Description des cas d'utilisation....52
2.3. Diagramme de séquence....54
2.4. Diagramme de classe....56
3. Architecture de la solution....56
3.1. Architecture matérielle proposée....56
3.2. Architecture fonctionnelle....57
4. Protocoles utilisés....61
4.1. Protocoles de streaming vidéo RTP, RTCP et RTSP....61
4.2. Protocole UDP....62
4.3. Le protocole http....62
5. Mise en œuvre....63
5.1. Outils de développement....63
5.2. Réalisation de la solution....63
Conclusion et perspective....70
Références....71

Tables des figures

Figure 1 présentation des tâches de vision par ordinateur de reconnaissance d'objets. 7
Figure 2a Intelligence Artificielle Figure 2b : de la fiction à la réalité 8
Figure 3 : exemple de détecteur de points d'intérêt : le détecteur de Harris. 12
Figure 4 : la relation entre l'intelligence artificielle, le machine Learning et le deep Learning 14
Figure 5 la différence de performance entre le Deep Learning et la plupart des algorithmes de ML en fonction de la quantité de données [23]. 16
Figure 6 le procédé du ML classique comparé à celui Deep Learning. 16
Figure 7 : description du fonctionnement d'un neurone 19
Figure 8 : exemple d'une convolution 2D [29]. 22
Figure 9: Interaction du Sparse 24
Figure 10: exemple d'une image 1000X1000. 24
Figure 11: Average pooling avec des cases 25
Figure 12: Entrée d'une image 2D dans un réseau convolutif, composé d'une couche convolutif, d'une fonction non linéaire, d'une couche MAX pooling et enfin un perceptron multi couche. 25
Figure 13 : (à gauche) un RNN (à droite), sa version déroulée Source : Nature 27
Figure 14 Backpropagation Through Time 29
Figure 15 : (à gauche) la fonction d'activation ReLu. 32
Figure 16: (a) Sigmoïde. (b) tangente hyperbolique 33
Figure 17: résumé de l'architecture du modèle R-CNN tiré des hiérarchies de fonctionnalités riches pour une détection précise des objets et une segmentation sémantique. 34
Figure 18 : résumé de l'architecture du modèle Fast R-CNN. 36
Figure 19: Résumé de l'architecture du modèle Faster R-CNN vers la détection d'objets en temps réel avec les réseaux de propositions de régions 37
Figure 20: résumé de prédiction par le modèle YOLO. 39
Figure 21: exemple de représentation choisie lors de la prévision de la position et de la forme de la boîte englobante. 40
Figure 22: comparaison des méthodes de détection en fonction de la précision par rapport à la vitesse. 41
Figure 23 : diagramme de cas d'utilisation. 51
Figure 24 : le diagramme de séquence pour l'authentification 55
Figure 25 : diagramme de séquence de consultation des données. 55
Figure 26: diagramme de séquence de la gestion des comptes par l'administrateur 55
Figure 27: diagramme de classe 56
Figure 28: architecture de la solution proposée 57
Figure 29 : architecture fonctionnelle du système 57
Figure 30: encapsulation des paquets RTP 61
Figure 31: liste des classes de Yolov3 64
Figure 33 : ensemble des images du répertoire imgs 67
Figure 34: Résultats de la sortie de la commande 68
Figure 35 : suite des résultats avec un tableau de résumé 68
Figure 36: page de connexion sur l'application web 69
Figure 37: page d'inscription 69

Liste des tableaux

Tableau 1: les différentes étapes constituants un système de détection d'objets 10
Tableau 2: comparaison des algorithmes de détection d'objets 38
Tableau 3:comparaison des frameworks 44
Tableau 4: Yolo avantages et inconvénients.....49
Tableau 5: le formalise de la description des cas d'utilisation. 52
Tableau 6: la description du cas d'utilisation d'authentification. 52
Tableau 7: description du cas d'utilisation consultation des données. 53
Tableau 8: description du cas d'utilisation traitement des données 53
Tableau 9: description du cas d'utilisation de la gestion des comptes. 54
Tableau 10: comparaison des caméras AXIS 59
Tableau 11: caractéristiques du serveur de détection et reconnaissance d'objets.....59
Tableau 12: caractéristiques du serveur de base de données 60

Liste des acronymes

B

BBTT: Backpropagation Through Time

C

CNN : Convolutional Neural Networks

CNTK: Microsoft Cognitive Toolkit

G

GPU: Graphics Processing Unit

GRU: Gated Recurrent Unit

H

HTTP: Hypertext Transfer Protocol

I

IA: Intelligence Artificielle

ILSVRC: Imagenet Large Scale Visual Recognition Challenge

L

LSTM: Long Short Term Memory

M

ML: Machine Learning

MLP: Multiple Layer Perceptron

O

OpenCV: Open Source Computer Vision

P

Pixel: pixel element

R

ReLu: Rectified Linear Unit

R-CNN: Recurrent Convolutional Neural Networks

RNN: recurrent neural network

RTP: Real-Time Protocol

RTSP: Real Time Streaming Protocol

S

SGD: Stochastic Gradient Descent

SSD: Single Shot Detector

SVM: Support Vector Machine

U

UDP: User Datagram Protocol

UML: Unified Modeling Language

Y

YOLO: You Only Look Once

Introduction générale

Introduction générale

Au Sénégal, il ne passe pas un jour ou presque sans accidents mortels de la circulation. Malgré les nombreuses campagnes de sensibilisation sur les dangers de la route, le nombre d'accidents ne baisse pas. 400 décès pour l'année 2013, 43 pour l'année 2014 et plus de 500 décès pour l'année 2016. Le rapport 2016 de la gendarmerie nationale note une augmentation des accidents de la circulation avec « 503 personnes tuées et 5653 blessés à l'occasion d'accidents impliquant 844 véhicules légers, 405 véhicules poids lourds, 516 véhicules de transport en commun, 212 véhicules à deux routes et hippomobiles et 242 piétons ». En 2019, le nombre d'accident s'élèvent à plus de 600 décès. Et 45% des causes d'accidents sont du fait de l'imprudence du conducteur [1].

Certains de ces accidents pouvaient être évités si le conducteur avait su l'objet qui était devant la voiture 3 secondes avant.

La préoccupation majeure est de trouver comment réduire le nombre d'accidents dans le transport routier au Sénégal. La meilleure solution est la prudence et le respect du code de la route par les conducteurs et les piétons. Une solution qui va dans ce sens consiste à détecter et reconnaitre les objets en temps réel dans le trafic routier. Celle-ci repose sur la mise en place d'un système de détection et de reconnaissance d'objets sur une carte Raspberry est qui relié à une caméra embarquée placée sur la voiture. La caméra va envoyer les images capturées à une base de données MySQL, cette dernière va servir de stockage pour un autre système de détection et reconnaissance qui sera couplé avec une application web.

Notre objectif est de proposer un système de détection et de reconnaissance des objets rapide, flexible et portable en temps réel.

Les problèmes qui se posent dans cette situation sont :

- ✓ Comment imiter artificiellement cette faculté d'utilisation de la vision ?

- ✓ Peut-on détecter un objet en mouvement ? c'est-à-dire étiqueter chaque pixel d'une image suivant s'il correspond ou non à une région en mouvement dans une scène
- ✓ Comment reconnaitre un objet en mouvement dans une scène observer à temps réels et l'étiqueté par son nom ?
- ✓ Peut-on installer ce système de vision artificiel sur une nano carte, couplé avec une application web pour le transport routier?

Les objectifs de ce travail sont les suivants :

- ✓ La détection d'objets qui désigne la tâche de localisation des objets d'une catégorie donnée,
- ✓ Etablir un modèle de de reconnaissance d'objets en temps réels rapide dans le domaine du transport routier,
- ✓ La portabilité de ce système de vision et surtout sa flexibilité

Pour se faire, ce document sera divisé en quatre chapitres :

- ✓ Le chapitre I traite la reconnaissance des objets : dans ce chapitre nous allons expliquer la vision par ordinateur, de la reconnaissance des objets et des techniques existantes.
- ✓ Dans le chapitre II, nous présentons l'état de l'art sur le Deep Learning précisément les algorithmes de réseaux de neurones avec une étude comparative
- ✓ Dans le chapitre III, nous allons exposer la conception de notre solution en expliquant son fonctionnement, les outils et les techniques utilisées.
- ✓ Enfin la partie implémentation qui sera teste puis validation et avec des perspectives.

CHAPITRE 1 : la reconnaissance des objets

Introduction

A l'heure actuelle, les systèmes de vision sont de plus en plus répandus (webcam, caméra, …) et les caméras peuvent être installées partout dans notre quotidien. Elles sont utilisées pour réaliser de la vidéo surveillance (dans les magasins, rues ou aéroports), de l'aide à la conduite (aide ou guidage ou la détection d'obstacle), et bien d'autres applications encore…Pour l'être humain, voir est une tâche innée et nous ne mesurons souvent pas la difficulté pour obtenir les mêmes performances artificiellement. Malgré les avancées de la vision par ordinateur, les systèmes développés sont très loin d'égaler les performances de l'œil et du cerveau humains en rapidité. Il s'agit de mettre au point un système capable de détecter et reconnaitre les objets comme des piétons, des voitures, des tableaux, … à partir d'images provenant d'une simple caméra. Le problème est d'autant plus difficile que le système que le système de vision est embarqué dans un véhicule se déplaçant en milieu urbain : l'environnement visuel contient de nombreux éléments, est aussi à sujet de multiples variations, et de plus, le processus doit prendre place en temps réel pour éviter une éventuelle collision.

1. Définition

La reconnaissance d'objets est un terme général pour décrire un ensemble de tâches de vision par ordinateur associées qui impliquent l'identification d'objets dans les photographies numériques.

La classification d'image implique de prédire la classe d'un objet dans une image. La localisation d'objet fait référence à l'identification de l'emplacement d'un ou plusieurs objets dans une image et au dessin d'une boîte abondante autour de leur étendue. La détection d'objets combine ces deux tâches et localise et classe un ou plusieurs objets dans une image.

Lorsqu'un utilisateur ou un praticien fait référence à la « reconnaissance d'objet », ils signifient souvent « détection d'objet ».

Nous utiliserons le terme de reconnaissance d'objet au sens large pour englober à la fois la classification d'images (une tâche nécessitant un algorithme pour déterminer quelles

classes d'objets sont présentes dans l'image) ainsi que la détection d'objets (une tâche nécessitant un algorithme pour localiser tous les objets présents dans une image).

A ce titre nous pouvons distinguer ces trois tâches de vision par ordinateur :

- Classification de l'image : prédisez le type ou la classe d'un objet dans une image.
- Entrée : une image avec un sujet objet, telle qu'une photographie.
- Sortie : une étiquette de classe (par exemple un ou plusieurs entiers mappés à des étiquettes de classe).
- Localisation d'objets : localisez la présence d'objets dans une image et indiquez leur emplacement avec un cadre de sélection.
- Entrée : une image avec un ou plusieurs objets, comme une photographie.
- Sortie : une ou plusieurs boîtes englobantes (par exemple définies par un point, une largeur et une hauteur).
- Détection d'objets : localisez la présence d'objets avec un cadre de sélection et les types ou classes des objets localisés dans une image.
- Entrée : une image avec un ou plusieurs objets, comme une photographie
- Sorite : une ou plusieurs boîtes englobantes (par exemple définies par un point, une largeur et une hauteur) et une étiquette de classe pour chaque boîte englobante.

Une autre extension de cette répartition des tâches de vision par ordinateur est la segmentation d'objet, également appelée « segmentation d'instance d'objet » ou « segmentation sémantique », où les instances d'objets reconnus sont indiquées mettant en surbrillance les pixels spécifiques de l'objet au lieu d'une zone de délimitation grossière.

A partir de cette ventilation, nous pouvons voir que la reconnaissance d'objet se réfère à une suite de tâches de vision par ordinateur difficiles. [2]

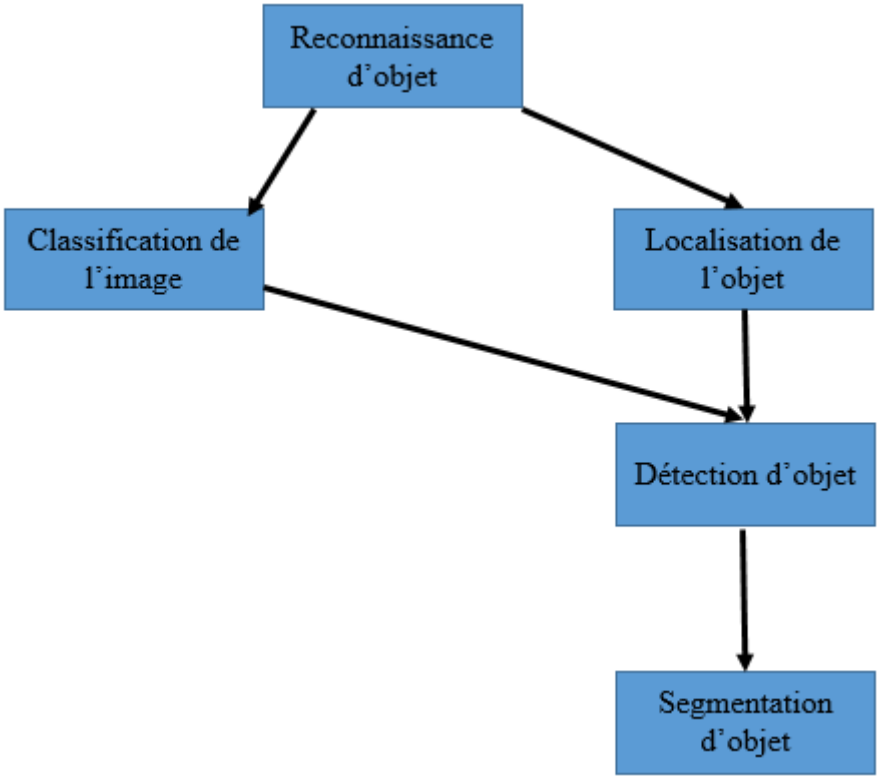


Figure 1 présentation des tâches de vision par ordinateur de reconnaissance d'objets.

Source de la figure 1: A gentle Introduction to Object Recognition With Deep Learning par Jason Browlee, Jason Brownlee on May 22, 2019 in Deep Learning for Computer Vision.

Cette figure explique les différentes de la reconnaissance d'un objet.

- Classification de l'image : les algorithmes produisent une liste des catégories d'objets présentes dans l'image.
- Localisation d'un seul objet : les algorithmes produisent une liste des catégories d'objets présentes dans l'image, ainsi qu'un cadre de délimitation aligné sur l'axe indiquant la position et l'échelle d'une instance de chaque catégorie d'objets.
- Détection d'objets : les algorithmes produisent une liste des catégories d'objets présentes dans l'image ainsi qu'un cadre de délimitation aligné sur l'axe indiquant la position et l'échelle de chaque instance de chaque catégorie d'objets.

2. Reconnaissance des formes

2.1. Introduction à l'Intelligence Artificielle

Le terme d'Intelligence Artificielle (IA) désigne toutes les techniques et méthodes qui permettent de doter des systèmes informatiques des capacités proches de celles de l'être humain. La science-fiction s'est approprié ce domaine en imaginant des robots, dotés d'une intelligence qui serait semblable à celle de l'homme. Toutefois, de nos jours,

maîtrise de l'IA est bien plus modeste, les systèmes développés ne sont capables que d'imiter certaines capacités de l'homme. Prenons par exemples des robots alliant des capacités de perception de l'environnement et de commande qui sont capables de se déplacer de façon autonome en évitant les obstacles (voir figure 1) [2].

Le but de ce mémoire est de réaliser un système de détecter des objets devant un véhicule à partir d'une caméra embarquée. Le travail se situe au niveau de la perception, effectivement, il s'agit de recréer artificiellement les capacités de perception d'un conducteur qui chercherait de façon active les objets (piétons, animaux, voitures, tableaux, etc…) situés devant son véhicule avec lesquels il pourrait entrer en collision. Le système désiré est donc apparenté à un système d'Intelligence Artificielle.



Figure 2a Intelligence Artificielle



Figure 2b : de la fiction à la réalité

a) Exemple d'un robot imaginé par la science-fiction (extrait du film I-Robots) ayant les mêmes capacités que l'être humain,
b) Exemple des capacités d'un robot (NAO de la société Aldebaran Robotics).

2.2. L'intelligence artificielle à la reconnaissance des formes

Mais l'IA est un très vaste domaine et les applications sont diverses et variées. Il est possible de distinguer plusieurs branches, les travaux de ce mémoire s'inscrivent dans le cadre de la « reconnaissance des formes ». Cette appellation désigne les méthodes qui, à partir de l'observation d'un objet, lui attribuent une classe. Par exemple, un système qui, à partir d'une observation radar, classifie des objets volants soit dans une

classe avion, soit dans une classe non-avion, réalise une tâche de reconnaissance d'objets.

L'observation d'un objet peut-être très différente selon le domaine d'activité dans lequel nous évoluons. Souvent c'est une observation visuelle qui est utilisée (photo, séquence vidéo, image médicale, image satellite, etc.), mais il est également possible d'avoir une analyse radar, laser ou sonore, le champ d'application étant très large. De même, le motif à reconnaitre et à classer peut-être simple ou complexe, et les outils développés sont adaptés à chaque utilisation. Par exemple, un cube sera plus facile à reconnaitre qu'un visage puisqu'il a des propriétés physiques constantes dans le temps et l'espace alors qu'un visage est sujet à beaucoup plus de changements.

Dans ce mémoire, nous cherchons à reconnaitre les objets (piétons, véhicules, arbres, panneaux de signalisation, etc.) à partir d'images vidéo. Grâce à une caméra embarquée, une observation visuelle de la scène à l'avant d'un véhicule est obtenue ; la problématique principale est alors de reconnaitre les objets présents dans les images, c'est-à-dire d'être capable d'indiquer si les objets présents dans l'environnement devant le véhicule appartiennent à telle classe.

Dans ce rapport, nous présentons les différentes étapes permettant de réaliser ce système de détection et de reconnaissance des objets. Actuellement, il existe différentes méthodes, les plus utilisées sont basées sur le modèle et l'apprentissage. Les premières utilisent un modèle d'objet et identifient un nouvel objet en calculant sa distance au modèle [3]. Les secondes font appel à une phase d'apprentissage hors ligne qui génère un classificateur capable de différencier un objet (par exemple une personne) par rapport à un objet. Dans notre travail nous avons utilisé des techniques d'apprentissage du deep Learning.

a. Segmentation d'images

Nous avons vu précédemment que la première étape d'une reconnaissance consiste à réaliser une segmentation de l'image. Cette étape n'intervient généralement pas lors de l'apprentissage hors ligne lorsqu'une base d'apprentissage comprenant déjà des images « prédécoupées » autour de l'objet à reconnaitre est utilisée. Par contre, lors de la détection et reconnaissance des objets en ligne, les données proviennent d'une caméra

dont le champ dont le champ couvre une largue scène et l'objet recherché n'en occupe qu'une partie. Afin de pouvoir détecter puis reconnaître cet objet, il est nécessaire de diviser l'image initiale en une ou plusieurs zones d'intérêt où l'objet est susceptible de se trouver. Cette étape de prétraitement de l'image permet ainsi de limiter les temps de calcul qui suivent puisque seules les zones conservées seront traitées.

Selon les capacités du descripteur qui est ensuite mis en œuvre dans la chaîne de traitement, ces zones pourront être découpées à nouveau en une ou plusieurs petites images centrées autour de l'objet et mises aux bonnes dimensions afin de mettre en adéquation les données courantes avec les besoins du classificateur. Celui-ci pourra alors attribuer chacune d'entre elles à la classe correspondante.

Tableau 1: les différentes étapes constituants un système de détection d'objets

	1. Traitement hors ligne Apprentissage	2. Traitement en ligne Détection et reconnaissance
Entrées :	Un ensemble d'images avec leurs étiquettes de classe : positif ou négatif	Une nouvelle image
Segmentation :	Segmentation des images en vignettes autour des exemples	Découpage de l'image initiale en des zones « intéressantes », c'est-à-dire des petites images où l'objet recherché est susceptible de se trouver
Description :	Analyse via un descripteur adéquat de chaque exemple de la base d'apprentissage pour en extraire des informations	Extraction des informations sur chaque petite image en utilisant le même descripteur que pendant la phase d'apprentissage
Classification :	Traitement des informations fournies précédemment par le descripteur avec les étiquettes de classe de créer un classificateur qui sera capable de distinguer les exemples positifs des négatifs	Utilisation du classificateur entraîné pendant l'apprentissage pour que le système indique si l'objet recherché est présent ou non dans chaque petite image.

En premier, l'apprentissage hors ligne et en second, la détection et reconnaissance en ligne.

Pour sélectionner ces zones, deux types de segmentation peuvent être mises en œuvre. Dans un premier temps, si des connaissances a priori sur la scène sont disponibles, elles peuvent être utilisées ; par exemple, en vidéosurveillance, la recherche du visage d'une personne peut être ciblée sur la porte d'entrée d'un magasin, à une hauteur adéquate. Mais des informations sur la scène ne sont pas toujours connues, et même parfois, cela n'est pas suffisant voire inadaptable, notamment lorsqu'il s'agit d'une caméra mobile.

Dans un deuxième temps, si aucune information n'est connue, des méthodes particulières peuvent être employées pour segmenter les images, celles-ci regroupent par zones des pixels ayant des propriétés communes (citons par exemple les approches par seuillage, par frontières ou par région) sachant qu'il est possible de fusionner plusieurs techniques pour segmenter au mieux l'image. Tout cela dépend de l'application et du temps imparti pour la réalisation.

b. Description d'images

Lorsque des zones de l'image ont été définies et sélectionnées, il faut analyser l'information qu'elles contiennent. Deux types de traitement sont envisageables.

Le premier type consiste en une amélioration de la qualité de l'image, par exemple pour atténuer les défauts du capteur et les bruits associés aux conditions d'acquisition (flou, mauvaise illumination…). Ces traitements sont parfois réalisés avant la phase de segmentation de l'image mais peuvent l'être après de façon à avoir moins de zones à s'occuper et ainsi réduire les temps de calcul.

Le deuxième type est directement lié à la tâche de reconnaissance d'objets. Il faut extraire des zones d'intérêt des informations relatives à ce qui est contenu dans l'image, il existe des descripteurs d'images qui permettent de caractériser l'information disponible. Deux façons de procéder sont envisageables :

- Soit l'image est décrite en des points particuliers et significatifs : c'est un descripteur local et ces points intéressants sont des points d'intérêt ;

Soit tous les pixels de l'image correspondants à la zone d'intérêt sont pris en compte dans la description : c'est un descripteur global.

Descripteur local : en ce qui concerne les descripteurs locaux, des méthodes permettent de détecter les points d'intérêt. Par exemple, le détecteur de points de Harris [3] qui est parmi les plus répandu ; son principe est de détecter les changements brusques d'intensité sur l'image pour ainsi mettre en évidence des coins.



Figure 3 : exemple de détecteur de points d'intérêt : le détecteur de Harris.

Source https://quizlet.com/96002313/test-foe-da-seniors-flash-cards/

Le détecteur de Harris permet de sélectionner dans l'image initiale (à gauche) un ensemble de points d'intérêt (en rouge dans l'image à droite) qui correspondent à des changements brusques de l'intensité dans 2 directions.

Puis un descripteur caractérise chaque point d'intérêt par son voisinage. Le plus souvent, il s'agit de l'information qu'a calculée le détecteur.

L'utilisation de points d'intérêt est difficile à mettre en place dans le cas d'un système de reconnaissance avec caméra embarquée. En effet, le piéton est un objet déformable et de plus en mouvement ; les points d'intérêt risquent de ne pas être les mêmes d'une image à l'autre (par exemple en fonction de la posture de la personne ou des conditions d'illumination) et le classificateur qui suit sera incapable de faire le lien avec les classes d'objets envisagées. Néanmoins, il existe des travaux de catégorisation [4] utilisant des points d'intérêt.

Descripteur global en reconnaissance d'objets, un descripteur global est plus facile à utiliser car il traite l'image dans son intégralité. Le descripteur est ainsi moins sensible aux déformations d'un objet d'une image à l'autre. Deux images proches doivent donc conduire à deux descripteurs proches. Un classificateur peut ainsi être entraîné sur ces

données pendant l'apprentissage et il pourra associer ces deux images à une classe grâce à une notion de distance entre les descripteurs (par exemple distance euclidienne). Il en sera de même pour la reconnaissance : en utilisant le même descripteur que pendant l'apprentissage, le classificateur pourra associer un nouvel objet à une classe d'objets apprise.

De nombreuses méthodes existent, le choix dépend avant tout de l'application visée et du temps de calcul disponible pour la réaliser. En effet, dans le cas de reconnaissance et de détection en temps réel, il faut souvent faire un compromis entre qualité de la réponse et temps d'exécution. Or si les informations fournies par le descripteur ne sont pas assez précise, le classificateur sera difficile à entrainer pendant l'apprentissage et les résultats qu'il fournira en reconnaissance d'objets seront donc peu fiables.

c. Deep Learning

Le Deep Learning est un nouveau domaine de recherche du Machine Learning, qui a été introduit dans le but de rapprocher le Machine Learning de son objectif principal : l'Intelligence Artificielle. Il concerne les algorithmes inspirés par la structure et le fonctionnement du cerveau. Ils peuvent apprendre plusieurs niveaux de représentation dans le but de modéliser des relations complexes entre les domaines.

Le deep Learning est basé sur l'idée des réseaux de neurones artificiels et il est taillé pour gérer de larges quantités de données en ajoutant des couches au réseau. Un modèle de deep Learning a la capacité d'extraire des caractéristiques à partir des données brutes grâce aux multiples couches de traitement composé de multiples transformations linéaires et non linéaires et apprendre sur ces caractéristiques petit à petit à travers chaque couche avec une intervention humaine minime. [5] [6] [7] [8] [9] [10][11].

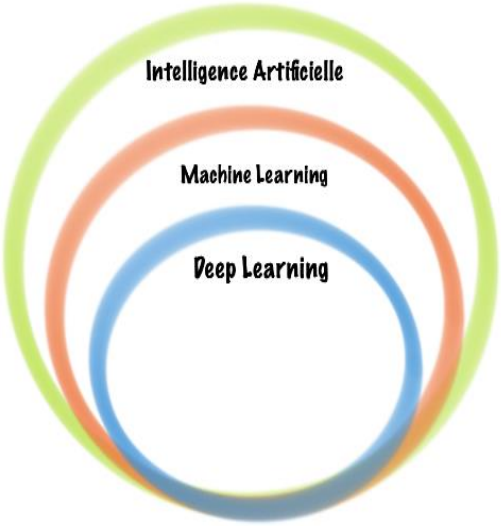


Figure 4 : la relation entre l'intelligence artificielle, le machine Learning et le deep Learning

La figure ci-dessus permet d'expliquer les relations qui existent entre l'IA, le ML et le DL. Extrait de L'Intelligence Artificielle et le Deep Learning — Introduction publié sur le site https://medium.com/un-regard-sur-lintelligence-artificielle/l-i-a-et-le-deep-learning-introduction-8e77d583186e

De 2012 à 2017, le deep Learning est passé d'un marché de niche ou seulement une poignée de chercheurs s'y intéressait au domaine le plus prisé par les chercheurs. Les recherches en relation avec le deep Learning apparaissent maintenant dans les tops journaux comme Science [12], Nature [13] et Nature Methods [14] pour ne citer que quelques-uns. Le deep Learning a coquerie le GO [15], appris à conduire une voiture [16], diagnostiquer le cancer [17] et l'autisme [18] et même devenu un artiste [19].

Le terme « Deep Learning » a été introduit pour la première fois au ML par Dechter (1986) [20], et aux réseaux de neuronaux artificiels par Aizenberg et al (2000) [21].

i. Histoire du deep Learning

Le concept d'apprentissage profond prend forme dans les années 2010, avec la convergence de quatre facteurs :

- Des réseaux de neurones artificiels multicouches (eux-mêmes issus entre autres du concept de perceptron, datant de la fin des années 1950) ;
- Des algorithmes d'analyse discriminante et apprenants (dont l'émergence remonte aux années 1980) ;

- Des machines dont la puissance de traitement permet de traiter des données massives ;
- Des bases de données suffisamment grandes, capables d'entraîner des systèmes de grandes tailles.

En octobre 2015, le programme AlphaGO, à qui l'on a « appris » à jouer au jeu de Go grâce à la méthode de l'apprentissage profond, bat le champion européen Fan Hui par 5 parties à 0. En mars 2016, le même programme bat le champion du monde Lee Sedol par 4 parties à 1.

ii. Pourquoi le deep Learning ?

Les algorithmes de deep Learning fonctionnent bien pour une grande variété de problèmes. Cependant ils ont échoués à résoudre quelques problèmes majeurs de l'IA telle que la reconnaissance vocale et la reconnaissance d'objets.

Le développement du deep Learning fut motivé en partie par l'échec des algorithmes traditionnels dans de telle tâche de l'IA.

Mais ce n'est qu'après que de plus grandes quantités de données ne soit disponibles grâce notamment au Big Data et aux objets connectés et que les machines de calcul soient devenues plus puissantes qu'on a pu comprendre le potentiel réel du Deep Learning.

Une des grandes différences entre le Deep Learning et les algorithmes de ML traditionnelles c'est qu'il s'adapte bien, plus la quantité de données fournie est grande plus les performances d'un algorithme de Deep Learning sont meilleurs. Contrairement à plusieurs algorithmes de ML classiques qui possèdent une borne supérieure à la quantité de données qu'ils peuvent recevoir des fois appelée "plateau de performance", les modèles de Deep Learning n'ont pas de telles limitations (théoriquement) et ils sont même allés jusqu'à dépasser la performance humaine dans des domaines comme l'image processing.

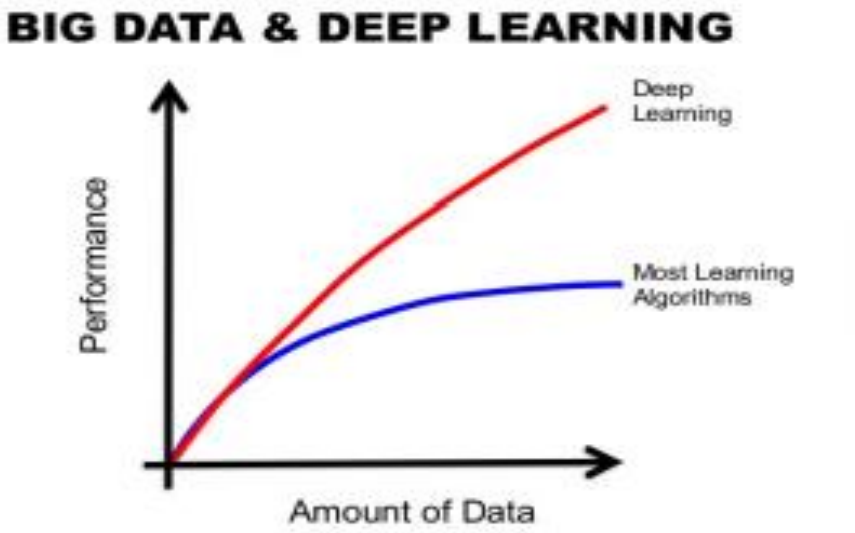


Figure 5 la différence de performance entre le Deep Learning et la plupart des algorithmes de ML en fonction de la quantité de données [23].

Source mémoire de master (spécialité Modèle intelligent et décision) Deep Learning pour la classification des images page 30.

Autre différence entre les algorithmes de ML traditionnels et les algorithmes du Deep Learning c'est l'étape de l'extraction de caractéristiques. Dans les algorithmes de ML traditionnels l'extraction de caractéristiques et faite manuellement c'est une étape difficile et coûteuse en temps et requiert un spécialiste en la matière alors qu'en Deep Learning cette étape est exécutée automatiquement par l'algorithme [24].

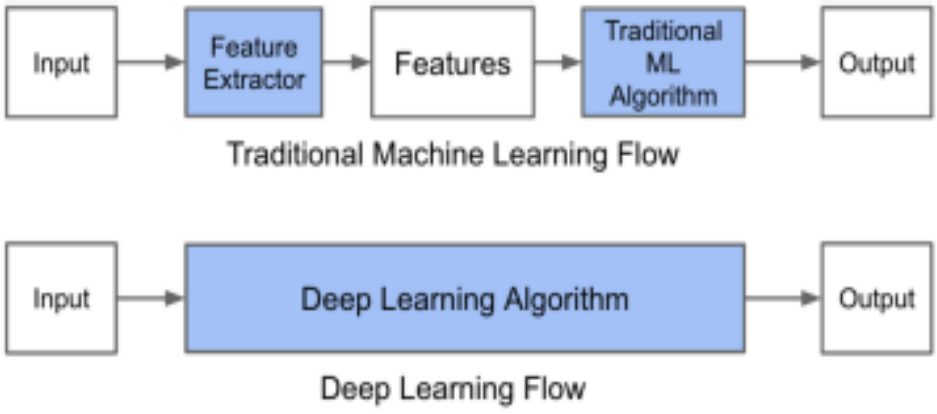


Figure 6 le procédé du ML classique comparé à celui Deep Learning.

Source mémoire de master (spécialité Modèle intelligent et décision) Deep Learning pour la classification des images page 30.

3. Quelques projets de reconnaissance et de détection d'objets

Depuis quelques années, de nombreux chercheurs travaillent sur des projets concernant l'aide à la conduite automobile par vision artificielle. Des projets comme par exemple « la route automatisée » ou « PATH » ont pour but le développement de systèmes capables de réguler la vitesse de véhicules routiers (contrôle longitudinal) ou de contrôler la position des véhicules par rapport aux voies de circulation (contrôle latéral). Le LASMEA est impliqué dans de tels projets et des travaux sont engagés afin de mettre en œuvre, sur notre véhicule expérimental (VELAC 2), un dispositif de type régulateur de vitesse adaptatif (ou ACC : Adaptative Cruise Control). Ce type de système a pour but de réguler la vitesse du véhicule afin de respecter les distances de sécurité avec les véhicules situés à l'avant [59].

La plupart des projets qui ont un grand succès dans ce domaine sont basés sur le Deep Learning, comme le projet PIXALARM. Ce projet permet de faire :

- ✓ La recherche de présence : que ce soit pour des personnes, des véhicules ou d'autres types d'objets, la recherche d'objet est capable de repérer distinctement ce qui est présent dans la scène puis indiquer leur nombre. La vidéo peut être divisée en zones d'intérêt qui posséderont chacune leur compteur indépendant. Ces informations peuvent par la suite être exploitées dans les scénarios ou être exportées pour générer des statistiques. Elle peut servir à la détection d'intrusion, compter le nombre de personne dans une file d'attente ou compter le nombre de véhicule dans un parking.
- ✓ Le compteur de passage: dans ce mode, à l'aide de lignes de franchissement positionnées sur l'image, Pixalarm permet de compter les objets souhaités (humains, véhicules, etc). La ligne peut également être directionnelle afin de compter que ce qui passe dans un sens précis. Pixalarm propose une interface graphique dédiée au comptage de passage qui permet de visualiser en temps réel l'évolution du compteur. Avec le compteur de passage vous pouvez aussi faire des analyses de trafic routier, un compteur de visite ou un comptage de foule.
- ✓ Des analyses spécifiques : pour des besoins particuliers ce projet de Deep Learning peut être étendu comme le comptage d'animaux, la détection d'objets abandonné ou dans la domotique [60].

Conclusion

Dans ce chapitre nous avons fait la présentation sur la détection et reconnaissance des objets dans l'image avec des techniques existantes, puis avons abordé l'intelligence artificielle et le Deep Learning. Dans la suite de ce chapitre nous avons fait l'état de l'art sur les algorithmes utilisés par le Deep Learning à savoir les réseaux de neurones artificiels pour la détection et reconnaissance des objets dans l'image.

CHAPITRE2 : Etat de l’art sur les algorithmes du Deep Learning

Introduction

Le Deep Learning a de nombreuses utilités. C'est cette technologie qui est utilisée pour la reconnaissance faciale de Facebook par exemple, afin d'identifier automatiquement vos amis sur les photos. C'est également cette technologie qui permet à la reconnaissance faciale Face Id dans les smart phones de s'améliorer au fil du temps. Parmi les techniques du Deep Learning, on compte les réseaux de neurones artificiels. C'est sur ces algorithmes que repose le Deep Learning. Les réseaux de neurones artificiels sont inspirés par les neurones du cerveau humain. Ils sont constitués de plusieurs neurones artificiels connectés entre eux. Plus le nombre de neurones est élevé, plus le réseau est « profond ».

1. Fonctionnement d'un réseau de neurones



Figure 7 : description du fonctionnement d'un neurone

Source : publication Deep Learning ou apprentissage profond, définition, concept par Bastien L, le 18 juillet 2018.

Au sein du cerveau humain, chaque neurone reçoit environ 100 000 signaux électriques des autres neurones. Chaque neurone en activité peut produire un effet excitant ou inhibiteur sur ceux auxquels il est connecté. Au sein d'un réseau artificiel, le principe est similaire. Les signaux voyagent entre les neurones. Toutefois, au lieu d'un signal électrique, le réseau de neurones assigne un certain poids à différents neurones. Un neurone qui reçoit plus de charge exercera plus d'effet sur les neurones adjacents. La couche finale de neurones émet une réponse à ces signaux.

Pour comprendre comment fonctionne le Deep Learning, prenons un exemple concret de reconnaissance d'images. Imaginons que le réseau de neurones soit utilisé pour reconnaître les photos qui comportent au moins un chat. Pour pouvoir identifier les chats sur les photos, l'algorithme doit être en mesure de distinguer les différents types de chats, et de reconnaître un chat de manière précise quel que soit l'angle sous lequel il est photographié.

Afin d'y parvenir, le réseau de neurones doit être entraîné. Pour se faire, il est nécessaire de compiler un ensemble d'images d'entraînement pour pratiquer le deep Learning. Cet ensemble va regrouper des milliers de photos de chats différents, mélangés avec des images qui ne sont pas des chats. Ces images sont ensuite converties en données et transférées sur le réseau. Les neurones artificiels assignent ensuite un poids aux différents éléments. La couche finale de neurones va alors rassembler les différentes informations pour déduire s'il s'agit ou non d'un chat.

Le réseau de neurones va ensuite comparer cette réponse aux bonnes réponses indiquées par les humains. Si les réponses correspondent, le réseau garde cette réussite en mémoire et s'en servira plus tard pour reconnaître les chats. Dans le cas contraire, le réseau prend note de son erreur et ajuste le poids placé sur les différents neurones pour corriger son erreur. Le processus est répété des milliers de fois jusqu'à ce que le réseau soit capable de reconnaître un chat sur une photo dans toutes les circonstances [25].

2. Les différents types de modèles

Il existe un grand nombre de variantes d'architecture profondes. La plupart d'entre elles sont dérivées de certaines architectures parentales originales. Il n'est pas toujours possible de comparer les performances de toutes les architectures, car elles ne sont pas toutes évaluées sur les mêmes ensembles de données. Le Deep Learning est un domaine à croissance rapide, et de nouvelles architectures, variantes ou algorithmes apparaissent toutes les semaines.

2.1. Les réseaux de neurones convolutifs

Convolutional Neural Network (CNN) (réseaux de neurones convolutifs) sont un type de réseau de neurones spécialisés pour le traitement de données ayant une topologie semblable à une grille. Les exemples comprennent des données de type série temporelle, qui peuvent être considérées comme une grille 1D en prenant des échantillons à des intervalles de temps réguliers et des données de type image, qui peuvent être considérées comme une grille 2D de pixels. Les réseaux convolutifs ont connu un succès considérable dans les applications pratiques. Le nom « réseau de neurones convolutif » indique que le réseau emploie une opération mathématique appelée convolution. La convolution est une opération linéaire spéciale. Les réseaux convolutifs sont simplement des réseaux de neurones qui utilisent la convolution à la place de la multiplication matricielle dans au moins une de leurs couches.

Ils ont de larges applications dans la reconnaissance de l'image et de la vidéo, les systèmes de recommandations [26] et le traitement du langage naturel [27].

2.1.1. Inspiration

Dans le ML, un réseau convolutif est un type de réseau de neurones feed-forward, il a été inspiré par des processus biologiques [28]. Le modèle de connectivité entre les neurones d'un CNN s'inspire de l'organisation du cortex visuel animal.

Les neurones corticaux individuels répondent aux stimuli dans une région restreinte de l'espace connu sous le nom de champ réceptif. Les champs réceptifs des différents neurones se chevauchent partiellement de sorte qu'ils couvrent le champ visuel.

La réponse d'un neurone individuel aux stimuli dans son champ réceptif peut être approchée mathématiquement par une opération de convolution.

2.1.2.L'opération de convolution

Dans sa forme la plus générale, la convolution est une opération sur deux fonctions d'argument réel. Pour comprendre la motivation derrière la convolution, nous commençons par des exemples de deux fonctions qu'on pourrait utiliser.

Supposons que nous suivons l'emplacement d'un vaisseau spatial avec un capteur laser. Notre capteur laser fournit une sortie *x(t)* qui est la position du vaisseau spatial au moment t. x et t sont des réelles, c'est-à-dire que nous pouvons obtenir une lecture différente du capteur laser à tout moment.

Supposons maintenant que notre capteur laser soit quelque peu bruité. Pour obtenir une estimation moins bruitée de la position du vaisseau spatial, nous aimerions combiner plusieurs mesures. Bien sûr, les mesures plus récentes sont plus pertinentes, donc nous voulons que ces mesures soient une moyenne pondérée et donner plus de poids aux mesures récentes. Nous pouvons le faire avec une fonction de pondération w(a), où a est l'âge de la mesure.

Si nous appliquons une telle opération de moyenne pondérée à chaque instant, nous obtenons une nouvelle fonction qui fournit une estimation lisse de la position du vaisseau spatial :

$$S(t)= \int x(a)w(t-a)da \qquad (1)$$

Cette opération s'appelle convolution. L'opération de convolution est généralement désignée par un astérisque :

$$S(t) = (x+w)t \qquad (2)$$

Dans notre exemple, l'idée d'un capteur laser qui peut fournir des mesures à chaque instant dans le temps n'est pas réaliste. Habituellement, lorsque nous travaillons avec des données sur l'ordinateur, le temps sera discrétisé (numérisé), et notre capteur fournira des données à des intervalles réguliers. Dans notre exemple, il est plus réaliste de supposer que notre laser fournit une mesure une fois par seconde. L'index de temps t ne peut alors prendre que des valeurs entières. Si on suppose maintenant que x et w sont des entiers, on peut définir la convolution discrète :

$$S(t) = (x+w)t = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \qquad (3)$$

Enfin, nous utilisons souvent des convolutions sur plus d'un axe à la fois. Figure « 8 ».

Dans la terminologie du réseau convolutif, le premier argument (dans cet exemple, la fonction *x*) de la convolution est souvent appelé l'entrée (input) et le second argument (dans cet exemple, la fonction *w*) comme noyau (kernel). La sortie est parfois appelée feature map.

2.1.3.La couche convolutif

La convolution s'appuie sur trois idées importantes qui peuvent aider à améliorer un système de ML : sparse interactions, parameter sharing et équivariant représentations.

- Sparse interactions : Les réseaux de neurones traditionnels utilisent la multiplication matricielle par une matrice de paramètres avec des paramètres séparés décrivant l'interaction entre chaque unité d'entrée et chaque unité de sortie. Cela signifie que chaque unité de sortie interagit avec chaque unité d'entrée ce qui n'est pas le cas des réseaux de neurones convolutifs. Ceci est accompli en rendant le noyau plus petit que l'entrée. Par exemple, lors du traitement d'une image, l'image d'entrée peut avoir des milliers ou des millions de pixels, mais nous pouvons détecter de petites caractéristiques significatives telles que les bords avec des noyaux qui n'occupent que des dizaines ou des centaines de pixels. Cela signifie que nous pouvons stocker moins de paramètres, ce qui réduit les besoins en matière de mémoire du modèle et améliore son efficacité. Cela signifie également que le calcul des sorties nécessite moins d'opérations. Ces améliorations en matière d'efficacité sont généralement assez importantes. Figure 9.

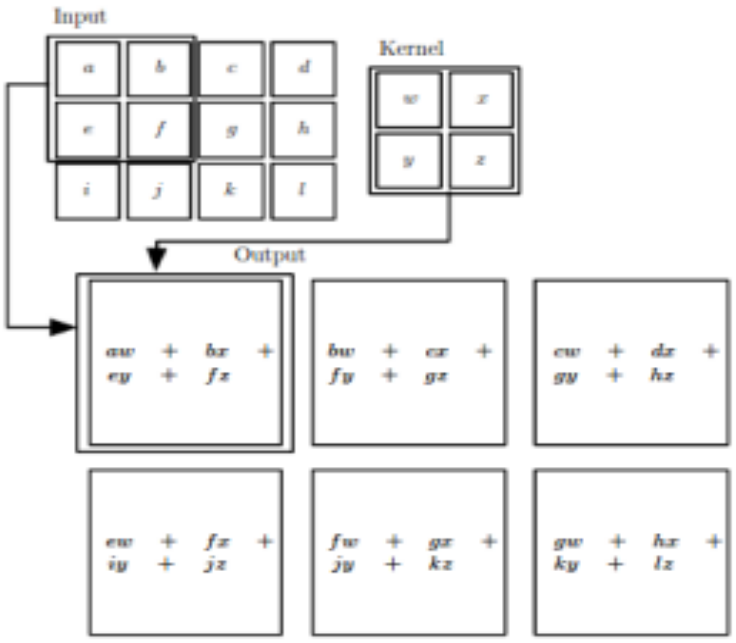


Figure 8 : exemple d'une convolution 2D [29].

Source de la figure 8 mémoire de master (spécialité Modèle intelligent et décision) Deep Learning pour la classification des images page 33.

- parameter sharing : (le partage des paramètres) Se réfère à l'utilisation du même paramètre pour plus d'une fonction dans un modèle. Dans un réseau de neurones convolutif, chaque élément du noyau est utilisé à chaque position de l'entrée (sauf peut-être quelques-uns des pixels des bords, selon le choix de conception concernant la frontière). Le parameter sharing utilisé par l'opération de convolution signifie que, plutôt que d'apprendre un ensemble de paramètres distincts pour chaque emplacement, nous n'apprenons qu'un ensemble ce qui réduit encore davantage les exigences de stockage du modèle. Figure 10.
- équivariant représentations : dans le cas de la convolution, la particularité du parameter sharing fait que la couche possède une propriété appelée équivariant représentations. Pour dire qu'une fonction est équivariante signifie que si l'entrée change, la sortie change de la même manière. Spécifiquement, une fonction f(x) est équivariante à la fonction g si f(g(x)) = g(f(x)). Dans le cas du traitement d'image, la convolution crée une carte 2-D de certaines caractéristiques apparaissant dans l'entrée. Si nous déplaçons l'objet dans l'entrée, sa représentation va se déplacer de la même façon dans la sortie. Par exemple, dans le cas d'une image, Il est utile de détecter les arêtes dans la première couche d'un réseau convolutif. Les mêmes arêtes apparaissent plus ou moins partout dans l'image, donc il est pratique de partager des paramètres sur l'image entière.

2.1.4. La couche de pooling

Une architecture atypique d'un réseau convolutif se compose de trois types de couches différentes. D'abord une couche convolutif pour générer un ensemble d'activations linéaires ensuite, on le fait passer à travers une couche d'activation non linéaire telle que Rectified Linear Unit (ReLu), enfin on utilise la fonction pooling. [30].

- Il permet de réduire progressivement la taille des représentations afin de réduire la quantité de paramètres et de calcul dans le réseau et, par conséquent, de contrôler également le sur-apprentissage.
- Il permet l'invariance aux petites translations
- Utile lorsqu'on préfère savoir si une caractéristique est présente plutôt que la région de sa présence.
- Plusieurs types de pooling différent (MAX pooling (très populaire), AVG pooling, …)

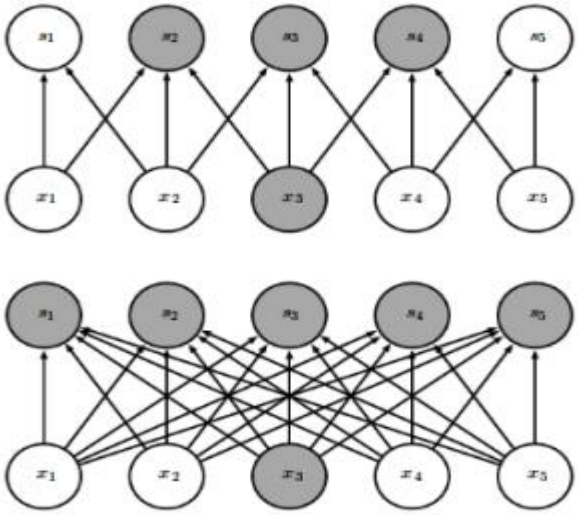


Figure 9: Interaction du Sparse

Sur la figure ci-dessus, on met en évidence l'unité d'entrée, x 3, et les unités de sortie qui sont affectées par cette unité (en haut). Lorsque ***s*** est formé par convolution avec un noyau de largeur 3, seules trois sorties sont affectées par x. (en bas) Lorsque ***s*** est formé par une multiplication matricielle, toutes les sorties sont atteintes par *x* 3. [12]

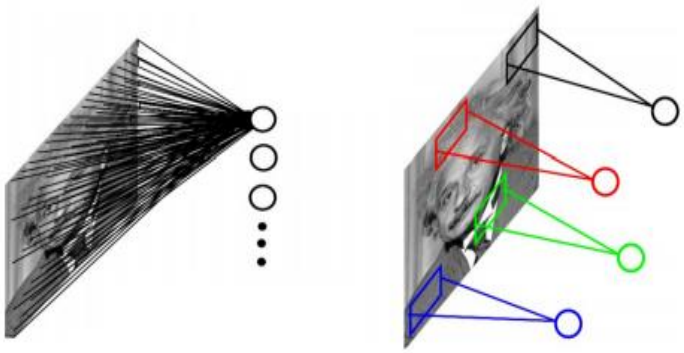


Figure 10: exemple d'une image 1000X1000.

Source des figures 9 et 10 : mémoire de master (spécialité Modèle intelligent et décision) Deep Learning pour la classification des images page 34.

Le non parameter sharing nous oblige à concevoir une cachée de 10^6 pixels, en tout ça fait 10^12 paramètres. Avec un noyau de 10X10 et une couche cachée de 10^6 neurones, le nombre de paramètres est de 10^8.

2.1.5. Le perceptron

Après avoir extrait les caractéristiques des entrées, on attache à la fin du réseau un perceptron ou bien un MLP. Le perceptron prend comme entrée les caractéristiques extraites et produit un vecteur de N dimensions ou N est le nombre de classe ou chaque élément est la probabilité d'appartenance à une classe. Chaque probabilité est calculée à l'aide de la fonction softmax dans le cas où les classes sont exclusivement mutuelles.

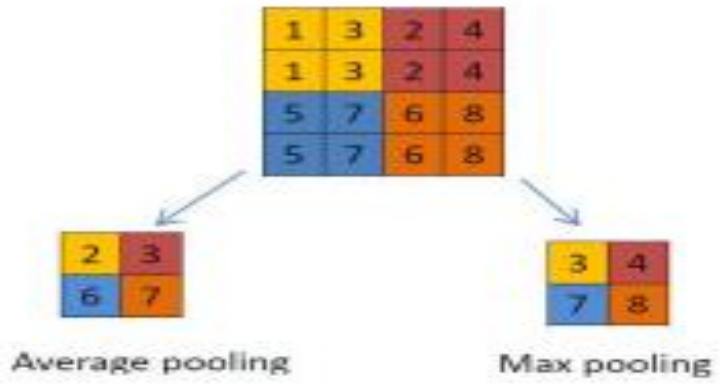


Figure 11: Average pooling avec des cases

Chaque case correspond à la moyenne du carré d'entrée de la même couleur, exemple de la case jaune : (1+3+1+3)/4=2. Max pooling : chaque case correspond à la valeur maximale du carré d'entrée de la même couleur, exemple de la case bleu :

max (5, 7, 5,7)=7.

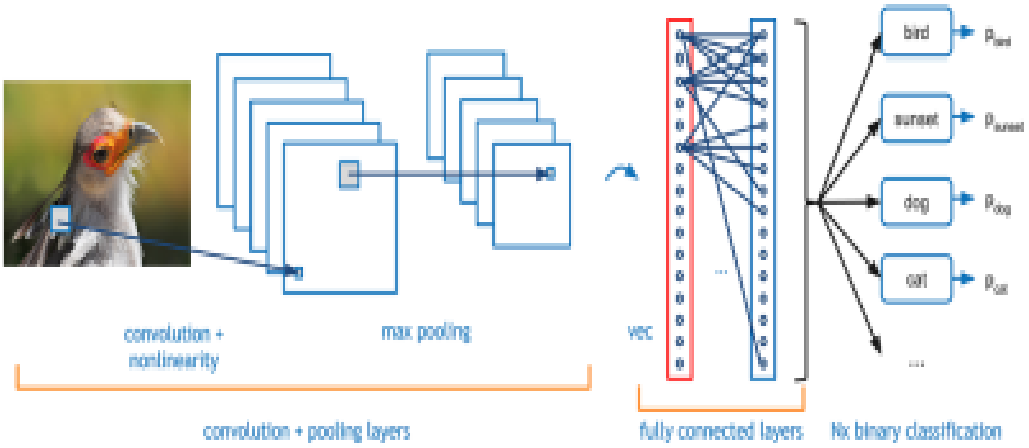


Figure 12: Entrée d'une image 2D dans un réseau convolutif, composé d'une couche convolutif, d'une fonction non linéaire, d'une couche MAX pooling et enfin un perceptron multi couche.

Source des figures 11 et 12: mémoire de master (spécialité Modèle intelligent et décision) Deep Learning pour la classification des images page 35.

2.1.6. Quelques réseaux convolutifs célèbres

- **LeNet** [30] : les premières applications réussies des réseaux convolutifs ont été développées par Yann LeCun dans les années 1990. Parmi ceux-ci, le plus connu est l'architecture LeNet utilisée pour lire les codes postaux, les chiffres, etc.
- **AlexNet** [31] : Le premier travail qui a popularisé les réseaux convolutifs dans la vision par ordinateur était AlexNet, développé par Alex Krizhevsky, Ilya Sutskever et Geoff Hinton. AlexNet a été soumis au défi ImageNet ILSVRC [32] en 2012 et a nettement surpassé ses concurrents. Le réseau avait une architecture très similaire à LeNet, mais était plus profond, plus grand et comportait des couches convolutifs empilées les unes sur les autres (auparavant, il était commun de ne disposer que d'une seule couche convolutif toujours immédiatement suivie d'une couche de pooling).
- **ZFNet** [33] : le vainqueur de ILSVRC challenge 2013 était un réseau convolutif de Matthew Zeiler et Rob Fergus. Il est devenu ZFNet (abréviation de Zeiler et Fergus Net). C'était une amélioration d'AlexNet en ajustant les hyper-paramètres de l'architecture, en particulier en élargissant la taille des couches convolutifs et en réduisant la taille du noyau sur la première couche.
- **GoogLeNet [34] :** le vainqueur de ILSVRC challenge 2014 était un réseau convolutif de Szegedy et al. De Google, sa principale contribution a été le développement d'un module *inception* qui a considérablement réduit le nombre de paramètres dans le réseau (4M, par rapport à AlexNet avec 60M). En outre, ce module utilise le **global AVG pooling** au lieu du PMC à la fin du réseau, ce qui élimine une grande quantité de paramètres. Il existe également plusieurs versions de GoogLeNet, parmi elles, Inception-v4 [36].
- **ResNet** [37] : Residual network développé par Kaiming He et al. A été le vainqueur de ILSVRC 2015. Il présente des sauts de connexion et une forte utilisation de la batch normalisation. Il utilise aussi le **global AVG pooling** au lieu de PMC à la fin.

2.2. Réseaux de neurones récurrents

Les humains ne commencent pas leurs pensées à zéro à chaque seconde. Lorsqu'on lit un livre, on comprend chaque mot en fonction de la compréhension des mots précédents. On n'oublie pas tout et recommençons à réfléchir à nouveau. Nos pensées ont une persistance.

Les réseaux de neurones traditionnels ne peuvent pas le faire, et cela est un inconvénient majeur. Par exemple, imaginons qu'on souhaite classer quel genre d'évènement se produit à chaque étape du film. On ne sait pas très bien comment un réseau de neurones traditionnels pourrait utiliser son raisonnement sur les évènements précédents dans le film pour en informer les derniers.

RNN (les réseaux de neurones récurrents) traitent ce problème. Ce sont des réseaux avec des boucles, permettant aux informations de persister.

2.2.1. C’est quoi un RNN ?

L’idée derrière les RNN est d’utiliser des informations séquentielles. Dans un réseau neuronal traditionnel, nous supposons que toutes les entrées (et les sorties) sont indépendantes les unes des autres. Mais pour de nombreuses tâches, c’est une très mauvaise idée. Si on veut prédire le prochain mot dans une phrase, il faut connaître les mots qui sont venus avant. Les RNN sont appelés récurrents car ils exécutent la même tâche pour chaque élément d’une séquence, la sortie étant dépendante des calculs précédents. Une autre façon de penser les RNN est qu’ils ont une « mémoire » qui capture l’information sur ce qui a été calculé jusqu’ici.

En théorie, les RNN peuvent utiliser des informations dans des séquences arbitrairement longues, mais dans la pratique, on les limite à regarder seulement quelques étapes en arrière. Voici à quoi ressemble un RNN typique :

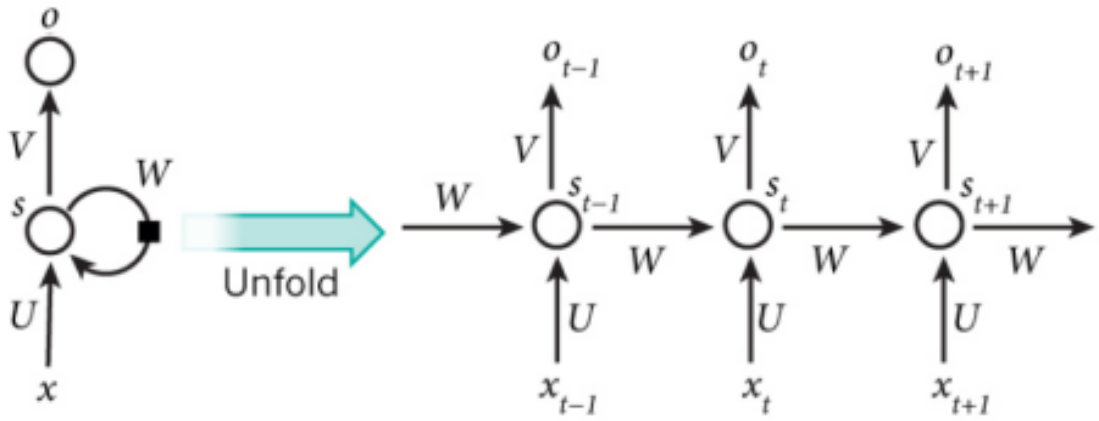


Figure 13 : (à gauche) un RNN (à droite), sa version déroulée Source : Nature

Source mémoire de master (spécialité Modèle intelligent et décision) Deep Learning pour la classification des images page 37.

Le schéma ci-dessus montre un RNN déroulé. En déroulant, nous signions simplement qu’on montre le réseau pour la séquence complète. Par exemple, si la séquence qui nous intéresse est une phrase de 5 mots, le réseau serait déroulé en un réseau de neurones de 5 couches, une couche pour chaque mot. Les formules qui régissent les calculs dans un RNN sont les suivantes :

- *X(t)* est l’entrée au moment *t.*
- *U,V,W* sont les paramètres que le réseau va apprendre des données de l’apprentissage.
- *S(t)* est l’état caché au moment *t.* C’est la « mémoire » du réseau. *S(t)* est calculé en fonction de l’état caché précédent et de l’entrée à l’étape actuelle :

$$s\,t = f(Ux\,t + Ws\,t-1) \qquad (4)$$

Où *f* est une fonction non linéaire telle que : ReLu ou Hyperbolic tangent (tanh).

- o t est la sortie au moment t. Par exemple, si on veut prédire le prochain mot dans une phrase, ce serait un vecteur de probabilités dans un vocabulaire.

$$o\ t = \text{softmax}(V\ s\ t) \tag{5}$$

Quelques remarques importantes :

- On peut considérer l'état caché s t comme la mémoire du réseau. s t capture des informations sur ce qui s'est passé dans toutes les étapes précédentes. La sortie o t est calculée uniquement en fonction de la mémoire au moment t. Comme mentionné brièvement ci-dessus, c'est un peu plus compliqué dans la pratique car s t ne peut généralement pas capturer des informations depuis trop longtemps.
- Contrairement à un réseau de neurones traditionnel, qui utilise différents paramètres à chaque couche, un RNN partage les mêmes paramètres (parameter sharing) (ici : U, V, W) à travers toutes les étapes. Cela reflète le fait que nous effectuons la même tâche à chaque étape, juste avec des entrées différentes. Ce qui réduit le nombre total de paramètres que le réseau devra apprendre.
- Le schéma ci-dessus présente des sorties à chaque moment, mais selon la tâche, ceci peut ne pas être nécessaire. Par exemple, lors de la prédiction du sentiment véhiculé par phrase, nous ne pouvons-nous soucier que de la sortie finale, pas du sentiment après chaque mot. De même, nous ne pouvons pas avoir besoin d'entrées à chaque étape.

2.2.2. Apprentissage

Pour l'apprentissage d'un RNN on utilise une version légèrement modifiée de la retro propagation appelé Backpropagation Through Time (BBTT) [50]. Etant donné que les paramètres sont partagés à tous moment dans le réseau, le gradient à chaque sortie dépend non seulement des calculs du moment actuel, mais aussi des étapes précédentes. On applique alors la règle de la chaîne.

On considère *O(t)* la prédiction du réseau au moment *t.* On traite la séquence complète (exemple une phrase complète) comme un seul exemple d'apprentissage, donc l'erreur totale est la somme des erreurs à chaque moment (chaque mot).

Le but est de calculer le gradient de l'erreur pour les paramètres *U, V, W* et d'apprendre de bon paramètre en utilisant Stochastic Gradient Descent (SGD) comme on a sommé les erreurs, on somme également les gradients à chaque étape pour un seul exemple d'apprentissage :

$$\frac{\gamma J}{\gamma W} = \sum \frac{\gamma \mathrm{Jt}}{\gamma W} \tag{6}$$

Pour calculer ces gradients, nous utilisons la règle de la chaîne. Par exemple l'erreur au moment t = 3 :

$$\frac{\partial J3}{\partial V} = \frac{\partial J3}{\partial o3}\frac{\partial o3}{\partial V} = \frac{\partial J3}{\partial o3}\frac{\partial o3}{\partial z3}\frac{\partial z3}{\partial V} \qquad (7)$$

Avec z3=Vs3, la remarque importante ici c'est que $\frac{\partial J3}{\partial V}$ dépend uniquement des valeurs du moment actuel o3, s3.

Mais c'est différent pour $\frac{\partial J3}{\partial W}$ (et pour U). Pour comprendre pourquoi, on écrit la règle en chaîne suivante :

$$\frac{\partial J3}{\partial W} = \frac{\partial J3}{\partial o3}\frac{\partial o3}{\partial s3}\frac{\partial s3}{\partial W} \qquad (8)$$

On note que s3=*f (Ux3+Ws2)* dépend de s2 qui dépend de W et de s1 et ainsi de suite. Donc si on prend la dérivée par rapport à W on ne peut pas traiter s2 comme une constante, on doit appliquer à nouveau la règle de la chaîne et ce qu'on obtient, c'est ceci :

$$\frac{\partial J3}{\partial W} = \sum \frac{\partial J3}{\partial o3}\frac{\partial o3}{\partial s3}\frac{\partial s3}{\partial sk}\frac{\partial sk}{\partial W} \qquad (9)$$

On somme la contribution au gradient de chaque moment. En d'autres termes, puisque *W* est utilisé à chaque moment jusqu'à la sortie, on doit retro propager le gradient de *t*=3 vers *t*=0 à travers tout le réseau.

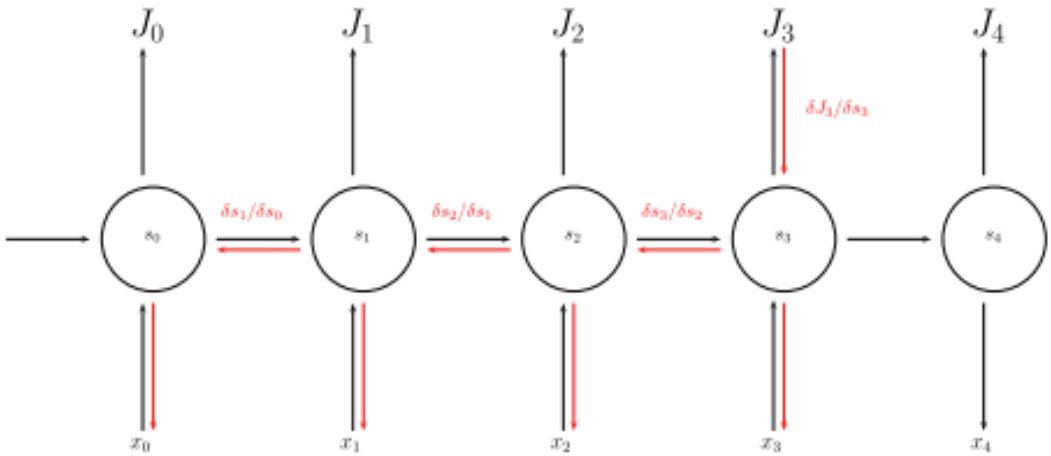


Figure 14 Backpropagation Through Time

Source mémoire de master (spécialité Modèle intelligent et décision) Deep Learning pour la classification des images page 38.

Les RNN ont des difficultés à apprendre des dépendances sur le long terme et les interactions entre des mots qui sont éloignés les uns des autres [38]. C'est problématique parce que la signification d'une phrase est souvent déterminée par des mots qui ne sont pas très proches. Dans ce cas précis si la fonction d'activation est une *tanh* ou bien une sigmoïde alors nous aurons le problème du **vanishing gradient**.

C'est pour cette raison qu'on se tourne vers la fonction d'activation ReLu qui elle ne souffre pas de ce problème, mais il existe une solution encore plus utilisée c'est l'utilisation des architectures **Long Short Term Memory (LSTM)** [39] ou **Gated Recurrent Unit (GRU)** [40].

2.2.3. Application

Les RNN ont connu un grand succès dans de nombreuses tâches de traitement du langage naturel. Les plus grands exploits des RNN ont été accomplis par l'architecture LSTM car ils sont bien meilleurs pour capturer des dépendances à long terme.

- Modélisation du langage et génération de texte [41] [42] [43] : compte tenu d'une séquence de mot, nous voulons prédire la probabilité de chaque mot sachant les mots précédents.
- Traduction automatique [44][45][46] : la traduction automatique est semblable à la modélisation du langage en ce sens que l'entrée est une séquence de mots dans une langue source (par exemple, l'arbre) .
- La reconnaissance vocale [47] : la reconnaissance vocale est une technique informatique qui permet d'analyser la voix humaine pour la transcrire sous la forme d'un texte exploitable par une machine.
- Description des images [48][49][50] : ensemble avec les réseaux neurones convolutifs, les RNN ont été utilisés pour générer des descriptions pour des images non étiquetées. Il est tout à fait incroyable de voir à quel point cela semble fonctionner. Le modèle combiné aligne même des mots générés avec des caractéristiques trouvées dans les images.

2.3. Deep generative model

Alors qu'un modèle discriminatif (exemple : CCN, RNN, MLP) essaye de prédire $p(y|x)$ avec y étant le label et x l'entrée, un modèle génératif décrit comment les données sont générées, il apprend $p(x,y)$ et fait des prédictions en utilisant la loi de Bayes pour calculer $p(y \mid x)$. [51]

Si le but est juste la classification, alors il faut utiliser un modèle discriminatif, cependant les modèles génératifs sont capables de bien plus que la simple classification comme par exemple générer de nouvelles observations.

Voici quelques exemples de modèle génératif :

- Boltzmann Machines [52]
- Restricted Boltzmann Machines [53] [54]
- Deep Belief Networks [55]
- Deep boltzmann Machines [56]
- Generative Adversarial Networks [57] [58] [59]
- Generative Stochastic Networks [60]
- Adversarial autoencoders [60]

3. La fonction d'activation

La fonction d'activation est une fonction mathématique appliquée à un signal en sortie d'un neurone artificiel. Le terme de "fonction d'activation" vient de l'équivalent biologique "potentiel d'activation", seuil de stimulation qui, une fois atteint entraîne une réponse du neurone. La fonction d'activation est souvent une fonction non-linéaire. Leur but est de permettre aux réseaux de neurones d'apprendre des fonctions plus complexes qu'une simple régression linéaire car le fait de multiplier les poids d'une couche cachée est juste une transformation linéaire.

3.1. Les caractéristiques

- **Non-linéarité** [51] : quand une fonction est non linéaire, un réseau neuronal à deux couches peut être considéré comme un approximateur de fonction universel.
- **Partout différentiable** [52] : Cette propriété permet de créer des optimisations basées sur les gradients.
- **Etendue** [53] : quand la plage d'activation est finie, les méthodes d'apprentissage basées sur les gradients sont plus stables (impact sur un nombre de poids limités). Quand la plage est infinie, l'apprentissage est généralement plus efficace (impact sur davantage le poids).
- **Monotone** [54] : lorsque la fonction est monotone, la surface d'erreur associée avec un modèle monocouche est certifié convexe.
- **Dérivée monotone :** les fonctions à dérivée monotone ont été montrées comme ayant une meilleure capacité de généraliser dans certains cas. Ces fonctions permettent d'appliquer les principes comme le rasoir d'Ockham.
- **Identité en 0** : ces fonctions permettent de faire un apprentissage rapide en initialisant les poids de manière aléatoire. Si la fonction ne converge pas vers l'identité en 0, alors un soin spécial doit être apporté lors de l'initialisation des poids.

3.2. Quelques fonctions d'activation

3.2.1. Rectified Linear Units et leurs généralisations

Les ReLu [55] utilisent la fonction d'activation

$$G(z)=\max\{0,z\}$$

Relu sont faciles à optimiser car ils sont tellement similaires aux unités linéaires. La seule différence entre une unité linéaire et ReLu délivre zéro sur tout l'ensemble R, ceci rend les dérivées via ReLu importante chaque fois que l'unité est active.

Des recherche montrent que ReLu sont un modèle encore meilleur de neurones biologiques [56] et produisent une performance égale ou meilleure que la tangente hyperbolique et un de leur avantage majeur est la réduction de la probabilité du vanishing gradient. Cela se produit lorsque z > 0. Dans ce cas, le gradient a une valeur constante puisque la dérivée est égale à 1 partout. En revanche, le gradient des sigmoïdes devient de plus en plus petit car la valeur absolue de z augmente. Le gradient constant de ReLu entraîne un apprentissage plus rapide.

Lors de l'initialisation des paramètres il peut être judicieux d'initialiser les valeurs des biais à une petite valeur positive, telle que 0,1. Il est très probable que ReLu seront initialement actives pour la plupart des entrées de l'ensemble d'apprentissage et permettent aux dérivées de passer à travers.

Il existe plusieurs généralisations de ReLu. La plupart de ces généralisations fonctionnent de manière comparable et de temps à autre, se comporte mieux.

Un inconvénient de ReLu est qu'ils ne peuvent pas apprendre par la descente de gradient sur des exemples pour lesquels leur activation est nulle « *dying ReLu» problem.* Une variété de généralisation de ReLu garantit qu'elles reçoivent le gradient partout.

Les généralisations de ReLu sont basées sur l'utilisation d'une pente non nulle α i lorsque z i < 0 : h i = g(z, α) = max(z i , 0) + α i min(0, z i). Absolute value rectification fixe α i = −1 pour obtenir g(z) = |z|. Leaky ReLu (LReLu) [57] fixe α i à une petite valeur comme 0.01 alors que parametric ReLu (PReLu) [58] traite α i comme un paramètre apprenable.

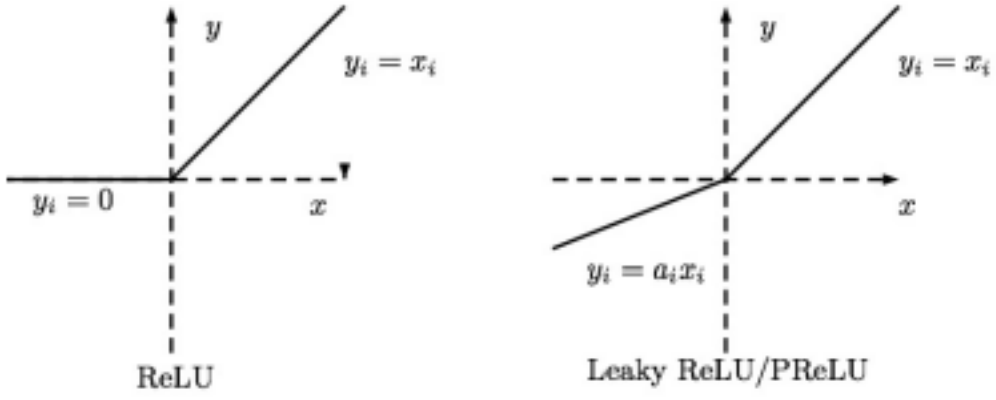


Figure 15 : (à gauche) la fonction d'activation ReLu.

Cette fonction d'activation est la fonction d'activation par défaut recommandée pour l'utilisation des réseaux de neurones feedforward. L'application de cette fonction à la sortie d'une transformation linéaire donne une transformation non linéaire. Ils

préservent plusieurs des propriétés qui rendent les modèles linéaires faciles à optimiser avec la descente de gradient. (à droite) Leaky ReLu et Parametric ReLu qui corrige l'inconvénient de ReLu et qui sont différentiables à 0.

3.2.2. Sigmoïde et tangente hyperbolique

Avant l'introduction de ReLu la plupart des réseaux neuronaux utilisent la sigmoïde ou tanh.

La sigmoïde : $g(z) = = \frac{1}{1+e^{-\lambda z}}$ (10)

tanh : $g(z) = = \frac{1+e^{-2z}}{1+e^{-2z}}$ (11)

Ce qu'on reproche aux unités sigmoïdales c'est qu'elles saturent dans la plupart de leur domaine : elles saturent à une valeur élevée lorsque z est très positif et à une valeur faible lorsque z est très négatif et ne sont que très sensibles à leur entrée lorsque z est proche de 0. La saturation généralisée des unités sigmoïdales peut rendre l'apprentissage par la méthode de la descente de gradient très difficile. Pour cette raison, leur utilisation dans les couches cachées est maintenant déconseillée.

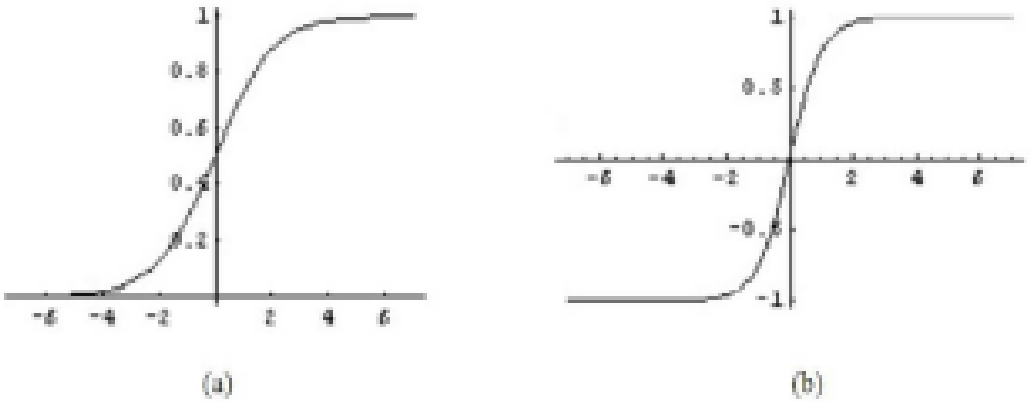


Figure 16: (a) Sigmoïde. (b) tangente hyperbolique

4. La famille de modèles R-CNN

La famille de méthodes R-CNN fait référence au R-CNN, qui peut signifier « régions avec des caractéristiques CNN » ou « réseau neuronal convolutionnel basé sur une région », développé par Ross Girshick, et al.

Cela inclut les techniques R-CNN, Fast R-CNN et Faster-RCNN conçues et démontrées pour la localisation et la reconnaissance d'objets.

4.1. Le R-CNN

Le R-CNN a été décrit dans l'article de 2014 par Ross Girshick et al. De UC Berkely intitulé « *Hiérarchies de fonctionnalités riches pour une détection précise des objets et une segmentation sémantique* ».

Il a peut-être été l'une des premières applications importantes et réussies de réseaux de neurones convolutifs au problème de la localisation, de la détection et de la segmentation des objets. L'approche a été démontrée sur des ensembles de données de référence, obtenant alors des résultats de pointe sur l'ensemble de données VOC-2012 et l'ensemble de données de détection d'objets ILSVRC-2013 de classe 200.

Leur modèle R-CNN proposé comprend trois modules, elles sont :

- **Module 1: Proposition de région**. Générez et extrayez des propositions de régions indépendantes des catégories, par exemple des boîtes de délimitation candidates.
- **Module 2 : Extracteur de fonctionnalités**. Extraire la fonction de chaque région candidate, par exemple en utilisant un réseau de neuronal convolutionnel profond.
- **Module 3** : **classificateur.** Classer les entités comme l'une des classes connues, par exemple le modèle de classificateur SVM linéaire.

L'architecture du modèle est résumée dans l'image ci-dessous, extraite du papier.

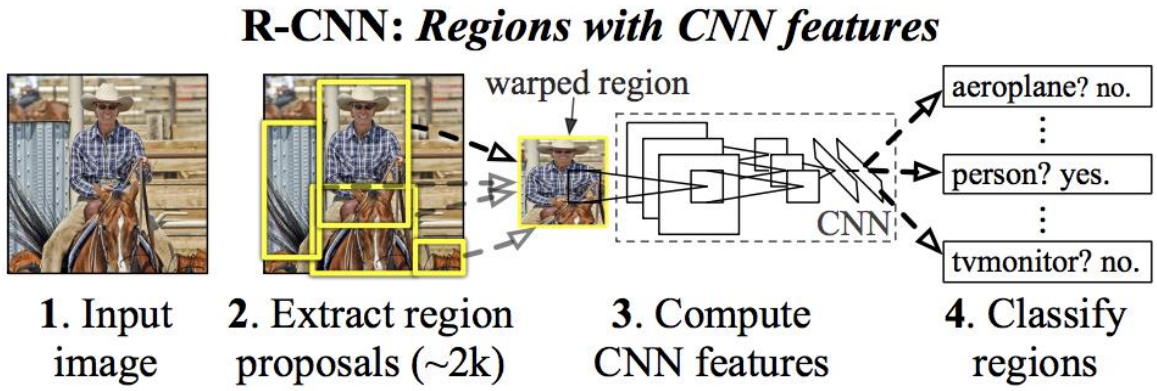


Figure 17: résumé de l'architecture du modèle R-CNN tiré des hiérarchies de fonctionnalités riches pour une détection précise des objets et une segmentation sémantique.

Source https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46

Une technique de vision par ordinateur est utilisée pour proposer des régions candidates ou des boîtes englobantes d'objets potentiels dans l'image appelée « *recherche sélective* », bien que la flexibilité de la conception permette d'utiliser d'autres algorithmes de proposition de région.

L'extracteur de fonctionnalités utilisé par le modèle était le CNN profond d'AlexNet qui a remporté le concours de classification d'images ILSVRC-2012. La sortie du CNN était un vecteur d'éléments 4 096 qui décrit le contenu de l'image qui est envoyée à un SVM linéaire pour la classification, en particulier un SVM est formé pour chaque classe connue.
Il s'agit d'une application relativement simple et directe des CNN au problème de la localisation et de la reconnaissance d'objets. Un inconvénient de l'approche est qu'elle est lente, nécessitant une passe d'extraction de fonctionnalités basée sur CNN sur chacune des régions candidates générées par l'algorithme de proposition de région. Il s'agit d'un problème car l'article décrit le modèle fonctionnant sur environ 2 000 régions proposées par image au moment du test.

4.2. Le Fast R-CNN

Étant donné le grand succès de R-CNN, Ross Girshick, alors à Microsoft Research, a proposé une extension pour résoudre les problèmes de vitesse du R-CNN dans un article de 2015 intitulé « Fast R-CNN ».
Le document commence par un examen des limites du R-CNN, qui peut être résumé comme suit :

- **La formation est un pipeline en plusieurs étapes.** Implique la préparation et le fonctionnement de trois modèles distincts.
- **La formation coûte cher en espace et en temps.** Former un CNN profond sur autant de propositions de régions par image est très lent.
- **La détection d'objets est lente.** Faire des prédictions en utilisant un CNN profond sur tant de propositions de région est très lent.

Un travail antérieur a été proposé pour accélérer la technique appelée réseaux de mise en commun de pyramides spatiales, ou SPPnets, dans l'article de 2014 intitulé *« Mise en commun de pyramides spatiales dans les réseaux convolutionnels profonds pour la reconnaissance visuelle* ». Cela a accéléré l'extraction des entités, mais a essentiellement utilisé un type de l'algorithme de mise en cache de la transmission directe.
Fast R-CNN est proposé comme un modèle unique au lieu d'un pipeline pour apprendre et produire des régions et des classifications directement.
L'architecture du modèle prend la photographie d'un ensemble de propositions de régions en entrée qui sont passées à travers un réseau neuronal convolutionnel profond. Un CNN préformé, tel qu'un VGG-16, est utilisé pour l'extraction des fonctionnalités. La fin du CNN profond est une couche personnalisée appelée couche de regroupement de régions d'intérêt, ou regroupement RoI, qui extrait des fonctionnalités spécifiques à une région candidate d'entrée donnée.

La sortie du CNN est ensuite interprétée par une couche entièrement connectée, puis le modèle se divise en deux sorties, une pour la prédiction de classe via une couche softmax, et une autre avec une sortie linéaire pour la boîte englobante. Ce processus est ensuite répété plusieurs fois pour chaque région d'intérêt dans une image donnée.
L'architecture du modèle est résumée dans l'image ci-dessous, extraite du papier.

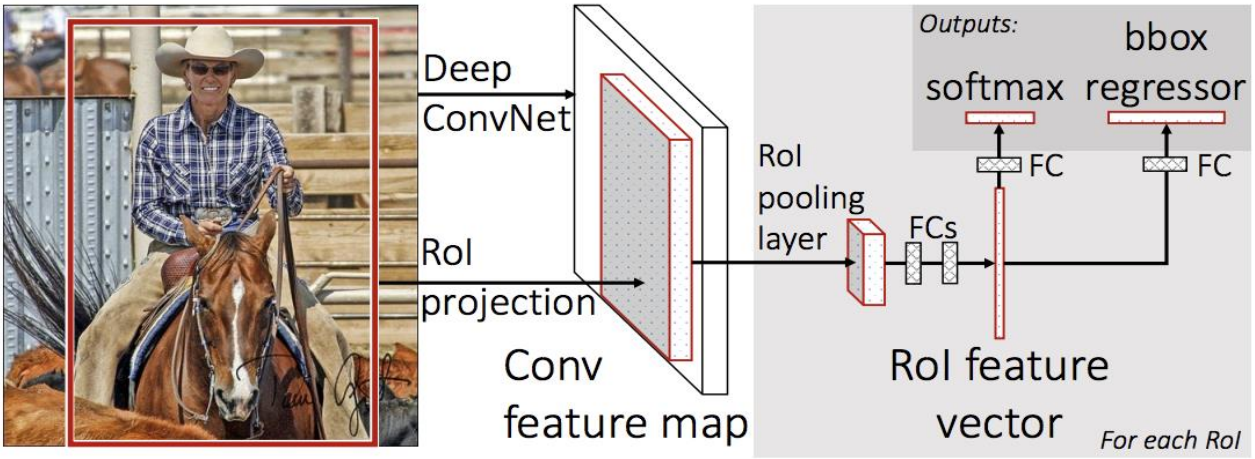


Figure 18 : résumé de l'architecture du modèle Fast R-CNN.

Source https://towardsdatascience.com/faster-r-cnn-for-object-detection-a-technical-summary-474c5b857b46

Le modèle est beaucoup plus rapide à former et à faire des prévisions, mais nécessite toujours un ensemble de régions candidates à proposer avec chaque image d'entrée.

4.3. Le Faster R-CNN

L'architecture du modèle a été encore améliorée pour la vitesse d'entrainement et la détection par Shaoqinq Ren, et al, à Microsoft Research dans le document de 2016 intitulé « *Faster R-CNN: Towards Real-Time Object Detection with Region Proposition Networks* ».

L'architecture a été la base des premiers résultats obtenus sur les tâches de compétition de reconnaissance et de détection d'objets.

L'architecture a été conçue pour proposer et affiner les propositions régionales dans le cadre du processus de formation, appelé réseau de propositions régionales ou RPN. Ces régions sont ensuite utilisées de concert avec un modèle Fast R-CNN dans un modèle unique. Ces améliorations permettent à la fois de réduire le nombre de propositions de régions et d'accélérer le fonctionnement en temps d'essai du modèle en temps quasi réel avec des performances de pointe.

Les premiers tests de ce système de détection avaient une fréquence d'images de 5 images par seconde (toutes les étapes comprises) sur un GPU, tout en garantissant une précision de détection des objets de pointe sur les jeux de données PASCAL VOC 2007, 2012 et MS COCO avec seulement 300 propositions par image. Dans les compétitions ILSVRC et COCO 2015, Faster R-CNN et RPN sont les fondements de la première place gagnante sur plusieurs pistes.

Le Faster R-CNN : vers la détection d'un modèle unifié unique, l'architecture est composée de deux modules :

- **Module 1 : réseau de propositions régionales.** Réseau de neurones convolutifs pour proposer des régions et le type d'objet à considérer dans la région.
- **Module 2 : Fast R-CNN**. Réseau de neurones convolutifs pour extraire des caractéristiques des régions proposées et produire la boîte englobante et les étiquettes de classe.

Les deux modules fonctionnent sur la même sortie d'un CNN profond. Le réseau de proposition de région agit comme un mécanisme d'attention pour le réseau Fast R-CNN, informant le deuxième réseau de l'endroit où chercher ou prêter attention.

L'architecture du modèle est résumée dans l'image ci-dessous, extraite du papier.

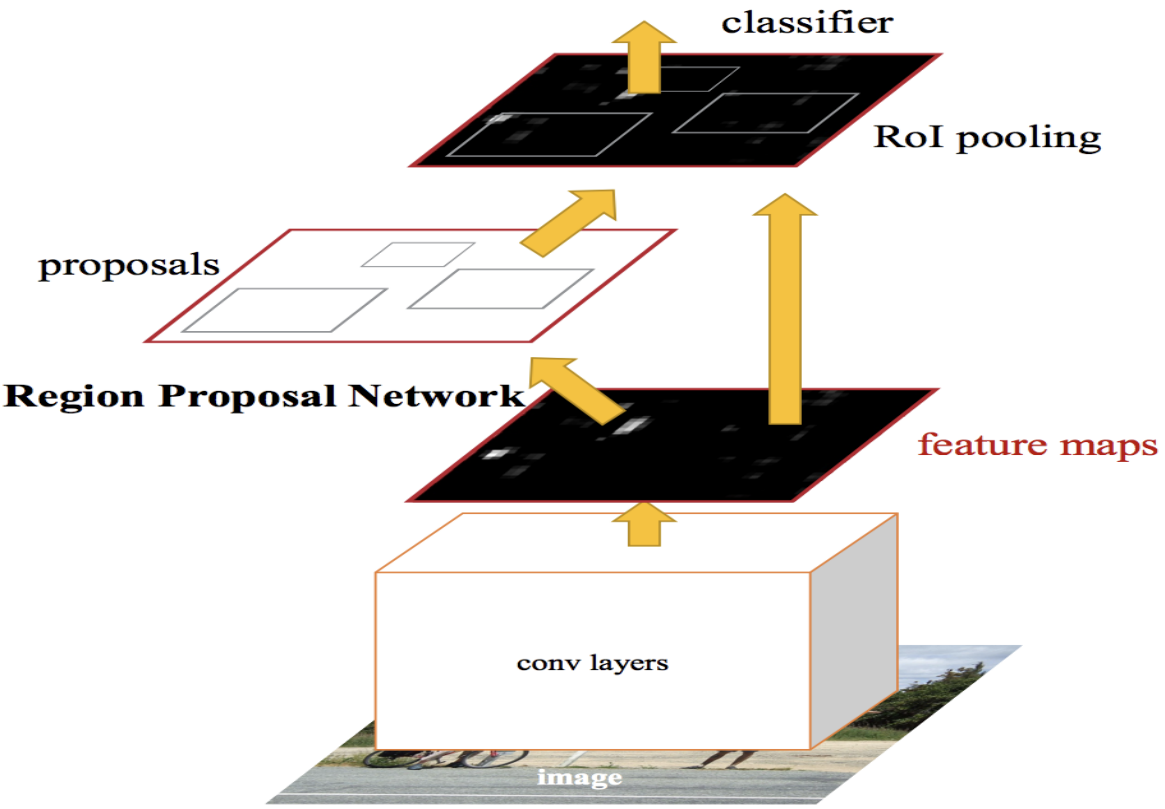


Figure 19: Résumé de l'architecture du modèle Faster R-CNN vers la détection d'objets en temps réel avec les réseaux de propositions de régions

Source https://zhuanlan.zhihu.com/p/64134690

Le RPN fonctionne en prenant la sortie d'un CNN profond préformé, tel que VGG-16, et en passant par un petit réseau sur la carte des fonctionnalités et en émettant plusieurs propositions de région et une prédiction de classe pour chacune. Les propositions de région sont des boîtes englobantes, basées sur des boîtes dites d'ancrage ou des formes prédéfinies conçues pour accélérer et améliorer la proposition de régions. La prédiction de classe est binaire, indiquant la présence ou non d'un objet, appelé « *objectivité* » de la région proposée.

Une procédure de formation alternée est utilisée lorsque les deux sous-réseaux sont formés en même temps, bien qu'entrelacés. Cela permet aux paramètres dans le détecteur de caractéristiques CNN profond d'être adaptés ou affinés pour les deux tâches en même temps.

Au moment de la rédaction de cet article, cette architecture R-CNN plus rapide est le summum de la famille des modèles et continue d'atteindre des résultats de pointe sur les tâches de reconnaissance d'objets. Une autre extension ajoute la prise en charge de la segmentation d'image, décrite dans l'article de 2017 « Mask R-CNN ».

Ainsi les trois familles des R-CNN sont évaluées par rapport à la rapidité.

Tableau 2: comparaison des algorithmes de détection d'objets

Algos / **Critères**	**R-CNN**	**FAST R-CNN**	**FASTER R-CNN**
Temps (s)	50	2	0,2
Rapidité (x)	1	25	250

5. La famille de modèles YOLO

Une autre famille populaire de modèles de reconnaissance d'objets est appelée collectivement YOLO ou « You Only Look Once », développée par Joseph Redmon, et al.

5.1. YOLO

Le modèle YOLO a été décrit pour la première fois par Joseph Redmon et al, dans l'article de 2015 intitulé « Vous ne regardez qu'une seule fois : détection unifiée et en temps réel des objets ». Notez que Ross Girshick, développeur de R-CNN, était également auteur et contributeur à ce travail, puis à Facebook Al Research.

L'approche implique un réseau neuronal unique formé de bout en bout qui prend une photo en entrée et prédit directement les boîtes englobantes et les étiquettes de classe pour chaque boîte englobante. La technique offre une précision prédictive plus faible (par exemple plus d'erreurs de localisation), bien qu'elle fonctionne à 45 images par seconde et jusqu'à 155 images par seconde pour une version du modèle à vitesse optimisée.

Le modèle fonctionne en divisant d'abord l'image d'entrée en une grille de cellules, où chaque cellule est responsable de la prédiction d'un cadre de délimitation si le centre d'un cadre de délimitation s'y trouve. Chaque cellule de la grille prédit une boîte englobante impliquant les coordonnées x, y et la largeur et la hauteur et la confiance. Une prédiction de classe est également basée sur chaque cellule.

Par exemple, une image peut être divisée en une grille 7 × 7 et chaque cellule de la grille peut prédire 2 boîtes de délimitation, résultant en 94 prédictions de boîtes de délimitation proposées. La carte des probabilités de classe et les boîtes englobantes avec des confidences sont ensuite combinées en un ensemble final de boîtes englobantes et d'étiquettes de classe. L'image tirée de l'article ci-dessous résume les deux sorties du modèle.

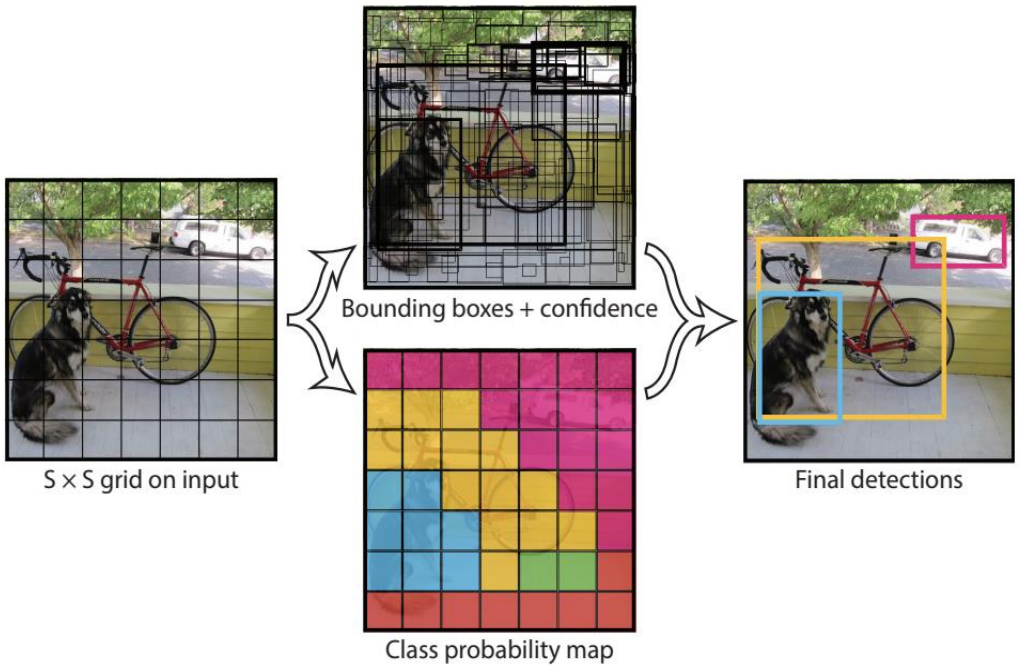


Figure 20: résumé de prédiction par le modèle YOLO.

Source http://aidev.co.kr/deeplearning/7737

5.2. YOLOv2 (YOLO900) et YOLOv3

Le modèle a été mis à jour Joseph Redmon et Ali Farhadi dans le but d'améliorer encore les performances du modèle dans leur article de 2016 intitulé « YOLO9000 : Better, Faster, Stronger ».

Bien que cette variation du modèle soit appelée YOLO v2, une instance du modèle est décrite qui a été formée sur deux ensembles de données de reconnaissance d'objets en parallèle, capables de prédire 9 000 classes d'objets, d'où le nom de « *YOLO9000* ».
Un certain nombre de modifications de formation et d'architecture ont été apportées au modèle, telles que l'utilisation de la normalisation par lots et des images d'entrée à haute résolution.
Comme Faster R-CNN, le modèle YOLOv2 utilise des boîtes d'ancrage, des boîtes de délimitation prédéfinies avec des formes et des tailles utiles qui sont adaptées pendant la formation. Le choix des boîtes englobantes pour l'image est prétraité à l'aide d'une analyse k-means sur l'ensemble de données d'apprentissage.
Surtout, la représentation prédite des boîtes englobantes est modifiée pour permettre à de petits changements d'avoir un effet moins dramatique sur les prédictions, résultant en un modèle plus stable. Plutôt que de prédire directement la position et la taille, des décalages sont prévus pour déplacer et remodeler les boîtes d'ancrage prédéfinies par rapport à une cellule de grille et amortis par une fonction logistique.

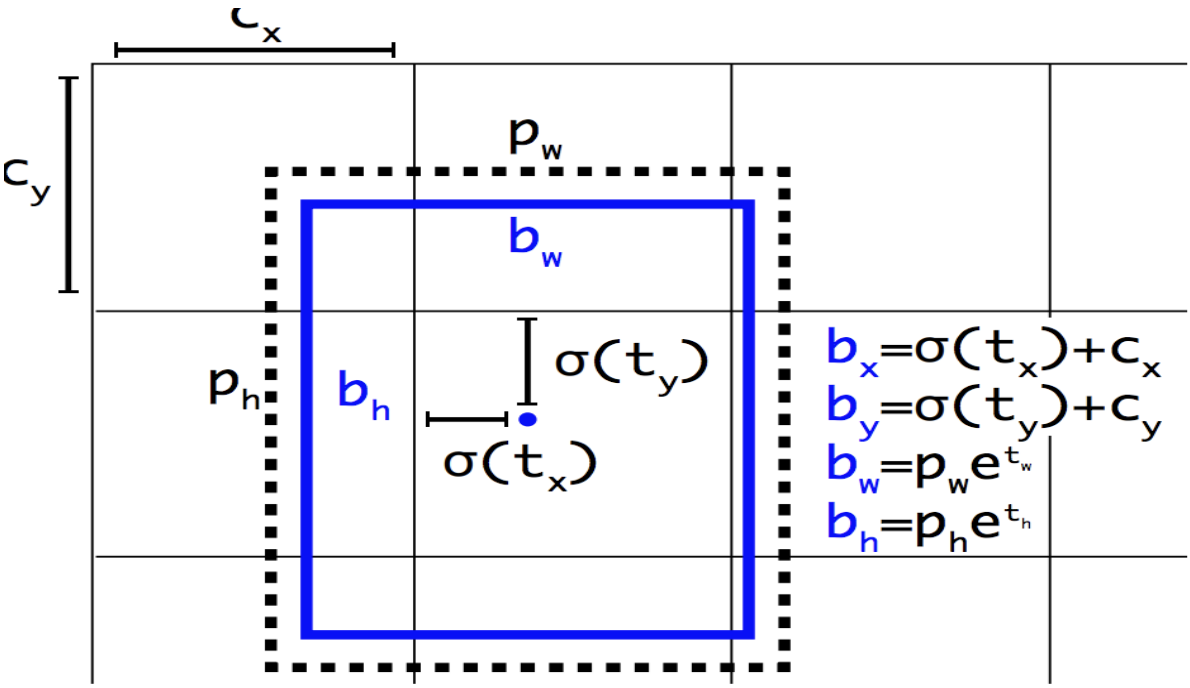


Figure 21: exemple de représentation choisie lors de la prévision de la position et de la forme de la boîte englobante.

6. SSD (**S**ingle **S**hot **D**etector)

SSD réalise un bon équilibre entre vitesse et précision. SSD exécute un réseau de convolution sur l'image d'entrée une seule fois et calcule une carte de caractéristiques. A présent, nous exécutons un petit noyau de convolution de taille 3x3 sur cette carte de fonctions pour prédire les boîtes englobantes et la probabilité de classification. Le SSD utilise également des boîtes d'ancrage dans divers formats similaires à celui de Faster R-CNN et apprend le décalage plutôt que l'apprentissage de la boîte. Afin de gérer l'échelle, SSD prédit les boîtes englobantes après plusieurs couches convolutifs. Etant donné que chaque couche de convolution fonctionne à une échelle différente, elle est capable de détecter des objets de différentes échelles.

Cela fait beaucoup d'algorithmes. Lequel devriez-vous utiliser? Actuellement, Faster-RCNN est le choix si vous êtes fanatique des chiffres de précision. Cependant, si vous êtes prêt au calcul (probablement sur Nvidia Jetsons), SSD est une meilleure recommandation. Enfin, si la précision n'est pas une préoccupation majeure mais que vous souhaitez aller très vite, YOLO sera la solution. Tout d'abord, une compréhension visuelle du compromis vitesse / précision.



Figure 22: comparaison des méthodes de détection en fonction de la précision par rapport à la vitesse.

7. Le software

Plusieurs framework open sources sont disponibles dans la littérature, la grande majorité supporte le langage Python. Voici une liste non exhaustive de quelques framework.

7.1. Theano

Créé par Frédéric Bastien et l'équipe de recherche derrière le laboratoire de l'Université de Montréal, MILA.

- **Avantages**
  - Python
  - Performant si utilisé correctement
- **Inconvénients**
  - Nécessite une solution de contournement 49 pour supporter le Multi GPU.
  - Les grands modèles peuvent nécessiter un long temps de compilation.
  - API de bas niveau.

7.2. TensorFlow

TensorFlow fut créé par l'équipe Google Brain pour mener des recherches sur le ML et le Deep Learning. Il est considéré comme une version moderne de Theano.

- **Avantages**
  - Python
  - Supporté par Google
  - Une très grande communauté
  - Le support de multi-GPU.
- **Inconvénients**
  - Plus lent que les autres framework dans de nombreux benchmarks, bien que Tensorflow se rattrape.
  - Le soutien des RNN est encore surclassé par Theano.

7.3. Keras

Le framework le plus haut niveau, le plus convivial de la liste. Il permet aux utilisateurs de choisir si les modèles qu'ils construisent sont exécutés sur Theano ou TensorFlow. Il est écrit et entretenu par Francis Chollet, un autre membre de l'équipe Google Brain.

- **Avantages**
  - Python
  - Le backend par excellence pour Theano ou TensorFlow
  - Interface haut niveau, intuitive.
- **Inconvénients**
  - Moins flexible que les autres API.

7.4. PyTorch

Le dernier né de la liste, il est soutenu par l'équipe de recherche en intelligence artificielle de Facebook. Il permet le traitement des entrées et des sorties de longueur

variable (dynamic computation graphs), ce qui est utile lorsque on travaille avec des RNN. Une caractéristique absente de TensorFlow et Theano.

- **Avantages**
  - Python
  - Soutenu par Facebook
  - Mélange d'API de haut niveau et de bas niveau.
  - Supporte des graphes dynamiques
  - Il semblerait que c'est le meilleur outil pour les chercheurs
- **Inconvénients**
  - Beaucoup moins sûr que ses concurrents (Dans leurs propres termes : ''We are in an earlyrelease Beta. Expect some adventures'')
  - Référence/ressources limitées en dehors de la documentation officielle.

7.5. Caffee

Caffe est une bibliothèque d'apprentissage en profondeur Python développée par Yangqing Jia de l'Université de Berkeley pour des problèmes de vision assistée par ordinateur. Il s'agissait de la bibliothèque d'apprentissage en profondeur la plus populaire utilisée. Écrit en C ++, Caffe est l'une des bibliothèques les plus anciennes et les plus largement supportées pour CNN et la vision par ordinateur. La plate-forme Nvidia Jetson pour l'informatique embarquée supporte profondément Caffe (ils ont ajouté le support pour d'autres Framework comme Tensorflow mais ce n'est toujours pas suffisant). Il en va de même pour OpenCV, la bibliothèque de vision par ordinateur largement utilisée qui a commencé à ajouter le support pour les modèles Deep Learning à partir de Caffe. OpenCV est depuis des années le moyen le plus répandu d'ajouter des fonctionnalités de vision par ordinateur aux appareils mobiles. Donc, si vous avez une application mobile qui exécute openCV et que vous souhaitez maintenant déployer un modèle basé sur un réseau Neural, Caffe serait très pratique.

7.6. CNTK

La structure CNTK (Microsoft Cognitive Toolkit) est gérée et prise en charge par Microsoft. Puisque nous avons une expérience limitée avec CNTK, nous ne faisons que le mentionner ici. Cependant, ce n'est pas extrêmement populaire comme Tensorflow / Pytorch / Caffe.

7.7. Caffe2

Un autre cadre supporté par Facebook, construit sur le Caffe original, a été conçu par le créateur de Caffe, Yangqing Jia. Il a été conçu dans un souci d'expression, de rapidité et de modularité, en particulier pour le déploiement en production, ce qui n'a jamais été l'objectif de Pytorch.

Récemment, Caffe2 a été fusionné avec Pytorch afin de fournir des capacités de déploiement de production à Pytorch, mais nous devons attendre et regarder comment cela se passe. La feuille de route de Pytorch 1.0 parle de la prise en charge du déploiement de la production à l'aide de Caffe2.

7.8. MXNet

Promu par Amazon, MxNet est également pris en charge par Apache Foundation. Il est très populaire parmi la communauté R bien qu'il ait une API pour plusieurs langues. Keras est également l'un des serveurs principaux.

7.9. Torche

Torch (également appelé Torch7) est un Framework d'apprentissage en profondeur développé par Clement Farabet, Ronan Collobert et Koray Kavukcuoglu pour la recherche et le développement d'algorithmes d'apprentissage en profondeur. La torche a été utilisée et développée par le laboratoire d'IA de Facebook. Cependant, l'essentiel de la force derrière la torche s'est déplacé à Pytorch.

7.10. Darknet

Darknet est un Framework de réseau de neurones open source écrit en C et CUDA. Il est rapide facile à installer et prend en charge le calcul des processus et des GPU. Nous précisons que Darknet est le framework natif de YOLOv3 qui est un algorithme de détection.

7.11. Comparaison des frameworks de l'IA

Tableau 3:comparaison des frameworks

Critère	Gagnant	Finaliste
Communauté et soutien	TensorFlow	Pytorch
Facile à utiliser	Pytorch	TensorFlow
Académie	PyTorch	Torche/Caffe
Industrie	TensorFlow	Caffe2
Vision par ordinateur embarque	Caffe	TensorFlow

Conclusion

Dans ce chapitre, nous avons fait l'état de l'art sur les algorithmes du deep Learning à savoir les réseaux de neurones artificiels pour comprendre les concepts liés à la détection et reconnaissance des objets dans l'image. Puis dans le chapitre suivant nous allons faire

une analyse de la conception de notre système puis expliqué les étapes d'implémentation de ce dernier.

Chapitre 3 : mise en œuvre, analyse et conception

3.1. Logiciels et librairies utilisés dans l'implémentation

3.1.1. Analyse des besoins du système

3.1.2. Besoins fonctionnels

Ce sont les exigences liées au fonctionnement du système. Le système intelligent de détection d'objets devra être capable d'établir les fonctionnalités suivantes :

- ✓ Détecter un objet un mouvement
- ✓ Reconnaitre l'objet c'est-à-dire le classifier
- ✓ Enregistrer les objets détectés.

3.1.3. Besoins non fonctionnels

Les exigences non fonctionnelles répondent à des besoins transversaux aux besoins métiers nous distinguons :

- ✓ La rapidité ;
- ✓ La performance ;
- ✓ La précision ;
- ✓ L'extensibilité

3.2. Matériels et logiciels

3.2.1. Matériels

Pour la mise en œuvre de notre solution nous avons besoin :

- ✓ 1 serveur qui va servir pour la détection des objets.
- ✓ 1 serveur qui va faire office de SGBD (Base de données).
- ✓ Une caméra Ip pour la prise de vues, l'enregistrement et la transmission des images photographiques successives.

3.2.2. Logiciels et plateforme

- OpenCV

OpenCV a été lancé chez Intel en 1999 par **Gary Bradsky** et la première publication est sortie en 2000. **Vadim Pisarevsky** a rejoint **Gary Bradsky** pour gérer l'équipe logicielle OpenCV d'Intel en Russie. En 2005, OpenCV a été utilisé sur Stanley, le véhicule qui a remporté le DARPA Grand Challenge 2005. Plus tard, son développement actif s'est poursuivi avec le soutien de Willow Garage, dirigé par Gary Bradsky et Vadim Pisarevsky. À l'heure actuelle, OpenCV prend en charge de nombreux algorithmes liés à la vision par ordinateur et à l'apprentissage automatique et se développe jour après jour.

Actuellement, OpenCV prend en charge une grande variété de langages de programmation tels que C ++, Python, Java, etc., et est disponible sur différentes plateformes, notamment Windows, Linux, OS X, Android, iOS, etc. De plus, les interfaces basées sur CUDA et OpenCL sont également en cours d'opérations GPU à

grande vitesse. OpenCV-Python est l'API Python d'OpenCV. Il combine les meilleures qualités de l'API OpenCV C ++ et du langage Python.

- OpenCV-Python

Python est un langage de programmation généraliste lancé par Guido van Rossum , qui est devenu très populaire en peu de temps, principalement pour sa simplicité et sa lisibilité du code. Il permet au programmeur d'exprimer ses idées en moins de lignes de code sans réduire la lisibilité.

Comparé à d'autres langages tels que C / C ++, Python est plus lent. Mais une autre caractéristique importante de Python est qu'il peut être facilement étendu avec C / C ++. Cette fonctionnalité nous aide à écrire des codes gourmands en calculs en C / C ++ et à créer un wrapper Python pour pouvoir les utiliser en tant que modules Python. Cela nous donne deux avantages: premièrement, notre code est aussi rapide que le code original C / C ++ (puisqu'il s'agit du code C ++ réel travaillant en arrière-plan) et deuxièmement, il est très facile de coder en Python. C'est ainsi que fonctionne OpenCV-Python, c'est un wrapper Python autour de l'implémentation C ++ originale.

Et le support de Numpy facilite la tâche. Numpy est une bibliothèque hautement optimisée pour les opérations numériques. Il donne une syntaxe de style MATLAB. Toutes les structures de tableau OpenCV sont converties dans et à partir de tableaux Numpy. Ainsi, quelles que soient les opérations que vous pouvez effectuer dans Numpy, vous pouvez les combiner avec OpenCV, ce qui augmente le nombre d'armes dans votre arsenal. En plus de cela, plusieurs autres bibliothèques comme SciPy, Matplotlib qui supporte Numpy peuvent être utilisées avec cela.

OpenCV-Python est donc un outil approprié pour le prototypage rapide de problèmes de vision par ordinateur.

- YOLO

YOLO (You Only Look Once) est un réseau qui utilise des algorithmes d'apprentissage approfondi (DL) pour la détection d'objets. YOLO détecte les objets en classant certains objets dans l'image et en déterminant leur emplacement.

Par exemple, si vous saisissez une image d'un troupeau de moutons dans un réseau YOLO, un fichier de boîtes englobantes sera généré en sortie pour chaque mouton et classé comme tel.

Tableau 4: Yolo avantages et inconvénients

Avantages	Inconvénients
-Accès aux données système via des vues -Open Source -Compression des données et des sauvegardes -Plusieurs moteurs de stockage-Procédures stockées -Déploiement et prise en main facile	-Cluster par clonage de base -Support incomplet des triggers et procédures stockées.

- ❖ L'algorithme YOLO

Une fois que vous avez inséré une image dans un algorithme YOLO, les images sont divisées en une grille SxS qu'il utilise pour prédire si le cadre de sélection contient ou non l'objet (ou des parties de celui-ci), puis utilise ces informations pour prédire une classe pour l'objet.

Avant de pouvoir entrer dans les détails et expliquer le fonctionnement de l'algorithme, nous devons comprendre comment cet algorithme construit et spécifie chaque cadre de sélection. L'algorithme YOLO utilise quatre composants et une valeur supplémentaire pour prédire une sortie.

Le centre d'un cadre de sélection (**b x b y**)

- Largeur (**b w**)
- Hauteur (**b h**)
- La classe de l'objet (**c**)

La valeur prédite finale est la confiance (**p c**). Il représente la probabilité de l'existence d'un objet dans la boîte englobante.

Les coordonnées (x, y) représentent le centre du cadre de sélection.

Généralement, la plupart des boîtes englobantes ne contiennent pas d'objet. Nous devons donc utiliser la prédiction pc. Nous pouvons utiliser un processus appelé suppression non-max pour supprimer les boîtes inutiles ayant une faible probabilité de contenir des objets et celles partageant de grandes zones avec d'autres boîtes. Yolo présente 3 version nous allons utiliser la dernier version Yolov3.

- ❖ YOLO V3

YOLO V3 est une mise à niveau incrémentielle par rapport à YOLO V2, qui utilise une autre variante de Darknet. Cette architecture YOLO V3 est composée de 53 couches formées sur ImageNet et de 53 autres chargées de la détection d'objets, soit

106 couches. Bien que cela ait considérablement amélioré la précision du réseau, il a également réduit la vitesse de 45 Fps à 30 Fps.

Les mises à niveau de YOLO V3 par rapport à YOLO V2 comprennent:

- Amélioration de la prédiction du cadre de sélection: elle utilise la régression logistique pour effectuer un résultat de prévision pour tous les objets de chaque cadre de sélection ;
- La prédiction de classes plus précises : le logiciel softmax utilisé pour YOLO V2 a été remplacé par des classificateurs logistiques pour chaque classe à des fins de multi-étiquetage.
- Capacités améliorées à différentes échelles - permet d'effectuer 3 prévisions pour chaque emplacement de l'image d'entrée afin de permettre au sur échantillonnage des couches précédentes d'obtenir des informations détaillées, une information sémantique complète et d'améliorer la qualité des résultats.

Il existe aussi un très petit modèle pour l'environnement contraint yolov3-tiny. C'est-à-dire que ce modèle peut être déployé dans un Raspberry, smartphone, etc.

Dans ce travail nous avons utilisé des poids convolutionnels préformés sur ImageNet. Nous utilisons des poids du modèle darknet53. Vous pouvez simplement téléchargez les poids pour les couches convolutifs et l'appliquer sur votre yolo et vous serez en mesure de détecté 80 classes.

- Open Image Dataset

Vous pouvez entrainer YOLO à partir de zéro si vous souhaite jouer avec différents régimes d'entrainement, hyper-paramètres ou jeux de données. Open Image est une version collaborative d'environ 9 millions d'images annotées avec des étiquettes au niveau de l'image, des cadres de délimitation d'objet, des masques de segmentation d'objet et des relations visuelles. Cet ensemble de données unique et diversifié est conçu pour stimuler les progrès de pointe dans l'analyse et la compréhension des images.

La sortie d'Open Images V5 de cette année a permis au deuxième Open Images Challenge d'inclure les 3 pistes suivantes :

- ✓ Piste de détection d'objets pour détecter les boîtes englobantes autour des instances d'objets, relancée à partir de 2018.
- ✓ Piste de décision de relation visuelle pour détecter des paires d'objets dans les relations particulières, également relancée à partir de 2018.
- ✓ Piste de segmentation d'instance pour la segmentation de masques d'objets en images, toute nouvelle pour 2019.

Google AI espère qu'un système de données unique avec des annotations unifiées pour la classification des images, la détection des objets, la détection des relations visuelles

et la segmentation des instances stimulera les progrès vers une véritable compréhension des scènes.

- ArgoUML

ArgoUML est un logiciel libre de création de diagrammes. Il propose aux développeurs un outil de représentation UML. L'application est compatible avec tous les diagrammes UML 1.4 standards et fonctionne sur n'importe quelle plateforme Java. ArgoUML est livré avec des profils pour le développement d'application C++ et Java. Il supporte les diagrammes de classe, les diagrammes d'état, les diagrammes "Use Case" ou encore les diagrammes de séquence, et plus encore. Enfin, ArgoUML offre des fichiers de sauvegarde ouverts basés sur du XML, et peut exporter les différents diagrammes aux formats GIF, PNG, PostScript, Encapsulated PS, PGML et SVG.

3.4. Conception

Pour une démarche cohérente, nous allons commencer la conception de la solution avec le langage UML.

3.5. Diagramme du cas d'utilisation

Bien souvent, la maîtrise d'ouvrage et les utilisateurs ne sont pas des informaticiens. Il leur faut donc un moyen simple d'exprimer leurs besoins. C'est précisément le rôle des diagrammes de cas d'utilisation qui permettent de recueillir, d'analyser et d'organiser les besoins, et de recenser les grandes fonctionnalités d'un système.

Il s'agit donc de la première étape UML d'analyse d'un système. Un diagramme de cas d'utilisation capture le comportement d'un système, d'un sous-système, d'une classe ou d'un composant tel qu'un utilisateur extérieur le voit. Il scinde la fonctionnalité du système en unités cohérentes, les cas d'utilisation, ayant un sens pour les acteurs. Les cas d'utilisation permettent d'exprimer le besoin des utilisateurs d'un système, ils sont donc une vision orientée utilisateur de ce besoin au contraire d'une vision informatique.

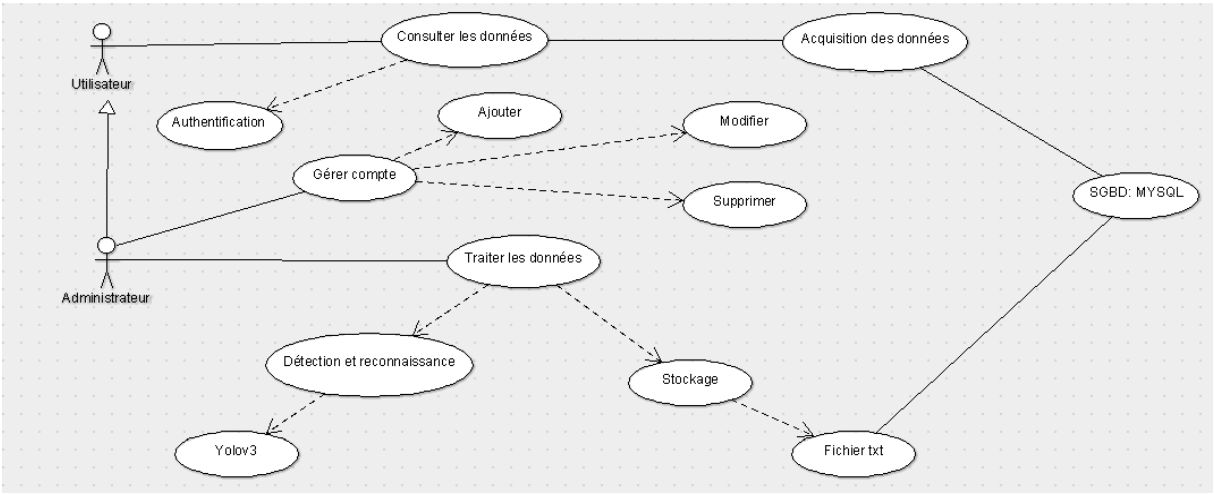


Figure 23 : diagramme de cas d'utilisation.

3.6. Description des cas d'utilisation

La description des cas d'utilisation est libre. Cependant, cette description prend souvent une forme rédigée qui convient mieux à la communication avec l'utilisateur. Des règles de structuration doivent être appliquées pour en faciliter l'expression, la compréhension et la cohérence. Pour exprimer les cas d'utilisation de notre système, nous avons choisi le formalise suivant.

Tableau 5: le formalise de la description des cas d'utilisation.

Numéro du cas d'utilisation	Nom du cas d'utilisation
Résumé	But du cas d'utilisation
Acteurs	Acteurs participants au cas d'utilisation
Scénario	Séquence d'actions normales associées au cas d'utilisation
Alternative	Exception séquence d'actions alternatives pouvant conduire également à un succès

Tableau 6: la description du cas d'utilisation d'authentification.

Numéro du cas	N°1 Authentification
Résumé	Vérification de l'identité des acteurs
Acteurs	Administrateur, Utilisateur
Scénario	[Début] 1. Lancement de l'application 2. Le système affiche la fenêtre d'authentification ; 3. L'utilisateur introduit son login et son mot de passe ; 4. Le système vérifie la conformité des informations introduites ; 5. Le système donne l'accès à l'interface d'accueil. [Fin] Si les informations saisies sont incorrectes, le système réaffiche le formulaire avec le message d'erreur.

Tableau 7: description du cas d'utilisation consultation des données.

Numéro du cas	N°2 Consultation des données
Résumé	Consulter les données
Acteurs	Administrateur, Utilisateur
Scénario	[Début] 1. Lancement de l'application par l'administrateur ou utilisateur 2. L'application vérifie les données d'authentification; 3. L'utilisateur introduit son login et son mot de passe ; 4. L'administrateur ou l'utilisateur consulte les données ; [Fin]
Alternative exception	Si la connexion internet est interrompue l'application demande une connexion internet est requise. Si les données d'authentification sont incorrectes, l'application indique que les informations saisies sont incorrectes.

Tableau 8: description du cas d'utilisation traitement des données

Numéro du cas	N°3 Traitement des données
Résumé	Détecte et reconnaitre les objets qui sont dans les images envoyées par la caméra Ip
Acteurs	Administrateur, Utilisateur
Scénario	[Début] 1. L'administrateur ou l'utilisateur lance l'appli qui charge les images pour la détection et reconnaissance des objets. 2. L'ensemble des objets détectés sur les images seront sauvegardé dans un fichier txt 3. Le fichier txt doit être enregistré dans la base de données ; 4. L'application fera la lecture et affichera les données du fichier. [Fin]

Tableau 9: description du cas d'utilisation de la gestion des comptes.

Numéro du cas	N°4 Gestion des comptes
Résumé	L'administrateur a la possibilité de gérer les comptes des utilisateurs : 1. Ajouter 2. Réinitialiser un compte 3. Mettre à jour les informations des utilisateurs du système. 4. Supprimer un compte d'utilisateur.
Acteurs	Administrateur
Scénario	[Début] 1. L'administrateur lance l'application 2. La page d'authentification est affichée. 3. L'administrateur s'authentifie au système. 4. L'application affiche la page d'accueil qui contient les fonctionnalités suivantes : - Liste des utilisateurs - Ajouter un nouvel utilisateur - Modifier un compte d'utilisateur ; - Supprimer un compte ; [Fin]
Alternative exception	Si les informations de l'identification sont incorrectes : - L'application indique que les informations saisies sont incorrectes. - Le scénario reprend. - Un message de confirmation s'affiche avant que l'application exécute une opération critique effectuée par l'administrateur

3.7. Diagramme de séquence

Un diagramme de séquence est un diagramme d'interaction dont le but est de décrire comment les objets collaborent au cours du temps et quelles responsabilités ils assument. Il décrit un scénario d'un cas d'utilisation.

- Diagramme de séquence : authentification

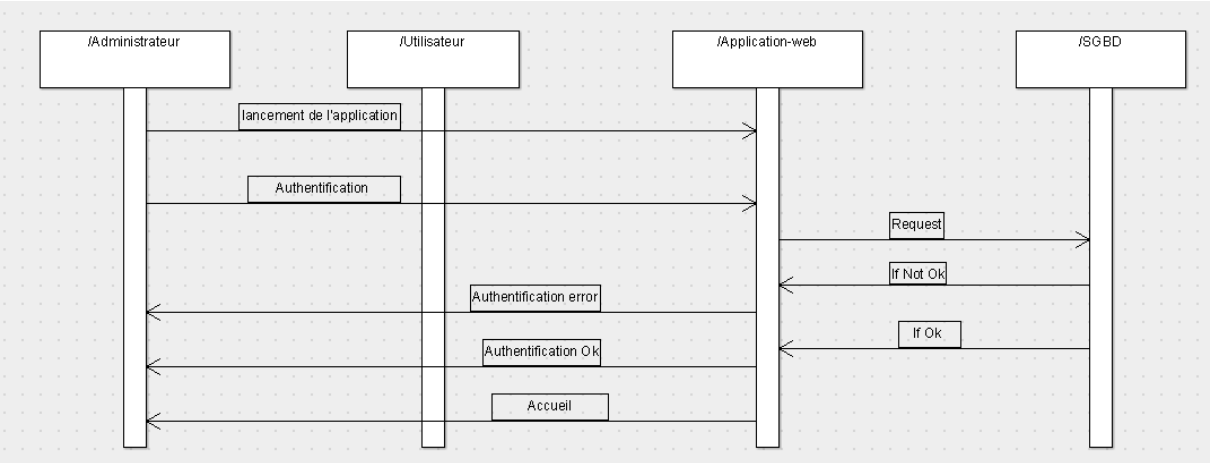


Figure 24 : le diagramme de séquence pour l'authentification

- Diagramme de séquence : consulter les données

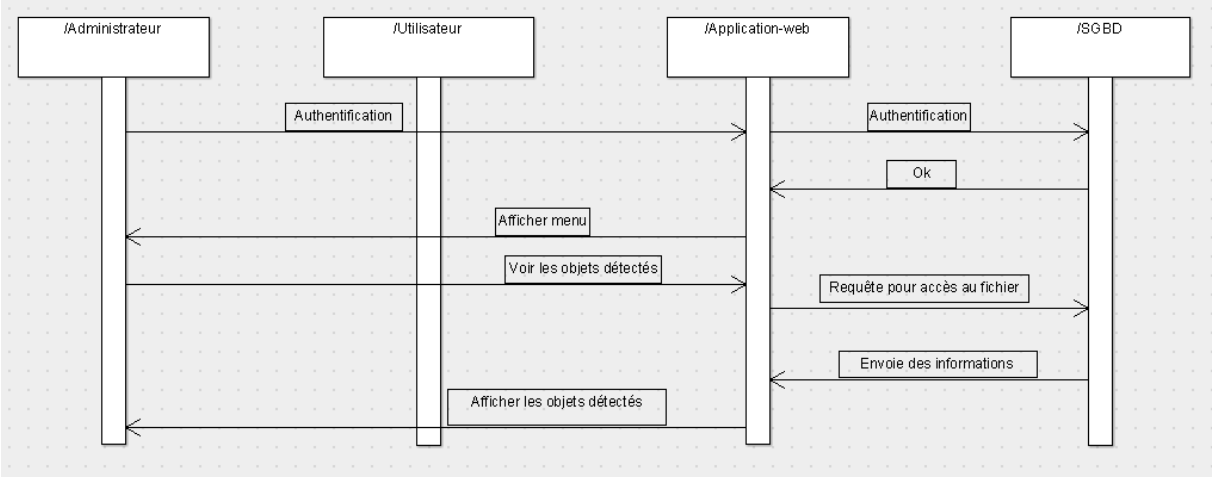


Figure 25 : diagramme de séquence de consultation des données.

- Diagramme de séquence : gérer les comptes

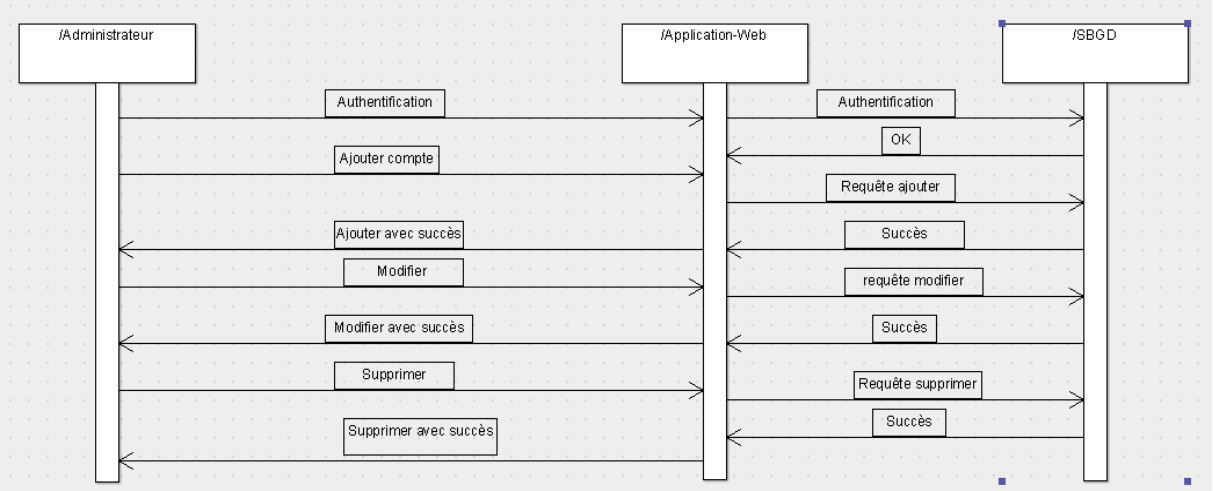


Figure 26: diagramme de séquence de la gestion des comptes par l'administrateur

3.8. Diagramme de classe

Le diagramme de classe est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation. Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne.

Il permet de fournir une représentation abstraite des objets du système qui vont interagir ensemble pour réaliser les cas d'utilisation. Il s'agit d'une vue statique car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application.

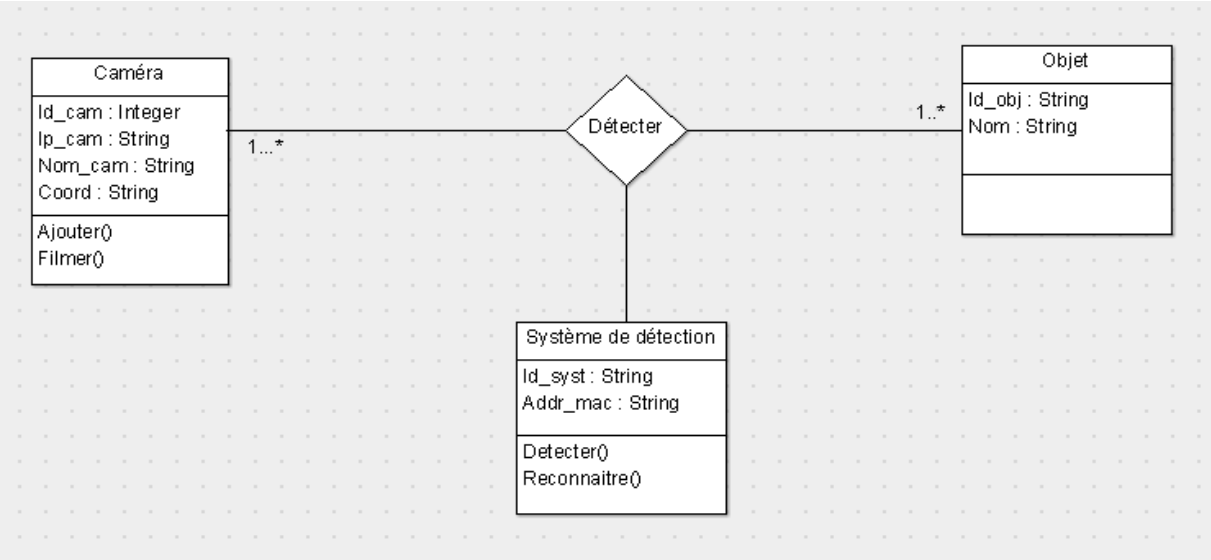


Figure 27: diagramme de classe

4. Architecture de la solution

L'architecture en informatique peut être considérée comme un modèle pour le placement optimal et conforme aux objectifs dans l'environnement informatique.

L'architecture constitue l'organisation fondamentale d'un système situé à l'intérieur de ses composants, ses relations les unes avec les autres, l'environnement et les principes qui régissent sa conception et son évolution.

Un plan informatique est représenté par l'ensemble constituant des plateformes, des réseaux, des applications et tous les autres composants de l'organisation intervenant dans le traitement de l'information.

4.4. Architecture matérielle proposée

L'architecture matérielle décrit l'agencement interne des composants électroniques ainsi que leurs interactions.

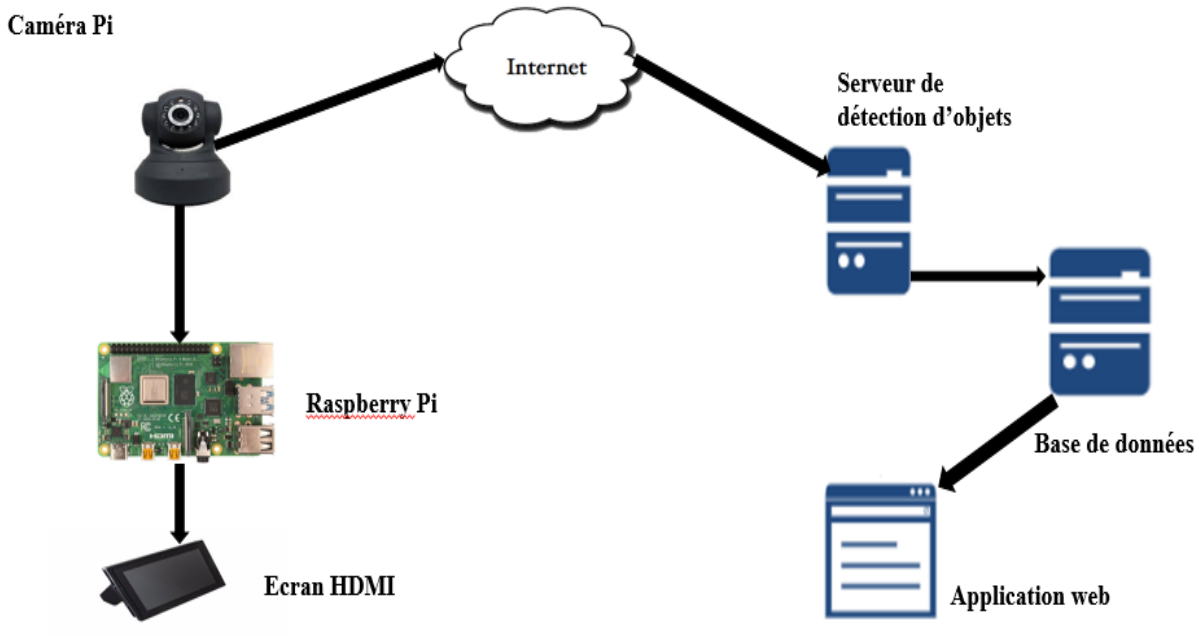


Figure 28: architecture de la solution proposée

4.5. Architecture fonctionnelle

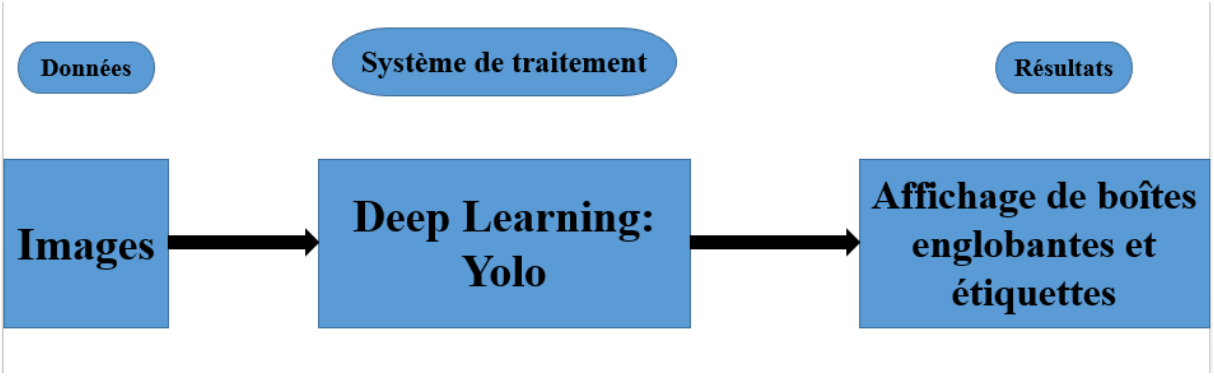


Figure 29 : architecture fonctionnelle du système

- Choix de la caméra

La précision de reconnaissance d'objets dans les images dépend fortement de la qualité de la vidéo d'entrée. Les facteurs les plus importants affectant la précision sont le placement de la caméra et la qualité de la vidéo. La caméra que nous voulons utiliser doit être capable de filmer, d'enregistrer des captures et les envoyés via internet. Pour obtenir les performances les plus élevées possibles pour notre système, nous devons optimiser les variables suivantes :

- Eclairage ;
- Angle de vue ;
- Pixels sur la cible

- Eclairage

Un éclairage adéquat est essentiel pour capturer une image claire. Cela nécessite parfois l'installation d'un illuminateur externe. La quantité d'éclairage requise pour capturer des objets se déplaçant rapidement (sans l'aide de la lumière du soleil) est souvent plus que ce à quoi vous pouvez vous attendre. Les caméras ajustent automatiquement leur vitesse d'obturation en fonction de la quantité de lumière disponible. À midi, par une journée claire et ensoleillée, l'obturateur ne peut s'ouvrir que pour 1/10000^{e} de seconde pour capturer une image vidéo. Parce qu'il y aurait tellement de lumière dans l'environnement, si l'obturateur s'ouvrait plus longtemps, les photos seront surexposées. Par une nuit sombre, l'obturateur peut rester ouvert pendant une seconde entière pour attirer suffisamment de lumière. La vitesse d'obturation est importante car elle est directement corrélée à la netteté d'un objet comme un véhicule en mouvement. Malheureusement, une simple augmentation de la vitesse d'obturation n'améliorera pas les résultats. Si trop peu de lumière est disponible, l'image peut apparaître complétement noire. Au lieu de cela, vous devez augmenter l'éclairage. Si la caméra est capable de jour/nuit et possède un filtre anti-IR (pour filtrer la lumière « couleur » visible), vous pouvez pointer un illuminateur infrarouge externe en direction de certains objets comme les véhicules.

- Angle de vue

L'angle de vue est aussi un critère à ne surtout pas négliger pour bien choisir la position de la caméra. Elle définit le champ de vision de la caméra embarquée. La majorité des caméras installées sur les véhicules ont des objectifs grands-angles. Une grande partie de la route se trouve donc dans l'objectif de la caméra. Pour être plus concret, un angle de vision de 140° offre un champ de vision assez large. Tout ce qui se passe en avant et légèrement sur le côté du véhicule n'échappe pas aux caméras. La référence avec cette valeur est de 160°, c'est la moyenne. Cet angle varie selon les caméras installées mais la fourchette est de 120° et 180°. Il en existe quelques-unes qui atteignent un angle visuel de 190°. Il est bon de souligner qu'un angle de vision important n'est pas toujours mieux. Plus la valeur de l'angle augmente, mais bonne sera la qualité des images. Avec une résolution de 1920p, la qualité est moins bonne pour un angle visuel de 180° que pour un 140°.

- Pixels sur la cible

Le nombre de pixels capturés pour chaque objet est plus important que la distance la distance entre la caméra et l'objet. Le réglage de la résolution de votre appareil peut également les pixels de chaque objet. Cependant, l'augmentation du nombre de pixels augmentera également le temps de traitement. Par conséquent, si les ressources CPU ne sont limitées, une augmentation excessive de la résolution peut diminuer la précision.

- ✓ Caméra proposé

Nous allons énumérer quelques exemples de caméras pouvant être à la hauteur de nos suggestions élaborées précédemment.

Tableau 10: comparaison des caméras AXIS





Caméra IP	AXIS Q1615 Mk II	AXIS Q1615-E Mk II	AXIS Q1645-LE
Max vidéo résolution H.264	1920x1080	1920x1080	1920x1080
Nombre d'images par seconde max.	Entre 100 et 120	120	120
Taille du capteur d'image	1 /2.8	1/2	1/2
Fonction jour/nuit	Oui	Oui	Oui
Stockage local (emplacement pour une carte mémoire)	Oui	Oui	Oui
Prix (F CFA)	35222.23	4578889.95	52833.35

Notre choix personnel pour ces caméras est la caméra AXIS Q1615-E Mk II parce qu'elle combine les en même temps un coût acceptable et une très bonne qualité.

- Serveur de détection et reconnaissance d'objet

Tableau 11: caractéristiques du serveur de détection et reconnaissance d'objets

Caractéristiques	Ordinateur
Système d'exploitation	Windows 10 Professionnel
Architecture	64 bits
Processeur	Intel(R) i7-3520 CPU@2.90GHz

Ram	8 Go

- Serveur de base de données

Nous allons proposer dans cette section les caractéristiques du serveur de base de données qui va stocker la base de données et les flux vidéo venant de la caméra. Ce serveur permettra à l'administrateur d'informer les types d'objets les plus fréquemment enregistré d'un itinéraire à un autre. Ces informations peuvent être aussi utilisées comme preuves matérielles lors d'une poursuite judiciaire contre les actes de vandalisme ou de revoir les scènes d'un accident.

Tableau 12: caractéristiques du serveur de base de données

Caractéristiques	Ordinateur
Système d'exploitation	Windows 10 Professionnel
Architecture	64 bits
Processeur	Intel(R) i7-3520 CPU@2.90GHz
Ram	8 Go

- Caractéristique de la carte Raspberry Pi 3

- un CPU 64 bit **quad core** ARM Cortex-A53 intégré et cadencé à **1,4 GHz ;**
- un contrôleur graphique Broadcom Videocore IV ;
- 1 Go de mémoire vive pour assurer la fluidité de votre système ;
- 1 lecteur micro SD/SDHC ;
- 4 sorties USB 2.0 ;
- 1 port RJ45 (Ethernet 10/100/300 Mbps) ;
- 1 port HDMI ;
- 1 audio Jack 3,5 mm ;
- GPIO 40 broches + 4 nouvelles broches pour le PoE ;
- Compatible Wifi 802.11 b/g/n/ac double bande 2,4 GHz et 5 GHz et bluetooth 4.2 (Bluetooth Classique et LE) ;
- Une alimentation électrique d'au moins 2,5A est fortement recommandée ;
- Coût financier 32734,45 FCFA ;

- Caractéristique de l’écran reliée à la carte raspberry

- Taille 7 ;
- Matériau de l’écran TFT ;
- Résolution 800x480 ;
- Drive IC : TDB ;
- Mode d’interface : RVB ;
- Taille du module : 164.9 mm *124.27 mm;
- Poids du module : 458 ;
- Tactile : cinq points tactile capacitif ;
- Touch version du firmware : Rev2.1 ;

5. Protocoles utilisés

5.4. Protocoles de streaming vidéo RTP, RTCP et RTSP

Les protocoles de streaming vidéo RTP, RTCP, RTSP sont utilisés pour transmettre la vidéo sous forme de paquets de données sur Internet et d'autres réseaux IP.

Au lieu de stocker de gros fichiers multimédias et de les lire, le multimédia peut être envoyé sur le réseau en flux. Les données sont organisées comme une séquence de paquets de taille appropriée pour la transmission entre les serveurs et les clients. Lorsque l'application lit un paquet, les paquets suivants peuvent déjà être au stade de la décompression ou du démultiplexage.

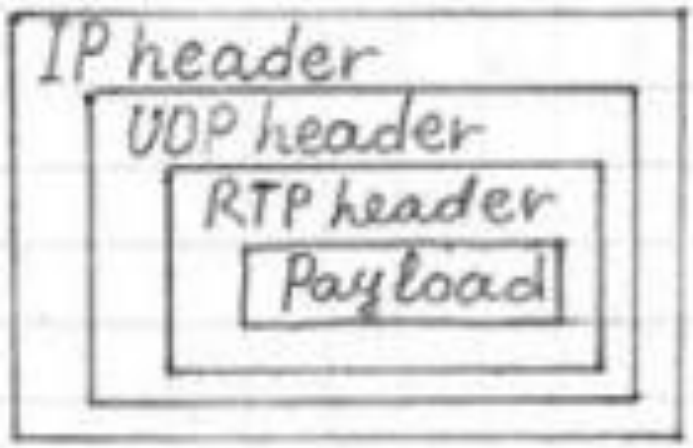


Figure 30: encapsulation des paquets RTP

RTP (Realtime Transport Protocol) est un protocole dédié au transport de flux vidéo et audio en temps réel. IL peut être utilisé pour le transport à sens unique tel que la vidéo à la demande ainsi que pour les services interactifs tels que la téléphonie Internet. RTP fournit des mécanismes pour la reconstruction du temps, la détection des pertes, la

sécurité et l'identification du contenu. RTP appartient à la fois à la couche session (couche 5) et à la couche présentation (couche 6) du modèle OSI.

Le protocole de contrôle RTP (RTCP) est un protocole complémentaire de couche supérieure qui permet de surveiller la livraison des données. Il est conçu pour donner un retour sur la qualité de la transmission des données et des informations sur les participants à la session en cours.

Le protocole RTP est conçu pour être utilisé avec le protocole Transport Layer

UDP / IP. Les paquets RTP et RTCP sont encapsulés dans des paquets UDP / IP.

5.5. Protocole UDP

Le User Datagram Protocol, abrégé en UDP, est un protocole permettant l'envoi sans connexion de datagrammes dans des réseaux basés sur le protocole IP. Afin d'atteindre les services souhaités sur les hôtes de destination, le protocole utilise des ports qui constituent un élément essentiel de l'entête UDP. À l'instar de nombreux autres protocoles de réseau, l'UDP fait partie de la suite des protocoles Internet. Il intervient au niveau de la couche transport et joue ainsi le rôle d'intermédiaire entre la couche réseau et la couche application.

En utilisant le User Datagram Protocol, une application peut donc envoyer très rapidement des informations, étant donné qu'aucune connexion au destinataire n'est établie et qu'aucune réponse ne doit être attendue. En revanche, il n'y a aucune garantie que les paquets arrivent entiers et dans le même ordre que celui dans lequel ils ont été envoyés. Par ailleurs, le protocole n'offre aucune protection contre les manipulations ou accès de tiers. Les paquets erronés peuvent toutefois être identifiés à l'aide d'une somme de contrôle facultative (obligatoire avec IPv6).

5.6. Le protocole http

http est un protocole transactionnel :

- Le client envoie une requête au serveur, généralement à la suite d'une action de l'utilisateur, tel que le "clic" sur un hyper-lien, ou lors du démarrage du client (page d'accueil)
- Le serveur répond à la requête indépendamment des requêtes précédentes et sans rien conserver pour de futures requêtes.

Les requêtes du client peuvent être de différents types (appelés méthodes). La méthode la plus courante est la méthode GET, utilisée par le client pour obtenir le texte d'un document à afficher à l'écran de l'utilisateur

Le client envoie, dans la requête, un certain nombre d'informations, comme par exemple le nom de l'utilisateur, le nom du logiciel client, l'URL du document qui fait référence au document demandé, les types de documents que le client est capable de traiter, etc.

Le serveur répond par une ligne de statut suivie du corps du document. Le statut peut indiquer une condition d'erreur ou une situation particulière, telle la nécessité de fournir un nom d'utilisateur et un mot de passe.

Le client contacte le serveur sur un port particulier (par défaut 80), envoie sa requête en spécifiant la commande HTTP, suivie de l'adresse du document et de la version de HTTP.

6. Mise en œuvre

6.4. Outils de développement

Nous allons présenter une description des outils et les langages de programmation utilisés dans la réalisation de système de détection d'objet dans le transport routier en milieu urbain au Sénégal et notre application web. Dans notre application on a choisi :

Langage de programmation et plateformes de développement :

- PHP : Hypertext Preprocessor est un langage de programmation libre, principalement utilisé pour produire des pages Web dynamiques via un serveur HTTP mais pouvant également fonctionner comme n'importe quel langage interprété de façon locale. PHP est un langage impératif orienté objet.
- Python : Python est le langage de programmation le plus utilisé dans le domaine du Machine Learning, du Big Data et de la Data Science. Créé en 1991, le langage de programmation Python apparu à l'époque comme une façon d'automatiser les éléments les plus ennuyeux de l'écriture de scripts ou de réaliser rapidement des prototypes d'applications. Python est un langage rapide et polyvalent.

6.5. Réalisation de la solution

Nous avons utilisé un modèle préformé c'est-à-dire qui est déjà entraîné de Yolov3 avec 80 classes (pearson, car, bicycle, motorbike, aeroplane, bus, train, truck, boat, traffic light, fire hydrant, stop, sign, parking meter, etc.)



Figure 31: liste des classes de Yolov3

- Etape 1 : Les prérequis
  - Anaconda-Python 3.6 (windows 10) https://www.anaconda.com/download/
  - Conda Env- Yolo.yml

Vous pouvez le télécharger via le lien suivant https://github.com/reigngt09/yolov3workflow/tree/master/2_YoloV3_Execute

Puis dans votre répertoire principal C : cd C:\yolo

conda env create –f yolo.yml

Si cela ne fonctionne pas essayer d'installer les dépendances avec :

pip install –r requirements.txt

Nvidia GPU-GTX 1050 ou plus

- CUDA Toolkit-v9.0 : (Nvidia GPU - GTX 1050 ou supérieur) CUDA et CuDNN peuvent désormais être installés via Anaconda, mais si vous choisissez de les installer en utilisant la méthode ol skool, suivez les liens ci-dessous.

https://developer.nvidia.com/cuda-90-download-archive?target_os=Windows&target_arch=x86_64&target_version=10&target_type=exelocal

- CuDNN-v7.05 peut être télécharger via ce lien https://developer.nvidia.com/rdp/cudnn-archive

- Copiez les fichiers suivants dans le répertoire CUDA Toolkit.

Copiez \ cuda \ bin \ cudnn64_7.dll dans C: \ Program Files \ NVIDIA GPU Computing Toolkit \ CUDA \ v9.0 \ bin.

Copiez \ cuda \ include \ cudnn.h dans C: \ Program Files \ NVIDIA GPU Computing Toolkit \ CUDA \ v9.0 \ include.

Copiez \ cuda \ lib \ x64 \ cudnn.lib dans C: \ Program Files \ NVIDIA GPU Computing Toolkit \ CUDA \ v9.0 \ lib \ x64.

- Ajoutez les chemins suivants aux variables d'environnement C: \ Program Files \ NVIDIA GPU Computing Toolkit \ CUDA \ v9.0 \ bin

C: \ Program Files \ NVIDIA GPU Computing Toolkit \ CUDA \ v9.0 \ include

C: \ Program Files \ NVIDIA GPU Computing Toolkit \ CUDA \ v9.0 \ lib \ x64

- Assurez-vous que les derniers pilotes graphiques nvidia sont installés sur votre PC. Vous pouvez le faire sur le site Web de nvidia.

- Etape 2

Remplacez le répertoire par un lieu de travail où vous souhaitez télécharger le référentiel

- Clone Yolov3 Repo

```
git clone https://github.com/ayooshkathuria/pytorch-yolo-v3.git
```

- Télécharge les poids

https://pjreddie.com/media/files/yolov3.weights

- Etape 3

Charger le répertoire en référentiel cloné :

```
cd C :\ C:\yolo\pytorch-yolo-v3-master\
```

puis active yolov3 avec la commande `conda activate yolov3`

Ainsi vous pouvez charger une vidéo et lancer la démo mais attention OpenCV ne prend que les vidéos de format .avi ou .mp4 avec la commande suivante:

```
python video_demo.py --video video.mp4
```

Test de notre demo sur le webcam de la machine avec la commande suivante :

```
python cam_demo.py
```

Nous allons tester la détection avec un ensemble d'image qui est dans le répertoire **imgs.**

Comme notre solution est destinée au transport routier, la simulation devrait se faire avec une voiture au moins. Par manque de ressources, nous allons charger des images de différents objets susceptibles de rencontrer sur une route ou une autoroute.

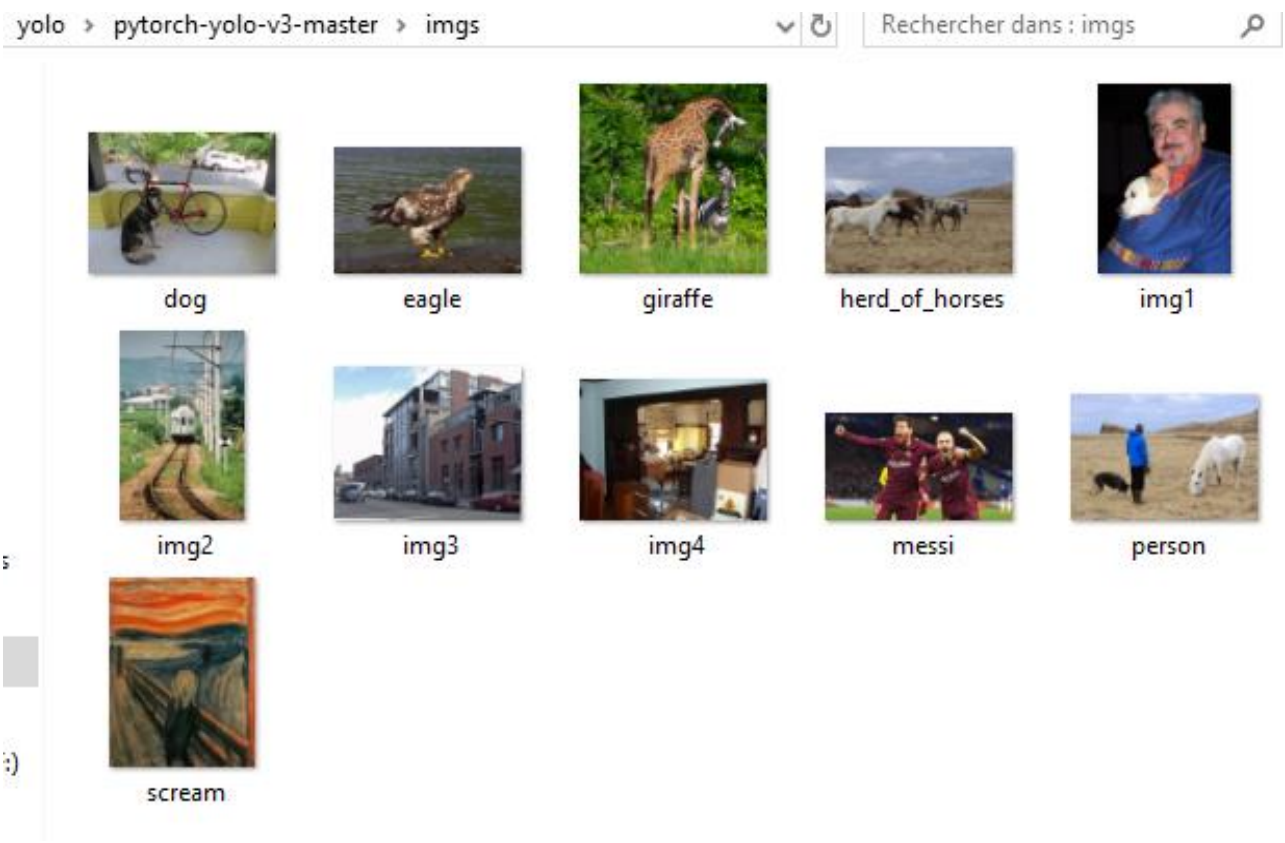


Figure 32 : ensemble des images du répertoire imgs

Avec la commande suivante nous allons détecter les images présentes dans les images du répertoire imgs :

python detect.py - - images imgs - - det

```
(yolov3) C:\yolo\pytorch-yolo-v3-master>python detect.py --images imgs --det det
Loading network.....
Network successfully loaded
C:\Users\BAYE-DJIBY\Anaconda3\envs\yolov3\lib\site-packages\torch\nn\modules\upsampling.
instead.
  warnings.warn("nn.Upsampling is deprecated. Use nn.functional.interpolate instead.")
C:\yolo\pytorch-yolo-v3-master\imgs\dog.jpg predicted in  2.759 seconds
Objects Detected:    bicycle truck dog
----------------------------------------------------------
C:\yolo\pytorch-yolo-v3-master\imgs\eagle.jpg predicted in  2.511 seconds
Objects Detected:    bird
----------------------------------------------------------
C:\yolo\pytorch-yolo-v3-master\imgs\giraffe.jpg predicted in  2.547 seconds
Objects Detected:    zebra giraffe giraffe
----------------------------------------------------------
C:\yolo\pytorch-yolo-v3-master\imgs\herd_of_horses.jpg predicted in  2.563 seconds
Objects Detected:    horse horse horse horse
----------------------------------------------------------
C:\yolo\pytorch-yolo-v3-master\imgs\img1.jpg predicted in  2.479 seconds
Objects Detected:    person dog
----------------------------------------------------------
C:\yolo\pytorch-yolo-v3-master\imgs\img2.jpg predicted in  2.543 seconds
Objects Detected:    train
----------------------------------------------------------
C:\yolo\pytorch-yolo-v3-master\imgs\img3.jpg predicted in  2.515 seconds
Objects Detected:    car car car car car car car truck traffic light
----------------------------------------------------------
C:\yolo\pytorch-yolo-v3-master\imgs\img4.jpg predicted in  2.571 seconds
Objects Detected:    chair chair chair clock
----------------------------------------------------------
C:\yolo\pytorch-yolo-v3-master\imgs\messi.jpg predicted in  2.527 seconds
Objects Detected:    person person person sports ball
----------------------------------------------------------
C:\yolo\pytorch-yolo-v3-master\imgs\person.jpg predicted in  2.539 seconds
Objects Detected:    person dog horse
----------------------------------------------------------
```

Figure 33: Résultats de la sortie de la commande

Conclusion pour le test

A la sortie de la commande nous avons les noms des objets des images qui sont dans le répertoire imgs avec la durée en seconde.

Et les résultats fournis par le détecteur sont exactes.

```
SUMMARY
----------------------------------------------------------------
Task                        : Time Taken (in seconds)

Reading addresses           : 0.004
Loading batch               : 4.723
Detection (11 images)       : 28.504
Output Processing           : 0.000
Drawing Boxes               : 0.096
Average time_per_img        : 3.032
----------------------------------------------------------------
```

Figure 34 : suite des résultats avec un tableau de résumé

Dans la suite nous essayons de récupérer la sortie de cette commande sur un fichier txt qui l'administrateur va afficher à la page de l'utilisateur à la demande et ce fichier doit être dans une base de données. Notre application web sera gérer par un administrateur qui se chargera de la création des comptes des utilisateurs du stockage des images, des vidéos et l'affichage des résultats sur un fichier qu'il fournira aux utilisateurs.



Figure 35: page de connexion sur l'application web

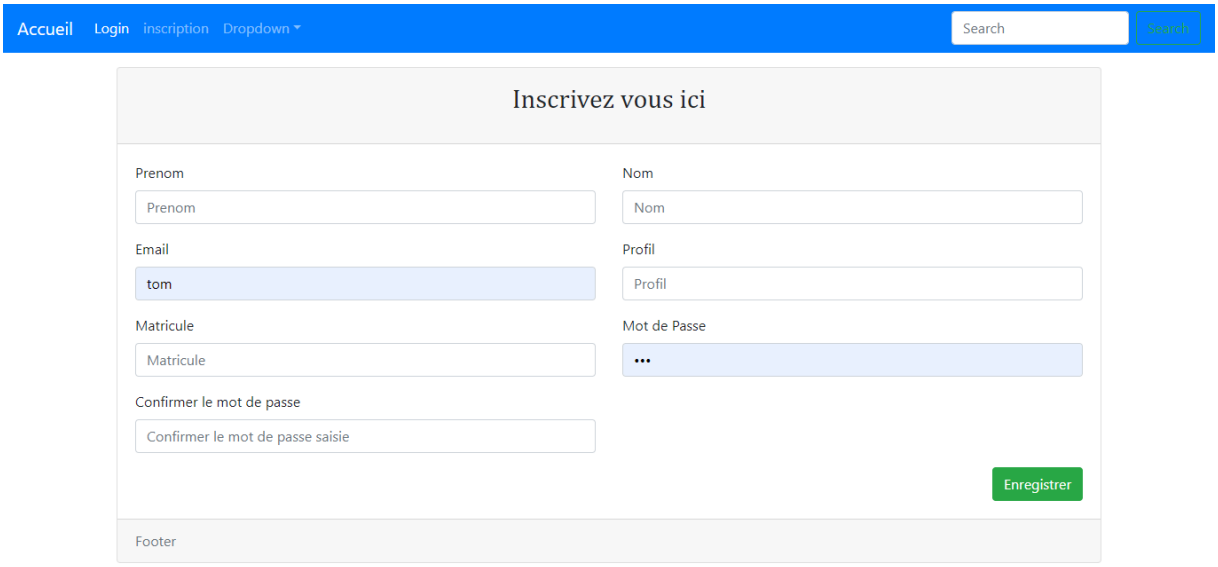


Figure 36: page d'inscription

Conclusion et perspective

Grâce aux Deep Learning, l'avenir de l'intelligence artificielle est prometteur de plus en plus, chaque jour il y a des publications dans ce domaine.

Dans ce travail, nous avons exploré le domaine de la détection des objets dans le trafic routier qui comme tous les autres domaines de l'intelligence artificielle ont connu une évolution majeure depuis l'apparition du Deep Learning.

Afin d'aboutir à ces résultats, nous avons passé beaucoup de temps à lire et à étudier les publications et les articles pour voir ce qui se fait de mieux en matière de détection des objets et pour pouvoir concevoir un système de détection d'objets dans le transport routier.

Pour finir, avant de passer aux perspectives, ce travail nous a permis de mettre en pratique nos connaissances sur les réseaux de neurones et d'en acquérir d'autres et le temps passé à lire des articles nous a servi d'une bonne initiation à la recherche.

Comme perspectives dans le transport routier nous pouvons citer :

- Implémenter un système de détection et reconnaissance des plaques d'immatriculation.
- Implémenter un système de détection des fraudes des feux de circulations.

Références

Bibliographie

[2] Thèse : *Reconnaissance d'objets en vision artificielle : application à la reconnaissance de piétons.* LAETITIA LEYRIT, novembre 2010

[3] Chris Harris and Mike Stephens. A combined corner and edge detector. *In Alvey Vision Conference*, pages 147–151, Manchester - United Kingdom, 31 August 1988.

[4]

[5] L. Deng, D. Yu, et al., "Deep learning: methods and applications," Foundations and Trends R in Signal Processing, vol. 7, no. 3–4, pp. 197–387, 2014.

[6] Y. Bengio et al., "Learning deep architectures for ai," Foundations and trends R in Machine Learning, vol. 2, no. 1, pp. 1–127, 2009.

[7] Y. Bengio, A. Courville, and P. Vincent, "Representation learning : A review and new perspectives," IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 8, pp. 1798–1828, 2013.

[8] J. Schmidhuber, "Deep learning in neural networks : An overview," Neural networks, vol. 61, pp. 85–117, 2015.

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[10] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep machine learning-a new frontier in artificial intelligence research [research frontier]," IEEE Computational Intelligence Magazine, vol. 5, no. 4, pp. 13–18, 2010.

[11] J. Schmidhuber, "Deep learning.," Scholarpedia, vol. 10, no. 11, p. 32832, 2015.

[12] H. Y. Xiong, B. Alipanahi, L. J. Lee, H. Bretschneider, D. Merico, R. K. Yuen, Y. Hua, S. Gueroussov, H. S. Najafabadi, T. R. Hughes, et al., "The human splicing code reveals new insights into the genetic determinants of disease," Science, vol. 347, no. 6218, p. 1254806, 2015.

[13] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al., "Mastering the game of go with deep neural networks and tree search," Nature, vol. 529, no. 7587, pp.

484–489, 2016.

[14] F. Buggenthin, F. Buettner, P. S. Hoppe, M. Endele, M. Kroiss, M. Strasser, M. Schwarzfischer, D. Loeffler, K. D. Kokkaliaris, O. Hilsenbeck, et al., “Prospective identification of hematopoietic lineage choice by deep learning,” Nature Methods, vol. 14, no. 4, pp. 403–406, 2017.

[15] E. Gibney, “Google reveals secret test of ai bot to beat top go players.,” Nature, vol. 541, no. 7636, p. 142, 2017.

[16] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al., “End to end learning for self-driving cars,” arXiv preprintar Xiv :1604.07316, 2016.

[17] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, “Dermatologistlevel classification of skin cancer with deep neural networks,” Nature, vol. 542, no. 7639, pp. 115–118, 2017.

[18] H. C. Hazlett, “Early brain development in infants at high risk for autism spectrum disorder,” in Biological Psychiatry, vol. 73, pp. 115S–115S, ELSEVIER SCIENCE INC 360 PARK AVE SOUTH, NEW YORK, NY 10010-1710 USA, 2013.

[19] L. A. Gatys, A. S. Ecker, and M. Bethge, “A neural algorithm of artistic style,” arXiv preprintar Xiv :1508.06576, 2015.

[20] R. Dechter and J. Pearl, The cycle-cutset method for improving search performance in AI applications. University of California, Computer Science Department, 1986.

[21] I. Aizenberg, N. N. Aizenberg, and J. P. Vandewalle, Multi-Valued and Universal Binary Neurons:Theory, Learning and Applications. Springer Science & Business Media, 2013.

[22] A. Van den Oord, S. Dieleman, and B. Schrauwen, “Deep content-based music recommendation,” in Advances in neural information processing systems, pp. 2643–2651, 2013.

[23] Moualek Djaloul Youcef ‘’Deep Learning pour la classification des images’’ 3 juillet 2017

[24] Moualek Djaloul Youcef ‘’Deep Learning pour la classification des images’’ 3 juillet 2017, pp 30

[25] Bastien L, ‘’Data Analytics, Intelligence artificielle, Deep Learning ou apprentissage

profond : définition, concept, 16 juillet 2018

[26] A. Van den Oord, S. Dieleman, and B. Schrauwen, “Deep content-based music recom-dation’’ in Advances in neural information processing systems, pp. 2643–2651, 2013

[27] R. Collobert and J. Weston, “A unified architecture for natural language processing : Deep neural networks with multitask learning,” in Proceedings of the 25th international conference on Machine learning, pp. 160–167, ACM, 2008.

[28] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, “Subject independent facial expression recognition with robust face detection using a convolutional neural network,” Neural Networks, vol. 16, no. 5, pp. 555–559, 2003.

[30] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Advances in neural information processing systems, pp. 1097–1105, 2012.

[32] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet Large Scale Visual Recognition Challenge,” International Journal of Computer Vision (IJCV), vol. 115, no. 3, pp. 211–252, 2015.

[33] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in European conference on computer vision, pp. 818–833, Springer, 2014.

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–9, 2015.

[35] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning,” arXiv preprint arXiv :1602.07261, 2016.

[36] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2016.

[37] M. C. Mozer, “A focused backpropagation algorithm for temporal,” Backpropagation : Theory, architectures, and applications, p. 137, 1995.

[38] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training recurrent neural networks.,” ICML (3), vol. 28, pp. 1310–1318, 2013.

[39] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[40] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” arXiv preprint arXiv :1406.1078, 2014.

[41] T. Mikolov, M. Karafiát, L. Burget, J. Cernock`y, and S. Khudanpur, “Recurrent neural network based language model.,” in Interspeech, vol. 2, p. 3, 2010.

[42] T. Mikolov, S. Kombrink, L. Burget, J. Černock`y, and S. Khudanpur, “Extensions of recurrent neural network language model,” in Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on, pp. 5528–5531, IEEE, 2011.

[43] I. Sutskever, J. Martens, and G. E. Hinton, “Generating text with recurrent neural networks,” in Proceedings of the 28th International Conference on Machine Learning (ICML-11), pp. 1017–1024, 2011.

[44] S. Liu, N. Yang, M. Li, and M. Zhou, “A recursive recurrent neural network for statistical machine translation,” 2014.

[45] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in Advances in neural information processing systems, pp. 3104–3112, 2014.

[46] M. Auli, M. Galley, C. Quirk, and G. Zweig, “Joint language and translation modeling with recurrent neural networks.,” in EMNLP, vol. 3, p. 0, 2013.

[47] A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks.,” in ICML, vol. 14, pp. 1764–1772, 2014.

[48] A. Karpathy and L. Fei-Fei, “Deep visual-semantic alignments for generating image descriptions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3128–3137, 2015.

[49] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and tell : A neural image caption generator,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3156–3164, 2015.

[50] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, “Long-term recurrent convolutional networks for visual

recognition and description," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2625–2634, 2015.

[51] A. Y. Ng and M. I. Jordan, "On discriminative vs. generative classifiers : A comparison of logistic regression and naive bayes," Advances in neural information processing systems, vol. 2, pp. 841–848, 2002.

[52] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," Cognitive science, vol. 9, no. 1, pp. 147–169, 1985.

[53] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in Proceedings of the 24th international conference on Machine learning, pp. 791–798, ACM, 2007.

[54] G. Hinton, "A practical guide to training restricted boltzmann machines," Momentum, vol. 9, no. 1, p. 926, 2010.

[55] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," Neural computation, vol. 18, no. 7, pp. 1527–1554, 2006.

[56] R. Salakhutdinov and G. Hinton, "Deep boltzmann machines," in Artificial Intelligence and Statistics, pp. 448–455, 2009.

[57] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," in Advances in neural information processing systems, pp. 2222–2230, 2012.

[58] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in Advances in neural information processing systems, pp. 2672–2680, 2014.

[59] R. AUFRÈRE, F. MARMOITON, R. CHAPUIS, F. COLLANGE, J.P. DÉRUTIN, ''Détection de route et suivi de véhicules par vision pour l'ACC''

Webographie

[1] https://machinelearningmastery.com/object-recognition-with-deep-learning/ Le 28/12/2019

[29] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[60] http://www.pixalarm.net/fr/analyse-et-comptage-via-deep-learning/