الجـــمـــهـــوريـة الجـــزائـــريـــة الديمـــقـــراطـيـــة الـشـــعـــبـيـــة

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التـــــــــــــــــــــــعـليـم العـــــالي و البـــــــحث العـــلـمـــي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

جـــــــــــامعة أ أبو بكـــر بـلـقـايد – تـلمســـــــــــان –

Université Aboubakr Belkaïd –Tlemcen –
Faculté de TECHNOLOGIE

MEMOIRE

Présenté pour l'obtention du diplôme de: **MASTER**

En: Télécommunications

Spécialité : Réseaux et télécommunications

Par : KORTI Djazila Souhila

Sujet

Classification des obstacles par la technique d'apprentissage profond

Soutenu le 20/09/2020, devant le jury composé de:

Mr. F.T BENDIMERAD	Professeur	Univ. Tlemcen	Président
Mr. F. DERRAZ	MCA	Univ. Tlemcen	Encadrant
Mr. S.M MERIAH	Professeur	Univ. Tlemcen	Examinateur
Mr. G. ABDELLAOUI	MCB	Univ. Tlemcen	Examinateur

Année Universitaire 2019-2020

Remerciements

En première instance, Je remercie ALLAH le tout puissant et miséricordieux, qui m'a donné la force, la volonté, et surtout la patience de mener à terme le présent travail.

Je remercie mon encadreur Mr. F. DERRAZ qui m'a dévolu tout son temps et son énergie afin de me transmettre son savoir, tout au long de l'élaboration de ce mémoire.

Mes remerciements vont au Professeur T.F BENDIMERAD pour avoir accepté de présider ce jury, ainsi qu'aux membres du jury, Mr.S.M MERIAH, Mr. G. ABDELLAOUI, pour l'intérêt qu'ils ont porté à ma recherche en acceptant d'examiner mon travail et de l'enrichir par leurs propositions.

Mes plus précieux remerciements vont à ma famille qui m'a pleinement soutenu durant toute la période de mes études. Je remercie ma mère et son aimable âme pour le sacrifice qu'elle s'est donnée pour mon réconfort, mon très cher père pour la confiance qu'il m'a accordé et pour sa présence et sa serviabilité aux moments où j'avais toujours besoin de lui, mes chers frères et uniques fidèles amis de les avoir si proches de moi, et à ma très chère grand-mère Mansouria BENYELLES pour ses prières et douâs. Ce mémoire n'aurait **JAMAIS** pu être mené sans leur soutien indéfectible.

Résumé

L’industrie automobile en Europe et dans le monde a investi des ressources conséquentes (considérables) dans le développement de véhicules connectés et autonomes de la génération future. Les progrès réalisés dans le domaine de l'intelligence artificielle, en particulier des "réseaux de neurones profonds" (DNN), ont encouragé le développement et le déploiement des techniques de perception de l'environnement orientées vers l'automobile avec une bonne efficacité.

Ce mémoire présente les travaux menés dans le cadre du domaine de la vision par ordinateur. Notre objectif est de classifier les différents obstacles détectés sur la trajectoire du véhicule, circulant en milieu urbain en traitant les images fournis par la caméra embarquée. Nous tenons à répondre à certaines exigences notamment :

- Développer un système fonctionnant de façon indépendante et en temps réel pour être capable d’agir avant un éventuel impact.

- Obtenir un système capable d’identifier les obstacles quel que soit la complexité de leurs formes et apparences.

Mes travaux consistent à mettre en œuvre une méthode basée sur l’apprentissage profond qui permet de répondre à ces attentes. Le problème se traite en deux phases : dans un premier temps, une modélisation des objets est effectuée en faisant appel à une méthode de segmentation. Cette étape va nous permettre de détecter les objets avec une très grande précision quel que soit la complexité de leurs apparences. Dans un second temps nous avons eu recours à un réseau de neurones profond afin de répondre à la problématique de classification. Notre approche a été validée à travers différentes expérimentations sur des images et des séquences routières dans un milieu urbain.

Mots clés : reconnaissance d’objets, vision par ordinateur, intelligence artificielle, apprentissage profond, détection, classification, temps-réel.

Abstract

The automotive industry around the world and specially Europe has invested significant (considerable) resources in the development of next generation autonomous and connected vehicles. The progress made in the artificial intelligence's field, particularly in the "deep neural networks" (DNN), has encouraged the development and deployment of car-oriented environmental perception techniques with good efficiency.

This master thesis presents the work carried out in the field of computer vision. Our objective is to classify the various obstacles detected on the vehicle's path, circulating in an urban environment by processing the images provided by the onboard camera. We want to answer some expectations including:

- Develop a system operating independently and in real time to be able to act before a possible impact.

- Obtain a system capable of identifying obstacles regardless of the complexity of their forms and appearances.

My work is to implement a method based on deep learning that meets these expectations. The problem is dealt with in two phases: first, a modeling of objects is carried out using a segmentation method. This step will allow us to detect objects with great precision regardless of the complexity of their appearances. Second, we used a network of deep neurons to answer the classification problem. Our approach has been validated through different experiments on images and road sequences in an urban environment.

Keywords: object recognition, computer vision, artificial intelligence, deep learning, detection, classification, real-time.

ملخص

تستثمر صناعة السيارات في أوروبا وحول العالم موارد كبيرة في تطوير الجيل التالي من المركبات المتصلة والمستقلة. شجع التقدم المحرز في مجال الذكاء الاصطناعي، ولا سيما "الشبكات العصبية العميقة"(NND), على تطوير ونشر تقنيات الإدراك البيئي للسيارة بكفاءة جيدة. تقدم هذه المذكرة الأعمال التي تم تنفيذها في مجال الرؤية الحاسوبية. هدفنا هو تصنيف االحواجزالمختلفة المكتشفة على مسار السيارة المتنقلة في بيئة حضرية من خلال معالجة الصور التي توفرها الكاميرا الموجودة على متن المركبة. نود تلبية متطلبات معينة، على وجه الخصوص:

- تطوير نظام يعمل بشكل مستقل وفي الوقت الحقيقي ليكون قادرًا على التصرف قبل حدوث تأثير محتمل.

- الحصول على نظام قادر على تحديد طبيعة الحواجز بغض النظر عن مدى تعقيد أشكالها ومظاهرها.

يقوم عملي على تنفيذ طريقة تعتمد على التعلم العميق الذي يجعل من الممكن تلبية هذه التوقعات. يتم التعامل مع المشكلة على مرحلتين: أولاً، يتم نمذجة الحواجز من خلال استدعاء طريقة التجزئة. ستسمح لنا هذه الخطوة باكتشاف الأشياء بدقة كبيرة، بغض النظر عن مدى تعقيد مظهرها. ثانيًا، استخدمنا شبكة عصبية عميقة للرد على مشكلة التصنيف. تم التحقق من صحة نهجنا من خلال تجارب مختلفة على الصور وتسلسل الطرق في بيئة حضرية.

الكلمات المفتاحية: التعرف على الأشياء، الرؤية عبر الكمبيوتر، الذكاء الاصطناعي، التعلم العميق، الكشف، التصنيف، الوقت الحقيقي.

Table des matières

Listes des figures

Liste des tableaux

Liste des acronymes

1D	Une dimension
2D	Deux dimensions
BN	Batch Normalization
CNN	Convolutional neural network
ConvNets	Convolutional network
DeconvNet	Réseau déconvolutif multicouche
DL	Deep learning
DNN	Deep neural network
ELU	Exponential Linear Unit
FC	Fully connected
FFNN	Feed-Forward Neural Network
FPN	Feature Pyramid Network
HD	Haute définition
HOG	Histogram of oriented gradients
IA	Intelligence artificielle
IoU	Intersection over union
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
KNN	K nearest neighbour
LoG	Laplacian of gaussian
LPE	La ligne de partage des eaux
LSE	Level-set evolution
ML	Machine learning

MLP	Multi-layer Perceptron
NMS	Non-max suppression
OSM	OpenStreetMap
PCA-SIFT	Principal Component Analysis- Scale-invariant feature transform
PMC	Perceptron Multi Couches
ReLU	L'unité linéaire rectifiée
ResNet	Residual Neural Network
RNA	Réseaux de Neurones Artificiels
RNN	Recurrent Neural Network
SIFT	Scale-invariant feature transform
SURF	Speeded up robust features
SAE	Society of Automotive Engineers
SDF	Fonction de distance signée
SSE	Somme des erreurs carré
SVM	Support vector machine
Tanh	Hyperbolic tangent.
TRI-AD	Toyota Research Institute-Advanced Development
VGG	Visual geometry group network
YOLO	You only look once
YOLO v3	You only look once version 3

Introduction générale

Contexte :

Le véhicule automobile s'impose comme le principal moyen de déplacement dans la majorité des pays du monde. L'utilisation massive de ce dernier contribue à l'augmentation sans cesse croissante du nombre d'accidents de la route et de morts. D'après des recherches focalisées sur l'analyse des causes d'accidents [1], le facteur humain apparait dans plus de 90% des accidents.

De ce fait les constructeurs de l'automobile ont déplacé leur champ d'intérêt sur la conception et le développement de véhicules autonomes fonctionnant de façon indépendante, mais mise en relation avec l'environnement contribuant à l'amélioration de la sécurité routière, l'efficacité du trafic, et permettant au conducteur de bénéficier d'une assistance accrue d'aide à la conduite.

Les constructeurs d'automobile sont de plus en plus demandeurs de systèmes d'assistance de haut niveau et surtout intelligents. L'intégration d'un module de détection et d'identification d'obstacles permettra de fournir en temps réel des informations liées à l'environnement proche du véhicule à partir des capteurs embarqués, comme les bords de route, les obstacles routiers, la distance d'éloignement, etc… dont le but d'éviter toute collision avec n'importe quel type d'obstacle. Le système pourra intervenir dans des situations à risques en utilisant différents déclencheurs tels que l'assistance au freinage ou le freinage autonome.

L'intégration d'un tel module permettra au véhicule d'acquérir et de traiter des informations visuelles, dans le but de comprendre l'environnement dans lequel il évolue et de réagir à d'éventuels changements dans celui-ci.

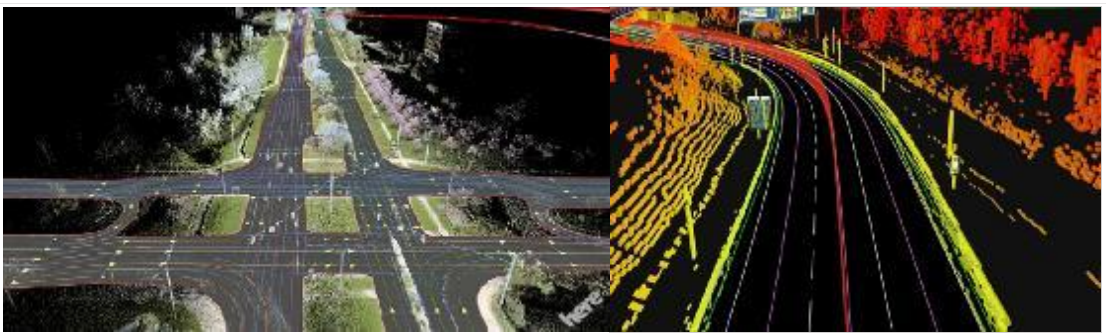
La course à la voiture à conduite totalement autonome, ou autonomie de niveau 5 selon la classification de SAE Internationale, a permis des avancés remarquables dans le domaine de l'apprentissage profond. Aujourd'hui un niveau d'autonomie respectable est atteint lors de la conduite sur des routes, avec des scénarios de conduite classique.

Problématique et objectifs de recherche :

La navigation en toute autonomie dans un environnement complexe et dynamique implique un certain nombre d'exigences auxquelles le véhicule doit faire face. Il est nécessaire de garantir la sécurité du véhicule et des humains qui l'entourent.

Le véhicule autonome doit avoir une connaissance sur l'environnement où il évolue ainsi que sa position dont le but de planifier des actions pour une prise de décision. A partir des objets détectés et identifiés sur sa trajectoire le véhicule est en mesure de crée une carte haute

définition (HD) en plaçant les éléments observés sur cette carte. Cette dernière permet au véhicule d'établir un chemin, définir une destination, et se localiser (Figure 1).



a) HERE b) TomTom

Figure 1 : Exemple de carte HD [2].

La détection et l'identification du véhicule aux différents obstacles observés sur sa trajectoire à partir d'une caméra embarquée est une tâche assez complexe. Des difficultés pouvant être rencontrées sont en général :

- Les images fournies par la caméra peuvent être bruitées à cause de la qualité du capteur.
- Les objets urbains de par leurs localisations, sont soumis à des occultations partielles, et à la présence d'ombres.
- Les objets urbains possèdent de nombreuses représentations au sens géométriques et textuel, ce qui les rend difficilement identifiable (variation inter-classe).
- Les objets urbains sont multiples et peuvent être collés les uns aux autres.
- Les conditions d'éclairage variables, rend la phase de détection difficile. Les changements soudains d'intensité lumineuse seront gérés différemment par le système de détection.
- Les objets urbains peuvent être pris à partir de différents angles de vue.

L'objectif de notre travail est de proposer un module de détection et d'identification d'obstacle routiers pour véhicule autonome adapté aux environnement urbain et capable d'améliorer le processus de recherche automatique d'objets d'intérêts pour les applications temps-réel. Nous plaçons cette problématique dans un cadre urbain dans lequel les images prises en compte sont acquises dans un environnement extérieur, dynamique et complexe pour la reconnaissance des différents objets détectés sur la trajectoire du véhicule.

Dans ce travail, nous nous intéressons particulièrement aux réseaux neuronaux formés pour exceller dans les tâches liées à la détection et à la segmentation des objets sur des images ou des flux vidéo.

Une première étape consiste à rechercher d'éventuelles zones d'intérêts sur les images fournies par la caméra embarquée, susceptibles de contenir des objets dont le but de les localiser avec précision avant d'être en mesure de les classifier.

D'un simple seuillage des niveaux de gris, aux techniques plus complexes comme les contours actifs, nous nous sommes intéressé a l'une d'elles parmi les plus robustes, précises et rapides : la technique d'ensemble de niveaux «Level-set»[3], qui fournit des objets segmentés susceptibles d'être utilisés comme support pour guider la voiture et la rendre consciente des obstacles situés devant elle. La méthode «Level-set » utilise ce que l'on appelle les contours actifs. Elle englobe l'objet désiré avec une courbe qui évolue d'une position initiale vers une position finale dans le temps, jusqu'à l'obtention d'une courbe bien alignée aux contours de l'objet d'intérêt. Reconnue par sa robustesse, la méthode «Level-set » utilisée dans notre travail, permet la segmentation d'objets non rigides s'adaptant à des formes complexes et des changements topologiques tels que la fusion et la division, ce qui permet de traiter les objets occlus.

Une deuxième étape consiste à analyser et interpréter les zones segmentées pour essayer de reconnaitre de quel objet s'agit-il. L'analyse de l'image consiste à extraire les informations contenues dans divers objets de la scène afin de développer une représentation symbolique puis sémantique de l'information. Quant à l'interprétation elle consiste à déterminer des caractéristiques de haut niveau d'objets à partir de celles extraites lors de l'analyse. Son objectif est d'identifier la nature des objets.

Il existe deux types de méthodes d'extraction :

1) Les méthodes basées sur les caractéristiques calculées manuellement qui nécessitent leur extraction au préalable.
2) Les méthodes basées sur l'apprentissage profond.

Dans le cadre de notre travail, nous avons proposé d'utiliser des méthodes basées sur l'apprentissage profond (Deep learning), qui consiste à fournir au réseau de neurones convolutif (CNN) les résultats obtenus de la phase de segmentation. Le CNN permet d'extraire automatiquement les caractéristiques nécessaires et faire la correspondance avec celles des

classes connues pour prédire la nature de l'objet. L'approche d'apprentissage en profondeur présente de multiples avantages qui tournent principalement autour de l'aspect temps réel, et les quantités phénoménales de données à traiter.

Contribution :

Notre contribution consiste à ajouter une étape de représentation et de modélisation des obstacles de façon rapide et très précise dont le but de pouvoir les analyser et les traiter d'une façon similaire à la vision humaine. Cette étape va non seulement permettre à notre système de détecter les objets quel que soit la complexité de leurs formes, mais aussi de traiter le problème d'occultation.

Notre approche s'appuie sur la combinaison de la méthode «Level-set» [3] avec le réseau neuronal YOLO v3 [83]. La première étape consiste à segmenter les différents objets fournis à partir des images captées par la caméra monoculaire embraquée sur le véhicule. Ensuite fournir le résultat de segmentation au YOLO v3 qui va détecter, localiser et classifier les différents objets simultanément. Notre approche permet un traitement d'images rapide ce qui convient le mieux aux applications temps-réel.

Description de l'organisation du manuscrit :

Notre travail sera donc divisé en trois parties :

- chapitre 1 : Ce chapitre est présenté en deux parties. Nous exposons dans la partie I des généralités sur les techniques de détection et les algorithmes de classification utilisés antérieurement. Leurs limites nous ont poussés à les abandonner pour des applications telles que la reconnaissance d'objets. La partie II présente une vue d'ensemble sur les cartes HD ainsi que leurs utilités pour les véhicules autonomes.

- chapitre 2 : Dans ce chapitre nous présentons les réseaux de neurones, ainsi que leurs avantages qui nous ont incités à les choisir comme modèle pour notre application. Nous détaillons les différents éléments de construction ainsi que les opérateurs utilisés. Nous expliquons comment entraîner un réseau en utilisant un algorithme de descente de gradient. Finalement, nous détaillons les différentes architectures qui ont été utilisées au fil des années en expliquant les évolutions et améliorations qu'elles ont entraînées.

- Chapite3 : Dans ce chapitre, nous avons introduit la méthodologie à suivre pour le développement d'un détecteur/ identificateur d'objet à partir d'un réseau pré-entrainé. Ce dernier servira de détecteur d'obstacles pour un nombre bien défini de classes d'objets. Dans une deuxième étape, pour plus de précision nous avons combiné la détection par le réseau YOLO v3 avec la segmentation par « Level-set ». Dans une fenêtre englonbante le réseau YOLO v3 cherche à prédire un objet que la « Level-set » tente de contourner pour définir la notion d'un objet sémantique.

Chapitre I :

Rappel sur les techniques de détection, classification et cartographie.

I.1 Introduction :

Le déploiement des véhicules autonomes sur nos routes est bientôt réalisable grâce à l'évolution des technologies remarquables dans le domaine de l'intelligence artificielle (IA).

Pour atteindre un haut niveau de sécurité souhaitée des véhicules autonomes, la détection et l'identification des objets observés sur la trajectoire du véhicule doivent être assurées. Au moyen des capteurs embarqués sur l'automobile, le système de navigation analyse la scène captée et doit être capable d'extraire l'ensemble des objets d'intérêt qui la compose. Le processus d'interprétation fait intervenir plusieurs outils issus de la vision par ordinateur, dont les plus fondamentaux concernent la segmentation et la classification d'objets, qui peuvent analyser et extraire des objets de la scène pout générer une description interprétant la scène étudiée. La segmentation permet le passage d'une image de pixels à une image de régions, en rassemblant les pixels/blocs en fonction de certains critères de similarité prédéfinis permettant ainsi d'obtenir des zones homogènes, rendant l'image simple, exploitable et plus facile à analyser et interpréter. Les régions, obtenues suite à la décomposition, sont étiquetées selon un nombre prédéfini de classes. Chaque région est caractérisée par des descripteurs physiques. L'étape de la classification permet de caractériser les régions par leurs informations sémantiques selon les différentes classes précédemment définies.

Tenant compte de la complexité de l'environnement externe, pour qu'un véhicule autonome se déplace de manière sécurisée il doit savoir à quoi ressemble l'environnement qui l'entoure, dont le but de se localiser mais également de trouver sa trajectoire optimale, ce qui donne lieu à une solution sous forme de cartographie. Les cartes haute définition (HD) facilitent le processus de navigation et ajoutent une couche supplémentaire de sécurité et de compréhension. Le véhicule utilise ses capteurs embarqués pour comparer ce qu'il perçoit à un moment donné avec ce qui est stocké dans sa mémoire.

Le chapitre I est présenté en deux parties. La première partie est un rappel sur les différentes techniques utilisées pour la détection et la classification des objets. La seconde partie nous introduisons la cartographie haute définition ainsi que sa nécessité pour la navigation autonome.

Partie I : Rappel sur les techniques de détection et de classification

I.2 La détection :

Les objets sont définis comme étant des régions fermées contenant des informations et des caractéristiques qui doivent être extraites afin d'identifier leur nature. Les techniques de détection sont considérées comme un prétraitement des images qui permettent de décider quelles parties des images (pixels ou régions) correspondent à des objets d'intérêt dont le but de réduire la quantité d'information à analyser et à traiter. La section qui suit présente un bref aperçu des méthodes les plus couramment utilisées et les différents types d'approches existantes pour la détection.

I.2.1 La segmentation :

La segmentation est un processus utilisé pour l'analyse et la compréhension des images, par exemple pour la reconnaissance d'objets, le diagnostic médical [4], la détection et la localisation d'objets dans des scènes vidéo [5], etc... L'objectif principal de cette tâche est d'extraire les régions d'intérêt en divisant l'image initiale en une ou plusieurs zones où l'objet est susceptible de se trouver. Le partitionnement de l'image vise à regrouper des pixels en respectant une certaine homogénéité selon un critère défini au préalable (intensité de pixel, la couleur, la texture, etc…). Cette étape de prétraitement, permet de changer la représentation d'une image en quelque chose de plus significatif et plus facile à analyser, ainsi que limiter le temps de calcul qui suit puisque seules les zones conservées seront traitées.

Figure I. 1: Segmentation d'image naturelle [6].

Les régions d'intérêt doivent respecter un ensemble de critères pour représenter le contenu de l'image. Dans l'article [7], les auteurs ont formulé le problème de la segmentation comme suit, considérons :

- ∏ un ensemble de régions R de l'image I.
- R_i une région de la partition ∏.
- N le nombre de régions de la partition ∏.
- P (R_i) un prédicat qui permet de vérifier l'homogénéité de la région R_i.

Une partition Π doit vérifier les propriétés suivantes :

1. ∏ = ∪R_i , i ∈ {1...N} : une partition ∏ doit être complète.

2. ∀ (i, j) $R_i \cap R_j = \emptyset$: les régions doivent être disjointes.

3. ∀i $R_i = \emptyset$.

4. ∀i P (R_i) = vrai : toutes régions doivent satisfaire le critère de similarité.

5. ∀i P($R_i \cup R_j$) = faux si R_i et R_j sont adjacentes.

Ainsi, il s'agit de s'assurer qu'il n'existe pas deux régions adjacentes dont l'union peut satisfaire le critère de similarité.

I.2.1.1 Les techniques de segmentation pour la détection :

De nombreuses méthodes de segmentation ont été proposées, mais aucune ne peut être considérée comme meilleure que d'autres. Le choix d'une technique de segmentation par rapport à une autre est déterminé par le type particulier d'image à traiter et les caractéristiques du problème considéré. Cependant les approches de segmentation d'image sont actuellement regrouper en deux classes principales :

- **Segmentation basée discontinuité** : La segmentation se fait par rapport à une recherche des frontières des régions. Ces frontières sont caractérisées par une forte variation des niveaux d'intensité ou des niveaux de gris d'une image.
- **Segmentation basée homogénéité** : La détection de similarité vise à partitionner une image en régions similaires selon un ensemble de critères d'homogénéité prédéfini (niveaux de gris, couleur, texture, etc…).

La dualité entre les deux approches vient du fait que les régions sont séparées par des contours et que les contours fermés définissent des régions. Les deux approches peuvent être d'ailleurs, utilisées d'une manière coopérative pour une meilleure segmentation.

I.2.1.1.1 Détection de contours :

Un contour représente les pixels limites qui relient deux régions distinctes avec des attributs d'amplitude d'image changeants tels que différentes valeurs de luminance. La détection de contours consiste à trouver des endroits dans une image où l'intensité change rapidement, en utilisant l'un des deux critères généraux.

1. Trouvez des endroits où la dérivée première de l'intensité est supérieure en magnitude à un seuil spécifié (ordre 1).
2. Trouvez les endroits où la dérivée seconde de l'intensité a un passage par zéro (ordre 2).

La représentation des contours d'une image réduit considérablement la quantité de données à traiter, tout en conservant les informations essentielles concernant les formes des objets. L'inconvénient majeur de ces techniques est la non fermeture des contours cela aboutit donc souvent à une image de contours bruitée et fractionnée. Il faudra ajouter une étape appelée fermeture de contours [8], qui consiste à relier les extrémités des contours qui sont supposés appartenir à la même région.

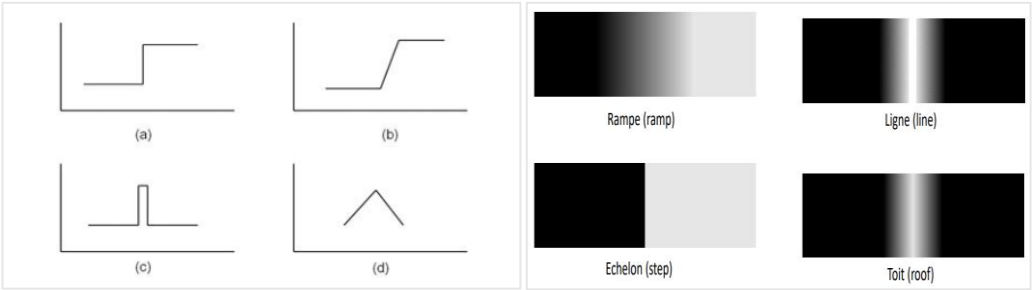


Figure I. 2: Types de contour (a) Echelon (b) rampe (c) ligne (d) toit [9].

a) Operateurs du premier et du second ordre :

La table I.1 résume les différents opérateurs utilisés pour la détection des contours.

L'opérateur	L'ordre	Filtres utilisés	Particularité	Références
L'opérateur de Sobel	opérateur d'ordre 1	-Filtres horizontaux et verticaux (3x3). -Filtre à 8 directions différentes: 0°, 45°,90°,135°,180°, 225°,270°,315°	-Principe simple. -Un petit calcul. -Une vitesse de fonctionnement rapide. -Filtre à 8 directions permet de détecter des contours plus obliques avec précision.	[10]
L'opérateur Prewitt	Opérateur d'ordre 1	-Filtres horizontaux et verticaux (3x3).	- Principe simple. -Détection des contours verticaux et horizontaux et leurs orientations. -Sensible au bruit. -Imprécis.	[11]
L'opérateur de Kirsch	Opérateur d'ordre 1	-Utilise un seul masque et en le tournant dans huit directions principales de la boussole : nord, nord-ouest, ouest, sud-ouest, sud, sud-est, est et nord-est par incréments de 45°.	- Précis.	[12]
L'opérateur Laplacien-Gaussien (LoG)	Opérateur d'ordre 2	-Appliqué à une image qui a d'abord été lissée pour réduire sa sensibilité au bruit. -Produit des contours doubles, la localisation des bords consiste à trouver les passages par zéro entre les bords doubles.	-Sensible au bruit. -Peu utilisé en raison des contours doubles.	[13]

Table I. 1 : Les différents opérateurs utilisés pour la détection des contours.

I.2.1.1.2 Détection de régions :

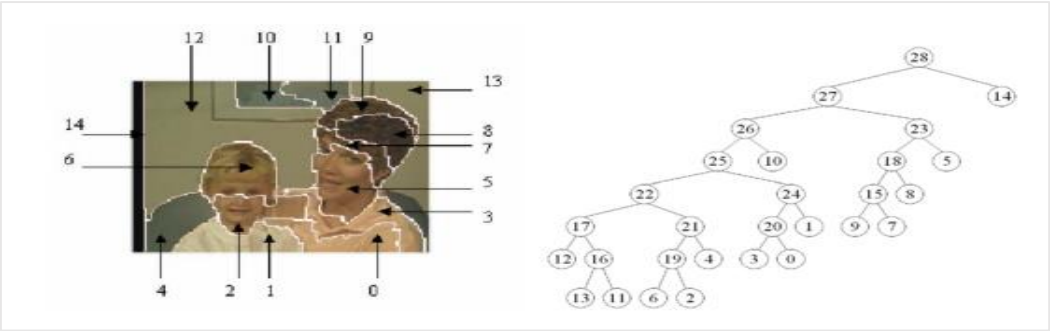
Repose sur le principe de l'uniformité. Elle consiste à découper l'image en régions homogènes, et considérant que les pixels adjacents à l'intérieur d'une région possèdent des caractéristiques similaires (niveaux de gris, couleur, texture, etc…). Le problème fondamental associé à cette catégorie d'approches est lié à la définition du critère d'homogénéité utilisé dans le processus de segmentation de l'image.

a) Croissance de régions : Cette approche consiste à sélectionner des pixels fixés dans des endroits spécifiques de l'image comme étant des points de départ. Ensuite regrouper les pixels se trouvant sur le voisinage avec chaque point de départ qui possède des propriétés similaires, tout en essayant de maintenir l'homogénéité de chaque région qu'on souhaite obtenir.

La croissance des régions peut être traitée en quatre étapes [14] :

- Sélectionner un groupe de pixels de départ dans l'image d'origine.
- Sélectionner un ensemble de critères de similitude tels que l'intensité ou la couleur du niveau de gris et définissez une règle d'arrêt.
- Agrandir les régions en ajoutant à chaque point de départ les pixels voisins qui ont des propriétés prédéfinies similaires.
- Arrêter la croissance de la région lorsqu'il n'y a plus de pixels répondant au critère d'inclusion dans cette région (Taille, ressemblance entre un pixel voisin et un pixel de départ).

b) Approche par division-fusion : La technique division-fusion consiste soit à diviser de manière très fine une image et ensuite à fusionner les régions adjacentes selon des critères d'homogénéité، soit à diviser l'image de façon itérative tant que les régions ne sont pas assez homogènes [15]. La division consiste en une décomposition par blocs de l'image, conduisant à une représentation de type arborescence appelée quadtree [16]. L'inconvénient majeur de cette approche est la sur-segmentation. L'approche proposée dans l'article [17] vise à minimiser le nombre de régions à obtenir en convertissant l'image en niveaux de gris (256 valeurs) qui seront divisés uniquement sur 32 régions.



a) Régions dans l'image b) Régions dans l'arbre

Figure I. 3: Illustration de l'arbre de partition.

Figure I. 4: Exemple de fusion des régions.

c) La ligne de partage des eaux (LPE) : S'applique à des images en niveau de gris qui peuvent être perçues comme un relief si l'on associe l'intensité de chaque point à une altitude. Les pixels de forte intensité correspondent à des sommets, tandis que les pixels de faible intensité correspondent à des fonds de vallées. L'idée est de rechercher, pour chaque vallée quelle est le bassin versant correspondant, ou de manière complémentaire, quelle est l'ensemble des lignes de crêtes qui séparent les différents bassins versants. Les zones où se rencontrent l'eau des différents bassins versants forment les frontières entre les régions et qui définis la ligne de partage des eaux (Figure I.5). Ce genre de méthode est très rapide mais fournit un nombre très grand de régions qu'il faudra par la suite fusionner pour obtenir une segmentation correcte des objets de la scène [18].

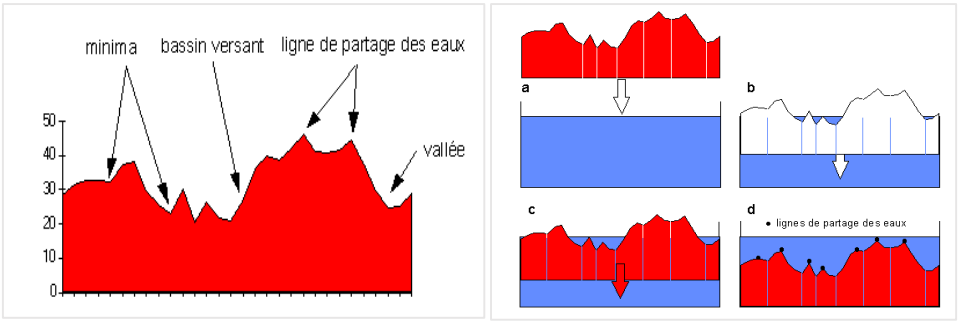


Figure I. 5: Exemple de ligne de partage des eaux.

a) Image Fleur b) Résultat de LPE

Figure I. 6: Image et résultat de l'algorithme de la ligne de partage des eaux.

I.2.1.1.3 Détection par coopération régions-contours :

L'approche coopérative donne une meilleure segmentation en combinant les techniques de segmentation basées sur les régions et celles basées sur les contours. Nous distinguons trois types de segmentation d'images par coopération régions-contours :

a) **La coopération séquentielle :** L'une des techniques de segmentation (région ou contour) est réalisée initialement، puis le résultat va être exploité par l'autre technique pour renforcer les critères et les paramètres de la segmentation (voir Figure I.7).
b) **La coopération des résultats :** Les deux types de segmentations seront réalisés indépendamment، leurs résultats seront intégrés afin d'obtenir une meilleure segmentation.
c) **La coopération mutuelle :** Les deux types de segmentations coopèrent mutuellement au cours de leur processus d'exécution.

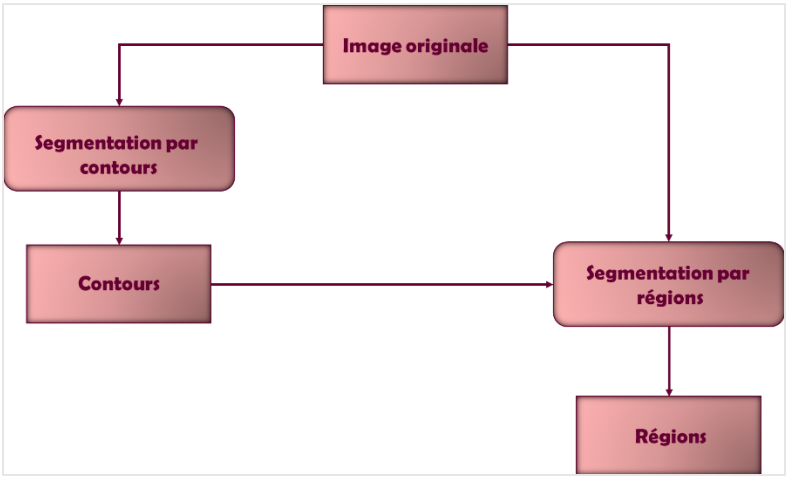


Figure I. 7: Le principe de la coopération séquentielle.

I.2.2 Détection basée sur les attributs :

La détection d'objets basée sur les attributs repose sur la représentation du contenu des régions de l'image en un ensemble de caractéristiques de manière fidèle et compacte. La caractérisation des objets consiste à les modéliser en extrayant des descripteurs de forme. Il est important de noter que pour obtenir une modélisation robuste, il est nécessaire de couvrir autant que possible les transformations géométriques et photométriques que peut avoir un objet.

a) **Les Histogrammes de gradient orienté (HOG) :** Se concentrent sur la forme de l'objet décrite par la distribution d'intensité des gradients ou de directions des contours. La mise en œuvre de ces descripteurs peut être obtenue par le calcul d'un angle de gradient pour chaque pixel. L'image est ensuite découpée en un ensemble de régions adjacentes, de taille fixe, appelées blocs. Pour chaque bloc, un histogramme local de directions des gradients ou d'orientations des contours est calculé par l'accumulation des votes des pixels. Pour que ce descripteur soit robuste aux changements d'illuminations et de contraste, une normalisation est effectuée à chaque histogramme calculé. La dernière étape consiste alors à concaténer l'ensemble des histogrammes pour définir un vecteur de caractéristiques [19]. La plus grande limitation du HOG est la non résolution du problème d'occultation, du fait que le découpage de l'image ce fais selon un modèle rigide, alors que les objets peuvent prendre différente posture comme par exemple le corps humain. Une solution alternative très prometteuse, est la caractérisation multi-parties [20] [21], qui consiste à considérer qu'un objet est constitué de différentes parties non homogènes devant ainsi être caractérisées et détectées indépendamment comme le montre la Figure I.8.

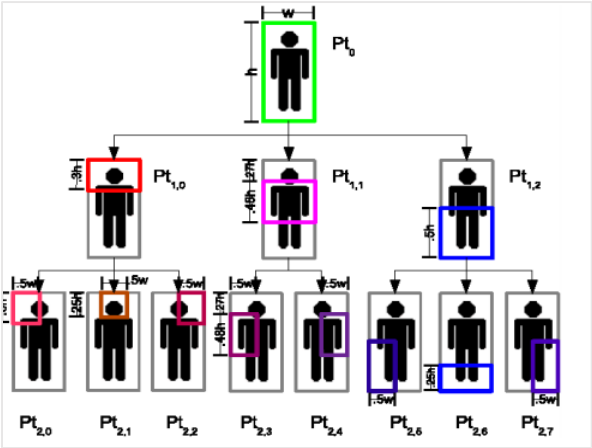


Figure I. 8: Hiérarchie des parties du corps humain [21].

b) SIFT (scale-invariant feature transform) : Permet une détection de points d'intérêts et une extraction de descripteurs locaux invariants à l'échelle et aux rotations, et partiellement invariants aux changements d'illumination. Il est ainsi utilisé pour de nombreuses applications comme la mise en correspondance entre images (voir Figure I.9).Différentes variantes de SIFT ont été développées : PCA-SIFT [22], GSIFT [23], CSIFT [24], SURF [25] et ASIFT [26], mais elles reposent toutes sur le même principe :

1. Dans un premier temps des points-clés sont extraits à partir d'un espace-échelle. La détection est invariante à différentes transformations (translation, changement d'échelle et d'illumination, rotation) et robuste au bruit (supposé gaussien la plupart du temps).

2. Une ou plusieurs orientations principales sont associées à chaque point. On obtient des points-clés caractérisés par quatre paramètres : une position (x, y), une échelle σ et une orientation θ, assurant ainsi l'invariance à la translation, au changement d'échelle et à l'orientation.

3. A chaque point-clé est ensuite associé un descripteur qui caractérise la géométrie locale de l'image autour du point. En utilisant les quatre paramètres du point, ce descripteur est invariant à différentes transformations.

4. Enfin, dans le cas où l'on cherche à mettre en correspondance deux images, les descripteurs de deux images sont comparés (en calculant des distances par exemple) afin d'apparier leurs points-clés respectifs.

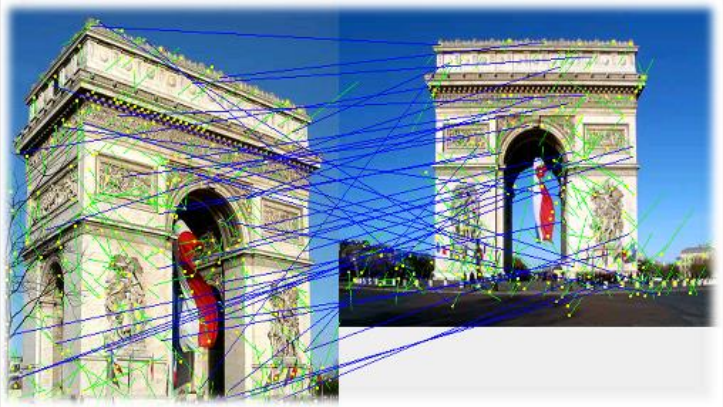


Figure I. 9: Mise en correspondance entre deux images.

I.3 La classification :

Le problème de classification peut se présenter sous la forme d'un ensemble de données (x_i, y_i) issues de l'observation d'un phénomène. Chaque exemple x_i est défini par un ou plusieurs attributs et une étiquette (label) $y_i \in \{1,\ldots, K\}$. Par le biais d'un algorithme d'apprentissage un classifieur H est construit en analysant l'ensemble des caractéristiques d'un objet extraites manuellement afin d'établir une relation entre ces observations et les étiquettes pendant une phase d'apprentissage. C'est à ce classifieur que revient ensuite la tâche d'attribuer des étiquettes adéquates à de nouveaux exemples distincts de ceux utilisés en apprentissage et en les affectant aux classes prédéfinies. L'objectif de la classification est de trouver un modèle d'apprentissage tel que $H(x_i) = y_i$.

Les algorithmes d'apprentissage peuvent se catégoriser selon le type d'apprentissage qu'ils emploient :

- **Apprentissage supervisé :** Les différentes classes possibles sont déjà connues. Pour fonctionner, l'algorithme a besoin d'un ensemble d'apprentissage contenant des exemples de toutes les classes avec leurs étiquettes associés.
- **Apprentissage non supervisé :** Les classes ne sont pas connues à l'avance. L'algorithme reçoit uniquement un ensemble d'observations. A lui ensuite de trouver, en fonction de la structure des données, les classes les plus pertinentes. Ces algorithmes peuvent être utilisés pour regrouper des observations proches en différents ensembles appelés clusters.

- **Apprentissage semi-supervisé :** Est à mi-chemin entre les approches supervisées et non-supervisées. Les classes sont connues lors de l'apprentissage, mais l'algorithme accepte en entrée des exemples qui ne sont pas forcément étiquetés.

I.3.1 Notion de base :

Le processus de classification est fondé sur deux principales notions : **la similarité** et **la distance**. La similarité est une valeur utilisée pour quantifier la ressemblance de deux objets en termes de points communs. Elle peut être mesurée par un calcul de distance. Les similaires entre eux en une distance nulle, la distance maximale sépare deux objets différents.

Le nom	L'équation
Distances de Minkowski	$d_{mink} = \sqrt[p]{\sum_{i=1}^{n} \lvert x_i - y_i \rvert^p}$
La distance euclidienne	$d_{eucl}=\sqrt{\sum_{i=1}^{n} \lvert x_i - y_i \rvert^2}$
La distance Manhattan ou City-Block	$d_{manh} = \sum_{i=1}^{n} \lvert x_i - y_i \rvert$
La distance Chebishev ou Taxicab	$d_{cheb} = max_i \lvert x_i - y_i \rvert$
Divergence de Kullback Leibler	$d_{KL}(X,Y) = \sum_{i=0}^{n} x_i \log(\frac{x_i}{y_i})$
La distance de Bhattacharyya	$d_{Bha}(X,Y) = -\ln(\sum_{i=0}^{n} \sqrt{x_i y_i})$

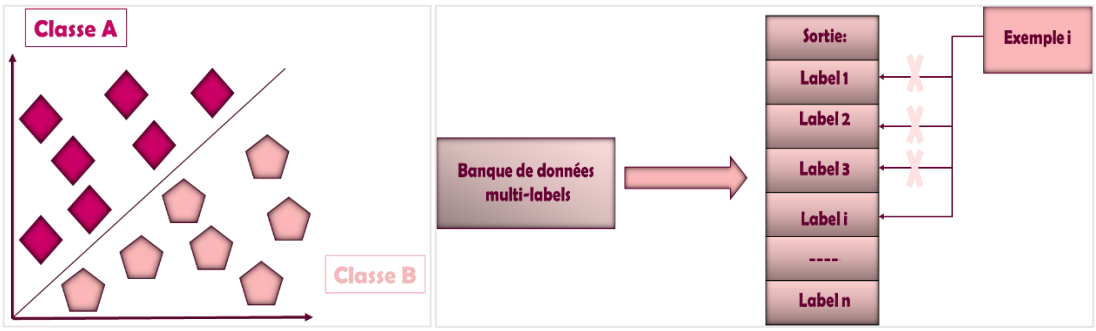
Table I. 2: Les distances utilisées pour le calcul de similarité.

I.3.2 Les différents contextes de classification :

a) Classification mono-label : Consiste à attribuer une seule et unique étiquette à chaque exemple d'entrée, à partir d'un ensemble d'étiquettes. Le classifieur apprend par la suite a

associé à chaque nouvelle observation sa classe la plus probable. Les problèmes de classification mono-étiquette sont divisés en deux catégories :

- **Classification binaire :** Implique la classification des échantillons de données d'entrée dans l'une des deux catégories possibles.
- **Classification multi-classes :** Chaque élément des échantillons de données appartient à une seule et unique classe malgré la présence de plusieurs.



a) Classification binaire b) Classification multi-classes

Figure I. 10: Exemple de classification mono-label.

b) Classification Multi-labels : La classification multi-labels, ou multi-étiquettes consiste à attribuer à un exemple d'entrée plusieurs étiquettes simultanément.

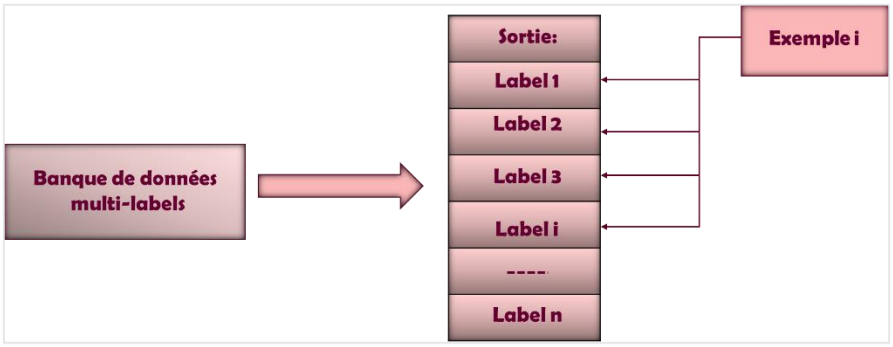
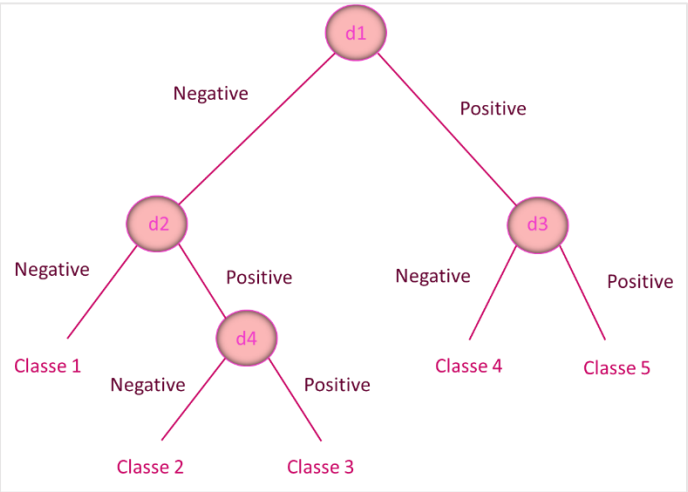


Figure I. 11: Exemple de classification multi-labels.

I.3.3 Les algorithmes de classification :

a) Les arbres de décision : Est un classifieur présenté sous forme d'une structure arborescente (voir Figure I.12). Comporte trois types de nœuds : 1) un nœud racine correspondant à l'ensemble des données d'apprentissage,2) des nœuds internes dont les exemples vont être testés, 3) les nœuds terminaux qui représentent les classes. L'arbre est parcouru de la racine jusqu'à un nœud terminal. Les exemples seront testés dans les nœuds

internes généralement sur un seul attribut dont le but d'obtenir plusieurs sous-ensembles ne contenant que des exemples appartenant à des classes homogènes. Une fois arriver à un nœud terminal qui correspond à une classe l'exemple sera prédit comme faisant partie de cette classe [27].



d_1 : Nœud racine.
d_2, d_3 : Nœuds internes.
Négative, Positive : les attributs.

Figure I. 12: Exemple d'arborescence pour un problème à 5 classes.

b) KNN (Les K plus proches voisins) : Dans un contexte de classification d'une nouvelle observation x_i, l'idée fondatrice est de supposer qu'une observation est similaire à celle de ses plus proches voisins. On suppose par ailleurs qu'il existe une notion de distance, la classe de x_i est déterminée en fonction de la classe majoritaire parmi les k plus proches voisins de l'observation x_i. La méthode KNN est donc une méthode à base de voisinage, non-paramétrique, ceci signifiant que l'algorithme permet de faire une classification sans faire d'hypothèse sur la fonction $y_i = H\ (x_i)$.

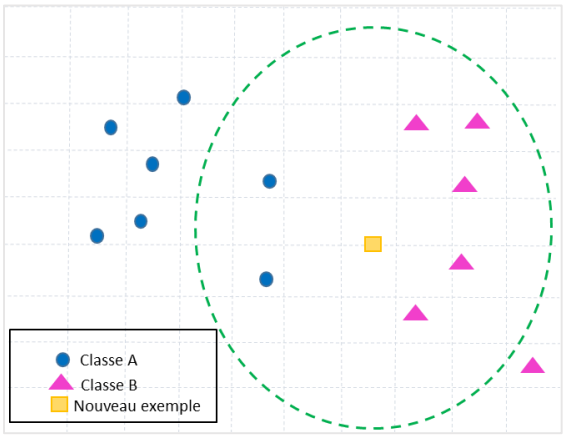


Figure I. 13: Exemple de classification par l'algorithme K-Plus Proches voisins.

c) K-means : Un algorithme d'apprentissage non supervisé itératif de clustering. Il permet à partir d'une base de données non étiquetée de former des clusters homogènes. Les observations qui sont considérées similaires sont associées au même cluster. L'algorithme est initialisé aléatoirement avec un certain nombre de clusters K pour lesquels un point moyen, appelé centroïde, est évalué. A chaque itération, la distance entre chaque exemple aux différents centroïdes est calculée ; chaque exemple est alors associé au cluster dont la distance au centroïde est la plus proche. Puis les centroïdes sont réévalués. L'algorithme se termine lorsqu'il n'y a plus aucun changement. Un exemple simple de fonctionnement est montré sur la Figure I.13, où les données de départ sont représentées sur l'image à gauche avec une initialisation de 3 centroïdes. Le résultat final est représenté sur l'image à droite.

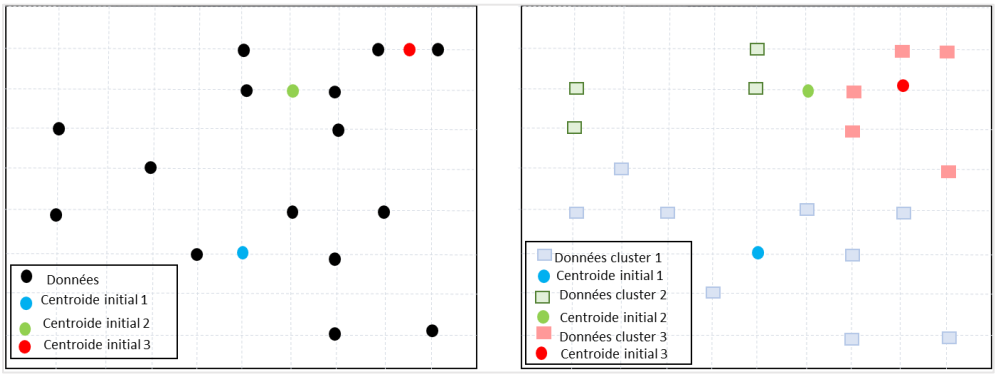
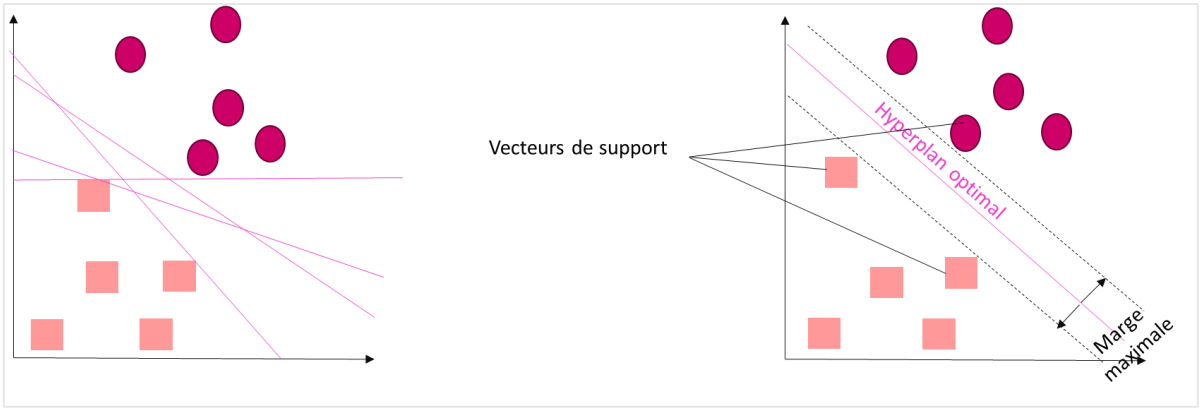


Figure I. 14: Exemple simple de classification non-supervisé par l'algorithme des K-means.

d) SVM (les machines à vecteurs de support) : Initialement proposées pour résoudre des problèmes de classification linéaires, les machines à vecteurs de support ont été étendues à des problèmes non linéairement séparables. Cette technique de classification binaire tente de séparer deux classes d'exemples en cherchant un séparateur linéaire optimal appelé hyperplan dans un espace approprié, tout en garantissant que la distance entre les deux classes soit maximale.

1. Cas linéairement séparable :

L'idée des SVMs est de rechercher un hyperplan qui sépare le mieux deux classes. Il est évident qu'il existe une multitude d'hyperplan valide mais la propriété remarquable des SVM est que cet hyperplan doit être optimal. Formellement, cela revient à chercher un hyperplan dont la distance minimale aux exemples d'apprentissage est maximale. On appelle cette distance « marge » entre l'hyperplan et les exemples.



a) représentation des multiples hyperplans b) L'hyperplan optimal

Figure I. 15: Exemple de cas linéairement séparable.

2. Cas non linéairement séparable :

La plupart des problèmes réels sont difficiles à séparer par un simple hyperplan. Si par exemple les données des deux classes se chevauchent sévèrement, aucun hyperplan séparateur ne sera satisfaisant. L'idée principale est donc de changer l'espace de représentation des données d'entrées en un espace de plus grande dimension, éventuellement de dimension infinie dit espace de re-description dans lequel il est probable qu'il existe une séparation linéaire. Cette transformation non-linéaire est réalisée grâce à une fonction noyau ϕ qui doit être symétrique et semi-définie positive.

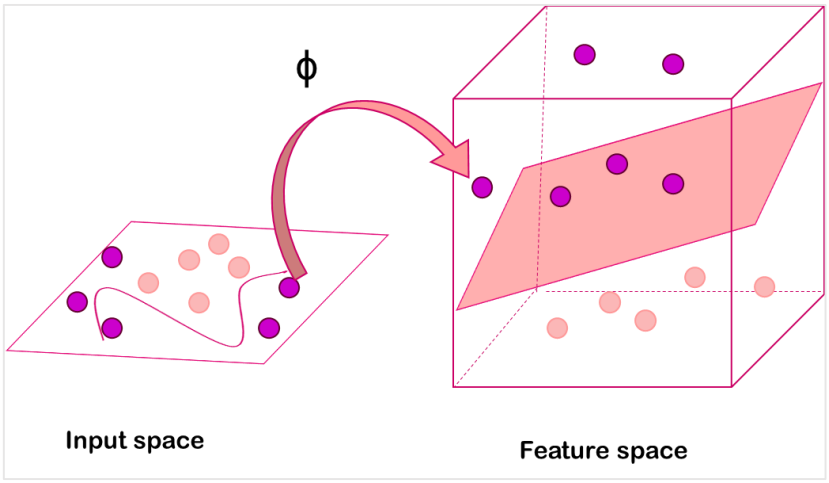


Figure I. 16: Exemple d'un problème non linéairement séparable.

I.3.4 Les limitations des algorithmes de classification :

La table I.3 résume les différentes limitations pour chacun des algorithmes de classification cités ci-dessus.

Algorithme de classification	Limitations
Les arbres de décision	-L'inconvénient majeur de l'arbre de décision est l'instabilité, ce qui mène à des difficultés à contrôler sa taille.
KNN	-La classification par KNN est relativement coûteuse. Elle nécessite le calcul de la distance des k voisins les plus proches, avec la croissance de la taille de l'ensemble d'entraînement, l'algorithme devient intensif en calcul. -Les caractéristiques non pertinentes entraîneront une dégradation de la précision. -C'est un apprenant paresseux ; il calcule la distance sur k voisins. Il ne fait aucune généralisation sur les données d'entraînement et les conserve toutes. -Se laisse facilement tromper par des attributs non pertinents.
K-means	-La difficulté réside dans le choix du nombre de cluster K. Il n'existe pas de procédé automatisé pour trouver le bon nombre de clusters. -Influence du partitionnement initial : Un nombre K grand peut conduire à un partitionnement trop fragmenté des données, ce qui empêchera de découvrir des patterns intéressants dans les données. Par contre un nombre de clusters trop petit, conduira à

	avoir potentiellement des clusters trop généralistes contenant beaucoup de données. -Les performances se dégradent lorsqu'il existe une différence de taille et de densité entre les clusters. -Les points isolés sont mal gérés. -Moins précis que l'apprentissage supervisé.
SVM	-Leurs performances diminuent avec un volume de données important en raison de l'augmentation du temps d'entrainement. -Difficile de trouver la fonction de noyau appropriée. -SVM ne fonctionne pas bien lorsque les données sont bruitées.

Table I. 3: Limitations des différents algorithmes de classification.

Partie II : nécessité d'une cartographie haute définition

I.4 Contexte :

Pour qu'un véhicule se déplace et intervienne dans un environnement complexe et dynamique en toute sécurité et en parfaite autonomie, il a besoin d'être doté de capacités d'observation lui permettant de percevoir son environnement, afin de recueillir différentes informations dont le but de construire une carte HD reflétant la structure spatiale du monde réel pour planifier des actions et d'assurer la sécurité durant la navigation.

La nécessite d'une carte HD à deux origines. Premièrement, une carte est souvent nécessaire pour que le véhicule sache où il doit rouler, elle permet alors d'établir un chemin et définir une destination. Deuxièmement, une carte permet de réduire les erreurs de localisation en utilisant des points de référence (landmarks) et en revisitant des lieux déjà explorés. Les cartes HD ont une haute précision, souvent de l'ordre de centimètre. Outre l'exigence de haute précision, les cartes HD fournissent des informations cartographiques détaillées. Il peut s'agir d'informations sur les limitations de vitesse, les voies de circulation, ainsi que des descriptions des bâtiments et des objets situés autour des routes.

La localisation se fait en comparant les données provenant des capteurs d'un véhicule avec celles sur la carte HD, la voiture peut obtenir une meilleure précision sur sa position actuelle. Cela peut également fonctionner comme une sauvegarde au cas où la voiture perdrait son signal GPS. Pour cela, il faudrait que la carte soit à la fois très précise, riche, et mise à jour.

L'approche proposée par Ryan W et al [28], permet de localiser un véhicule autonome en milieu urbain. La localisation est obtenue en comparant les images d'une caméra monoculaire à une carte LiDAR 3D. Le système proposé par Christian H et al [29], a été utilisé avec succès sur les véhicules autonomes du projet V-Charge. Il contient quatre caméras montées sur la voiture de manière à ce qu'elles permettent une couverture à 360°. Il permet de générer des cartes denses précises, puis localiser visuellement la voiture par rapport à ces cartes. De plus il permet la détection d'obstacles en temps réel. Xia et al [30], proposent un système de localisation conçu pour des applications de conduite entièrement autonomes pour des scènes de conduite difficiles tel que les tunnels. Le système fusionne des informations multi-capteurs GNSS, LiDAR et IMU avec une carte locale préétablie afin de corriger les erreurs de localisation. Le système permet d'estimer la position, la vitesse et l'attitude du véhicule. L'approche proposée par Cai H et al dans [31], consiste à utiliser un récepteur GPS, une caméra monoculaire et une carte HD. La

caméra monoculaire embarquée sur le véhicule, est utilisée pour calculer la distance latérale entre le véhicule et les voies voisines. La carte HD, permet de corréler la distance latérale locale et les coordonnées GPS et les alimenter en tant que mesures dans un filtre de Kalman pour obtenir la localisation du véhicule. W Ma et al [32], proposent un système capable de localiser un véhicule autonome en faisant la correspondance entre les objets sémantiques (les limites de voie et les panneaux de signalisation) stockés sur la carte HD, et les données fusionnés provenant de LiDAR, caméra, GPS, et IMU. Cette approche permet des mises à jour cartographiques et des coûts de stockage réduits.

I.5 Solutions existantes pour les cartes HD :

Les cartes pour la conduite n'est pas une chose nouvelle, et certaines entreprises impliquées dans le domaine de la cartographie commencent à réaliser que la conduite autonome est l'avenir. En plus des entreprises déjà établies comme TomTom [33], il existe un certain nombre de start-up qui visent à créer des cartes HD ou à fournir des produits liés aux cartes. Certaines de ces entreprises sont présentées dans le tableau I.3 ci-dessous.

Entreprise	Carte HD	Open source	Service
Comma.ai	Oui	Oui	-Le produit principal est OpenPilot pour le suivi de voie et le régulateur de vitesse adaptatif. -Les solutions de comma.ai sont pour la plupart open source.
Mipillary	Non	Non	-Améliorer les cartes avec des images au niveau de la rue. -Détecter les caractéristiques des cartes à partir d'images au niveau de la rue.
DeepMap	Oui	Non	-Cartes HD précises et maintenables pour une conduite autonome.
Tomtom	Oui	Non	-Carte HD pour la conduite autonome. -Propose également le RoadDNA, à des fins de localisation en comparant les signaux d'entrée avec le RoadDNA.
lvl5	Oui	Non	-Vise à réaliser des cartes HD à partir de capteurs bon marché, équivalents à ceux d'un téléphone intelligent.

HERE	Oui	Non	-HERE prétend fournir une carte HD continuellement mise à jour pour la conduite autonome. Ils soulignent en particulier l'importance de données précises sur les limites des voies, et l'importance de cartes actualisées.
Civil Map	Oui	Non	-Fournit une plate-forme évolutive de cartographie et de localisation HD en périphérie.
Carmera	Oui	Non	-L'un des produits de CARMERA est une carte HD pour la conduite autonome. CARMERA a fait équipe avec le Toyota Research Institute-Advanced Development, Inc. (TRI-AD) pour coopérer sur une solution de carte HD.

Table I. 4: Un aperçu des entreprises dans le domaine de la cartographie [34].

I.6 Techniques pour la génération de cartes HD :

La réalisation d'une carte HD peut être considérée comme comprenant deux parties principales, la collecte de données et la génération de la carte. Le processus de collecte de données consiste à obtenir toutes les données pertinentes d'une certaine zone, toutes les données que le véhicule peut éventuellement utiliser. Il existe principalement deux façons d'aborder ce problème. La première approche consiste à utiliser un véhicule spécial configuré pour la collecte de données fournies par des capteurs, puis les réutiliser comme base pour la génération de cartes. La deuxième approche consiste à utiliser les bases de données cartographiques déjà existantes et traitées. La méthode de génération des cartes se base largement sur les types de données disponibles, ainsi que sur le type de résultats censé être générer.

I.6.1 Cartes HD à partir de données de capteurs :

La génération de cartes à partir de données de capteurs est probablement la méthode la plus utilisée. Cette méthode consiste à équiper un véhicule de capteurs qui lui permettent de percevoir, d'observer et de réaliser des mesures sur les éléments des scènes environnantes. Ces capteurs sont classés en deux types : extéroceptifs et proprioceptifs. Les capteurs extéroceptifs sont les capteurs qui permettent au véhicule d'obtenir les informations caractéristiques d'une position que le véhicule peut acquérir dans son environnement [35], tandis que les capteurs proprioceptifs fournis des informations internes au véhicule qui le renseignent, dans le cas de la navigation, sur son déplacement dans l'espace. Ces informations peuvent provenir de la

mesure de la rotation de ses roues ou de la mesure de l'accélération grâce à une centrale inertielle [35]. L'approche proposée par Joshi A et al dans [36], permet d'estimer la structure des voies et de générer une carte routière en combinant les informations OpenStreetMap préétablies et des données LiDAR 3D obtenues à partir d'un véhicule d'essai. Gwon et al [37], proposent un système utilisant un scanner laser 3D mobile et un GPS pour extraire les données de géométrie de la route, la position, les marquages de voie et produire une représentation de carte routière avec des courbes polynomiales.

Généralement les véhicules sont équipés de capteurs suivants [38] :

GPS/IMU :	L'unité GPS/IMU enregistre les coordonnées qui constitueront la base des routes. Il est important de disposer d'un GPS de haute précision pour obtenir des cartes très précises.
Camera :	Il peut y avoir une caméra ou plusieurs orientées dans des directions différentes. Il s'agit de détecter et de localiser des éléments de la route ou autour d'elle, comme les voies de circulation.
LiDAR :	Utilisé avec les caméras pour localiser des objets et des détails autour de la route, certaines entreprises essayent de générer des cartes sans utiliser un LiDAR pour réduire le coût des capteurs car les LiDAR sont très chers.

Table I. 5: Les types de capteurs utilisés pour générer des cartes HD.

I.6.2 Cartes HD à partir des données cartographiques disponibles :

Les cartes HD peuvent également être réalisées en combinant et en modifiant les sources de données existantes [39]. Il est ainsi possible de représenter une zone sur une carte sans avoir à collecter les données des capteurs. Il existe un certain nombre de bonnes sources de données pour la topologie des routes ainsi que des sources pour les limites de vitesse, les panneaux de signalisation et autres éléments routiers. Cependant, cette approche pose un problème de précision. La plupart des sources disponibles sont compréhensibles pour les humains, avec une exigence de précision différente de celle des voitures. Si la carte doit être utilisée pour la localisation, une carte précise au centimètre près peut être nécessaire. Si la seule exigence est de connaître certaines données sur les routes ainsi qu'un modèle approximatif de la topologie de la route, cela peut être une bonne solution ne nécessitant pas de capteurs.

Par exemple, Google Maps offre des cartes, des images satellites et de rues, des informations de circulation routière et un service de calcul d'itinéraire. Elle dispose d'une équipe d'opérateurs qui assurent la validité et la cohérence des cartes Google [40], et par conséquent,

toute modification éventuelle du réseau routier doit être approuvée par l'un des opérateurs. Un autre exemple OpenStreetMap (OSM) : projet de constitution collaborative d'une base de données géographique [39], placé sous licence Creative Commons. Surnommé le « Wikipédia de la cartographie », OSM repose sur les contributions de milliers de cartographes amateurs, qui éditent et enrichissent bénévolement des portions de cette carte collective. Les données publiées en open data sont utilisées par les contributeurs pour compléter cette base de données. OpenStreetMap (OSM), met à jour environ 1 million de nœuds par jour. Ces cartes ont une précision raisonnablement élevée.

Cependant ces approches présentent des limites fondamentales en raison de l'intervention humaine. Elles souffrent d'une réponse de mise à jour lente lorsque des changements d'environnement se produisent. Pour combler cette lacune, Stanojevic R et al [41] proposent MapFuse, un système qui fusionne deux cartes, une de haute qualité mise à jour lentement (par exemple OSM) et une carte réalisée automatiquement mais à couverture incomplète et une structure topologique imparfaite.

I.7 Conclusion :

Ce chapitre a été consacré pour rappeler les techniques de base nécessaires à la détection et à la reconnaissance de classes d'objets à partir de l'interprétation des images. L'analyse des différentes méthodes de segmentation précédentes a montré qu'il est difficile d'aboutir à un résultat satisfaisant en appliquant une seule méthode de segmentation, d'où la combinaison de plusieurs est nécessaire. Les performances des algorithmes de classification sont fortement influencées par la représentation et la caractérisation des données.

Les algorithmes simples de classification décrits dans ce chapitre n'ont pas réussi à résoudre les problèmes centraux de l'IA, tels que la reconnaissance d'objets en vue de l'utilisation des grandes bases de données qui imposent également des coûts de calcul élevés. Le développement de l'apprentissage profond a été motivé en partie par l'échec des algorithmes traditionnels à bien généraliser sur de telles tâches. Le chapitre suivant sera consacré à l'apprentissage profond.

Chapitre II :

Apprentissage profond.

II.1 Introduction :

L'apprentissage profond (Deep Learning : DL) est un concept d'Intelligence Artificielle (IA), dérivé de l'apprentissage machine (Machine Learning : ML) [42] (voir Figure II.1). L'IA correspond à un ensemble de technologies et outils qui permettent de simuler l'intelligence et accomplir des tâches que les humains exécutent de manière intuitive et quasi automatique tel que la perception, la compréhension et la prise de décision [43]. Le DL est l'une des raisons qui ont conduit au progrès de l'IA, et qui fait penser que finalement, il existe une possibilité pour l'IA de devenir plus réaliste. Le DL s'intéresse à la reconnaissance des formes et à l'apprentissage à partir des données en se basant sur les modèles de réseaux de neurones artificielles.

Dans ce qui suit, nous allons rappeler les concepts de ML. Nous présentons ensuite le modèle du réseau de neurones qui est à la base de DL. Nous présentons ensuite les différentes architectures profondes de réseaux de neurones et discutons leurs utilités.

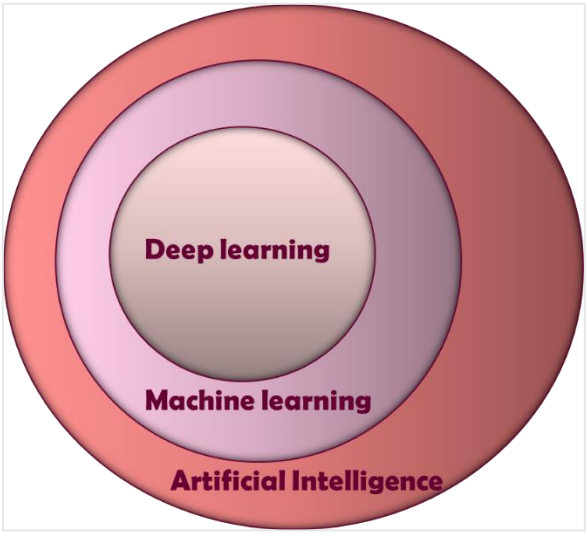


Figure II. 1: Diagramme décrivant le DL comme un sous-domaine de l'apprentissage automatique qui est à son tour un sous-domaine de l'IA [43].

II.2 Apprentissage machine :

La théorie de ML est un domaine qui croise les aspects mathématiques et informatiques découlant de l'apprentissage itératif [43]. Une définition formelle de ML a été proposée par T. Mitchell [44], lequel affirme qu'un programme informatique est dit capable d'exécuter une tâche lorsqu'il peut améliorer ses performances avec l'expérience. C'est grâce à l'apprentissage qu'il pourra apprendre à exécuter de nouvelles tâches et acquérir de nouvelles compétences.

L'apprentissage nécessite une expérience sous forme d'une base de données que le modèle analyse. Cet apprentissage d'une tâche se fait à l'aide d'une fonction de coût calculée sur une base de données d'apprentissage, distincte de celle utilisée pour le test, afin de mesurer les performances. Lors de l'apprentissage, les paramètres peuvent être ajustés pour optimiser le modèle afin de réduire la fonction coût sur la base de données test. Le fait que ces performances soient mesurées sur des bases de données différentes cela implique une capacité de généralisation d'un modèle dit d'apprentissage [43].

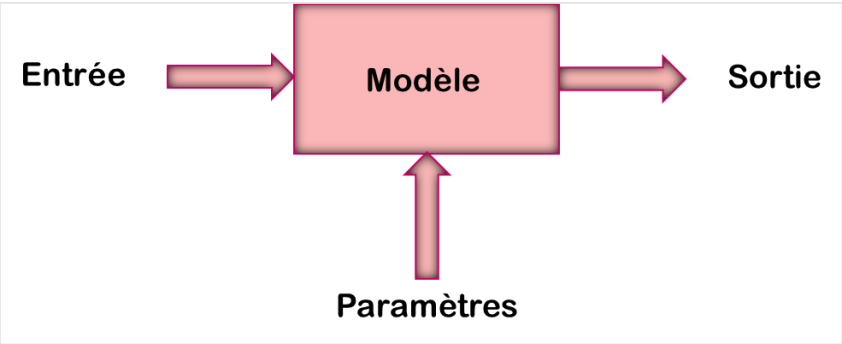


Figure II. 2: Modèle d'apprentissage.

II.3 Rappel sur les réseaux de neurones :

Les réseaux de neurones existent depuis les années 40 et ont subis divers changements de nom, y compris la cybernétique, le connexionnisme et les Réseaux de Neurones Artificiels (RNA) les plus familiers [42]. Leur conception et fonctionnement sont fortement inspirés des neurones biologiques [45]. Ils se composent d'un ensemble d'unités élémentaires de calcul non linéaire fonctionnant en parallèle appelées « neurones formels ». Disposés sous forme de couches successives et interconnectés entre eux de diverses manières, ils forment un graphe orienté définissant la topologie du réseau [45].

En 1943, Warren McCulloch et Walter Pitts [46], proposent un classifieur binaire, constitué d'un réseau de neurone formel capable de réaliser diverses opérations logiques ou arithmétiques. Une des limitations du neurone formel était que les pondérations utilisées pour l'apprentissage devaient être ajustées manuellement.

En 1957, Frank Rosenblatt définit le perceptron [47]. Il s'agit d'un réseau de neurones acyclique, composé d'une couche de neurones en entrée, et d'une autre en sortie. Il est le premier réseau capable d'apprendre par expérience, et le premier véritable réseau de neurones artificiels. Ce modèle pouvait déterminer automatiquement les poids nécessaires. Cependant, les travaux de Minsky et Papert [48], ont démontré les limitations théoriques du perceptron.

Ces limitations concernent l'impossibilité de traiter des problèmes non-linéaire avec un perceptron à une seule couche cachée quel que soit le nombre de neurones utilisés. Pour résoudre ce problème, il a été nécessaire de passer d'un perceptron monocouche à un perceptron multi-couches. Cependant, l'apprentissage devient plus difficile pour ce dernier. En 1975 [49], Paul Werbos formule un algorithme de descente de gradient pour l'entrainement des réseaux de neurones à multi-couches. Cet algorithme de rétro-propagation fut appliqué pour entraîner des perceptrons multi-couches avec succès pour la première fois au milieu des années 80 [50]. Pour résoudre des problèmes plus complexes, il était nécessaire de multiplier encore plus les couches dans les réseaux de neurones. Dès lors, deux difficultés apparaissent :

1) Plus le réseau est profond, plus il faut de puissance de calcul pour l'apprentissage et l'utilisation.

2) Plus le réseau est profond, moins l'algorithme d'apprentissage fonctionne correctement.

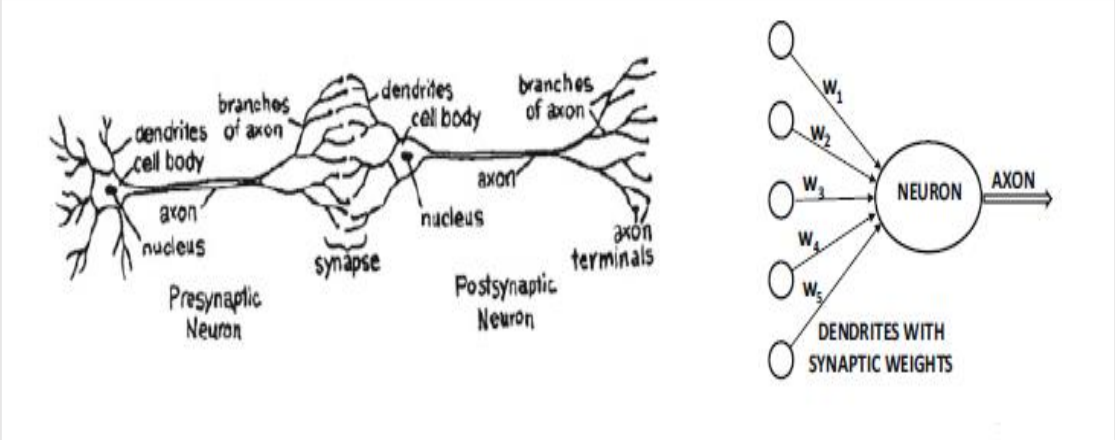
Ce dilemme a limité l'utilisation des réseaux de neurones pendant près de deux décennies. En effet, il était fréquent de devoir utiliser des réseaux pour lesquels on ne parvenait pas à faire correctement l'apprentissage.

Dans la suite de cette partie, nous présentons le cadre théorique des réseaux de neurones artificiels et aux méthodes permettant leur entraînement pour différentes tâches avant de s'intéresser plus particulièrement aux modèles convolutifs.

II.3.1 Modélisation d'un neurone artificiel :

Un neurone formel est une représentation artificielle et schématique d'un neurone biologique (Voir Figure II.3) :

- Les synapses sont modélisées par des poids.
- Le soma ou corps cellulaire est modélisé par la fonction d'activation.
- L'axone par l'élément de sortie.



a) Neurone biologique b) Neurone artificiel

Figure II. 3: Exemple d'un neurone biologique/artificiel à action direct (Feedforward) [45].

Le neurone formel peut être assimilé à un classifieur linéaire adapté pour la résolution des problèmes binaire, et ne peut avoir donc que deux états soit actif (1) soit inactif (0). C'est une unité, qui calcule une sortie ŷ en fonction des données d'entrées qu'ils lui ont été fournies comme suit :

$$\hat{y} = \sigma(s) = \sigma(x \cdot w + b) = \sigma\left(\sum_{i=1}^{n} x_i w_i + b\right) \quad (2.1)$$

Le neurone formel est défini par les caractéristiques suivantes :

1. Le vecteur de données d'entrées $(x_1, x_2, \dots \dots, x_n)$.
2. Les poids $(w_1, w_2, \dots, w_n)$ accordés à chacune des entrées pour l'activation du neurone.
3. La fonction d'agrégation : effectue la somme des produits des informations par leurs poids associé et traite cette donnée $s = \sum_{i=1}^{n} x_i w_i + b$.
4. Le seuil ou biais "b" prend souvent les valeurs -1 ou +1. Il dicte à partir de quelle valeur la somme des produits des poids et des sorties de la couche précédente passe le seuil où un neurone est considéré comme actif.
5. Fonction d'activation σ (.) qui associe à la valeur agrégée une unique valeur, le neurone compare cette valeur à un seuil et en décide la sortie soit actif ou inactif. Cette fonction peut prendre plusieurs formes.

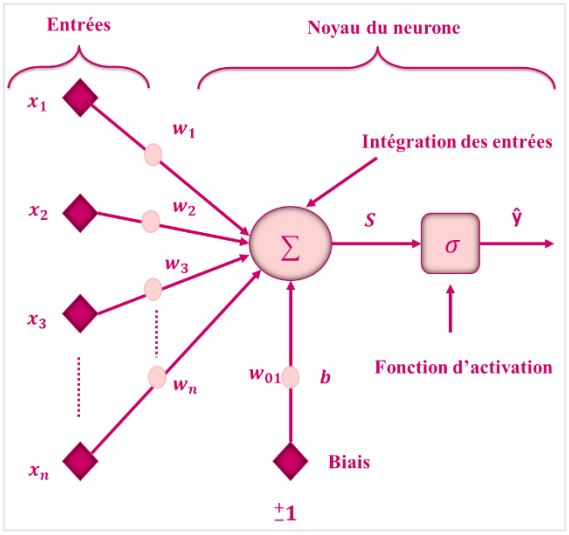


Figure II. 4: Neurone formel.

Un modèle neuronal à base d'un seul neurone est rarement utilisé. En fait, dans son état isolé, un neurone ne peut offrir de fonction intéressante. Autrement dit, un modèle neuronal est qualifié par les interconnexions entre ces unités élémentaires.

II.3.2 Fonctions d'activation :

L'apprentissage des réseaux de neurones requiert une fonction d'activation qui grâce à une transformation **linéaire** ou **non-linéaire** des entrées, prend la décision de transmettre ou non le signal selon la valeur de sortie obtenue. Dans les réseaux de neurones profonds, les fonctions d'activations **σ** sont choisies pour leurs nature dérivables en tout point et leurs dérivées **σ'** soient monotones croissantes. Elles aident également à normaliser la sortie de chaque neurone dans une plage comprise entre 1 et 0 ou entre -1 et 1. Dans ce qui suit nous présentons quelques exemples des fonctions d'activations :

La fonction	L'équation	Particularité	Réfé-rences
Sigmoïde	$\sigma(x) = \frac{1}{1+e^{-x}}$	-Utilisée principalement dans les réseaux de neurones à action directe. -Utilisée pour des tâches de classification binaire. -Fonction saturante.	

tangente hyperbolique	$\sigma(x) = \frac{1-e^{-2x}}{1+e^{-2x}}$	-Représente une version symétrique de la sigmoïde. -Fonction saturante.	
La fonction d'unité linéaire rectifiée (ReLU)	$\sigma(x)=$ $\max(0,x)\begin{cases}0 & si\ x<0\\ x & si\ x\geq 0\end{cases}$	-Effectue uniquement une opération de seuillage pour chaque élément d'entrée, d'où les valeurs inférieures à zéro sont mises à zéro. -Traite le problème de saturation.	[51]
LeakyReLU	LReLU_α(x) = max (0, x)-αmax (0, x), avec α un hyperparamètre.	-Une variation de ReLU, elle permet la rétro propagation, même pour des valeurs d'entrée négatives. -Elle converge beaucoup plus rapidement.	[52]
PReLU	PReLU(x,α) = max (0, x)- αmax (0, -x), avec α optimisable.	-Résout le problème des neurones inactifs causés par les dérivées nulles qui bloquent la rétro-propagation en utilisant un paramètre entrainable α.	[53]
ELU	$ELU_\alpha(x)=\max(0,x)+\alpha(\exp(-\max(0,-x))-1)$.	-Plus coûteuse en terme de calcul que le ReLU en raison de l'opération exponentielle incluse.	[54]

Softmax	$\sigma_k(u_k) = \frac{e^{u_k}}{\sum_{i=1}^{k} e^{u_i}}$, où k désigne la classe considérée.	-Utilisée généralement pour la dernière couche d'un réseau de neurone. -Utilisée pour les tâches de classification multi classes.	

Table II. 1: Les différentes fonctions d'activations.

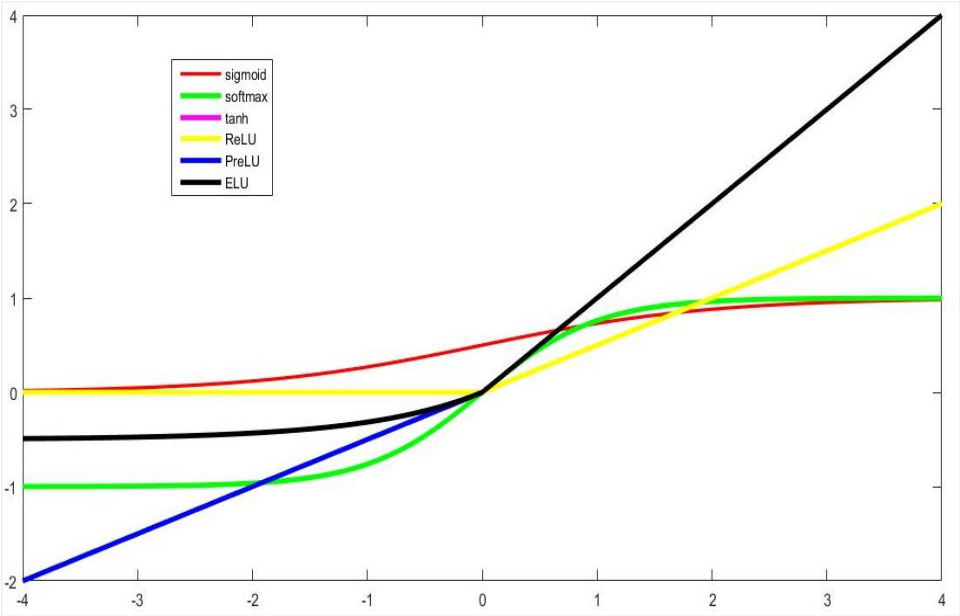
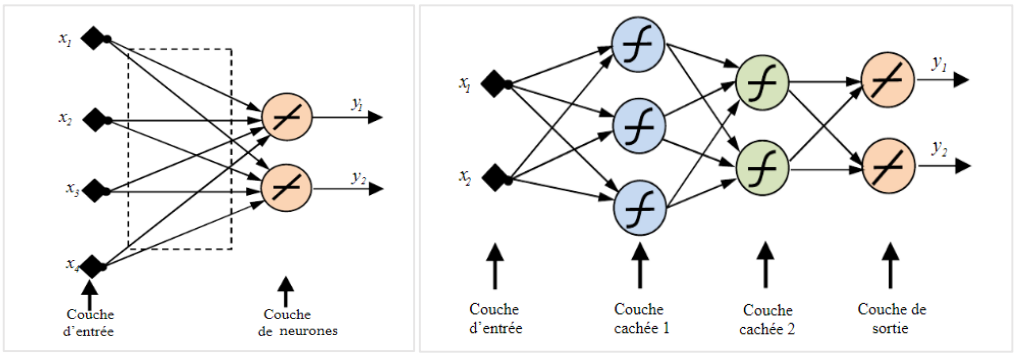


Figure II. 5: Exemples de fonctions d'activation.

II.3.3 Architecture des réseaux de neurones :

Depuis l'apparition des réseaux de neurones artificiels, plusieurs modèles neuronaux ont été proposés. Chaque modèle est caractérisé par sa règle d'apprentissage, les fonctions intra neuronales (fonction d'activation) et l'architecture des connexions inter-neuronales. Le choix d'une architecture adéquate est un facteur tout aussi important. Certains éléments d'un réseau de neurones reste facile à déterminer comme le nombre d'entrées, ainsi que le nombre de sorties.

a) **Réseaux de neurones monocouche (perceptron simple):**
b) **Réseaux de neurones multi-couches (perceptron multicouches MLP):**



a) réseau de neurone mono-couche b) réseau de neurone multi-couches

Figure II. 6: Schéma d'un réseau de neurone.

La question qui se pose souvent est comment définir l'architecture d'un réseau étant donné la multitude de configurations possibles de nombre de couches cachées et de neurones artificiels par couche à utiliser ? Le problème de la conception d'un MLP réside toutefois dans le fait qu'aucune méthode théorique n'est disponible pour déterminer de manière optimale la structure du réseau, le nombre de couches cachées et les nombres de neurones dans chaque couche cachée [55], qui contrôlent la capacité d'apprentissage. Souvent, pour décider combien de couches cachées, nous pouvons utiliser une technique qui consiste à commencer avec un petit nombre de couche cachée et de l'adapter jusqu'à trouver une architecture satisfaisante. Choisir le bon nombre de nœuds à utiliser par couche est difficile et il s'agit d'un des éléments qui font que le réseau peut apprendre correctement ou non.

D'après Jeff Heaton [55], le nombre de neurones d'une couche cachée doit rester entre le nombre de neurones de la couche d'entrée et du nombre de neurones de la couche de sortie.

Finalement, afin de trouver la meilleure configuration, nous devons faire plusieurs tests car il n'y a pas de règle précise qui nous indique comment créer l'architecture d'un réseau à partir d'un problème donné.

c) **Réseaux de neurones récurrents (Recurrent Neural Network - RNN) :** Ont une topologie de connexions quelconque, comprenant notamment des boucles dont le sens de propagation du flux de données reste aléatoire [56,57]. Il s'agit donc de réseaux de neurones avec une contre réaction (voir Figure II.7).

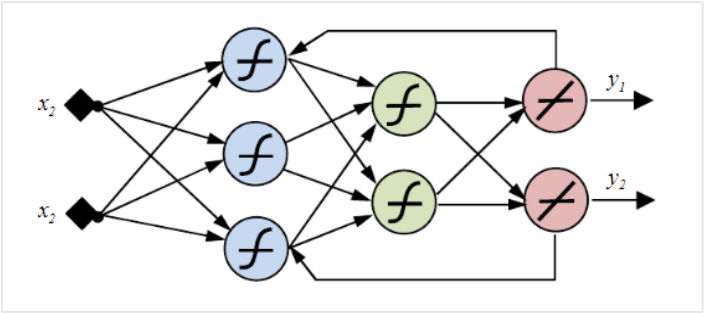


Figure II. 7: Schéma de réseau de neurones récurrent.

d) Réseaux de neurones à connexions locales : Il s'agit aussi d'un réseau multicouches‹ ou chaque neurone entretient des relations avec un nombre réduit et localisé de neurones de la couche avale. Les connexions sont donc moins nombreuses que dans le cas d'un réseau MLP(Figure II.8).

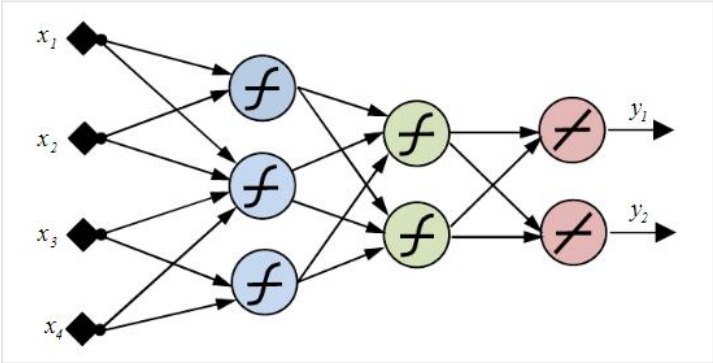


Figure II. 8: Schéma d'un réseau de neurones à connexions locales.

II.3.4 Développement d'un réseau de neurones :

Procédure de développement d'un réseau de neurones :

a) Choix d'une base de données : Afin de développer une application à base de réseaux de neurones‹ il est nécessaire de disposer de deux bases de données : une base pour effectuer l'apprentissage et une autre pour tester le réseau obtenu et déterminer ses performances. L'objectif de cette étape est donc de recueillir et de rassembler un nombre de données suffisant pour constituer une base représentative des données susceptibles d'intervenir en phase d'utilisation du système neuronal. Il est souvent préférable d'effectuer une analyse des données de manière à déterminer les caractéristiques discriminantes pour détecter ou différencier ces données. Cette détermination des caractéristiques a des conséquences à la fois sur les performances du réseau, et sur le temps d'apprentissage.

b) **Choix d'une architecture adéquate :** La fonction désirée، la nature des données à traiter، ressources matérielles/logicielles، contraintes temps-réel، les efforts de préparation de la base d'apprentissage/tests، délais d'apprentissage.

c) **Apprentissage du réseau :** L'apprentissage est une phase de développement du réseau de neurones durant laquelle le comportement de ce dernier est modifié jusqu'à l'obtention du comportement désiré. Initialisés au départ, les poids synaptiques et biais du réseau sont actualisés au cours de la phase d'apprentissage de la dernière couche vers la première afin d'essayer de minimiser la fonction de coût noté J sur une base de données d'apprentissage. J représente les erreurs faites par le modèle, en d'autres termes elle mesure l'écart entre la prédiction attendue et la prédiction effectivement donnée par le réseau. Le mécanisme d'apprentissage consiste alors à voir itérativement comment l'erreur évolue en fonction de chaque pondération, puis rétro propager cette erreur et d'optimiser au mieux cette rétro-propagation. Pour cela, nous utilisons un algorithme regroupant les étapes énumérées sur le graphe de la figure ci-dessous :

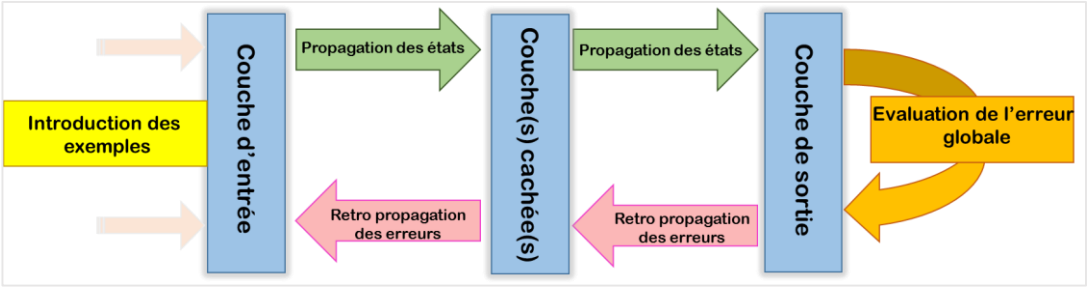


Figure II. 9: Apprentissage des réseaux de neurones par l'algorithme de rétro propagation.

1) Initialisation des poids du réseau :

Les poids d'un réseau de neurones, sont l'ensemble des connexions synaptiques existantes, qui correspondent à la connexion entre deux neurones artificiels. Comme le montre la figure ci-dessous, le poids w_{ij}^{l} relie le $i^{\text{ième}}$ neurone de la $l^{\text{ième}}$ couche au $j^{\text{ième}}$ neurone de la $(l-1)^{\text{ième}}$ couche [58]. Le choix des poids initiaux des couches cachées du réseau est un point fondamental de la phase d'apprentissage, notamment dans le contrôle de la vitesse de convergence de l'algorithme d'apprentissage [58].

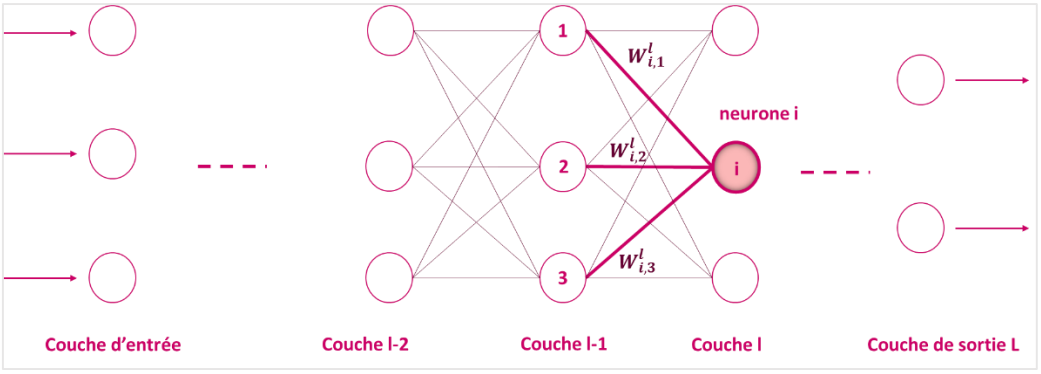


Figure II. 10 : Les poids des connexions synaptique du $i^{ième}$ neurone de la $l^{ième}$ couche issues du $j^{ième}$ neurone de la $(l-1)^{ième}$ couche précédente (en rouge) [58].

Il existe une variété de méthodes utilisées pour l'initialisation des paramètres du réseau on peut citer par exemple :

- **Initialisation zéro :** Les poids de tous les neurones sont mis à zéro. L'ensemble du réseau équivaut à un seul neurone et la prédiction sera aléatoire, ce qui ne servira à rien.
- **Initialisation aléatoire :** Les poids sont considérés comme des distributions gaussiennes N(0,1). Cela aide à briser la symétrie et chaque neurone n'effectue plus le même calcul [59].
- **Initialisation de Xavier :** les poids sont considérés comme des distributions gaussiennes N(0,1/n). La variance d'un nœud donné dépend de la variance des poids de la couche précédente. n est le nombre de neurones entrants [59].

2) Propagation avant :

La propagation avant désigne que l'information ne se déplace que dans une seule direction, vers l'avant, à partir des nœuds d'entrée, en passant par les couches cachées et vers les nœuds de sortie. La propagation avant peut être formulée à l'aide de la fonction d'agrégation z^l et la fonction d'activation σ :

$$a^l = \sigma(z^l) \quad (2.2)$$

$$z^l = w^l * a^{l-1} + b^l, \quad (2.3)$$

Avec w^l est une matrice de taille i x j (i lignes et j colonnes) qui contient tous les poids de la $l^{ième}$ couche. b^l est le vecteur qui contient tous les biais de la $l^{ième}$couche. $z^l = (z_1^l, z_2^l, \dots\dots, z_n^l)$ et $a^l = (a_1^l, a_2^l, \dots\dots, a_n^l)$, a_i^l est la valeur d'activation du $i^{ième}$ neurone de la $l^{ième}$couche, donnée par $a_i^l = \sigma(z_i^l)$. z_i^l est la valeur d'agrégation du $i^{ième}$ neurone de la $l^{ième}$couche donnée

par $z_i^l = \sum_{j=1}^{n} w_{ij}^{l-1} a_j^{l-1} + b_i^{l-1}$. w_{ij}^l est le poids qui relie le $i^{\text{ième}}$ neurone de la $l^{\text{ième}}$couche au $j^{\text{ième}}$ neurone de la $(l-1)^{\text{ième}}$ couche. b_i^l est le biais associé au $i^{\text{ième}}$ neurone de la $l^{\text{ième}}$couche.

L'information de sortie du $i^{\text{ième}}$ neurone de la $l^{\text{ième}}$ couche se calcule à l'aide des informations d'entrée de ce neurone qui sont donc les informations de sortie des neurones de la $(l-1)^{\text{ième}}$ couche qui sont connectés à lui, ainsi qu'à l'aide des poids des connexions synaptiques mises en jeu.

3) Rétro-propagation du gradient de l'erreur :

L'algorithme de rétro-propagation du gradient (back-propagation) permet de calculer l'erreur pour chaque neurone, de la dernière couche vers la première, et ce pour tous les paramètres du réseau. Cet algorithme se déroule en deux temps :

1. Calculer l'erreur $\delta^{(sortie)}$ en comparant la sortie avec le résultat attendu.

2. Rétro-propager l'erreur des couches supérieurs vers les couches antérieures.

3.1 Calculer l'erreur pour la couche de sortie

Pour calculer l'erreur de la couche de sortie $\delta^{(sortie)}$ nous utilisons le théorème de la dérivation des fonctions composées (ou règle de dérivation en chaîne). Pour mesurer l'erreur de prédiction, on utilise la fonction de coût J. Grâce à l'algorithme de rétro-propagation du gradient, nous pouvons savoir chaque erreur individuelle $\frac{\partial J}{\partial w_{ij}^l}$ et $\frac{\partial J}{\partial b_i^l}$ de tous les poids et biais du réseau et leur degré de contribution à l'erreur finale $\delta^{(sortie)}$. Concernant les poids, on peut écrire pour la couche de sortie L du réseau :

$$\frac{\partial J}{\partial w_{ij}^L} = \frac{\partial z_i^L}{\partial w_{ij}^L} \frac{\partial a_i^L}{\partial z_i^L} \frac{\partial J}{\partial a_i^L} \tag{2.4}$$

$\frac{\partial J}{\partial a_i^L} = J'(a_i^L)$: Est la variation de la fonction J en fonction de la sortie du réseau.

$\frac{\partial a_i^L}{\partial z_i^L} = \sigma'(z_i^L)$: Est la variation de la fonction d'activation en fonction de l'agrégation.

$\frac{\partial z_i^L}{\partial w_{ij}^L} = a_j^{L-1}$: Est la variation de la fonction d'agrégation en fonction d'un seul poids w_{ij}.

Ainsi, on peut écrire l'équation suivante :

$$\frac{\partial J}{\partial w_{ij}^{L}} = a_j^{L-1} * \sigma'(z_i^L) * J'(a_i^L). \tag{2.5}$$

Ou $\sigma'(z_i^L) * J'(a_i^L)$ n'est que $\frac{\partial a_i^L}{\partial z_i^L}\frac{\partial J}{\partial a_i^L} = \frac{\partial J}{\partial z_i^L} = \delta_i^L$. Par conséquent, l'erreur calculée pour la couche de sortie est par donnée par :

$$\frac{\partial J}{\partial w_{ij}^{L}} = a_j^{L-1} * \delta_i^L \tag{2.6}$$

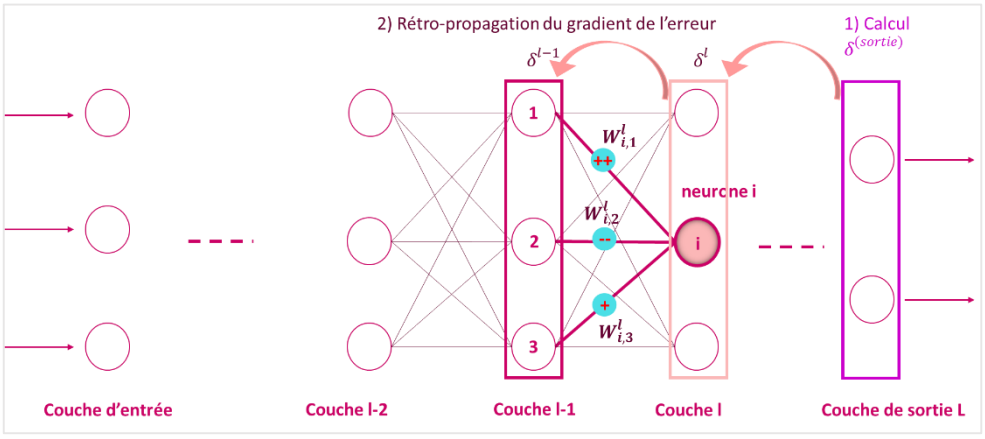


Figure II. 11: Schéma de la rétro-propagation du gradient de l'erreur [58].

La figure ci-dessus représente le schéma de la rétro-propagation du gradient de l'erreur depuis la couche de sortie jusqu'aux poids des connexions synaptique du $i^{ième}$ neurone de la $l^{ième}$ couche issues du$j^{ième}$ neurone de la $(l-1)^{ième}$ couche précédente (en rouge).

3.2 Rétro-propager l'erreur :

La variation de la fonction de coût de la sortie du $i^{\text{ième}}$ neurone de la $l^{\text{ième}}$ couche peut s'écrire comme [58] :

$$\frac{\partial J}{\partial a_i^l} = \sum_K \frac{\partial z_k^L}{\partial a_k^l}\frac{\partial J}{\partial z_k^L} \tag{2.7}$$

Avec $\frac{\partial J}{\partial z_k^L} = \delta_k^L$ et $\frac{\partial z_k^L}{\partial a_k^l} = w_{ki}^L$

L'algorithme de rétro-propagation décide dans quelle mesure chaque pondération du réseau doit être mise à jour après avoir comparé la sortie prévue à la sortie souhaitée pour un exemple précis. Pour cela, nous devons calculer comment l'erreur évolue en fonction de chaque

pondération. On peut résumer les équations nécessaires à l'implémentation de l'algorithme de rétro-propagation du gradient [58] :

$\delta_i^L = \sigma'(z_i^L) * J'(a_i^L)$ pour la couche de sortie *L*.

$\delta_i^l = \sigma'(z_i^L) * \sum_j w_{ji}^{l+1}\ \delta_i^{l+1}$ pour les couches cachées *l* du réseau.

$\frac{\partial J}{\partial w_{ij}^l} = a_j^{l-1} * \delta_i^l$ pour les poids.

$\frac{\partial J}{\partial b_i^l} = \delta_i^l$ pour les biais.

4) Mise à jour des poids du réseau :

Le processus d'apprentissage revient à entrainer le réseau neuronal dont le but est de minimiser la fonction coût à de très faibles valeurs, en utilisant l'algorithme itératif d'optimisation par la descente du gradient. Cet algorithme vise à s'approcher d'un minimum global où le gradient de la fonction de coût est nul. Après une descente de gradient, les poids synaptiques qui contribuent à engendrer une erreur importante sont modifiés de manière plus significative que les poids qui engendrent une erreur marginale, et le modèle est mis à jour selon les équations :

$$w_{ij}^l = w_{ij}^l - \alpha \frac{\partial J}{\partial w_{ij}^l} \quad (2.8)$$

$$b_i^l = b_i^l - \alpha \frac{\partial J}{\partial b_i^l} \quad (2.9)$$

où α est appelé le taux d'apprentissage et contrôle la vitesse d'apprentissage. Un taux trop élevé pourrait nuire à l'algorithme qui sauterait par-dessus le déplacement idéal.

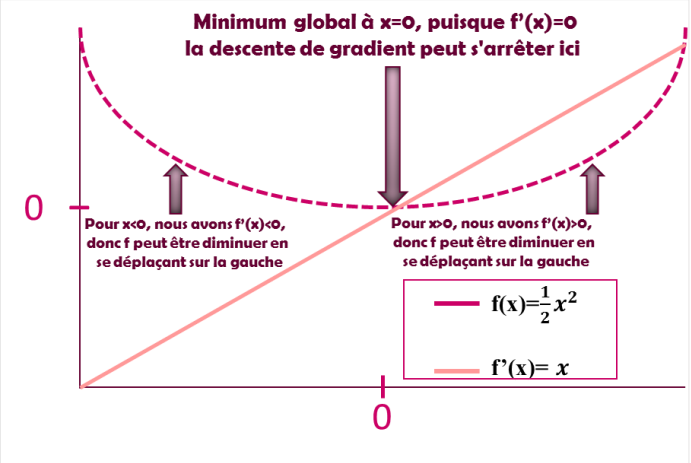


Figure II. 12 : Illustration d'une façon pour l'algorithme de descente de gradient de guider une fonction jusqu'à un minimum via le calcul de sa dérivée par exemple.

Il existe trois principaux types de variantes de l'algorithme de descente de gradient. La principale différence entre eux est la quantité de données que nous utilisons lorsque nous calculons le gradient pour chaque étape d'apprentissage.

- **De manière globale (batch gradient) :** utilise toutes les données en une seule fois, pour calculer le gradient et ajuster les poids. Le terme "batch" désigne un groupe d'exemples
- **Par des lots (mini-batch gradient) :** on envoie au réseau les données par petits groupes d'une taille définie par l'expérimentateur dont on calcule les erreurs, puis l'erreur moyenne est calculée et enfin les poids sont mis à jour.
- **De façon unitaire, stochastique (stochastic gradient) :** On envoie une seule donnée à la fois au réseau et donc les poids sont mis à jour à chaque fois juste après.

d) Validation:

En général, dans un réseau de neurones, le plus important est sa capacité de généralisation. Pour atteindre le meilleur modèle, la fonction de coût de l'ensemble de validation (créé) doit diminuer, et dire que l'apprentissage continue. Dès que la fonction de coût commence à augmenter, nous nous arrêtons (voir Figure II.13). Il existe différentes approches de validation dont :

- **Validation croisée :** phase apprentissage/test répétée avec un mélange arbitraire des données.

- **Validation par blocs :** découpage des données d'apprentissage en « n » blocs distincts, chaque bloc va être utilisé une seule fois pour la phase de test et se retrouva dans l'ensemble d'apprentissage le reste du temps.
- **validation jackknife (leave-one-out) :** entrainer le modèle sur l'ensemble des données d'apprentissage sauf un sur lequel on mesure l'erreur de test. On répète l'opération pour toutes les données restantes.

II.3.5 Fonction de coût :

La fonction de coût assume un rôle actif et décisif pendant l'apprentissage du réseau. Il est important d'utiliser la fonction la plus appropriée et de sélectionner les bons paramètres pour cette dernière. La convergence du réseau, les performances et le résultat de l'apprentissage, seront affectés par la non utilisation de la fonction appropriée ou ses paramètres ne sont pas applicables. Plusieurs fonctions de coût ont été proposées :

Acronyme	Nom	Equation
$\mathcal{L}_1$	Fonction de Perte de type L_1	$\|y - o\|_1$
$\mathcal{L}_2$	Fonction de perte de type Perte L_2	$\|y - o\|_2^2$
$\mathcal{L}_1 \circ \sigma$	Perte d'expectation	$\|y - \sigma(o)\|_1$
$\mathcal{L}_2 \circ \sigma$	Perte d'expectation régularisée	$\|y - \sigma(o)\|_2^2$
$\mathcal{L}_\infty \circ \sigma$	Perte de Chebyshev	$\max_j \|\sigma(o)^{(j)} - y^{(j)}\|$
Hinge	Perte de charnière	$\sum_j \max(0, \frac{1}{2} - \hat{y}^{(j)} o^{(j)})$
$hinge^2$	Perte de charnière carrée	$\sum_j \max(0, \frac{1}{2} - \hat{y}^{(j)} o^{(j)})^2$
$hinge^3$	Perte de charnière cubique	$\sum_j \max(0, \frac{1}{2} - \hat{y}^{(j)} o^{(j)})^3$
Log	Entropie croisée	$-\sum_j y^{(j)} \log \sigma(o)^{(j)}$
log^2	Entropie croisée carrée	$-\sum_j [y^{(j)} \log \sigma(o)^{(j)}]^2$

Tan	Perte Tanimoto	$\frac{-\sum_j \sigma(\mathrm{o})^{(j)}\mathrm{y}^{(j)}}{\|\sigma(\mathrm{o})\|_2^2 + \|\mathrm{y}\|_2^2 - \sum_j \sigma(\mathrm{o})^{(j)}\mathrm{y}^{(j)}}$
$\mathrm{D_{cs}}$	Divergence de Cauchy-Schwarz	$-\log\frac{\sum_j \sigma(\mathrm{o})^{(j)}\mathrm{y}^{(j)}}{\|\sigma(\mathrm{o})\|_2\|\mathrm{y}\|_2}$

Table II. 2: Listes de quelques fonctions de coût utilisées pour la classification [60].

y représente la sortie désirée, o représente la sortie obtenue, σ(.) indique une estimation de probabilité.

II.3.6 Convergence de l'apprentissage :

Un problème récurrent revient lors de l'apprentissage des réseaux neuronaux : le sur-apprentissage (overfitting), se retrouve dans le cas où la base de données n'est pas assez riche. Le modèle commence à apprendre par cœur les données, menant à des difficultés à généraliser sur de nouvelles données dont le coût global est faible.

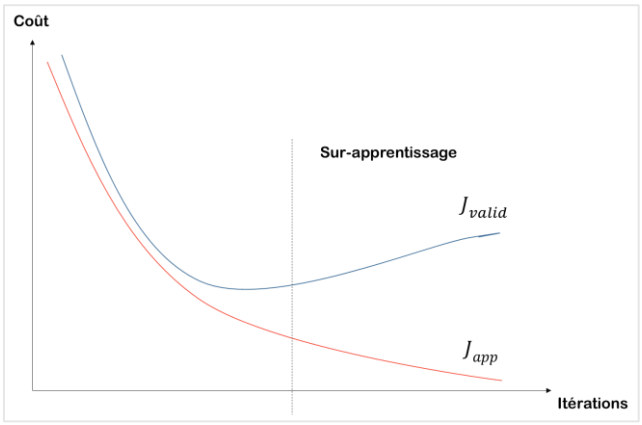


Figure II. 13: Evolution de la fonction de coût [61].

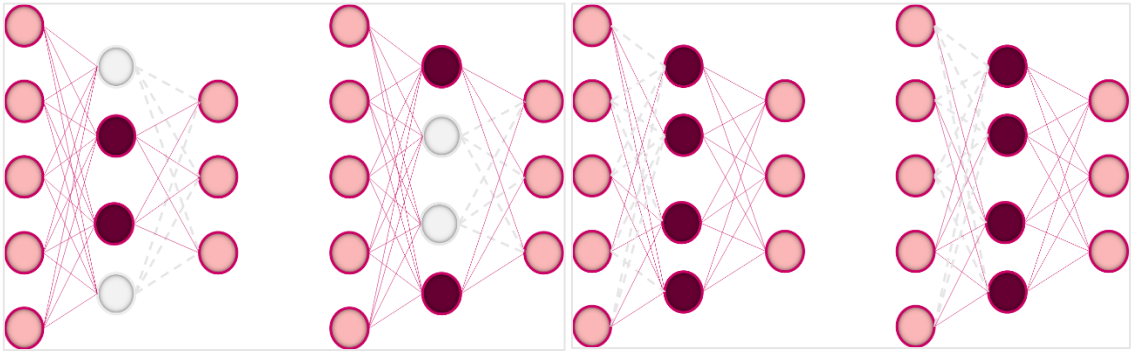
La fonction de coût est calculée sur l'ensemble d'apprentissage J_{app} (courbe en rouge) et sur l'ensemble de validation J_{valid} (courbe en bleu) en fonction du nombre d'itérations.

Plusieurs solutions ont été proposées pour remédier à ce problème. Une méthode consiste à augmenter la variété des exemples auxquels le modèle est exposé. On parle alors d'augmentation de données. Une autre méthode classique : La dégradation des pondérations (weight decay) [62], qui consiste à limiter l'amplitude des poids des connexions du réseau.

Plus récemment, Le Dropout [63] [64] a été proposé pour lutter contre le problème de sur-apprentissage. L'idée clé est de supprimer temporairement des unités (cachées et visibles) ainsi que leurs connexions entrantes et sortantes du réseau pendant la phase d'entrainement (voir

Figure II.13). Le Dropout permet de générer un grand nombre de sous-réseaux entraînés en parallèle, un réseau avec N unités peut être vu comme un ensemble de 2^N sous-réseaux. Lors de la phase de test, un seul réseau est utilisé, correspondant à la moyenne de ces réseaux réduits.

DropConnect [65], consiste à supprimer un sous-ensemble de pondérations plutôt que des neurones. Cette approche rend les couches entièrement connectées, mais faiblement. Il permet cependant de produire encore plus de modèles possibles.



a) Dropout b) DropConnect

Figure II. 14: Un exemple de réseau réduit.

.

Batch Normalization (la normalisation par lot) [66], permet d'éviter le problème de Covariance shift en normalisant les activations de chaque couche (en transformant les entrées en moyenne de 0 et en variance unitaire). Ceci permet à chaque couche d'apprendre sur une distribution plus stable des entrées.

II.4 Réseaux de neurones profonds :

L'entrainement des algorithmes de classification cités précédemment, ainsi que les réseaux de neurones peu profonds (avec une seule couche cachée) requirent un ensemble de caractéristiques extraites manuellement à partir de données brutes en utilisant par exemple des descripteurs tel que SIFT, HOG et mises sous formes de vecteur a une dimension 1D. L'extraction de ces caractéristiques demande de bonnes connaissances sur ces données brutes et sur la tâche d'apprentissage, ainsi qu'un travail d'ingénierie pour adapter les méthodes d'extraction. Cette opération est relativement coûteuse à la mise en place, ainsi qu'une mauvaise extraction des caractéristiques mène à de très mauvaises performances en termes d'apprentissage. L'idée des architectures profondes consiste alors à intégrer cette extraction de

caractéristiques, normalement faite "à la main", par un processus d'apprentissage dans les premières couches du réseau de neurones, en traitant ainsi des données a deux dimensions 2D , à savoir des images, plutôt que des données à une dimension 1D. Cela met en jeu une capacité à traiter de grandes quantités d'information. Le terme profond réfère donc au nombre de couches (plus de 2 couches cachées) des réseaux de neurones profonds entre l'entrée et la sortie.

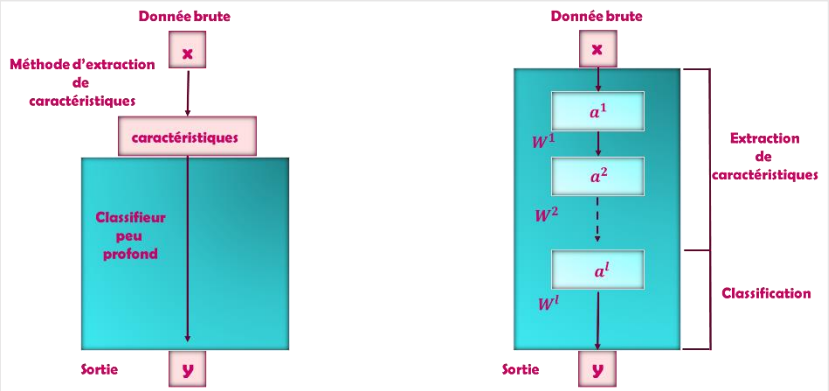


Figure II. 15: La différence entre l'apprentissage automatique classique (à gauche) et l'apprentissage profond (à droite). La zone en bleu est la zone d'apprentissage [82].

II.4.1 Les réseaux de neurones convolutifs :

Le réseau neuronal convolutif s'inspire des travaux de Hubel et Wiesel [67] sur les cortex visuels des chats et des singes. En 1980, Fukushima introduit le Neocognitron [68] un réseau hiérarchique composé de nombreuses couches qui permettent d'extraire à partir des images des caractéristiques robustes aux légères déformations. En 1989, LeCun et al. [69] proposent une architecture de perceptron multi-couches dont la première couche est convolutive, entraîné par rétro-propagation. Sur ce principe, ils élaborent ensuite l'architecture LeNet-5 [70], premier réseau de neurones convolutif moderne dédié spécifiquement à la classification d'images de chiffres manuscrits la base de données MNIST. Malgré les succès, des ConvNets, ils ont été en grande partie délaissés en vue de leurs architectures, en particulier concernant leur profondeur. Cette limitation s'est effondrée en 2010 avec le début de la compétition de reconnaissance d'objets ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). À cette période, on observe en effet une disponibilité accrue de puissance de calcul et une disponibilité assez nouvelle de grands volumes de données. Un million d'images sont annotées pour mille classes d'intérêt différentes. C'est ainsi qu'en 2012 la compétition est remportée par Krizhevsky, Sutskever et Hinton, qui ont proposés AlexNet [71] un réseau de neurones convolutif largement inspiré par LeNet-5 constitué de plusieurs couches convolutives dont la particularité d'utilisation a été de le faire tourner sur des processeurs graphiques(GPU) capables de puissants

calculs matriciels. Le succès des réseaux convolutifs profonds depuis 2012 est donc dû à la convergence de trois facteurs : des avancées théoriques (ReLU, réseaux convolutifs) permettant d'entraîner des réseaux plus profonds, la mise à disposition de grandes bases de données annotées pour l'apprentissage et des implémentations efficaces sur GPU rendant les temps de calcul acceptables. Aujourd'hui les CNN sont les plus utilisés dans de nombreuses applications tel que la reconnaissance des panneaux de signalisation [72][73], reconnaissance des expressions faciales en temps réel [74] etc…. on vue de leurs architectures profondes et complexes qui permettent d'apprendre des concepts simples dans les couches inférieures et des motifs plus abstraits dans les couches supérieures.

II.4.2 Structure des réseaux de neurones convolutifs :

Les réseaux de neurones convolutifs sont à ce jour les modèles les plus performants et les plus répandus. Ils sont conçus en particulier pour traiter les données avec une topologie spatiale (par exemple, des images statiques). Comme leur nom l'indique les CNN sont simplement définis comme des réseaux de neurones artificiels qui utilisent une opération de convolution pour connecter les neurones d'une paire de ses couches consécutives. Un CNN classique est généralement composé de quatre types de couches :

— les couches convolutives, qui contiennent plusieurs opérations de convolutions appliquées sur la même entrée.

— les couches d'opérations de mise en commun.

— les couches d'activations.

— les couches toutes connectées.

a) La couche de convolution:

Prenons l'exemple d'un perceptron multi-couches que l'on souhaite utilisé sur des images de taille 28x28 (784 pixels pour uniquement une image). Une phase de prétraitement est nécessaire, qui consiste à extraire manuellement des caractéristiques et les mettre sous forme de vecteur. En faisant cela nous n'exploitons pas les informations spatiales des pixels (quel pixel se situe où dans l'image) et le réseau ne possède pas les informations de localisation (il ne sait pas quels sont les voisins d'un pixel). Vu le nombre élevé de pixels, la première couche utilise de nombreux neurones, ce qui entraîne une augmentation importante du nombre de paramètres du réseau.

L'idée est donc de remplacer les premières couches d'un réseau de neurones par des couches convolutives. La couche de convolution est l'élément central des CNN. Son objectif est de détecter la présence de caractéristiques (features) dans les images. Elle reçoit en son entrée une image découpée en petites zones appelées champ récepteurs. Ce découpage permet d'extraire les valeurs de pixels corrélé localement. Les neurones de la couche de convolution sont regroupés formant un filtre d'une taille donnée que nous glissons sur l'image d'entrée et calculons ainsi le produit de convolution entre le filtre et chaque portion de l'image balayée. A chaque convolution on obtient à la sortie une carte caractéristique (features map) qui représente l'image d'entrée mais contenant uniquement les caractéristiques que l'on souhaite que notre réseau apprend. A noter que Les features maps sont de dimension inferieur de celle de l'image d'entrée. Pour simplifier le modèle, les poids sont partagés, tous les neurones calculent la même convolution. Cela permet de détecter les caractéristiques quelle que soit leur position dans le champ visuel. Plusieurs convolutions peuvent être calculées en parallèle. Les noyaux de convolution sont optimisés durant l'apprentissage.

En pratique, une couche de convolution d'un réseau de neurones est paramétrée par :

- la profondeur de la couche (le nombre de noyaux de convolution, ou nombre de neurones associés à un même champ récepteur).
- Le nombre C de convolutions parallèles, qui définit le nombre de cartes d'activations en sortie de couche.
- le pas : il contrôle le chevauchement des champs récepteurs. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.
- le remplissage à zéro (zero padding) : parfois, il est commode de mettre des zéros à la frontière du volume d'entrée. Cela contrôle la dimension spatiale du volume de sortie. Parfois il est souhaitable de conserver la même surface que celle du volume d'entrée.

L'intérêt de la convolution dans les réseaux de neurones profonds est triple [75] :

- Les interactions convolutives sont parcimonieuses, la taille des noyaux de convolution étant très faible devant la taille des images.
- Les caractéristiques extraites par convolution sont équivariantes aux translations de l'image ; une translation de l'image d'entrée translate les cartes d'activation de la même façon.

- Les paramètres de la convolution sont partagés pour l'ensemble de l'image, ce qui permet de détecter les mêmes caractéristiques peu importe leur position dans l'image avec un très faible coût de stockage en mémoire des paramètres.

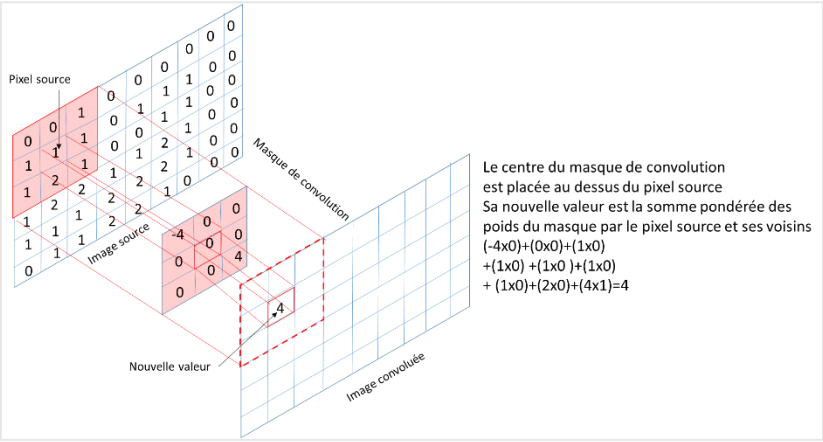


Figure II. 16: Illustration d'un opérateur de convolution. L'image source est convoluée avec un masque de convolution (ici de taille 3 × 3).

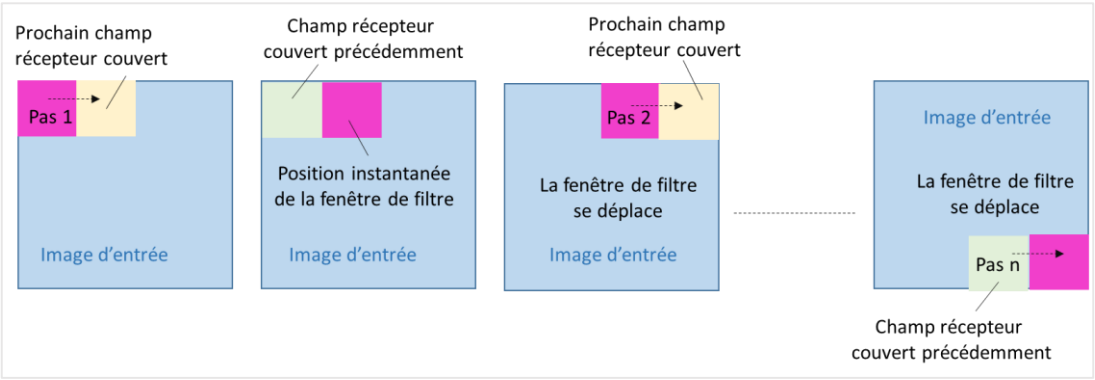


Figure II. 17: Schéma du glissement de la fenêtre de filtre sur l'image d'entrée.

b) La couche d'opération de mise en commun:

Un autre concept important des CNN est le pooling (mise en commun), ce qui est une forme de sous-échantillonnage de l'image. La couche de Pooling permet d'appliquer à chacune des cartes de caractéristiques une réduction de leur taille tout en préservant les caractéristiques les plus importantes. Il est donc fréquent d'insérer périodiquement une couche de pooling entre deux couches convolutives successives d'une architecture CNN. Le sous-échantillonnage permet de réduire le nombre de paramètres du réseau et donc les calculs nécessaires. L'opération de pooling permet d'extraire des combinaisons de caractéristiques invariantes aux décalages, aux translations et aux petites distorsions. Cela permet de simplifié la complexité du réseau, augmenter la généralisation et réduire le phénomène de sur-apprentissage. Il s'agit

généralement d'appliquer un filtre par fenêtre glissante non-recouvrante sur les données d'entrée. Ce filtre est en règle générale l'opérateur max ou l'opérateur de moyenne sur une fenêtre de taille fixe. On parle alors de max pooling ou average pooling .Un exemple de sous-échantillonnage en 2D est illustré sur la Figure II.18.

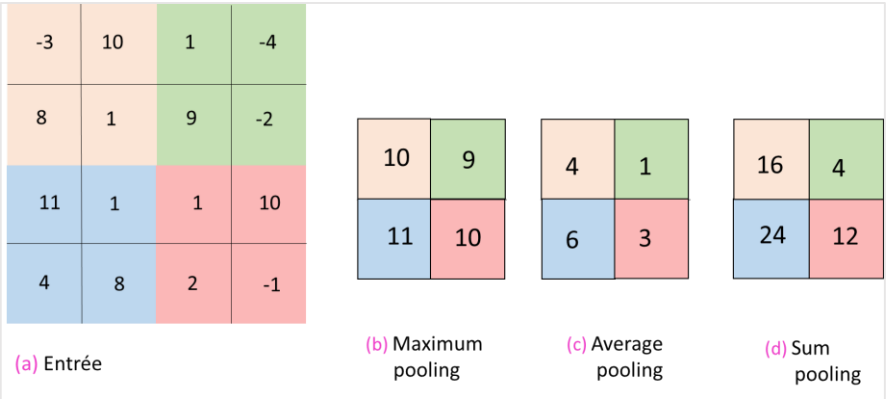


Figure II. 18: Exemples de sous-échantillonnages. De gauche à droite : l'entrée, les résultats.

- Un sous-échantillonnage par valeur maximale (figure b), appelé maximum pooling, qui consiste à récupérer la valeur maximale dans la fenêtre d'observation.
- Un sous-échantillonnage par valeur moyenne (figure c), appelé average pooling, qui consiste à calculer la moyenne des valeurs dans la fenêtre d'observation.
- Un sous-échantillonnage par somme des valeurs (figure d), appelé sum pooling, qui consiste à calculer la somme des valeurs dans la fenêtre d'observation.

c) La couche d'activation:

La couche d'activation est généralement placée après chaque couche de convolution et chaque couche entièrement connectée. Elle permet d'introduire la non-linéarité dans un système qui vient de calculer des opérations linéaires pendant la convolution. Généralement on utilise la fonction d'activation ReLu f(x) = max (0, x), appliquée par pixel et remplace toutes les valeurs négatives par des 0.

d) La couche entièrement connectée:

La couche entièrement connectée (fully-connected) ressemble à un simple perceptron multi-couches ordinaire. Elle constitue toujours la dernière couche d'un réseau de neurones, convolutif .Ce type de couche reçoit un vecteur en entrée et produit un nouveau vecteur en sortie. Pour cela, elle applique une combinaison linéaire puis éventuellement une fonction d'activation aux valeurs reçues en entrée. Le nombre de neurones de sortie de cette couche est

alors égal à N (N représente le nombre de classes ainsi que la taille du vecteur de sortie). En effet, ces couches contiennent un nombre important de paramètres et sont sensibles au sur-apprentissage. La couche entièrement connecté utilise souvent à sa sortie la fonction d'activation Softmax car elle permet de calculer la distribution de probabilité contrairement aux autres fonctions d'activation qui affichent seulement des valeurs numériques.

II.4.3 Architectures de quelques réseaux convolutifs célèbres :

Les réseaux de neurones profonds ont généralement une architecture complexe, ce qui contribue à une phase d'apprentissage intensive qui s'étale parfois sur des semaines, voire des mois .Ces défis ont menés à proposer de nouvelle architecture des CNN en améliorant leurs performances liées à la détection d'objet et la classification des images par l'optimisation des paramètres et le réajustement des connexions au sein du réseau.

Le tableau suivant résume les différentes architectures que nous allons les détaillées par la suite :

Nom de l'architecture	La date	Caractéristiques	Para-mètres	Taux d'erreur	Profondeur	Refe-rence
LeNet-5	1998	-Première architecture CNN populaire. -Extraction automatique des features. -Utiliser sur des images en niveaux de gris 32x32.	60.000	MNIST 0.95	7	[70]
AlexNet	2012	-Première architecture profonde. - Remporte la compétition ILSVRC en 2012. -Utilisé sur divers catégories d'images (niveau de gris et en couleur) 224x224. - ReLU(vanishing gradient),Dropout(sur-apprentissage). -Filtres de dimension large (11x11).	60M	Image-Net: 16.4	8	[71]

		-temps de calcul réduit (GPU).				
ZefNet	2013	-Visualiser les performances du réseau. -Operations inverses de convolution et sous-échantillonnage qui permet d'expliquer le fonctionnement du réseau. - Vainqueur d'ILSVRC en 2013. -Utilisation des filtres (7x7).	60M	Image-Net: 11.7	8	[76]
VGG-16	2014	-Filtres de dimension petite. -Réseau plus profond (environ deux fois plus profond qu'AlexNet). -Dropout pour les couches FC. - Coût de calcul élevé.	Environs 140M	Image-Net: 7.3	16	[77]
GoogLeNet (Inception v1)	2014	- Vainqueur du concours ILSVRC en 2014. -Introduire le bloc inception. - Coût de calcul réduit. -Grande précision. -Insertion des classifieurs au niveau des couches intermédiaires.	4M	Image-Net: 6.7	22	[78]
ResNet	2015	- Remporte le concours ILSVRC 2015. -Optimisation du réseau par l'usage de Batch Normalization et l'apprentissage résiduel.	6.8 M 1.7 M	Image-Net:3.6 CIFAR-10: 6.43	152 110	[79]

		-Basé sur le saut des couches. -Evite le problème du vanishing gradient. -Accélère la convergence du réseau. -Couteux en mémoire. -Peu pratique sur les images de grandes dimensions. -Utilise global AVG pooling au lieu du PMC à la fin.				
Xception	2016	- Convolution séparable en profondeur : Depthwise convolution, Pointwise convolution. - Moins de paramètres a calculé.	22.8M	Image-Net: 0.055	36	[80]
DenseNet	2017	-Utilisation des blocs denses. -Règle le problème des couches redondantes.	25.6 M 25.6 M 15.3 M 15.3 M	CIFAR -10+: 3.46 CIFAR -100+: 17.18 CIFAR -10: 5.19 CIFAR -100: 19.64	190 190 250 250	[81]

Table II. 3: Les architectures CNN les plus populaires.

a) LeNet-5 (1998) :

LeNet-5 est la première architecture CNN proposée par LeCun en 1998 [70] pour la reconnaissance de chiffres manuscrits dans des images en niveaux de gris de dimensions 32 x32. .Il a permis de montrer sa capacité à classer les chiffres sans être affecté par de petites distorsions, rotation et variation de position. LeNet-5 a non seulement réduit le nombre de paramètres et de calculs, mais a également pu apprendre automatiquement les features.

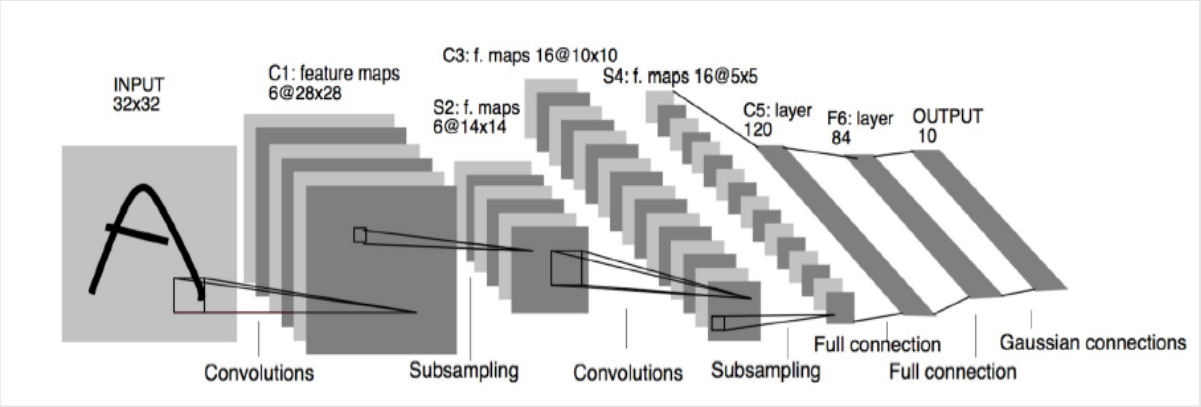


Figure II. 19: Architecture LeNet-5 [50].

b) Alex-Net (2012) :

Krizhevesky et al proposent AlexNet [71], architecture approfondie du réseau LeNet-5 applicables à diverses catégories d'images. L'apprentissage de ce réseau ne prend que près d'une semaine de calcul grâce à l'utilisation de cartes graphiques (GPU). La structure d'AlexNet est conçue de telle sorte que le nombre de carte d'activation augmente de façon inversement proportionnelle à la réduction des dimensions spatiales des images, cela permet de conserver un nombre raisonnable de paramètres à calculer.

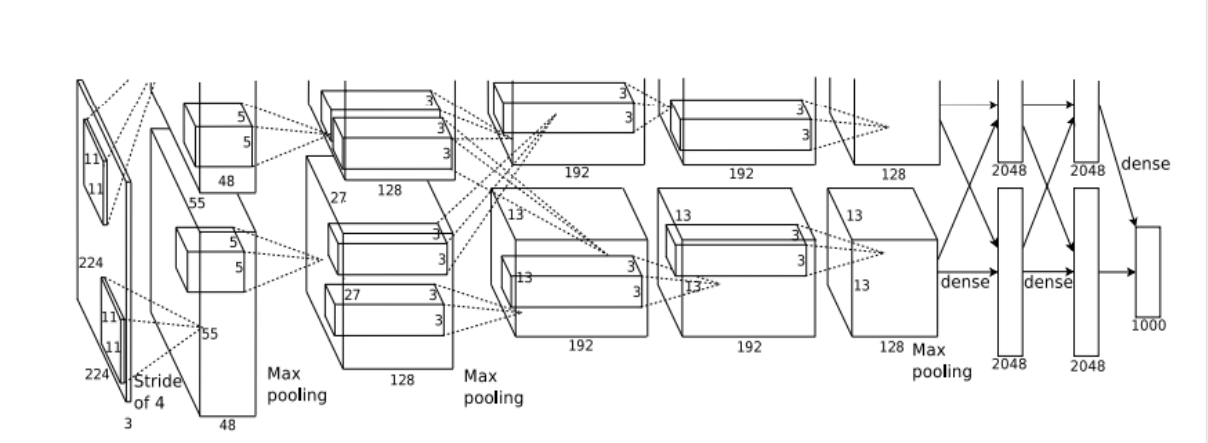


Figure II. 20 : Architecture AlexNet [71].

c) ZefNet(2013) :

Zeiler et Fegus proposent en 2013 un réseau déconvolutif multicouches (DeconvNet) [76], dont le but de visualiser et d'expliquer le fonctionnement interne des CNN. L'idée principale a été d'inverser les opérations de convolution et sous-échantillonnage. Cette approche met en évidence plusieurs propriétés : 1) les couches profondes visualisent davantage des features spécifiques (haut niveau). 2) les couches initiales ont tendance à visualiser les modèles généraux comme les bords, la texture (bas niveau). ZefNet a non seulement été le vainqueur de l'ILSVRC 2013, mais fournit également un aperçu des améliorations des architectures de réseau.

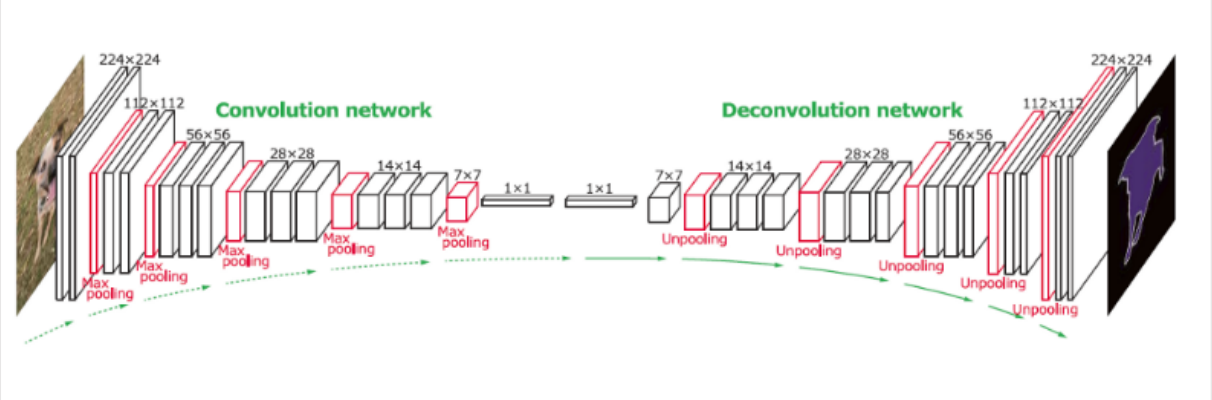


Figure II. 21: Architecture globale de DeconvNet [75].

d) VGG-16 (2014) :

Karen Simonyan et Andrew Zisserman proposent un principe de conception simple et efficace pour les architectures CNN [77]. En se basant sur la suggestion de ZefNet qui énonce que plus les filtres utilisés sont de petites dimensions, plus cela améliore les performances du CNN, VGG suggère d'utiliser plusieurs convolution successives de taille 3x3 au lieu d'une unique convolution de dimension11x11. Cela offre l'avantage de réduire la complexité de calcul en réduisant le nombre de paramètres. La principale limitation associée à VGG est celle du coût de calcul élevé en raison de l'utilisation d'environ 140 millions de paramètres.

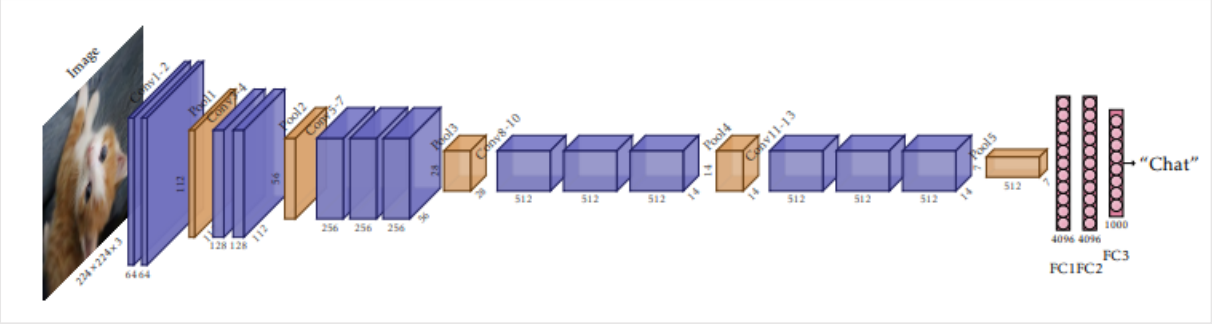
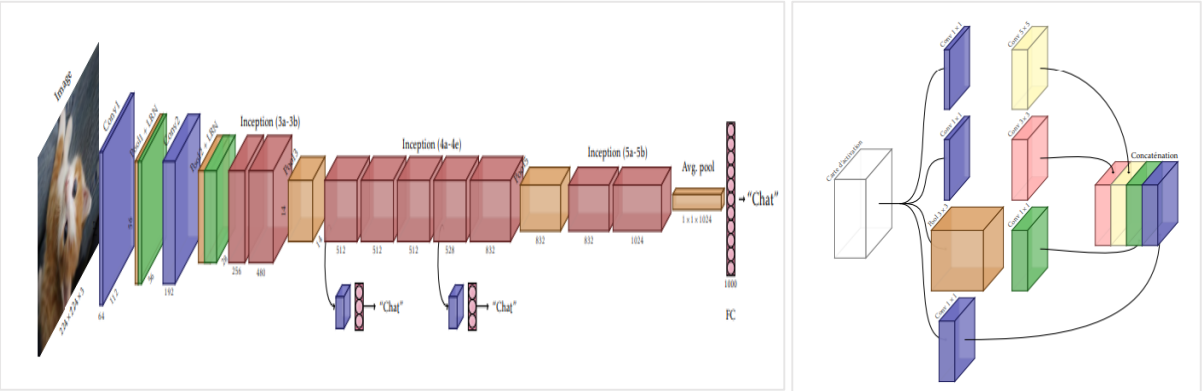


Figure II. 22: Architecture VGG-16 [75].

e) GoogleNet (Inception-v12014) :

Szegedy et al, proposent un modèle qui atteint une grande précision avec un coût de calcul réduit en introduisant le module inception [78]. Inception empile plusieurs couches de convolution de différentes tailles (3x3, 5x5) en parallèles, permettant l'extraction de caractéristiques à différentes échelles. Le module inception est utilisé afin de permettre un calcul plus efficace et des réseaux plus profonds grâce à une réduction de dimension. Ces réglages de paramètres ont provoqués une diminution significative de nombre de paramètres mais peuvent parfois entrainer une perte d'informations utiles.



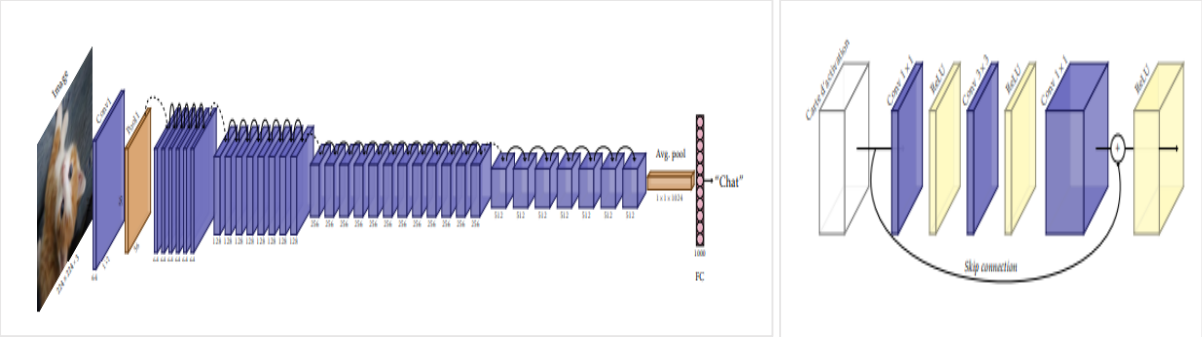
a) GoogleNet b) Module Inception

Figure II. 23: Architecture GoogleNet [75].

f) ResNet (2015) :

En 2015**,** He et al proposent un réseau très profond optimisé par l'insertion des blocs résiduels [79]. Un bloc résiduel sert a brisé la structure séquentielle des réseaux profonds en insérant des connexions dites connexions résiduelles comme raccourci qui permettent de sauter par-dessus certaines couches. Cette approche permet de propager les cartes d'activation intermédiaires aux couches supérieures, garantissant ainsi d'éviter le problème du « vanishing gradient », et

d'accélérer la convergence des réseaux profonds. L'inconvenient de cette approche est le nombre élévée d'activation et de gradients intermédiares a calculer rend les ResNet couteux en mémoire et peu pratiques sur des images de grandes dimensions.



a) ResNet-34 b) Bloc Résiduel

Figure II. 24: Architecture ResNet [75].

g) Xception (2016) :

Introduit par François Chollet [80], est considéré comme une architecture inception extrême qui exploite l'idée de convolution séparable en profondeur effectuée en deux étapes, une Depthwise convolution suivie d'une Pointwise convolution. Le terme séparable réfère à l'indépendance entre ces deux étapes. Xception modifie le bloc inception d'origine en remplaçant les différentes dimensions spatiales des filtres (1x1, 3x3, 5x5) par une seule dimension (3x3) suivie d'une convolution 1x1 pour régulé la complexité du réseau. L'architecture Xception comprend notamment des connexions résiduelles ce qui rend l'architecture facile a modifié. En comparant la convolution classique à la convolution séparable en profondeur, cette dernière nécessite moins de paramètres a calculé ce qui diminue la complexité du réseau et rend son fonctionnement plus rapide.

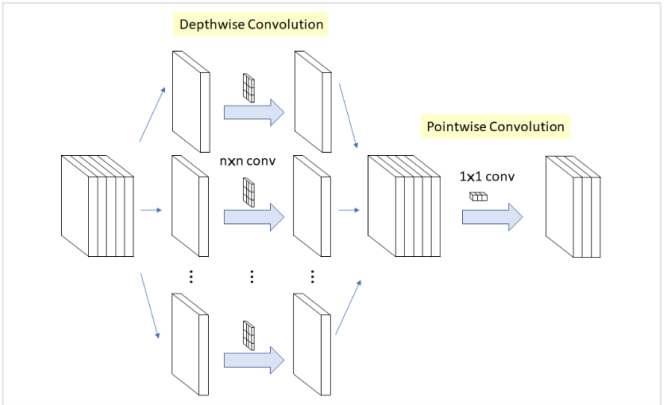


Figure II. 25: Principe de la convolution séparable en profondeur.

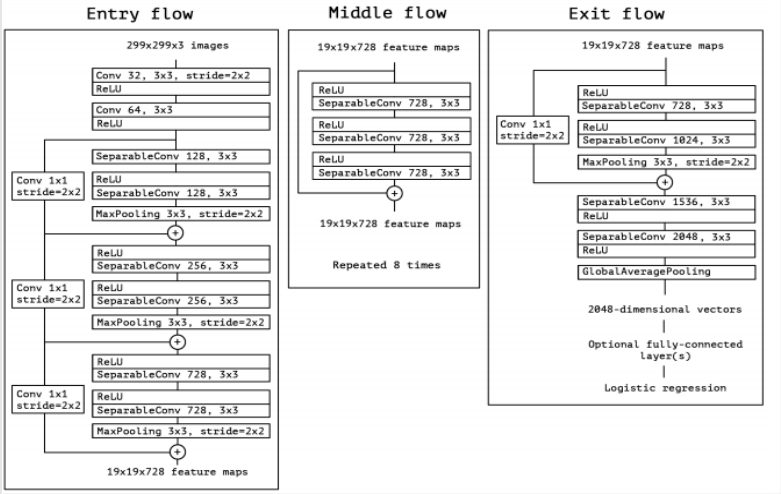
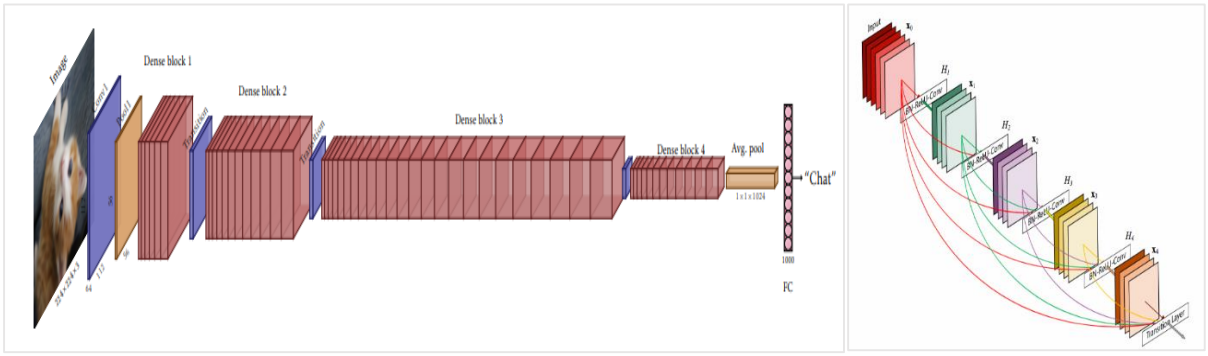


Figure II. 26: Architecture Xception.

h) DenseNet (2017):

Huang et al proposent une architecture dans laquelle les couches sont densément connectées entre elles [81] : chaque couche reçoit en entrée toutes les cartes d'activations de sortie des couches précédentes. Un bloc dense regroupe des couches connectées à toutes leurs couches précédentes. Une seule couche comporte : Batch Normalization,ReLU activation et 3x3 Convolution. Il est impossible de concaténer des activations de différentes tailles, pour cela une couche convolutive de transition (Batch Normalization, 1x1 convolution,average pooling) est appliquée entre deux blocs, pour réduire le nombre de plans et est suivie d'un max-pooling pour réduire les dimensions spatiales.



a) DenseNet b) Bloc dense

Figure II. 27: Architecture DenseNet [75].

II.5 Conclusion :

Dans ce chapitre nous avons vu les différents éléments de construction d'un réseau de neurones ainsi que les algorithmes utilisés pour les entraîner. Nous avons pu constater que le choix d'une architecture adéquate ainsi que ses paramètres demande une bonne compréhension de leur fonctionnement. Nous avons vu comment ont évolué les différentes architectures dont la tendance est d'ajouter toujours plus de couches de calculs, créant des réseaux de plus en plus profonds. Aujourd'hui, l'apprentissage profond a prouvé sa grande efficacité et a permis des avancées considérables dans de nombreux domaines. Néanmoins en vision par ordinateur l'une des raisons importantes du succès des algorithmes d'apprentissage en profondeur est l'invention du CNN. Les réseaux convolutifs sont les plus répandus et les plus utilisés pour résoudre les problèmes de classification en vue du développement du matériel, de la puissance des ressources de calcul, ainsi que l'apparition des grandes bases de données.

Chapitre III :

Détection et classification d’obstacles, résultats et interprétations.

III.1 Introduction :

Lors de l'implémentation d'un système de détection et de reconnaissance d'obstacles routiers, en vue de l'application de l'assistance à la conduite automobile, la première question qui se pose est : Comment peut-on représenter un obstacle de manière compacte et fidèle, dans le but d'être en mesure de l'identifier ?

Pour répondre à la problématique de reconnaissance d'objets dans les images, nous avons centré notre réflexion autour de deux axes principaux : la représentation et la classification.

Pour ce qui est de la représentation, elle consiste à définir une structure qui modélise la forme de l'objet, par exemple un ensemble de points, une fenêtre englobante, un contour (Figure III.1). Notre travail consiste à représenter et modéliser les obstacles routiers par des contours en utilisant la méthode « Level-Set » [3]. Cette étape va permettre une détection très précise des obstacles.

Pour ce qui est de la classification, elle consiste à attribuer à un objet x un label y à partir d'un ensemble fixe de classes. Afin d'attribuer l'étiquette adéquate, il est nécessaire de calculer un ensemble de caractéristiques pour bien distinguer un objet des autres. Cette caractérisation consiste, d'une façon générale, à extraire des descripteurs de forme, de texture d'apparence ou bien d'échelle dans le but de représenter au mieux l'objet. Ensuite faire plusieurs prédictions pour l'ensemble des objets d'une image pour les identifier ce qui est très couteux en termes de calcul. Pour cette tache nous avons opté pour le réseau de neurones convolutifs YOLO v3 (You only look once version3) [83].

Nous avons proposé de combiner la méthode « Level-set » et le réseau YOLO v3 en vue d'effectuer une tache de modélisation des différents objets ensuite, effectuer la localisation, la détection et classification simultanément.

Nous décrivons en premier lieu les deux techniques dont notre travail s'appuie tout en justifiant les choix qui nous ont poussés à les choisir. La partie qui suit présente les résultats de l'application réalisée sous linux pour évaluer les performances de notre approche.

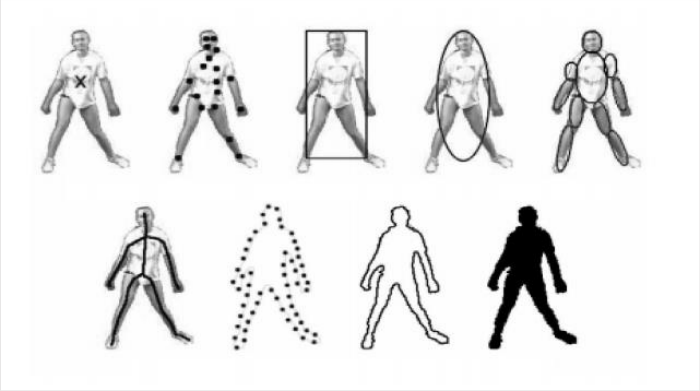


Figure III. 1: Les représentations de formes de l'objet [84].

III.2 Level-set (Ensemble des niveaux) :

Un contour actif est une courbe du plan déformable. Elle évolue d'un état initial à un état final, afin de l'adapter au contour apparent d'un objet. L'évolution de cette courbe se fait sous l'action d'une force suivant les directions normale et tangentielle et d'une vitesse, jusqu'à ce qu'elle s'aligne avec précision sur les bords de l'objet (voir Figure III.2). Contrairement aux détecteurs de contours classiques abordés précédemment qui souffrent de la non fermeture des contours, l'approche des contours actifs exploite déjà un contour fermé, la connexité n'est plus à vérifier. Cette approche est considérée comme une opération de haut niveau contrairement aux segmentations plus classiques.

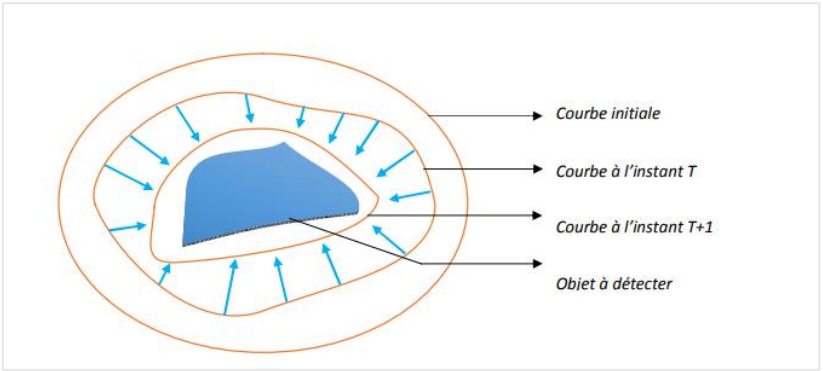


Figure III. 2: Principe des contours actifs

La théorie des Level-Set [3], est une formulation pour implémenter les contours actifs. Son principe consiste à considérer le contour$C(t)$, comme le niveau zéro d'une fonction «Level-set » (LSF) $\phi(x,t)$ (voir Figure III.3) :

$$C(t) = \{x \in \Omega | \phi(x,t) = 0\} \quad (3.1)$$

La question centrale est de savoir comment faire évoluer la fonction $\phi(x,t)$ de telle sorte que la courbe $C(t)$ suit un mouvement prescrit.

Soit le mouvement de la courbe $C(t)$ donné par :

$$\frac{\partial C(t)}{\partial t}=V\vec{N} \tag{3.2}$$

Où $\overrightarrow{N}$ est le vecteur unitaire dans la direction normale intérieure de la courbe et V indique la vitesse le long de la direction normale. Par définition, pour tout instant, la fonction $\phi(x,t)$ est nulle en tous points de la courbe.

$$\forall t,\text{t } \phi(C(t),t)=0 \tag{3.3}$$

En conséquence, la dérivée temporelle de cette expression doit être nulle.

$$\frac{\partial \phi(C(t),t)}{\partial t}=\nabla_{\phi}.\frac{\partial C}{\partial t}+\frac{\partial \phi}{\partial t} \tag{3.4}$$

Nous pouvons résoudre l'évolution temporelle de ϕ en insérant (3.2) et la définition de la normale externe $N=-\frac{\nabla_{\phi}}{|\nabla_{\phi}|}$

$$\frac{\partial \phi}{\partial t}=-\nabla_{\phi}.\frac{\partial C}{\partial t}=-\nabla_{\phi}\,VN=\nabla_{\phi}V\frac{\nabla_{\phi}}{|\nabla_{\phi}|}=V\left|\nabla_{\phi}\right| \tag{3.5}$$

La dérivation ci-dessus montre que pour une évolution de courbe avec une vitesse V dans le sens normal, la fonction ϕ au niveau zéro doit suivre l'équation :

$$\frac{\partial \phi}{\partial t}=V\left|\nabla_{\phi}\right| \tag{3.6}$$

Les courbes peuvent donc évoluer simplement en itérant cette équation différentielle appelée équation de niveaux (LSE). Pour la visualisation de la courbe, on lit simplement le niveau zéro de $\phi(x,t)$ en tout moment t. Alors que le LSE spécifie le mouvement de ϕ à la frontière (courbe), l'évolution en dehors de la localisation de la courbe peut être en principe arbitraire. Typiquement, on impose que le LSF reste une fonction de distance signée (SDF), c'est-à-dire :

$$\phi(x,t)=\pm dist(x,C) \tag{3.7}$$

Où ϕ est positif à l'intérieur et négatif à l'extérieur de la courbe.

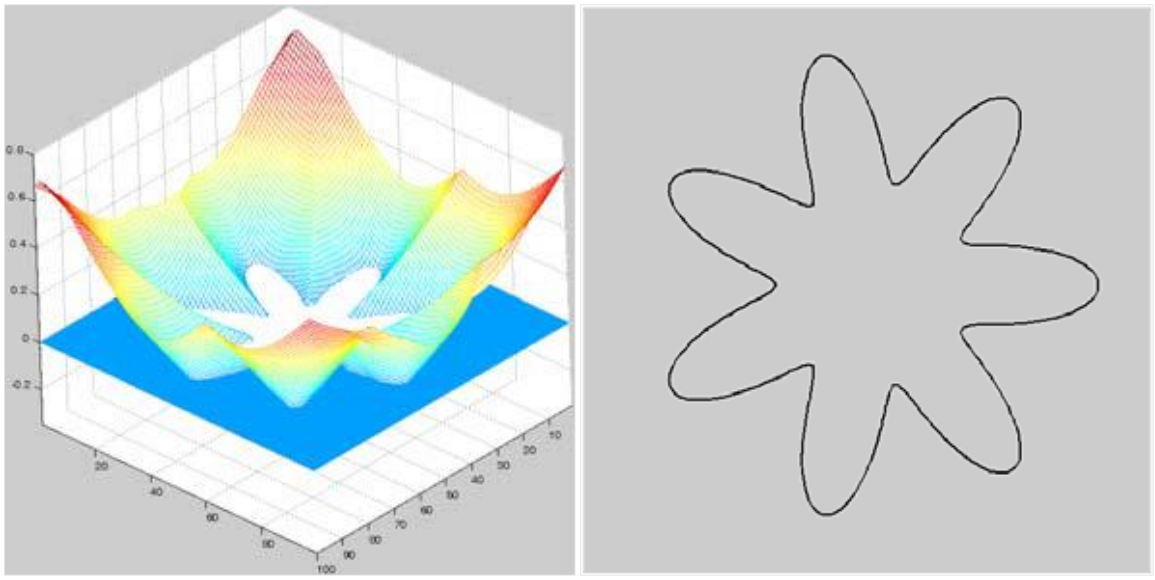


Figure III. 3: Contour correspondant au niveau zéro d'une fonction.

III.2.1 Le choix de la méthode « Level-set » se justifie par :

- Sa capacité à fournir des contours très précis, alignés avec une très grande précision sur les bords des objets d'intérêts.
- Sa capacité à traiter les différentes formes complexes, par exemple la variabilité d'apparences et l'articulation du corps humain.
- Sa capacité à gérer les topologies complexes telle que la fusion et la division.
- Sa capacité à modéliser les objets occultés.

III.3 YOLO (You only look once):

Proposé par J. Redmon et al [85], est l'un des algorithmes de détection et de classification d'objets en temps réel les plus puissants. Le YOLO possède une architecture très simple, ce qui le rend extrêmement rapide. On l'appelle ainsi car contrairement aux algorithmes de détection et de classification d'objets cités précédemment, qui examinent successivement plusieurs régions de l'image pour trouver les objets qui sont présents, puis faire plusieurs prédictions sur chacune des régions, le YOLO a changé cela en raisonnant au niveau de l'image globale. Plutôt que d'utiliser la méthode en deux étapes pour la classification et la localisation de l'objet, YOLO applique un CNN unique pour la classification et la localisation de l'objet simultanément. Il analyse l'image une seule fois en la découpant en S× S cellules. Si le centre d'un objet tombe

dans une cellule, cette cellule est "responsable" de la détection de l'existence de cet objet. Chaque cellule prédit :

1) L'emplacement des fenêtres englobantes :

-Les coordonnées de la fenêtre sont définies par un tuple de 4 valeurs, (centre (x,y), largeur w, hauteur h). De plus, x, y, w et h sont normalisés par la largeur et la hauteur de l'image, et donc leurs valeurs varient entre (0, 1).

2) Un score de confiance :

-Il indique la probabilité que la cellule contienne un objet : P_r(contenant un objet) x IoU(pred, vérité) ; où P_r= probabilité et IoU = interaction sur union.

3) Une probabilité de classe d'objet conditionnée par l'existence d'un objet dans la boîte de délimitation :

- Si la cellule contient un objet, elle prédit une probabilité que cet objet appartienne à chaque classe $C_i, i = 1, \ldots, K$: P_r (l'objet appartient à la classe C_i | contenant un objet). À ce stade, le modèle ne prédit qu'un seul ensemble de probabilités de classe par cellule, quel que soit le nombre de fenêtres englobantes.

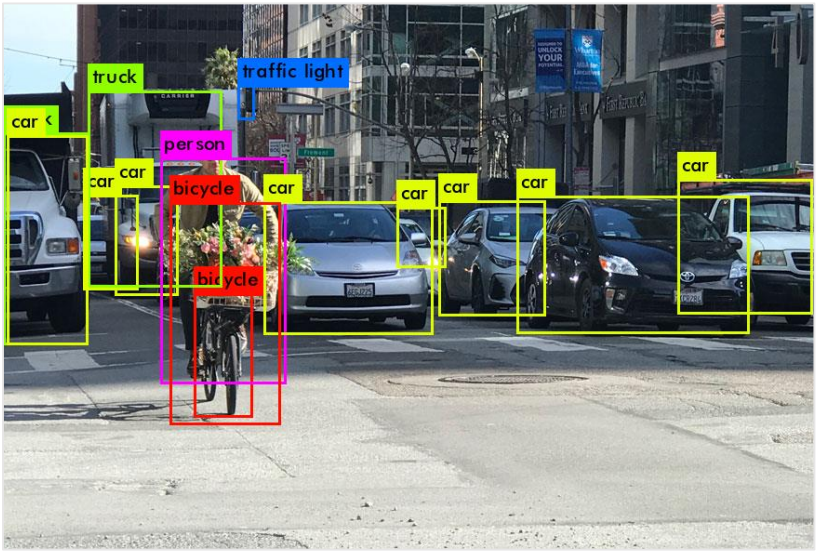


Figure III. 4: Exemple de prédiction avec YOLO.

III.3.1 YOLO v3 (You only look once version 3):

Le YOLO v3 [83], est une amélioration par rapport aux précédentes versions YOLO [85], YOLO v2, YOLO9000 [86]. La caractéristique déterminante de ce réseau est sa capacité à combiner trois phases en une seule étape. :

1) La détection d'objet («y a-t-il un objet?»).

2) La classification («quel type d'objet est-ce?»).

3) La localisation («où dans l'image se trouve l'objet?»).

Cela le rend plus efficace et plus robuste en termes de calcul que les autres réseaux qui n'effectuent qu'une ou deux de ces tâches simultanément.

Par rapport aux versions précédentes, YOLO v3 présente de multiples avantages, une détection multi-échelles, un extracteur de caractéristiques plus puissant et certains changements dans la fonction de perte. Ce réseau peut désormais détecter beaucoup plus de cibles, des plus grandes aux plus petites, ainsi que détecter avec précision les objets collés les uns aux autres.

III.3.2 Architecture YOLOv3 :

L'ensemble du réseau peut être divisé en deux composants principaux :

1) Extracteur de caractéristiques.
2) Détecteur de cibles.

Le YOLOv3 utilise le principe de multi échelles, l'image fournie à son entrée passe d'abord par l'extracteur de caractéristiques afin d'obtenir des images incorporées à trois échelles différentes. Ensuite, ces caractéristiques sont introduites dans trois branches du détecteur pour obtenir des boîtes englobantes et des informations de classe.

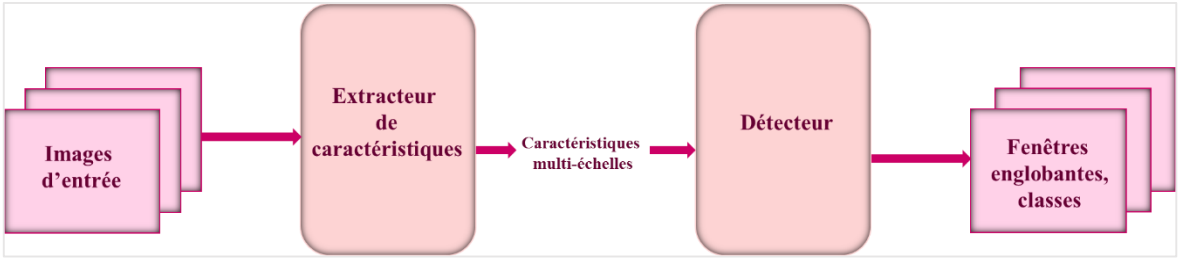


Figure III. 5: Architecture YOLO v3 simplifiée.

1) L'extracteur de caractéristiques :

L'extraction des caractéristiques ce fait à l'aide du module Darknet-53 (voir figure). Il utilise 53 couches convolutives successives 3x3 et 1x1 et s'inspire de l'idée du ResNet [79]. Le Darknet-53 contient 5 blocs résiduels, où le concept du saut de couches en utilisant les connexions résiduelles va permettre d'éviter le problème de disparition (vanishing) de gradient. Les paramètres du Darknet-53 sont fixés et ne sont pas ajustés lors de l'apprentissage.

	Type	filtres	dimension	La sortie
	Convolution	32	3x3	256x256
	Convolution	64	3x3/2	128x128
1x	Convolution	32	1x1	
	Convolution	64	3x3	
	Residuelle			128x128
	Convolution	128	3x3/2	64x64
2x	Convolution	64	1x1	
	Convolution	128	3x3	
	Residuelle			64x64
	Convolution	256	3x3/2	32x32
8x	Convolution	128	1x1	
	Convolution	256	3x3	
	Residuelle			32x32
	Convolution	512	3x3/2	16x16
8x	Convolution	256	1x1	
	Convolution	512	3x3	
	Residuelle			16x16
	Convolution	1024	3x3/2	8x8
4x	Convolution	512	1x1	
	Convolution	1024	3x3	
	Residuelle			8x8
	Avgpool	global		
	Connected	1000		
	Softmax			

Figure III. 6: Darknet-53 [83].

YOLO V3 détecte des cibles de taille différente à 3 échelles différentes. Il extrait les caractéristiques de ces échelles en utilisant le principe des réseaux pyramidaux (FPN : Feature Pyramid Network) [87]. Le concept consiste à créer une pyramide d'une même image à différentes échelles (voir Figure III.7) pour détecter des objets de petites ou grandes tailles.

Le FPN se compose d'une voie ascendante et descendante. La voie ascendante est le réseau convolutif habituel pour l'extraction de caractéristiques. Au fur et à mesure que nous montons, la résolution spatiale diminue en effectuant un sous-échantillonnage par un facteur de 2. Avec plus de caractéristiques de haut niveau détectées, la valeur sémantique de chaque couche augmente. À l'échelle 3, la carte des caractéristiques est utilisée pour détecter les petites cibles. À l'échelle 2, la carte des caractéristiques est utilisée pour détecter des cibles de taille moyenne. À l'échelle 1, la carte des caractéristiques est utilisée pour détecter les grandes cibles.

Alors que les couches obtenues contiennent des caractéristiques de haut niveau, l'emplacement des objets n'est pas précis après tout sous-échantillonnage. Les cartes caractéristiques de faible

résolution fournissent des informations sémantiques approfondies, alors que les grandes cartes d'entités fournissent des informations plus fines sur les cibles. Une fusion entre ces deux types de cartes est nécessaire pour aider le détecteur à prédire la localisation des objets

Pour cela la voie descendante est utilisée pour redimensionner les cartes. Elle consiste à sur-échantillonner spatialement plus grossièrement les entités des niveaux de pyramide supérieurs en utilisant un facteur de 2.

Les différentes cartes d'entités à différentes échelles auront la même taille. A partir des connexions résiduelles, le YOLO v3 fusionne les cartes caractéristiques de la même taille spatiale à partir du chemin ascendant et du chemin descendant. Plus précisément, les cartes caractéristiques de la voie ascendante subissent des convolutions 1x1 et sont ensuite fusionnées par ceux du chemin descendant par un ajout d'élément par élément. Enfin une convolution 3x3 est effectuée sur chaque carte fusionnée pour générer les cartes de caractéristiques finales {P1, P2, P3}.

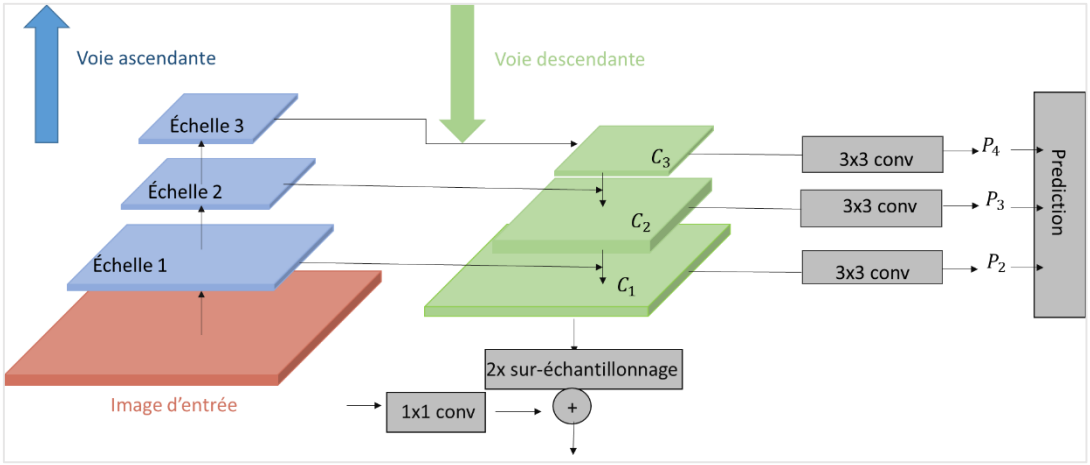


Figure III. 7: Principe du FPN.

2) Détecteur de cibles :

Le détecteur est un second réseau de 53 couches très proche de l'extracteur de caractéristiques, à l'exception des dernières couches. La dernière couche du réseau est de dimension **SxS x 3 x [(nb de classes + 1) + 4]**. Lors de l'apprentissage du modèle, les poids de ce second réseau seront ajustés aux données d'apprentissage. Les cartes caractéristiques à différentes échelles sont traitées par le détecteur comme étant une grille divisée en plusieurs cellules (13×13, 26×26 et 52×52 à trois échelles). Pour chaque cellule il doit prédire (voir Figure III.8) :

- 3 fenêtres englobantes où chacune est définie par 4 attributs (b_x,b_y,b_h,b_h). Chaque cellule doit prédire un objet à travers l'une de ses fenêtres si le centre de l'objet se situe dans le champ de réception de cette cellule.

- Le score de confiance pour indiquer si cette boîte contient un objet «c».

- Les probabilités de classe pour indiquer à quelle classe appartient cette boîte p_c.

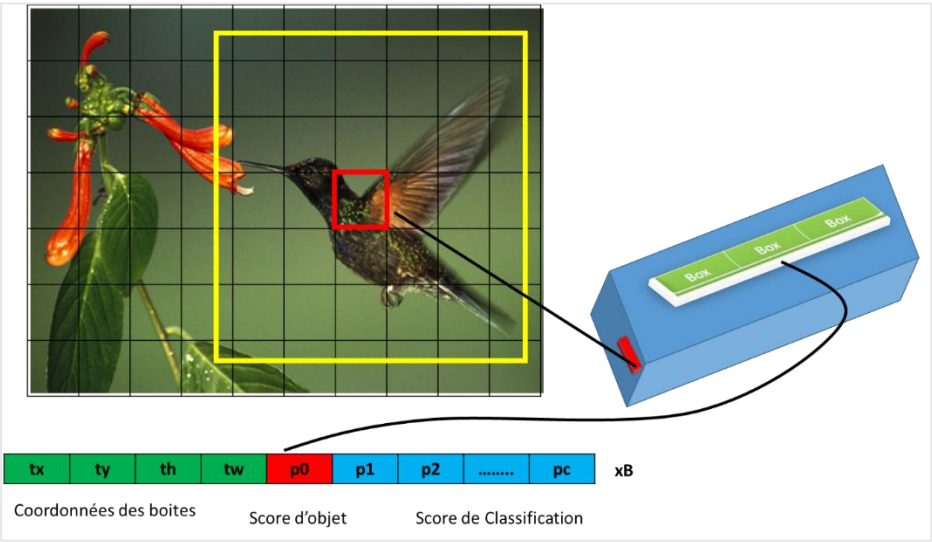


Figure III. 8: La prédiction sur la carte des caractéristiques.

YOLO V3 a introduit l'idée de boîtes d'ancrage utilisées dans Faster R-CNN [88]. Les boîtes d'ancrage sont un ensemble de boîtes englobantes initiales avec une largeur et une hauteur fixes. Ces boites sont définies pour capturer l'échelle et le rapport hauteur / largeur des classes d'objets spécifiques que l'on souhaite détecter et sont généralement choisies en fonction de la taille des objets dans l'ensemble de données d'entraînement. L'utilisation de boîtes d'ancrage permettra au réseau de détecter plusieurs objets, d'échelles différentes et ceux qui se chevauchent. Le choix des boîtes d'ancrage initiales affectera ainsi la précision de détection et sa vitesse. Au lieu de choisir les boîtes d'ancrage à la main, YOLO V3 exécute le clustering K-means sur l'ensemble de données pour trouver automatiquement à priori les bons.

Le principe du K-means est très simple :

- Il suffit de déterminer les centroides de toutes les boîtes de vérité-terrain étiquetées de l'image d'entrée ainsi que leurs coordonnées, x_j , y_j , w_j , h_j, $j \in \{1,2, \ldots . N\}$.

$-x_j, y_j$ sont le centre de la boite.

$-w_j, h_j$ la largeur et la hauteur.

−N est le nombre de toutes les étiquettes des boîtes.

- Donner ensuite k=9 points centraux des clusters $w_i, h_i, \mathrm{i} \in \{1,2..,\mathrm{k}\}$, avec $\mathrm{W_i}, \mathrm{H_i}$ sont la hauteur et largeur des boites d'ancrages.
- Ensuite regrouper les 9 clusters en 3 groupes différents selon leur échelle. Chaque cluster va être affecté à une carte des caractéristiques spécifiques lors de la détection d'objets
- Un calcul de distance $\mathrm{d} = 1 - \mathrm{IoU}$ (boîte de vérité terrain étiquetée, centre de chaque cluster) est effectué. Lors de ce calcul, le point central de chaque boite de vérité terrain étiquetée coïncide avec le centre du cluster, de sorte que la valeur IoU peut être calculée $\mathrm{d} = 1 - \mathrm{IoU}[(\mathrm{x_j}, \mathrm{y_j}, w_j, h_j), (\mathrm{x_j}, \mathrm{y_j}, w_i, h_i,)], \ \mathrm{j} \in \{1,2, \ldots .. \mathrm{N}\}, \mathrm{i} \in \{1,2, \ldots, \mathrm{k}\}$.
- Attribuer la boîte de vérité terrain au centre de cluster le plus proche de distance.

IoU (Intersection over Union) : est une technique utilisée pour mesurer la qualité de la détection. Elle permet de faire la comparaison entre les fenêtres englobantes prédites et la fenêtre englobante de vérité de terrain et voir à quel point elles se recouvrent. Il s'agit du rapport entre la surface commune aux deux fenêtres et la surface totale occupée par les deux fenêtres (voir Figure III.9). L'IoU est compris entre 0 et 1, un IoU de 1 correspond à deux fenêtres superposées et un IoU de 0 à deux fenêtres sans aucune correspondance.

$$\mathrm{IoU}=\frac{\mathrm{Intersection}}{\mathrm{Intersection-Union}}$$

$$\mathrm{Intersection}= \mathrm{Min}(\mathrm{w_{1,}w_2})\ \mathrm{Min}(\mathrm{h_1}, \mathrm{h_{2)}}$$

$$\mathrm{Union}= \mathrm{w_1h_1} + \mathrm{w_2h_2}$$

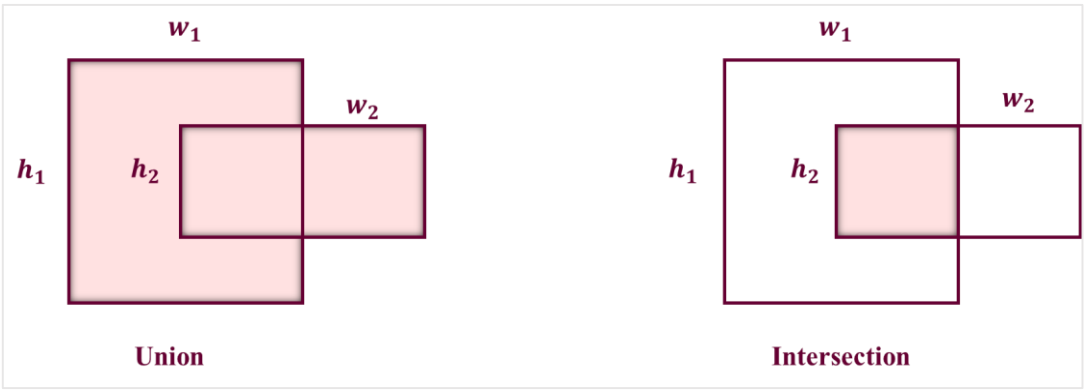


Figure III. 9: Principe d'IoU.

Pour chacune des fenêtres englobantes à différentes échelles dont la boite de vérité terrain a été attribuée le YOLO v3 doit prédire leurs dimensions à ce qu'elle soit presque similaire à celle des objets à détecter.

A partir de la Figure III.10, la boite en pointillés représente une boîte d'ancrage dont la largeur et la hauteur sont données respectivement par $P_w$etP_h. Le réseau prédit les paramètres $t_w$et t_h qui, lorsqu'ils sont exponentiels, mettent à l'échelle les dimensions de la boîte d'ancrage pour s'adapter aux dimensions de l'objet à détecter. Le réseau prédit également le centre b_x et b_y de la fenêtre englobante en utilisant les coordonnées du centre initial t_x t_y et en ajoutant le décalage de la cellule par rapport à l'image d'entrée de (C_x,C_y). La fonction sigmoïde est utilisée pour forcer la valeur de sortie à être comprise entre 0 et 1 vu que les dimensions des cellules sont déjà normalisées Cela donne le cadre final de délimitation prévu, affiché en bleu. Les prédictions correspondent à :

$$b_x=\sigma(t_x)+C_x \quad (3.8)$$

$$b_y=\sigma(t_y)+C_y \quad (3.9)$$

$$b_w=P_w e^{t_w} \quad (3.10)$$

$$b_h=P_h e^{t_h} \quad (3.11)$$

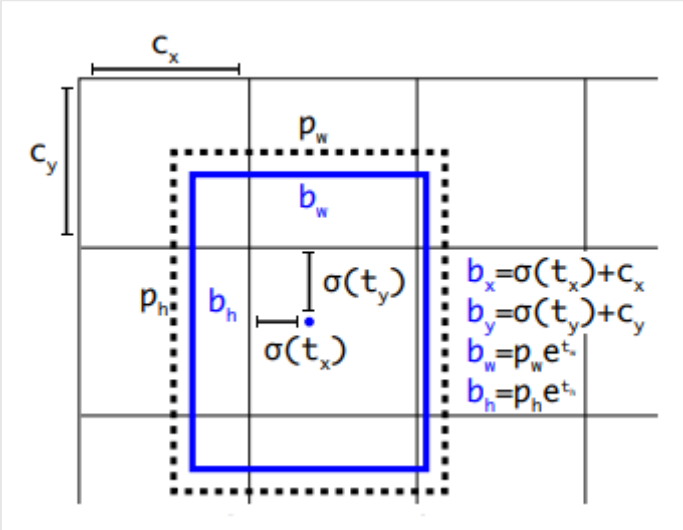


Figure III.10 : Fenêtre englobante avec les dimensions et prédiction d'emplacement [83].

III.3.3 Fonction de coût :

L'apprentissage du YOLO v3 se fait par l'ajustement des divers poids du détecteur, en minimisant la fonction de coût suivante :

$$\mathcal{L} = \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [\sqrt{(x_i - \hat{x}_i)^2} + \sqrt{(y_i - \hat{y}_i)^2}] + \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [\sqrt{(w_i - \hat{w}_i)^2} + \sqrt{(h_i - \hat{h}_i)^2}] + \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{obj} [\sqrt{(C_i - \hat{C}_i)^2} + \lambda_{noobj} \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \sqrt{(C_i - \hat{C}_i)^2} + \sum_{i=0}^{s^2} 1_i^{obj} \sum_{c \in class} \sqrt{(p_i(c) - \hat{p}_i(c))^2}. \quad (3.12)$$

Cette fonction est une composition de SSE (somme des erreurs carré) et peut se séparer en une somme de trois blocs. La perte de localisation pour la prédiction du décalage de la boîte englobante, la perte sur le score de confiance et la fonction de perte de classification pour les probabilités de classe conditionnelle. Pour une prédiction parfaite, le coût sera de 0.

Le premier bloc calcule l'erreur de position de la fenêtre par rapport à la vérité de terrain. La constante λcoord représente le poids que l'on donne à cette partie de la fonction.

La deuxième partie de la fonction calcule l'erreur sur le score de confiance. L'objectif est de trouver 1 pour la cellule qui contient un objet et 0 pour la cellule qui ne contient rien. La plupart des boîtes sont vides. Cela provoque un problème de déséquilibre de classe. Pour y remédier, la perte est pondérée d'un facteur λnoobj.

Enfin, la dernière partie de la fonction est la perte de classification. Si un objet est détecté, la perte de classification à chaque cellule est l'erreur quadratique des probabilités conditionnelles pour chaque classe.

III.3.4 Le choix du YOLO v3 se justifie par :

- Extraction automatique des caractéristiques discriminantes sans intervention humaines. Cela permet un gain en temps remarquable.
- Identification des objets en analysant l'image une seule fois puis fournir le résultat de prédiction.
- Rapidité qui convient aux applications temps réel.
- Traitement des images 2D, qui met en jeu une capacité à traiter de grandes quantités d'informations.

III.4 Description de la data set :

L'entrainement du modèle requiert un volume de données très important en vue de la grande diversité des scénarios routiers, et de la large variabilité d'apparence et formes des différents objets urbains. Nous avons besoin d'une data set conséquente et adaptée pour entraîner le modèle.

Plusieurs data set ont été présentées dans la littérature, nous avons opté pour KITTI [89], téléchargeable depuis : http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark. Elle comprend 7481 images annotées de scènes de conduite, avec 80 classes d'objets. La taille d'une seule image est de 800 à 900 ko et la taille totale des 7481 images est de 6,2/12 Go. Parmi ces classes, seuls les objets fréquents (voiture, camion, piéton, cycliste, bicyclettes et personnes assises) sont étiquetés indépendamment, toutes les autres classes sont étiquetées comme "Others" ou "Dont Care". Les 7481 images ont été divisées en 70 % (5237 images) pour l'entrainement et 30 % (2244 images) pour les tests. La figure ci-dessous montre les échantillons des ensembles de données KITTI.

Figure III. 11 : Exemple des images des différentes classes de la data set KITTI [89].

Les fichiers d'annotation contiennent les paramètres suivants : type de classe, troncature, occlusion, orientation et boîte englobante. Leur description est donnée dans le tableau :

Valeur	Type du Label	Description
1	Type	Les types d'objets : [Car, Van, Truck, Pedestrian, Person sitting , Cyclist, Tram, Others or Dont care]
1	Truncation	Flotter de 0 (non tronqué) à 1 (tronqué), ou tronqué fait référence à l'objet quittant les limites de l'image.
1	Occlusion	Chiffre (0,1,2,3) indiquant l'état d'occlusion: 0 = entièrement visible 1 = partiellement occlus 2 = largement occlus 3 = inconnu.
1	Orientation	Angle d'observation de l'objet allant de [-180°, 180°]
4	Fenêtre englobante	Boîte englobante 2D de l'objet dans l'image (index basé sur 0) : contient les coordonnées de pixels gauche, haut, droite et bas.

Table III. 1: Description des paramètres des labels KITTI [89].

III.5 Présentation des outils :

a) Le software

Python 3.5 : est un langage de programmation de haut niveau interprété (pas d'étape de compilation) et orienté objet avec une sémantique dynamique. Très sollicité par une large communauté de développeurs et de programmeurs open-source, python offre des modules pour interagir avec une variété de bases de données et de datasets, ce qui en fait un excellent choix pour l'analyse de données. C'est un langage simple, facile à apprendre et permet une bonne réduction du cout de la maintenance des codes.

TensorFlow 1.0 : est un framework de programmation dédié au calcul numérique. Il a été développé à l'origine par Google Brain Team, au sein de l'organisation de recherche 'Machine Intelligence' et mis sous licence open source en 2015 [90]. L'un des framework les plus utilisés pour le Deep Learning, ou les opérations actuelles des réseaux de neurones s'effectuent principalement via une table multidimensionnelle de données, appelée Tenseurs (Tensor). L'un des avantages du Tensorflow : plateforme-croisée (cross-platform,). En effet l'exécution peut être lancée sur plusieurs types de machines et processeurs, notamment les CPU (Central Processing Unit), les GPU (Graphic Processing Unit) et les TPU (Tensor Processing Unit). A

noter que le développement des applications repose sur le langage de programmation Python, tandis que l'exécution de ces applications s'effectue en C++ haute-performance.

Pytorch 0.4 : est une bibliothèque d'apprentissage machine basée sur Python. Il a été développé par le groupe de recherche sur l'IA de Facebook en 2016. PyTorch a été conçu pour offrir une flexibilité et une vitesse de traitement maximales. Il est connu pour fournir deux des fonctionnalités de plus haut niveau :

- manipuler des tenseurs et effectuer des calculs efficaces sur CPU ou GPU
- la construction de réseaux neuronaux profonds.

Les principales raisons qui ont également rendues PyTorch l'une des plateformes de recherche en apprentissage en profondeur préférées sont [91] :

- Un graphe de calcul dynamique qui permet de faire évoluer le réseau de neurones au fur et à mesure de l'apprentissage, ajouter de nouveaux nœuds, modifier les connexions entre eux, y compris d'une couche à l'autre à l'aide du package autograd.
- Meilleures performances : PyTorch est plus rapide que d'autres framework en termes d'images traitées par seconde.

OpenCV 3.2 : développé par Intel, est une bibliothèque open source de vision par ordinateur et d'apprentissage automatique, initialement développé en C ++, puis suivi par Java et Python. OpenCV fonctionne sur diverses plates-formes telles que Windows, macOS, Android, iOS et Linux [92]. OpenCV est un outil parfait pour la vision par ordinateur utilisé pour accélérer la reconnaissance automatique en temps réel des images, des objets et des applications de traitement vidéo. Il permet de :

- Lire et écrire des images, capturer et enregistrer des vidéos.
- Traiter des images comme le filtrage et la transformation.
- Détecter des caractéristiques / objets sur vidéo ou image.
- Analyse des vidéos.

b) Le hardware

L'apprentissage des réseaux de neurones profonds requirent une puissance de calculs accrue. La disponibilité des ressources (Particulièrement GPU), influent fortement sur la durée de l'apprentissage.

Les différentes expérimentations de notre projet ont été effectuées sur une machine qui offre des performances acceptables dont les caractéristiques sont :

CPU	Intel i7
GPU	NVIDIA Quadro 2000
Disque dur	256 Go en SSD
OS	Ubuntu 18.04

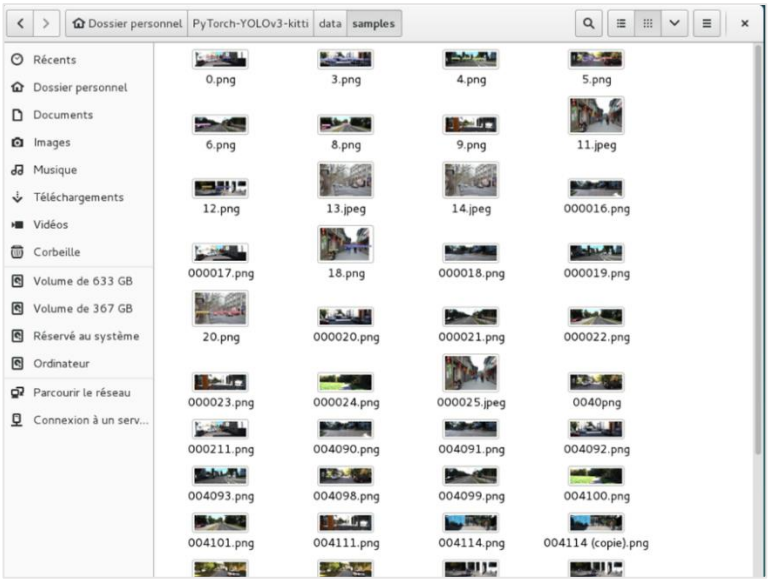
III.6 Démarches adoptées :

Le réseau YOLO v3 utilisé est déjà pré-entrainé sur la base de données COCO. Nos expérimentations seront effectuées uniquement sur la prédiction de 3 classes d'objets : piétons, véhicules et bicyclettes. L'entrainement sera effectué en fournissant les images en lot au réseau.

Avant de commencer l'apprentissage nous avons besoin de suivre quelques étapes :

1) Crée l'environnement d'exécution, ainsi que télécharger toutes les bibliothèques nécessaires.
2) Charger le fichier du code source.

3) Charger les données.



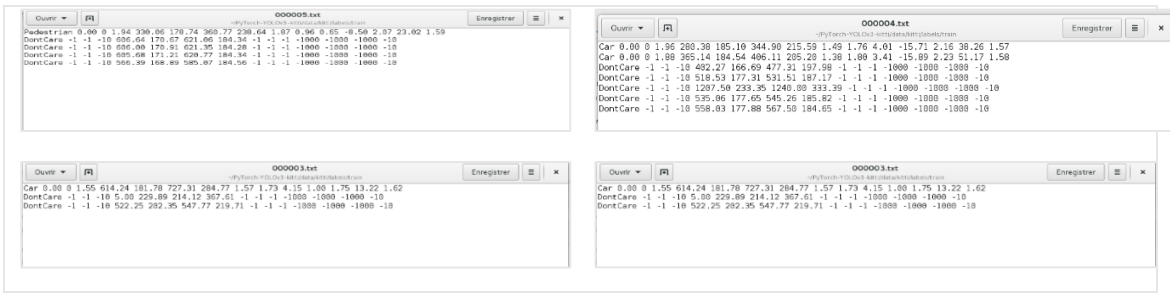
4) Séparer les données d'entrainement des données tests.

Préparation des données :

Les données d'entrainement sont constituées d'images et d'étiquettes. Notre réseau est compatible avec la plupart des formats d'images, comme .png et .jpeg. Cependant, les fichiers d'étiquettes sont dans des formats différents que nous avons convertis en un format unifié que le réseau peut comprendre. Dans notre système, les fichiers d'étiquettes portent le même nom que les images, mais avec l'extension .txt. Le fichier d'étiquettes comprend trois autres valeurs, "Orientation", "Occlusion" et " Troncature", comme indiqué dans le tableau ci-dessous :

Classe	X	Y	Largeur	Hauteur	Orientation	Occlusion	Troncature

Voici quelques captures des fichiers d'étiquettes :



Les coordonnées (Centre-X, Centre-Y, Largeur, Hauteur) d'une boîte englobante sont en valeurs de pixels et doivent être normalisées. Nous devons les convertir en ratio en divisant la longueur et la largeur de l'image. Comme le montre la figure, nous chargeons l'image avec la largeur et la hauteur. Ensuite, le programme lit les fichiers d'étiquettes et convertit en même temps les coordonnées des valeurs de pixels en valeurs de rapport.

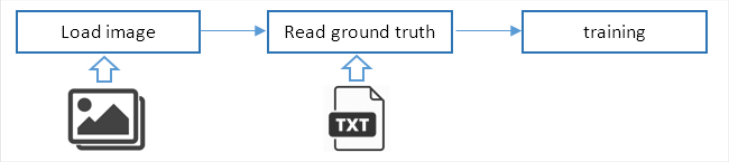
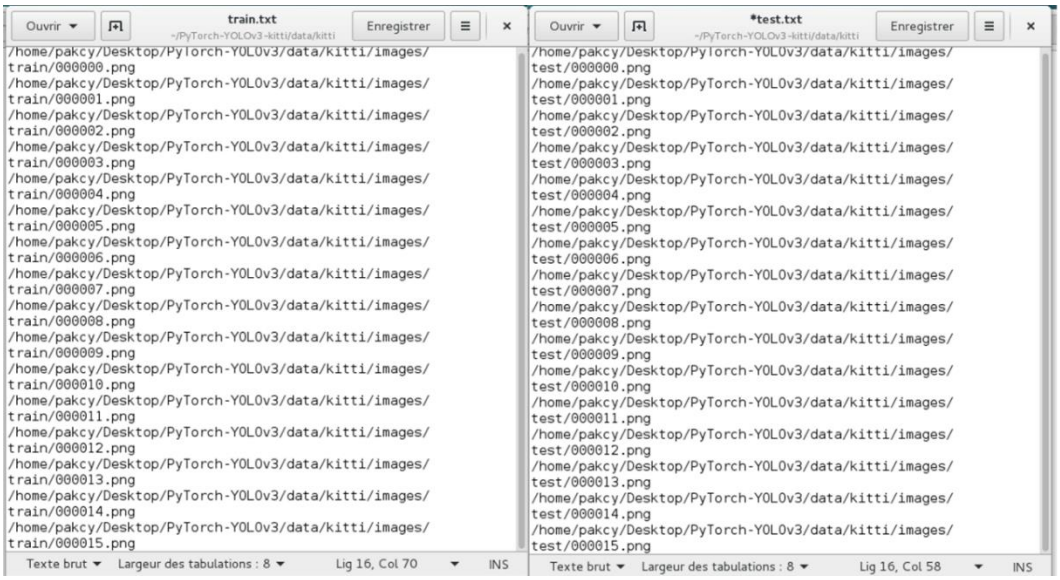


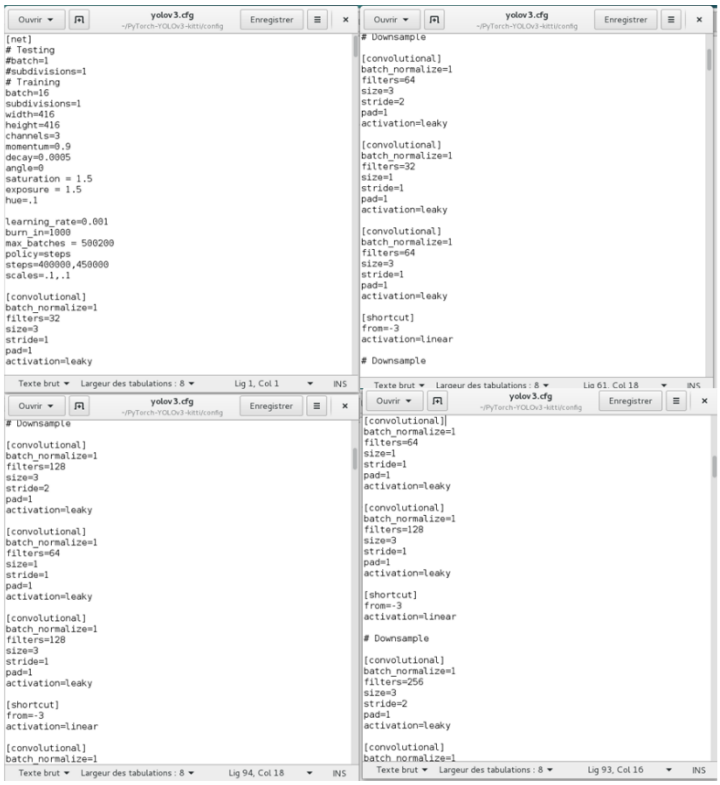
Figure III.12 : Processus de conversion d'étiquettes au moment de l'exécution.

5) Une fois que les fichiers d'étiquettes et les fichiers d'images sont préparés, nous devons créer deux listes de chemins d'accès aux fichiers, qui contenant respectivement les fichiers d'entrainement et les fichiers de tests.



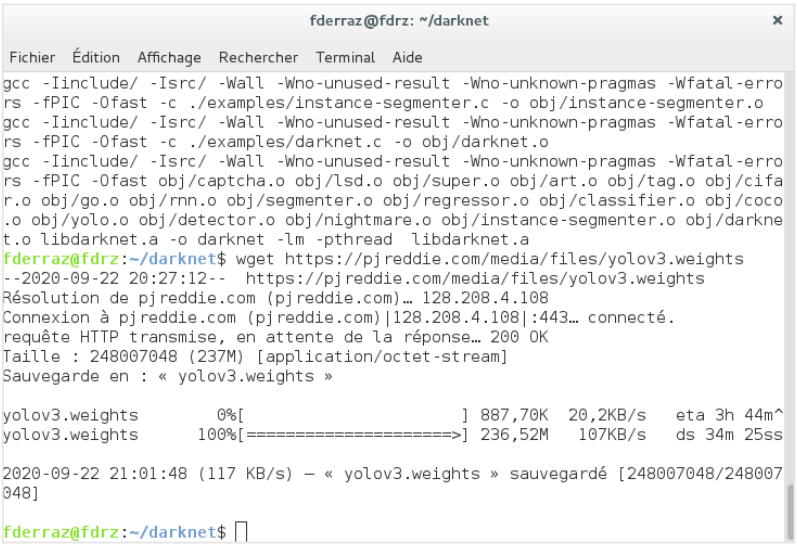
6) Fichier .txt avec les noms de toutes les classes.

7) Un fichier de configuration avec toutes les couches du réseau YOLO v3.



```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=16
subdivisions=1
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 500200
policy=steps
steps=400000,450000
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

# Downsample

[convolutional]
batch_normalize=1
filters=128
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=256
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
```

8) Poids convolutifs pré-entraînés.



```
fderraz@fdrz: ~/darknet
Fichier  Édition  Affichage  Rechercher  Terminal  Aide
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-erro
rs -fPIC -Ofast -c ./examples/instance-segmenter.c -o obj/instance-segmenter.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-erro
rs -fPIC -Ofast -c ./examples/darknet.c -o obj/darknet.o
gcc -Iinclude/ -Isrc/ -Wall -Wno-unused-result -Wno-unknown-pragmas -Wfatal-erro
rs -fPIC -Ofast obj/captcha.o obj/lsd.o obj/super.o obj/art.o obj/tag.o obj/cifa
r.o obj/go.o obj/rnn.o obj/segmenter.o obj/regressor.o obj/classifier.o obj/coco
.o obj/yolo.o obj/detector.o obj/nightmare.o obj/instance-segmenter.o obj/darkne
t.o libdarknet.a -o darknet -lm -pthread  libdarknet.a
fderraz@fdrz:~/darknet$ wget https://pjreddie.com/media/files/yolov3.weights
--2020-09-22 20:27:12--  https://pjreddie.com/media/files/yolov3.weights
Résolution de pjreddie.com (pjreddie.com)… 128.208.4.108
Connexion à pjreddie.com (pjreddie.com)|128.208.4.108|:443… connecté.
requête HTTP transmise, en attente de la réponse… 200 OK
Taille : 248007048 (237M) [application/octet-stream]
Sauvegarde en : « yolov3.weights »

yolov3.weights        0%[                    ] 887,70K  20,2KB/s   eta 3h 44m^
yolov3.weights      100%[===================>] 236,52M   107KB/s   ds 34m 25ss

2020-09-22 21:01:48 (117 KB/s) — « yolov3.weights » sauvegardé [248007048/248007
048]

fderraz@fdrz:~/darknet$
```

III.7 Application du YOLO v3 sur les données KITTI :

III.7.1 Expérimentation 1 :

Nous avons commencé par effectuer un entrainement du YOLO v3 sur l'ensemble de données KITTI. Cette étape n'était pas vraiment indispensable, vu qu'il est déjà pré-entrainé. Notre but était de choisir avec précaution des scènes où les objets étaient soit occultées, soit sous différentes conditions d'éclairages, pour atteindre un niveau de généralisation acceptable et que le réseau arrive à répondre à nos exigences. Cette première expérimentation a été réalisée afin de visualiser les performances du réseau seul.

La figure ci-dessous représente les démarches à suivre pour cette première expérimentation.

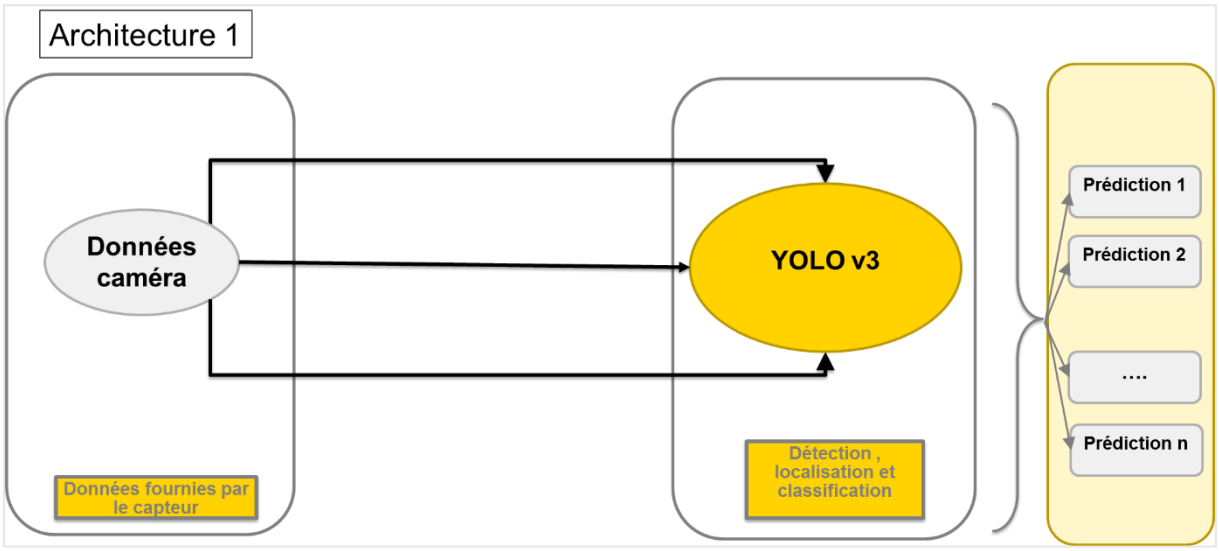


Figure III.13: Première architecture proposée.

Dans la section qui suit, nous présentons la configuration de l'entrainement, les résultats, les discussions et l'analyse.

a) Configuration d'entrainement :

Dans cette expérience, nous avons utilisé des lots, de 64 images chacun. Nous avons entrainé le réseau pour un total de 1000 lots d'une durée de 20 heures. L'entrainement est réalisé sur 70% des données KITTI (5237 images), et les tests sur 30% des données (2244 images). A noter que nous avons changé le nombre de classes et nous avons évalué les performances du YOLOv3 uniquement sur les 3 classes candidates : piétons, bicyclettes et voitures.

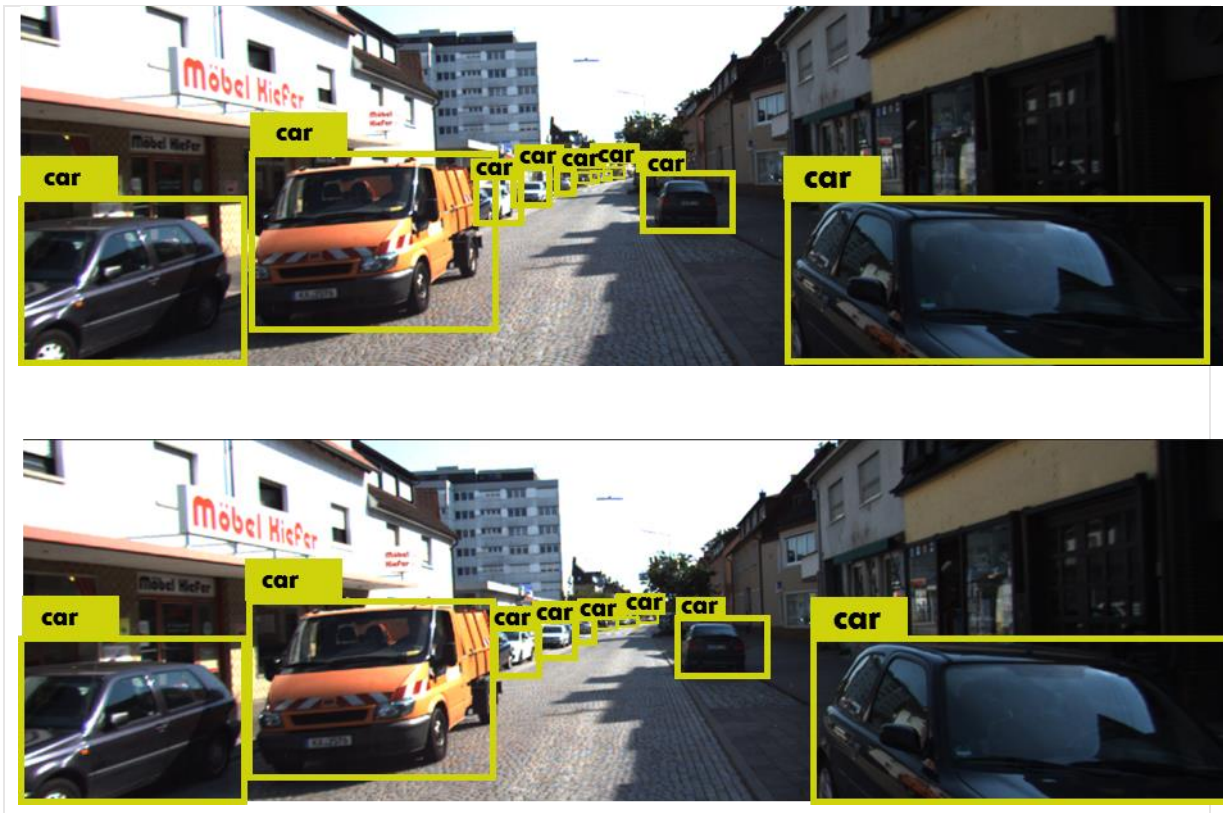
b) Discussion des résultats :

Notre modèle a pu atteindre une précision globale de 68.01% avec une vitesse de détection de 30 secondes par image (le temps de calcul CPU). Dans cette section, nous présentons le résultat sur les échantillons de tests des images KITTI. Nous avons choisi de présenter les résultats sur quelques séquences d'image et nous précisons uniquement les situations appropriées. Nous avons fait exprès de choisir des scènes complexes, comme le cas d'occlusion entre véhicules, l'effet d'éblouissement de la lumière, les effets de l'ombre et l'effet d'échelle. Nous avons aussi pris en considération des scénarios divers sur la route, aux carrefours de la route. Les captures que nous présentons ne sont pas les seules situations rencontrées.

Figure III. 14: Exemples des captures des séquences de la dataset de KITTI.

c) Résultats sur les images tests :

Les captures suivantes montrent un groupe de résultats de détection sur 3 classes d'objets. Sur les premiers jeux de données que nous proposons à notre système, nous remarquons que tous les obstacles y compris ceux qui sont loin sont détectés avec précision. Le réseau arrive à détecter les objets occultés et ceux qui apparaissent comme collés les uns aux autres. Nous remarquons notamment la défaillance du système où il détecte le camion comme étant une voiture, ainsi que le siège intérieur comme étant une personne sur les captures 3 et 4.



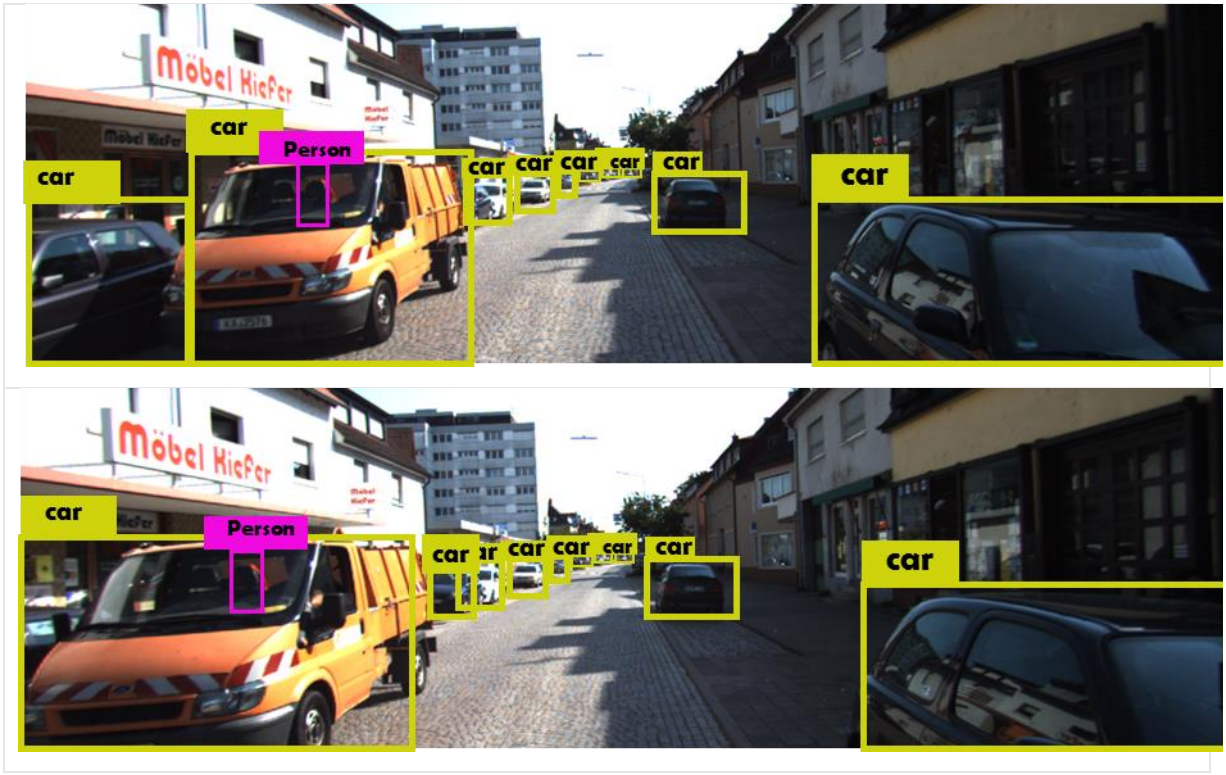


Figure III.15 : Résultats sur les premières séquences tests de KITTI.

Dans un deuxième exemple dont nous représentons les résultats sur les figures ci-dessous, nous avons choisi les séquences avec des piétons, une bicyclette et une voiture en mouvement. Nous constatons sur la capture 1 que le système n'arrive pas à détecter les objets lointains sur la zone ensoleillée jusqu'à ce qu'il se rapproche d'eux (capture 2). Mais en revanche, il nous propose la détection de la personne et de la bicyclette en zone ombrée. Pour la capture 2 il n'arrive à détecter que la personne sans la bicyclette. Pour les personnes assises sur un banc assez loin dans une zone ombrée, il n'arrive à détecter qu'une seule. Pour les captures 3 et 4 même les voitures éclairée par la lumière, le système les détecte avec succès, exception faite pour le cycliste qui se trouve à une distance assez lointaine.

Nous avons examiné les différentes séquences d'image et nous constatons que le système répond au mieux pour les cas difficiles que nous avons choisis (conditions d'éclairage variable).

Figure III.16: Résultats sur les secondes séquences tests de KITTI.

Dans une troisième situation, nous avons choisi une séquence d'un carrefour. Cet exemple est proposé afin de mettre à l'épreuve notre détecteur en présence des occlusions et l'effet de variation de lumière lors du passage des voitures. Il détecte les véhicules partiellement occultés

avec précision. Cependant on observe une défaillance du système qui détecte la centrale des feux tricolores comme étant une personne.

On peut dire que ses premiers tests s'avèrent partiellement concluant.

Figure III. 17: Résultats sur les troisièmes séquences tests de KITTI.

III.7.2 Expérimentation 2 :

Afin d'améliorer notre système face aux défaillances possibles, nous avons ajouté la couche de ségmentation sémantique basée sur les « Level-sets» comme montré sur la figure ci-dessous.

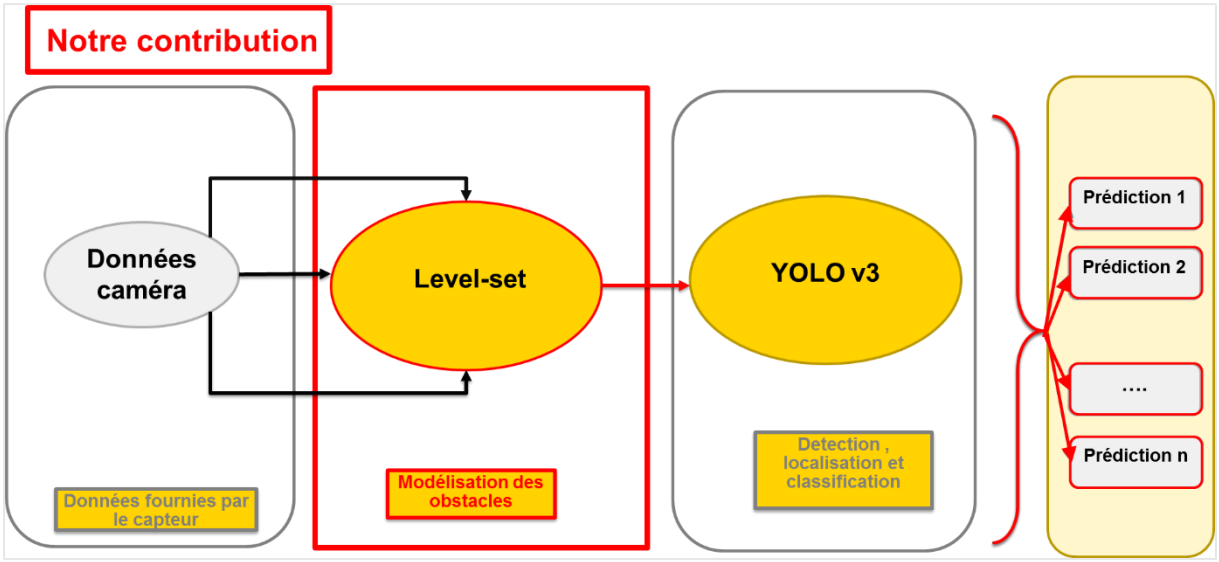


Figure III.18: Seconde architecture proposée.

a) Configuration d'entrainement

Dans cette expérience, nous avons suivi les mêmes démarches que la première. Sauf que cette fois ci le YOLO v3 est entrainé sur 70% des données KITTI segmentées par « Level-set ». Il a été ensuite testé sur les différentes scènes déjà choisis afin d'évaluer les performances du système. Les captures ci-dessous montrent les résultats de cette seconde expérimentation.

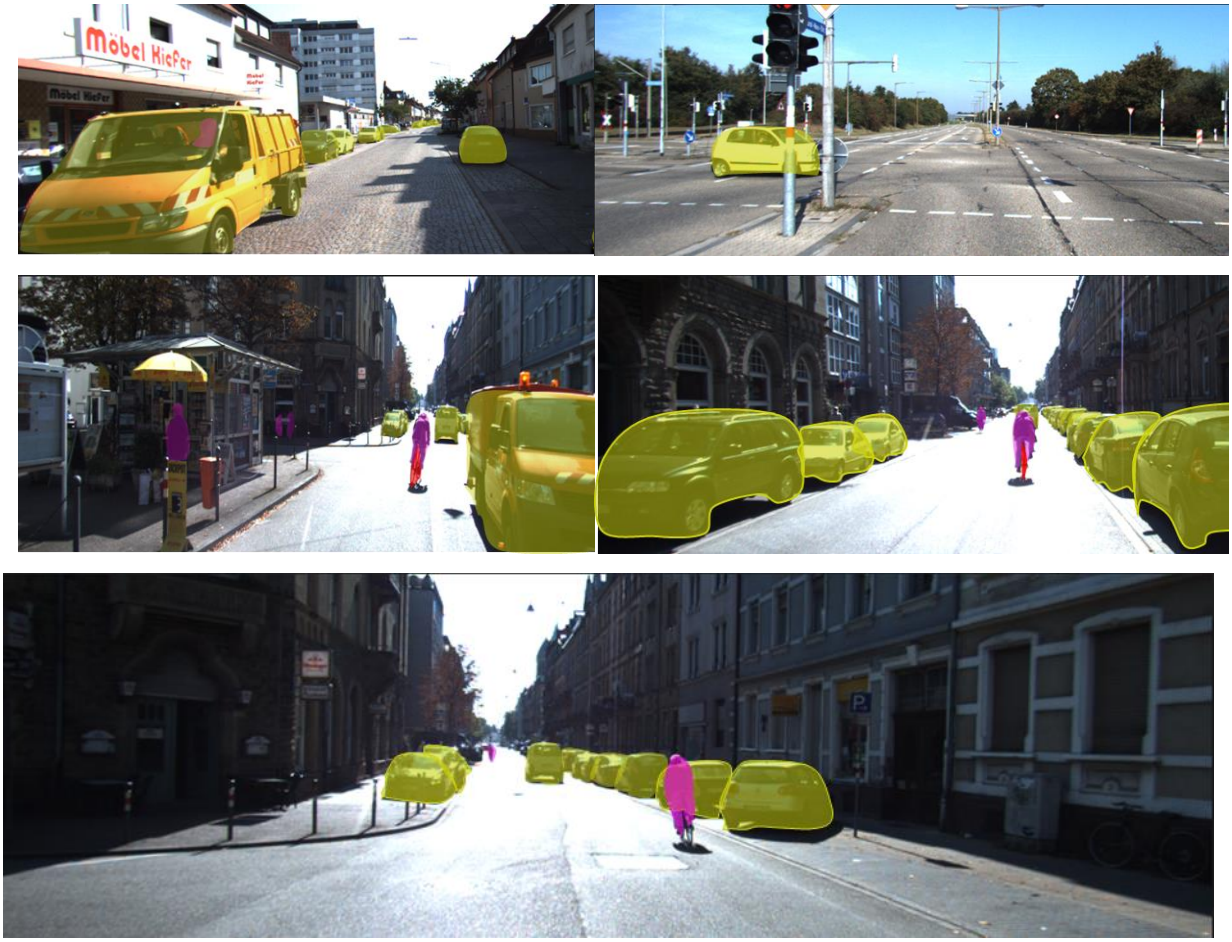


Figure III.19: Résultats du YOLO v3 combiné avec «Level-set» sur quelques séquences proposées.

b) Discussion des résultats :

Le modèle a pu atteindre une précision globale de 87% avec une vitesse de détection d'environ 35 secondes par image (le temps de calcul CPU). Malgré l'absence d'une unité puissante de calcul pour évaluer les performances sur de très large dataset, ce premier résultat montre la faisabilité de combiner un détecteur rapide de type YOLO avec une segmentation sémantique dans le but de traiter des tâches compliquées de reconnaissance.

III.8 Conclusion :

Ce dernier chapitre avait pour objectif de déterminer une solution qui répond au mieux aux différentes contraintes pour un système de détection et d'identification notamment les occlusions, la mise en échelle, les variations d'éclairages et le temps-réel. La solution proposée constitue un moyen pour l'amélioration des performances en termes de précision.

Les expérimentations montrent que l'approche proposée produit des résultats assez satisfaisants. Néanmoins, la contrainte du temps réel n'a pas été respectée en vue du manque d'unité de puissance de calcul.

Conclusion générale et perspectives

Porté par un intérêt accru envers l'amélioration de la sécurité routière, la diminution du nombre d'accidents, et l'avantage d'une assistance d'aide à la conduite de haut niveau, les constructeurs d'automobiles cherchent toujours à atteindre les meilleures performances afin de garantir une bonne qualité de circulation. La complexité, temps de traitement, coût, performances etc..., restent un défi qui occupe les chercheurs.

Les travaux du présent mémoire de master s'inscrivent dans le domaine de la vision par ordinateur, visant à concevoir et développer un module pour la détection et la classification des obstacles routiers pour les véhicules autonomes. Dans le premier chapitre de ce mémoire, les généralités sur les techniques de détection et de classification ont été présentées d'une manière générale afin de se familiariser avec le concept de base de ces notions. Nous avons présenté dans le second chapitre les réseaux de neurones classiques et les réseaux de neurones convolutifs étant donné qu'ils constituent une bonne solution pouvant améliorer les performances de notre système de détection et de classification qui s'appuie sur les technologies de perception de l'environnement qui entoure le véhicule. La complexité de la tache de perception provient de la grande diversité des scénarios routiers, la large variabilité d'apparence et formes des différents objets urbains, et de la qualité des capteurs utilisés.

La problématique de détection et d'identification d'obstacles routiers dans un environnement externe, dynamique, et complexe ne peut être résolue convenablement sans avoir recours aux techniques d'apprentissage profond qui permettent la reconnaissance de classes d'objets à partir de l'analyse des images. Pour réponde à notre problématique nous avons centré notre réflexion autour de deux axes principaux : La représentation des objets et leur classification.

Une partie de la réponse à la problématique a pu être apportée par un choix pertinent de la représentation. Nous nous sommes proposé d'utiliser une méthode de représentation basée sur les « Level-set », qui définit l'objet avec une très grande précision, permettant ainsi de résoudre le problème des formes complexes, des changements topologiques et des problèmes d'occultation des obstacles routiers.

Le deuxième axe que nous avons souligné est la classification. Notre choix s'est penché vers un réseau de neurones profond grâce à ses nombreux avantages en particulier son architecture simple, sa rapidité, et sa capacité à déceler les différents objets urbains lorsqu'ils sont collés les uns aux autres. Notre conception a été entamée par fournir un jeu de données pour le système que nous l'avons par la suite entrainé sur une grande partie de ces données.

Ce système a été validé à travers différentes expérimentations sur des images et des séquences

routières dans un milieu urbain. Les expérimentations montrent que le système proposé est rapide et produit des résultats satisfaisants face aux problèmes des formes complexes et d'occultations partielles.

Une des limitations constatées lors des expérimentations, est la faible précision pour la détection des objets faiblement contrastés lorsque le YOLO v3 est utilisé seul. Avec la combinaison des « Level-sets » les performances du détecteur sont améliorées d'environ 20%. Une présence d'un ensemble de véhicule dans le même flux, le YOLOv3 aura parfois du mal à mettre les objets en échelle. Il fusionne deux observations proches de même couleur dans un même objet. La combinaison de YOLOv3 avec les « Level-set » confond toujours un camion avec un véhicule. Pour les pétions sur les trottoirs, le YOLOv3 avec « Level-set » extrait toujours les personnes même lorsqu'ils sont dans une foule.

Après avoir déterminé et analysé les limites de la méthode proposée dans ce mémoire, nous aurons à présenter à l'avenir les perspectives envisageables de nos travaux de recherche. Les pistes à explorer et les applications étudiées seront nombreuses :

- Elargir la base d'entrainement pour atteindre un niveau acceptable de généralisation.
- Etendre le système de reconnaissance pour qu'il englobe d'autres objets routiers comme les panneaux routiers, les feux de circulation, les bords de la route,… .
- Fusionner les données issues de multiples capteurs.
- Tester notre approche dans des conditions réelles avec des caméras plus rapides.

Références

[1] Quelles sont les principales causes des accidents de la route ?. [Consulté le 31 aout 2020]. Disponible à l'adresse : https://www.jurifiable.com/conseil-juridique/droit-routier/causes-accidents-de-la-route

[2] K Jo, C Kim, M Sunwoo, Simultaneous Localization and Map Change Update for the High Definition Map-Based Autonomous Driving Car 2018.

[3] S. Osher et J.A. Sethian. Fronts propagating with curvature-dependent speed : Algorithms based on hamilton-jacobi formulation. Journal of Computational Physics, 79 :12-49, 1988.

[4] Messaouda, L., Abdelghani, R., MESSALI, Z., & LARBI, S. (2019). Medical Image Segmentation Based on Wavelet Transformation and Level set method. 2019 International Conference on Applied Automation and Industrial Diagnostics (ICAAID). doi:10.1109/icaaid.2019.8934970.

[5] Mohan, A. S., & Resmi, R. (2014). Video image processing for moving object detection and segmentation using background subtraction. 2014 First International Conference on Computational Systems and Communications (ICCSC). doi:10.1109/compsc.2014.7032664.

[6] Rafiq Sekkal. Techniques visuelles pour la détection et le suivi d'objets 2D. Traitement du signal et de l'image [eess.SP]. INSA de Rennes, 2014. Français. ffNNT : 2014ISAR0032ff. fftel-00981107v2f.

[7] Steven L. Horowitz, Theodosios Pavlidis. Picture segmentation by a tree traversal algorithm. J. ACM, 23(2) :368388, avril 1976.

[8] X. Jiang. _ An adaptive contour closure algorithm and its experimental evaluation. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 22(11) :1252_1265, 2000.

[9] N.Senthilkumaran and R. Rajesh,"Edge Detection Techniques for Image Segmentation – A Survey of Soft Computing Approaches", International Journal of Recent Trends in Engineering, Vol. 1, No. 2, May 2009.

[10] Zhang, Y., Han, X., Zhang, H., & Zhao, L. (2017). Edge detection algorithm of image fusion based on improved Sobel operator. 2017 IEEE 3rd Information Technology and Mechatronics Engineering Conference (ITOEC). doi:10.1109/itoec.2017.8122336.

[11] Ganesan P, G.Sajiv.(2017) A Comprehensive Study of Edge Detection for Image Processing Applications. 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). **doi:** 10.1109/ICIIECS.2017.8275968

[12] Sivaiah Bellamkonda., N. P. Gopalan .(2018) Facial Expression Recognition Using Kirsch Edge Detection, LBP and Gabor Wavelets. 2018 IEEE 2nd Conference on Intelligent Computing and Control Systems(ICICCS). **doi:** 10.1109/ICCONS.2018.8662971.

[13] Ravi S and A M Khan, "Operators Used In Edge Detection Computation: A Case Study", International International Journal of Applied Engineering Research, vol. 7 , 2012.

[14] K. K. Singh, A. Singh,"A Study of Image Segmentation Algorithms for Different Types of Images", International Journal of Computer Science Issues, Vol. 7, Issue 5, 2010.

[15] S.L. Horowitz et T. Pavlidis. Picture segmentation by a directed split and merge procedure. Dans Computer Methods in Images Analysis, pages 10111, 1977.

[16] William I. Grosky, Ramesh Jain. _ Optimal quadtrees for image segments. Pattern Analysis and Machine Intel- ligence, IEEE Transactions on, PAMI-5(1) :77_83, 1983.

[17] Safia Bekhouche, Yamina Mohamed Ben Ali (2018).Improvement of Quadree-Based Region Segmentation. 2018 International Conference on Advances in Computing and Communication Engineering (ICACCE) .doi:10.1109/ICACCE.2018.8441699.

[18] Baoan Han .Department of Computer Engineering :Watershed Segmentation Algorithm Based on Morphological Gradient Reconstruction 2015.

[19] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. IEEE Conference on Computer Vision and Pattern Recognition, pages 886-893, 2005.

[20] B. Wu and R. Nevatia. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. IEEE International Conference on Computer Vision, pages 90-97, 2005.

[21] Bo Wu • Ram Nevatia Detection and Segmentation of Multiple, Partially Occluded Objects by Grouping, Merging, Assigning Part Detection Responses 2008

[22] Ke, Y., Sukthankar, R. (2004). PCA-SIFT: A more distinctive representation for local image descriptors. In Computer Vision and Pattern Recognition (CVPR 2004), 27 June – 2 July 2004. IEEE, Vol. 2, 506-513.

[23] Mortensen, E.N., Deng, H., Shapiro, L. (2005). A SIFT descriptor with global context. In Computer Vision and Pattern Recognition (CVPR 2005), 20-25 June 2005. IEEE, Vol. 1, 184-190.

[24] Abdel-Hakim, A.E., Farag, A.A. (2006). CSIFT: A SIFT descriptor with color invariant characteristics. In Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006. IEEE, Vol. 2, 1978-1983.

[25] Bay, H., Tuytelaars, T., Gool, L.V. (2006). SURF: Speeded up robust features. In Computer Vision – ECCV 2006 : 9th European Conference on Computer Vision, 7-13 May 2006. Springer, Part II, 404-417.

[26] Morel, J.M., Yu, G. (2009). ASIFT: A new framework for fully affine invariant image comparison. SIAM Journal on Imaging Sciences, 2 (2), 438-469.

[27] Patil, S., & Kulkarni, U. (2019). Accuracy Prediction for Distributed Decision Tree using Machine Learning approach. 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI). doi:10.1109/icoei.2019.8862580.

[28] Ryan W Wolcott and Ryan M Eustice. Visual localization within LIDAR maps for automated urban driving. In Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on, pages 176–183. IEEE, 2014.

[29] Christian Häne, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, Paul Furgale, Torsten Sattler and Marc Pollefeys. 3D visual perception for self-driving cars using a multi-camera system: Calibration, mapping, localization, and obstacle detection. Image and Vision Computing, vol. 68, pages 14–27, 2017.

[30] Xia, Z., & Tang, S. (2019). Robust self-localization system based on multi-sensor information fusion in city environments. 2019 International Conference on Information Technology and Computer Application (ITCA). doi:10.1109/itca49981.2019.00011

[31] Cai H., Hu Z., Huang G., Zhu D., Su X. Integration of GPS, monocular vision, and high definition (HD) map for accurate vehicle localization. Sensors. 2018;18:3270. doi: 10.3390/s18103270.

[32] W Ma et al, "Exploiting Sparse Semantic HD Maps for Self-Driving Vehicle Localization", arXiv 1908.03274, 2019

[33]Tomtom hd map with roaddna. [Consulté le 1 septembre 2020]. Disponible à l'adresse : https://www.tomtom.com/automotive/automotive-solutions/automated-driving/hd-map-roaddna/. .

[34] Christoffer Wilhelm Gran: HD-Maps in Autonomous Driving Master's thesis in Computer Science Supervisor: Frank Lindseth June 2019

[35] Dai-Duong Nguyen. A vision system based real-time SLAM applications. Hardware Architecture [cs.AR]. Université Paris-Saclay, 2018. English. ffNNT : 2018SACLS518ff. fftel-02398765f

[36] Joshi, A., & James, M. R. (2015). Generation of Accurate Lane-Level Maps from Coarse Prior Maps and Lidar. IEEE Intelligent Transportation Systems Magazine, 7(1), 19–29. doi:10.1109/mits.2014.2364081

[37] Gwon, G.-P., Hur, W.-S., Kim, S.-W., & Seo, S.-W. (2017). Generation of a Precise and Efficient Lane-Level Road Map for Intelligent Vehicle Systems. IEEE Transactions on Vehicular Technology, 66(6), 4517–4533. doi:10.1109/tvt.2016.2535210

[38] Zheng, S., & Wang, J. (2017). High definition map-based vehicle localization for highly automated driving: Geometric analysis. 2017 International Conference on Localization and GNSS (ICL-GNSS). doi:10.1109/icl-gnss.2017.8376252

[39]Consulté[le28aout2020].Disponible à l'adresse: https://www.openstreetmap.org/#map=5/28.413/1.653

[40] Lookingbill A, Weiss-Malik M (2013) Project ground truth: Accurate maps via algorithms and elbow grease, google i/o, 2013. URL https://www. youtube.com/watch?v=FsbLEtS0uls

[41] Stanojevic R, Abbar S, Thirumuruganathan S, Chawla S, Filali F, Aleimat A (2018), MapFuse: Road Network Fusion for Incremental Map Updates arXiv:1802.02351

[42] Deep Learning Yann LeCun, Yoshua Bengio Geoffrey Hinton. 2015.

[43]Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. http://www.deeplearningbook.org. MIT Press, 2016.

[44] Tom MITCHELL, Machine learing, McGraw-Hill Science/Engineering/Math, 1997.

[45] Neural Networks and Deep Learning, Charu C. Aggarwal. Springer, September 2018

[46] edited by James A. Anderson and Edward Rosenfeld. Cambridge, MA, USA: MIT Press,1988. Chapter A Logical Calculus of the Ideas Immanent in Nervous Activity, pages 15–27.ISBN: 0-262-01097-6.

[47] Frank Rosenblatt. The Perceptron : A Probabilistic Model for Information Storage and Organization In The Brain. 1957

[48] M. Minsky and S. Papert. Perceptrons. Cambridge, MA: MIT Press, 1969

[49] Paul JohnWerbos. « Beyond Regression : New Tools for Prediction and Analysis in the Behavioral Sciences ». Harvard University, 1975.

[50] D. E. Rumelhart, G. E. Hinton et R. J. Williams. « Learning Internal Representations by Error Propagation ». Dans : sous la dir. de David E. Rumelhart, James L. Mc-Clelland et CORPORATE PDP Research Group. Cambridge, MA, USA : MIT Press, 1986, p. 318-362.

[51] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," Haifa, 2010

[52] Andrew L. Maas, Awni Y. Hannun et Andrew Y. Ng. « Rectifier Nonlinearities Improve Neural Network Acoustic Models ». Dans : ICML Workshop on Deep Learning for Audio, Speech and Language Processing. 2013

[53] Kaiming He et al. « Delving Deep into Rectifiers : Surpassing Human-Level Performance on ImageNet Classification ». Dans : Proceedings of the IEEE International Conference on Computer Vision. IEEE International Conference on Computer Vision (ICCV). Déc. 2015, p. 1026-1034.

[54] Djork-Arné Clevert, Thomas Unterthiner et Sepp Hochreiter. « Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs) ». Dans : Proceedings of the International Conference on Learning Representations (ICLR). 23 nov. 2015.

[55] The Number of Hidden Layers. In : Heaton Research [en ligne]. 1 juin 2017. [Consulté le 14 avril 2020]. Disponible à l'adresse : https://www.heatonresearch.com/2017/06/01/hidden-layers.html.

[56] J. L. Elman, "Finding structure in time," Cognitive science, vol. 14, no. 2, pp. 179– 211, 1990.

[57] M. I. Jordan, "Serial order : A parallel distributed processing approach," Advances in psychology, vol. 121, pp. 471–495, 1997.

[58] Caroline Etienne. Apprentissage profond appliqué à la reconnaissance des émotions dans la voix. Intelligence artificielle [cs.AI]. Université Paris-Saclay, 2019. Français. ffNNT : 2019SACLS517ff. fftel02479126f.

[59] Priyesh Patel, Meet Nandu, Purva Raut "Initialization of Weights in Neural Networks".November 2018

[60] Katarzyna Janocha, Wojciech Marian Czarnecki, On Loss Functions for Deep Neural Networks in Classification Feb 2017

[61] Corentin HARDY. Contribution au développement de l'apprentissage profond dans les systèmes distribués. Université RENNES 2019.

[62] Anders Krogh et John A. Hertz. « A Simple Weight Decay Can Improve Generalization ». Dans : Proceedings of the 4th International Conference on Neural Information Processing Systems. NIPS'91. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1991, p. 950-957. isbn : 978-1-55860-222-9

[63] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

[64] G. E. Hinton_, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov (2012) Improving neural networks by preventing co-adaptation of feature detectors

[65] Li Wan et al. « Regularization of Neural Networks Using DropConnect ». Dans : International Conference on Machine Learning. International Conference on Machine Learning. 13 fév. 2013, p. 1058-1066.

[66] Sergey Ioffe et Christian Szegedy. « Batch Normalization : Accelerating Deep Network Training by Reducing Internal Covariate Shift ». Dans : *Proceedings of the 32nd International Conference on Machine Learning*. International Conference on Machine Learning (ICML). 2015

[67] D. H. Hubel & T. N. Wiesel. Receptive Fields and Functional Architecture of Monkey Striate Cortex. Journal of Physiology (London), vol. 195, pages 215–243, 1968.

[68] Kunihiko Fukushima. « Neocognitron : A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position ». Dans : Biological Cybernetics 36.4 (1er avr. 1980), p. 193-202. issn : 0340-1200, 1432-0770.

[69] Yann LeCun et al. « Backpropagation Applied to Handwritten Zip Code Recognition ».Dans : Neural Computation 1.4 (déc. 1989), p. 541-551.

[70] Yann LeCun et al. « Gradient-Based Learning Applied to Document Recognition ».Dans : Proceedings of the IEEE 86.11 (nov. 1998), p. 2278-2324.

[71] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Adv. Neural Inf. Process. Syst., pp. 1–9, 2012.

[72] Autonomous Driving System with Road Sign Recognition using Convolutional Neural Networks february 2019

[73] Road sign recognition with Convolutional Neural Network Amal Bouti ; Mohamed Adnane Mahraz ; Jamal Riffi ; Hamid Tairi april 2018.

[74] Real-Time Facial Expression Recognition Based on CNN Keng-Cheng Liu ; Chen-Chien Hsu ; Wei-Yen Wang ; Hsin-Han Chiang 20-21 July 2019.

[75] Nicolas Audebert. Classification de données massives de télédétection. Intelligence artificielle [cs.AI]. UNIVERSITE BRETAGNE LOIRE SUD, 2018. Français. fftel-01960350f

[76] M. D. Zeiler and R. Fergus, “Visualizing and Understanding Convolutional Networks,” arXiv Prepr. arXiv1311.2901v3, vol. 30, no. 1–2, pp. 225–231, 2013.

[77] Karen Simonyan et Andrew Zisserman. « Very Deep Convolutional Networks for Large-Scale Image Recognition ». Dans : *Proceedings of the International Conference on Learning Representations (ICLR)*. Mai 2015

[78] Christian Szegedy et al. « Going Deeper with Convolutions ». Dans : Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Juin 2015, p. 1-9

[79] Kaiming He et al. « Deep Residual Learning for Image Recognition ». Dans : Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, United States, juin 2016, p. 770-778

[80] François Chollet. « Xception : Deep Learning with Depthwise Separable Convolutions ». Dans : Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Honolulu, United States, juil. 2017, p. 1800-1807

[81] Gao Huang et al. « Densely Connected Convolutional Networks ». Dans : Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. T. 1. 2017

[82] Corentin HARDY. Contribution au développement de l’apprentissage profond dans les systèmes distribués. Université RENNES 2019.

[83] Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. arXiv 2018, arXiv:1804.02767.

[84] Object Tracking: A Survey, Alper Yilmaz ,Ohio State UniversityOmar Javed,ObjectVideo, Inc.and Mubarak Shah ,University of Central Florida.

[85] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, 2016, pp. 779-788. doi: 10.1109/CVPR.2016.91.

[86] Redmon, J.; Farhadi, A. YOLO 9000: Better, Faster, Stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 22–25 July 2017; pp. 6517–6525.

[87] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2117–2125, 2017.

[88] He, K.; Gkioxari, G.; Dollar, P.; Girshick, R. Mask R-CNN. In Proceedings of the International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.

[89] https://github.com/keshik6/KITTI-2d-object-detection [Consulté le 18 septembre 2020].

[90] UNRUH, Unruh, 2017. What is the TensorFlow machine intelligence platform? In : Opensource.com [en ligne]. 9 novembre 2017. [Consulté le 17 mai 2020]. Disponible à l'adresse : https://opensource.com/article/17/11/intro-tensorflow.

[91] https://github.com/u39kun/deep-learning-ben [Consulté le 17 mai 2020].

[92] https://opencv.org/about/ [Consulté le 17 mai 2020].