

Table des matières

Table des matières	2
Table des figures	4
Liste des tableaux	6
1 Concepts de base	11
1.1 Introduction à l'analyse de l'image	11
1.1.1 Définition d'une image	11
1.1.2 Les différents types de format d'image	13
1.1.3 Caractéristiques de l'image	13
1.2 La vision par ordinateur	14
1.2.1 Extraire des informations à partir d'images	15
1.3 Généralisation sur la Robotique	16
1.3.1 Historique	16
1.3.2 Définition d'un robot	16
1.3.3 Programmation des robots	17
1.4 Les véhicules autonomes	17
1.4.1 Aspect Technique	18
2 État de l'art de la détection d'objets	23
2.1 Reconnaissance d'objets	23
2.1.1 Pré-traitements	24

2.1.2	Extraction des caractéristiques	26
2.1.3	Classification	26
2.2	La Détection d'objets	27
2.3	Travaux connexes	28
2.3.1	Histogramme de Gradient Orienté	28
2.3.2	Structure Picturale	29
3	L'apprentissage automatique et l'apprentissage en profondeur	31
3.1	Définition de l'apprentissage automatique	31
3.1.1	Types d'apprentissage	32
3.1.2	Généralisation, sur-apprentissage et régularisation	32
3.2	Définition des réseaux de neurones	33
3.2.1	Perceptrons	33
3.2.2	L'architecture des réseaux de neurones	37
3.2.3	L'apprentissage en profondeur	38
4	La méthode utilisée pour la détection d'objet	47
4.1	Réseaux de neurones à convolution (CNN)	47
4.1.1	Le filtre	48
4.1.2	Le Pooling	50
4.1.3	La convolution	50
4.1.4	La classification : Couche entièrement connectée (couche FC)	51
4.2	R-CNN	52
4.2.1	Fast R-CNN	53
4.2.2	Faster R-CNN :	55
4.3	Détecteur SSD	56
4.4	Les architectures neuronales classiques	56
4.4.1	Inception Network	59
5	Développement, Résultats et Expérimentations	61
5.1	Environnements de travail	61
5.1.1	Environnement matériel	61
5.1.2	Environnement logiciel	62
5.2	Les données utilisées	64
5.2.1	Les sources	64
5.2.2	Le format et l'organisation	65
5.3	Le suivi des performances	66
5.3.1	Les métriques d'évaluations	66
5.3.2	La technique	67
5.4	Configuration de notre modèle	67
5.4.1	Configuration de modèles	68

5.4.2	Configuration de train_config	68
5.5	Test et prédiction	69
5.6	Implémentation	76
Bibliographie		80

Table des figures

1.1.1	Exemple de codage un morceau d'image en niveaux de gris à l'aide d'entiers compris entre 0 pour le noir et 255 pour le blanc.	12
1.1.2	variation du nombre de pixels.	14
1.1.3	Quantification des niveaux de gris : passage d'une suite continue de valeurs à un ensemble discret de N valeurs.	14
1.1.4	Variation du nombre de niveaux de gris pour la même image.	14
1.2.1	Vision par ordinateur	15
1.4.1	Les 6 degrés d'autonomisation selon la Society of Automotive Engineers	18
1.4.2	Emplacement et types de capteurs d'une voiture autonome de niveau 4/5.	19
1.4.3	Voiture autonome	21
2.1.1	Les étapes de la reconnaissance de formes	24
2.1.2	La détection de contours d'un visage	25
2.2.1	La détection des voitures sur une route	27
2.2.2	Pyramide d'images	28
3.2.1	Modèle d'un réseau de neurone, perceptron	34
3.2.2	Illustration des étapes pré-activation et activation sur un perceptrons .	36
3.2.3	La fonction ReLU	37
3.2.4	Réseau de neurones multicouches	37
3.2.5	Exemple d'architecture d'un réseau qui déterminer si une image montre un visage humain ou non	38
3.2.6	illustration de la méthode de descente de gradient dans le cas où $j(w) = w^2$	40
3.2.7	Fonction objective : L'erreur en fonction du paramètre «a». L'erreur minimum se situe aux alentours de 3	41

3.2.8 Différentes pentes pour différentes valeurs de a .	42
3.2.9 Accélération de SGD par la méthode du momentum et réduction des oscillation	44
4.1.1 Présentation d'une images RVB	48
4.1.2 Illustration du déplacement du filtre au long de l'image	49
4.1.3 La multiplication de matrice est effectuée entre les piles Kernel_n et longeur_n , et tous les résultats sont additionnés avec le biais pour nous donner une sortie d'entité conjuguée de canal à profondeur réduite.	49
4.1.4 Exemple de calcul de max et average pooling	50
4.1.5 Illustration de la couche de convolution	51
4.1.6 Couche entièrement connectée (couche FC)	52
4.2.1 Démonstration du Convolutional Neural Networks (R-CNN)	52
4.2.2 Architecture du Convolutional Neural Networks (R-CNN)	53
4.2.3 Architecture de Fast R-CNN	54
4.2.4 RoI Max Pooling	54
4.2.5 Architecture de Faster R-CNN	55
4.4.1 Illustration de l'architecture ZF-NET	57
4.4.2 Illustration de l'architecture VGG16	58
4.4.3 Illustration de l'architecture ResNet	58
4.4.4 La convolution 5x5 la plus à gauche de l'ancien module de création est maintenant représentée par deux convolutions 3x3.	59
5.1.1 Les logos de Python, TensorFlow, et NumPy	63
5.1.2 Capture de l'outil LabelImg	63
5.2.1 Exemples d'images utilisées	64
5.5.1 Images d'évaluation du modèle SSD_inception_v2	70
5.5.2 Les performances du modèle SSD_inception_v2	71
5.5.3 Images d'évaluation du modèle Faster_resnet	72
5.5.4 Les performances du modèle Faster_resnet	73
5.5.5 Images d'évaluation du modèle créé par Faster_inception_v2	74
5.5.6 Les performances du modèle créé par Faster_inception_v2	75

<i>Liste des tableaux</i>	6
---------------------------	---

5.6.1 ROBUCAR du centre de recherche CDTA	77
---	----

Liste des tableaux

5.1 Les résultats de l'ensemble de tests avec le metrique de PASCAL VOC	66
5.2 Tableaux comparatifs des résultats	75

Introduction générale

Pour les humains la détection d'objets est une tâche qui s'effectue d'une manière fiable, inconsciente, rapide et sans effort, en revanche pour les machines programmables comme les ordinateurs et les robots, l'interprétation des informations visuelles de l'environnement reste plus compliquée et un défi pour les chercheurs.

Citons que ces dernières années le développement des véhicules autonomes voit une croissance remarquable, ces véhicules doivent pouvoir analyser l'environnement pour extraire les informations nécessaires qui leurs permettent de détecter les objets routiers. La détection doit se faire en temps réel pour que le véhicule évite toutes sorte d'obstacles ce qui exige un programme de détection rapide.

À partir des images des caméras, les voitures autonomes sont chargées de détecter tous les obstacles et de les classer par catégories (piétons, vélos, motos, bus, camions...), d'identifier les panneaux de signalisations et les feux de circulation routière, et prédire leur positions. Mais avant de pouvoir monter à bord d'une voiture, les algorithmes qui effectuent les tâches de détection et localisation doivent d'abord faire la preuve de leur efficacité en laboratoire.

Le présent travail traite donc cette problématique de reconnaissance automatique des objets et des obstacles qui entourent les véhicules autonomes, quel que soit le type d'objet (piétons, véhicules, animaux, panneaux ...), vu que l'apprentissage profond (Deep Learning), et en particulier les réseaux de neurones convolutifs (Convolutional Neural Network), ont largement contribué à la progression de l'état de l'art et à l'amélioration de la détection d'objets, nous proposons une solution de classification, de localisation et de détection d'objets en exploitant l'apprentissage des réseaux de neurones profonds.

Dans un premier temps, on entraîne les algorithmes d'apprentissages sur un vaste jeu de données collectées par les capteurs dans de multiples conditions de circulation : autoroutes, routes (rurale, nationale), en ville, sous différentes conditions météo ou avec plus ou moins de trafic... etc. Sur des millions d'images réunissant plusieurs catégories d'objets, les algorithmes apprennent ainsi à les détecter et à les classer. Cet apprentissage se fait de façon supervisée.

Pour répondre au problème de la détection, de localisation et de classification dans des environnements dynamiques, nous nous sommes fixés sur les objectifs

suivants :

- Générer un ensemble de données contenant les images des objets routiers, et faire la correspondance entre les entrées et les sorties.
- Concevoir un modèle d'apprentissage automatique et utiliser l'ensemble de données pour effectuer son entraînement.
- L'algorithme doit atteindre une performance en temps réel de façon à minimiser le temps lorsqu'un objet détecté peut être un obstacle.
- La validation finale des algorithmes en conditions réelles en les testant et les améliorant sur des données différentes de celles sur lesquelles ils ont été entraînés.

Pour réaliser notre travail nous l'avons structuré de la façon suivante :

Le premier chapitre est dédié à l'explication des concepts de base et des notions générales dans les domaines de la robotique, la vision par ordinateur.

Le deuxième chapitre est consacré à la détection d'objets et à la présentation de quelques algorithmes utilisés pour la détection.

Le troisième chapitre traite les notions d'apprentissage profond/automatique ainsi que des réseaux de neurones.

Dans le quatrième chapitre nous introduisons le réseau de neurones convolutif et ces différents modèles basés sur les régions d'intérêts.

L'explication du choix de l'architecture et la base de données utilisée ainsi que les améliorations que nous avons réalisé pour la rendre plus performante, et la comparaisons des résultats sont présentés et discutés dans le cinquième chapitre.

Chapitre 1 : Concepts de base

Chapitre 1

Concepts de base

Introduction

Dans de nombreuses industries, les robots ont besoin d'un mécanisme de reconnaissance autonome des objets, cela exigerait la construction des représentations visuelles souples et robustes du monde. À bien des égards, la vision par ordinateur est un problème complet de l'intelligence artificielle, et comme pour d'autres problèmes d'intelligence artificielle, le défi de la vision peut être décrit en termes de construction d'un convertisseur de signal en symbole.

Nous commençons ce chapitre par la présentation de la vision par l'ordinateur, suivi par quelques notions sur le traitement d'image, et nous terminons par citer les notions de bases de la Robotique et des véhicules autonomes.

1.1 Introduction à l'analyse de l'image

Les images sont partout autour de nous, à la télévision, dans la presse papier, sur Internet, ... Ce monde d'images, dans lequel nous vivons toutes et tous quotidiennement, possède ses codes, ses références, ...

1.1.1 Définition d'une image

Une image est une représentation planaire d'une scène ou d'un objet situé en général dans un espace tridimensionnel. Elle est issue du contact des rayons lumineux provenant des objets formant la scène avec un capteur (caméra, scanner, rayons X, ...). Il ne s'agit en réalité que d'une représentation spatiale de la lumière[16]. L'image est considérée comme un ensemble de points auxquels sont affectés une grandeur physique (luminance, couleur). Ces grandeurs peuvent être continues (image analogique) ou bien discrètes (images digitales). Mathématique-

ment, l'image représente une fonction continue IF, appelée fonction image, de deux variable spatiale représentée par IF(x, y) mesurant la nuance du niveau de gris de l'image aux coordonnées (x, y).

La fonction Image peut se représenter sous la forme suivante :

$$\begin{aligned} IF: \mathbb{R}^2 &\rightarrow \mathbb{R} \\ (x, y) &\rightarrow IF(x, y) \end{aligned}$$

où \mathbb{R}^2 : ensemble des réelles.

x et y : deux variables réelles.

La plupart des images que nous traitons en vision par ordinateur sont numériques. Cette discrétisation est obtenue grâce à l'échantillonnage¹ d'un espace bidimensionnel sur une grille régulière, produisant finalement une représentation de l'image sous la forme d'une matrice de valeurs entières.

Ayant utilisé cette définition des images, nous pouvons les écrire sous forme de coordonnées dans une matrice :

$$\begin{vmatrix} \ddots & & \vdots & & \vdots & & \vdots & & \ddots \\ \cdots & if[-1, 1] & & if[0, 1] & & if[1, 1] & & \cdots \\ \cdots & if[-1, 0] & & if[0, 0] & & if[0, 1] & & \cdots \\ \cdots & if[-1, -1] & & if[0, -1] & & if[1, -1] & & \cdots \\ \ddots & & \vdots & & \vdots & & \vdots & & \ddots \end{vmatrix}$$

Une image peut également être traitée comme un mappage de fonctions $\mathbb{R}^2 \rightarrow \mathbb{R}^3$. Par exemple, les intensités RVB d'un pixel donné peuvent être écrites sous la fonction g ci-dessous :

$$g[x, y] = \begin{bmatrix} r[x, y] \\ g[x, y] \\ b[x, y] \end{bmatrix}$$

où r, g, b : $[a, b] \times [a, b] \rightarrow [0, 255]$

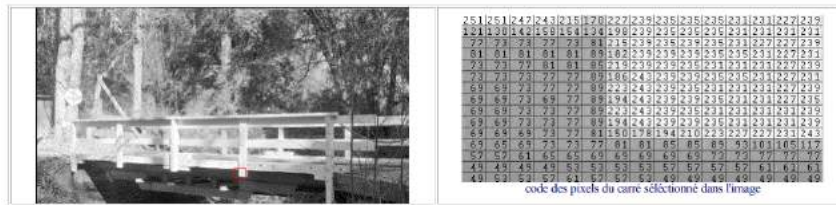


FIGURE 1.1.1 – Exemple de codage un morceau d'image en niveaux de gris à l'aide d'entiers compris entre 0 pour le noir et 255 pour le blanc.

1. échantillonnage détermine la résolution spatiale de l'image numérisée

1.1.2 Les différents types de format d'image

- **Image couleur RVB** : L'œil humain analyse la couleur à l'aide de trois types de cellules photo[16]. Ces cellules sont sensibles aux basses, moyennes, ou hautes fréquences (rouge, vert, bleu). Pour représenter la couleur d'un pixel, il faut donc donner trois nombres, qui correspondent au dosage de trois couleurs de base : *Rouge*, *Vert*, *Bleu*. On peut ainsi représenter une image couleur par trois matrices, chacune correspondant à une couleur de base.
- **Image d'intensités** : C'est une matrice dans laquelle chaque élément est un réel compris entre 0 (noir) et 1 (blanc). On parle aussi d'image en niveaux de gris, car les valeurs comprises entre 0 et 1 représentent les différents niveaux de gris.
- **Image binaire** : Une image binaire est une matrice rectangulaire[16] dont l'élément vaut 0 ou 1. Lorsque l'on visualise une telle image, les 0 sont représentés par du noir et les 1 par du blanc.

1.1.3 Caractéristiques de l'image

L'image est un ensemble structuré d'informations caractérisées par les paramètres suivants :

- **Pixel** : Le pixel est l'abréviation du mot « Picture élément », il s'agit d'une unité de surface permettant de définir la base d'une image numérique. Il matérialise un point donné (x, y) du plan de l'image. L'information présentée par le pixel est le niveau de gris (ou la couleur) prélevée à l'emplacement correspondant dans l'image réelle [16]. La différence entre image monochrome et image couleur réside dans la quantité d'informations contenue dans chaque pixel, par exemple dans une image couleur (RVB : Rouge, Vert, Bleu) la valeur d'un pixel est représentée sur trois octets pour chaque couleur.
- **Dimension & Résolution** : La dimension est la taille de l'image. Elle se présente sous forme d'une matrice dont les éléments sont des valeurs numériques représentatives des intensités lumineuses (pixels). Le nombre de lignes de cette matrice multiplié par le nombre de colonnes nous donne le nombre total de pixels dans une image. Par contre, la résolution est la clarté ou la finesse de détails atteinte par un moniteur ou une imprimante dans la production d'images. Sur les moniteurs d'ordinateurs, la résolution est exprimée en nombre de pixels par unité de mesure (pouce ou centimètre)[16]. On utilise aussi le mot résolution pour désigner le nombre total de pixels horizontaux et verticaux sur un moniteur. Plus ce nombre est grand, plus la résolution est meilleure.

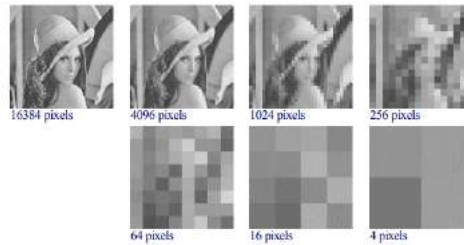


FIGURE 1.1.2 – variation du nombre de pixels.

- **Niveau de gris** : C'est la valeur d'intensité lumineuse d'un pixel. Cette valeur peut aller du noir (0) jusqu'au blanc (255) en passant par les nuances qui sont contenues dans l'intervalle $[0, 255]$.

FIGURE 1.1.3 – Quantification des niveaux de gris : passage d'une suite continue de valeurs à un ensemble discret de N valeurs.

Elle correspond en fait à la quantité de la lumière réfléchie. Pour 8 bits, on dispose de 256 niveaux de gris dont 40 sont reconnus à l'œil nu. Plus le nombre de bit est grand plus les niveaux sont nombreux et plus la représentation est fidèle.



FIGURE 1.1.4 – Variation du nombre de niveaux de gris pour la même image.

1.2 La vision par ordinateur

Vision par ordinateur (Computer Vision) : utiliser un ordinateur pour interpréter le monde extérieur via des images.

Images \longrightarrow Objets

Que ce soit un ordinateur ou un animal, la vision se compose de deux éléments :

Tout d'abord, un dispositif de détection capture autant de détails d'une image que possible. L'œil captera la lumière qui traverse l'iris et la projettera sur la rétine, où des cellules spécialisées transmettront des informations au cerveau par le biais de neurones. Une caméra capture les images de la même manière et transmet les pixels à l'ordinateur. Dans cette partie, les caméras sont meilleures que les humains car elles peuvent voir l'infrarouge, voir plus loin ou avec plus de précision.

Deuxièmement, le dispositif d'interprétation doit traiter les informations et en extraire le sens. Le cerveau humain résout ce problème en plusieurs étapes dans différentes régions du cerveau.

La vision par ordinateur est l'une des disciplines de l'intelligence artificielle à la croissance la plus rapide et la plus excitante du monde universitaire et industriel, elle est capable de percevoir, de comprendre et de reconstruire le monde visuel complexe[5].

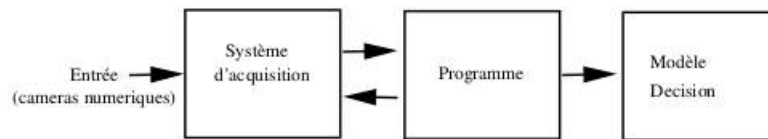


FIGURE 1.2.1 – Vision par ordinateur

1.2.1 Extraire des informations à partir d'images

Nous pouvons diviser les informations tirées des images en vision par ordinateur en deux catégories : **les mesures** et **les informations sémantiques**.

1. La vision comme instrument de mesure

Les robots naviguant dans un lieu inconnu doivent pouvoir analyser leur environnement pour calculer le meilleur chemin. À l'aide de la vision par ordinateur, nous pouvons mesurer l'espace autour d'un robot et créer une carte de son environnement. Les caméras stéréo donnent des informations sur la profondeur, comme nos deux yeux, par triangulation²[5].

1. Une source d'informations sémantiques

En plus des informations de mesure, une image contient une quantité très dense d'informations sémantiques³. Nous pouvons étiqueter des objets dans une image,

2. Une stratégie de recherche qualitative permettant de tester la validité par la convergence d'informations provenant de différentes sources.

3. Des informations vitales extraites du contenu de l'image, principalement des objets d'image significatifs et de leurs relations mutuelles.

étiqueter toute la scène, reconnaître des personnes, reconnaître des actions, des gestes, des visages[5].

1.3 Généralisation sur la Robotique

La notion de robot, ou d'automate, remonte à l'époque médiévale. Même s'il n'existait pas de terme pour décrire ce que nous appelons aujourd'hui des robots, les gens de cette époque ont tout de même imaginé des mécanismes capables d'exécuter des tâches humaines.

1.3.1 Historique

- En Égypte, mâchoire articulée d'un masque Anubis, le bras de Amon bouge pour designer le nouveau pharaon[11].
- En 1818, Mary Shelly a écrit *Frankenstein*, un récit qui relate la fabrication d'une créature d'apparence humaine. Le robot imaginé par cette auteure ressemblait à un homme, mais fonctionnait comme une machine. Il était composé d'éléments aux formes humaines maintenus ensemble à l'aide de boulons et d'écrous[11].

Même si le concept de robot trouve son origine dans une époque très lointaine, il a fallu attendre les années 1940⁴ et l'arrivée des ordinateurs pour que les robots des temps modernes fassent leur apparition.

1.3.2 Définition d'un robot

Un robot est un système mécanique poly-articulé mû par des actionneurs et commandé par un calculateur qui est destiné à effectuer une grande variété de tâches[12].

Il existe plusieurs types de robots répondant à différentes fonctions. Nous citons les plus importants :

— Robots manipulateurs

Un manipulateur reprogrammable multifonctionnel conçu pour déplacer des matériaux, des outils, des pièces ou des composantes spécialisées à travers une série de mouvements programmés pour effectuer une tâche précise[12].

— Robots mobiles

Les robots mobiles ont une place particulière en robotique. Leur intérêt réside dans leur mobilité qui ouvre des applications dans de nombreux domaines. Comme les

4. "Elsie the tortoise" de Gray ("Machina speculatrix") et "la bête" de Johns Hopkins. "Shakey" est une petite boîte instable sur roues qui utilise la mémoire et le raisonnement logique pour résoudre les problèmes et naviguer dans son environnement.

robots manipulateurs, ils sont destinés à assister l'homme dans les tâches pénibles (transport de charges lourdes), monotones ou en ambiance hostile (nucléaire, marine, spatiale, lutte contre l'incendie, surveillance...)[12].

L'autonomie du robot mobile est une faculté qui lui permet de s'adapter ou de prendre une décision dans le but de réaliser une tâche malgré un manque d'informations préliminaires ou éventuellement erronées.

A l'heure actuelle, on peut distinguer 3 générations de robots :

1. **Le robot est passif** : Il est capable d'exécuter une tâche qui peut être complexe, mais de manière répétitive, il ne doit pas y avoir de modifications intempestives de l'environnement. L'auto-adaptativité est très faible. De nombreux robots sont encore de cette génération.

2. **Le robot devient actif** : Il devient capable d'avoir une image de son environnement, et donc de choisir le bon comportement (sachant que les différentes configurations ont été prévues). Le robot peut se calibrer tout seul.

3. **Le robot devient « intelligent »** : Le robot est capable d'établir des stratégies, ce qui fait appel à des capteurs sophistiqués, et souvent à l'intelligence artificielle.

1.3.3 Programmation des robots

Le système de programmation est celui qui réalise l'interface entre les actions spécifiées par l'opérateur et le système de commande du robot et des périphériques de la cellule. Les fonctions essentielles que doit réaliser un système de programmation de robots sont les suivantes :

- Spécifier des mouvements correspondant à la grande diversité d'opérations que l'on trouve en robotique.
- Synchroniser les mouvements du robot.
- Gérer les actions des outils et communications entrées/sorties .
- Intégrer des informations sensorielles⁵ dans les primitives de mouvement.
- Contrôler l'exécution des programmes.
- Communiquer avec l'opérateur via une interface appropriée.

1.4 Les véhicules autonomes

Le véhicule autonome désigne un véhicule intelligent capable de se déplacer tout seul sans l'aide de conducteur. La voiture autonome est considérée comme un robot à savoir « un appareil automatique capable de manipuler des objets ou d'exécuter des opérations selon un programme fixe, modifiable ou adaptable[15].

5. Ces informations proviennent le plus souvent de capteurs.

Il existe 6 niveaux d'autonomisation des véhicules selon la Society of Automotive Engineers (SAE), du niveau 0 (notre bonne vieille voiture classique) au niveau 5 [15](automatisation complète).

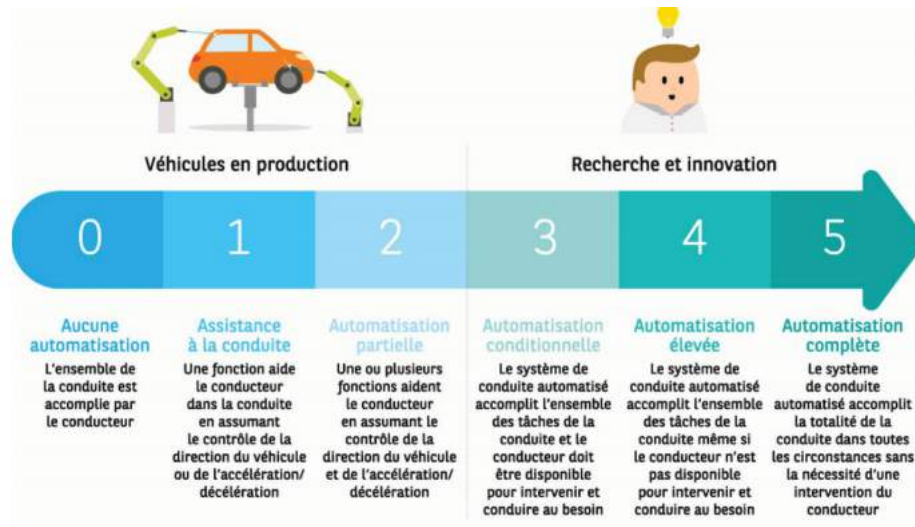


FIGURE 1.4.1 – Les 6 degrés d'autonomisation selon la Society of Automotive Engineers

1.4.1 Aspect Technique

Une voiture totalement autonome est équipée de plusieurs éléments matériels (capteurs) et logiciels qui coopèrent automatiquement grâce à une intelligence artificielle.

Acquisition d'informations

La voiture doit collecter les informations de son environnement, grâce aux capteurs dont le véhicule est équipé et à l'information partagée par les autres utilisateurs.

— Les capteurs

Les voitures autonomes sont équipées de capteurs, grâce auxquels elles sont censées pouvoir détecter les obstacles, les panneaux de signalisations, mais aussi les autres véhicules, les piétons et en anticiper les mouvements. Ces capteurs sont conçus pour fonctionner de jour comme de nuit et quelles que soient les conditions météorologiques.

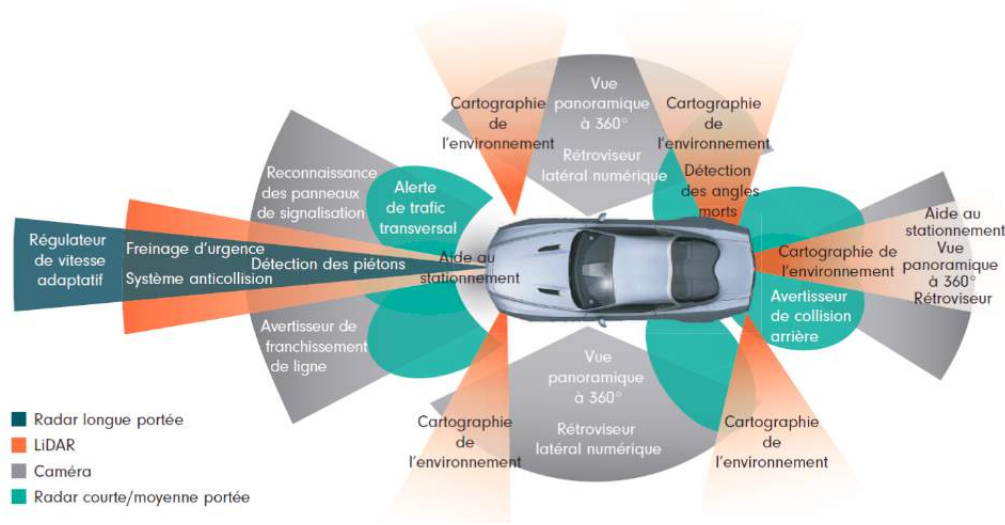


FIGURE 1.4.2 – Emplacement et types de capteurs d'une voiture autonome de niveau 4/5.

Une combinaison de capteurs sera utilisée dans les premières voitures entièrement autonomes :

-Les caméras : Les caméras remplacent les yeux de l'utilisateur permettant à la voiture de « voir » le monde qui l'entoure. Une voiture autonome peut être équipée de différents types de caméras : des caméras mono vision, stéréo vision, des caméras infrarouges, et il existe même des situations dans lesquelles les voitures sont munies de caméras 3D. Ces dernières sont utilisées pour détecter les obstacles puis les identifier grâce aux images de la chaussée et de la route. En outre, on dispose d'une caméra frontale pour identifier le marquage au sol, les panneaux de signalisation, les piétons... On trouve également des caméras sur les rétroviseurs ainsi qu'à l'arrière de la voiture pour couvrir les angles morts. Par conséquent, les caméras prendront un grand nombre d'images de manière continue tout au long du voyage et, par la suite, ces images seront analysées par des algorithmes afin qu'elles soient exploitables par le véhicule intelligent[15].

-Le RaDAR : « ou Radio Detection and Ranging » (détection de radio et mesure à distance) une technologie relativement ancienne et standardisée utilisée afin de détecter les objets métalliques, surtout en cas de mauvais temps lorsque les caméras éprouvent des difficultés[15].

-Le LiDAR : ou « Light Detection and Ranging » (détection de la lumière et mesure à distance), est une forme évoluée du radar qui utilise des lasers afin de créer des images en 3D des objets environnants avec un champ de vision de 360 degrés. Cette technologie aide à reconnaître les piétons et les autres véhicules, elle

est la plus avancée mais aussi la plus chère de tous les capteurs. Mais, plusieurs sociétés rivalisent afin d'améliorer le LiDAR[15].

-**Les ultrasons** : qui utilisent des ondes sonores afin d'aider à se diriger à faible vitesse (par exemple, au moment de se garer)[15].

-**Le GPS** : qui utilise la triangulation satellitaire afin de géolocaliser la position de la voiture. Sa précision n'est que d'environ 10 mètres, contre près de 10 centimètres pour les cartographies haute résolution. Il est utilisé en tant que système de redondance pour les autres capteurs[15].

-**Communication VtoX** : Afin que les voitures autonomes soient en mesure de communiquer en temps réel avec leurs paires mais aussi avec les infrastructures routières et leur environnement. La communication entre véhicules est essentielle afin d'éviter les accidents. Les communications entre véhicules utilisant un signal WiFi extrêmement puissant afin de parler aux autres voitures[15].

Planification action

La capacité de planification de l'ordinateur qui pilotera la voiture autonome est ensuite cruciale. Cette phase doit absolument rendre sûr le comportement du véhicule dès lors qu'il évolue dans un milieu ouvert en présence d'êtres humains imprévisibles. Elle s'appuie sur les éléments captés par l'ordinateur qu'il doit analyser très rapidement et confronter à une batterie de situations théoriques tout en évaluant l'impact de sa décision sur son environnement. Ainsi, en temps réel (voir même en anticipant), l'intelligence artificielle doit être capable de simuler et reproduire le comportement d'un humain au volant.

A ce stade, le véhicule autonome sera capable de prendre des décisions efficaces, efficientes et sécurisées qui le pousseront à l'action. Il sera ainsi en mesure de contrôler l'ensemble des éléments mécaniques de la voiture comme le volant, les freins, l'accélérateur, etc.

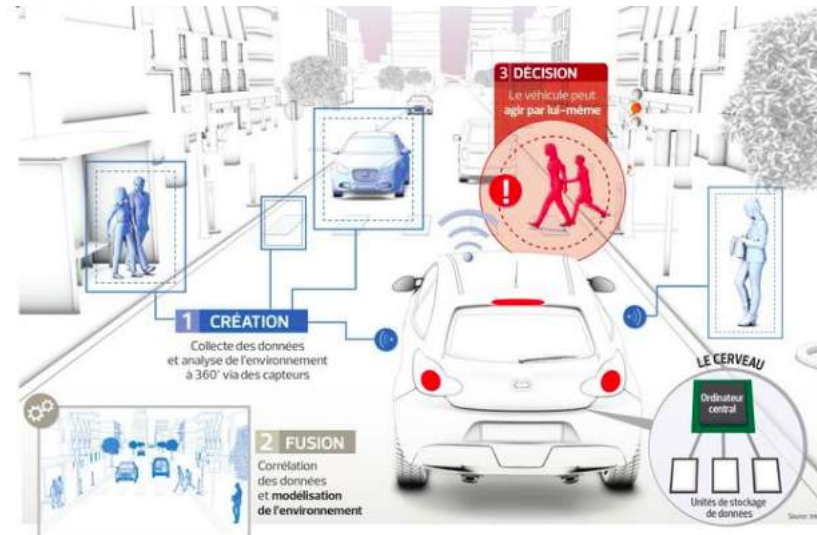


FIGURE 1.4.3 – Voiture autonome

Conclusion

Dans ce premier chapitre nous avons présenté quelques notions de base sur les domaines d'imagerie et de robotique. Le chapitre suivant fera mention des applications et méthodes utilisées pour la détection d'objets.

Chapitre 2 : État de l'art de la détection d'objets

Chapitre 2

État de l'art de la détection d'objets

Introduction

Depuis toujours, l'homme est fasciné par les automates. Les premiers automates amusaient et effectuaient des tâches répétitives. De nos jours ils sont dotés d'organes sensoriels leur permettant de voir, d'entendre et même de goûter. Parmi les automates les plus perfectionnés, on retrouve ceux qui sont capable de reconnaître les objets sur un milieu statique ou dynamique. Ce domaine est très actif en vision par ordinateur.

Dans le chapitre suivant nous abordons l'état de l'art de chaque phase de la reconnaissance d'objets à savoir : les pré-traitements, l'extraction des primitives, la classification et la localisation et détection, puis nous citons quelques travaux connexes à la détection d'objets.

2.1 Reconnaissance d'objets

Notre étude porte sur la reconnaissance des objets provenant de la segmentation des images. En général, l'objectif de la reconnaissance des objets est de développer un système qui se rapproche le plus de l'être humain dans sa capacité de voir et distinguer. Cependant, cette reconnaissance d'objets consiste à extraire d'une forme inconnue (image de chat, chien,...etc) une description plus simple et à établir sur celle-ci une décision. Cette décision est effectuée généralement en mesurant la ressemblance d'une forme inconnue avec un ensemble de références stockées en mémoire et décrites dans une représentation analogue. Les références sont obtenues lors d'une phase antérieure qualifiée d'apprentissage. Cette phase est très importante dans tout système de reconnaissance d'objets. Autrement dit

c'est un passage de l'espace observable vers un espace de décision d'appartenance à une classe. La construction d'un système de reconnaissance d'objets comprend plusieurs étapes distinctes. Les étapes qui sont décrites par Duda Hart[6] sont représentées par cette figure :

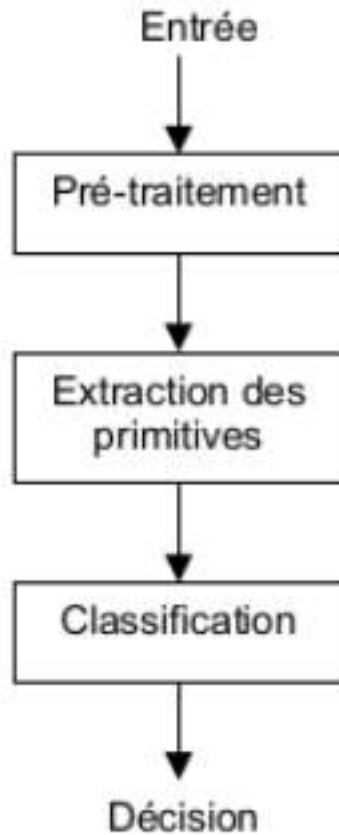


FIGURE 2.1.1 – Les étapes de la reconnaissance de formes

2.1.1 Pré-traitements

L'objectif des pré-traitements est de faciliter la caractérisation de la forme (caractère, chat, visage,... etc) ou de l'entité à reconnaître soit en nettoyant l'image représentant la forme ou en réduisant la quantité d'informations à traiter pour ne garder que les informations les plus significatives.

La segmentation est une phase des pré-traitements. Son but est de localiser et d'extraire le plus précisément possible les informations à reconnaître. Elle comprend deux étapes : la binarisation et la localisation des informations à reconnaître.

1. **La binarisation** : consiste à réduire la quantité d'informations à traiter lorsque l'image est en niveau de gris.
2. **La localisation des entités à reconnaître** : concerne des formes comme le visage ou le corps humain.



FIGURE 2.1.2 – La détection de contours d'un visage

Généralement, la segmentation d'une image est effectuée par l'utilisation de l'une des deux grandes approches basée sur l'extraction de contours (frontières) ou la croissance des régions.

L'approche contour : dans cette approche la détection de contours consiste à balayer l'image avec une fenêtre définissant la zone d'intérêt. A chaque position, un opérateur est appliqué sur les pixels de la fenêtre afin d'estimer s'il y a une transition significative au niveau de l'attribut choisi. La littérature est très riche par ces techniques.

Approche par filtrage optimal : Canny[2] est parti de la définition de certains critères qu'il faut maximiser pour détecter de façon correcte les contours dans une image. Il a ensuite formulé ces critères de façon mathématique et cherché la solution analytique du problème.

Le premier critère est l'efficacité de la détection, le deuxième critère est la précision de la localisation. Il est nécessaire d'ajouter un troisième critère pour assurer qu'on ait une seule réponse à un seul contour.

2.1.2 Extraction des caractéristiques

L'objectif de l'extraction des caractéristiques dans le domaine de la reconnaissance consiste à exprimer les caractéristiques sous une forme numérique ou symbolique appelée codage. Selon le cas, les valeurs de ces caractéristiques peuvent être réelles, entières ou binaires. Le vecteur des n caractéristiques constituées représente un point dans le nouvel espace à n dimensions. Cette étape de la reconnaissance consiste à extraire des caractéristiques permettant de décrire de façon non équivoque les formes appartenant à une même classe de caractères tout en les différenciant des autres classes.

Cette extraction se fait sur des images en niveaux de gris, en binaire, en contour binaire et selon une représentation vectorielle. Les caractéristiques sont classées en deux catégories : les caractéristiques structurelles (ou caractéristiques locales) basées sur une représentation linéaire du caractère (décomposition du caractère en segments de droites et courbes, contours du caractère, squelette) et les caractéristiques statistiques (ou primitives globales) basées sur la distribution des points, les transformations et les mesures physiques faites sur le caractère (surface, moments, projections). Ces deux types peuvent être combinés formant un vecteur de primitives.

2.1.3 Classification

Contrairement à la segmentation d'images, ici on cherche à identifier ce que représente chacune des régions segmentées. En imagerie aérienne par exemple, on cherche à déterminer les zones urbaines, les forêts, zones d'eau, les montagnes, . . .

Il s'agit de regrouper les différents éléments (pixels) en thèmes correspondants à la vérité du terrain. Le résultat est bien entendu une image segmentée. Dans ce domaine on procède, par l'attribution des pixels de l'image à des classes connues a priori (c'est la classification supervisée) ou à des classes inconnues (classification non supervisée).

Plusieurs approches sont utilisées à ce sujet ; nous décrivons ici les principes de deux d'entre elles :

- **Minimisation de distance** : consiste à rechercher la classe la plus proche pour chaque pixel, ou groupe de pixels si l'on travaille dans une fenêtre d'analyse centrée sur le pixel courant. La notion de proximité est liée à la distance considérée.

- **Les Réseaux de neurones** : s'avèrent capables de traiter des problèmes complexes de reconnaissance de forme, ou de simulation de processus non linéaires et/ou dynamiques. Ils sont souvent utilisés pour leur capacité de classifieur et ils sont classiquement employés dans des problèmes d'approximation pour simuler des fonctions de transfert non-linéaires et multidimensionnelles et pour résoudre des problèmes d'inversion.

2.2 La Détection d'objets

La détection d'objet est modélisée comme un problème de classification dans lequel nous prenons des fenêtres de tailles fixes à partir d'une image d'entrée à tous les emplacements possibles et transmettons ces correctifs à un classificateur d'images.

Chaque fenêtre est transmise au classificateur qui prédit la classe de l'objet dans la fenêtre (ou l'arrière-plan s'il n'y en a pas).

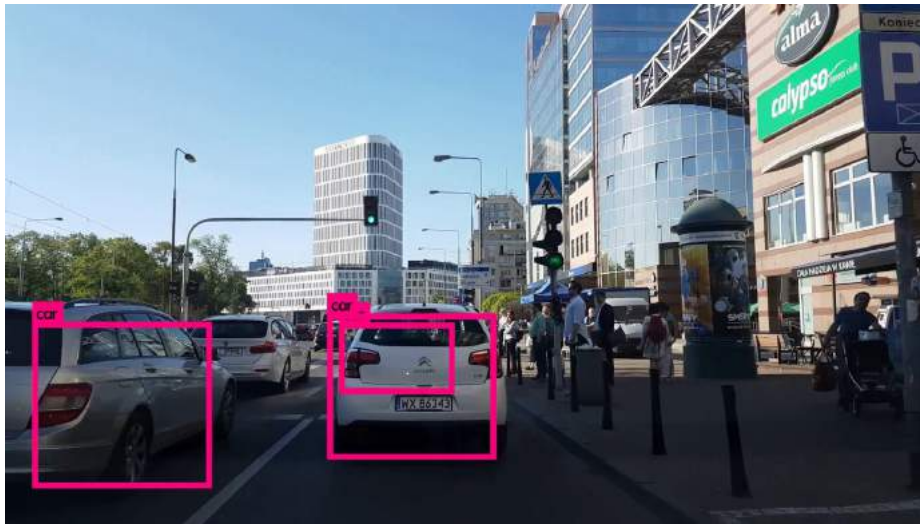


FIGURE 2.2.1 – La détection des voitures sur une route

L'objet peut être de différentes tailles. Pour résoudre ce problème, une pyramide d'images est créée en redimensionnant l'image. Nous redimensionnons l'image à plusieurs échelles et nous comptons sur le fait que la taille de la fenêtre que nous avons choisie contient complètement l'objet dans l'une de ces images redimensionnées. Le plus souvent, l'image est sous-échantillonnée (la taille est réduite) jusqu'à ce que certaines conditions, généralement une taille minimale, soit atteinte. Sur chacune de ces images, un détecteur de fenêtre de taille fixe est exécuté. Il est courant d'avoir jusqu'à 64 niveaux sur de telles pyramides. Désormais, toutes ces fenêtres sont envoyées à un classificateur pour détecter l'objet d'intérêt. Cela nous aidera à résoudre le problème de taille et d'emplacement.

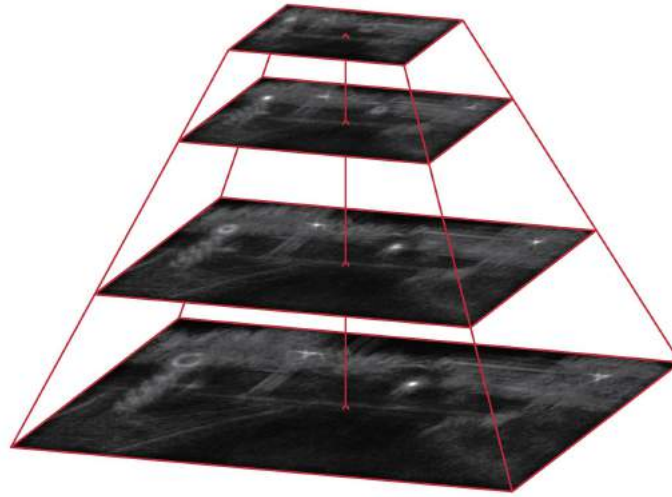


FIGURE 2.2.2 – Pyramide d'images

2.3 Travaux connexes

L'un des paradigmes les plus étudiés en matière de détection d'objet est le modèle déformable qui peut être formulé comme un modèle bayésien¹, il combine à la fois deux algorithmes : détection d'objet par le descripteur de l'Histogramme de Gradient Orienté HOG, proposé par Dallal & Triggs[4] et les structures picturales, s'inspirant ainsi du concept : chercher un objet articulé par la recherche de ses composantes .

Balayant l'image par l'approche : Fenêtre coulissante « Sliding Window », à la recherche d'un objet sur l'image en exploitant l'algorithme proposé par Dalal & Triggs[4], une fois l'objet détecté nous entamons la localisation ces composantes.

2.3.1 Histogramme de Gradient Orienté

Les descripteurs HoG sont introduits par Dallal et Triggs[4]. L'idée essentielle derrière l'histogramme de gradient orienté est que l'apparence locale et la forme d'un objet dans une image peuvent être décrites par la distribution d'intensité des gradients ou de direction des contours. La mise en œuvre de ces descripteurs peut être obtenue en divisant l'image en petites régions connectées, appelées cellules, et pour chaque cellule on calcule un histogramme des directions de gradient ou

1. Un modèle qui tire ses conclusions de la distribution postérieure, c'est-à-dire qui utilise une distribution antérieure et une probabilité reliées par le théorème de Bayes.

des orientations de contour pour les pixels dans la cellule. La combinaison de ces histogrammes représente alors le descripteur.

2.3.2 Structure Picturale

Le modèle basé structure picturale est une collection de pièce d'un objet disposé dans une configuration déformable. Chaque partie du modèle codifie des propriétés visuelles locales de l'objet, et la configuration déformable peut être comparé à un ressort en forme de connexions entre certaines paires de pièces. Le modèle d'apparence de chaque partie est donné par le biais d'une fonction, qui mesure la ressemblance d'un emplacement dans une image à la partie correspondante. Le meilleur résultat d'un tel modèle sur l'image donnée se trouve en minimisant la fonction d'énergie qui mesure à la fois le coût d'un résultat pour chaque partie et un coût de déformation pour chaque paire de parties reliées

Conclusion

Ce chapitre nous a permis d'avoir un aperçu global d'un système de reconnaissance de formes à savoir : les pré-traitements, l'extraction des primitives et la classification , et l'une des ces approches basées sur un modèle déformable. Dans le chapitre suivant nous abordons en détails les réseaux de neurones.

Chapitre 3 : L'apprentissage automatique et l'apprentissage en profondeur

Chapitre 3

L'apprentissage automatique et l'apprentissage en profondeur

Introduction

L'apprentissage automatique, et plus particulièrement sa version «L'apprentissage en profondeur» aide à la détection des obstacles pour les voitures autonomes. Au stade du traitement de l'information la voiture est capable de réceptionner des signaux lui permettant de reconstruire en temps réel l'environnement qui l'entoure et peut ainsi déterminer la nature de ces objets et donc adopter un comportement et des règles de conduite appropriés.

L'apprentissage automatique est un domaine de l'intelligence artificielle. Il s'inspire du fonctionnement du cerveau humain. Ce domaine étudie la façon dont des algorithmes peuvent apprendre en étudiant des exemples.

Ce type d'approche continue, jusqu'à ce jour, à attirer de plus en plus de chercheurs pour l'exploitation dans la proposition de nouvelles résolutions quotidiennes, industrielles, commerciales, ...etc.

L'objectif de ce projet d'étude est d'utiliser l'apprentissage profond dans la reconnaissance d'objets routiers. Pour cela, les bases théoriques de ce type de réseaux, l'apprentissage ainsi que son utilisation dans la vision par ordinateur sont présentées dans ce chapitre.

3.1 Définition de l'apprentissage automatique

L'apprentissage automatique (Machine learning) consiste à extraire des connaissances à partir de données. C'est un domaine de recherche situé à l'intersection de la statistique, de l'intelligence artificielle et de l'informatique. Il est également connu sous le nom d'analyse prédictive ou d'apprentissage statistique. Au lieu d'expli-

quer à un ordinateur avec précision comment résoudre un problème, l'apprentissage automatique permet de lui apprendre à résoudre un problème par lui-même.

3.1.1 Types d'apprentissage

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

- L'apprentissage non-supervisé

Le contexte non supervisé est celui où l'algorithme doit opérer à partir d'exemples non annotés ou la transformation du jeu de données et leur regroupement (clustering).

Les transformations non supervisées d'un jeu de données sont des algorithmes qui créent une nouvelle représentation des données qui pourraient être plus faciles à comprendre pour les humains ou d'autres algorithmes d'apprentissage automatique, par rapport à la représentation originale des données.

Une application courante des transformations non supervisées est la réduction de dimensionnalité, qui prend une représentation en haute dimension des données, comprenant de nombreuses fonctionnalités, et trouve un nouveau moyen de représenter ces données, qui résume les caractéristiques essentielles avec moins de fonctionnalités.

Les algorithmes de clustering, en revanche, partitionnent les données en groupes distincts d'éléments similaires.

- L'apprentissage supervisé

L'apprentissage automatique supervisé est l'un des types d'apprentissage automatique les plus utilisés et les plus performants. Il est utilisé chaque fois qu'on veut prédire un résultat donné à partir d'une entrée donnée et qu'on a des exemples de paires entrée / sortie. On construit un modèle d'apprentissage automatique à partir de ces paires entrée / sortie, qui constituent l'ensemble de l'apprentissage. L'objectif est de faire des prévisions précises pour de nouvelles données jamais vues auparavant.

L'apprentissage supervisé nécessite souvent un effort humain pour construire l'ensemble des données d'entraînement, mais il s'automatise et accélère ensuite une tâche par ailleurs laborieuse ou infaisable.

3.1.2 Généralisation, sur-apprentissage et régularisation

En apprentissage supervisé, si un modèle est capable de faire des prédictions précises sur des données invisibles, on dit qu'il est capable de généraliser à partir du jeu d'apprentissage au jeu de tests. Si les ensembles de données et de test ont suffisamment en commun, nous nous attendons à ce que le modèle soit également précis sur l'ensemble du test.

Sur-apprentissage intervient lorsque l'algorithme sur-apprend (overfit), autrement dit, lorsqu'il apprend à partir des données mais aussi à partir des patterns (schémas, structures) qui ne sont pas liés au problème, comme du bruit. Ainsi, le sur-apprentissage est caractérisé par une erreur de type variance très élevée. En d'autres termes, ce type de modèle conduit à de mauvaises performances car il manque de capacité de généralisation.

La technique la plus courante pour limiter le phénomène est la régularisation qui permet de réduire l'erreur de type variance. La régularisation permet d'imposer une contrainte pour favoriser les modèles simples au détriment des modèles complexes. Autrement dit, cela permet de réduire l'erreur de type variance et d'améliorer la généralisation de la solution.

3.2 Définition des réseaux de neurones

Les réseaux de neurones, un paradigme de programmation inspiré biologiquement qui permet à un ordinateur d'apprendre à partir de données d'observation. Dans l'approche classique de la programmation, nous disons à l'ordinateur ce qu'il doit faire, décomposant de gros problèmes en plusieurs petites tâches précisément définies que l'ordinateur peut facilement effectuer. En revanche, dans un réseau de neurones, nous ne disons pas à l'ordinateur comment résoudre notre problème. Au lieu de cela, il apprend à partir de données d'observation, en trouvant sa propre solution au problème à résoudre.

Apprendre automatiquement à partir de données semble prometteur. Cependant, jusqu'en 2006[10], nous ne savions pas comment former les réseaux de neurones pour dépasser les approches plus traditionnelles, à l'exception de quelques problèmes spécialisés. Ce qui a changé en 2006[10], c'est la découverte des techniques d'apprentissage dans des réseaux de neurones dits profonds. Ces techniques sont maintenant appelées apprentissage en profondeur (deep learning). Ils ont été développés plus avant et, aujourd'hui, les réseaux de neurones profonds et l'apprentissage en profondeur permettent d'atteindre des performances exceptionnelles pour de nombreux problèmes importants en matière de vision par ordinateur, de reconnaissance vocale et de traitement du langage naturel.

3.2.1 Perceptrons

Les perceptrons sont un type de neurone artificiel¹ qui ont été développés dans les années 1950 et 1960 par le scientifique Frank Rosenblatt[18], inspirés par les

1. Un réseau de neurones artificiels, ou réseau neuronal artificiel, est un système dont la conception est à l'origine schématiquement inspirée du fonctionnement des neurones biologiques, et qui par la suite s'est rapproché des méthodes statistiques.

travaux antérieurs de Warren McCulloch et Walter Pitts[13].

Un perceptron prend plusieurs entrées binaire x_1, x_2, x_3, \dots , et produit une seule sortie binaire :

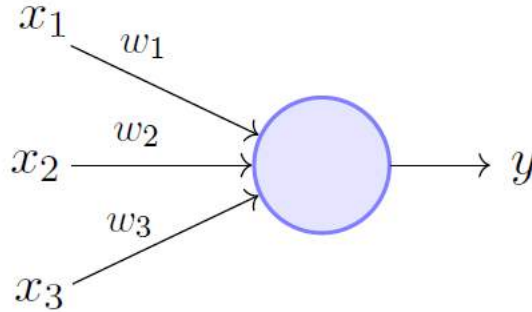


FIGURE 3.2.1 – Modèle d'un réseau de neurone, perceptron

Dans l'exemple présenté, le perceptron a trois entrées x_1, x_2, x_3 . En général, il pourrait avoir plus ou moins d'entrées. Rosenblatt a proposé une règle simple pour calculer la sortie. Il a introduit des poids w_1, w_2, w_3, \dots , des nombres réels exprimant l'importance des entrées respectives pour la sortie.

La sortie du neurone 0 ou 1, est déterminée par le fait que la somme pondérée $\sum_j w_j x_j$ est inférieure ou supérieure à une valeur de seuil.

Tout comme les poids, le seuil est un nombre réel qui est un paramètre du neurone. Pour le dire en termes algébriques plus précis :

$$y = \begin{cases} 0 & \text{si } \sum_j w_j x_j \preceq \text{seuil} \\ 1 & \text{si } \sum_j w_j x_j \succ \text{seuil} \end{cases}$$

Donc le perceptron est un appareil qui prend des décisions en soupesant des preuves.

Exemple : Supposons que le week-end approche et que vous sachiez qu'il y aura un festival dans votre ville. Donc vous voulez décider d'aller ou non. Vous pouvez prendre votre décision en pesant trois facteurs :

- Le temps est-il bon ?
- êtes-vous accompagner ?
- Le festival est-il proche du transport en commun ?

Nous pouvons représenter ces trois facteurs par les variables binaires correspondantes x_1, x_2, x_3 . Par exemple, nous aurions $x_1 = 1$ si le temps

est beau et $x_1 = 0$ si le temps est mauvais. De même, $x_2 = 1$ si accompagne et $x_2 = 0$ sinon. Et pareillement encore pour x_3 et le transport en commun.

Maintenant, supposons que vous détestez vraiment le mauvais temps, et il n'y a aucune chance d'aller au festival si le temps est mauvais. Vous pouvez utiliser des perceptrons pour modéliser ce type de prise de décision. Une façon de faire est de choisir un poids $w_1 = 6$ pour les conditions météorologiques, et $w_2 = 2$ et $w_3 = 2$ pour les autres conditions.

La valeur plus grande de w_1 indique que la météo compte beaucoup pour vous, bien plus que le fait que vos amis vous rejoignent ou la proximité des transports en commun. Enfin, supposons que vous choisissiez un seuil de 5 pour le perceptron. Avec ces choix, le perceptron implémente le modèle de décision souhaité, en indiquant 1 par beau temps et 0 par mauvais temps.

En variant les poids et les seuils, nous pouvons obtenir différents modèles de prise de décision.

Les produits pondérés en entrée sont additionnés, puis la somme est transmise à la fonction dite activation d'un nœud, afin de déterminer si et dans quelle mesure ce signal doit progresser plus avant dans le réseau pour affecter le résultat final. Dans ce cas, il s'agit d'une fonction simple avec un seul paramètre qui est le seuil.

La condition $\sum_j w_j x_j \geq \text{seuil}$ est lourde, et nous pouvons apporter deux modifications de notation pour le simplifier.

La première modification consiste à écrire $\sum_j w_j x_j$ en tant que produit scalaire, $w \cdot x = \sum_j w_j x_j$, où w et x sont des vecteurs dont les composantes sont respectivement les pondérations et les entrées.

Le deuxième changement consiste à déplacer le seuil de l'autre côté de l'inégalité et à le remplacer par ce que l'on appelle le biais du perceptron, $b \equiv -\text{seuil}$. En utilisant le biais² au lieu du seuil, la règle de perceptron peut être réécrite :

$$y = \begin{cases} 0 & \text{si } w \cdot x + b \preceq \text{seuil} \\ 1 & \text{si } w \cdot x + b \succ \text{seuil} \end{cases}$$

Pour un perceptron avec un très gros biais, il est extrêmement facile pour le perceptron de produire un 1. Mais si le biais est très négatif, il est alors difficile pour le perceptron de produire un 1. Évidemment, l'introduction du biais n'est qu'un léger changement de comment nous décrivons les perceptrons.

2. Le biais est une mesure de la facilité avec laquelle le perceptron produit un 1. Ou, pour le dire en termes plus biologiques, le biais est une mesure de la facilité avec laquelle le perceptron se déclenche.

Cette étape correspond à la pré-activation, c'est-à-dire l'étape qui précède l'activation³. Ensuite, la fonction d'activation intervient où le résultat y est interprété par une fonction d'activation non-linéaire $A(y)$ produisant en sortie un résultat z .

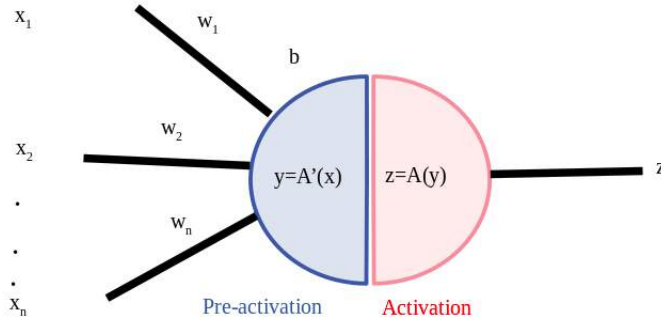


FIGURE 3.2.2 – Illustration des étapes pré-activation et activation sur un perceptrons

Parmi les fonction d'activation non-linéaire les plus utilisées dans la classification, on a la fonction d'unité de rectification linéaire (Rectified linear Units : ReLU) et softmax.

Le choix de la fonction diffère selon l'architecture du réseau. Dans les Réseaux de neurones à convolution 4.1 généralement la fonction ReLU (Rectified Linear Unit) est utilisée pour les couches convolutives 4.1.3 et la fonction Softmax pour la couche de sortie.

La fonction ReLU

La fonction ReLU est interprétée par la formule :

$$f(x) = \max(0, x)$$

Si l'entrée est négative, la sortie est 0 et sinon la sortie est de x .

Étant donné les résultats remarquables obtenus suite à l'emploi de cette fonction, elle est ces derniers temps la plus employée en pratique. Elle augmente considérablement la convergence du réseau et ne sature pas.

3. L'activation a pour but de transformer le signal de manière à obtenir une valeur de sortie sous forme simple probabilité à partir de transformations complexes entre les entrées.

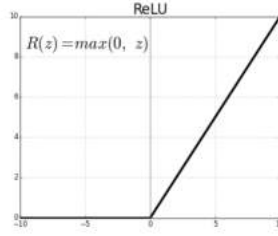


FIGURE 3.2.3 – La fonction ReLU

Fontion Softmax

Cette fonction est souvent utilisée dans la couche finale d'un classificateur basée sur un réseau neuronal. Elle attribue des probabilités décimales à chaque classe d'un problème à plusieurs classes. La somme des ces probabilités doit être égal à 1. Cette contrainte permet de faire converger l'apprentissage plus rapidement qu'il ne ferait autrement. Elle est définie comme suit :

$$f(x) = \frac{\exp a_i}{\sum_k \exp a_k}$$

3.2.2 L'architecture des réseaux de neurones

Dans un perceptron la couche intermédiaire est appelée couche cachée, car les neurones de cette couche ne sont ni des entrées ni des sorties.

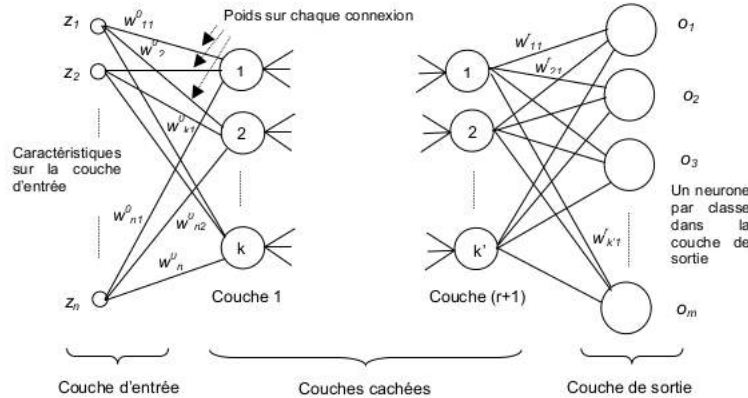


FIGURE 3.2.4 – Réseau de neurones multicouches

Le réseau ci-dessus 3.2.4 montre qu'on peut avoir plusieurs couches cachées : de tels réseaux sont parfois appelés perceptrons multicouches ou MLP.

3.2.3 L'apprentissage en profondeur

Supposons que nous voulions déterminer si une image montre un visage humain ou non en utilisant les pixels de l'image en tant qu'entrée dans un réseau de neurones, avec la sortie du réseau un seul neurone indiquant "Oui, c'est un visage" ou "Non, ce n'est pas un visage".

Supposons que nous le fassions, mais que nous n'utilisons pas d'algorithme d'apprentissage. Au lieu de cela, nous essayons de concevoir un réseau manuellement, en choisissant des poids et des biais appropriés. Une heuristique que nous pourrions utiliser consiste à décomposer le problème en sous-problèmes : l'image a-t-elle un œil en haut à gauche ? A-t-il un œil en haut à droite ? A-t-il un nez au milieu ? A-t-il une bouche en bas au milieu ? Y a-t-il des cheveux sur le dessus ? Etc.

Si les réponses à plusieurs de ces questions sont "oui", ou même simplement "probablement oui", alors nous concluons que l'image est susceptible d'être un visage. L'heuristique suggère que si nous pouvons résoudre les sous-problèmes à l'aide de réseaux de neurones, nous pourrions peut-être créer un réseau de neurones pour la détection des visages, en combinant les réseaux des sous-problèmes.

Voici une architecture possible, avec des rectangles indiquant les sous-réseaux :

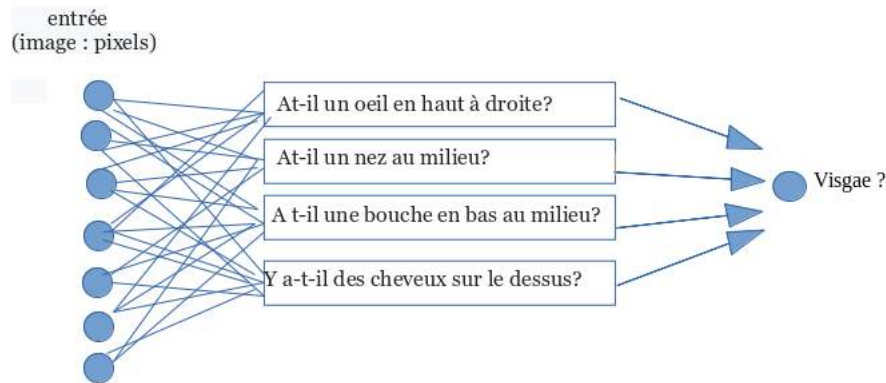


FIGURE 3.2.5 – Exemple d'architecture d'un réseau qui déterminer si une image montre un visage humain ou non

Il est également plausible que les sous-réseaux puissent être décomposés. Supposons que nous examinions la question : "Y a-t-il un œil en haut à gauche ?" Cela peut être décomposé en questions telles que : "Y a-t-il un sourcil ?" ; "Y a-t-il des cils ?" ; "Y a-t-il un iris ?" ; ..etc.

Le résultat final est un réseau qui décompose une question très compliquée (cette image montre-t-elle un visage ou non ?) en de très simples questions répondant au niveau des pixels simples. Cela se fait à travers une série de nombreuses

couches, les premières couches répondant à des questions très simples et spécifiques sur l'image d'entrée, et les couches ultérieures constituant une hiérarchie de concepts abstraits de plus en plus complexes.

Les réseaux avec ce type de structure à plusieurs couches sont appelés réseaux de neurones profonds.

Depuis 2006, un ensemble de techniques ont été développé pour permettre l'apprentissage dans des réseaux neuronaux profonds. Ces techniques reposent sur la descente et la rétro-propagation de gradient, mais introduisent également de nouvelles idées. Ces techniques ont permis de former des réseaux beaucoup plus profonds (et plus vastes).

Les réseaux d'apprentissage en profondeur effectuent l'extraction automatique des caractéristiques sans intervention humaine, contrairement à la plupart des algorithmes traditionnels d'apprentissage automatique. Ces réseaux se terminent par une couche de sortie : un classificateur logistique, ou softmax, qui attribue une probabilité à un résultat ou à un libellé particulier.

À partir de données brutes, sous la forme d'une image, le réseau peut décider, par exemple, que les données d'entrée ont 90% de chances de représenter une personne.

Rétro-propagation

L'algorithme de rétro propagation a été introduit dans les années 1970, mais son importance n'a pas été pleinement appréciée avant la publication d'un article célèbre de David Rumelhart, Geoffrey Hinton et Ronald Williams en 1986[?].

Aujourd'hui, l'algorithme de rétro-propagation est le fer de lance de l'apprentissage dans les réseaux de neurones. C'est le messenger qui dit au réseau si le réseau a commis une erreur lorsqu'il a fait une prédiction.

Les modèles de réseaux de neurones non formés sont comme des nouveau-nés : ils sont créés ignorants du monde et ce n'est que par l'exposition au monde qu'ils éprouvent que leur ignorance est lentement revue. Les algorithmes font l'expérience du monde à travers les données. Donc, en formant un réseau de neurones sur un jeu de données pertinent, nous cherchons à réduire son ignorance. Nous mesurons les progrès en surveillant l'erreur provoquée par le réseau.

La connaissance d'un réseau de neurones est capturée par ses pondérations, les paramètres qui modifient les données d'entrée lorsque son signal traverse le réseau de neurones vers la couche finale du réseau qui décidera de cette entrée. Ces décisions sont souvent fausses, car les paramètres transformant le signal en décision sont mal calibrés⁴.

4. Ils n'en ont pas encore assez appris.

Une instance de données qui traverse les paramètres d'un réseau vers la prévision à la fin est une propagation en avant. Une fois que cette prévision est faite, sa distance par rapport à la vérité peut être mesurée. Ainsi, les paramètres du réseau de neurones ont une relation avec l'erreur produite par le réseau, et lorsque les paramètres changent, l'erreur le fait également. Nous changeons les paramètres en utilisant un algorithme d'optimisation appelé descente de gradient, utile pour trouver le minimum d'une fonction. Nous cherchons à minimiser l'erreur.

La descente de gradient

La descente de gradient est l'algorithme d'optimisation le plus courant dans l'apprentissage automatique et l'apprentissage en profondeur. C'est un algorithme d'optimisation du premier ordre⁵.

Les valeurs numériques trouvées des gradients indiquent les taux d'influence qu'ont chaque poids et seuil sur les neurones de sortie. La mise à jour de ces paramètres nous permet d'accéder au minimum local (ou global) de la fonction étudiée.

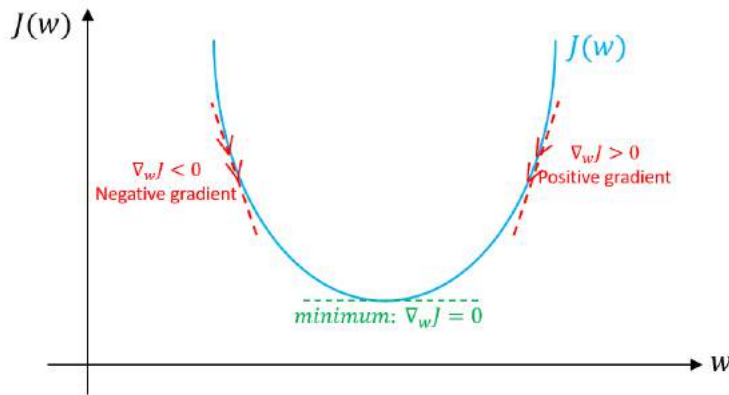


FIGURE 3.2.6 – illustration de la méthode de descente de gradient dans le cas où $j(w) = w^2$

Une illustration 3.2.6 de la façon dont l'algorithme de descente de gradient utilise la première dérivée de la fonction de perte pour suivre en descente son minimum.

5. Il ne prend en compte que la première dérivée lors de la mise à jour des paramètres.

Dans l'apprentissage en profondeur nous cherchons à minimiser la fonction de coût, appelé aussi fonction d'objectif ou encore de perte.

Fonction de coût : Il en existe plusieurs types pour des utilisations bien précises. En effet, selon le type de problème que l'on cherche à résoudre, on aura une sortie différente, et donc une fonction de coût bien précise concernant notre problème. Dans certains cas, on souhaite avoir un résultat en sortie compris entre $(0, 1)$, ou $(-1, 1)$, ou encore, un vecteur $[(0, 1), (0, 1) \dots]$ correspondant à plusieurs probabilités. Cette fonction représente la somme de l'ensemble des erreurs de l'ensemble du réseau. La somme des écarts au carré entre les valeurs observées (y_i) et les valeurs calculées par la droite d'équation $y = ax$. Cette fonction s'écrit de la façon suivante :

$$f(a) = \frac{1}{n} \sum_{i=0}^n (y_i - ax_i)^2$$

« a », le paramètre à estimer de la droite de régression $y = ax$

la valeur observée au point i

ax_i la valeur prédite par la droite au point i

n le nombre de points

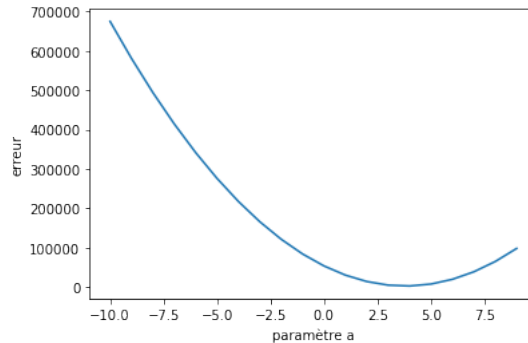


FIGURE 3.2.7 – Fonction objective : L'erreur en fonction du paramètre « a ». L'erreur minimum se situe aux alentours de 3

La méthode de descente en gradient consiste à prendre une valeur de « a » au hasard et la faire varier plus ou moins fortement par rapport à la pente de la fonction objective. Au lieu de tester toutes les valeurs de « a », on fait varier sa valeur avec des pas variables qui deviennent de plus en plus petits au fur et à mesure que l'on se rapproche du minimum.

A chaque itération, nous mettons à jour les paramètres dans le sens opposé du gradient de la fonction objectif.

$$f(a) = \frac{1}{n} \sum_{i=0}^n (y_i - ax_i)^2$$

Se développe en :

$$f(a) = \frac{1}{n} \sum_{i=0}^n (y_i^2 - 2y_i ax_i + (ax_i)^2)$$

Soit :

$$f(a) = \frac{1}{n} \sum_{i=0}^n (-2y_i ax_i + a^2 x_i^2)$$

En dérivant par rapport à «a» nous obtenons :

$$f(a) = \frac{-2}{n} \sum_{i=0}^n (x_i(y_i - ax_i))$$

La figure 3.2.8 ci-dessous montre les valeurs des pentes pour différentes valeurs de «a» allant de -10 à 10. Plus on se rapproche du minimum et plus la pente diminue. Elle est négative à gauche et positive à droite. Pour trouver la bonne valeur de «a», il suffit de faire varier «a» proportionnellement à ce gradient. Si la pente diminue, on augmente «a», si elle augmente on diminue «a».

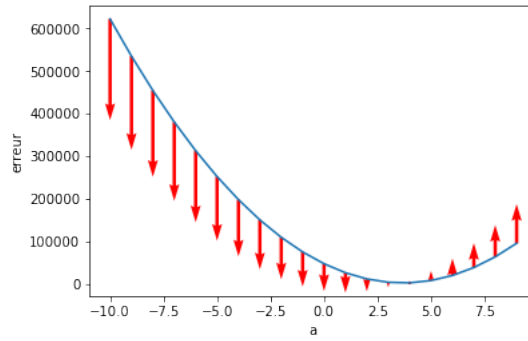


FIGURE 3.2.8 – Différentes pentes pour différentes valeurs de a .

La valeur «a» est appelée taux d'apprentissage, qui nous permet de faire varier la vitesse de correction. Celle-ci doit être bien calibrée, car si celle-ci se trouve trop faible la convergence prendra un temps infini, soit au contraire celle-ci se trouve trop grande, et la convergence oscillera sans trouver le minimum. Nous utilisons un taux adaptatif en fonction de l'apprentissage. En effet, nous prenons un taux élevé au début pour améliorer la convergence, puis on la réduit progressivement au fil des itérations pour améliorer la précision.

Variantes de descente de gradient

Il existe trois variantes de cette méthode. En fonction de la quantité de données, nous ferons un compromis entre la précision de la mise à jour des paramètres et le temps d'exécution de cette mise à jour.

- Batch gradient descent : C'est la descente de gradient classique, on calcule le gradient de la fonction coût pour l'ensemble d'apprentissage. Comme on doit calculer le gradient pour l'ensemble des données pour exécuter juste une seule mise à jour, cette méthode peut être très longue et irréalisable si les données ne peuvent pas être stockées en mémoire. Avec cette méthode on est sûr de converger vers l'optimum global si la fonction coût est convexe et vers un optimum local si elle est non convexe.
- Descente de gradient stochastique SGD (la descente de gradient stochastique) : met à jour les paramètres pour chaque exemple de l'ensemble des données et labels. Cette méthode est plus rapide mais les mises à jours des paramètres trop fréquentes causent à la fonction objectif des oscillations. Ces oscillations permettent d'une part d'atterrir dans des minimums locaux potentiellement meilleurs mais d'autre part rendent la convergence plus difficile. Cependant il a été montré qu'en baissant le taux d'apprentissage, SGD montre la même convergence que la batch gradient descent.
- Mini-batch gradient descent : Cette méthode prend le meilleur des deux méthodes précédentes et met à jour les paramètres pour chaque mini groupes de n exemples. Cette méthode réduit la variance des mises à jour des paramètres ce qui conduit à une convergence plus stable. Généralement les mini groupes contiennent de 50 à 256 exemples. C'est une méthode de choix pour entraîner un réseau de neurones.

Algorithmes d'optimisation de la descente de gradient

Dans ce qui va suivre, on exposera quelques algorithmes largement utilisés par la communauté du Deep Learning pour résoudre les challenges cités précédemment.

- **Momentum** : SGD a du mal dans les régions où la surface est beaucoup plus courbée en une dimension plutôt que dans une autre et ils sont communs autour des minimums locaux. Dans ce cas là, SGD oscille à travers les pentes de ces régions et n'achève que des progrès hésitant vers des optimums locaux. Une des méthodes qui peut aider le réseau à se sortir de ces pièges est d'utiliser le coefficient du momentum :

$$v_t = \eta v_{t-1} + a \nabla j(a)$$

$$a = a - v_t$$

où $\eta \in [0, 1]$ représente le coefficient de momentum.

Lors de l'utilisation du momentum, on pousse une balle en bas d'une colline. La balle accumule du momentum alors qu'elle roule vers le bas devenant de plus en plus rapide. La même chose se passe lors de la mise à jour des paramètres. Le momentum augmente pour les dimensions dont le gradient pointe vers la même direction et réduit les mises à jours dont le gradient change de direction. Comme résultat, on a une convergence plus rapide et on réduit les oscillations.

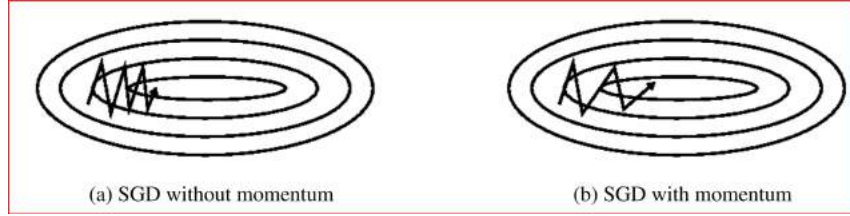


FIGURE 3.2.9 – Accélération de SGD par la méthode du momentum et réduction des oscillations

- **RMSprop** : Il s'agit d'une méthode de taux d'apprentissage adaptative non publiée, proposée par Geoff Hinton dans Lecture 6e de sa Coursera Class. Au lieu d'accumuler tous les carrés des gradients précédents, on restreint la fenêtre des gradients accumulés à une taille fixée w . Au lieu de stocker les w carrés des gradients précédents, on applique une moyenne mobile exponentielle des carrés des gradients précédents. La moyenne courante $E[g^2]_t$ à l'étape t dépend uniquement de la moyenne précédente et de gradient courant.

$$E[g^2]_t = \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2$$

$$a_{t+1} = \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t$$

Hinton suggère que γ soit fixé à 0.9, tandis qu'une bonne valeur par défaut pour le taux d'apprentissage η est de 0.001.

- **Adam optimizer** : Adaptive Moments (ADAM) optimizer est une autre méthode qui calcule un taux d'apprentissage adaptatif pour chaque paramètre. En plus de stocker une moyenne décroissante exponentielle des précédents carrés des gradients v_t comme RMSprop, ADAM conserve aussi une moyenne décroissante exponentielle des précédents gradients m_t .

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)g_t$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$$

m_t et v_t sont respectivement des estimations d'ordre 1 (la moyenne) et d'ordre 2 (la variance) de gradient. Comme m_t et v_t sont initialisés comme des vecteurs de 0, les auteurs ont observé qu'ils sont biaisés vers zéro surtout durant les premières étapes, et surtout lorsque le coefficient de décroissance est petit (β_1, β_2 proche de 1) alors ils ont calculé une correction des biais pour les estimations du premier et du deuxième ordre.

Les auteurs proposent des valeurs par défaut de 0.9 pour β_1 , 0.999 pour β_2 et 10^{-8} pour ε .

Conclusion

Ce chapitre nous a permis d'avoir un aperçu général sur les réseaux de neurones. Notre étude s'est intéressée principalement aux réseaux de neurones de type MLP. Nous avons décrit les étapes de l'apprentissage du réseau et les différents critères d'arrêt de cet algorithme. L'objectif de notre projet est d'améliorer la performance du système de reconnaissance d'objets.

La méthode proposée pour la détection d'objets exploitant l'apprentissage profond en utilisant les réseaux de neurones à convolution à base de région est abordée dans le chapitre suivant.

Chapitre 4 : La méthode utilisée pour la détection d'objet

Chapitre 4

La méthode utilisée pour la détection d'objet

Introduction

Aujourd'hui, les Réseaux de neurones à convolution sont au centre de l'attention, vu les résultats impressionnants qu'ils produisent sur différentes tâches, en particulier en vision par ordinateur.

Cependant, ils ne sont pas parfaits et leur étude au sein de la communauté est très active, notamment sur les problématiques de complexité calculatoire, d'optimisation pendant l'apprentissage et de choix des hyper-paramètres. De plus, pour être appris efficacement, les Réseaux de neurones à convolution ont besoin d'être entraînés sur un grand nombre de données pour éviter le sur-apprentissage. Ces données ne sont pas toujours disponibles pour résoudre un problème spécifique, et pour remédier à toutes ces contraintes les recherches ont abouti à d'autres réseaux plus performants comme le R-CNN, FAST et FASTER R-CNN.

Dans ce chapitre, nous allons introduire les différents modèles des Réseaux de neurones à convolution utilisés, plus particulièrement les réseaux neuronaux convolutifs profonds à base de région (R-CNN /Faster R-CNN) et ainsi leur différentes architectures.

4.1 Réseaux de neurones à convolution (CNN)

L'objectif du domaine de vision par ordinateur est de permettre aux machines de voir le monde comme les humains le voient, le perçoivent de la même manière et utilisent même les connaissances pour une multitude de tâches telles que la reconnaissance d'images et de vidéos, l'analyse et la classification d'images, etc[10].

Les progrès de la vision par ordinateur avec l'apprentissage en profondeur ont été construits et perfectionnés avec le temps, principalement sur un algorithme particulier - un réseau de neurones convolutionnels.

Un réseau à convolution est un type de réseau neuronal très utilisé pour la classification d'image ou la reconnaissance visuelle. Le but principal d'un réseau à convolution est d'extraire des caractéristiques d'une image donnée.

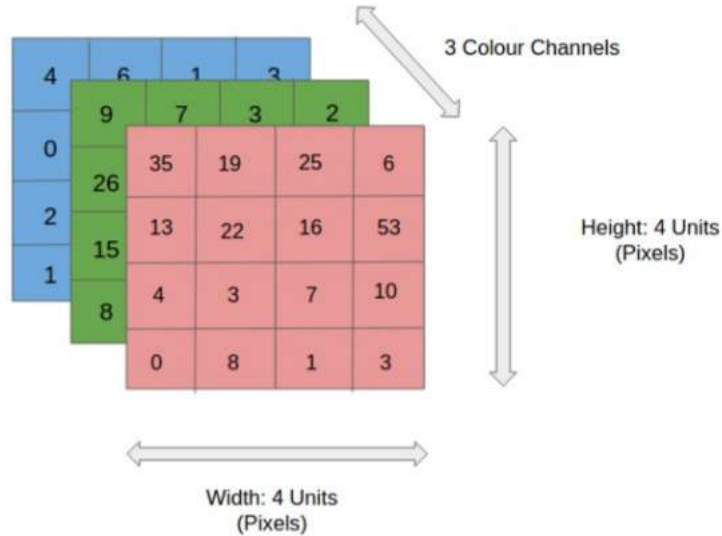


FIGURE 4.1.1 – Présentation d'une images RVB

Dans la figure 4.1.1, nous avons une image RVB, qui a été séparée par ses trois plans de couleur. Vous pouvez imaginer comment les choses nécessitent beaucoup de calcul une fois que les images atteindraient les dimensions, par exemple 8K (7680×4320). Le rôle de ConvNet est de réduire les images à une forme plus facile à traiter, sans perdre les caractéristiques indispensables à une bonne prédiction.

4.1.1 Le filtre

Le CNN compare les images fragment par fragment. Les fragments qu'il recherche sont appelés les caractéristiques. Un filtre sert à faire ressortir certaines caractéristiques d'une image donnée (bas niveaux : couleur, contour, luminosité, netteté, etc...). Ce filtre va être déplacé par pas successifs sur l'ensemble de l'image[19].

La couche de convolution n'a pas besoin d'être limitée à une seule couche convolutionnelle. Conventionnellement, la première couche de convolution est chargée de capturer les caractéristiques de bas niveau. Avec l'ajout de couches, l'architecture

s'adapte également aux fonctionnalités de haut niveau, ce qui nous donne un réseau offrant une compréhension complète des images du jeu de données.

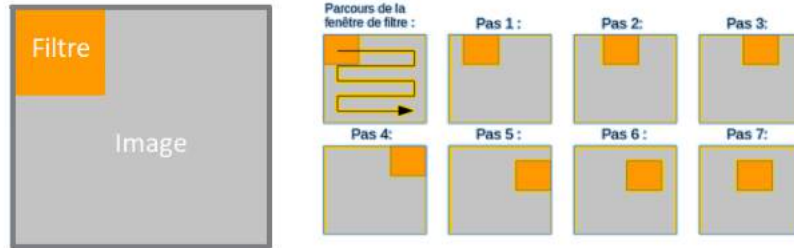


FIGURE 4.1.2 – Illustration du déplacement du filtre au long de l'image

Le filtre se déplace vers la droite avec une certaine valeur "Stride" jusqu'à ce qu'il analyse toute la largeur. Pour continuer, il saute au début (à gauche) de l'image avec la même valeur de cadence et répète le processus jusqu'à ce que toute l'image soit parcourue.

Dans le cas d'images avec plusieurs canaux (par exemple, RVB), le noyau a la même profondeur que celle de l'image d'entrée.

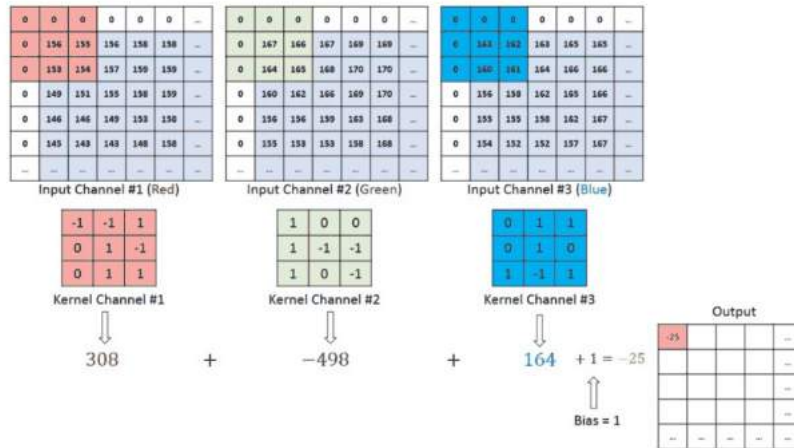


FIGURE 4.1.3 – La multiplication de matrice est effectuée entre les piles Kernel_n et longueur_n , et tous les résultats sont additionnés avec le biais pour nous donner une sortie d'entité conjuguée de canal à profondeur réduite.

Le nombre de filtres déterminera le nombre de caractéristiques détectées. Ce nombre est appelé la profondeur. C'est à dire que si 10 filtres seront appliqués à une image donnée, la valeur de sa profondeur sera de 10.

4.1.2 Le Pooling

Le Pooling est un procédé important dans un réseau à convolution. En extrayant les valeurs importantes des pixels, il permet de réduire une image tout en conservant les caractéristiques pertinentes. Il existe deux types de pooling : pooling maximum et pooling moyen.

Max Pooling renvoie la valeur maximale de la partie de l'image couverte par le noyau. D'autre part, le pooling moyen renvoie la moyenne de toutes les valeurs de la partie de l'image couverte par le noyau[19].

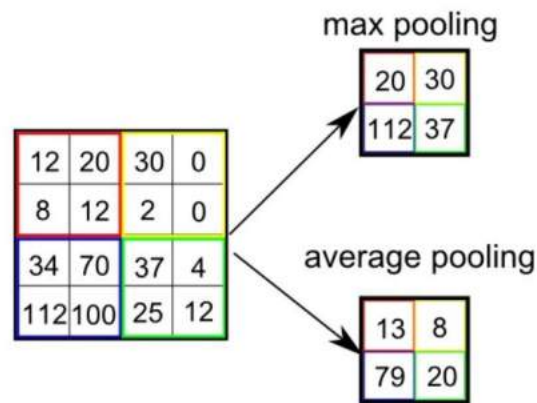


FIGURE 4.1.4 – Exemple de calcul de max et average pooling

Max Pooling, joue également le rôle de supprimeur de bruit. Il élimine complètement les activations bruyantes et effectue également le débruitage avec réduction de la dimensionnalité. D'autre part, le pooling moyen effectue simplement une réduction de la dimensionnalité en tant que mécanisme de suppression de bruit. Par conséquent, nous pouvons dire que Max Pooling fonctionne beaucoup mieux que le Pooling moyen. Pour permettre au CNN de rester en bonne santé (mathématiquement parlant) en empêchant les valeurs apprises de rester coincé autour de 0 ou d'exploser vers l'infinie, alors on utilise une couche ReLU.

4.1.3 La convolution

La couche de convolution et la couche de regroupement forment ensemble la i -ème couche d'un réseau de neurones de convolution. En fonction de la complexité des images, le nombre de couches de ce type peut être augmenté afin de capturer encore plus les détails de bas niveau, mais au prix d'une plus grande puissance de calcul[19].

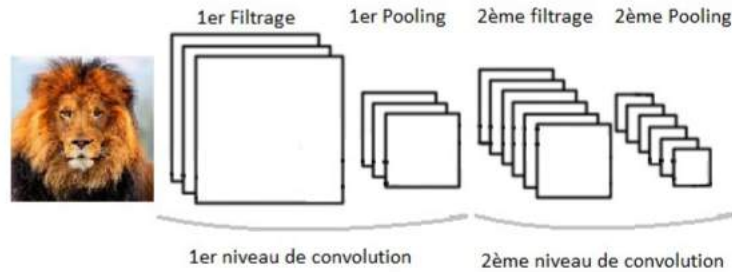


FIGURE 4.1.5 – Illustration de la couche de convolution

Dans cet exemple, trois filtres ont été créés donnant trois nouvelles "images" qui représentent certaines caractéristiques particulières de l'image de départ.. Ensuite, ces images sont réutilisées en entrée de l'étape suivante (filtrage + pooling), on a deux niveaux de convolution.

On peut observer qu'une que fois les deux niveaux de convolution ont opéré, six nouvelles images sont obtenues à partir de la première. Après avoir suivi le processus ci-dessus, nous avons permis au modèle de comprendre les fonctionnalités.

Nous allons ensuite aplatir le résultat final et le transmettre à un réseau de neurones régulier à des fins de classification.

4.1.4 La classification : Couche entièrement connectée (couche FC)

La classification : Couche entièrement connectée (couche FC). Pour injecter les patterns issus du réseau à convolution dans le réseau neuronal, on passe par une étape dite de "Flattening" (ou aplatissement). Cette opération consiste à mettre à plat toutes les données dans un seul vecteur. Ce vecteur permettra la création d'une première couche de neurones entièrement connectée. C'est-à-dire que chacune des valeurs de ce vecteur sera connectée aux neurones de la première couche du réseau permettant la classification de l'image[19].

Donc la sortie aplatie est acheminée vers un réseau de neurones à réaction et une rétro-propagation est appliquée à chaque itération de la formation. Sur une série d'époques, le modèle est capable de distinguer les caractéristiques dominantes de certaines caractéristiques de bas niveau dans les images et de les classer à l'aide de la technique de classification Softmax.

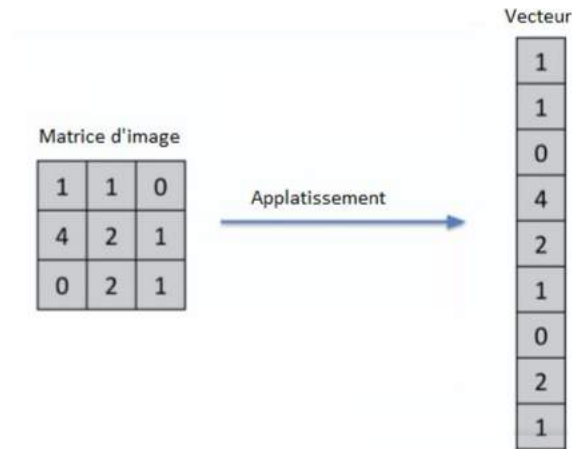


FIGURE 4.1.6 – Couche entièrement connectée (couche FC)

Il existe différentes architectures de CNN disponibles qui ont joué un rôle clé dans la construction d'algorithmes qui alimentent et alimenteront l'IA dans son ensemble dans un avenir proche (LeNet, AlexNet, VGG_n, GoogLeNet (inception), ResNet, ZFNet)

4.2 R-CNN

L'idée qui a été développée à travers les travaux de Girshick et al [17], est de mixer la méthode de segmentation et les réseaux de neurones à convolution. En effet, avec cette méthode, nous pouvons classifier chacune des zones présente dans notre image.

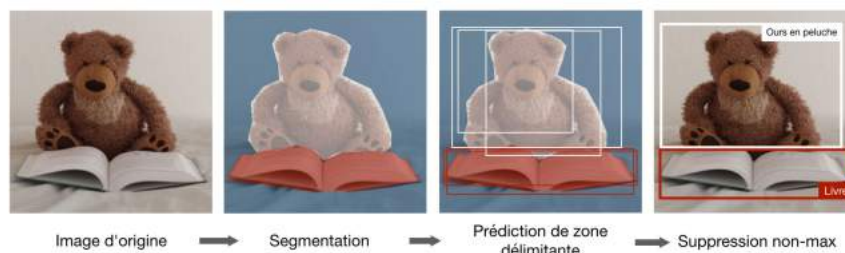


FIGURE 4.2.1 – Démonstration du Convolutional Neural Networks (R-CNN)

Ce système nommé R-CNN (Regions and Convolutional Neural Networks) pour la recherche des régions d'intérêts et les réseaux de neurones à convolution. Ce

réseau va prendre en entrée une image et, nous allons obtenir en sortie un ensemble de boîte qui englobe un objet. Bien entendu, chacune des boîtes est labellisée.

Ainsi, ce réseau, va dans un premier temps chercher les zones d'intérêts à l'aide de l'algorithme de segmentation de l'image « la recherche sélective¹ ». Puis, il va générer, à partir de ces zones, plusieurs boîtes qui englobent les objets. Pour chacune d'entre elles, il va les classer à l'aide d'un réseau de neurones à convolution. Enfin, il va sélectionner la boîte qui englobe le mieux l'objet en fonction des résultats de la prédiction.

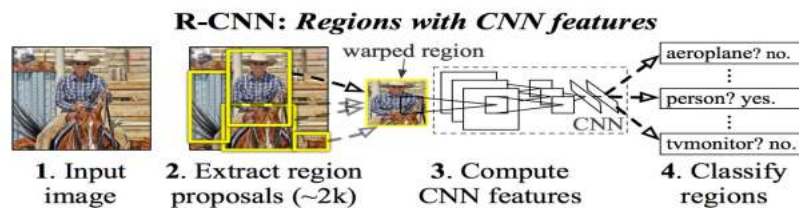


FIGURE 4.2.2 – Architecture du Convolutional Neural Networks (R-CNN)

4.2.1 Fast R-CNN

L'un des problèmes conséquent du R-CNN est que nous réalisons énormément de convolution. L'idée qui a été énoncée par Ross Girshick[17] est de réaliser une seule convolution. Ainsi, nous allons pouvoir économiser énormément de temps.

Afin de pouvoir réaliser ce système, tout d'abord, nous allons avoir en entrée de notre réseau une image ainsi que différentes zones d'intérêts. Ensuite, nous allons extraire les caractéristiques présentes sur l'image. Pour cela, nous passons notre image dans différentes couches de convolution et de pooling. Une fois les caractéristiques extraites, nous allons avoir une couche appelée RoI pooling layer (Regions of Interest). Cette couche va pour chacun des objets que nous avons détecté, extraire un vecteur de caractéristiques correspondant à l'objet. Pour chacun de ces vecteurs, nous allons les envoyer dans une série de couches entièrement connectées. Par la suite, nous allons prendre la sortie de cette couche que nous allons propager dans deux chemins différents composés :

1. La recherche sélective (selective search) : Utilise le regroupement ascendant des régions d'image pour générer une hiérarchie de régions petites à grandes. Prend en considération plusieurs critères de regroupement et détecte les différences de couleur, texture, luminosité.

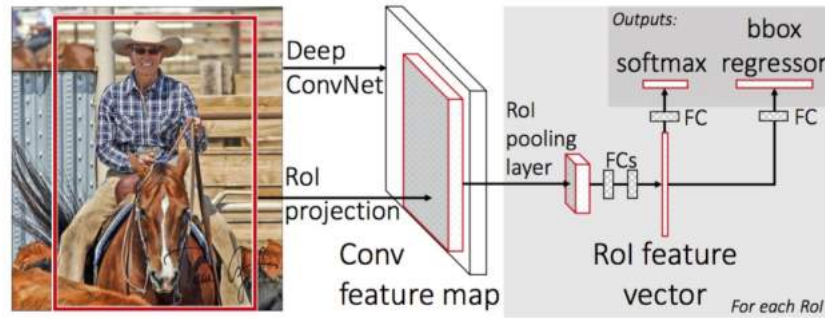


FIGURE 4.2.3 – Architecture de Fast R-CNN

RoI Pooling

Ce qu'il faut savoir, c'est que notre couche de RoI aura une taille prédéfinie (7x7). Nous sommes obligé de faire cela, car nous ne pouvons pas avoir en sortie un vecteur qui a une taille qui varie. Pour ce faire, nous allons adapter notre pooling. Donc nous allons dans un premier temps sélectionner sur la couche de convolution précédente la taille de notre région d'intérêts, nous allons la sélectionner de manière proportionnelle. En effet, nous connaissons la taille de notre image d'origine, la taille de notre boîte ainsi que la taille de nos données de sortie. Ainsi, il nous suffit de réaliser un produit. Enfin, afin d'adapter notre max pooling, nous allons modifier les paramètres et les adapter en fonction de notre région d'intérêts que nous allons obtenir.

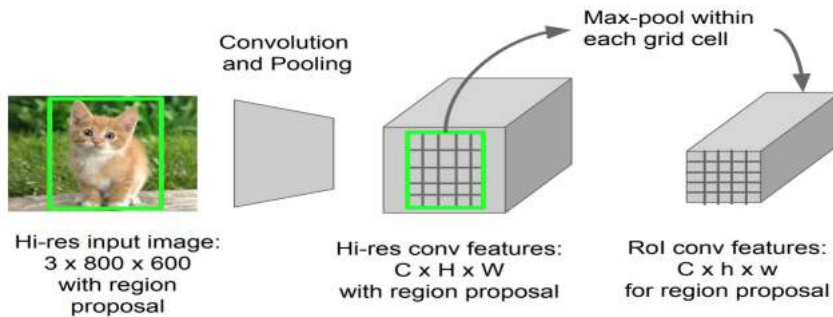


FIGURE 4.2.4 – RoI Max Pooling

4.2.2 Faster R-CNN :

Cependant, avec le FAST R-CNN, nous ne pouvons toujours pas réaliser une analyse en temps réel. Et c'est un réel problème même si, avec ce système, nous avons eu une avancée considérable en terme de temps. Ainsi, un nouveau système nommé FASTER R-CNN a vu le jour.

L'objectif premier du FASTER R-CNN est d'enlever la partie liée à la segmentation de l'image que nous réalisons en premier.

L'idée qui a été développée par Ross Girshick[17] est de supprimer le système de segmentation de l'image et d'intégrer dans notre réseau de neurones, un système permettant de suggérer des zones d'intérêts, dite le Region Proposal Network ou RPN. Comme visible sur la 4.2.5

Autrement dit, Faster R-CNN repose sur une détection entièrement faite avec des réseaux de neurones de convolution :

1. un premier réseau de neurones de convolution prend en entrée une image de taille quelconque et donne en sortie des régions dans lesquelles pourraient se trouver les objets à détecter.
2. le second réseau prend en entrée les régions proposées par le premier réseau et recherche si elles contiennent l'objet à détecter.

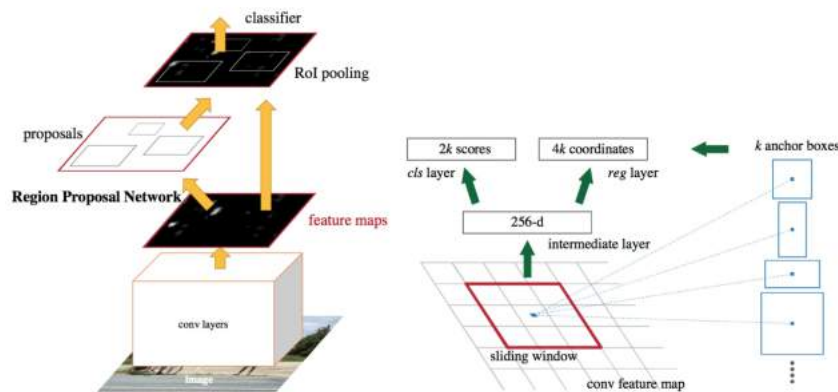


FIGURE 4.2.5 – Architecture de Faster R-CNN

Fonctionnement du RPN

RPN a un classificateur et un régresseur. Ils ont introduit le concept d'ancres (anchors). L'ancrage est le point central de la fenêtre coulissante. Pour ZF Model, une extension d'AlexNet, les dimensions sont 256-d et pour VGG-16, 512-d.

Il génère un ensemble de régions proposées à partir de l'image d'entrée puis un deuxième composant (comprenant un classifieur et un opérateur de régression des patches) qui vérifie si chaque région proposée correspond à un objet recherché ou non.

4.3 Détecteur SSD

En utilisant Single Shot Multi Box Detector (SSD) il suffit d'une seule prise de vue pour détecter plusieurs objets dans l'image, tandis que les approches basées sur un réseau de propositions régionales (RPN) nécessitent deux prises de vues. Ainsi, le SSD est beaucoup plus rapide que les approches à deux coups basées sur la RPN. La détection d'objet SSD se compose de deux parties :

1. Extraction des cartes de caractéristiques : SSD utilise VGG16 pour l'extraction des caractéristiques.
2. Application des filtres de convolution pour la détection des objets.

4.4 Les architectures neuronales classiques

Les bons ConvNets sont des bêtes avec des millions de paramètres et de nombreuses couches cachées. En fait, une mauvaise règle empirique est la suivante : «plus le nombre de couches cachées est élevé, meilleur est le réseau». AlexNet, VGG, Inception, ResNet sont quelques-uns des réseaux les plus populaires.

ZF-NET

ZF Net est une architecture composée de cinq couches de convolution et de deux couches de fully connected. Elle utilise des filtres de taille 7×7 et une valeur de foulée réduite. Le but de cette modification est qu'une taille de filtre plus petite dans le premier calque de convolution permette de conserver de nombreuses informations sur les pixels d'origine dans le volume d'entrée.

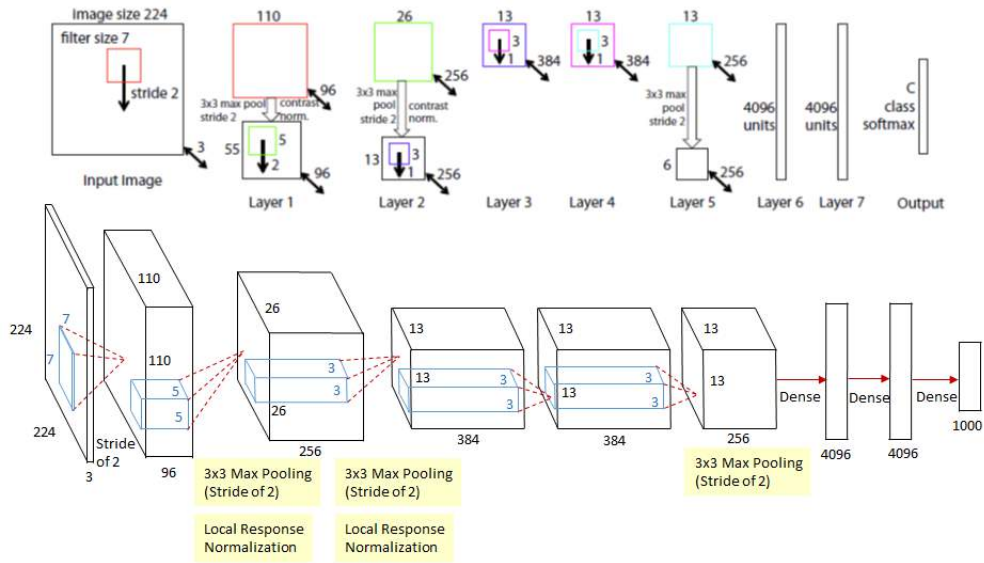


FIGURE 4.4.1 – Illustration de l'architecture ZF-NET

VGG16

Cette architecture utilise des séquences de convolution. De plus, les filtres convolutifs sont de plus petite taille (noyau de taille 3×3)

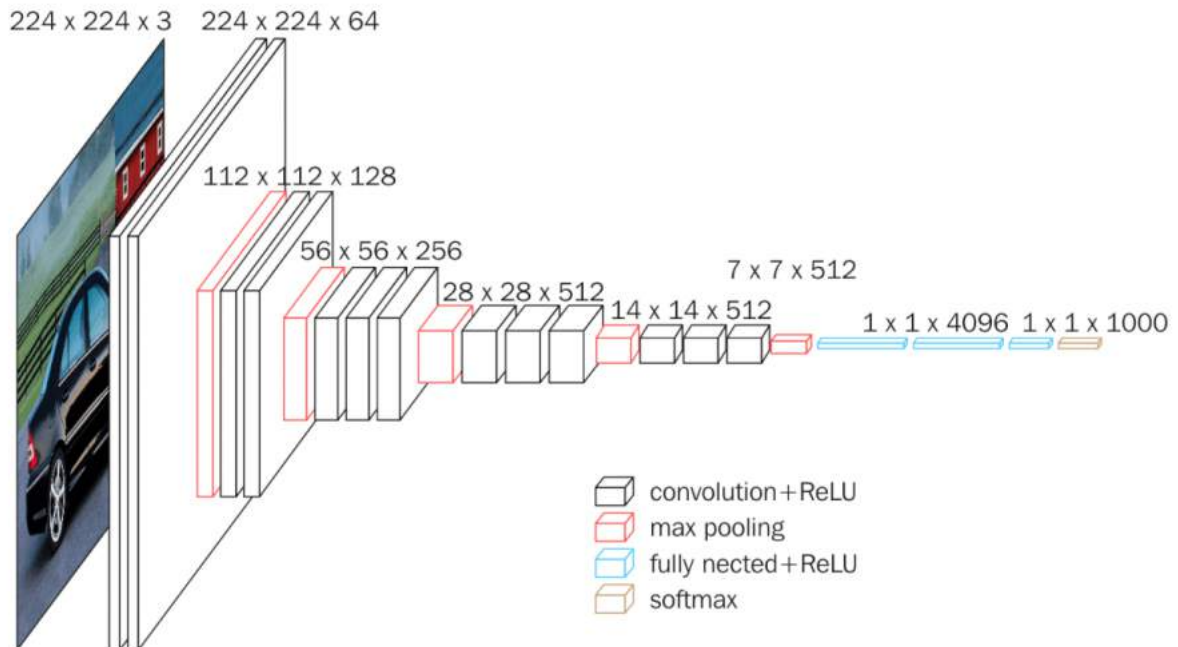


FIGURE 4.4.2 – Illustration de l'architecture VGG16

ResNet

L'architecture du réseau résiduel (en anglais Residual Network), aussi appelé ResNet, utilise des blocs résiduels avec un nombre élevé de couches et a pour but de réduire l'erreur de training.

Il s'agit d'une architecture Ultra-profond composée de 152 couches qui utilisent principalement les filtres 3×3 .

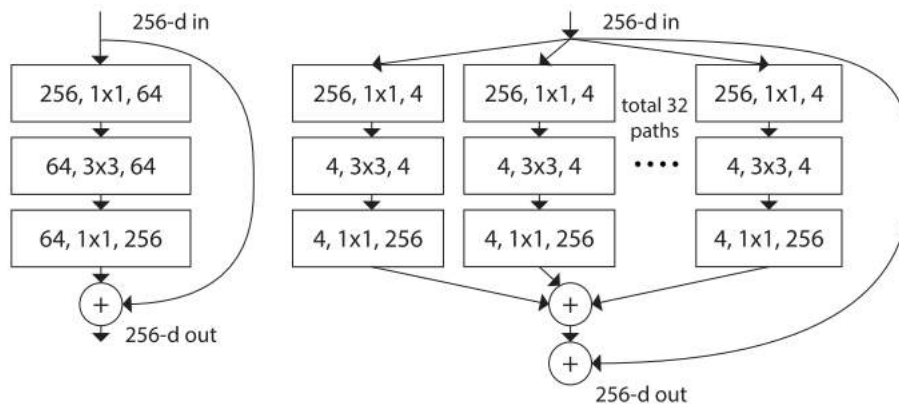


FIGURE 4.4.3 – Illustration de l'architecture ResNet

4.4.1 Inception Network

Cette architecture utilise des modules d'inception et a pour but de tester toutes sorte de configuration de convolution pour améliorer sa performance en diversifiant ses attributs. En outre, il utilise des convolutions de différentes tailles pour capturer des détails à différentes échelles (5X5, 3X3, 1X1), et limiter sa complexité de calcul.

Pour l'Inception v2 : Factorise la convolution 5x5 en deux opérations de convolution 3x3 pour améliorer la vitesse de calcul. Bien que cela puisse sembler contre-intuitif, une convolution 5x5 coûte 2,78 fois plus cher qu'une convolution 3x3. Donc, empiler deux convolutions 3x3 en fait améliore les performances. Ceci est illustré dans l'image ci-dessous.

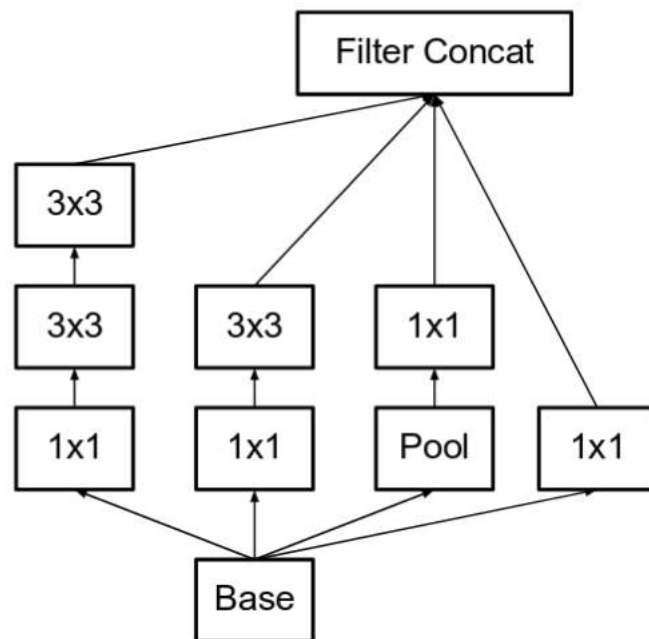


FIGURE 4.4.4 – La convolution 5x5 la plus à gauche de l'ancien module de création est maintenant représentée par deux convolutions 3x3.

Conclusion

Ce chapitre énumère les différents modèles de CNN qui ont été développés afin d'améliorer les performances et réduire le temps de calcul des réseaux de neurones convolutifs. Ainsi nous avons défini les différentes architectures que peuvent être utilisées par les CNN.

Chapitre 5 : Développement, Résultats et Expérimentations

Chapitre 5

Développement, Résultats et Expérimentations

Introduction

Après avoir décrit dans le chapitre précédent les différentes méthodes proposées pour construire un système de détection et de classification d'objets, nous allons dans ce chapitre présenter les différents résultats obtenus pour chaque étape du système de Détection et de classification développé. Nous commençons par la description de nos données et leurs formats, suivie de la présentation des outils, matériels et logiciels utilisés au cours de la réalisation de notre stage. Nous terminons par l'analyse des résultats du modèle d'apprentissage automatique développé.

5.1 Environnements de travail

L'environnement utilisé pour le développement de notre système est comme suit :

5.1.1 Environnement matériel

- Operating System : Ubuntu 18.04 64-bit
- Processor : Intel® Core™ i7-6500U CPU @ 2.50GHz × 4
- Graphics : Intel® HD Graphics 520 (Skylake GT2)
- Memory : 11.6 GiB

5.1.2 Environnement logiciel

Afin de réaliser une application portable et efficace à la fois, les programmes sont implémentés en Python 3. Python est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet. Pour l'apprentissage automatique, nous avons utilisé la bibliothèque Tensorflow [8]. TensorFlow est une bibliothèque open source de Machine Learning, créée par Google, permettant de développer et d'exécuter des applications de Machine Learning et de Deep Learning. Il s'agit d'une boîte à outils permettant de résoudre des problèmes mathématiques extrêmement complexes avec aisance. Elle permet aux chercheurs de développer des architectures d'apprentissage expérimentales et de les transformer en logiciels TensorFlow[8].

TensorFlow [8] permet aux développeurs de créer des graphiques de dataflow (dataflow graphs), à savoir des structures qui décrivent la façon dont les données se déplacent sur un graphique ou une série de noeuds de traitement. Chaque noeud du graphique représente une opération mathématique, et chaque connexion entre ces noeuds est une flèche de données multidimensionnelle : un tensor. Cette bibliothèque permet d'exécuter un code de bas niveau écrit en C++ qui peut effectuer des calculs massivement en parallèles, en utilisant toutes les unités de traitement disponibles et en bénéficiant des instructions de vectorisation matérielles existant dans ces unités.

TensorFlow [8] offre aussi un outil intégré appelé TensorBoard qui fournit la visualisation et l'outillage nécessaires à l'expérimentation par apprentissage automatique :

- Suivi et visualisation des métriques telles que la perte et la précision
- Visualiser le graphe de modèle (ops et layers)
- Affichage des histogrammes de poids, de biais ou d'autres tenseurs à mesure qu'ils évoluent dans le temps
- Affichage d'images, de texte et de données audio
- Profilage des programmes TensorFlow
- Et beaucoup plus

Nous avons également utilisé la bibliothèque NumPy[9]. NumPy [9]est une extension du langage de programmation Python, destinée à manipuler des matrices ou tableaux multidimensionnels ainsi que des fonctions mathématiques opérant sur ces tableaux. Plus précisément, cette bibliothèque logicielle libre et open source fournit de multiples fonctions permettant de créer directement un tableau depuis un fichier ou au contraire de sauvegarder un tableau dans un fichier, et manipuler des vecteurs, matrices et polynômes



FIGURE 5.1.1 – Les logos de Python, TensorFlow, et NumPy

Pour réaliser la phase d'apprentissage, les objets souhaités nécessitent d'être étiquetés dans chaque image. Pour cela, nous avons opté pour l'outil LabelImg qui est un outil d'annotation d'image graphique et des boîtes de sélection d'objet d'étiquette dans les images. Il est écrit en Python et utilise Qt¹ pour son interface graphique.

Le fonctionnement de cet outil est comme suit :

- création d'une fenêtre de sélection sur les objets souhaités à étiqueter dans l'image.
- Etiquetage des objets.

Ce processus doit être répété pour l'ensemble des images de la base de données.



FIGURE 5.1.2 – Capture de l'outil LabelImg

1. Qt : une API orientée objet et développée en C++, conjointement par The Qt Company et Qt Project. Qt offre des composants d'interface graphique, d'accès aux données, de connexions réseaux, de gestion des fils d'exécution, d'analyse XML, etc

5.2 Les données utilisées

L'algorithmes d'apprentissages a besoins d'être entraîné sur un vaste jeu de données.

5.2.1 Les sources

Deux bases de données sont employées pour l'apprentissage automatique qui sont respectivement kitti et Opene Image V4[14, 1]. Chacune d'elles contient des milliers de types d'images d'objets tels que les véhicules, les feux de circulation,...etc, qui ont été acquises dans des conditions différentes (en termes de résolution, d'éclairage (matin ou soir, ensoleillé, nuageux, etc) et de conditions de prises de vue).

L'ensemble des données comprend 1099 images de taille de $1240 \times 376 \text{ pixels}$, dont 80% (980) sont utilisées pour la phase d'apprentissage. Les instances restantes étant réservées comme données pour la phase validation des performances du modèle. Nos images de l'ensemble de données représentent des scènes différentes d'environnements de conduite : autoroute et urbain. Les niveaux de luminosité dans les scènes sont différents car certaines sont plus éclairées que d'autres.



FIGURE 5.2.1 – Exemples d'images utilisée

La base de données KITTI 2015 est ensemble de données qui ont été enregistrées à partir d'une plate-forme en mouvement lors de la conduite dans et autour de Karlsruhe, en Allemagne. Elle comprend des images des caméras, des numérisations laser de haute précision : mesures GPS et accélérations IMU à partir d'une combinaison de Système GPS / IMU. Le but principal de cet ensemble de

données est de faire avancer le développement des algorithmes de la vision par ordinateur et de la robotique pour des applications ciblant la conduite autonome. La taille des images de cette base de données est de $1240 \times 376 \text{ pixels}$ [14].

Opene Image V4 est un ensemble de données composées d'environ neuf millions d'URL vers des images annotées couvrant plus de 6 000 catégories[1].

5.2.2 Le format et l'organisation

Chaque image est sauvegardée sous le format JPEG. Une fois que les annotations sont effectuées, elles sont enregistrées sous forme de fichiers XML². Par la suite l'ensemble des fichiers sont scindé en deux sous-ensembles, l'un est utilisé au cours du processus d'entraînement et le second est employé pour le test et l'évaluation du modèle.

L'API de détection d'objets TensorFlow ne prend pas les fichiers Xml en entrée, mais elle a besoin de fichiers d'enregistrement pour former le modèle. L'utilisation de ce genre de fichier permet d'avoir un impact significatif sur les performances de notre pipeline³. Pour cela on a convertis nos données au format binaire « Tf Record »⁴. On obtient par la suite deux fichiers (train.tfrecord, test.tfrecord).

TensorFlow nécessite une carte d'étiquettes (label map) qui regroupe tous les objets qui vont être détecté par notre modèle. Cette carte d'étiquettes est utilisée à la fois par les processus de formation et de détection. On a créé notre carte avec sept classes (Animaux, Personne, Voiture, Camion, Vélo, Panneau de signalisation, Feu de circulation).

Pour entraîner notre modèle on a besoin d'un modèle pré-formé donc on a tester plusieurs modèles (Faster , SSD) avec différentes architectures (resnet, Inception v2,...) qui ont été apprises sur différentes base de données (coco, Open Image, Kitti). Après avoir calculer la difference entre le temps d'apprentissage, le temps de la détection en temps réel et la précision voire le tableau 5.1.

2. Le XML ou eXtensible Markup Language est un langage informatique de balisage générique.

3. pipeline : le traitement en pipeline chevauche le prétraitement et l'exécution du modèle d'une étape d'apprentissage. Pendant que l'accélérateur effectue l'étape d'apprentissage n , le CPU prépare les données pour les étapes $n + 1$. Cela réduit le temps de pas au maximum dans l'apprentissage et le temps nécessaire pour extraire et transformer les données

4. Tf Record : est un simple format binaire d'enregistrement orienté, de nombreuses application Tensorflow l'utilisent pour l'apprentissage. Les données binaire occupent moins de temps à copier et peuvent être lues beaucoup plus efficacement.

5.3 Le suivi des performances

Nous avons effectué plusieurs exécutions en variant les configurations du modèle, des données et de l'environnement. Chacune d'elles consomme plusieurs heures. Nous avons commencé par l'ensemble de données de COCO. À partir de cet ensemble, nous avons lancé l'apprentissage de deux modèles de Faster R-CNN avec des architectures CNN différentes un avec resnet et l'autre avec inception v2.

Nom du modèle	Précision mAP
(le plus rapide) <code>ssd_mobilenet</code> (basse résolution)	21
(le plus rapide) <code>ssd_inception_v2</code> (basse résolution)	24
(bon) <code>faster_rcnn_resnet50</code>	30
(plus précis) <code>faster_rcnn_resnet101</code>	32

TABLE 5.1 – Les résultats de l'ensemble de tests avec le metrique de PASCAL VOC

En général, Faster R-CNN est plus précis tandis que SSD est plus rapide. Nous cherchons à trouver un compromis entre précision et rapidité.

L'apprentissage est effectué sur notre ensemble de données avec deux modèles : `ssd_inception_v2` qui est le plus rapide et `faster_rcnn_resnet50` qui donne une bonne précision tout en étant le moins lent parmi les modèles de faster.

5.3.1 Les métriques d'évaluations

Ces métriques sont utilisées principalement pour évaluer les réseaux de neurones. Les mesures actuelles utilisées par le défi de détection d'objet VOC PASCAL actuel sont la courbe Précision x Rappel et la précision moyenne.

Précision : est la capacité d'un modèle à identifier uniquement les objets pertinents. C'est le pourcentage de prédictions positives correctes qui est donné par :

$$Précision = \frac{TP}{TP+FP}$$

Rappel : c'est la capacité d'un modèle à trouver tous les cas pertinents (toutes les boîtes englobant la vérité au sol). C'est le pourcentage de vrais positifs détectés parmi toutes les vérités de terrain pertinentes et est donné par :

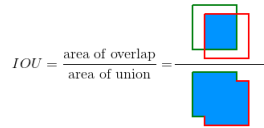
$$Rappel = \frac{TP}{TP+FN}$$

avec True Positive (TP) : Une détection correcte. Détection avec $IOU \geq \text{seuil}$
Faux Positif (FP) : Une détection incorrecte. Détection avec $IOU < \text{seuil}$

Faux négatif (FN) : une vérité de terrain non détectée

True Negative (TN) : Ne s'applique pas. Cela représenterait une erreur de détection corrigée.

et Intersection Over Union (IOU) : Index qui évalue le chevauchement entre deux cadres de sélection. Cela nécessite une boîte englobant la vérité sur le sol et une boîte englobante prédite par la zone d'union entre eux.


$$IOU = \frac{\text{area of overlap}}{\text{area of union}} =$$

5.3.2 La technique

Au moment de l'apprentissage, des fichiers appelés fichiers d'événements contenant son avancement sont générés ou mis à jour. Ces fichiers sont persistants et peuvent être consultés même après la terminaison du processus. Ils contiennent les graphes représentant le changement des valeurs de l'erreur et l'exactitude en fonction du nombre d'itérations pour chacun des deux ensembles d'apprentissage et de validation. De plus, ils incluent une représentation graphique du modèle d'apprentissage ayant plusieurs niveaux de détails qui affichent chaque nœud et son rôle, ainsi que les différentes connexions entre eux et les dimensions des données transférées. Toutes ces informations peuvent être visualisées clairement avec l'outil TensorBoard.

5.4 Configuration de notre modèle

L'API de détection d'objets Tensorflow utilise des fonctions protobuf⁵ pour configurer le processus de l'apprentissage et d'évaluation à un niveau élevé, le fichier de configuration est divisé en 5 parties :

1. **models** : configuration du modèle, définition de quel type de modèle on doit utiliser pour l'apprentissage et l'extraction des caractéristiques.
2. **train_config** : qui décide des paramètres à utiliser pour entraîner les paramètres du modèles (paramètres SGD, valeurs de pré-traitement de l'extracteur de caractéristiques).
3. **eval_config** : qui détermine quel ensemble de métriques seront rapportés pour l'évaluation.

5. Une méthode de traduction des fichiers images dans un format qui peut être stocké dans un fichier, ce qui est utile pour développer des programmes permettant de communiquer entre eux par un fil ou pour stocker des données. Il est indépendant de la langue de la plateforme comme XML mais plus petit plus simple et plus rapide.

4. **train_input_config** : le répertoire de l'ensemble des données de l'apprentissage ainsi que la carte d'étiquettes.
5. **eval_input_config** : le répertoire de l'ensemble des données d'évaluation ainsi que la carte d'étiquettes.

5.4.1 Configuration de modèles

Le modèle Faster R-CNN a été choisi comme extracteur de caractéristique de l'architecture inception v2.

Pour le prétraitement, les images sont redimensionnées en 300 pixels comme minimum dimension et 600 pixels comme maximum .

Par défaut le nombre des anchors est à 9 dans cet configuration il est mis a 36 anchors de différentes tailles.

Afin de bien régulariser notre réseau on a activé le décrochage « dropout⁶ » avec une probabilité de 0.8.

5.4.2 Configuration de train_config

La plupart des réseaux de neurones nécessitent des hyper-paramètres à régler, ces paramètres sont les congurations qui nous permettent de contrôler leurs comportements depuis l'extérieur. Car contrairement aux paramètres internes (poids, biais, . . .), l'algorithme n'as pas la capacité de les ajuster, de plus que leurs valeurs diffèrent d'une tâche à une autre et d'un algorithme à un autre. Afin d'avoir le meilleur paramétrage du réseau, il faut effectuer plusieurs apprentissages avec les différentes combinaisons de valeurs de ces hyper paramètres. Cependant, nous étions limités par le temps et la puissance de calcul.

La taille du lot : Un batch représente le nombre d'exemples à traiter simultanément, vu qu'un ordinateur sans GPU, un batch de taille 1 nous a fournis de bons résultats en termes de performance et de temps.

Le taux d'apprentissage : est le pas avec lequel la fonction d'optimisation essaye d'améliorer l'erreur. Généralement sa valeur doit être petite (comprise entre 10^{-3} et 1) en explorer le maximum d'espaces. Il est à noter qu'une valeur trop petite ralentit le temps car l'apprentissage se déroule lentement. Le taux d'apprentissage et la taille du batch sont en corrélation car il est clair que plus on a d'exemples à traiter à la fois, plus l'apprentissage se déroule rapidement. Nous

6. Dropout : est une approche de la régularisation dans les réseaux de neurones qui aide à réduire l'apprentissage interdépendant entre les neurones. À chaque étape de l'apprentissage, les nœuds individuels sont soit supprimés du réseau avec la probabilité $1 - p$, soit conservés avec la probabilité p , de sorte qu'il reste un réseau réduit ; les arêtes entrantes et sortantes vers un nœud abandonné sont également supprimées.

avons essayés plusieurs valeurs et la valeur 0.0003 ainsi que l'étape de planification de valeur fixée à 50000*pas* ont donnée les meilleurs résultats.

Le nombre d'itérations : D'abord il faut expliquer la notion 'd'epochs' , lorsque l'algorithme parcourt tous les exemples d'apprentissage, on dit qu'il a effectué une 'epoch' . Le nombre d'itérations est donc déterminée par la formule suivante :

$$\frac{\text{Le nombre d'exemples}}{\text{La taille batch}}$$

Donc avec 980 exemples et un batch de taille 1, il nous a fallu 980 itérations pour compléter 1 époque.

5.5 Test et prédiction

Nous avons effectué plusieurs exécutions en variant les configurations du modèle, des données et de l'environnement. Chacune d'elles consomme plusieurs heures.

Les graphes des exécutions sont présentés dans les figures suivantes. Les courbes oranges représentent les valeurs issues de l'évaluation des performances sur les cycles de batch des données d'apprentissage.

Les graphes de fonction du coût de chaque modèle sont présentés dans les figures 5.5.2, 5.5.4, 5.5.6 respectivement.

Résultats SSD_inception_v2 :



FIGURE 5.5.1 – Images d'évaluation du modèle SSD_inception_v2

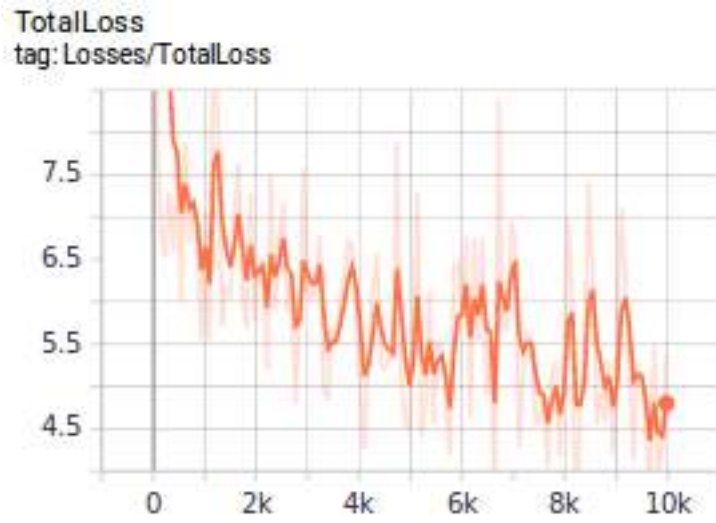


FIGURE 5.5.2 – Les performances du modèle SSD_inception_v2

5.5.2 ici, on voit que pour le modèle SSD_inception_v2 la fonction du coût diminue jusqu'à la valeur 4.5 dans la fin de l'apprentissage, ce qui est une très haute valeur.

Résultats Faster_resnet :



FIGURE 5.5.3 – Images d'évaluation du modèle Faster_resnet



FIGURE 5.5.4 – Les performances du modèle Faster_resnet

Pour la fonction du coût de ce modèle, la pente diminue jusqu'à la valeur 1.4, ce qui est considéré comme remarquable.

Résultats Faster_inception_v2 (notre modèle) :

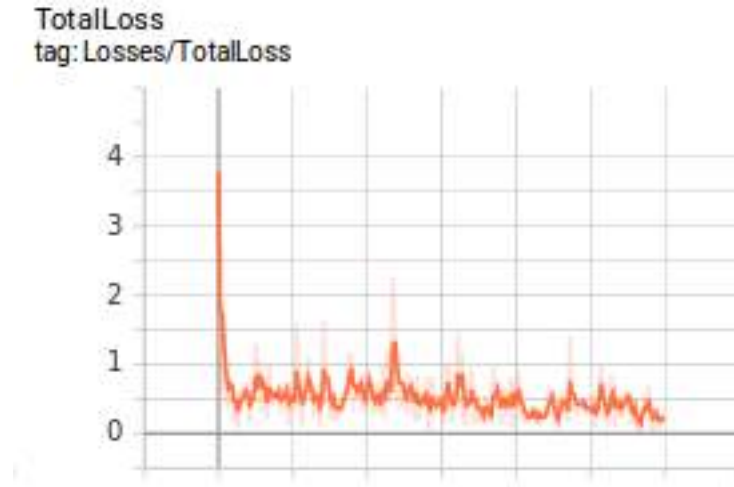


FIGURE 5.5.6 – Les performances du modèle crée par Faster_inception_v2

Concernant la fonction de perte de notre modèle, la valeur était très élevée au début vu que les poids étaient aléatoirement initialisés, puis elle a diminué à 0.2.

Nom du modèle	Temps training	images	nombre d'objets prédits	probabilité détection.
SSD_inception_v2	10h	1	5 (voitures)	98%~68%
		2	0	-
		3	3 (voitures)	83%~58//5
		4	2 (voiture, personne)	85% ,73%
Faster_resnet	16h	1	7 (voitures)	99%~87%
		2	2 (voitures)	98%, 50%
		3	6 (voitures) 1 (panneau)	99%- 85%
		4	1 (voiture) 2 (personnes)	97%~75%
Faster_inception_v2	7h	1	6(voitures)	99%~85%
		2	1 (voiture)	95%
		3	4 (voitures)	99%~65%
		4	2 (personne , voiture)	97%,95%

TABLE 5.2 – Tableaux comparatifs des résultats

Nous citons que le temps de détection en temps réel en utilisant ces modèles sur notre ensemble de données varie.

Nous pouvons conclure que nous avons réussi à établir un bon compromis entre précision et rapidité.

5.6 Implémentation

La plate-forme sur laquelle l'algorithme de détection sera implémenté est un véhicule intelligent ROBUCAR [7] Une voiture robot en cours d'expérimentation à la division robotique du centre CDTA [3] qui est composée de capteur et d'un ordinateur qui gère toutes les communications entre les périphériques et où Linux et ROS (Robot Operating System) sont déjà installés. Une image du ROBUCAR est présentée à la figure 5.6.1. L'idée est que le véhicule peut se déplacer et détecter des objets en mouvement. Les objets doivent être placés dans une position où le capteur peut les voir correctement.

ROS (Robot Operating System) : ROS (Robot Operating System) est un système d'exploitation pour robots. De même que les systèmes d'exploitation pour PC, serveurs ou appareils autonomes, ROS est un système d'exploitation complet pour la robotique de service.

ROS est un méta système d'exploitation, quelque chose entre le système d'exploitation et le middleware.

Il fournit des services proches d'un système d'exploitation (abstraction du matériel, gestion de la concurrence, des processus...) mais aussi des fonctionnalités de haut niveau (appels asynchrones, appels synchrones, base de données centralisées, système de paramétrage du robot...).



FIGURE 5.6.1 – ROBUCAR du centre de recherche CDTA

Conclusion générale

Le sujet traité dans ce mémoire de master est l'application et l'exploitation de l'apprentissage profond (deep learning) pour la détection, la localisation et la classification de différents objets existants sur la route.

La première étape de notre travail a été un travail de recherche pour comprendre les notions de base du traitement d'image, la vision par ordinateur et ainsi réaliser un état de l'art sur la détection et la classification d'objets. Ce travail de recherche nous a permis de savoir que toutes les méthodes, qu'elles soient classiques ou récentes, utilisant les réseaux de neurones ou pas, requièrent une chaîne de traitement d'images afin d'arriver à la tâche de détection. Les méthodes classiques comme le machine learning traitent davantage les images pour extraire toute l'information utile pour une bonne classification. Cette tâche demande un nombre considérable de calculs ne correspondant pas à une application en temps réel telle que nécessaire pour notre étude.

Nous avons opté pour une approche plus efficace en termes de temps et de précision basée sur les réseaux de neurones profonds à convolution (CNN). Cette méthode utilise les couches de convolution et un ensemble de filtres. Les CNN permettent l'extraction des caractéristiques d'une image jugée trop importante pour la détection en utilisant un Perceptron multicouche. Une multitude de modèles existe comme R-CNN, Faster R-CNN et SSD où chacun d'eux a sa propre hiérarchie et ses propres étapes pour classer et segmenter les données visuelles.

Nous avons, dans ce travail de recherche, étudié et analysé les différentes architectures de ces modèles. L'entraînement de ces réseaux exige beaucoup de ressources pour ses calculs (mise à jour de millions de paramètres) et ne se fait donc qu'une fois pour atteindre une fiabilité jugée acceptable.

Pour relever le défi de l'efficacité du modèle nous avons essayé plusieurs modèles pour adopter à la fin le modèle Faster R-CNN avec l'architecture Inception V2. Ce modèle est le plus adapté à notre étude mais répond aussi aux contraintes liées au matériel informatique utilisé.

Dans le but de réduire le temps du traitement et d'améliorer la précision de la détection, nous avons accompli un apprentissage des objets routiers. Nous avons également procédé à un changement dans les paramètres de configuration du modèle

Faster R-CNN qui nous a permis d'avoir une amélioration dans les résultats.

Ce travail demande un complément et un approfondissement. Nous souhaitons qu'il soit une initiative pour d'autres recherches sur ce sujet.

Bibliographie

- [1] Bases de donnees openimage v4. In *github.com/openimages/dataset*, 2017.
- [2] Paul Bao, Lei Zhang, and Xiaolin Wu. Canny edge detection enhancement by scale multiplication. *IEEE transactions on pattern analysis and machine intelligence*, 27(9) :1485–1490, 2005.
- [3] CDTA. Centre de recherche et de développement avancé. In *http://www.cdta.dz*, 1982.
- [4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *international Conference on computer vision & Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE Computer Society, 2005.
- [5] John G. Daugman. Computer vision computer science tripos : 16 lectures by j g daugman. 2010.
- [6] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.
- [7] Fatma Zohra Foudil. Robucar : la voiture intelligente made in dz. In *https://www.djazairess.com/fr/elwatan/428354*, 2013.
- [8] Google. outil d'apprentissage automatique developpe par google.. In *www.tensorFlow.com*, 2015.
- [9] Jim Hugunin. une extension du langage de programmation python. In *www.numpy.org*, 1995.
- [10] Joel saiak Josef le pavac, Jean mousterde. Projet scientifique collectif application d'un reseau de neurones a la predictions. page 31, 2006.
- [11] Ingo Lutkebohle. Planete robots 2009. In *www.planeterobots.com*, 2009.
- [12] Ingo Lutkebohle. Planete robots 2009. In *www.planeterobots.com*, 2009.
- [13] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4) :115–133, 1943.