MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABBES LAGHROUR DE KHENCHELA
FACULTE DES SCIENCES ET DE LA TECHNOLOGIE

Département mathématique et informatique

N° de série :…

Mémoire de fin d'études

Pour l'obtention du diplôme de Master (L.M.D)

Spécialité : Sécurité et technologie web

Thème : Apprentissage profond pour la détection et le suivi de véhicules dans un flux vidéo

Présenté par : Adami Fatima Zohra
Salmi Kamilia

Encadreur : Abbas Fayçal.

Membres de jury : Nessah Djamel
Hioual ouidad

Le :24/06/2018

Année universitaire : 2017/2018.

Table des matières

Remerciement

Nous tenons tout d'abord à remercier Dieu le tout puissant et miséricordieux, qui nous a donné le courage durant ces longues années d'étude, la force et la patience d'accomplir ce Modeste travail.

Nous tenons à remercier sincèrement notre encadreur Mr : Abbes Fayçal, pour ces précieux conseils, orientation, confiance et patience qui ont constitué un apport considérable sans lequel ce travail n'aurait pas pu être mené au bon port. Qu'il trouve dans ce travail un hommage vivant à sa haute personnalité.

Ces remerciements vont tout d'abord au corps professoral et administratif de la Faculté des Science et Technologie, ensuit à tous les professeurs qui nous ont enseigné et qui par leurs compétences nous ont soutenu dans la poursuite de nos études.

Nos vifs remerciements vont également aux membres du jury pour l'intérêt qu'ils ont porté à notre recherche.

Nous remercions de tous nos cœurs nos parents, pour nous avoir toujours soutenu et cru en nos.

A nos familles et nos amis qui par leurs prières et leurs encouragements, on a pu surmonter tous les obstacles.

Enfin, nous tenons également à remercier toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Merci à tous et à toute

Dédicace

D'un sentiment plain d'amour, de sincérité de fidélité, Je dédie ce modeste travail à :

À l'homme de ma vie, mon exemple éternel, mon soutien moral et source de joie et de bonheur, celui qui s'est toujours sacrifié pour me voir réussir, que dieu te garde dans son vaste paradis, à toi
*Mon père (**ABD ALHALIM**).*

*À la lumière de mes jours, la source de mes efforts, la flamme de mon cœur, ma vie et mon bonheur ; maman que j'adore (**NADIA**).*

*À mes frères (**HAKOU ET RAFIK**), mes anges gardiens et ma p'tite princesse (**DOUDI**).*
Pour l'âme de mes grands pères et mères
*À mon binôme Fatima et tous les membres de la famille **'SALMI'** et **'NEKKAA'**.*

À mes très chers amis(es).

KAMILIA

Je dédie ce travail :

À mes chers parents qui m'ont toujours épaulé et soutenu, sans eux je n'y serai jamais arrivé, Tous les mots du monde ne sauraient exprimer l'immense amour que je vous porte, ni la profonde gratitude que je vous témoigne pour tous les efforts et les sacrifices que vous navez jamais cessé de consentir pour mon instruction et mon bien-être.

À mes chères sœurs : ***KALTOUM, MERYEM, HAFSA.***

À mes chers frères : ***AHMED, IBRAHIM.***

À ***ARWA, MAYSSONE, ABD EL RAHMAN***

À ma grand-mère qui prie toujours pour moi et à laquelle je tiens tant.

*À tous les membres de ma grande famille (****ADAMI ET HABBECHE****)*

À mon cher binôme ***KAMILIA*** *pour tout ce qu'elle a fait pour la réussite de ce travail.*

À mes plus proches amies : ***SABRINA, KHAWLA***

À mes amis d'études.

À tous mes enseignants sans exception.

FATIMA

Liste des figures

Liste des tableaux

Liste des formules

Résumé

Dans ce mémoire on va travailler sur la détection et la reconnaissance des véhicules à partir d'un flux vidéo. Comparés aux méthodes traditionnelles de détection et de classification d'objets, les méthodes d'apprentissage profond sont un nouveau concept dans le domaine de vision par ordinateurs.

Notre modèle s'opère en deux étapes : une étape de préparation de données, elle consiste à appliquer des traitements sur les images composant la dataset afin d'extraire les caractéristiques, la seconde étape consiste à appliquer le concept de réseaux de neurones convolutifs afin de classifier les véhicules.

Notre méthode a donné des meilleurs résultats en termes de précision, de détection et de classification où nous avons obtenu une précision de 99.2%.

Mots Clés — Détection et reconnaissance de véhicule, apprentissage profond, classification, les réseaux de neurones convolutifs(CNN).

Abstract

In this memoir, we will work on the detection and recognition of vehicles from a video stream. Compared to traditional methods of object detection and classification, deep learning methods are a new concept in the field of computer vision.

Our model operates in two stages: a data preparation step, it consists of applying treatments on the images composing the dataset in order to extract the characteristics, the second step is to apply the concept of convolutional neural networks to classify vehicles.

Our method gave better results in terms of precision, detection and classification where we obtained an accuracy of 99.2%.

Keywords —Vehicle detection and recognition, deep learning, classification, convolutional neural networks (CNN).

ملخص

في هذه المذكرة، سنعمل على تحديد والتعرف على وجود السيارات في الطريق وذلك من خلال تسجيل الفيديو.
بالمقارنة مع طرق الكشف، التعرف والتصنيف التقليدية، فإن أساليب التعلم العميق هي مفهوم جديد في مجال رؤية الكمبيوتر.
يعمل نموذجنا على خطوتين: خطوة إعداد البيانات ، حيث يتم فيها تطبيق المعالجات على الصور التي تتألف منها مجموعة البيانات من أجل استخراج خصائص هاته الصور، والخطوة الثانية هي تطبيق مفهوم الشبكات العصبية التلافيفية على تصنيف و تحديد السيارات.

أعطت طريقتنا نتائج أفضل من حيث الدقة والكشف والتصنيف حيث حصلنا على دقة 99.2 ٪.

الكلمات المفتاحية -- تحديد المركبات والاعتراف بها، التعلم العميق، التصنيف، الشبكات العصبية التحويلية (CNN) .

INTRODUCTION GENERAL

L'une des nombreuses capacités humaines est la capacité remarquable de comprendre et d'analyser l'environnement. À partir des signaux fournis par son système oculaire, l'homme est capable de décrire les objets qui l'entourent de manière très précise et très rapidement. On peut notamment souligner la capacité de l'homme à reconnaître les objets et les localiser tout en les caractérisant finement : leurs formes, leurs couleurs, leurs orientations... L'un des nombreux objectifs des chercheurs en vision par ordinateur est de construire des systèmes informatiques "intelligents" capables d'analyser des vidéos de manière aussi performante que l'homme.

Depuis que le traitement des données video en temps réel est devenu sérieusement envisageable, pour des problématiques aussi diverses que la sécurisation de l'accès à des bâtiments, la surveillance de malades dans des hôpitaux, ou encore la facturation des véhicules aux péages des autoroutes. Nous avons utilisé ce traitement de données pour améliorés la détection de véhicule et de rendre la conduite plus facile.

La détection de véhicules permet d'utiliser une variété applications de système d'intelligence artificielle pour plusieurs buts, notamment : le transport intelligent, la surveillance automatique, la conduite autonome, et la garantie de sécurité du conducteur. Ce type de reconnaissance est un défi dans le domaine de la vision par ordinateur.

L'objectif final de cette thèse est de nous permettons de détecter les véhicules à partir d'un vidéo obtenu via une caméra placée au-dessous de rétroviseur ; cette partie est diviser en deux phases : la première phase permet l'extraction des caractéristiques à partir d'une vidéo obtenu et la deuxième phase permet de détecter les véhicules qui circulent devant nous et de tracer les lignes trajectoire de notre véhicule.

Dans ce travail nous nous focalisons sur la détection et la reconnaissance des véhicules dans un flux vidéo. Pour cette raison nous avons utilisé la technique de réseaux de neurones convolutionnels CNN qui est un algorithme d'apprentissage pour permettre la reconnaissance et classification des véhicules. Le but principal de notre travail est donc de réduire l'effort

humain et offre des perspectives de recherche pour rendre la conduite des véhicules plus agréables et presque autonome.

Ce document est divisé en 5 chapitres, il est organisé comme suit :

Chapitre1 : le premier chapitre est divisé en trois partie, la première partie contient l'intelligence artificielle, l'historique d'intelligence artificielle et ses domaines, la deuxième partie inclut l'apprentissage automatique, les types de systèmes d'apprentissage automatique et leur modèle. La troisième partie contient l'apprentissage profond et quelques algorithmes de ce apprentissage et enfin contient un tableau qui identifier la différence entre l'apprentissage automatique et l'apprentissage profond.

Chapitre2 : le deuxième chapitre expose les réseaux de neurone en terme général et les réseaux de neurones convolutif (CNN) en spécifique.

Chapitre3 : le troisième chapitre traite en global la détection d'objet, leurs défis, les techniques standards pour l'extraction des régions sur lesquelles va être appliqué le modèle (détection par fenêtre glissante (sliding window) et la détection par proposition d'objets), quelques méthode de détection (YOLO, R-CNN, HOG), et la détection de véhicule en spécifique.

Chapitre4 : le quatrième chapitre présent les détails d'implémentation. Ce chapitre définit l'architecture de modèle qu'on a créé pour détecter et classifier les véhicules et aussi pour calculer la déviation estimée du véhicule.

Le Chapitre5 conclut ce mémoire en présentant l'ensemble des résultats expérimentaux obtenus de flux video pour la détection de véhicule et des représentations graphiques pertinentes qui montre l'efficacité de notre approche.

Une conclusion synthétiser les contributions et les résultats obtenus et proposera des perspectives à nos travail.

I. L'intelligence artificielle

Depuis l'émergence de la robotique et de l'informatique, les chercheurs essaient d'injecter des notions d'intelligence humaine dans des machines. Étant conçue et fabriquée par l'homme, on qualifie cette forme d'intelligence comme "L'intelligence artificielle" ou IA.

L'intelligence artificielle est l'un des champs les plus importants et les plus passionnants de l'informatique. Cette dernière couvre les principes fondamentaux de l'informatique et des systèmes d'information afin de maximiser vos chances pour de futurs emplois. Il offre également la possibilité d'effectuer des recherches de pointe dans des domaines de haut niveau.

I.1 Quelques définitions

Qu'est-ce que l'intelligence ?

- « La faculté de connaître et comprendre, incluant la perception, l'apprentissage, l'intuition, le jugement et la conception. » (Petit Robert)
- « La faculté de connaître et de raisonner. » (Dictionnaire American Heritage)
- « Application de la connaissance à la résolution de problèmes. » (Newell et Simon).

L'intelligence artificielle (IA) est la science dont le but est de faire par une machine des tâches que l'homme accomplit en utilisant son intelligence.

I.2 Un court historique d'IA

L'intelligence artificielle (IA) est un domaine qui a une longue histoire et qui est toujours en constante évolution.

- 1943 : McCulloch et Pitts créent le modèle du neurone formel.
- 1948 : Création de la cybernétique (science des systèmes) par Norbert Wiener.
- 1949 : Hebb établit la première règle d'apprentissage neuronal.

- 1950 Shannon, 1952 Samuel, 1953 Turing : machine pour jouer aux échecs.
- 1956 Workshop ou est né le terme "intelligence artificielle".
- 1959 Rochester : Geometry Theorem Prover.
- 1958 McCarthy au MIT créé le LISP et le "time sharing". Créé DIGITAL.
- 1960 John McCarthy, Allen Bewell & Herbert Simon: L'ordinateur peut être utilisé pour autre chose que des calculs « manipuler des symboles » (idée proposée par Ada Lovelage amie de Babbage 1842).
- 1969 arrêt des RNs (Minsky & Paper 1969) limitations des perceptrons.
- 1969-1979 : systèmes experts.
- Depuis 1986 : retour des réseaux de neurone. [1]

I.3 Grands domaines d'IA

L'intelligence artificielle apparaît dans des domaines extrêmement variés, c'est un principe polyvalent. On l'utilise dans des domaines où on ne la voit pas, comme ceux de la vie quotidienne (GPS, informatique, reconnaissance vocale, météo...), et aussi on la trouve dans des domaines beaucoup plus spectaculaires, à la limite de la Science-fiction, comme la robotique ou les jeux. Voici quelques exemples :

- **Apprentissage / Adaptation :**
  Exemples : construction de systèmes experts, classification automatique de galaxie, contrôleurs de robots….
- **Reconnaissance et synthèse de la parole :**
  Exemples : réservation d'hôtel, annuaire téléphone.
- **Reconnaissance et synthèse d'images :**
  Exemples : effets spéciaux au cinéma, vidéo surveillance.
- **Reconnaissance de l'écriture :**
  Exemples : reconnaissance chèques, code postaux.
- **Langage naturel :**
  Exemples: interface, text mining, web mining.
- **Planification**
- **Aide à la décision :**

Exemple : contrôle de trajectoire du satellite voyagé, (Systèmes experts).

- **Aide à la programmation :**

  Exemple : agents d'interface.

- **Médecine**
- **Jeux :**

  Exemples : Echecs (DeeperBlue à 2600), Checkers (Champion), Othello (Champion), Back Gammon (champion), GO (bon amateur) [1].

II. L'apprentissage automatique

L'apprentissage automatique est un sous-domaine de l'intelligence artificielle (IA), il consiste à doter l'ordinateur de capacités pour se programmer lui - même, au lieu de le programmer manuellement, ainsi il pourra résoudre de nouveaux problèmes à partir de données déjà fournies. En général, l'objectif de l'apprentissage automatique est de comprendre la structure des données et de les intégrer dans des modèles qui peuvent être compris et utilisés par tout le monde. Bien que l'apprentissage automatique soit un domaine de l'informatique, il diffère des approches informatiques traditionnelles, en effet dans cette dernière, les algorithmes sont des ensembles d'instructions explicitement programmées utilisées par les ordinateurs pour calculer ou résoudre des problèmes. Les algorithmes d'apprentissage automatique permettent aux ordinateurs de s'entraîner sur les entrées de données et utilisent l'analyse statistique pour produire des valeurs qui se situent dans une plage spécifique. Pour cette raison, l'apprentissage automatique facilite l'utilisation des ordinateurs dans la construction de modèles à partir de données d'échantillonnage afin d'automatiser les processus de prise de décision en fonction des données saisies.

II.1 Quelques définitions :

L'apprentissage est une phase du développement d'un réseau de neurones durant laquelle le comportement du réseau est modifié jusqu'à l'obtention du comportement désiré. L'apprentissage neuronal fait appel à des exemples de comportement [2].

Apprentissage automatique (Machine Learning) est une branche de l'intelligence artificielle d´éditée à l'étude de modèles et d'algorithmes capables d'apprendre automatiquement. Ceci implique résoudre une variété de tâches en se basant sur des données existantes, avec un minimum d'`a priori humain [3]. Le but est double :

- Prédire un comportement face à une nouvelle donnée.
- Approximer une fonction ou une densité de probabilité.

Voici une définition un peu plus générale :

- **L'apprentissage automatique** est le domaine d'étude qui permet aux ordinateurs d'apprendre sans être explicitement programmés. (Arthur Samuel, 1959).

- (Tom Mitchell, *1997*) dit qu'un programme informatique apprend de l'expérience E par rapport à une tâche T et à une mesure de performance P, si sa performance sur T, telle que mesurée par P, s'améliore avec l'expérience E.

II.2 Pourquoi l'apprentissage automatique ?

Machine learning (ML) utilise des ordinateurs pour simuler l'apprentissage humain et permet aux ordinateurs d'identifier et d'acquérir des connaissances du monde réel, et d'améliorer les performances de certaines tâches en fonction de ces nouvelles connaissances [4].

Pour mieux comprendre l'utilisation de l'apprentissage automatique, nous allons sites quelques domaines de ce dernier :

- Auto-conduite de Google.
- Détection de cyber-fraude.
- Moteurs de recommandation en ligne comme des suggestions d'amis sur Facebook.
- Netflix présentant les films et émissions que vous aimerez et «plus d'éléments à prendre en compte» et «obtenez-vous un petit quelque chose» sur Amazon sont autant d'exemples d'apprentissage automatique appliqué.

Tous ces exemples font écho au rôle essentiel que l'apprentissage automatique a commencé à prendre dans le monde riche en données d'aujourd'hui. Les machines peuvent aider à filtrer des informations utiles qui aident à des avancées majeures, et nous voyons déjà comment cette technologie est mise en œuvre dans une grande variété d'industries.

Le flux de processus représenté ici représente le fonctionnement de l'apprentissage automatique [5].

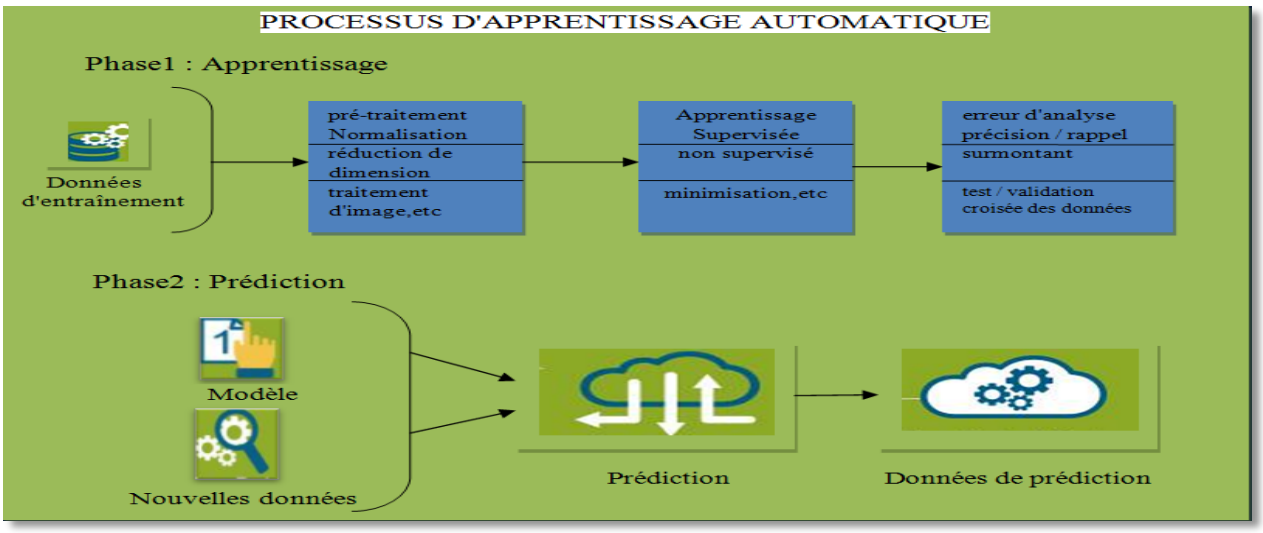


Fig 1.1 Processus d'apprentissage automatique [5].

II.3 Types de systèmes d'apprentissage automatique

Il existe différents systèmes d'apprentissage automatique son classé en plusieurs catégories en fonction de :

- Qu'ils soient entraînés ou non avec une supervision humaine (supervisée, non supervisée, semi-supervisée et apprentissage par renforcement).
- Qu'ils puissent ou non apprendre de manière incrémentale à la volée (en ligne ou par apprentissage par lots).
- Fonctionne-t-il simplement en comparant des nouveaux points de données à des points de données connus, ou détecte-t-elle les modèles dans les données d'apprentissage et construit un modèle prédictif, comme le font les scientifiques (apprentissage basé sur l'instance ou modèle). [6]

3.1 Supervision humaine

Les systèmes d'apprentissage automatique peuvent être classés en fonction de la quantité et du type de supervision qu'ils reçoivent pendant l'apprentissage. Il y a quatre catégories principales : l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage semi-supervisé et l'apprentissage par renforcement.

3.1.1 Apprentissage supervisé

En apprentissage supervisé, les données d'apprentissage que vous alimentez dans l'algorithme incluent les solutions souhaitées, appelées étiquettes [6].

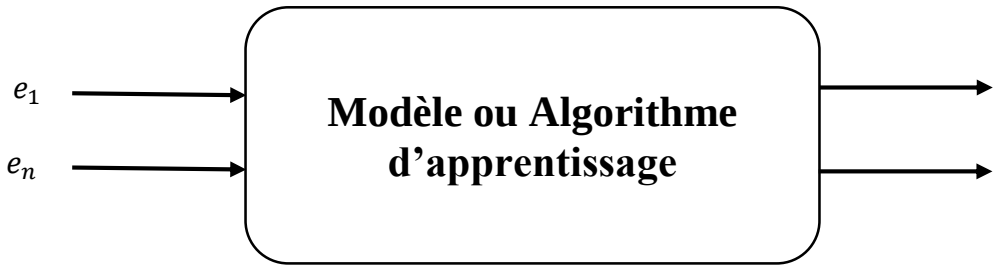


Fig 1.2 Schéma d'un modèle supervisé.

Voici quelques-uns des algorithmes d'apprentissage supervisé [6] les plus importants :

- k-Plus proches voisins.
- Régression linéaire.
- Machines vectorielles de support (SVM).
- Arbres de décision et forêts aléatoires.
- Les réseaux de neurones.

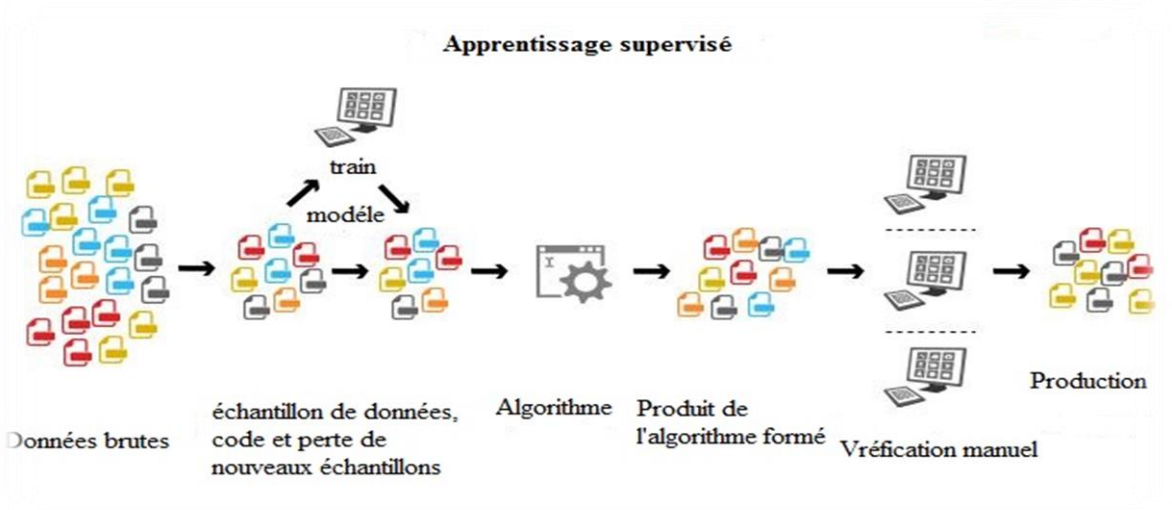


Fig 1.3 Explique le déroulement d'apprentissage supervisé [5].

3.1.2 Apprentissage non supervisé

Dans l'apprentissage non supervisé, comme vous pouvez le deviner, les données d'entraînement ne sont pas étiquetées. Le système essaie d'apprendre sans enseignant [6]**.**

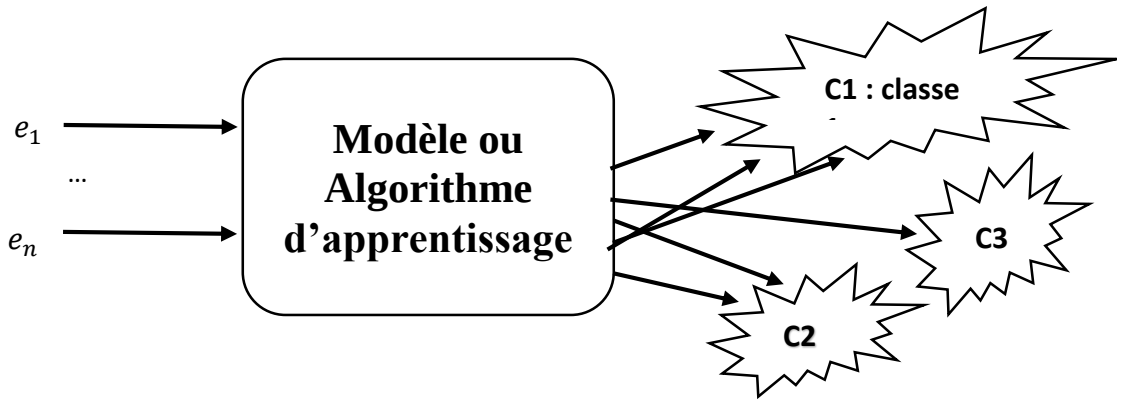


Fig 1.4 Schéma d'un modèle non-supervisé

Applications :

- Segmenter les consommateurs dans les bases de données d'achats.
- Compression d'images : réduire le nombre de couleurs utilisées.
- Bio-informatique : découvrir des patrons dans l'ADN.

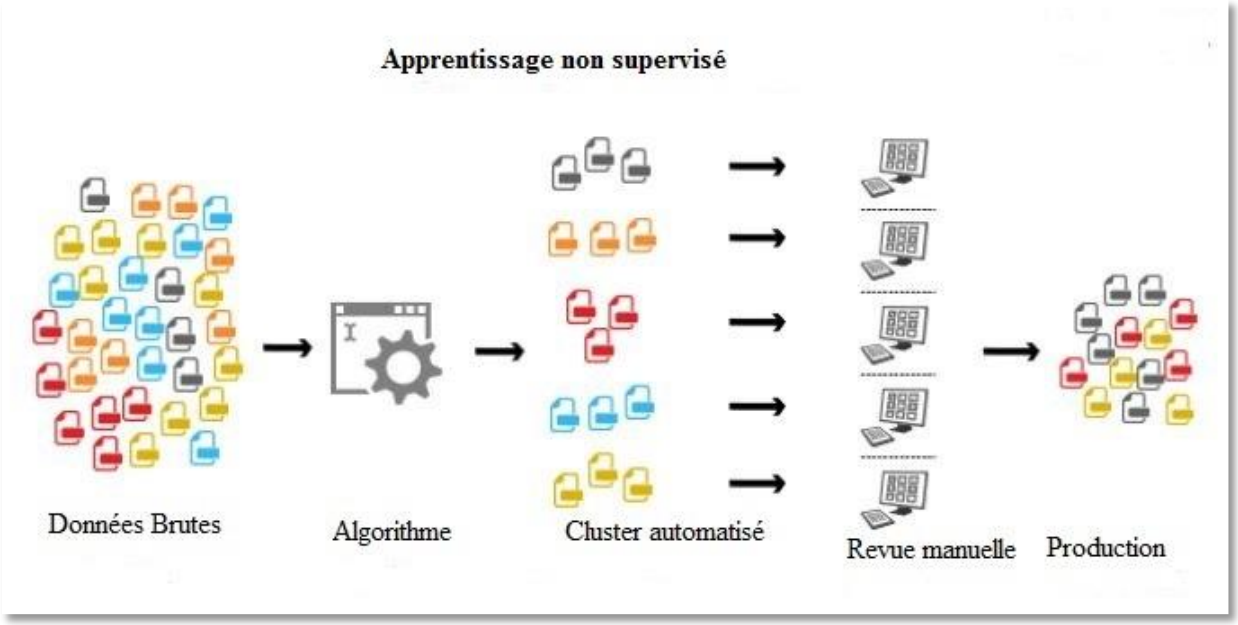


Fig 1.5 Explique le déroulement d'apprentissage non-supervisé [5].

Voici quelques-uns des algorithmes d'apprentissage non supervisé [6] les plus importants :

• Clustering :
  - k-Moyens.
  - Hierarchical Cluster Analysis (HCA).
• Visualisation et réduction de la dimensionnalité :
  - Analyse des composants principaux (PCA).
  - Inclusion linéaire-linéaire (LLE).
  -Intégration de voisin stochastique t distribuée (t-SNE).
• Apprentissage des règles d'association :
  -A priori.
  -Éclat.

3.1.3 Apprentissage par renforcement

L'apprentissage par renforcement est une bête très différente. Le système d'apprentissage, appelé agent dans ce contexte, peut observer l'environnement, sélectionner et effectuer des actions, et obtenir des récompenses en retour. Il doit ensuite apprendre par lui-même quelle est

la meilleure stratégie, appelée politique, pour obtenir le plus de récompense possible. Une politique définit quelle action l'agent doit choisir lorsqu'il se trouve dans une situation donnée [6].

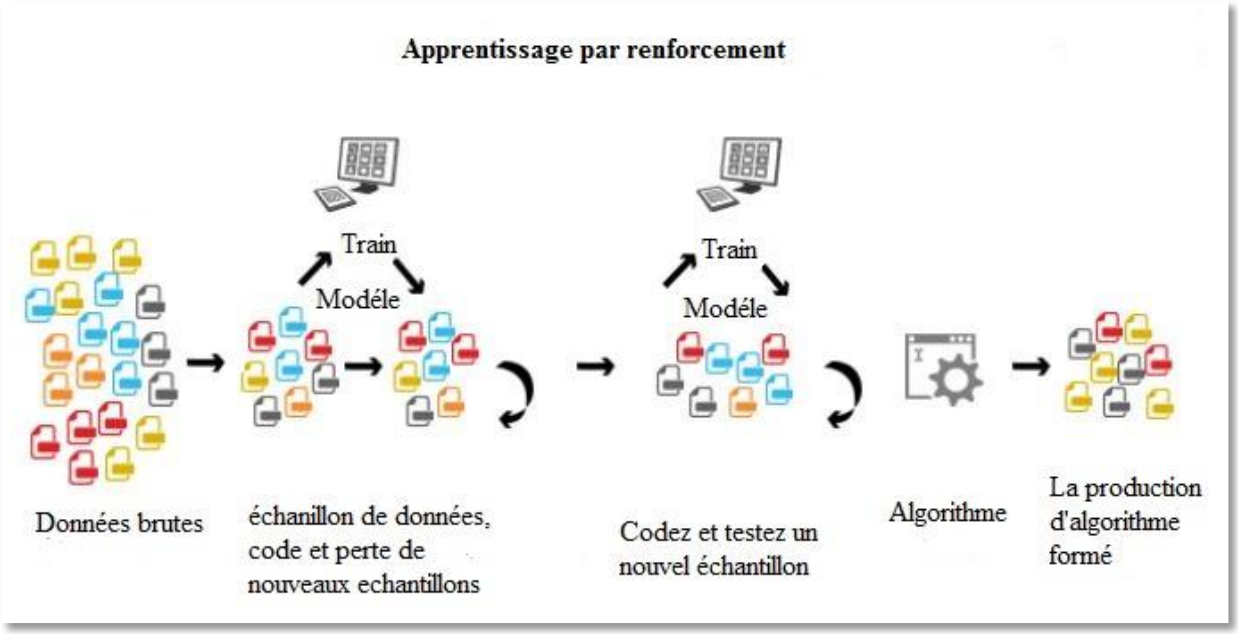


Fig 1.6 Explique le déroulement d'apprentissage par renforcement [5].

Applications :

- Jeux : avec un ou plusieurs participants.
- Robotique : navigation dans un environnement.
- Agents : prises de décision.

3.1.4 Apprentissage semi-supervisé

Les données d'entrée sont constituées d'exemples étiquetés et non étiquetés. Ce qui peut être très utile quand on a deux types de données, car cela permet de ne pas en laisser de côté et d'utiliser toute l'information [7].

Fig 1.7 Schéma d'un modèle semi-supervisé ou incrémental.

Il arrive parfois qu'on puisse avoir beaucoup de données d'entrée, mais qu'il soit coûteux d'obtenir des cibles pour ces données, ou bien qu'il soit possible d'avoir des cibles pour seulement un sous-échantillon de l'ensemble. Cela arrive souvent dans le cas de l'audio musical où il est facile de trouver des bases de données immenses d'exemples non-étiquetés en grande quantité. Par contre, selon la tâche à résoudre, il se peut que l'étiquetage nécessite le travail d'un expert qui devra annoter à la main chaque exemple. Il est possible de tirer parti de cette situation grâce au cadre semi-supervisé. On suppose dans ce cas qu'on peut tirer de l'information supplémentaire à partir des données non-étiquetées, ce qui améliorera la performance de la tâche supervisée [8].

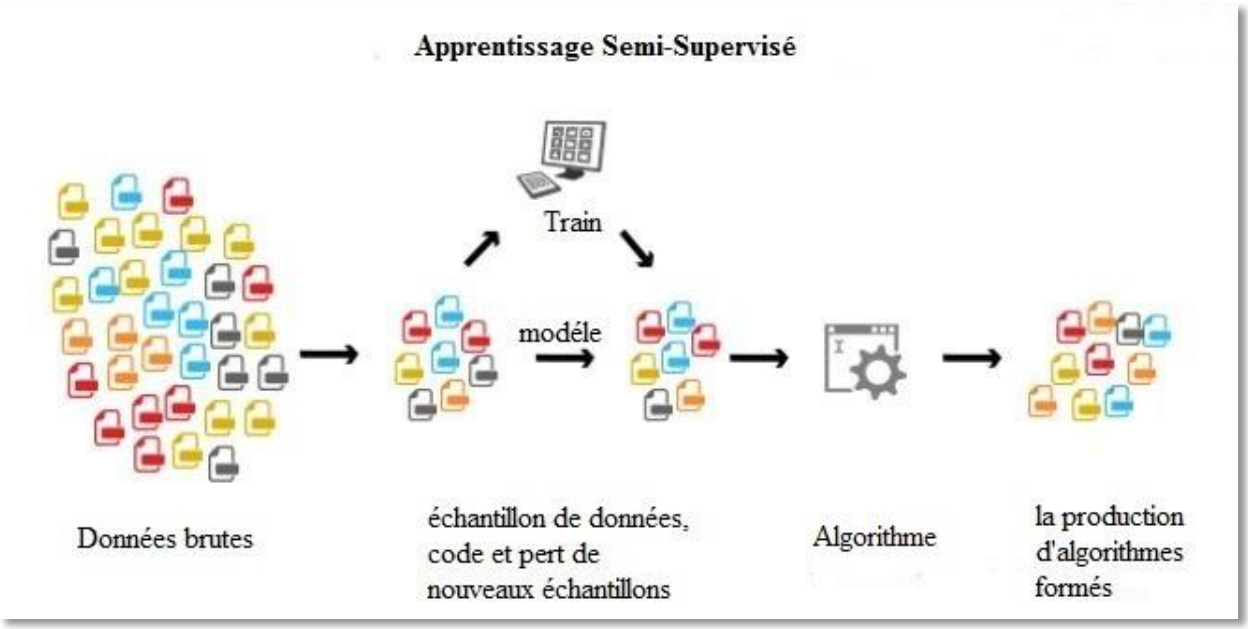


Fig 1.8 Explique le déroulement d'apprentissage semi-supervisé [5].

II.4 Principaux défis de l'apprentissage automatique

En bref, puisque votre tâche principale est de sélectionner un algorithme d'apprentissage et de l'entraîner sur certaines données, les deux choses qui peuvent mal tourner sont "mauvais algorithme" et "mauvaises données" [6].

Voici quelques exemples de mauvaises données :

- Quantité insuffisante de données d'entraînement.
- Données de formation non représentatives.
- Données de mauvaise qualité.
- Caractéristiques non pertinentes.
- Suréquiper les données d'entraînement.

II.5 Les types de modèle

Il existe une grande variété de modèles en apprentissage machine. Nous allons évoquer ici quelques caractéristiques de ces derniers [9].

- ✓ **Paramétriques / Non-paramétriques**

L'espace des fonctions dans lequel nous recherchons des solutions F peut soit :

- Etre défini par un ensemble fini de paramètres θ, dans ce cas on parle de modèle paramétrique.
- Dans le cas contraire, si on ne peut pas représenter de cette façon toute fonction appartenant à F, on parle de modèles non-paramétriques. La formulation de la solution dépend de l'ensemble des données.

- ✓ **Probabilistes / Non-probabilistes**

Les modèles en apprentissage machine peuvent être définis dans un cadre probabilistique ou non. Avec un modèle probabiliste, on peut utiliser la théorie des probabilités pour tirer des conclusions, obtenir des garanties et étendre ou manipuler le modèle. En contre-partie il faut respecter les contraintes mathématiques de ce cadre de travail.

- ✓ **Génératifs / Discriminatifs**

Certaines approches capturent implicitement ou explicitement la distribution marginale des entrées, P(X), ou jointe des entrées et sorties, P(X, Y). Ces modèles sont dit génératifs : on peut générer des données synthétiques par échantillonnage.
Les modèles qui capturent uniquement la distribution conditionnelle P (Y |X) ou qui apprennent une fonction f(x) qui relie directement l'espace d'entrée à celui des sorties, de façon à minimiser une fonction de coût : C (y, f(x)), sont dit discriminatifs.

- ✓ **Linéaires / Non-linéaires**

En classification, un modèle est un classifieur linéaire si la surface de séparation entre deux classes est définie par un hyperplan dans l'espace d'entrée. Cela est important pour déterminer le pouvoir discriminatif du modèle. Un exemple populaire : un classifieur linéaire ne peut apprendre l'opérateur XOR.

III. L'apprentissage profond

Après avoir présenté de manière globale l'apprentissage automatique, nous allons maintenant intéresser plus particulièrement à l'apprentissage profond.
L'apprentissage profond est un sous-ensemble de l'apprentissage automatique, utilise un niveau hiérarchique de réseaux neuronaux artificiels pour réaliser le processus d'apprentissage automatique.

III.1 Un tour d'horizon sur l'apprentissage profond

Dans les années 1950, le mathématicien britannique Alan Turing imagine une machine capable d'apprendre, une «Learning Machine». Au cours des décennies suivantes, différentes techniques de Machine Learning ont été développées pour créer des algorithmes capables d'apprendre et de s'améliorer de manière autonome. Parmi ces techniques, on compte les réseaux de neurones artificiels. C'est sur ces algorithmes que reposent l'apprentissage profond, mais aussi des technologies comme la reconnaissance d'images ou la vision robotique [10].

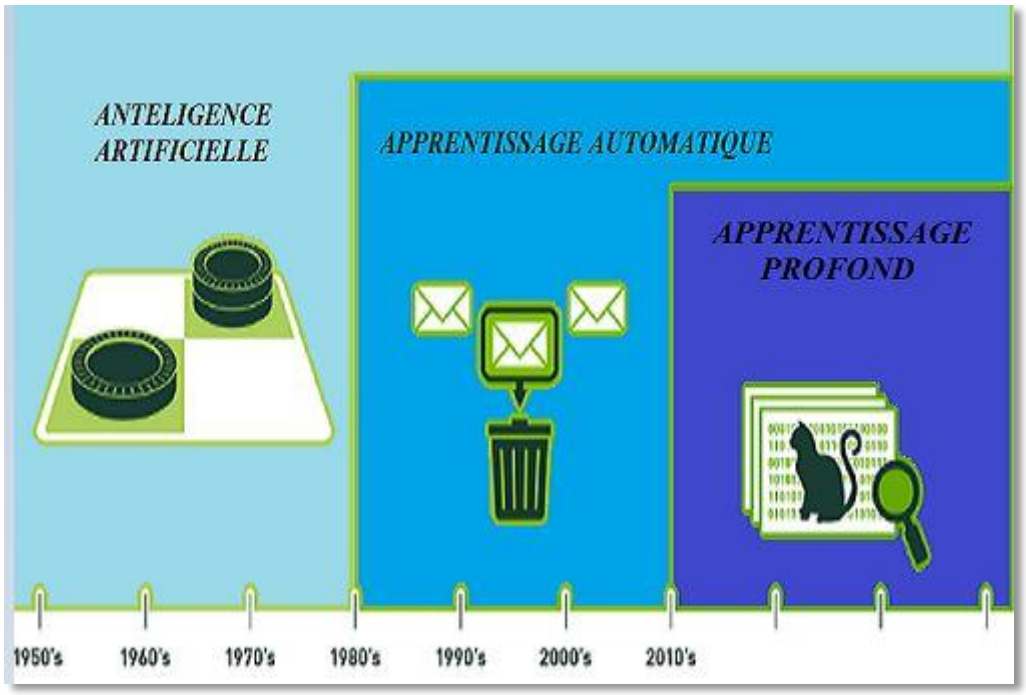


Fig 1.9 Evolution d'apprentissage profond [10].

III.2 Quelque définition de l'apprentissage profond

- ✓ L'apprentissage profond (deep learning) est un terme abrégé pour "apprentissage dans les réseaux de neurones profonds". Il s'agit des méthodes d'apprentissage automatique utilisant les réseaux de neurones profonds ; c'est donc un sous-domaine de l'apprentissage automatique (et un sous-sous-domaine de l'IA en général) [11].

- ✓ L'apprentissage profond est un ensemble d'algorithmes de machine learning cherchant à modéliser des abstractions de haut niveau au sein des données en utilisant des architectures de modèles composés de multiples transformations non linéaires [12].

- ✓ L'apprentissage profond (deep learning) est une forme d'intelligence artificielle, dérivée du machine learning (apprentissage automatique). Pour comprendre ce qu'est l'apprentissage en profondeur il convient donc de comprendre ce qu'est la machine learning [10].

III.3 Exemples d'application de l'apprentissage profond

Des millions de personnes profitent déjà des progrès réalisés grâce au l'apprentissage profond. Les champs d'action de cette technologie sont le traitement de l'image, de la vidéo, de la voix, et plus spécifiquement de la détection d'objets dans le domaine automobile (piétons, panneaux de signalisation, voitures, bus, marquage au sol...).

Des applications de l'apprentissage profond sont utilisées dans divers secteurs exemple :

- De nombreux secteurs d'activité exploitent des programmes de reconnaissance d'image : les sites marchands pour classifier, identifier et trouver des similitudes entre des produits ou des familles de produits.
- La robotique pour permettre par exemple à un robot-aspirateur de se déplacer en toute autonomie.
- Les réseaux sociaux et autres sites de partage de photos pour la reconnaissance faciale.
- Les moteurs de recherche pour détecter les similitudes entre des images.
- L'industrie automobile pour le développement des voitures autonomes.

- La médecine pour la recherche de cellules cancéreuses, etc.

III.4 Fonctionnement l'apprentissage profond

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

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

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

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

Une autre technique d'apprentissage est celle de «l'unsupervised learning », ou apprentissage non supervisé**.** Cette technique repose sur des données qui ne sont pas étiquetées. Les réseaux de neurones doivent reconnaître des patterns au sein des ensembles de données pour apprendre par eux-mêmes quels éléments d'une photo peuvent être pertinents [10].

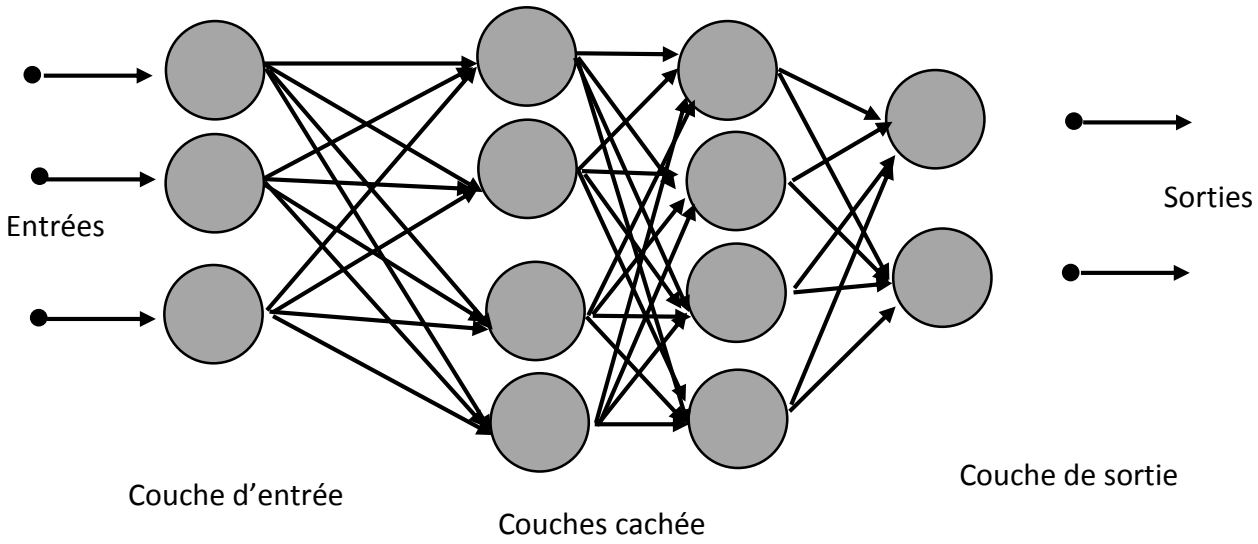


Fig 1.10 Réseau de neurone multicouche [10].

Les réseaux de neurones sont organisés en couches constituées d'un ensemble de nœuds interconnectés. Les réseaux peuvent être composés de plusieurs dizaines, voire plusieurs centaines de couches cachées.

III.5 Quelques algorithmes d'apprentissage profond

L'apprentissage profond consiste à apprendre plusieurs niveaux de représentation et d'abstraction qui aident à donner du sens à des données telles que des images, du son et du texte. Pour plus d'informations sur les algorithmes d'apprentissage profond.

Il existe différents algorithmes de l'apprentissage profond. Nous pouvons ainsi citer :

- **Les réseaux de neurones profonds** ((DNN) Deep Neural Networks). Ces réseaux sont : similaires aux réseaux MLP (multi-layer perceptron) mais avec plus de couches cachées. L'augmentation du nombre de couches, permet à un réseau de neurones de détecter de

légères variations du modèle d'apprentissage, favorisant le sur-apprentissage ou sur-ajustement (‘overfitting’).

- **Les réseaux de neurones convolutionnels** (CNN ou Convolutional Neural Networks). Le problème est divisé en sous parties, et pour chaque partie, un « cluster » de neurones sera créer afin d'étudier cette portion spécifique.

- **La machine de Boltzmann profonde** (Deep Belief Network) : Ces algorithmes fonctionnent suivant une première phase non supervisée, suivi de l'entrainement classique supervisé. Cette étape d'apprentissage non-supervisée, permet, en outre, de faciliter l'apprentissage supervisé. [13]

III.6 La différence entre l'apprentissage profond et l'apprentissage automatique

L'apprentissage profond est un sous-ensemble du l'apprentissage automatique et l'une des 15 approches différentes. Tout apprentissage profond est un apprentissage automatique mais pas tout apprentissage automatique est un apprentissage profond [14].

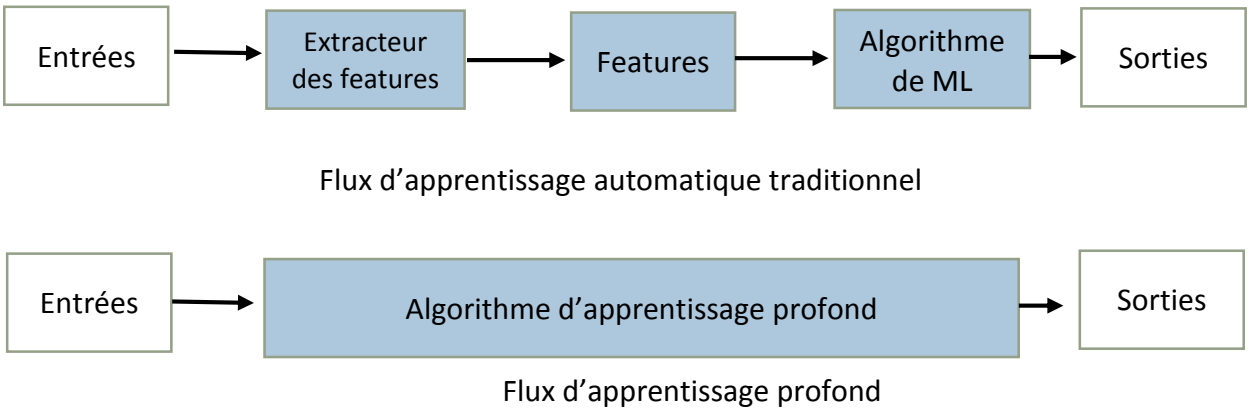


Fig 1.11 La différence entre l'algorithme de l'apprentissage profond et l'apprentissage automatique [14].

	L'apprentissage automatique	L'apprentissage profond
Volumes de données optimaux	Des milliers de points de données	Big Data millions de points de données
Les sorties	Valeur numérique, comme une classification ou un score	Quoi que ce soit des valeurs numériques aux éléments de forme libre, comme le texte libre et le son
Comment ça marche	Utilise différents types d'algorithmes automatisés qui apprennent à modéliser la fonction et à prédire les actions futures à partir des données.	Utilise des réseaux de neurones qui transmettent des données à travers de nombreuses couches de traitement pour interpréter les caractéristiques et les relations de données.
Comment cela est géré	Les algorithmes sont dirigés par des analystes de données pour examiner des variables spécifiques dans des ensembles de données.	Les algorithmes sont largement auto-orientés sur l'analyse des données une fois qu'ils sont mis en production.

Tableau 1.1 : La différence entre l'apprentissage profond et l'apprentissage automatique [15].

IV. Conclusion

Dans ce chapitre nous avons représentées trois parties ; l'intelligence artificielle, l'apprentissage automatique et l'apprentissage profond. Dans l'IA nous avons présentée quelques définitions, un historique et leur grand domaine. En suit, dans l'apprentissage automatique nous avons exposée quelques définitions, pourquoi l'apprentissage automatique ?, ses type et ses modèle. Enfin dans la partie d'apprentissage profond on a définie l'apprentissage profond, comment cela fonctionne et la différence entre l'apprentissage profond et l'apprentissage automatique. Dans le chapitre suivant, nous expliquerons les réseaux de neurones artificiels et les réseaux de neurones convolutionnels.

I. Introduction

Les réseaux neuronaux artificieux (également appelés réseaux neuronaux) sont inspirés par le cerveau humain, ils sont des systèmes de traitement de données composés de nombreuses petites unités de traitement. Chacune de ces unités de traitement est similaire à un neurone artificiel. Habituellement, beaucoup de neurones artificiels sont connectés les uns aux autres dans une structure stratifiée hiérarchique. De plus, pour modéliser mathématiquement le taux de déclenchement de chaque neurone, une fonction d'activation est utilisée sur la sortie de chaque neurone. Ce chapitre est divisé en deux parties, la première expose les réseaux de neurones qui sont un type populaire de modèle d'apprentissage automatique et la deuxième partie présente les réseaux de neurones convolutionnels (CNN).

II. Les réseaux de neurones

II.1 Définition

Les réseaux de neurones artificiels sont des réseaux fortement connectés de processeurs élémentaires fonctionnant en parallèle. Chaque processeur élémentaire calcule une sortie unique sur la base des informations qu'il reçoit. Toute structure hiérarchique de réseaux est évidemment un réseau [2].

II.2 Historique

Les premières formulations du réseau de neurones remontent à 1943, dans les travaux de Mc Culloch et Pitts. L'idée est de reproduire le fonctionnement d'un neurone humain, comme représenté sur la Fig2.1

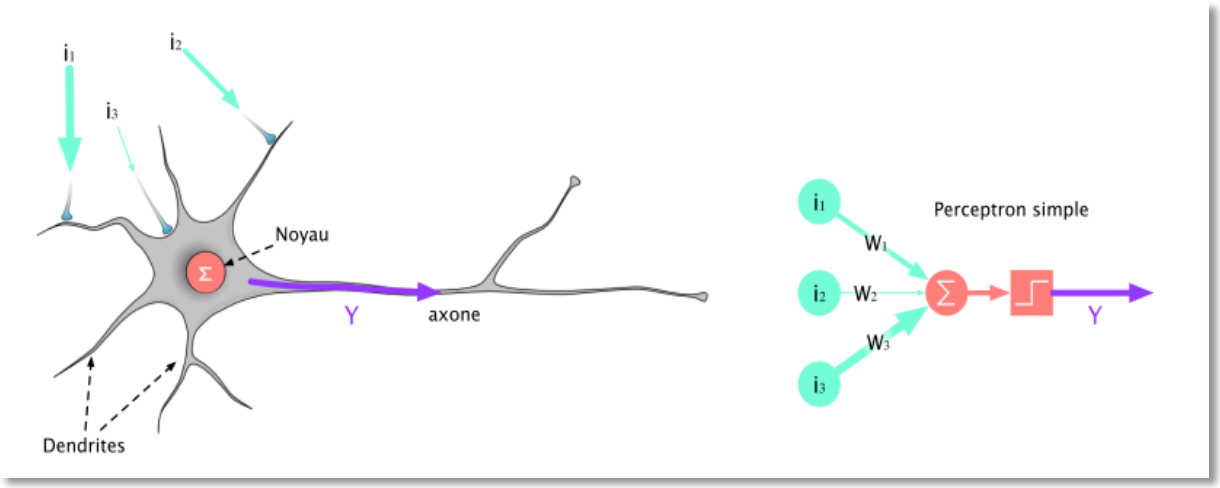


Fig 2.1 Schéma d'un neurone à gauche et représentation d'un neurone formel à droite. [16]

II.3 Neurone formel

Le neurone formel est une modélisation mathématique qui reprend les principes du fonctionnement du neurone biologique, en particulier la sommation des entrées. Sachant qu'au niveau biologique, les synapses n'ont pas toutes la même «valeur» (les connexions entre les neurones étant plus ou moins fortes), les auteurs ont donc créé un algorithme qui pondère la somme de ses entrées par des poids synaptiques (coefficients de pondération). De plus, les 1 et les -1 en entrée sont là pour figurer une synapse excitatrice ou inhibitrice [2].

Le fonctionnement d'un neurone formel est simple : c'est une somme pondérée d'entrées à laquelle on applique une fonction d'activation. Les coefficients de pondération sont appelés poids synaptiques et la fonction d'activation utilisée était initialement un seuil : la sortie valant 1 si la somme pondérée dépasse le seuil et 0 sinon. On note *b* la valeur du seuil [16].

$$Y = \begin{cases} 1 \text{ si } \sum_k w_k i_k > b \\ 0 \text{ sinon} \end{cases} \qquad (2.1)$$

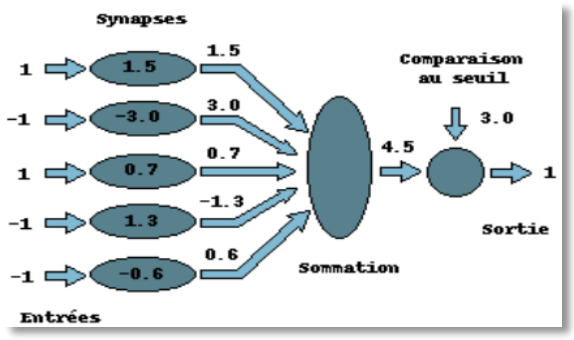


Fig 2.2 Neurone formel. [2]

On peut ensuite utiliser plusieurs neurones en parallèle, on obtient donc un réseau avec un vecteur de sortie comportant autant d'éléments que de neurones. Ce système, baptisé percetron, constitue l'un des premiers systèmes artificiels capables d'apprendre sur des exemples. Frank Rosenblatt proposa une méthode pour optimiser les poids synaptiques. Mais quelques années plus tard, en 1969, Marvin Lee Minsky et Seymour Papert ont publié un livre appelé <Perceptrons> dans lequel ils mettent en avant les limites du modèle. En effet, appliqué un seuil sur un vecteur pondéré revient à découper linéairement l'espace des vecteurs d'entrée en deux avec un hyperplan. Ce travail trop rapidement généralisé à tous les types de réseaux de neurones, a considérablement réduit les efforts dans le domaine qui n'a pas vu d'avancée majeure pendant une dizaine d'années [16].

II.4 Perceptron

Inspiré par une vision biologique (en particulier par les neurones), Rosenblatt a inventé le perceptron [2]. Chaque neurone dans le cerveau est connecté à d'autres neurones par des connexions synaptiques. Quand un neurone reçoit un stimulus, il peut tirer (c'est-à-dire envoyer un signal à d'autres neurones) ou ne rien faire. Le comportement perceptron est similaire à un neurone, il est connecté à de nombreuses entrées avec des poids (qui symbolisent la synapse) et il a une fonction qui détermine si le perceptron devrait envoyer un signal ou non à ces entrées. Nous appelons cette fonction une fonction d'activation. La Fig 2.3 est un exemple de perceptron qui reçoit en entrée un vecteur x = x1, x2, et donne une sortie o après avoir été calculée par une fonction f. La sortie d'un perceptron peut être exprimée comme :

$$O = f(\sum_{i=0}^{n} x_i \, w_i \,) \tag{2.2}$$

Avec :

x : Qui représente les entrées,

wi : Les poids synaptiques.

f : Une fonction d'activation qui décide si le neurone doit tirer ou non ou "combien" il doit tirer. [17].

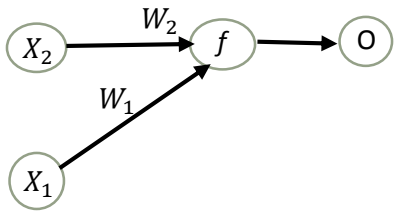


Fig 2.3 Un modèle de perceptron. [2]

II.5 Perceptron multicouche, algorithme de back propagation

Un nouveau type de réseau permettant de dépasser les limites du simple perceptron a été proposée dans les années 80 : le perceptron multicouche (Multi Layer Perceptron) (MLP). En réalité, cette formulation date de 1969 mais a été réellement utilisée à partir de 1986 grâce à l'article de Rumelhart et Hinton. Le principe est de superposer plusieurs perceptrons dont la fonction d'activation est la fonction sigmoïde.

L'algorithme d'entrainement des MLP est communément appelé algorithme de back-propagation (ou rétropropagation du gradient). C'est une descente de gradient sur la fonction d'erreur quadratique entre la sortie et la sortie désirée. Cet entraînement est supervisé, il nécessite donc de connaître pour chaque vecteur d'entrée, un vecteur de sortie désirée. Pour une tâche de classification par exemple, cela revient à connaître la classe du vecteur d'entraînement [16]. Il se déroule en deux étapes (voir la Fig 2.4) :

Rétropropagation : Le gradient en fonction de chaque paramètre se calcule assez facilement grâce à une propriété intéressante de la fonction sigmoïde : sa dérivée s'exprime en fonction d'elle-même.

$$\text{sigm0}(x) = \text{sigm}(x)(1 - \text{sigm}(x)) \tag{2.3}$$

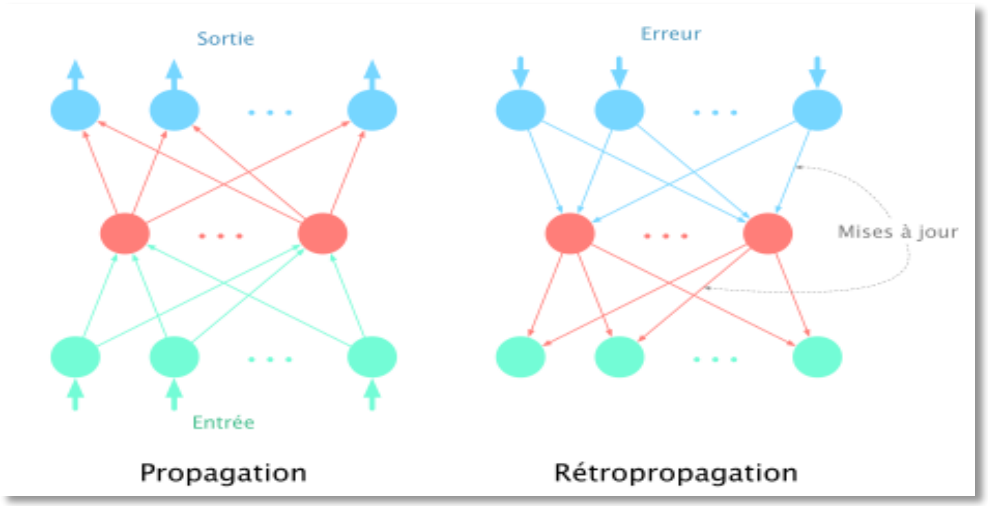


Fig 2.4 Schéma des deux phases de l'algorithme de rétropropagation. [16]

1. La propagation du vecteur d'entrée, c'est à dire le calcul du vecteur de sortie du réseau en fonction du vecteur d'entrée.
2. La rétropropagation du gradient, c'est à dire le calcul du gradient de la fonction d'erreur quadratique par rapport à chaque paramètre du réseau puis la mise à jour de ces paramètres.

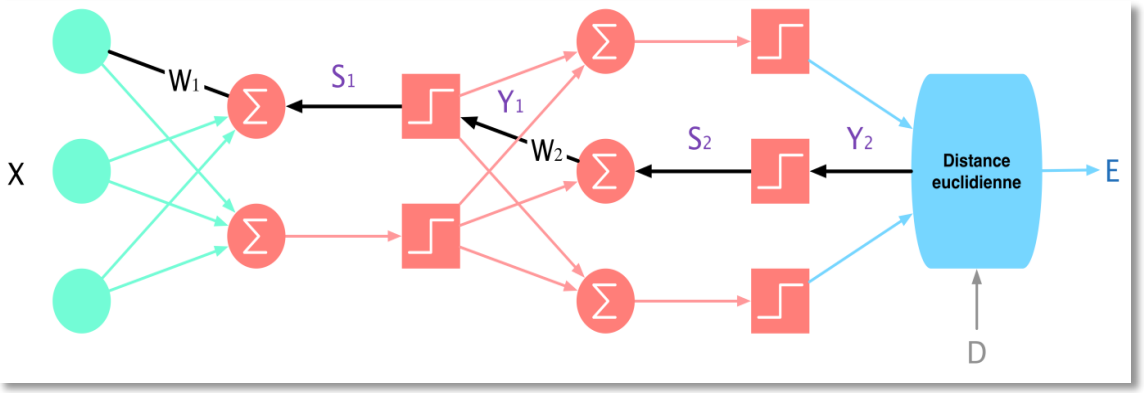


Fig 2.5 Phase de rétropropagation détaillée. [16]

II.6 Les limite du perceptron multicouche

Une des limitations que rencontre le MLP est que si l'architecture est trop profonde, l'optimisation des paramètres mène bien souvent à des minimas locaux non optimaux. En effet, comme expliqué dans la surface d'erreur n'est pas convexe et est d'autant plus irrégulière que réseau est profond. Par ailleurs, il semblerait que l'entrainement des couches les plus basses (proches du vecteur visible) soit peu efficace dans un MLP profond. L'intuition est que l'information de la mise à jour des paramètres est de moins en moins pertinente à mesure que l'on s'enfonce dans les couches les plus basses [16].

II.7 Les algorithmes d'apprentissage

7.1. L'algorithme d'apprentissage de Hebb

Modifie de façon itérative (petit à petit) les poids pour adapter la réponse obtenue à la réponse désirée. Il s'agit en fait de modifier les poids lorsqu'il y a erreur seulement [2].

1. Initialisation des poids et du seuil S à des valeurs (petites) choisies au hasard.
2. Présentation d'une entrée $E_1 = (e_1, ...e_n)$ de la base d'apprentissage.
3. Calcul de la sortie obtenue x pour cette entrée :
   $a = \sum(w_i . e_i) - S$ (La valeur de seuil est introduite ici dans le calcul de la somme pondérée)
   x = signe (a) (si a > 0 alors x = +1 sinon a ≤ 0 alors x = -1)
4. Si la sortie x est différente de la sortie désirée d_1 pour cet exemple d'entrée E_1 alors modification des poids (μ est une constante positive, qui spécifie le pas de modification des poids) :
   w_{ij}(t+1) = w_{ij} (t) + μ. ($x_i . x_j$)
5. Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement (Modification des poids), retour à l'étape 2.

7.2. L'algorithme d'apprentissage du perceptron

L'algorithme d'apprentissage du perceptron est semblable à celui utilisé pour la loi de Hebb. Les différences se situent au niveau de la modification des poids [2].

1. Initialisation des poids et du seuil S à des valeurs (petites) choisies au hasard.
2. Présentation d'une entrée $E_1 = (e_1, ...e_n)$ de la base d'apprentissage.
3. Calcul de la sortie obtenue x pour cette entrée :
   $a = \sum(w_i . e_i) - S$
   x = signe (a) (Si a > 0 alors x = +1 sinon a ≤ 0 alors x = -1)
4. Si la sortie x du Perceptron est différente de la sortie désirée dl pour cet exemple d'entrée E_1 alors modification des poids (μ le pas de modification) :
   w_i (t+1) = w_i (t) + μ. ((d_1 - x).ei)
   Rappel : d_1 = +1 si E est de la classe 1, d_1 = -1 si E est de la classe 2 et (d_1 - x) est une estimation de l'erreur.
5. Tant que tous les exemples de la base d'apprentissage ne sont pas traités correctement (modification des poids), retour à l'étape 2.

II.8 Applications de réseaux de neurones

- Statistiques : Analyse de données / Prévision / Classification.
- Robotique : Contrôle et guidage de robots ou de véhicules autonomes.
- Imagerie / Reconnaissance de formes.
- Traitement du signal.
- Simulation de l'apprentissage.

III. Les réseaux de neurone de convolution CNN

Les réseaux de neurones convolutionnels ou CNN pour « Convolutional Neural Network » sont une extension des MLP permettant de répondre efficacement aux principaux défauts des MLP. Ils sont conçus pour extraire automatiquement les caractéristiques des images d'entrée, sont invariants à de légères distorsions de l'image, et implémentent la notion de partage des poids permettant de réduire considérablement le nombre de paramètres du réseau. Ce partage des

poids permet en outre de prendre en compte de manière forte les corrélations locales contenues dans une image. Les réseaux de neurones convolutionnels ont initialement été inspirés par la découverte faite par Hubel et Wiesel de neurones sensibles aux aspects locaux et sélectifs en orientation dans le système visuel du chat. La première utilisation des réseaux de neurones convolutionnels a été réalisée par Fukushima. Les poids sont forcés à être égaux pour détecter des lignes, des points ou des coins à tous les endroits possibles de l'image, implémentant de fait l'idée du partage des poids.

Une avancée importante a été effectuée par Y. Lecun et al. Avec l'utilisation d'un réseau de neurones convolutionnels dont l'apprentissage a été réalisé par propagation arrière (backpropagation). Ce modèle a notamment été appliqué avec succès pour la reconnaissance de caractères manuscrits. Le premier réseau de neurones convolutif (CNN) a été introduit à la fin des années 80 par LeCun 1989. C'est le premier réseau de neurones pour la reconnaissance d'images. Ce réseau permettait la reconnaissance de chiffres manuscrits [18].

III.1 Les Différents modules d'un réseau de neurones convolutif

Nous présentons dans cette partie les différents modules utilisés dans les CNN : les convolutions, l'agglomération (pooling), les fonctions d'activation, le dropout et la couche Fully Connected.

1.1 La couche de convolution

La couche de convolution est la composante clé du CNN, qui applique un nombre spécifié de filtres de convolution à l'image en entrée. Cette couche permet d'effectuer un ensemble d'opérations mathématiques pour produire une valeur unique à la sortie. La sortie en résultant est appelée carte de caractéristiques.

La formule 2.4 permet d'expliquer comment calculer la sortie d'un neurone donné dans une couche de convolution [19].

$$g_{i,j,k=b_k} + \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{k\prime=1}^{f_{n\prime}} x_{i\prime,j\prime,k\prime} \cdot w_{u,v,k',k} \; avec \begin{cases} i' = u * s_h + f_h - 1 \\ j' = v * s_w + f_w - 1 \end{cases} \qquad (2.4)$$

- $g_{i,j,k}$ correspond à la sortie du neurone situé en ligne i et en colonne j dans la carte de caractéristique k de la couche de convolution (couche l).
- s_h et s_w sont les pas vertical et horizontal, f_h etf_wSont la hauteur et la largeur du champ récepteur, et f_n, est les nombres de cartes de caractéristique dans la couche précédente (couche l-1).
- $x_{i',j',k}$, correspond à la sortie du neurone situe dans la couche l-1, ligne i', colonne j', carte de caractéristique k'(ou canal k' si la couche précédente est la couche d'entrée).
- b_k est le terme constant de la carte de caractéristique k (dans la couche l). il peut être vu comme un réglage de la luminosité globale de la carte de caractéristique k.
- $w_{u,v,k',k}$ correspond au poids de la connexion entre tout neurone de la carte de caractéristique k de la couche l et son entrée située en ligne u et colonne v (relativement au champ récepteur de neurone) dans la carte de caractéristiques k'.

La fonction de convolution permet de construit une couche convolutionnel bidimensionnelle. Prend le nombre de filtres, la taille du noyau de filtre, le remplissage et la fonction d'activation comme arguments.

Exemple de calcule : 1*1+1*0+1*1+0*0+1*1+1*0+0*1+0*0+1*1=4

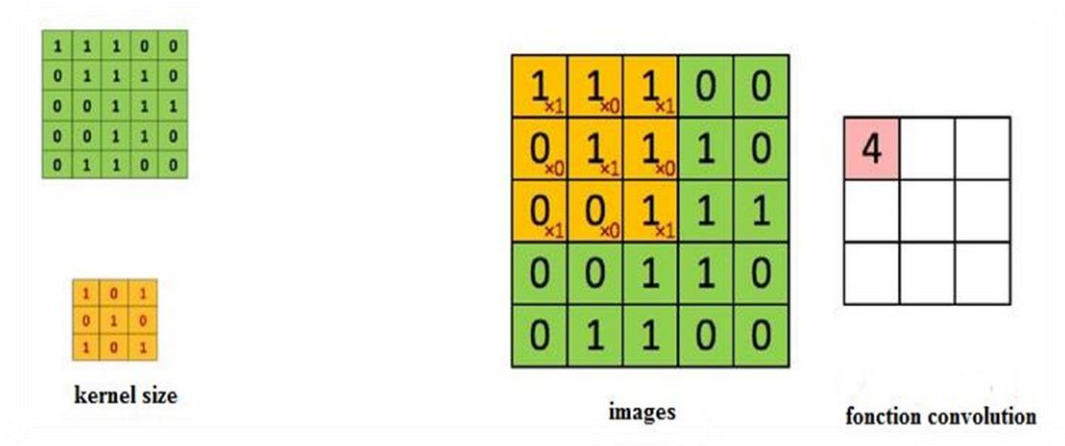


Fig 2.6 Une illustration d'une couche de convolution.

1.2 La couche pooling

La couche pooling (ou agglomération) permet de rajouter de l'invariance spatiale lors de l'extraction de caractéristiques tout en réduisant la dimension des entrées. Elle peut être de différentes natures mais les types de *pooling* les plus utilisés sont le *Max Pooling* (illustré dans la Fig 2.7) et l'*Average pooling*. Le *Max Pooling* renvoie l'élément maximum sur une fenêtre de calcul. L'*Average pooling* permet de renvoyer la moyenne des éléments sur une fenêtre de calcul [18].

1.3 La couche Maxpooling

La couche Maxpooling est généralement utilisée après la couche convolutionnel. Elle est utilisé pour réduire les dimensions d'une image afin de diminuer le temps de calcule et minimiser l'espace mémoire occupé.

La couche maxpooling permet de construit une couche de regroupement bidimensionnelle en utilisant l'algorithme *max-pooling*. Prend la taille du filtre de *pooling* et avance comme arguments.

La formule 2.5 ci-dessous illustre le calcul de la taille de Maxpooling [20].

$$W_2 = \frac{W_1 - F}{S} + 1$$

$$H_2 = \frac{H_1 - F}{S} + 1 \qquad (2.5)$$

W1, H1 : La taille du volume d'entrée.

F : La taille spatiale du volume de sortie.

S : Le pas.

W2, H2 : La taille du volume de sortie.

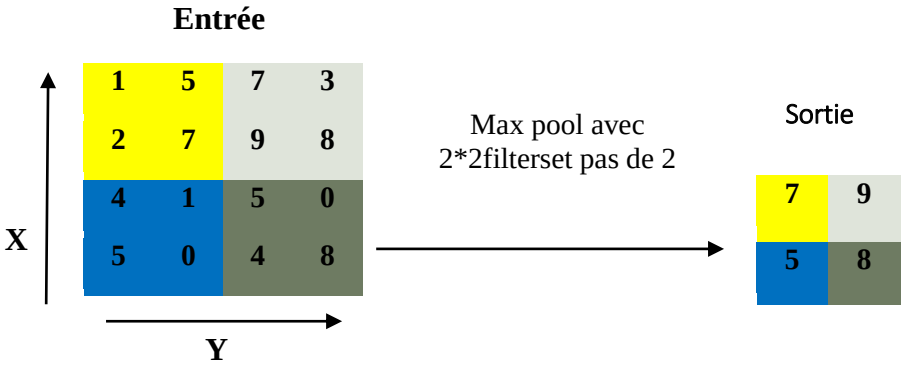


Fig 2.7 Une illustration d'une couche de Maxpooling. [18]

1.4 Les fonctions d'activation

Il existe différentes fonctions d'activation. Parmi les plus connues :

1.4.1 Fonction d'activation ReLU

La fonction d'activation (ReLU) est utilisée presque dans tous les réseaux de neurones convolutionnels et dans les méthodes d'apprentissage profond. C'est une fonction élémentaire qui est généralement s'exécutée selon deux cas : le premier cas ; la fonction est désactivée si les entrées sont négatives (la sortie est nulle), le deuxième cas où les entrées sont positives donc la sortie est la même que l'entrée. Elle est utilisée pour gagner la non-linéarité du réseau [21]. Comme illustré par la formule (2.6) suivante :

$$\boldsymbol{f(x)} = \begin{cases} \mathbf{0} \ \ pour\ x < 0 \\ \boldsymbol{x} \ \ pour\ x \geq 1 \end{cases} \tag{2.6}$$

Avec :

f: La fonction d'activation.

x : La valeur du neurone.

1.4.2 Fonction d'activation Sigmoïde

La fonction d'activation Sigmoïde est le choix le plus approprié, Elle est particulièrement utilisée pour les modèles, où il faut prévoir des possibilités en sortie.

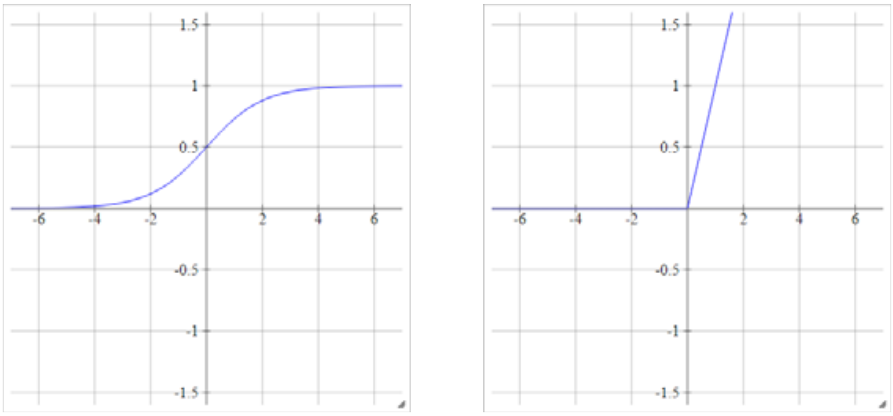
Dans notre travail on a choisi cette fonction pour assurer la bonne classification (véhicule, non véhicule), son propriété est : pour tout x réel, $\boldsymbol{y} \in \boldsymbol{R}: \mathbf{0} < y < 1$ telle que y est la sortie de la fonction sigmoïde [22] la formule (2.7).

$$\boldsymbol{f(x) = sigmoid(x) = \frac{1}{1+e^{-x}}} \tag{2.7}$$

Où :

f : La fonction d'activation.

x : La valeur du neurone.



(a) (b)

Fig 2.8 Deux fonctions d'activation. *(a)* la sigmoïde, *(b)* la fonction ReLU [22].

1.4.3 Fonction d'activation Softmax

Softmax est un autre type de fonction d'activation plutôt différent, ce qui normalise un ensemble de pré-activations de telle sorte que chacune puisse être interprétée comme une probabilité.

Habituellement, dans les problèmes de classification, s'il n'y a que deux classes, une seule unité Sigmoïde est utilisée. Cependant, s'il y a plus de deux classes, le Softmax est utilisé.
Considérant x comme un vecteur de pré-activations C, le Softmax sur ces classes C est défini comme suit [23] la formule (2.8).

$$\boldsymbol{f(x)} = \frac{\boldsymbol{e^x}}{\sum_{\boldsymbol{i=1}}^{\boldsymbol{c}} \boldsymbol{e^i}} \tag{2.8}$$

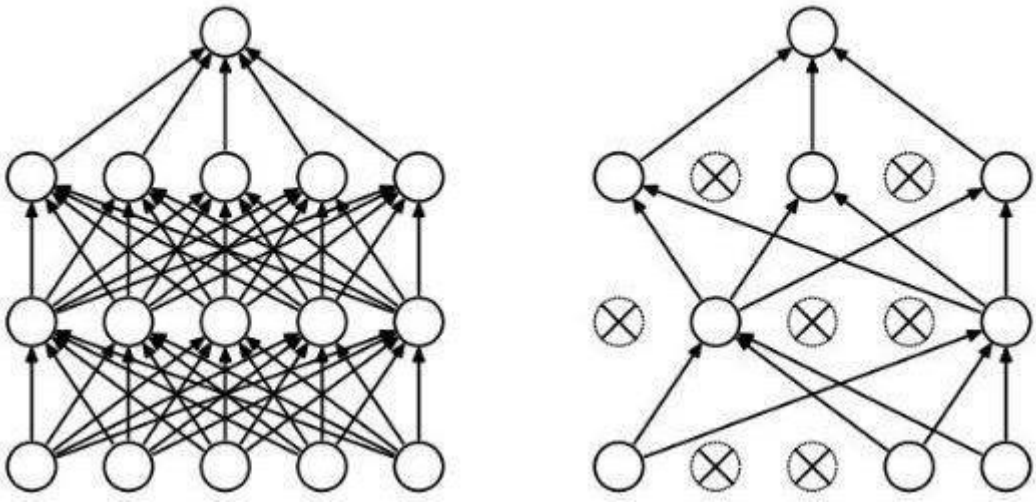
Où :

f: La fonction d'activation.

x : La valeur du neurone.

1.5 Le dropout

La couche de dropout a été introduite par [24]. Elle est introduite pour éviter le sur-apprentissage (*overfitting*), cette couche est utilisée pendant l'apprentissage. Elle permet de désactiver aléatoirement des neurones durant les différentes itérations de l'apprentissage. En d'autres termes, le dropout permet au réseau d'apprendre des sous-réseaux contenant moins de paramètres et donc moins sujets au sur-apprentissage. Cette manière de faire permet d'apprendre des paramètres plus génériques qui ne se focalisent pas sur des détails de la base d'apprentissage. Une fois l'apprentissage terminé, tous les neurones sont réactivés [18].



(a) Réseau neuronal standard. (b) Après l'application de dropout.

Fig 2.9 Une structure de réseau de neurones avant et après l'application de dropout. [24]

1.6 Couche fully connected (FC)

Après plusieurs couches de convolution et de Maxpooling, le raisonnement de haut niveau dans le réseau neuronal se fait via des couches entièrement connectées. Les neurones dans une couche entièrement connectée ont des connexions vers toutes les sorties de la couche précédente. Leurs fonctions d'activations peuvent donc être calculées avec une multiplication matricielle suivie d'un décalage de polarisation.

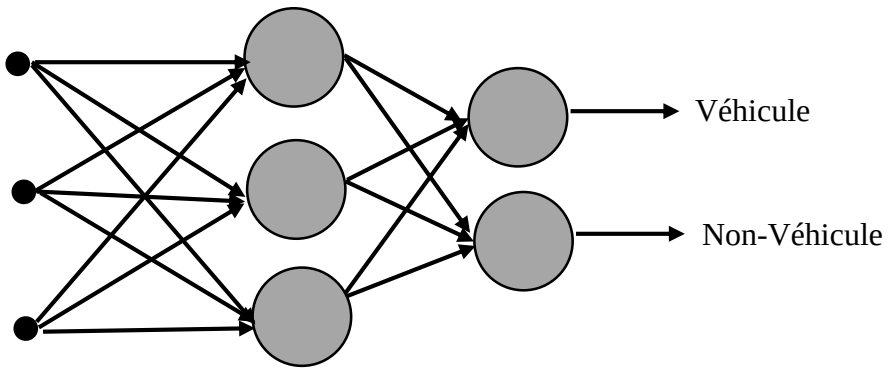


Fig 2.10 Architecture de la couche fully connected.

IV. Conclusion

Nous avons présenté dans ce chapitre un historique sur les réseaux de neurones, la définition des réseaux de neurone, aussi les réseaux de neurone convolutionnels et ses différents modules utilisés (la couche de convolution, les fonctions d'activation, la couche pooling, la couche dropout, et fully connected). Nous montrerons dans le chapitre suivant la définition de détection d'objets on général et les techniques de détection de véhicules on spécifique.

I. Introduction

Ce chapitre a pour l'objectif de présenter les techniques de détection d'objets on général et les techniques de détection de véhicules on spécifique, les défis de la détection, le principe général de la détection, les différentes approches de détection d'objets comme YOLO, R-CNN.

II. Détection d'objets

La détection d'objets est une des problématiques les plus étudiées en vision par ordinateur. Son objectif est de trouver dans une image d'entrée, des régions (boites englobantes) contenant des objets. On peut diviser la détection d'objets en deux catégories : les détecteurs à une classe et les détecteurs multi-classes. Un détecteur à une classe se concentre sur la détection d'un seul type d'objet. Le détecteur doit être capable de décider si une région de l'image correspond à l'objet ou à du fond. Dans le cadre d'un détecteur multi-classes, le détecteur doit être capable de séparer le fond et les objets, tout en séparant les objets entre eux (c'est-à-dire décider de quel type d'objets il s'agit). Il existe de nombreuses bases de données publiques permettant d'entraîner et d'évaluer des détecteurs [25]. La Fig 3.1 illustre l'objectif de la détection d'objets.

Fig 3.1 Exemples de détections d'objets Renvoyées par un détecteur multi-classes.[25]

Chaque couleur de boite englobant correspond à une classe d'objets [26].

II.1 Les défis de la détection

Un détecteur d'objets doit faire face à plusieurs difficultés :

- La première est le temps de calcul : un bon détecteur est un détecteur qui maximise les performances tout en étant le plus rapide possible à exécuter. Cette notion de temps de calcul est particulièrement importante dans la conception de détecteurs multi-classes de par le plus grand nombre de classes d'objets à détecter.
- La seconde concerne l'apparence variable des objets. Celle-ci peut varier suivant plusieurs facteurs : l'occultation, le point de vue sous lequel les objets sont observés et la taille des objets dans l'image. Cette variabilité d'apparence est également une difficulté pour les systèmes de classification d'images.
- Une troisième difficulté est en lien avec la variation d'apparence des régions qui ne correspondent pas à des objets (le fond). Un détecteur doit être capable de décider si une région correspond à du fond, et cela, même dans des images provenant d'environnements complexes. [25]

II.2 Le principe général de la détection

Un détecteur consiste en un modèle permettant de discriminer des régions dans l'image. Il peut être appris, par exemple, grâce à l'apprentissage supervisé. La détection consiste à extraire des régions dans l'image d'entrée et à leur appliquer ce modèle. Cette opération permet de renvoyer un score de confiance sur la région extraite modélisant la probabilité que la région contienne ou pas un objet. La manière d'extraire les régions est discriminante pour la rapidité et l'efficacité du détecteur : si le modèle est appliqué à toutes les régions possibles dans l'image, le temps de calcul devient énorme. Il y a deux techniques standards pour l'extraction des régions sur lesquelles va être appliqué le modèle. En particulier, le fonctionnement de la détection par fenêtre glissante (sliding window) et de la détection par proposition d'objets [25].

2.1 Détection par fenêtre glissante

Les approches de type fenêtre glissante utilisent un modèle de classification préalablement appris qui consiste en une fenêtre 2D de taille fixe permettant de discriminer le fond des objets. L'idée générale de ce type d'approche est d'appliquer cette fenêtre sur toutes les positions de l'image d'entrée à tester, d'où l'appellation "glissante". Autrement dit, cette fenêtre vient parcourir l'image et produit en chaque position de celle-ci un score de confiance.

- Étant donné une image en entrée, une pyramide d'image est calculée. Ce calcul consiste à redimensionner l'image en utilisant plusieurs facteurs d'échelle et plusieurs ratios largeur/hauteur. L'ensemble de ces images forme la pyramide et chaque image correspond à un niveau de celle-ci. Il est nécessaire de calculer une pyramide d'images car la fenêtre à appliquer sur l'image est de taille fixe : le modèle de détection est généralement appris pour une seule échelle et un seul ratio largeur/hauteur. Or, les objets présents dans une image peuvent être de différentes tailles et présenter des ratios variables. De cette manière, les objets présents dans l'image correspondent à la taille du modèle au moins pour un niveau de la pyramide.

- Des caractéristiques visuelles sont extraites sur les différents niveaux de la pyramide ce qui permet d'obtenir une carte de caractéristiques pour chaque niveau pyramidal.

- Un modèle préalablement appris est appliqué en chaque position des différentes cartes de caractéristiques, renvoyant un score de confiance pour chaque position et chaque niveau de la pyramide. Cette étape permet de sélectionner des boites candidates (celles avec les plus hauts scores de confiance).

- Dans les systèmes de détection, il est très fréquent que des boites candidates soient agglomérées autour du même objet. En d'autres termes, plusieurs boites peuvent correspondre au même objet. Pour supprimer ces détections redondantes, un algorithme appelé suppression des non-maxima (NMS) est appliqué. L'idée de cet algorithme repose sur le fait que des détections ne peuvent pas se recouvrir spatialement au-delà d'un certain seuil (le seuil de recouvrement, *overlap* en anglais). Si des détections se

recouvrent de manière trop importante, la détection avec le meilleur score de confiance est gardée, les autres sont supprimées [25].

Le schéma par fenêtre glissante a été largement utilisé pour la détection d'objets. Les différentes méthodes qui l'utilisent se différencient par la nature du modèle à appliquer sur la pyramide et les caractéristiques visuelles utilisées (caractéristiques de formes [28,27], HOG [29,30], caractéristiques profondes [31, 32]...)

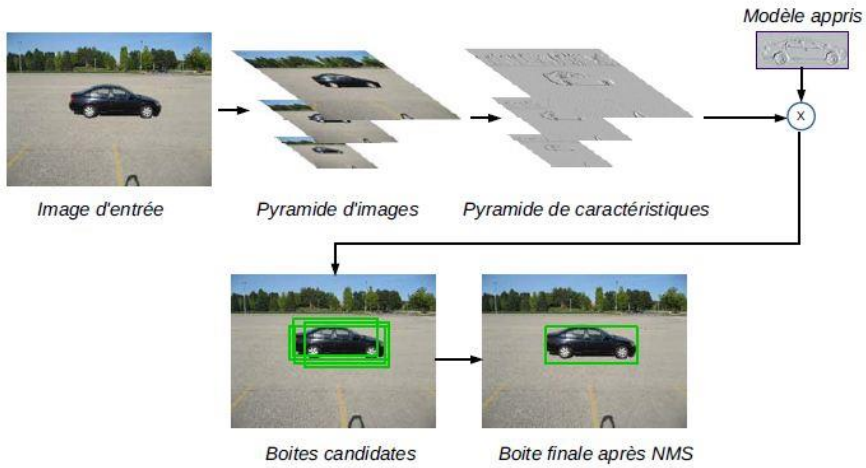


Fig 3.2 –Schéma classique de détection d'objets par fenêtre glissante. Une pyramide d'images est créée à partir d'une image d'entrée. [25]

Des caractéristiques visuelles sont calculées sur cette pyramide d'image. Un modèle est appliqué en chaque position et chaque niveau de pyramide renvoyant des boites candidates. Une suppression des non maxima est alors appliquée (NMS) pour supprimer les boites correspondant au même objet.

La fenêtre glissante est devenue incontournable, notamment après la publication de [33]. Les auteurs y introduisent une approche basée sur le principe du *boosting* [34]. L'idée est de scanner l'image avec des classifieur dits "faibles" appelés filtres de Haar. La somme des réponses de ces classifieurs faibles permet la prise de décision du détecteur. Cette méthode a été longtemps considérée comme l'état de l'art en détection de visages. L'avantage majeur de celle-ci est le

temps de calcul. En effet, les caractéristiques renvoyées par les filtres de Haar sont très rapides à calculer notamment grâce aux images intégrales.

De plus, dans cette méthode, les classifieurs faibles sont utilisés en cascade. En d'autres termes, ils permettent, dans un premier temps, de supprimer rapidement des régions qui ne contiennent pas d'objets. Pour les régions les plus problématiques, l'agrégation des classifieurs faibles permet de construire des classifieurs de plus en plus robustes. Des extensions basées sur le *boosting* ont également été introduites notamment pour résoudre le problème de détection multi-classes [35, 36]. Ces approches proposent de partager des caractéristiques visuelles entre les classes d'objets à détecter. En 2005, les auteurs de [29] proposent de nouvelles caractéristiques visuelles : les HOG. Ces caractéristiques, basées sur les gradients de l'image ont permis de faire un bond en avant dans les performances des systèmes de détection d'objets. Les auteurs proposent d'utiliser les HOG et les séparateurs à vaste marge (SVM) pour séparer les exemples d'apprentissage dans l'espace des caractéristiques.

Les HOG et les SVM ont également été utilisés dans l'approche DPM [30] (*Deformable Part models*) où le modèle de détection est basé sur des représentations locales et globales des objets. Cette méthode est restée quelques années à l'état de l'art en détection de personnes. Des approches basées fenêtre glissante et CNN ont également été introduites. Parmi elles, on peut citer [37] et plus récemment [31, 32]. Dans [31], les auteurs proposent de transformer un réseau de neurones standard en remplaçant les couches complètement connectées par des couches convolutives. Cela permet l'application du réseau de neurones sur n'importe quelle taille d'image. Cette manière de faire est très intéressante notamment pour passer dans le réseau différents niveaux d'une pyramide d'images. Le réseau complètement convolutif est entraîné pour renvoyer des scores de confiance pour chaque classe d'objets ainsi que les quatre coins de leur boite englobante. Dans [32], les auteurs proposent d'apprendre un réseau de régression permettant de renvoyer les masques des boites englobantes des objets.

2.2 Détection par proposition d'objets

Une alternative à la recherche exhaustive des objets (la fenêtre glissante) est l'utilisation d'algorithmes de proposition d'objets. Récemment, la proposition d'objets a permis d'améliorer

les performances et le temps d'exécution dans les systèmes de détection d'objets. L'objectif de ces méthodes est de proposer des boites avec une forte probabilité d'être un objet d'intérêt. Ces boites sont ensuite extraites et envoyées à un classifieur pour la décision finale. Ces méthodes réduisent le temps d'exécution car elles diminuent considérablement l'espace de recherche par rapport aux approches de recherche exhaustive de type fenêtre glissante : le modèle permettant de détecter les objets n'est pas appliqué sur toutes les positions de l'image (ni sur tous les niveaux de la pyramide) mais seulement sur un ensemble réduit de régions. Dans ce qui suit, nous présentons les méthodes existantes permettant la génération de propositions d'objets. [25]

Un des algorithmes pour la proposition d'objets 2D est la *selective search* introduit par [38]. Comme d'autres approches [39, 40], cette méthode se base sur une segmentation de l'image à différentes résolutions. En utilisant la méthode de segmentation introduite par [41], la *selective search* segmente l'image d'entrée sur plusieurs échelles. Cela produit un premier ensemble de régions d'intérêts. Les auteurs de [38] introduisent ensuite un calcul de similarité entre régions basé sur des informations de couleurs, de textures, de tailles et d'inclusions. Cette similarité permet de fusionner les régions redondantes (trop similaires) et de renvoyer un ensemble de propositions d'objets pertinent. La Fig 3.3 illustre l'algorithme de *selective search*. Cette méthode a été utilisée par la suite dans deux travaux de références pour la détection d'objets par réseau de neurones convolutif : le RCNN [42] et le Fast- RCNN [43]. Dans le RCNN [42], les propositions d'objets provenant de la *selective search* sont extraites dans l'image et redimensionnées à taille fixe. Ces régions sont ensuite envoyées à un CNN pour déterminer leur classe. Cette approche est très coûteuse en temps de calcul (pour l'apprentissage et pour le test) car chaque région passe dans toutes les couches du CNN. C'est pour cette raison que le Fast-RCNN [43] a été introduit. Il permet d'extraire les régions provenant de la *selective search* sur une carte de caractéristiques profonde. En d'autres termes, l'image d'entrée entière est passée dans un réseau de neurones convolutif fournissant une carte de caractéristiques à basse résolution (du fait des *Pooling* successifs). Les propositions d'objets sont alors extraites sur cette carte et envoyées à un classifieur consistant généralement en deux couches cachées (complètement connectées) et une couche de sortie permettant la classification de l'objet (classe de l'objet ou fond). Dans ces deux approches, une fonction supplémentaire est apprise par le réseau permettant de transformer les propositions d'objets originelles produites par la *selective search* afin que celles-ci collent au mieux à l'objet. Cette fonction est appelée régression sur les

boites. Afin d'encore réduire le temps de calcul et gagner en performances, le réseau de proposition d'objets [44] (*Region Proposal Network* RPN) a été introduit. Les auteurs proposent la création d'un seul CNN capable de proposer des objets d'intérêt, de les extraire sur une carte de caractéristiques et de classifier chaque région. Cette méthode pour la détection d'objets, a été largement utilisée et modifiée [45, 44, 46, 47] de par ses très bonnes performances pour la détection multi-classes et sa rapidité d'exécution.

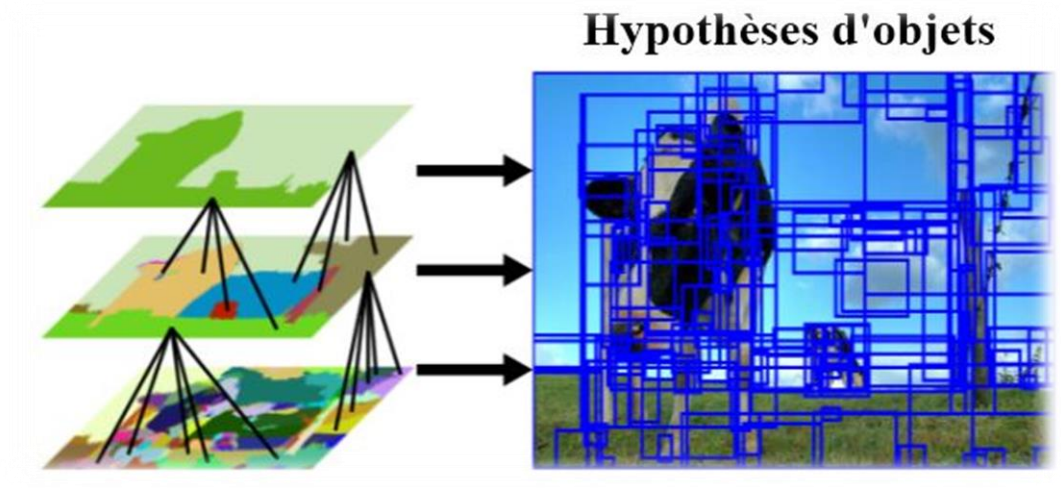


Fig 3.3 Illustration de la *selective search*. À gauche, des cartes de segmentation à différentes résolutions. À droite, les différentes propositions d'objets renvoyées par l'algorithme après la fusion des régions. Source du schéma [38].

Un travail très récent [49] l'utilise même pour la segmentation d'instances et l'estimation de la pose de personnes. Récemment, d'autres algorithmes de détection par CNN ont été introduits [50, 51]. Ils se basent sur le concept *one-shot* c'est-à-dire qu'ils n'utilisent plus d'étape d'extraction des propositions d'objets sur les cartes de caractéristiques. Cela permet d'économiser encore du temps de calcul lors de l'utilisation du CNN sur l'image. Un article très intéressant [52] fournit une analyse très poussée des différents détecteurs de l'état de l'art basés CNN [50, 44, 48] en testant différentes architectures convolutives. Dans le cadre de la conduite autonome, deux travaux ont proposé de générer des propositions d'objets directement en 3D [53, 54]. Dans ces deux approches, l'espace 3D est discrétisé (en localisation 3D par rapport à la caméra et en orientation) permettant de générer des boites 3D sur l'ensemble du plan du sol supposé planaire. L'approche 3DOP [53] (*3D Object Proposal*) utilise des images

stéréo permettant le calcul d'une carte de disparité. De cette manière, un score de confiance est attribué à chaque boite 3D en utilisant des heuristiques sur les points 3D contenus dans ces boites. Un score important est attribué aux boites 3D qui répondent à un certain nombre de critères : la densité du nuage de points 3D qu'elles contiennent mais également des contraintes géométriques sur celui-ci. Dans l'approche Mono3D [54], les auteurs proposent une génération de propositions d'objets 3D en utilisant des images monoculaires. Cette méthode projette des boites 3D dans l'image afin de récupérer la boite 2D associée. En se basant sur une carte de segmentation sémantique (obtenue grâce à l'algorithme de [55]), une carte de contour, et des informations de contexte, un score de confiance est attribué à chaque boite 3D. Dans ces deux approches, les propositions 3D sont générées sur le plan du sol puis sont projetées en 2D et envoyées à un détecteur d'objets de type Fast-RCNN [43].

III. Méthodes de détection

Dans cette partie, nous expliquons les divers algorithmes de détection. L'approche et les caractéristiques saillantes de chacun méthode.

Il existe plusieurs méthodes de détection comme HOG et SVM, R-CNN, YOLO, SSD et d'autres.

Nous mentionnerons les suivants :

III.1 Méthode de HOG et SVM

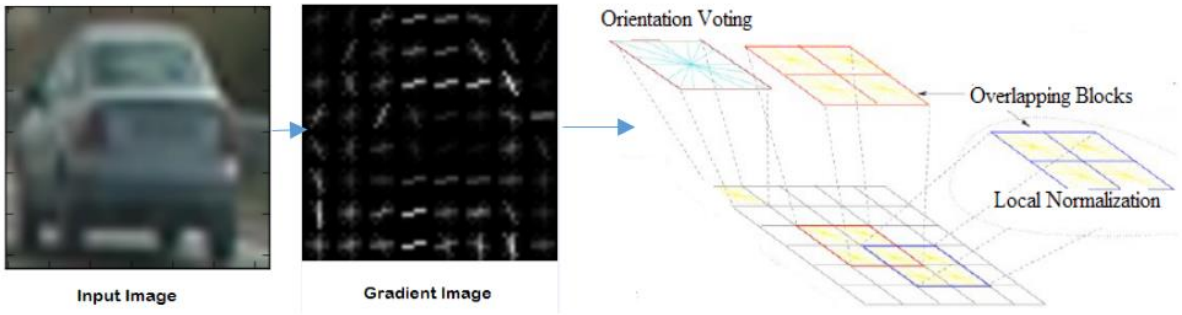


Fig 3.4 Architecture de HOG.

Histogram of Oriented Gradients (HOG) est un type de "descripteur de qualité". Il a été illustré par [56]. Ce descripteur narratif surpassait significativement les ensembles de caractéristiques existants pour la reconnaissance d'objet. L'idée principale de HOG est (La Fig 3.4) :

- L'extraction de données collectées.
- Avoir alimentées ses données à l'algorithme de classificateur Linear Support Vector Machine (L- SVM).
- Utiliser une technique de fenêtre coulissante pour vérifier si les sous-régions d'une trame contiennent des véhicules.
- Ensuite, utiliser des cartes de chaleur pour éliminer les faux positifs transitoires et gagner en confiance sur plusieurs détections au même endroit.

Le descripteur HOG maintient quelques avantages clés par rapport aux autres méthodes. Il opère sur les cellules localisées, la méthode maintient l'invariance à des transformations géométriques et photométriques, ces changements ne feront leur apparition que dans les larges régions d'espaces [57].

III.2 Méthode de YOLO

L'algorithme You Only Look Once (YOLO) est développé par [58] en 2015. Cet algorithme est utilisé pour la détection d'objets, il repose sur deux étapes : la première consiste à détecter les objets par des réseaux de neurones de convolution et la seconde correspond à quadriller l'image et la prédiction de la classe de l'objet détecté s'il existe**.** L'avantage de cette prédiction est qu'elle peut se faire indépendamment de la première détection mais qu'elle ne cible pas les objets en particulier. L'inconvénient de cette méthode est la difficulté de détecter les plus petits objets ainsi que les objets qui sont en chevauchement voir la Fig 3.5.

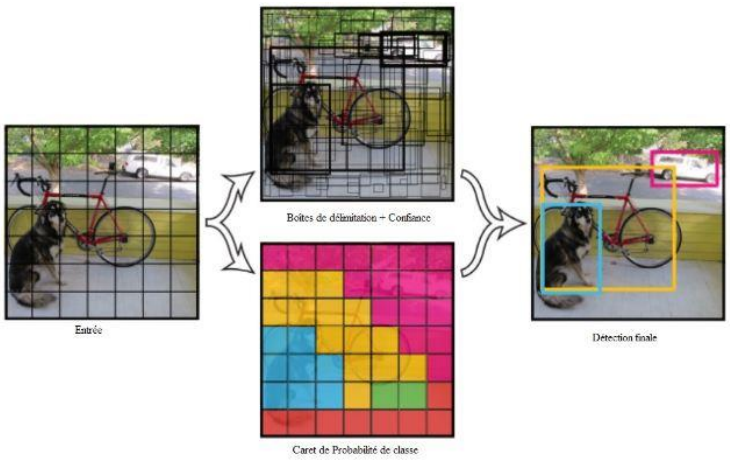


Fig 3.5 Architecture de YOLO**.** [58]

III.3 Méthode de R-CNN

R-CNN créé par [59] en 2015 qui repose sur une détection entièrement réalisée avec des réseaux de neurones de convolution. Cet algorithme va se procéder comme suit : dans la première phase le réseau de neurones de convolution prend en entrée une image de taille quelconque ensuite il va produire des régions d'intérêts dans lesquelles les R-CNN pourraient les détecter. Ces régions serons envoyées au second réseau afin d'identifier la présence de l'objet dans cette région voir la Fig 3.6.

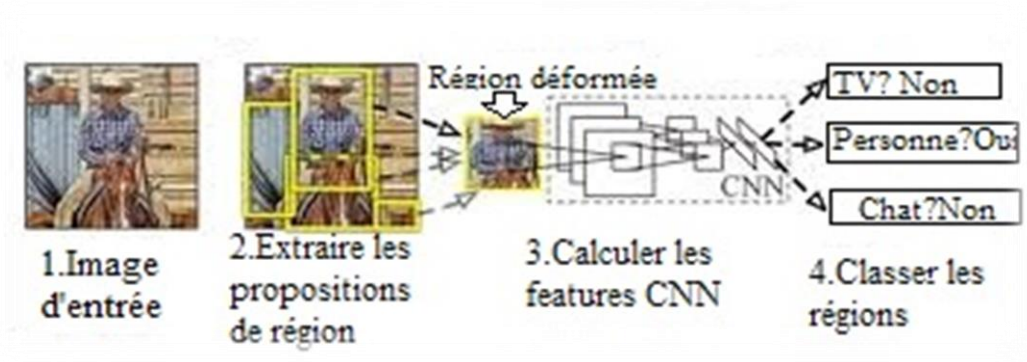


Fig. 3.6 Architecture de R-CNN**.** [59]

V. Détection de véhicules

La détection de véhicules dans un flux vidéo est un problème de détection d'objet. La détection de véhicules permet d'utiliser des diverses applications de système d'intelligence artificielle pour plusieurs buts, notamment : le transport intelligent, la surveillance automatique, la conduite autonome, et la garantie sécurité du conducteur.

Fig. 3.7 Exemples de détections de véhicules.

Il existe plusieurs méthodes de détection de véhicules nous avons cités quelque méthodes.

a) Une méthode de détection d'un véhicule présentée par Chabot.F et al dont le but est de reconnaître la marque et le modèle de véhicule. Cette approche repose sur la mise en correspondance entre le véhicule dans l'image et de modèle réel en 3D. Cette méthode est basée sur un détecteur fondé sur des réseaux de neurones convolutionnels (CNN), son principe repose sur l'extraction des points d'intérêts correspondant à des parties prédéfinies sur l'image du véhicule. Ces points seront ensuite filtrés et mis en correspondance avec les points du modèle 3D [60].

Fig. 3.8 Reconnaître la marque et le modèle de véhicule [60].

b) Une méthode de détection des véhicules à partir des images aériennes à haute résolution nommée HEM dont son processus d'apprentissage utilise *les réseaux de neurones* convolutifs. Cette méthode est proposée par Koga.Y et al [61].

Fig 3.9 Détection des véhicules dans les images aériennes (CNN) [61].

c) Yang.M et al ont adopté l'utilisation d'une fonction de perte à double focale (DFL-CNN) pour la détection des véhicules dans les images aériennes. Cette méthode est utilisée pour améliorer la capacité du réseau afin de distinguer les véhicules individuels dans une scène encombrée [62].

Fig 3.10 Détection des véhicules dans les images aériennes (DFL-CNN) [62].

d) Zhou.Y et al abordent les problèmes de détection et de classification des véhicules en utilisant les approches des réseaux neuronaux profonds (DNN). Ils répondent dans cet article aux trois questions spécifiques à leur application, La première question est posée sur l'utilisation du DNN pour la détection de véhicules , la deuxième question sur les fonctions utiles pour la classification des véhicules et la dernière question sur la façon d'étendre un modèle entraîné sur un ensemble de données de taille limitée aux cas d'éclairage extrême [63].

e) Mengxi.W a utilisé l'algorithme You Only Look Once (YOLO) pour détecter les véhicules à partir d'un flux vidéo de caméra de tableau de bord [64]. L'approche YOLO est composée de deux parties : la partie de réseau neurones qui prédit un vecteur d'une image, et la partie de post-traitement qui interpole le vecteur comme des boîtes de coordonnées et des probabilités de classe.

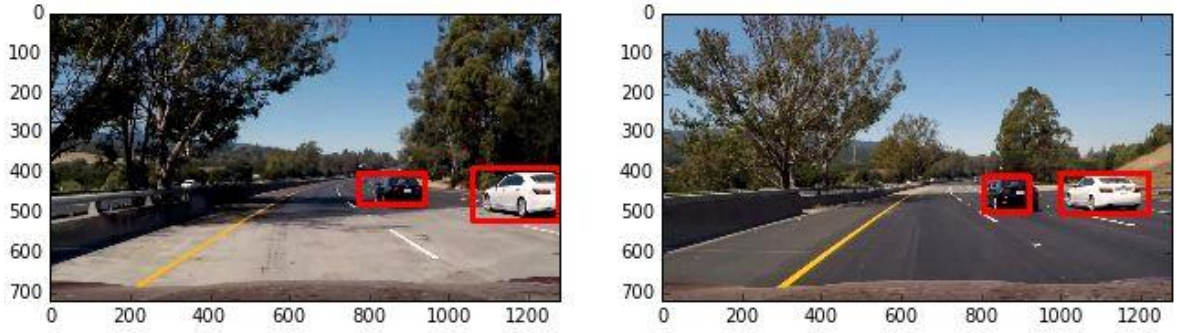


Fig 3.11 Détection des véhicules (YOLO) [64].

f) Mithi a utilisé l'algorithme Histogram of Oriented Gradients (HOG) et l'algorithme Linear Support Vector Machine (L- SVM) pour détecter et classifier les véhicules à partir d'un flux vidéo [65]. son travail est composé de plusieurs étapes :

- Effectuer une extraction de l'histogramme des dégradés orientés (HOG) sur un ensemble d'images d'entraînement étiqueté.
- Formé un classificateur avec l'ensemble des images de véhicule et de non-véhicule.
- Implémentation d'une fenêtre glissante pour obtenir des sous-régions d'une seule image vidéo.
- Utilisé un classificateur entraîné pour détecter les véhicules dans les sous-régions.
- Création d'une carte thermique des détections récurrentes image par image pour suivre et détecter les véhicules.

Fig 3.12 Détection des véhicules (HOG et L-SVM) [65].

VI. Conclusion

Nous avons présenté dans ce chapitre les technique de détection d'objets on général et les techniques de détection de véhicules on spécifique, les défis de la détection, le principe général de la détection (détection par fenêtre glissante et détection par proposition d'objets) et les différentes approches de détection d'objets comme YOLO, R-CNN, avec quelques exemple sur la détection des véhicules. Nous montrerons dans le chapitre suivant (Implémentation) notre approche qui nous avons utilisé pour la classification et de détection des véhicules basée sur les réseaux de neurones convolutionnels.

I. Introduction

Dans ce chapitre, nous allons définir l'architecture de notre modèle qu'on a créé pour détecter et classifier les véhicules et aussi pour calculer la déviation estimée du véhicule. Pour cela, on a utilisé le langage de programmation Python et ses bibliothèques : Open CV, Keras et d'autre pour développer notre modèle.

II. Configuration utilisé dans l'implémentation

L'environnement d'implémentation était un ordinateur portable TOSHIBA avec un processeur Intel Core i3-3110M 2,40 GHz, 4 Go de RAM et NVIDIA GeForce. Le système d'exploitation était Windows 7.

III. Logiciels et bibliothèques utilisés dans l'implémentation

Python

Python est un langage de programmation puissant de haut niveau, à la fois facile à apprendre, il prend en charge plusieurs modèles de programmation (procédural, fonctionnel et orienté objet). Les bibliothèques (packages) de python encouragent la modularité et la réutilisabilité des codes existants. Python et ses bibliothèques sont disponibles sans difficulté pour la majorité des plateformes et il peut être redistribué gratuitement. On estime que c'est l'un des langages de programmation les plus utilisés au monde [66].

Spyder

Spyder est un environnement de développement interactif puissant pour le langage Python, avec des fonctionnalités avancées, de test interactif, de débogage et d'introspection. De plus, Spyder est un environnement informatique numérique grâce à la prise en charge **d'IPython** et des bibliothèques Python populaires telles que **NumPy** (linear algebra), **SciPy** (signal and image processing) ou **matplotlib** (interactive 2D/3D plotting) [67]**.**

Scikit-Learn

Scikit-Learn est une bibliothèque de modèles d'apprentissage automatique développée grâce au langage de programmation Python. Scikit-Learn est un outil open source. Elle est très facile à utiliser. Elle est conçue pour s'harmoniser avec des autres bibliothèques libre Python, notamment **NumPy** et **SciPy**. Ce qui en fait un excellent point de départ pour apprendre l'apprentissage automatique [68].

Keras

Keras est une API (interface de programmation applicative) de réseaux neuronaux de haut niveau, écrite en Python et capable de fonctionner sur **TensorFlow** ou **Theano**. Il a été développé et maintenu par François Chollet pour mettre en place des modèles d'apprentissage en profondeur aussi rapides et faciles que possible pour la recherche et le développement. Keras fonctionne sur Python 2.7 ou 3.5 et peut parfaitement s'exécuter sur les processeurs graphiques GPU et les processeurs (unité centrale de traitement CPU) [69].

OpenCV

OpenCV est une bibliothèque de vision par ordinateur open source développée à l'origine par Intel. Il est gratuit pour une utilisation commerciale et de recherche sous licence BSD. Cette bibliothèque est multi-plateforme et elle fonctionne sous Mac OS X, Windows et Linux. Il se concentre principalement sur le traitement d'image en temps réel, l'importation de fichier vidéo, le traitement d'image de base (luminosité, contraste, seuil, ...), la détection d'objet (visage, corps, ...) et la détection de blobs.

OpenCV fournit un ensemble de fonctions de traitement d'image, ainsi que des fonctions d'analyse d'image et de motif. Les fonctions sont optimisées pour les processeurs d'architecture Intel® et sont particulièrement efficaces pour tirer parti de la technologie MMX. L'OpenCV implémente une grande variété d'outils pour l'interprétation d'images. Elle est compatible avec IPL (Inte Image Processing Library) qui implémente des opérations de bas niveau sur les images numériques. En dépit de primitives telles que binarisation, filtrage, statistiques d'images, pyramides. OpenCV est principalement une bibliothèque de haut niveau, elle implémente des algorithmes de calibrage (La Calibration du Camera), de détection de caractéristiques (Feature) et de tracking (Optical Flow), d'analyse de forme (Geometry, Contour Processing), l'analyse de mouvement (Motion Templates, Estimators), la reconstruction 3D (View Morphing), la segmentation et la reconnaissance d'objets (Histogram, Embedded Hidden Markov Models, Eigen Objects) [70].

IV.1 Base de données

Dans notre expérience nous avons utilisés une base de données qui contient deux fichiers un fichier qui porte le nom « véhicule » et un autre fichier qui s'appelle « non-véhicule ». Cette base de données possède des images extraites à partir de séquences vidéo (obtenues par une caméra frontale montée sur une voiture). Pour assurer le bon apprentissage des données, les images sont capturées dans des différentes conditions routières (loin, à gauche, à droite, Moyen Proche). Le fichier véhicule comprend 8798 images et le fichier non véhicule comprend 8971. Chaque taille d'image est de 64 * 64 pixels.

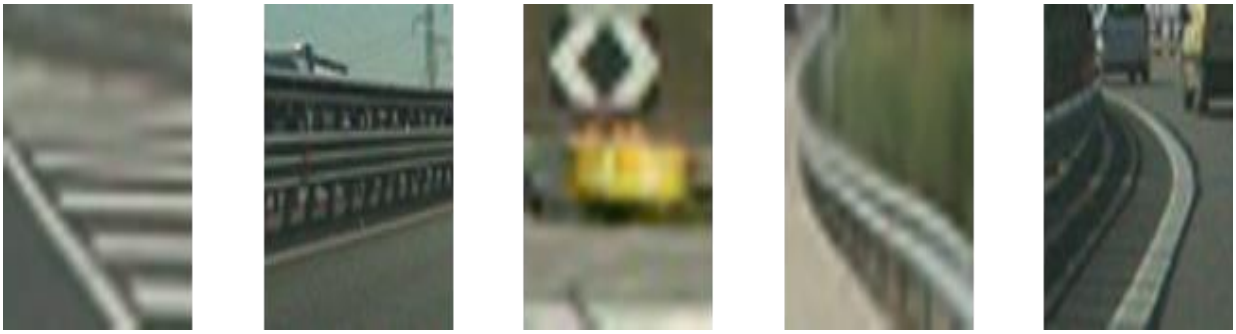


Fig 4.1 Images non-véhicule.

Fig 4.2 Images véhicule.

La base de données utilisée dans notre travail, elle est divisée en trois parties ; la première et la deuxième partie contient les images d'entrainement et de validation qui sont utilisées pour

extraire les caractéristiques de chaque image de la base de données (assure l'apprentissage à l'algorithme), la troisième partie représente 20% d'images, elle constitue un ensemble de tests qui permette de vérifier les effets de l'algorithme.

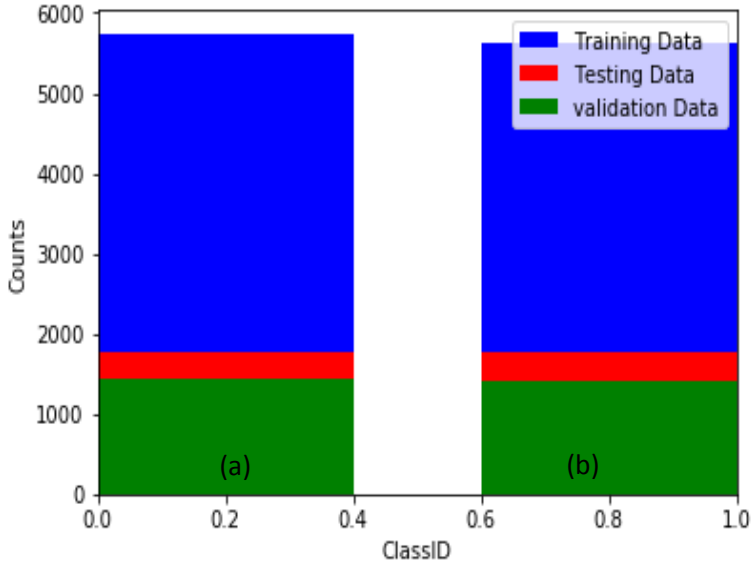


Fig 4.3 Histogramme de la base de données.

La Fig 4.3 est une représentions graphique de la base de données. La ressemblance entre la partie gauche (a) et la partie droite (b) de l'histogramme s'est produit à cause du nombre d'images non-véhicule qui est plus proche au nombre d'image véhicule.

IV.2 Explication de l'architecture de notre modèle

Au cours de nos expérimentations, nous avons créé un modèle qui se compose de six couches de convolution, une couche de Maxpooling, une autre couche de dropout et une couche de fully connected (FC) la Fig 4.4. Explique l'architecture de notre modèle :

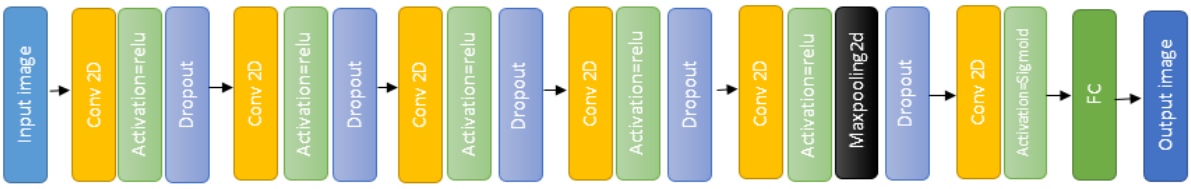


Fig 4.4 Architecture du CNN.

Nous avons utilisé la couche dropout (expliqué dans le chapitre 2) dans les cinq premières couches de convolutions et on a déterminé sa valeur avec 0.5.

L'image en entrée est de taille 64*64, l'image passe d'abord dans la première couche de convolution. Cette couche est composée de 8 filtres, suivies par une fonction d'activation ReLU. A la sortie de la première couche de convolution on aura 8 les carte de caractéristique (features maps).

Les 8 Features maps qui sont obtenus auparavant ils sont donnés en entrée de la deuxième couche de convolution qui est composée aussi de 16 filtres et une fonction d'activation ReLU qu'est appliquée sur la couche de convolution. À la sortie de cette couche, nous aurons 16 Features maps.

On répète la même chose avec la couche de convolutions trois, cette couche est composée de 32 filtres, suivie par la fonction d'activation ReLU, après cette couche de convolution 32 features maps auront comme sortie.

Par la suite, les 32 features maps seront donnée à la couche de convolution quatre, cette couche est composée de 64 filtres, suivie par la fonction d'activation ReLU, à la fin nous aurons 64 features maps.

Dans la cinquième couche en aura en entrée les 64 features maps obtenus de la couche précédente cette couche est composée de 128 filtres et d'une fonction d'activation ReLU, suivie par une couche Maxpooling de taille 8*8 pour réduire la taille de l'image à la sortie de cette couche nous aurons 128 features maps.

Dans la sixième couche en aura en entrée 128 features maps pris de la couche précédente cette couche est composée d'un seul filtre et d'une fonction d'activation Sigmoid pour assurer la classification, la taille de l'image à la sortie de cette couche et 1 features maps.

Après ces six couches de convolution, nous utilisons un réseau de neurones, une couche de fully connected (FC) pour entraîner la classification des données de la couche précédente (véhicule, non véhicule)).voir la Fig 4.5

Layer (type)	Output Shape	Param #
lambda_1_input (InputLayer)	(None, 64, 64, 3)	0
lambda_1 (Lambda)	(None, 64, 64, 3)	0
cv0 (Conv2D)	(None, 64, 64, 8)	224
dropout_1 (Dropout)	(None, 64, 64, 8)	0
cv1 (Conv2D)	(None, 64, 64, 16)	1168
dropout_2 (Dropout)	(None, 64, 64, 16)	0
cv2 (Conv2D)	(None, 64, 64, 32)	4640
dropout_3 (Dropout)	(None, 64, 64, 32)	0
cv3 (Conv2D)	(None, 64, 64, 64)	18496
dropout_4 (Dropout)	(None, 64, 64, 64)	0
cv4 (Conv2D)	(None, 64, 64, 128)	73856
max_pooling2d_1 (MaxPooling2	(None, 8, 8, 128)	0
dropout_5 (Dropout)	(None, 8, 8, 128)	0
fcn (Conv2D)	(None, 1, 1, 1)	8193
flatten_1 (Flatten)	(None, 1)	0

Total params: 106,577
Trainable params: 106,577
Non-trainable params: 0

Fig 4.5 Configuration de modèle.

L'image suivante est une illustration de notre modèle réseaux de neurones convolutional CNN et de son fonctionnement.

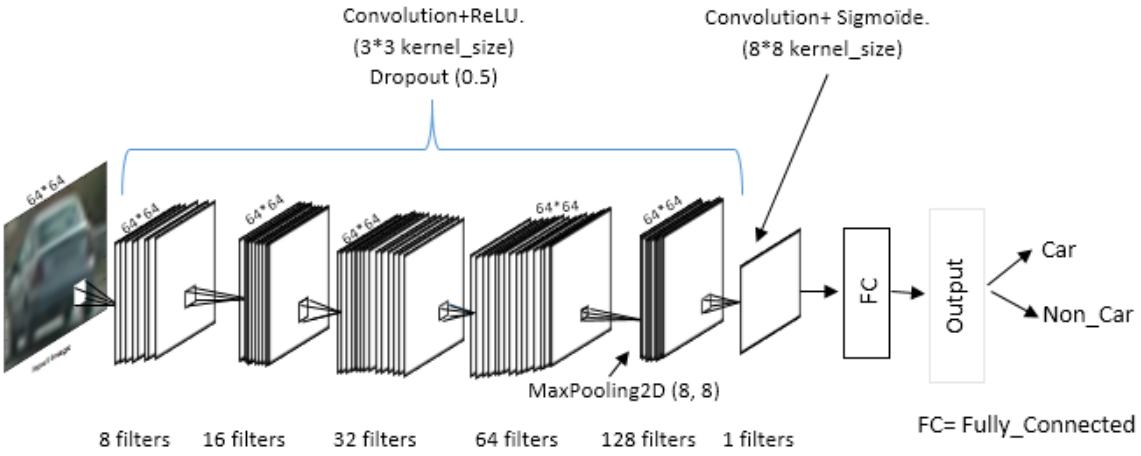


Fig 4.6 Architecture CNN plus détaillé de notre modèle.

IV.3 Heatmap opération

La fonction heatmap est généralement utilisée pour extraire les zones chaudes dans une image. Nous avons utilisé cette fonction pour faire détecter les véhicules qui circule devant la caméra.

Fig 4.7 Heat map.

IV.4 Calculer la déviation estimée du véhicule

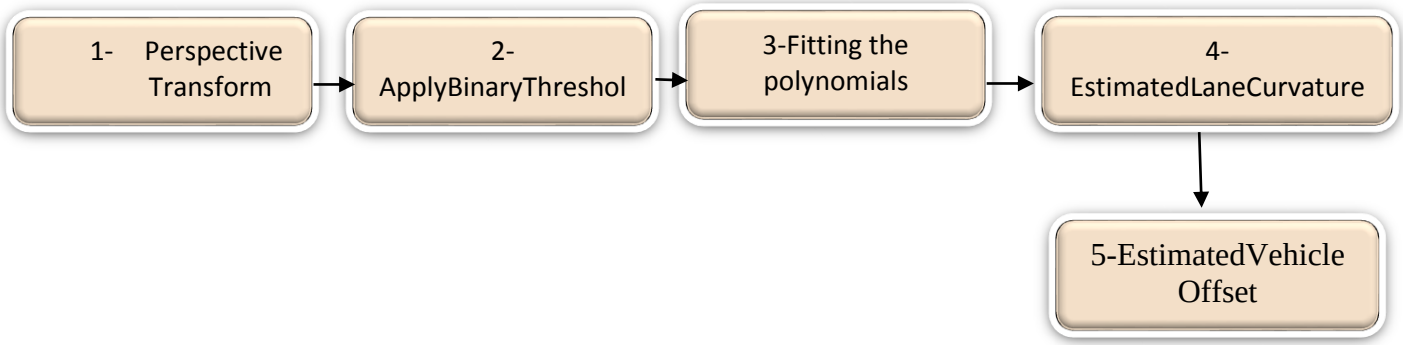


Fig 4.8 Les étapes de calculer la déviation estimé du véhicule.

Avant l'étape de transformation de perspective nous devons passer par l'étape de caméra calibration. Nous avons utilisé les fonctions **OpenCV `findChessboardCorners`** et **`drawChessboardCorners`** pour identifier les emplacements des coins sur une série d'images d'un échiquier pris sous différents angles. Ensuite, les emplacements des coins de l'échiquier

ont été utilisés comme entrée dans la fonction **OpenCV `calibrateCamera`** pour calculer la matrice d'étalonnage de la caméra et les coefficients de distorsion. Enfin, la matrice d'étalonnage de la caméra et les coefficients de distorsion ont été utilisés avec la fonction **OpenCV «undistort»** pour corriger la distorsion des images de conduite sur autoroute.

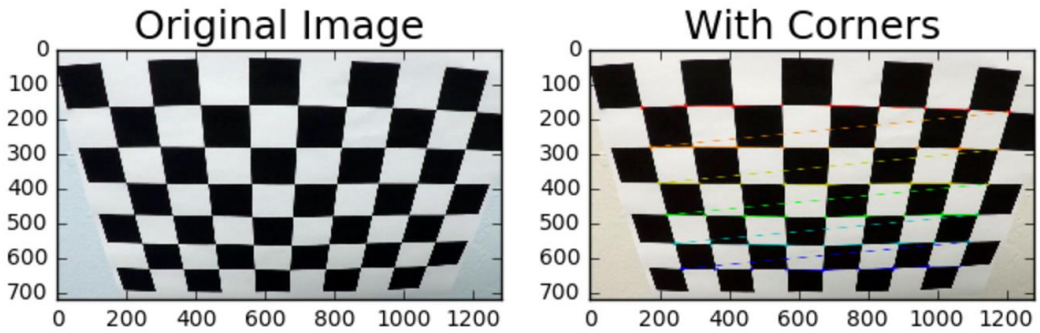
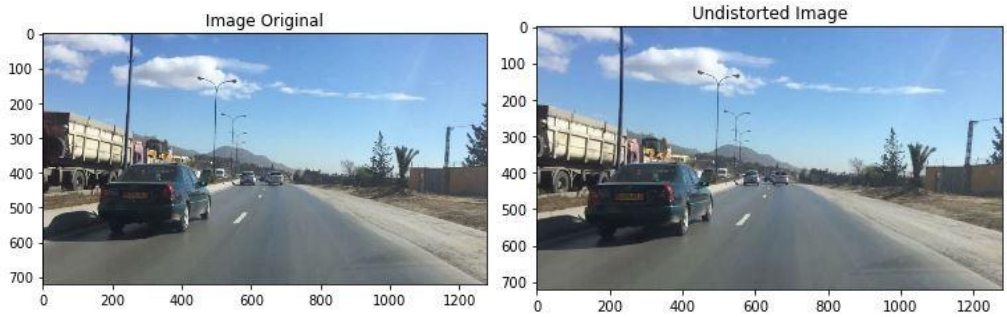


Fig 4.9 Explique l'identification les emplacements des coins.



(a) **(b)**

Fig 4.10 (a) Image original et **(b)** Image non-distorsion.

Dans la première étape nous transformons l'image non distorsion à une «vue d'oiseau» à l'aide de la fonction **OpenCV 'birdEye'** de la route qui se concentre uniquement sur les lignes de voie et les affiche de telle sorte qu'elles semblent être relativement parallèles entre elles. Pour réaliser la transformation de perspective, nous avons d'abord appliqué les fonctions **OpenCV `getPerspectiveTransform` et `Perspective`.**

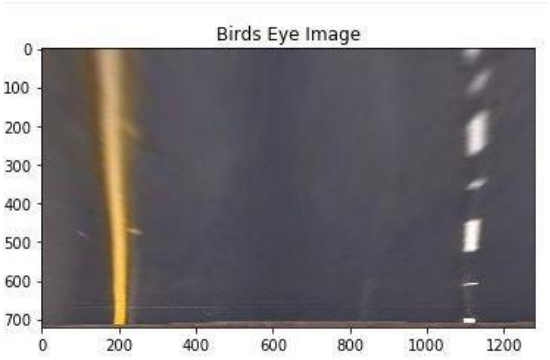


Fig 4.11 Vue d'oiseau.

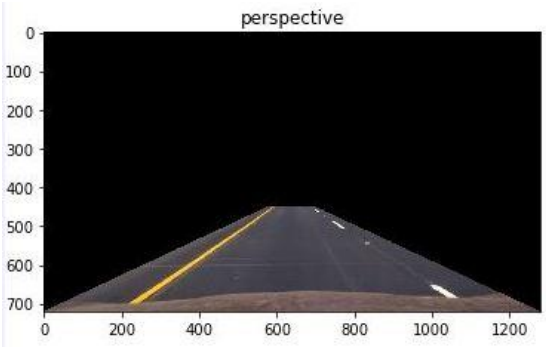


Fig 4.12 Perspective.

Dans la deuxième étape, nous avons convertie l'image en espaces de couleur (HLS, LUV) puis nous avons convertie les images en nuance de gris qui ne mettent en évidence que les lignes de la voie et ignorent tout le reste. Pour arriver à cet objectif nous avons appliqué des canaux de couleur (HLS, LUV) et nous avons précisé les seuils qui permettaient d'identifier les lignes de la voie dans les images.

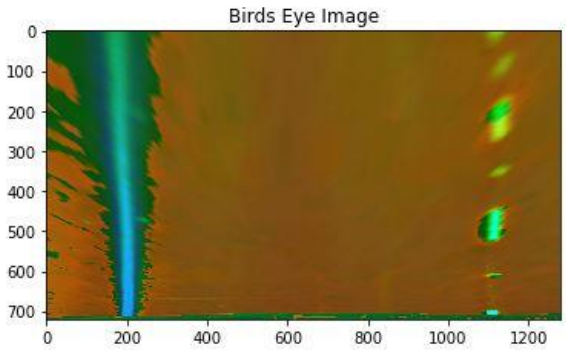


Fig 4.13 Vue d'oiseau avec l'espace HLS.

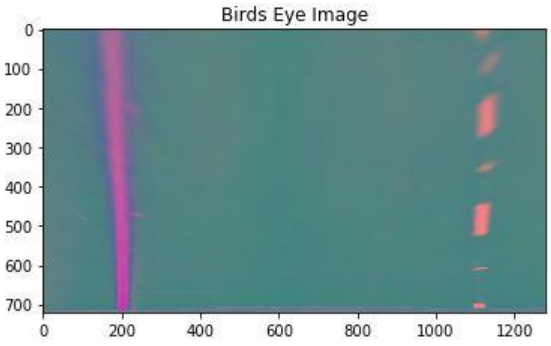


Fig 4.14 Vue d'oiseau avec l'espace YUV.

L'objectif de la troisième étape consiste à identifier les pics dans un histogramme de l'image pour déterminer l'emplacement des lignes de voie, puis elle identifie tous les pixels non-zéros autour des pics. Dans la quatrième étape, nous avons calculé le rayon de courbure on a appliqué les fonctions '**curvature, addCurvatureStamp** ', Basé sur le travail des auteurs [71], [72], la formule du rayon de courbure en tout point x de la courbe $y = f(x)$, l'équation (4.1).

$$f(x) = \frac{\left[1+\left(\frac{dy}{dx}\right)^2\right]^{3/2}}{\left|\frac{|d^2y|}{dx^2}\right|} \qquad (4.1)$$

Dans la dernière étape nous supposons que la caméra est montée au centre de la voiture au-dessous de rétroviseur. On calcule le décalage de sorte que le centre de la voie est le point médian les deux lignes que nous avons détectées. Le décalage est donc la différence entre le centre de la voiture et le centre de la voie **(la fonction addOffsetStamp)** voir la Fig 4.15.

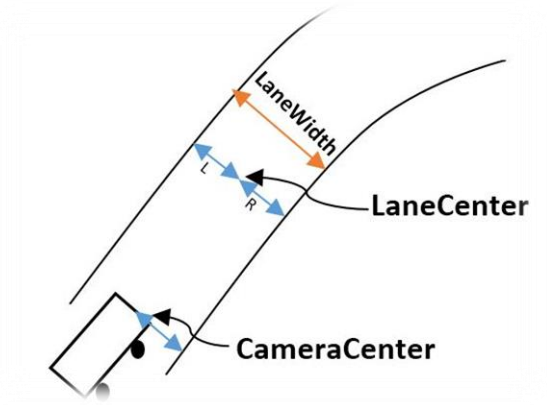


Fig 4.15 Le calcul de déviation du véhicule en circulation.

VII. Architecture général de notre approche

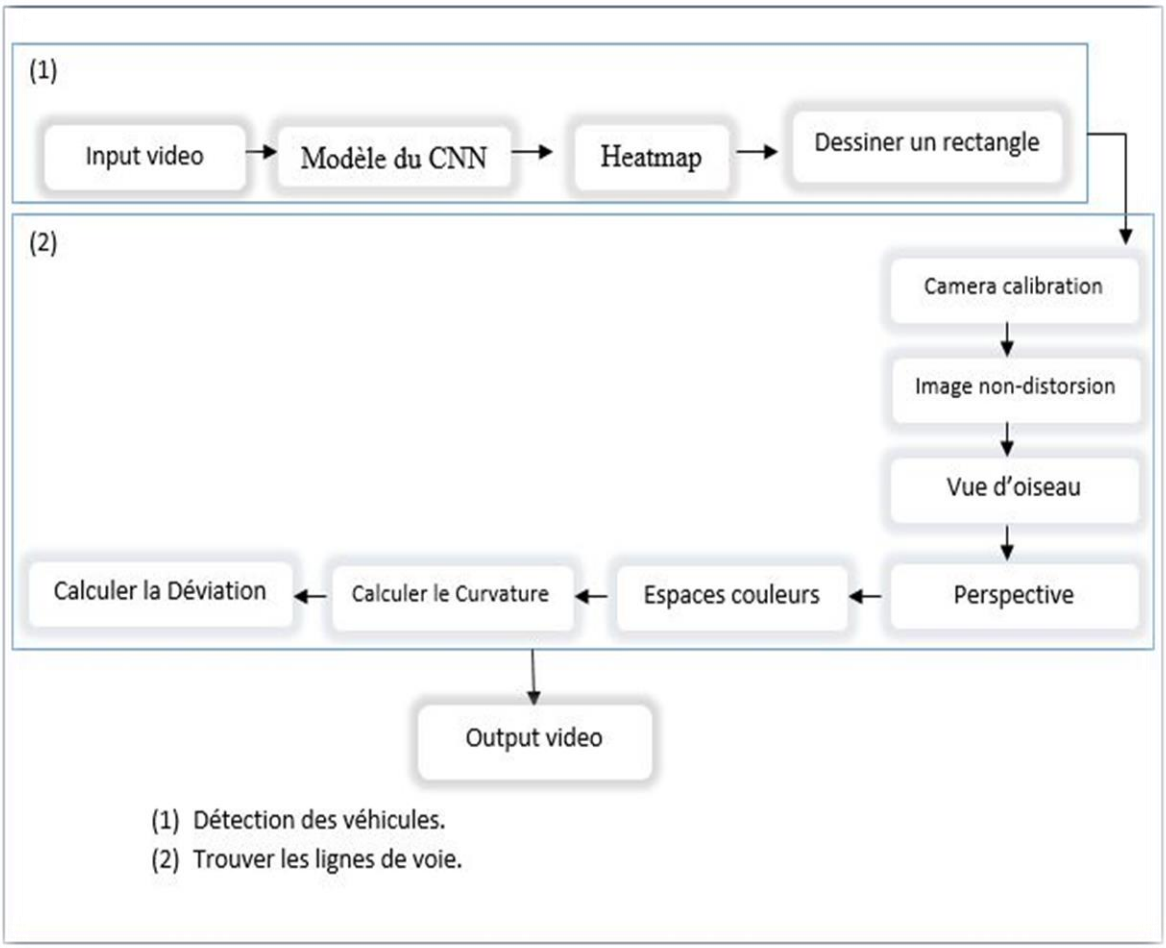


Fig 4.16 Architecture général.

IV. Conclusion

Nous avons présenté dans ce chapitre une approche de classification et de détection de véhicules basée sur les réseaux de neurones convolutionnels, aussi on a défini l'architecture de modèle qu'on a créé pour assurer la bon détecter et classifier des véhicules et aussi pour calculer la déviation estimée du véhicule. Pour cela, on a utilisé le langage de programmation python et ses bibliothèques pour développer notre modèle.

Nous montrerons dans le chapitre suivant les résultats obtenus en termes de précision et d'erreur.

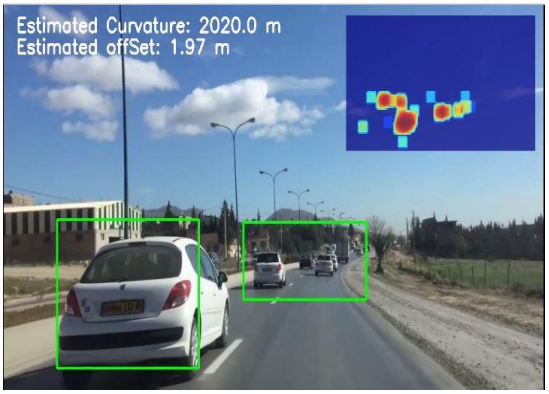
I. Introduction

Dans ce chapitre, nous allons présenter les résultats obtenus de flux video, et les courbes de précision sur le jeu d'apprentissage et le jeu de validation qui est montré l'efficacité de notre approche**.**

II. Résultat

Dans cette partie nous évaluons notre modèle en termes de précision et de performance pour montrer l'efficacité de notre approche.

Les figures ci-dessous montrent quelques résultats obtenus par notre modèle.



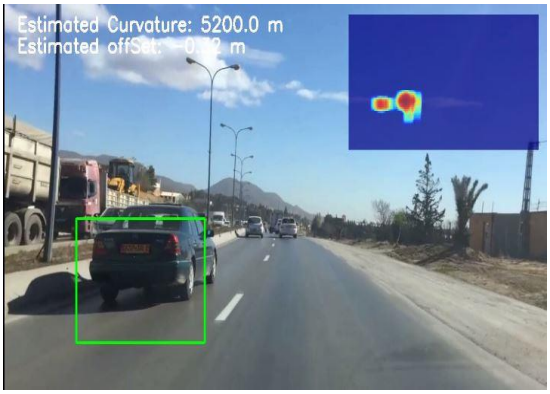


Fig 5.1 Détection de véhicule par notre approche.

En remarquant dans les figures précédentes que la détection des véhicules a été bien réalisée par notre modèle.

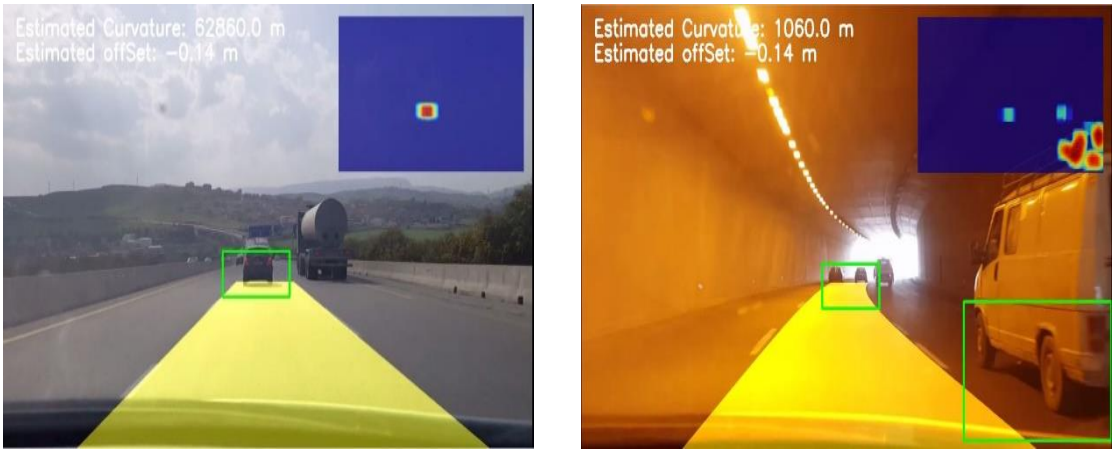


Fig 5.2 Détection des véhicules et la précision de la voie.

Dans cette figure nous constatons que notre système a bien précisé la voie sur laquelle le véhicule doit circuler.

Dans la Fig 5.3 nous montrons un graphe qui représente l'évolution de courbe de précision sur le jeu d'apprentissage et le jeu de validation. Sachant que nous avons utilisé cinq époques.

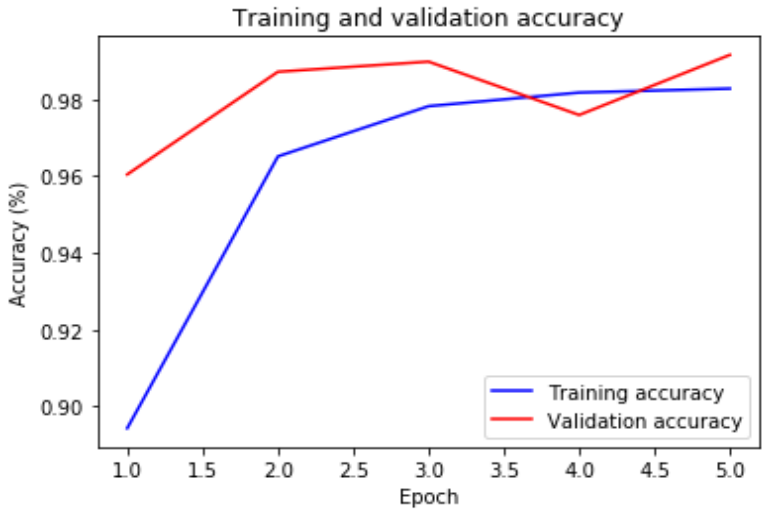


Fig 5.3 La précision d'entrainement et de validation.

Nous constatons que la précision augmente progressivement en fonction de nombre d'époques, nous observons qu'il y a un écart entre les deux courbes, cela signifie que le modèle se généralise d'une meilleur façon en particulier à partir de la quatrième époque.

Nous montrons ci-dessous un graphe qui représente l'évolution de courbe de la fonction d'erreur (Loss) sur le jeu d'apprentissage et le jeu de validation.

Fig 5.4 L'erreur d'entrainement et de validation.

La Fig 5.4 montre que la fonction d'erreur décroit au fur à mesure que le nombre d'époque augmente, ceci signifie que le modèle donne de bonnes prédictions sur le jeu d'apprentissage que sur le jeu de validation. En constatant que la fonction d'erreur sur le jeu de validation se commence à augmenter à partir de la quatrième époque, ceci signifie que le modèle sur-ajuste. On concluant que notre modèle est meilleur en atteignant un taux de précision satisfaisant qui arrive au 99.12 % et taux d'erreur très réduit qui égale à 0.88% .

```
Epoch 1/5
711/710 [==============================] - 5650s 8s/step - loss: 0.0790 - acc: 0.8944 - val_loss: 0.0319 - val_acc: 0.9604
Epoch 2/5
711/710 [==============================] - 5626s 8s/step - loss: 0.0270 - acc: 0.9650 - val_loss: 0.0133 - val_acc: 0.9872
Epoch 3/5
711/710 [==============================] - 5640s 8s/step - loss: 0.0169 - acc: 0.9782 - val_loss: 0.0101 - val_acc: 0.9898
Epoch 4/5
711/710 [==============================] - 5649s 8s/step - loss: 0.0147 - acc: 0.9818 - val_loss: 0.0189 - val_acc: 0.9759
Epoch 5/5
711/710 [==============================] - 5642s 8s/step - loss: 0.0137 - acc: 0.9828 - val_loss: 0.0068 - val_acc: 0.9916
Training complete. Weights for best validation accuracy have been saved to ppico_2018_4_2_10_46_10.h5.
Evaluating accuracy on test set.
test accuracy:  [0.0070340291539717076, 0.99128981171916242]
```

Fig 5.5 Exactitude du test.

III. Conclusion

On a montré dans ce chapitre que notre méthode de travail améliore grandement le taux de précision (99.2%) et diminue le taux d'erreur. Nous conclurons d'après les courbes et les figures précédents que notre modèle a donnée des bons résultats en termes de détection et de reconnaissance des véhicules dans un flux vidéo. Le nombre d'époque et la profondeur de réseaux, sont des facteurs importants pour l'obtention de meilleurs résultats.

CONCLUSION GENERALE

Ce mémoire traite la classification et la détection des véhicules en mouvement dans une séquence video et calculer la déviation estimé du véhicule.

Nous avons tendance à relever le défi de reconnaissance et la détection des véhicules en utilisant l'algorithme d'apprentissage en profondeur basée sur le réseau neuronal convolutif (CNN). Nous avons atteint ce but en découpons la phase de classification et de reconnaissance en plusieurs étapes, la première étape consiste à effectuer des prétraitements sur le jeu d'apprentissage. La deuxième étape est composée d'un réseau de convolution afin d'extraire les caractéristiques, la couche Maxpooling et pour améliorer les performances des modèles on va utiliser une technique simple et efficace dropout et à la fin nous utilisons un réseau de neurone entièrement connecté pour entraîner la classification des données de la couche précédente (véhicule, non véhicule)).

Les résultats obtenus ont montré que la fiabilité de détection de véhicules et calculer la déviation estimé du véhicule. Nous avons montré que notre méthode de travail améliore grandement le taux de précision (99.2%) et diminue le taux d'erreur. Les résultats montrent également que la valeur de fiabilité est assez fiable.

Nous avons rencontré quelques problèmes dans la phase d'implémentation, malgré l'utilisation des techniques de régularisation, de standardisation et d'optimisation l'utilisation d'un CPU a fait que le temps d'exécution était trop couteux. Afin de régler ce problème on doit utiliser des réseaux de neurones convolutionnels plus profonds déployé sur un GPU au lieu d'un CPU sur des bases plus importantes.

Dans les travaux futurs, nous voulons améliorer la méthode en utilisant un réseau de neurones préformés cela nous permettra d'augmenter les performances de notre modèle, Ainsi nous prévoyons de tirer parti de l'immense puissance de calcul des processeurs graphiques GPU en distribuant les calculs sur plusieurs GPU.

BIBLIOGRAPHIE

[1] Russell, S. J., & Norvig, P. (2016). Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited, « Meghyn. Bienvenu. Introduction à l'Intelligence Artificielle. Université Aix-Marseille 1. 2007-2008 et 2008-2009. ».

[2] Touzet, C. (1992). Les réseaux de neurones artificiels, introduction au connexionnisme. EC2.

[3] Thibodeau-Laufer, E. (2014). Algorithmes d'apprentissage profonds supervisés et non-supervisés : applications et résultats théoriques, (maître ès sciences, Université de Montréal).

[4] Portugal, I., Alencar, P., & Cowan, D. (2017). The use of machine learning algorithms in recommender systems: a systematic review. Expert Systems with Applications.

[5] Priyadharshini. (March 8, 2018). Machine Learning: What it is and Why it Matters, ''simplilearn,'' https://www.simplilearn.com/ .

[6] Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. "O'Reilly Media, Inc.".

[7] Chapelle, O., Scholkopf, B., & Zien, A. (2009). Semi-supervised learning (chapelle, o. et al., eds.; 2006) [book reviews]. IEEE Transactions on Neural Networks, 20(3), 542-542.

[8] Hamel, P. (2012). Apprentissage de représentations musicales à l'aide d'architectures profondes et multiéchelles, (Doctoral dissertation, Université de Montréal).

[9] Glorot, X. (2015). Apprentissage des réseaux de neurones profonds et applications en traitement automatique de la langue naturelle, (Doctoral dissertation, Université de Montréal).

[10] Bastien L, Deep Learning ou apprentissage profond : définition, le magazine Cloud and big data https://www.lebigdata.fr/ , 27 février 2018.

[11] Dorianne W, Intelligence artificielle, machine learning, deep learning: kézako?, ledigitalab https://www.ledigitalab.com/fr/ , 2 October 2017,page2/10.

[12] Rémi S, Le Deep Learning pour Tous, internetactu http://www.internetactu.net/ , 2 October 2014.

[13] Philippe B : Une première introduction au Deep Learning, Microsoft| Developer, Machine Learning France https://msdn.microsoft.com/fr-fr/ , April 28, 2016.

[14] Ludovic L: Machine learning et deep learning, comment ça marche?, siecledigital.fr https://siecledigital.fr/ ,22 December 2016.

[15] Ed Burns : Deep learning vs. machine learning: The difference starts with data, searchbusinessanalytics https://searchenterpriseai.techtarget.com/ , 18 May 2017.

[16] Fresnel, Q., & Peeters, G. (2015). *Apprentissage de descripteurs audio par Deep learning, application pour la classification en genre musical* (Doctoral dissertation, Univserité Paris 6).

[17] Bordes, F. (2017). Learning to sample from noise with deep generative models.

[18] Chabot, F. (2017). Analyse fine 2D/3D de véhicules par réseaux de neurones profonds (Doctoral dissertation, Université Clermont Auvergne).

[19] Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. «O'Reilly Media, Inc.".

[20] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[21] Xiang, W., Tran, H. D., & Johnson, T. T. (2017). Reachable Set Computation and Safety Verification for Neural Networks with ReLU Activations. arXiv preprint arXiv:1712.08163.

[22] Kanestrøm, P. Ø. (2017). Traffic flow forecasting with deep learning (Master's thesis, NTNU).

[23] Pezeshki, M. (2017). Towards deep semi supervised learning.

[24] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, *15*(1), 1929-1958.

[25] Chabot, F. (2017). *Analyse fine 2D/3D de véhicules par réseaux de neurones profonds* (Doctoral dissertation, Université Clermont Auvergne).

[26] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

[27] Ferrari, V., Jurie, F., & Schmid, C. (2007, June). Accurate object detection with deformable shape models learnt from images. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on* (pp. 1-8). IEEE.

[28] Ferrari, V., Jurie, F., & Schmid, C. (2010). From images to shape models for object detection. *International journal of computer vision*, *87*(3), 284-303.

[29] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (Vol. 1, pp. 886-893). IEEE.

[30] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, *32*(9), 1627-1645.

[31] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*.

[32] Szegedy, C., Toshev, A., & Erhan, D. (2013). Deep neural networks for object detection. In *Advances in neural information processing systems* (pp. 2553-2561).

[33] Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (Vol. 1, pp. I-I). IEEE.

[34] Freund, Y., & Schapire, R. E. (1995, March). A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory* (pp. 23-37). Springer, Berlin, Heidelberg.

[35] Torralba, A., Murphy, K. P., & Freeman, W. T. (2004, June). Sharing features: efficient boosting procedures for multiclass object detection. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on* (Vol. 2, pp. II-II). IEEE.

[36] Torralba, A., Murphy, K. P., & Freeman, W. T. (2007). Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(5), 854-869.

[37] Garcia, C., & Delakis, M. (2002). A neural architecture for fast and robust face detection. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on* (Vol. 2, pp. 44-47). IEEE.

[38] Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, *104*(2), 154-171.

[39] Carreira, J., & Sminchisescu, C. (2010, June). Constrained parametric min-cuts for automatic object segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (pp. 3241-3248). IEEE.

[40] Endres, I., & Hoiem, D. (2010, September). Category independent object proposals. In *European Conference on Computer Vision* (pp. 575-588). Springer, Berlin, Heidelberg

[41] Felzenszwalb, P. F., & Huttenlocher, D. P. (2004). Efficient graph-based image segmentation. *International journal of computer vision*, *59*(2), 167-181.

[42] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

[43] Girshick, R. (2015). Fast r-cnn. *arXiv preprint arXiv:1504.08083*.

[44] S. Ren, K. He, R.B. Girshick et J. Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. NIPS, 2015.

[45] Yang, F., Choi, W., & Lin, Y. (2016). Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2129-2137).

[46] Xiang, Y., Choi, W., Lin, Y., & Savarese, S. (2017, March). Subcategory-aware convolutional neural networks for object proposals and detection. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on* (pp. 924-933). IEEE.

[47] Kong, T., Yao, A., Chen, Y., & Sun, F. (2016). Hypernet: Towards accurate region proposal generation and joint object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 845-853).

[48] Dai, J., Li, Y., He, K., & Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems* (pp. 379-387).

[49] K. He, G. Gkioxari, P. DollÃ¡r et R. Girshick. *Mask R-CNN*. arXiv :1703.06870, 2017.

[50] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd : Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

[51] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

[52] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. (2017, July). Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE CVPR*.

[53] Chen, X., Kundu, K., Zhu, Y., Berneshawi, A. G., Ma, H., Fidler, S., & Urtasun, R. (2015). 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems* (pp. 424-432).

[54] Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., & Urtasun, R. (2016). Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2147-2156).

[55] Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., ... & Torr, P. H. (2015). Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1529-1537).

[56] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (Vol. 1, pp. 886-893). IEEE.

[57] ATRI, M., SAIDANI, T., & TOURKI, R. Détection d'Individu d'Histogramme Intégral.

[58] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

[59] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

[60] F. Chabot, M. Chaouch, J. Rabarisoa, T. Chateau, and C. Truliére, "Détection de pose de véhicule pour la reconnaissance de marque et modèle," *in Journées francophones des jeunes chercheurs en vision par ordinateur.* 2015.

[61] Y. Koga, H. Miyazaki, and R. Shibasaki, "A CNN-Based Method of Vehicle Detection from Aerial Images Using Hard Example Mining," *Remote Sensing,* vol. 10, pp. 124.2018.

[62] Yang, M. Y., Liao, W., Li, X., & Rosenhahn, B. (2018). Vehicle Detection in Aerial Images. *arXiv preprint arXiv:1801.07339*..

[63] Y. Zhou, H. Nejati, T. T. Do, N. M. Cheung, and L. Cheah, "Image-based vehicle analysis using deep neural network: A systematic study," *in Digital Signal Processing (DSP), 2016 IEEE International Conference on. IEEE* p. 276-280.

[64] W. Mengxi, (2017). Real time vehicle detection using YOLO. *The Medium website.www.medium.org.*

[65] Mithi, (2017). Vehicle Detection with HOG and Linear SVM. *The* Towards Data Science. *Website. www.*towardsdatascience.com.

[66] Van Rossum, G. (1990). The python language. See http://www. python. org.

[67] Raybaut, P. (2009). Spyder-Documentation. Available online at: pythonhosted. org.

[68] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., ... & Layton, R. (2013). API design for machine learning software: experiences from the scikit-learn project. arXiv preprint arXiv:1309.0238.

[69] Chollet, F. (2017). Deep learning with python. Manning Publications Co.

[70] Bradski, G., & Kaehler, A. (2008). Learning OpenCV: Computer vision with the OpenCV library. "O'Reilly Media, Inc.".

[71] Zamfir, S., Drosescu, R., & Gaiginschi, R. (2016, August). Practical method for estimating road curvatures using onboard GPS and IMU equipment. In IOP Conference Series: Materials Science and Engineering (Vol. 147, No. 1, p. 012114). IOP Publishing.

[72] Jin, C., Wang, X., Miao, Z., & Ma, S. (2017, October). Road curvature estimation using a new lane detection method. In Chinese Automation Congress (CAC), 2017 (pp. 3597-3601). IEEE