الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Département : Génie Électrique Et Informatique Industrielle

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en Électronique

Spécialité

Systèmes Embarqués

Étude et simulation d'une voiture autonome utilisant l'apprentissage approfondi (Deep Learning) : Application sur le simulateur Udacity

Réalisé par

Hannah Melissa KHELIL, Mohamed ZELLAT

Présenté et soutenu publiquement le (06/07/2022)

Composition du Jury :

Président	M. Amar REZOUG	M. C. A.	ENST
Promoteur	M. El-Hadi KHOUMERI	M. C. B.	ENST
Examinateur	M. Nouredine CHERABIT	M. A. A.	ENST
Examinateur	Mme. Karima REBAI	M. C. B.	ENST

Année universitaire 2021 – 2022

الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne Démocratique et Populaire
وزارة التعليم العالي والبحث العلمي
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Département : Génie Électrique Et Informatique Industrielle

Mémoire de projet de fin d'études
pour l'obtention du diplôme d'ingénieur d'état en Électronique

Spécialité

Systèmes Embarqués

Étude et simulation d'une voiture autonome utilisant l'apprentissage approfondi (Deep Learning) : Application sur le simulateur Udacity

Réalisé par

Hannah Melissa KHELIL, Mohamed ZELLAT

Présenté et soutenu publiquement le (06/07/2022)

Composition du Jury :

Président	M. Amar REZOUG	M. C. A.	ENST
Promoteur	M. El-Hadi KHOUMERI	M. C. B.	ENST
Examinateur	M. Nouredine CHERABIT	M. A. A.	ENST
Examinateur	Mme. Karima REBAI	M. C. B.	ENST

Année universitaire 2021 – 2022

Remerciements

Louanges à Allah le Tout Puissant.

Qu'Il nous pardonne et nous guide vers le droit chemin.

Nous tenons à rendre hommage et à remercier nos parents et nos familles qui ont toujours été présents et qui nous ont soutenu durant tout notre cursus.

Merci à Monsieur El-Hadi Khoumeri de nous avoir encadrés, conseillés, orientés et nous avoir poussés à persévérer dans nos efforts.

Nous remercions les membres du jury d'avoir donné de leur temps afin d'examiner cet humble travail.

Nous tenons aussi à remercier l'ensemble du corps professoral et administratif de l'École Nationale Supérieure de Technologie pour la richesse et la qualité de leur enseignement et les grands efforts qu'ils ont déployés pour assurer à leurs étudiants une formation actualisée.

Enfin, merci à toutes celles et ceux qui nous ont aidés dans l'accomplissement de ce mémoire.

Table des matières

Table des figures i
Liste des abréviations iii

Introduction générale **1**

I État de l'art sur les voitures autonomes **2**
I.1 Introduction 2
I.2 Définition du terme «voiture autonome» 2
I.3 Classification des voitures autonomes 2
I.3.1 Niveaux 0, 1 et 2 : Les fonctionnalités d'aide à la conduite 2
I.3.2 Niveaux 3, 4 et 5 : Les fonctionnalités de conduite automatique . . 3
I.4 L'histoire des voitures autonomes 3
I.5 Fonctionnement des voitures autonomes 6
I.5.1 Caméras 7
I.5.2 Stéréovision 7
I.5.3 Lidar 8
I.5.4 Radar 8
I.5.5 Sonar 9
I.5.6 Centrale inertielle 10
I.5.7 Odomètre 10
I.6 Perspectives d'évolution des voitures autonomes 10
I.7 Avantages, obstacles et inconvénients des voitures autonomes 11
I.7.1 Avantages des voitures autonomes 11
I.7.2 Obstacles et inconvénients des voitures autonomes 12
I.8 Conclusion 12

II Notions générales sur l'intelligence artificielle et l'apprentissage approfondi **13**
II.1 Introduction 13
II.2 L'intelligence artificielle 13
II.3 Apprentissage Automatique 14
II.3.1 Les réseaux de neurones artificiels 15
II.3.1.1 Fonction d'activation 16
II.3.1.1.a ReLU 16
II.3.1.1.b eLU 16
II.3.1.1.c Softmax 17
II.3.1.1.d Sigmoid 17

II.3.1.2 Fonction de perte . . . 18
II.3.1.3 Optimisation du gradient . . . 18
II.3.1.3.a Descente de gradient stochastique . . . 18
II.3.1.3.b Root Mean Square Propagation . . . 18
II.3.1.3.c Adam . . . 18
II.3.1.4 Rétropropagation . . . 19
II.3.2 Méthodes d'apprentissage . . . 19
II.3.2.1 Apprentissage supervisé . . . 19
II.3.2.2 Apprentissage non supervisé . . . 20
II.3.2.3 Apprentissage semi-supervisé . . . 20
II.3.2.4 Apprentissage par renforcement . . . 21
II.3.2.5 Apprentissage de bout en bout . . . 21
II.3.3 Résultat d'apprentissage . . . 22
II.3.3.1 Sous-apprentissage . . . 22
II.3.3.2 Bon-apprentissage . . . 22
II.3.3.3 Sur-apprentissage . . . 22
II.3.3.3.a L'arrêt précoce de l'apprentissage . . . 24
II.3.3.3.b Batch normalisation . . . 24
II.3.3.3.c Le décrochage ou *Dropout* . . . 25
II.3.4 Exemples d'application . . . 25
II.4 Apprentissage Approfondi . . . 26
II.4.1 Les réseaux de neurones convolutifs . . . 26
II.4.1.1 Couche convolutive . . . 27
II.4.1.2 Couche de Flatten . . . 27
II.4.1.3 Couche de Pooling . . . 28
II.4.1.4 Couche entièrement connectée . . . 29
II.4.2 Réseaux neuronaux récurrents . . . 30
II.5 Conclusion . . . 30

III Technologies et architectures utilisées **31**
III.1 Introduction . . . 31
III.2 Architectures CNN . . . 31
III.2.1 L'architecture du LeNet . . . 31
III.2.2 L'architecture NVIDIA . . . 31
III.3 Simulateur Udacity . . . 33
III.4 Python . . . 35
III.4.1 Pourquoi le choix de Python ? . . . 35
III.4.1.1 Syntaxe lisible . . . 36
III.4.1.2 Communauté . . . 37
III.4.2 Bibliothèques . . . 37
III.4.3 TensorFlow . . . 37
III.4.4 Keras . . . 37
III.4.5 NumPy . . . 38
III.4.6 Matplotlib . . . 38

III.4.7 Pandas . . . 38
III.4.8 OpenCV . . . 38
III.4.9 SciKit-Learn . . . 39
III.4.10 Environnement de travail . . . 39
III.4.10.1 Jupyter Notebook . . . 39
III.4.10.2 Google Colab . . . 39
III.4.11 Environnement d'exécution . . . 39
III.5 Conclusion . . . 40

IV Travail réalisé **41**
IV.1 Introduction . . . 41
IV.2 Simulation d'une voiture autonome . . . 41
IV.2.1 Préparation des données . . . 41
IV.2.1.1 Collection de l'ensemble de données . . . 41
IV.2.1.2 Distribution des données . . . 43
IV.2.1.3 Filtrage des données associées à l'angle 0 . . . 44
IV.2.1.4 Augmentation des données . . . 44
IV.2.1.4.a La fonction agrandissement . . . 45
IV.2.1.4.b La fonction décalage . . . 45
IV.2.1.4.c La fonction luminosité . . . 46
IV.2.1.4.d La fonction de rotation horizontale . . . 47
IV.2.1.4.e La fonction d'augmentation aléatoire . . . 47
IV.2.1.5 Préparation des images . . . 48
IV.2.1.6 Division des données en sous-ensembles d'entraînement et de validation . . . 48
IV.2.2 Création du modèle . . . 49
IV.2.2.1 Architecture NVIDIA . . . 49
IV.2.3 Connexion entre le modèle et le simulateur . . . 51
IV.2.3.1 Création de l'environnement de travail . . . 51
IV.2.3.2 Approche utilisée pour la connexion . . . 51
IV.2.4 Test de modèles . . . 52
IV.2.4.1 Modèle 1 . . . 52
IV.2.4.2 Modèle 2 . . . 52
IV.2.5 Résultats et analyses . . . 53
IV.2.5.1 Modèle 1 . . . 53
IV.2.5.2 Modèle 2 . . . 55
IV.3 Détection de lignes sur une voie droite . . . 57
IV.3.1 Détection sur une image . . . 57
IV.3.1.1 Importation de l'image . . . 57
IV.3.1.2 Traitement de l'image . . . 57
IV.3.1.3 Délimitation de la région d'intérêt . . . 58
IV.3.1.4 Identification des lignes les plus adéquates . . . 59
IV.3.1.5 Superposition du résultat obtenu avec l'image initiale . . . 61
IV.3.2 Détection sur une vidéo . . . 61

IV.4 Classification des panneaux de signalisation 62
IV.4.1 Importation des données 62
IV.4.2 Construction du modèle 63
IV.4.3 Test du modèle 64
IV.5 Conclusion 64

Conclusion générale **66**

Bibliographie **67**

Webographie **70**

Table des figures

I.1 Niveaux d'autonomie de véhicule selon la *SAE* [5] 4
I.2 Fonctionnement d'une voiture autonome [17] 6
I.3 Résumé graphique de la stéréovision [17] 7
I.4 Capteurs lidar [17] 8
I.5 Résumé graphique du lidar [17] 9
I.6 Résumé graphique du radar [17] 9
I.7 Résumé graphique du sonar [17] 10

II.1 Les domaines de l'intelligence artificielle [26] 14
II.2 Types d'apprentissage automatique [28] 14
II.3 Architecture d'un neurone biologique et artificiel [31] 15
II.4 Architecture d'un réseau de neurones artificiel [33] 15
II.5 Fonction ReLU [36] 16
II.6 Fonction eLU [36] 17
II.7 Fonction softmax [36] 17
II.8 Fonction sigmoid [36] 18
II.9 Formules de l'optimiseur Adam [39] 19
II.10 Aperçu de l'apprentissage supervisé [43] 19
II.11 Aperçu de l'apprentissage non supervisé [43] 20
II.12 Aperçu de l'apprentissage semi-supervisé [43] 21
II.13 Aperçu de l'apprentissage par renforcement [43] 21
II.14 Apprentissage de bout en bout [48] 22
II.15 Représentation des résultats d'apprentissage [50] 23
II.16 L'erreur entre les données de test et de training [51] 23
II.17 Illustration de l'arrêt précoce [52] 24
II.18 Batch Normalisation [54] 25
II.19 Illustration du mécanisme de dropout [55] 25
II.20 Architecture d'un CNN [60] 27
II.21 Exemple de couche de convolution [62] 27
II.22 Exemple de mise à plat d'une l'image [63] 28
II.23 Max pooling [63] 28
II.24 Mean pooling [63] 28
II.25 Illustration de couches entièrement connectées [64] 30
II.26 Architecture d'un RNN [66] 30

III.1 Architecture LeNet [60] . . . 32
III.2 Système d'apprentissage de conduite selon NVIDIA [68] . . . 32
III.3 Architecture NVIDIA [67] . . . 33
III.4 Écran de configuration initiale du simulateur . . . 34
III.5 Choix du mode et du Track . . . 35
III.6 Piste 1 du simulateur . . . 36
III.7 Piste 2 du simulateur . . . 36

IV.1 Approche utilisée pour la simulation . . . 42
IV.2 Illustration de 3 images prises, à un moment donné, par le simulateur . . . 43
IV.3 Extrait du fichier driving_log.csv . . . 43
IV.4 Répartition de notre ensemble de données selon l'angle de braquage . . . 44
IV.5 Nouvelle répartition des données de l'ensemble de données . . . 45
IV.6 Application de fonction zoom . . . 45
IV.7 Application de fonction pan . . . 46
IV.8 Application de la fonction de luminosité . . . 46
IV.9 Application de la fonction rotation horizontale . . . 47
IV.10 Augmentation aléatoire d'image . . . 48
IV.11 Prétraitement de l'image . . . 49
IV.12 Résumé de l'architecture NVIDIA . . . 50
IV.13 Approche utiliséé pour la connexion du modèle au simulateur . . . 52
IV.14 Processus de prédiction de l'angle de braquage [68] . . . 52
IV.15 Valeurs du loss et de l'accuracy à chaque époque du modèle 1 . . . 53
IV.16 Représentation du loss à chaque époque du training du modèle 1 . . . 53
IV.17 Représentation de l'accuracy à chaque époque du training du modèle 1 . . . 54
IV.18 Simulation du modèle 1 sur le circuit de training . . . 54
IV.19 Simulation du modèle 1 sur le circuit de validation . . . 54
IV.20 Valeurs du loss et accuracy à chaque epochs du modèle 2 . . . 55
IV.21 Représentation de l'accuracy à chaque époque du training du modèle 2 . . . 55
IV.22 Représentation du loss à chaque époque du training du modèle 2 . . . 56
IV.23 Simulation du modèle 2 sur le circuit de training . . . 56
IV.24 Simulation du modèle 2 sur le circuit de test . . . 56
IV.25 Image de route avec une voie droite [83] . . . 57
IV.26 Résultat de l'exécution de la fonction canny . . . 58
IV.27 Masque utilisé . . . 58
IV.28 Région d'intérêt . . . 59
IV.29 Lignes de voie optimisées . . . 60
IV.30 Détection des lignes de voie sur image . . . 61
IV.31 Partie du fichier signnames.csv . . . 63
IV.32 Résumé du modèle modifié de LeNet . . . 65
IV.33 Image test du modèle LeNet [86] . . . 65
IV.34 Résultat de la prédiction du modèle LeNet . . . 65

Liste des abréviations

ACC	*Adaptive Cruise Control*
ANN	*Artificial Neural Network*
CNN	*Convolutional Neural Network*
ELU	*Exponential Linear Unit*
FPS	*FRames Per Secound*
GPU	*Graphical Processing Unit*
IA	*Intelligence Artificielle*
IEEE	*Institute of Electrical and Electronics Engineers*
LIDAR	*LIght Detection And Ranging*
ML	*Machine Learning*
RADAR	*RAdio Detection And Ranging*
ReLU	*Rectified Linear Unit*
RNN	*Recurrent Neural Network*
SAE	*Society of Automotive Engineers*
SONAR	*SOund Navigation Ranging*
SUV	*Sport Utility Vehicle*
V2I	*Vehicle to Infrastructure*
V2V	*Vehicle to Vehicle*

Introduction générale

Chaque année, il y a plus de 1,3 million de morts à cause des accidents de la route. À ce rythme, le nombre de décès atteindra les 2,4 millions, en 2030 [1]. Ces accidents sont principalement causés par le facteur humain : non-respect du code de la route, utilisation du téléphone portable durant la conduite, somnolence, ...

Face à ce constat alarmant, les constructeurs automobiles tentent de développer des solutions pour réduire le risque d'accidents tout en préservant le confort du conducteur. C'est l'un des facteurs qui les a poussés à s'intéresser à la voiture autonome (qui a pour caractéristique principale de minimiser l'implication du conducteur) mais aussi en raison de son potentiel impact économique, social et même écologique.

Pour ce mémoire de projet de fin d'études, nous nous proposons d'étudier et de simuler une voiture autonome par contrôle de son angle de braquage. Pour cela, nous utiliserons le simulateur de Udacity et des techniques d'apprentissage approfondi. Plus précisément, nous emploierons un réseau de neurones convolutifs car nous allons avoir à traiter des images issues du simulateur. Nous nous aiderons de techniques d'augmentation de données pour permettre à la voiture de généraliser à un route inconnue son comportement acquis durant la période d'apprentissage.

Ce mémoire est organisé en cinq chapitres. Nous commencerons, dans le premier chapitre, par introduire la notion de voiture autonome. Nous y verrons, entre autres, comment ces voitures ont évolué dans le temps jusqu'à leur émergence avec l'arrivée de l'intelligence artificielle. Ensuite, dans le deuxième chapitre, nous introduirons différentes notions liées à l'intelligence artificielle et à l'apprentissage approfondi en mettant l'accent sur les notions utilisées pour ce projet de fin d'études comme les réseaux de neurones convolutifs et l'apprentissage de bout en bout. Dans le troisième chapitre, nous détaillerons les multiples technologies utilisées telles que les bibliothèques logicielles et les architectures de réseaux de neurones convolutifs. Enfin, dans le quatrième chapitre, nous aborderons le cœur de notre travail : la simulation d'une voiture autonome utilisant l'apprentissage approfondie. Nous déraillerons l'ensemble des étapes suivies dans la création d'un modèle optimal prédisant l'angle de braquage de la voiture. Puis, nous analyserons les résultats que nous avons obtenu sur simulateur. Nous finirons ce chapitre par implémenter deux codes additionnels liés à la voiture autonome qui pourrons être utilisés dans des travaux de réalisation de voitures réelles : la détection des lignes de voie sur une route droite et la classification des panneaux de signalisation.

Mots-clés : Voiture autonome, apprentissage en profondeur, CNN, intelligence artificielle, simulateur Udacity, vision par ordinateur.

Chapitre I

État de l'art sur les voitures autonomes

I.1 Introduction

Ce chapitre est constitué de six sections. D'abord, nous introduirons le concept de voiture autonome. Nous expliquerons par la suite les différents niveaux d'autonomie. Puis, nous verrons l'évolution de ces voitures. Ensuite, nous verrons le fonctionnement de ces voitures et à leurs perspectives d'évolution. Enfin, nous citerons les avantages, inconvénients et même les obstacles que rencontre la popularisation des voitures autonomes.

I.2 Définition du terme «voiture autonome»

La voiture autonome, du grec *autonomos* (où qui se régit par ses propres lois ou qui agit de soi-même) [2], désigne une «voiture équipée d'un système de pilotage automatique qui lui permet de rouler sans intervention humaine dans des conditions de circulation réelles> [3]. Une voiture est autonome si la plupart ou la totalité des opérations de conduite sont effectués sans un contrôle actif ou une surveillance continue par un être humain [4].

I.3 Classification des voitures autonomes

La *SAE* International [1] a mis en place une échelle de 6 niveaux dite «échelle d'autonomie» pour classifier les voitures autonomes. Cette échelle, figure I.1, est décrite comme suit [5] :

I.3.1 Niveaux 0, 1 et 2 : Les fonctionnalités d'aide à la conduite

Les niveaux 0, 1 et 2 ont en commun le contrôle permanent de la conduite par l'être humain même si des fonctionnalités d'aide à la conduite sont engagées.

Les trois premiers niveaux se distinguent les uns des autres par les fonctionnalités suivantes :

- **Niveau 0** : Le niveau 0 englobe les véhicules ayant des fonctionnalités envoyant des signaux de danger et d'aide temporaire.

[1] La *Society of Automotive Engineers International* est leader en matière de mise en contact et de formation des ingénieurs tout en promouvant, développant et faisant progresser l'ingénierie aérospatiale, automobile et des véhicules utilitaires

- **Niveau 1** : Les véhicules du niveau 1 ont une fonctionnalité d'assistance de conduite : soit le contrôle latéral (contrôle d'angle) ou longitudinal (contrôle de vitesse).
  Une voiture de ce niveau possède soit un régulateur de vitesse adaptatif (aussi appelé *ACC* pour *Adaptive Cruise Control*) par exemple soit le contrôle de la direction.
- **Niveau 2** : Les véhicules du niveau 2, possèdent les deux fonctionnalités d'assistance de conduite : le contrôle latéral et longitudinal. Donc, le véhicule est équipé du régulateur de vitesse adaptatif et du contrôle de la direction.

I.3.2 Niveaux 3, 4 et 5 : Les fonctionnalités de conduite automatique

Les véhicules de niveau supérieur (3, 4 et 5) ont comme particularité commune que la personne assise sur le siège conducteur ne conduit pas lorsque les fonctionnalités de conduite automatique sont engagées.

Le niveau 3 d'autonomie se différencie des autres niveaux du fait que si la fonctionnalité le demande, l'humain doit conduire. Alors que chez les véhicules de niveau 4 et 5, les fonctionnalités d'aide de conduite automatique ne demanderons pas à l'humain de reprendre la main.

La conduite autonome des niveaux 3 et 4 ne se fait que si toutes les conditions nécessaires sont réunies.

Les fonctionnalités de chacun de ces niveaux sont ainsi :

- **Niveau 3** : Les véhicules de ce niveau peuvent être doté de la fonction de *trafic jam chauffeur* (ou chauffeur dans les embouteillages) permettant le contrôle automatique du véhicule uniquement si la vitesse est modérée.
- **Niveau 4** : Comme exemple de véhicules de niveau 4, on peut citer les taxis locaux sans chauffeur dans un circuit prédéfini. La *SAE* précise que les pédales et le guidon peuvent être installés ou non dans les véhicules de ce niveau.
- **Niveau 5** : Les véhicules du niveau 5 est le plus avancé et opère en présence ou pas d'humains, d'une manière complètement autonome. Le volant ou la pédale ne sont plus nécessaires dans l'habitacle.

I.4 L'histoire des voitures autonomes

Plusieurs événements ont contribué à façonner les véhicules semi-autonomes modernes. Nous présentons dans ce qui suit les étapes majeurs de l'évolution de ces véhicules qui ont mené aux véhicules actuels d'après [6] et [7] sauf mention contraire.

Dès les années 1920, des voitures radiocommandées ont fait l'objet de démonstrations aux États-Unis.

En 1925, l'ingénieur électricien Francis R. Houdina a réalisé une voiture radiocommandée baptisée «American Wonder».

À l'exposition universelle de 1939, une première exposition de voitures guidées a été réalisée et sponsorisé par General Motors.

Dans les années 1950, un système de voiture miniature guidée et contrôlée par des fils électriques au sol est créé en laboratoire par RCA Labs.

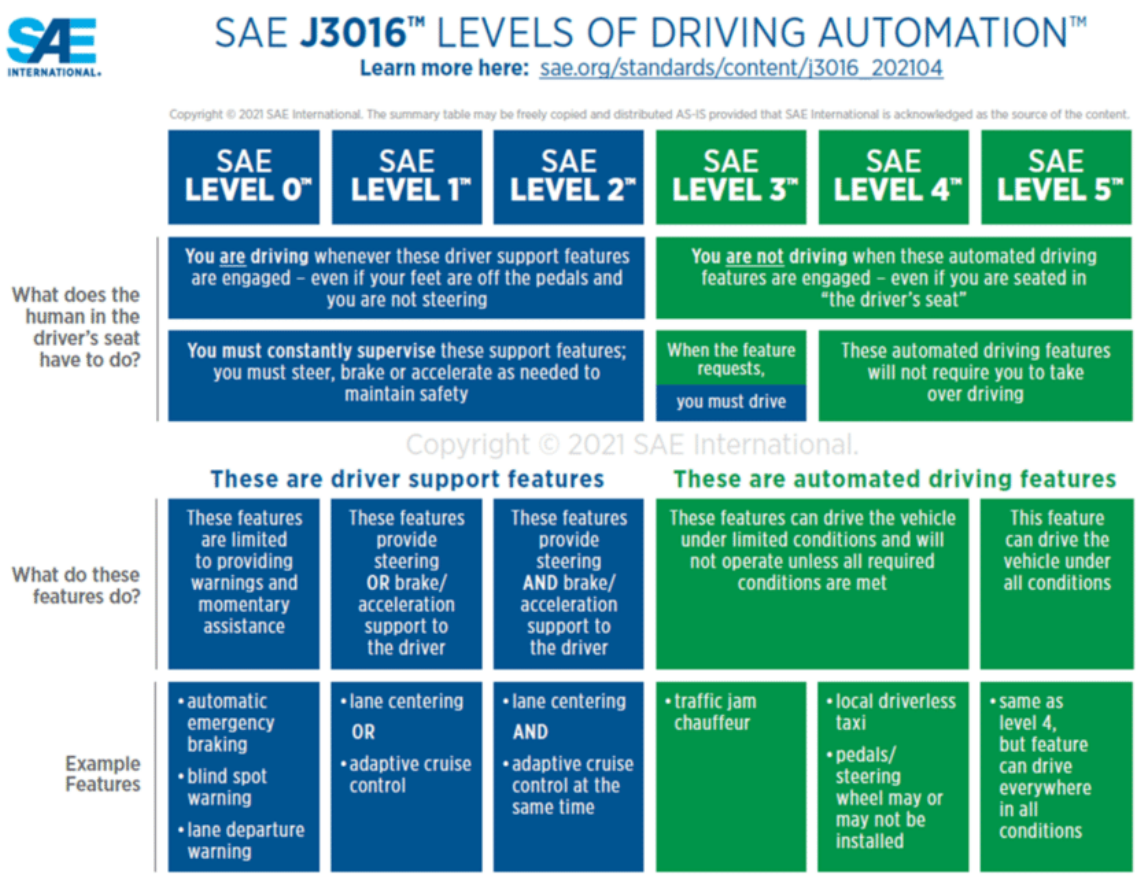


Fig. I.1 – Niveaux d'autonomie de véhicule selon la *SAE* [5]

General Motors a présenté, en 1960, une série de voitures expérimentales appelée Firebird. Ces voitures sont dotées de systèmes de guidage électroniques et sont capables de s'intégrer à la circulation.

Du début au milieu des années 1970, la logique automatisée intelligente fait l'objet de recherches préliminaires menées par l'université de l'Illinois.

En 1991, le congrès américain adopte un projet de loi sur l'autorisation de transport appelé ISTEA [2] qui charge l'USDOT [3] de faire réaliser un véhicule autonome et un système routier automatique avant 1997.

En 1995, le véhicule Mercedes-Benz, semi-autonome, utilisant une vision par ordinateur saccadée pour la conduite en temps réel, parcourt plus de 1590 km aller-retour de Munich à Copenhague souvent à 175 km/h et 95% du temps en conduite en mode autonome.

L'utilisation des réseaux de neurones sur les véhicules Daimler-Benz par l'université Carnegie Mellon permet le contrôle du volant mais, par sécurité, un conducteur actionnait le frein et l'accélérateur.

En 1997, le premier véhicule de transport automatisé ParkShuttle, a fait l'objet de deux projets pilotes aux Pays-Bas. Il n'a ni chauffeur ni steward et n'a ni pédales ni volant ; mais, il a une voie réservée et il cède le pas aux piétons, voitures et autres cyclistes. Ce véhicule vérifie sa position sur la route grâce à des points de référence matérialisés par des aimants

[2] *Intermodal Surface Transportation Efficiency Act*
[3] *U.S. Department of Transportation*, le département des transports américain

sur la chaussée.

En 1998, Toyota adapte un système régulateur de vitesse (ACC) sur un véhicule de série, au Japon.

La voiture BMW Série 3 arrivait à faire le tour du circuit de Hockenheim, en 2006 [8].

En 2008, des véhicules autonomes sont utilisés dans le système de transport minier Australien et notamment avec les véhicules Komatsu.

En 2011, les deux véhicules Spirit of Berlin et MadeInGermany, dirigés par le groupe AutoNOMOS et développés par l'université libre de Berlin, ont assuré le trafic interurbain en Allemagne.

L'état du Nevada est le premier à autoriser l'exploitation des voitures autonomes sur la voie publique dès le 1er mai 2012. Le premier test de conduite y a été fait sur un véhicule de Google puis en Floride et en Californie.

En août 2013, Daimler R&D réalise la Mercedes-Benz de classe S équipée de caméras stéréo et de radars. La Mercedes classe S est disponible à la vente avec les options de : conduite autonome, maintien de la voie, accélération et freinage, stationnement, prévention des accidents.

Dans la même année, Nissan dédie un terrain d'essai à la voiture autonome. Une voiture électrique Leaf est équipée de la technologie de voiture autonome et participe aux démonstrations en Californie.

En 2014, la SAE organise la normalisation, établit le système de classification à 6 niveaux et définit les termes relatifs aux systèmes de conduite automatisée des véhicules routiers.

Encore en 2014, des prototypes de véhicules autonomes avec perception à 360° de l'environnement sont en mesure d'éviter les accidents. Durant la même année et pour la première fois, un prototype BMW roule en mode autonome à Las Vegas prouvant la maîtrise de la conduite en situation critique. Dès lors, les voitures autonomes peuvent se garer en toute autonomie [8].

En mars 2015, Tesla a largement diffusé la capacité de conduite autonome aux voitures existantes avec des mises à jour de son programme [9]. Ces voitures sont semi-autonomes, avec des conducteurs humains censés rester engagés et prendre le relais s'ils détectent un problème potentiel. Il n'est pas encore clair si cette approche semi-autonome où le faite qu'ils sont moins susceptibles de prêter attention à la route, et qui plus est, quand le plus besoin se fait le plus sentir (comme lors des circuits en ville ou avec une météo capricieuse).

En 2017, dans des véhicules autonomes du programme appelé *Trusted Tester Program*, Google Car (rebaptisée Waymo) a commencé à embarquer des passagers, en Arizona [10].

Juillet 2018, Daimler (propriétaire de la marque Mercedes-Benz) et Bosh collaborent pour le lancement de taxis autonomes en Californie pour le deuxième semestre de l'année 2019 [11].

En septembre 2018, Renault-Nissan et Mitsubishi ont proposé trois modèles de voitures autonomes [11].

En 2019, le constructeur japonais Toyota a présenté la navette autonome qui a participé à la fluidification de la circulation dans Tokyo lors des jeux olympiques de 2020 [12].

En 2020, la start-up AutoX fondée par Alibaba a lancé à Shanghai un RoboTaxi complètement autonome [13]. Luminar a, de son côté, signé, avec Volvo Cars, son premier contrat de production de véhicules autonomes pour particuliers [14].

En 2021, Waymo étend son *Trusted Tester Program* à la ville San Francisco [10].

Les véhicules autonomes de Google et les voitures semi-autonomes de Tesla circulent aujourd'hui dans les rues de nombreuses villes.

Le 27 avril 2022, les entreprises Baidu et Pony ont reçu l'autorisation d'effectuer des courses pour particuliers à bord de leurs voitures autonomes sans conducteur dans une zone délimitée à Pékin [15].

I.5 Fonctionnement des voitures autonomes

Les véhicules autonomes, pour circuler et éviter les obstacles, emploient de multiples capteurs, actionneurs, algorithmes, systèmes d'apprentissage automatique et de puissants processeurs :

- Les capteurs radar surveillent la position des véhicules à proximité.
- Les caméras vidéo détectent les feux de circulation, lisent les panneaux de signalisation, suivent les autres véhicules et recherchent les piétons.
- Les capteurs lidar font rebondir les impulsions lumineuses de l'environnement de la voiture afin de déterminer avec précision les distances, détecter les bords de route et identifier les marques sur la voie.
- Les capteurs sonar sont chargés de détecter les bordures de la route et les autres véhicules, notamment lors du stationnement.
- Un logiciel utilise ensuite toutes ces entrées, pour déterminer le chemin à suivre et le comportement à adopter par le véhicule. Ce même logiciel envoie des commandes aux actionneurs qui contrôlent la vitesse, le freinage et la direction de la voiture [16].

La figure I.2 résume ces propos.

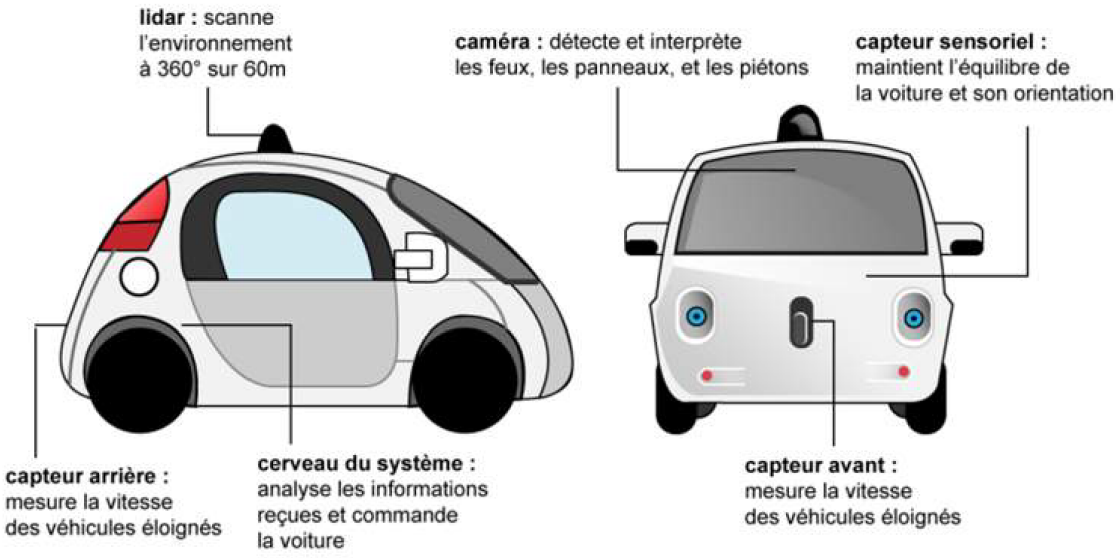


Fig. I.2 – Fonctionnement d'une voiture autonome [17]

Il est possible de retrouver un ensemble de capteurs dans la voiture autonome fournissant les mêmes informations, ce qui laisse penser qu'il existe une certaine redondance dans les données. Cependant, la disponibilité des mêmes informations permet au calculateur de disposer d'informations plus fiables [17]. Les véhicules autonomes combinent, ainsi, différentes

technologies de capteurs afin de complémenter les avantages et minimiser les inconvénients de chacune.

Nous nous intéresserons, dans ce qui suit, à certains des capteurs les plus courants.

I.5.1 Caméras

Les caméras, positionnées à différents emplacements du véhicule, servent à remplacer la vision du conducteur. Une voiture autonome peut être équipée de différents types de caméras : monovision, stéréo, infrarouge voire 3D.

Chaque caméra prend, pendant la conduite, plusieurs clichés à la suite ; ces clichés seront évalués par un algorithme pour une utilisation précise ; les caméras 3D sont utilisées pour localiser et identifier les obstacles et les trottoirs.

La caméra frontale est placée devant afin de détecter les panneaux de signalisation, les véhicules à l'avant et les piétons. Certaines caméras se logent dans les rétroviseurs et à l'arrière de la voiture pour couvrir les angles morts.

Afin d'obtenir une image de son environnement, la caméra 3D émet des rayons infrarouges modulés. Ces rayons seront réfléchis par tout obstacle dont l'image, après mesure du déphasage de la lumière réfléchie, sera en 3D [17].

I.5.2 Stéréovision

La stéréovision consiste à reproduire une image en trois dimensions à l'aide de plusieurs caméras. À cette fin, deux caméras sont généralement utilisées pour visualiser la même scène sous des angles différents. Ensuite, la position relative de chaque point de la scène est calculée à partir des images capturées par les caméras. Cela nécessite un algorithme de traitement d'images pour détecter des objets similaires dans les deux images capturées [18]. La figure I.3 résume les avantages et les inconvénients de la stéréovision.

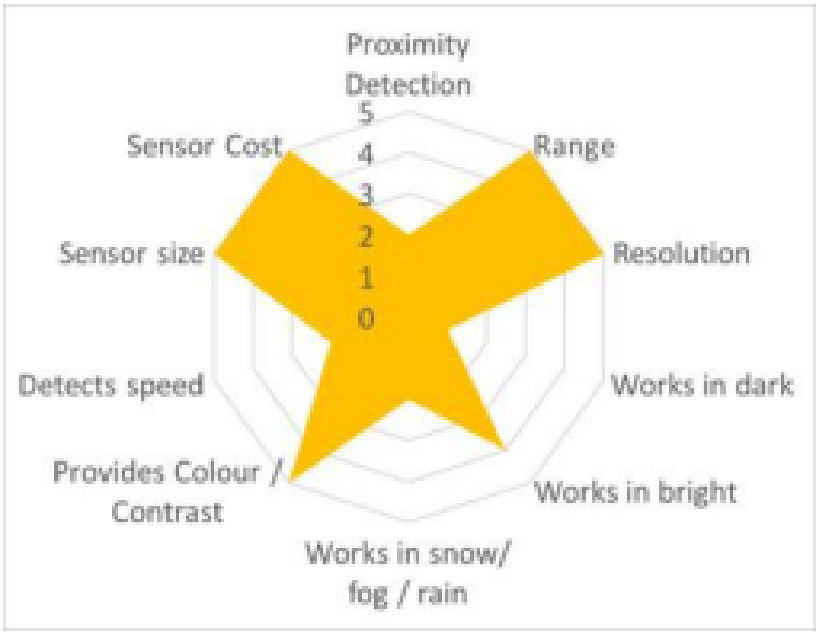


Fig. I.3 – Résumé graphique de la stéréovision [17]

I.5.3 Lidar

Le mot lidar pour *Light Detection And Ranging* ou *Laser Imaging Detection And Ranging* qui peut être traduit par détection et portée laser ou détection et portée de la lumière. Il désigne l'appareil qui mesure la distance entre lui et les objets environnants en utilisant la réflexion de la lumière dans le spectre infrarouge à ultraviolet, comme illustré figure I.4.

Le fonctionnement du lidar se fait en traitent la réflexion des faisceaux laser à l'émetteur dont les rayons sont toujours cohérents. La distance à un objet est calculée grâce aux rayons émis par le laser pulsé, puisque connaissant la vitesse de la lumière, il est possible de calculer la distance de la voiture à d'éventuels obstacles [17].

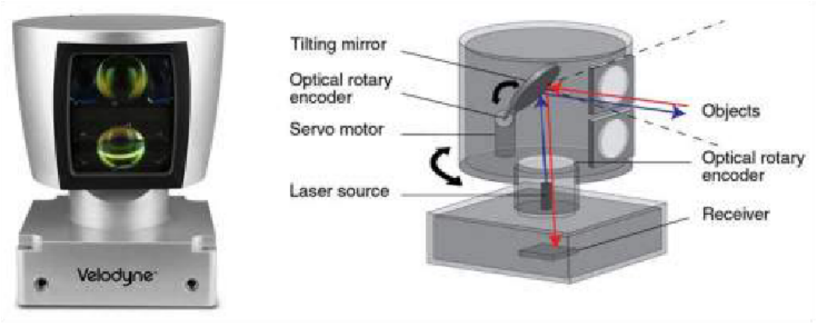


Fig. I.4 – Capteurs lidar [17]

Le *Lidar* est l'un des capteurs les plus importants et les plus coûteux utilisés dans les véhicules autonomes. Ce qui rend ce produit spécial, c'est sa capacité à effectuer des numérisations à grande échelle.

En fait, il peut même balayer plus de 100 mètres autour d'une voiture et générer une carte spatiale 3D précise et instantanée du monde qui l'entoure. Cela signifie que les voitures peuvent utiliser ces données pour prendre des décisions sur la façon de réagir dans différentes situations.

Cependant, la quantité d'informations générées par les conducteurs peut être énorme, ce qui peut être problématique. Dans le cas de la vision par ordinateur, les véhicules autonomes seront équipés de processeurs puissants capables de gérer toutes les informations obtenues en temps réel [17]. La figure I.5 illustre les forces et les faiblesses du lidar.

I.5.4 Radar

Le radar pour *Radio Detection And Ranging* (détection et télémétrie radio), contrairement au lidar, utilise des ondes radio, mais ne peut pas calculer la vitesse des objets autour de la voiture. Cet appareil envoie des ondes sonores de 27 GHz à 77 GHz et selon le même principe que le lidar, utilise le temps de propagation aller-retour, afin de déterminer la vitesse et la distance qui sépare la voiture de l'obstacle. Il existe deux grands types de radars dans le monde de l'automobile :

- D'une part, le radar à courte portée (d'environ 30 mètres) utilisé de basse vitesse.
- D'autre part, le radar à longue portée, (environ 200 mètres), qui est utilisé à grande vitesse.

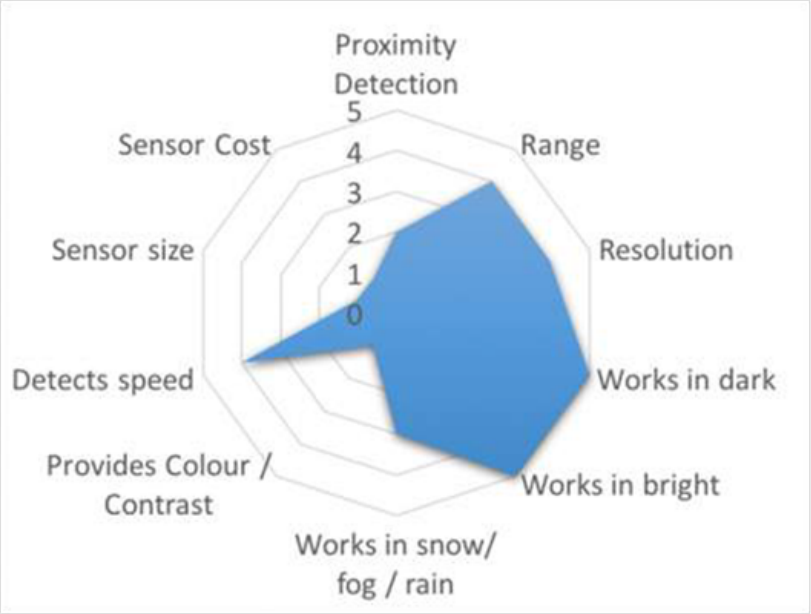


Fig. I.5 – Résumé graphique du lidar [17]

Si deux capteurs sont placés côte à côte, on obtient une cartographie 3D comme le ferait la vision humaine. Le radar capte environ 30 images par seconde qui sont analysées par différents algorithmes [17].

Le radar est l'un des appareils les plus utilisés pour ses nombreux avantages (figure I.6), mais suivant une étude recensée ces effets négatifs sur la santé des personnes exposées ont été mis en évidence (maux de tête, douleurs musculaires, ...) [19].



Fig. I.6 – Résumé graphique du radar [17]

I.5.5 Sonar

Le sonar (pour *Sound navigation Ranging*) utilise le même principe d'écholocation associé au Radar. Le sonar transmet des ondes sonores en dehors de la plage reconnaissable par l'oreille humaine. Il se compose d'un émetteur et d'un récepteur. Et grâce à l'écho des ondes,

il est capable de créer un espace tridimensionnel autour du véhicule permettant ainsi de détecter efficacement les obstacles. Le sonar est couramment utilisé dans les environnements marins car ses ondes sonores se propagent plus rapidement dans l'eau. C'est pourquoi, il n'opère que sur de courtes distances à l'air libre [17]. La figure I.7 résume les avantages et les inconvénients du sonar.

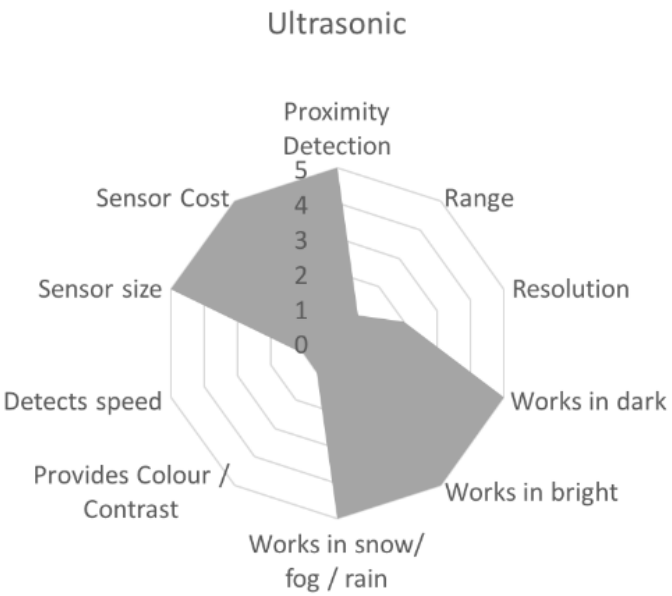


Fig. I.7 – Résumé graphique du sonar [17]

I.5.6 Centrale inertielle

La centrale inertielle est un équipement constitué de trois gyromètres et de trois accéléromètres. Le rôle des gyromètres est de calculer la vitesse et la position du véhicule. Tandis que les accéléromètres déterminent les trois composantes vectorielles spécifiques, à savoir le vecteur déplacement dans un repère tridimensionnel classique. La centrale inertielle joue le même rôle que le GPS. Cependant, elle ne nécessite pas de se connecter à Internet pour son fonctionnement, ce qui peut s'avérer important en cas de panne de la fonction GPS due au problème de connexion à Internet [17].

I.5.7 Odomètre

Appareil qui mesure la vitesse d'une voiture en fonction de la vitesse de la roue ainsi que du nombre de tours effectués par celle-ci. Sur la base de ces informations, cet appareil peut déterminer la distance parcourue par la voiture [17].

I.6 Perspectives d'évolution des voitures autonomes

- D'ici 2023, la filiale AutoX d'Alibaba prévoit d'étendre l'utilisation de son RoboTaxi dans d'autres municipalités en Chine, voire aux États-Unis [13]. De son coté, Baidu envisage de lancer son modèle Jidu Robo-01 [20]. La compagnie Cruise, quant à elle, prévoit de lancer une production en série en 2023 [21].

- D'ici 2024, Jaguar et Daimler prévoient la commercialisation d'un véhicule autonome. Ford prévoit qu'il aura le premier véhicule autonome de masse, mais n'a pas communiqué de date cible précise [22].
- D'ici 2025, la plupart des nouveaux véhicules de la compagnie General Motors seront dotés de fonctions de conduite automatisée ainsi que de la technologie de communication entre véhicules [22].
- D'après *Past reports*, le géant Apple devrait lancer sa propre voiture autonome aux environs de 2025 [23].
- D'ici à 2035, l'institut *Navigant Research* [4] prévoit que les véhicules autonomes gagneront progressivement du terrain sur le marché et que les ventes de ce type de véhicule atteindront 95,4 millions d'unités par an, soit 75 % de toutes les ventes de véhicules légers [22]. La plupart de ces véhicules seront entièrement autonomes [7].
- Les experts membres de l'IEEE (Institute of Electrical and Electronics Engineers) estiment eux que ce même pourcentage (de 75% de véhicules autonomes) ne serait atteint qu'à l'horizon de 2040 [22].

I.7 Avantages, obstacles et inconvénients des voitures autonomes

I.7.1 Avantages des voitures autonomes

Les véhicules autonomes présentent plusieurs avantages, notamment :

- La simplification de la conduite [24].
- La diminution du nombre d'accidents [11].
- L'augmentation de la capacité des routes et la réduction des embouteillages du fait d'une meilleure gestion de la circulation [22].
- Les coûts globaux de transports pourront être réduits de 40% [16].
- Grâce à une circulation plus fluide entraînée par l'avantage précédent, les véhicules autonomes permettront de consommer moins de carburant et de prolonger la durée de vie de leurs composants. Cela induira, à terme, à une diminution de la pollution de l'environnement [22].
- Un véhicule autonome constitue une aide certaine voire une délivrance aux personnes à mobilité réduite, à ceux ayant des déficiences d'acuité visuelle et aux personnes âgées [7], [24].
- Les véhicules autonomes diminueront les problèmes que rencontrent presque tous les citadins : Le stationnement. Ainsi, le véhicule pourra déposer ses passagers suivant leur destination puis d'aller chercher un parking de lui-même. Il pourra revenir récupérer ses passagers quand ceux-ci le demanderont [7].
- Les véhicules autonomes alimentés d'une source énergie renouvelable permettront de réduire de manière drastique la pollution de l'air et les gaz à effet de serre [22].

[4] Aujourd'hui connu sous le nom de *Guidehouse Insights*

I.7.2 Obstacles et inconvénients des voitures autonomes

Malgré des avantages considérables évidents, la voiture autonome rencontre de nombreux obstacles à sa généralisation et à son adoption par le grand public [22] :

- La perte d'emploi dans les métiers où le rôle essentiel est la conduite de véhicules.
- La sensibilité du système de navigation de la voiture aux conditions météorologiques.
- Les indications manuelles des policiers ou lors de travaux temporaires peuvent ne pas être reconnues par les voitures autonomes.
- Des bugs dans les logiciels de conduite autonome peuvent causer des accidents.
- Risques de piratage du véhicule durant le partage d'informations via les protocoles de communication V2V (véhicule à véhicule) et V2I (véhicule à infrastructure).
- Les zones de travaux temporaires qui ne sont répertoriées dans aucune carte ni base de données perturbent la circulation des véhicules autonomes.
- L'intelligence artificielle (IA) embarquée dans les voitures autonomes n'est toujours pas en mesure de gérer les environnements chaotiques des centres-villes.
- Nécessité de mise en œuvre d'un cadre juridique et d'établissement de réglementations gouvernementales pour les voitures autonomes.
- La perte d'emploi des transporteurs, des livreurs et autres travailleurs dont le rôle essentiel est la conduite de véhicules.

I.8 Conclusion

Dans ce chapitre, nous avons introduit la notion de voiture autonome avec ces différents niveaux. Nous nous sommes intéressés, par la suite, à l'évolution de celle-ci. Puis, nous avons expliqué comment fonctionnent la voiture autonome et ces perspectives d'évolution. Et enfin, malgré les nombreux avantages qu'offre cette technologie, plusieurs obstacles et inconvénients retardent sa popularisation.

Le chapitre suivant est consacré à la description des notions générales sur l'intelligence artificielle et l'apprentissage approfondi.

Chapitre II

Notions générales sur l'intelligence artificielle et l'apprentissage approfondi

II.1 Introduction

Ces dernières années, l'apprentissage en profondeur est devenu un domaine d'étude à la mode. Pourquoi en est-il ainsi ? De quoi s'agit-il vraiment ?

Pour répondre à toutes ces questions, ce chapitre présente une description générale sur l'apprentissage en profondeur en expliquant son concept de base en explorant sa pertinence pour l'intelligence artificielle et quelques autres concepts clés utilisés dans la réalisation de ce mémoire et la motivation derrière l'utilisation de ces concepts.

II.2 L'intelligence artificielle

Pendant des milliers d'années, nous avons essayé de comprendre comment nous, humains, pensons ; c'est-à-dire ce qui nous permet de percevoir, comprendre, prédire et manipuler un monde beaucoup plus complexe que notre cerveau. Malgré les grandes inconnues qui demeurent, nous essayons de rendre les machines que nous concevons de plus en plus intelligentes. C'est ainsi qu'est née l'intelligence artificielle (IA), qui est un domaine de l'informatique défini comme l'étude des «agents intelligents» [25].

Il s'agit d'un vaste domaine avec de nombreux sous-ensembles comme indiqué dans la figure II.1 tels que l'apprentissage automatique (*machine learning*) et l'apprentissage en profondeur (*deep learning*) [25].

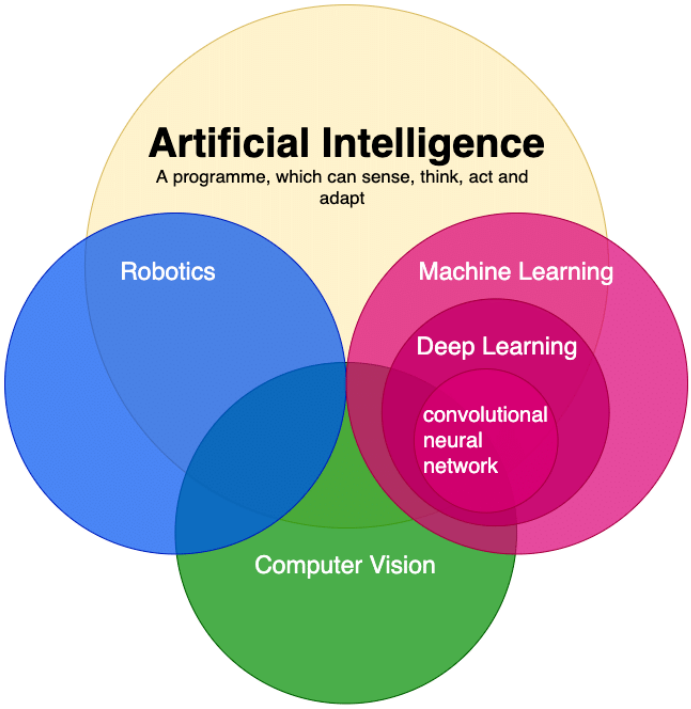


Fig. II.1 – Les domaines de l'intelligence artificielle [26]

II.3 Apprentissage Automatique

L'apprentissage automatique (*machine learning* ou ML) est un domaine dominant de l'intelligence artificielle défini comme «un domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être programmés». L'idée de base du machine learning est de construire des algorithmes capables d'apprendre et d'agir sur un ensemble de données [27].

L'apprentissage automatique peut être divisé en plusieurs types, comme indiqué figure II.2 : l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage semi-supervisé, ainsi que l'apprentissage par renforcement. Chaque type sera décrit plus en détail en II.3.2.

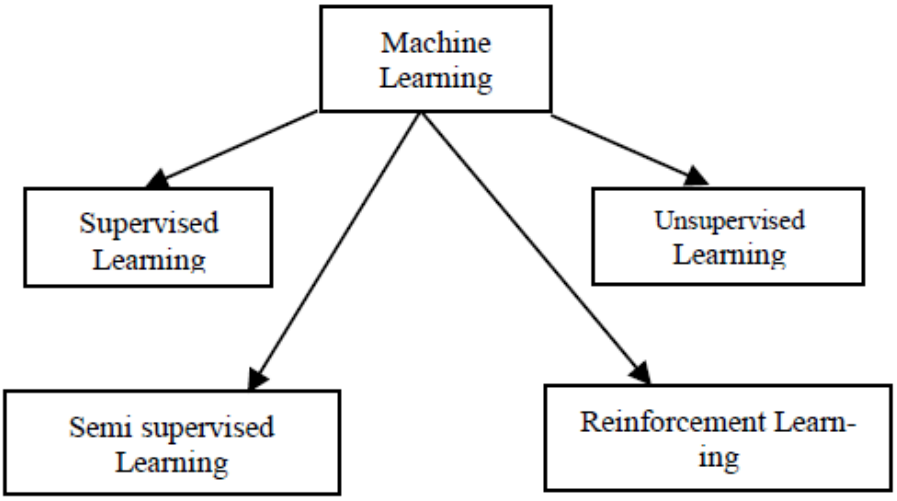


Fig. II.2 – Types d'apprentissage automatique [28]

II.3.1 Les réseaux de neurones artificiels

Un réseau de neurones artificiels (ANN) est composé de plusieurs unités de traitement simples, appelées neurones, interconnectées entre elles [29]. La construction d'un réseau neuronal et de ses structures s'inspire des neurosciences. Ainsi, un neurone d'un ANN est un modèle simplifié d'un neurone biologique [30], comme le montre la figure II.3.

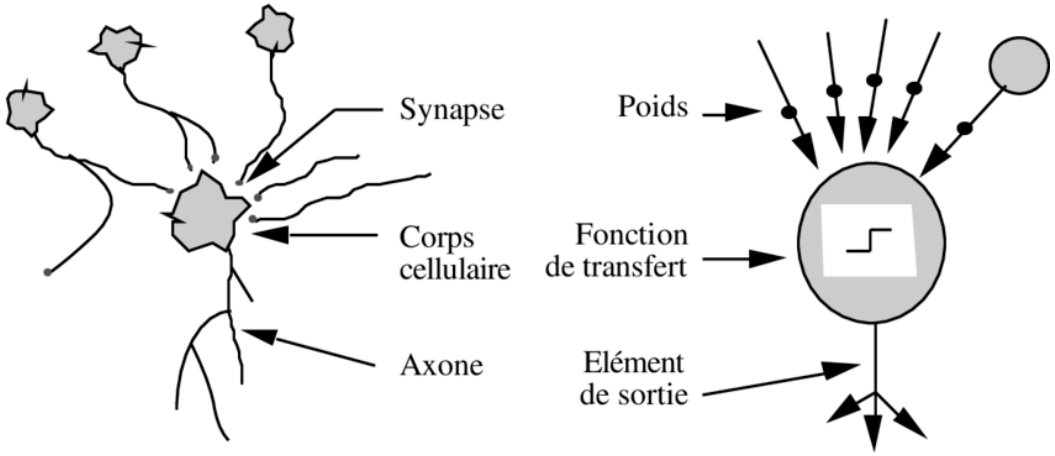


Fig. II.3 – Architecture d'un neurone biologique et artificiel [31]

Chaque neurone de l'ANN reçoit des signaux d'autres neurones ou depuis l'extérieur du réseau. Il applique un poids à chaque signal reçu, et somme le tout avec un biais afin de déterminer la valeur en sortie [32]. Les neurones de sortie de l'ANN peuvent influencer l'environnement en déclenchant des actions. L'apprentissage du réseau est perçu lorsque les pondérations ou poids du réseau sont ajustés de telle sorte que l'ANN se rapproche du comportement souhaité lorsque ces actions sont déclenchées [29]. La figure II.4 illustre un ANN avec à gauche les neurones d'entrée, à droite les neurones de sortie et au milieu les neurones formant la ou les couches cachées.

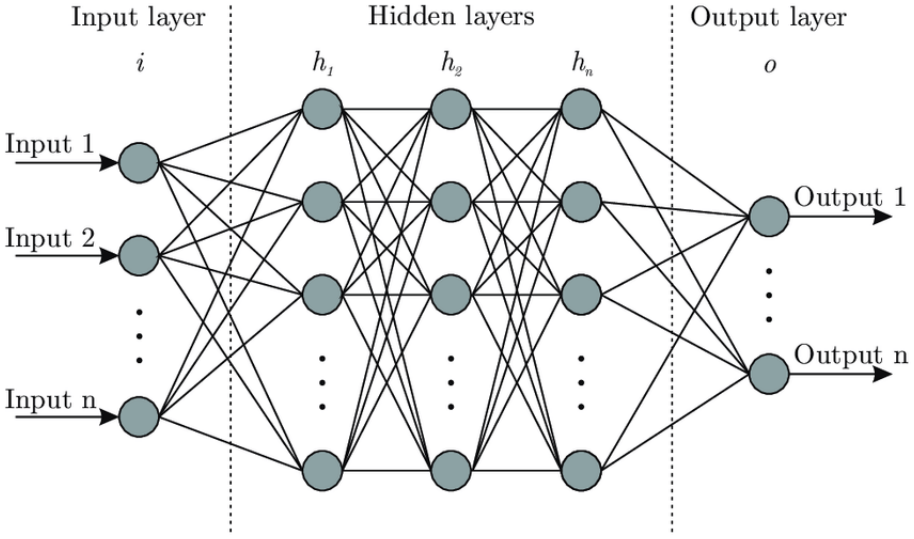


Fig. II.4 – Architecture d'un réseau de neurones artificiel [33]

II.3.1.1 Fonction d'activation

La fonction d'activation représente le potentiel d'activation trouvé dans le domaine de la biologie du cerveau humain. Il s'agit d'une fonction mathématique utilisée sur un signal et permettant la transmission d'informations (ou de non-information) si un seuil déterminé de stimulus en entrée est atteint. Concrètement, son rôle sera de décider de déclencher ou non une réponse neuronale [34]. Il existe plusieurs fonctions d'activation couramment utilisées :

II.3.1.1.a ReLU

La fonction ReLU (pour *Rectified Linear Unit*), illustrée figure II.5, est une fonction réelle linéaire. Elle permet de filtrer les données en annulant toutes les valeurs négatives et en ne laissant passer que les valeurs positives [35]. Elle peut être formulée par l'expression :

$$ReLU(x) = max(x, 0) \tag{II.1}$$

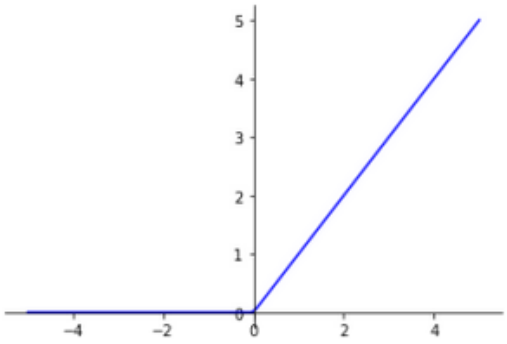


Fig. II.5 – Fonction ReLU [36]

II.3.1.1.b eLU

eLU (ou *Exponential Linear Unit*), figure II.6, est une amélioration de la fonction ReLU en permettant d'avoir des valeurs lisses pour $x < 0$ [36].

$$eLU(x) = \begin{cases} x & \text{if } x \geq 0 \\ \alpha * (exp(x) – 1) & \text{if } x < 0 \end{cases}$$

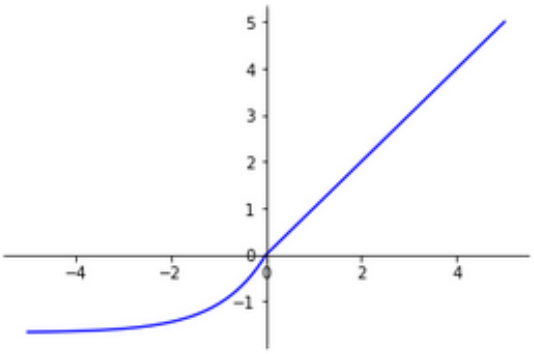


Fig. II.6 – Fonction eLU [36]

II.3.1.1.c Softmax

La fonction Softmax, figure II.7, permet de transformer un vecteur réel en un vecteur de probabilité.

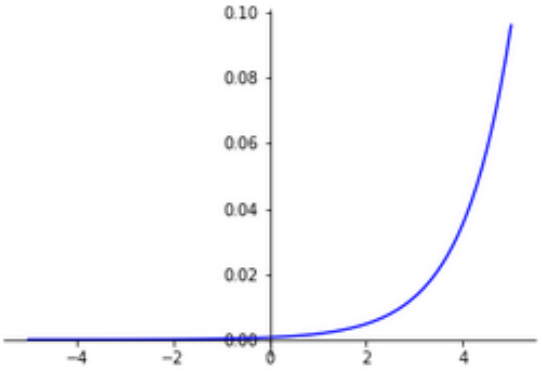


Fig. II.7 – Fonction softmax [36]

II.3.1.1.d Sigmoid

La sigmoïde, figure II.8 donne une valeur comprise entre 0 et 1. Par conséquent, elle est largement utilisée pour la classification binaire, lorsqu'un modèle ne doit définir que deux étiquettes.

$$Sigmoid(x) = 1/(1 + exp(-x)) \tag{II.2}$$

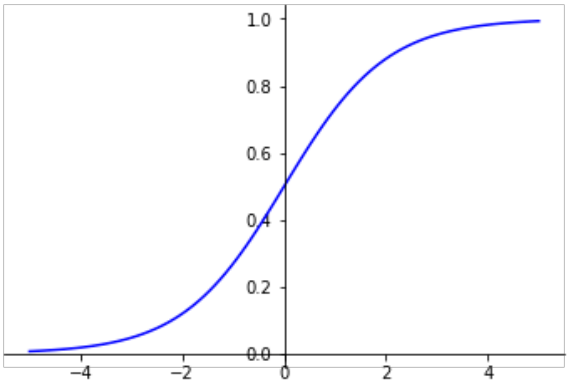


Fig. II.8 – Fonction sigmoid [36]

II.3.1.2 Fonction de perte

Une fonction de perte (*loss function*) est utilisée dans l'apprentissage supervisé pour calculer la différence entre la sortie d'un ANN et la sortie correcte, spécifiée par le superviseur.

II.3.1.3 Optimisation du gradient

La valeur de la fonction de perte définit une direction de recherche du minimum de la fonction de perte basée sur un décalage des poids de l'ANN. La taille du décalage est déterminée par un paramètre appelé taux d'apprentissage [37].

II.3.1.3.a Descente de gradient stochastique

La méthode de descente de gradient stochastique (ou SGD pour *Stochastic Gradient Descent*) a été proposée comme une forme d'optimisation de gradient. L'avantage de SGD est qu'il surpasse les optimiseurs adaptatifs pour un réglage réussi des hyperparamètres. Cependant, l'optimiseur SGD présente plusieurs écueils majeurs, notamment l'absence d'une approche adaptative pour évaluer le taux d'apprentissage optimal pour la phase d'entraînement [38].

II.3.1.3.b Root Mean Square Propagation

La méthode d'optimisation *Root Mean Square Propagation* ou RMSProp est un algorithme de gradient adaptif qui a été ajusté pour fonctionner sur des conditions non-convexes en convertissant le gradient cumulatif en une moyenne mobile fonctionnelle. RMSProp divise le taux d'apprentissage par la moyenne décroissante exponentielle des gradients au carré [38].

II.3.1.3.c Adam

Adam (*adaptive moment estimation*) est un algorithme d'optimisation de gradient de premier ordre pour des fonctions objectives aléatoires, basé sur des calculs adaptatifs des degrés de moments d'ordre inférieur. Il provient de l'estimation du temps d'adaptation. Il convient aux données et paramètres à grande échelle, ainsi qu'à une efficacité de calcul améliorée, à des exigences de mémoire simplifiées, ainsi qu'à une invariance aux gradients diagonaux [38].

La figure II.9 représente l'estimation du moment de premier ordre $(\hat{m}_t)$ et de deuxième ordre $(\hat{v}_t)$. Les équations de descente de gradient avec *Adam* sont :

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

Fig. II.9 – Formules de l'optimiseur Adam [39]

II.3.1.4 Rétropropagation

La technique de rétropropagation applique l'optimisation du gradient en agissant en amont à travers le réseau (c'est à dire de la sortie du réseau vers l'entrée), ajustant le poids du réseau pour compenser l'erreur calculée par la fonction de perte [40].

II.3.2 Méthodes d'apprentissage

II.3.2.1 Apprentissage supervisé

L'objectif de l'apprentissage supervisé (*supervised learning*), figure II.10, est d'apprendre à établir une correspondance entre les entrées et des sorties étiquetées, en telle sorte que le système puisse donner en sortie la bonne valeur apprise pour chaque paire entrée/sortie [41].
La présence d'un superviseur est nécessaire pour fournir les valeurs correctes de la sortie pour une entrée donnée [42].

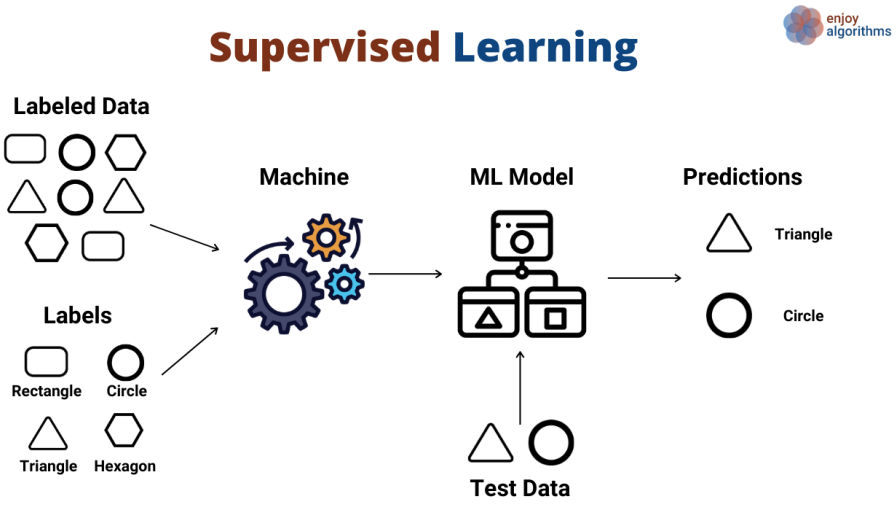


Fig. II.10 – Aperçu de l'apprentissage supervisé [43]

II.3.2.2 Apprentissage non supervisé

Dans l'apprentissage non supervisé (*unsupervised learning*), figure II.11, il n'y a pas de superviseur. Les seules informations accessibles sont les données en entrée [44]. L'objectif de l'apprentissage non supervisé est de découvrir des structures d'intérêt basées sur les données, notamment en y trouvant des modèles réguliers, motifs, groupes, ... L'apprentissage non supervisé est plus généralisable et plus complexe que l'approche supervisée [45].

L'apprentissage consiste en une estimation de densité inconditionnelle, c'est-à-dire un contexte où certains motifs apparaissent beaucoup plus que d'autres motifs à partir des données d'entrée [44].

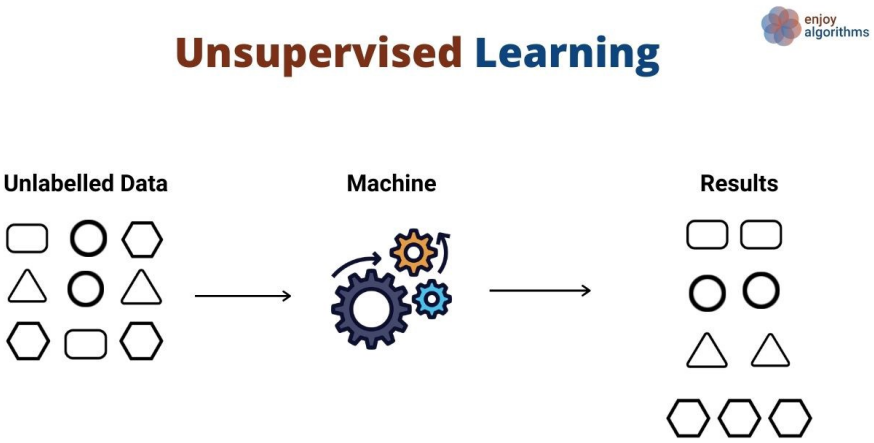


Fig. II.11 – Aperçu de l'apprentissage non supervisé [43]

II.3.2.3 Apprentissage semi-supervisé

L'apprentissage semi-supervisé, figure II.12, consiste à fournir des informations qualitatives (étiquettes de classification) ou des informations quantitatives (valeurs réelles à des fins de régression) à seulement un sous-ensemble de points d'entraînement soumis à l'algorithme d'apprentissage [46].

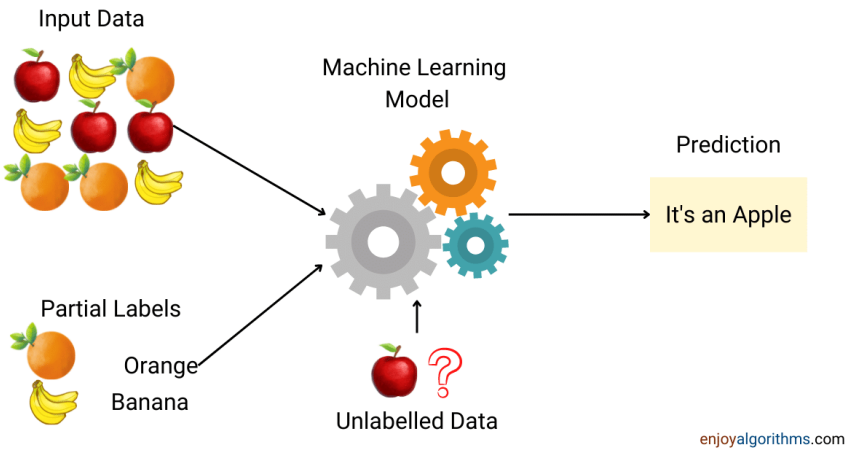


Fig. II.12 – Aperçu de l'apprentissage semi-supervisé [43]

II.3.2.4 Apprentissage par renforcement

L'apprentissage par renforcement, figure II.13, peut être décrit comme un apprentissage supervisé, où un agent intelligent prend des décisions tout en recevant des récompenses ou des pénalités en fonction du résultat de ses décisions. L'agent trouve les décisions qui maximisent sa récompense en exécutant le réseau de façon répétée [44].

Une seule action incorrecte est de moindre importance dans les algorithmes d'apprentissage par renforcement, c'est plutôt une séquence d'actions correctes qui constitue le comportement souhaité de l'algorithme [44].

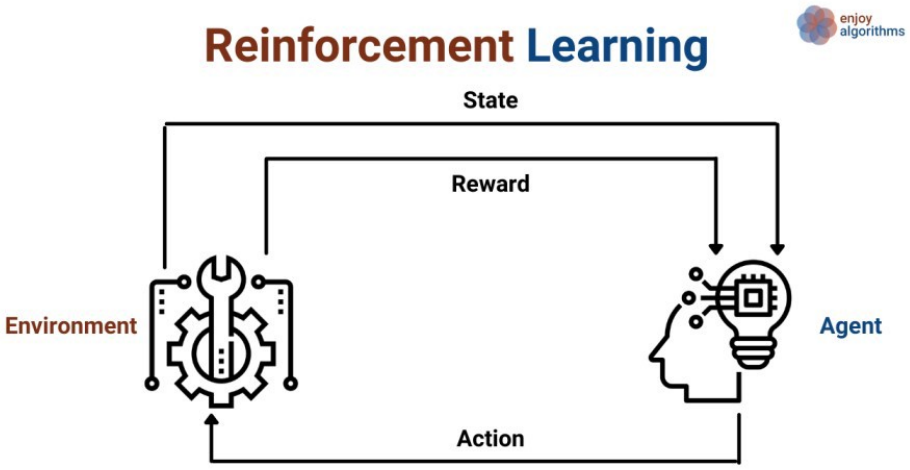


Fig. II.13 – Aperçu de l'apprentissage par renforcement [43]

II.3.2.5 Apprentissage de bout en bout

Dans l'apprentissage de bout en bout (*end-to-end learning*), figure II.14, toutes les étapes requises pour obtenir la sortie résultante du réseau de neurones artificiels sont effectuées au

sein du réseau. Un réseau d'apprentissage qui applique l'apprentissage de bout en bout ne comporte aucune étape intermédiaire ou sous-tâche effectuée en dehors du réseau [47].

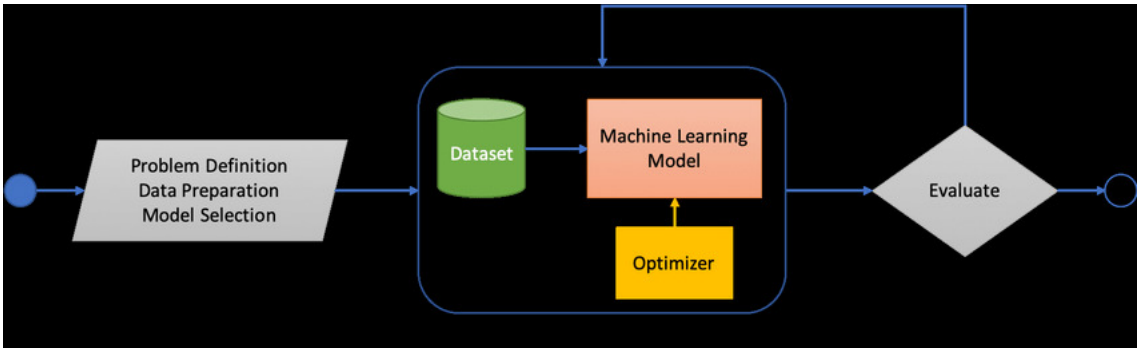


Fig. II.14 – Apprentissage de bout en bout [48]

II.3.3 Résultat d'apprentissage

Le processus d'apprentissage peut aboutir à l'un des trois cas suivant, illustrés figure II.15 et figure II.16 :

II.3.3.1 Sous-apprentissage

On parle de sous-apprentissage ou *underfitting* quand le modèle est incapable de représenter ou regrouper les données de la phase d'apprentissage. C'est à dire qu'il performe mal même sur les données sur lesquelles il a appris [49].

II.3.3.2 Bon-apprentissage

On dit qu'un modèle a fait un bon apprentissage lorsqu'il arrive à généraliser son comportement de manière adéquate sur des données autres que les données de la phase d'apprentissage [49].

II.3.3.3 Sur-apprentissage

Un modèle en sur-apprentissage *overfitting* est un modèle complexe qui apprend exactement les données d'apprentissage. Il ne peut pas généraliser pour les autres données.

Pour éviter l'*overfitting*, il y a plusieurs techniques de régularisation comme la modification aléatoire dans l'ordre des données utilisées pour l'apprentissage à chaque époque (epoch) [49].

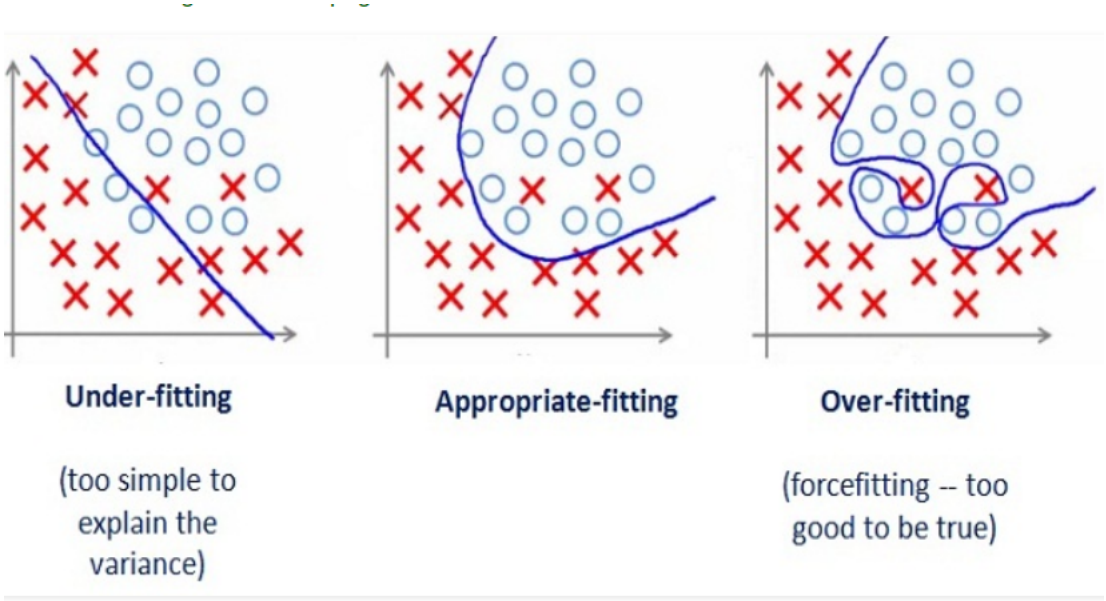


Fig. II.15 – Représentation des résultats d'apprentissage [50]

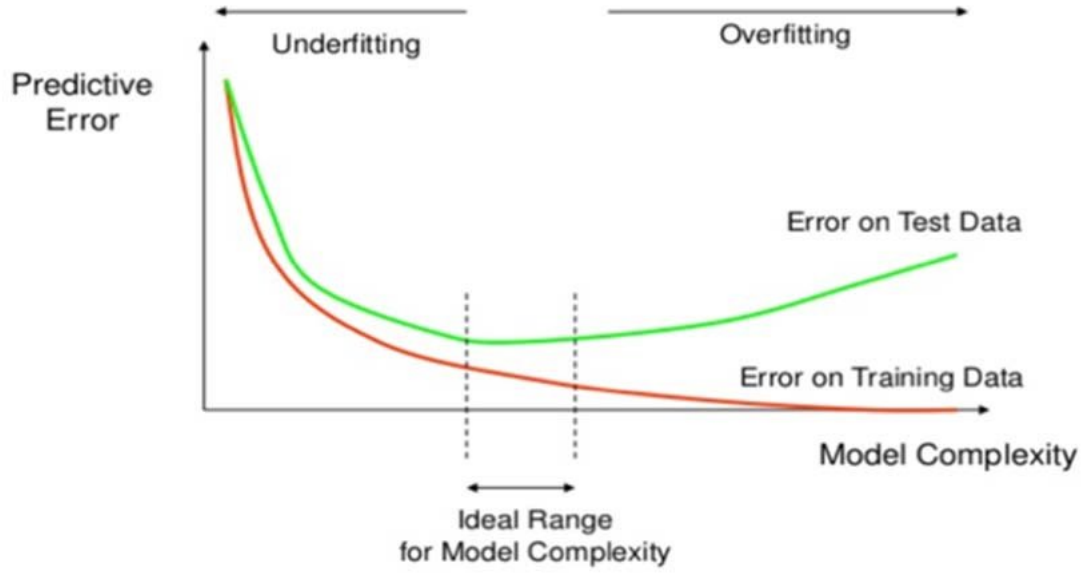


Fig. II.16 – L'erreur entre les données de test et de training [51]

II.3.3.3.a L'arrêt précoce de l'apprentissage

L'arrêt précoce de l'apprentissage (*Early stopping*) consiste à arrêter l'apprentissage avant le risque d'*Overfitting*. Pour cela, on suit l'évolution de l'erreur en considérant les données de validation et on arrête l'apprentissage avant que l'erreur recommence à augmenter [52], comme illustré figure II.17.

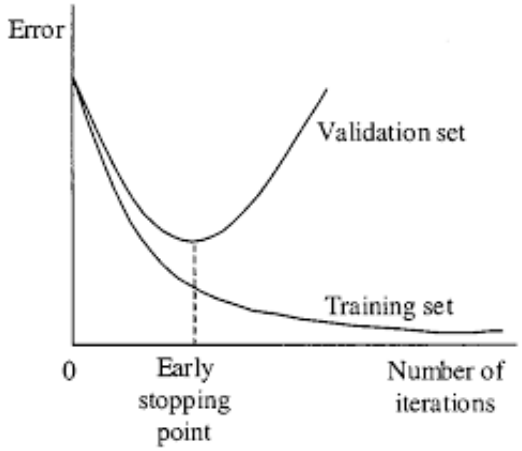


Fig. II.17 – Illustration de l'arrêt précoce [52]

II.3.3.3.b Batch normalisation

L'idée de *Batch normalisation*, figure II.18, est de normaliser les activités des neurones dans les couches intermédiaires pour chaque batch de données, de sorte que les poids d'une couche sont modifiés et auront une moyenne nulle et une variance égale à 1 après l'application du batch [53].

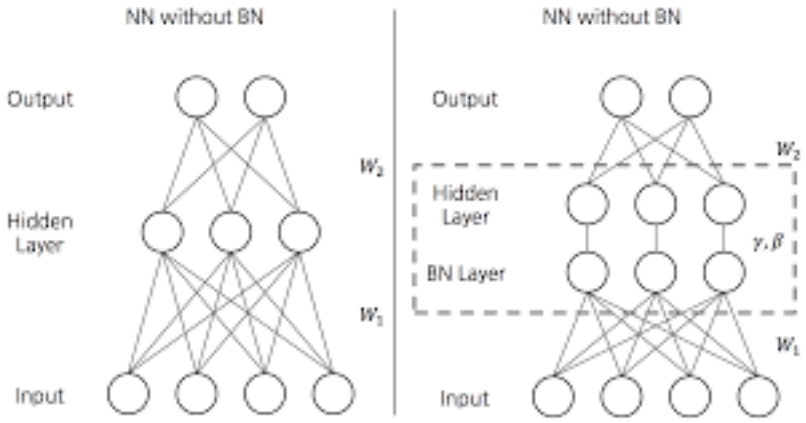
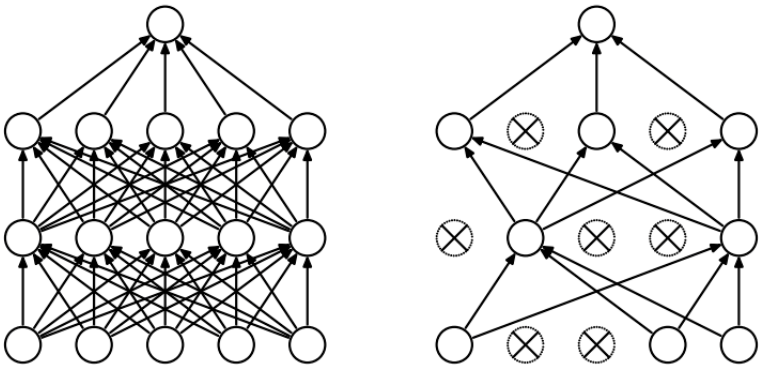


Fig. II.18 – Batch Normalisation [54]

II.3.3.3.c Le décrochage ou *Dropout*

La méthode de *Dropout* réduit le nombre de paramètres du modèle participant effectivement au calcul en déconnectant aléatoirement des neurones, comme illustré figure II.19.

La méthode restaurera toutes les connexions pendant les tests pour s'assurer des performances du modèle sur les données de test [55].



(a) Standard Neural Net (b) After applying dropout.

Fig. II.19 – Illustration du mécanisme de dropout [55]

II.3.4 Exemples d'application

Nous citons ici quelques exemples d'application des méthodes d'intelligence artificielle, d'après [56] :

- Classification : C'est la séparation des données en catégories (ou classes). C'est une méthode d'apprentissage automatique supervisé où la sortie est une grandeur discrète.
- Segmentation : Le *clustering* ou segmentation est une technique d'apprentissage automatique non supervisé qui consiste à regrouper des points de données ayant des caractéristiques similaires.
- Régression : C'est la construction d'un modèle capable de déterminer la valeur de la sortie (grandeur continue) en présence de certaines données.

- Génération : C'est la synthétisation d'images, de vidéos ou d'autres contenus multimédias.
- Contrôle : La commande à base de l'intelligence artificielle comme dans les véhicules autonomes ou la robotique.

II.4 Apprentissage Approfondi

L'apprentissage en profondeur (*deep learning* ou DL) a ouvert un nouveau domaine de possibilités à la fois dans le monde de l'industrie et dans le milieu universitaire de par sa capacité à apprendre des fonctions, des motifs et des représentations non-linéaires complexes.

Les modèles d´apprentissage profond sont utilisés dans toutes sortes de cas d'utilisation et de domaines. Leur puissance stimule la recherche dans plusieurs domaines, notamment la vision par ordinateur, l'analyse audiovisuelle, les chatbots et la compréhension du langage naturel [57].

Élu par le MIT (*Massachusetts Institute of Technology*) comme l'une des 10 plus grandes avancées technologiques en 2013 et déjà utilisé par les grands groupes comme Google ou Facebook pour la reconnaissance d'images ou la prédiction des comportements des utilisateurs en ligne, le DL est de plus en plus utilisé par les entreprises comme un service pour améliorer leur efficacité.

Ersatz Labs, une startup basée à San Francisco, crée une première plateforme commerciale de *deep learning* qui offre aux entreprises la possibilité d'utiliser des algorithmes pour l'analyse de leurs données, dans une version cloud ou via un logiciel spécifique, pour les entreprises qui développent des services d'assistance à la conduite et des voitures autonomes comme Google. Pour ce faire, des algorithmes d'apprentissage automatique sont généralement utilisés sur une grande quantité de données [58].

II.4.1 Les réseaux de neurones convolutifs

Les réseaux de neurones convolutifs (CNN), illustrés dans la figure II.20, sont similaires aux réseaux de neurones en général. L'hypothèse supplémentaire est que l'entrée soit une image (mathématiquement un tenseur ou une matrice tridimensionnelle) ce qui différencie les CNN des réseaux de neurones conventionnels et fait d'eux un choix optimal pour le traitement d'image.

Cette hypothèse explicite nous permet de concevoir des CNN profonds tout en contrôlant le nombre de paramètres pouvant être entraînés [59].

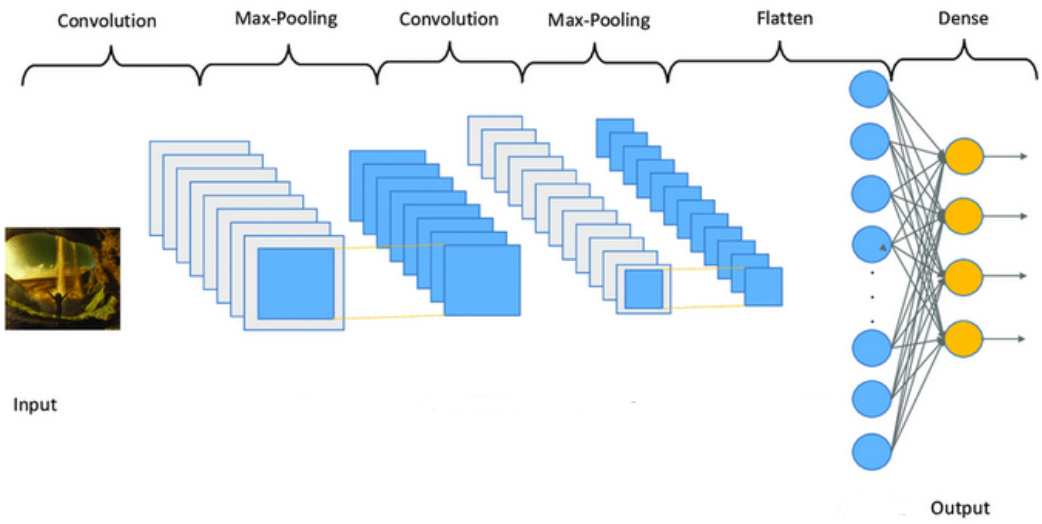


Fig. II.20 – Architecture d'un CNN [60]

II.4.1.1 Couche convolutive

Il s'agit de l'élément clé de différenciation d'un CNN par rapport à d'autres réseaux de neurones.

La couche convolutive, figure II.21, est un ensemble de filtres apprenables. Ces filtres permettent de capturer des entités spatiales. Ce sont généralement de petits motifs mais couvrent toute la profondeur (ou gamme de couleurs) de l'image.

Pendant la passe avant, nous glissons le filtre sur la largeur et la hauteur du l'image lors du calcul du produit scalaire entre les attributs de filtre et l'entrée à toutes positions.

La sortie est une carte d'activation bi-dimensionnelle de chaque filtre, qui est ensuite empilé pour obtenir la sortie finale [61].

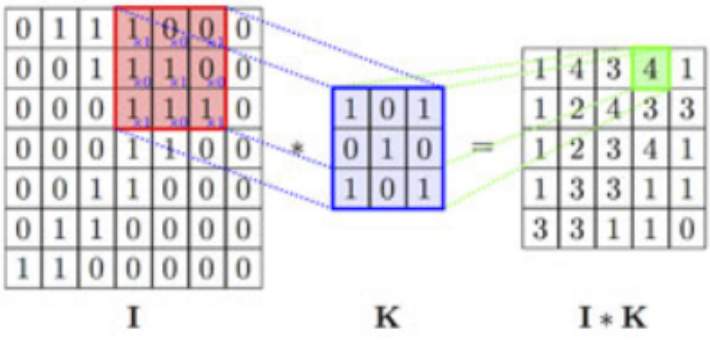


Fig. II.21 – Exemple de couche de convolution [62]

II.4.1.2 Couche de Flatten

Flatten (ou mise à plat) réorganise l'image en forme de vecteur unidimensionnelle [35]. La figure II.22 illustre la transformation d'une image par flatten.

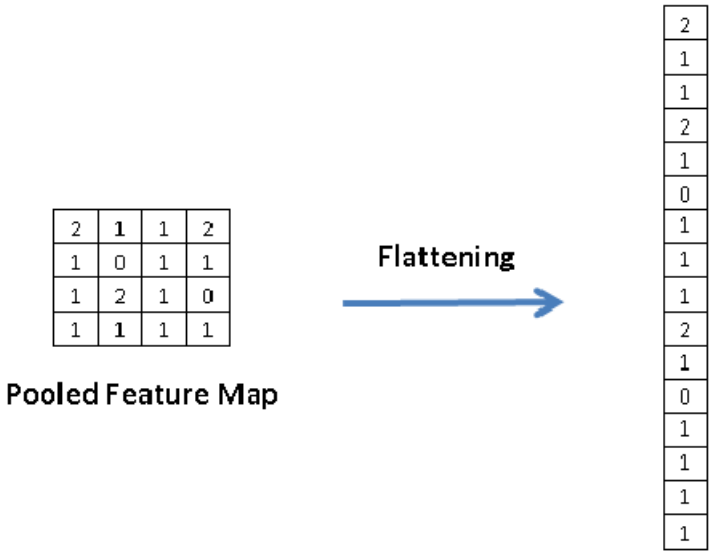


Fig. II.22 – Exemple de mise à plat d'une l'image [63]

II.4.1.3 Couche de Pooling

Le pooling est une opération qui consiste à remplacer un carré de pixels (généralement 2×2 ou 3×3) par une valeur unique. Cette opération permet de diminuer la taille de l'image et de la simplifier (lisser).

Il existe plusieurs types de pooling. On cite :

- *Max pooling* : figure II.23, consiste à sélectionner la valeur maximale de la sélection.

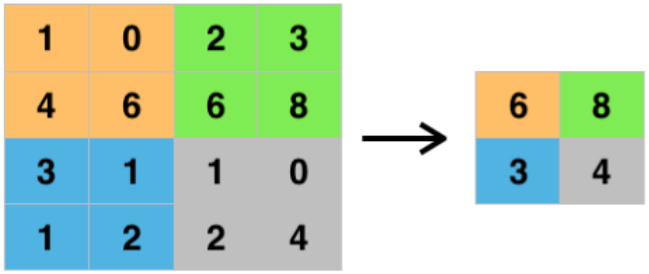


Fig. II.23 – Max pooling [63]

- *Mean pooling* ou *average pooling* : figure II.24, consiste à calculer la somme de toutes les valeurs de la sélection puis la diviser par le nombre de valeurs [35].

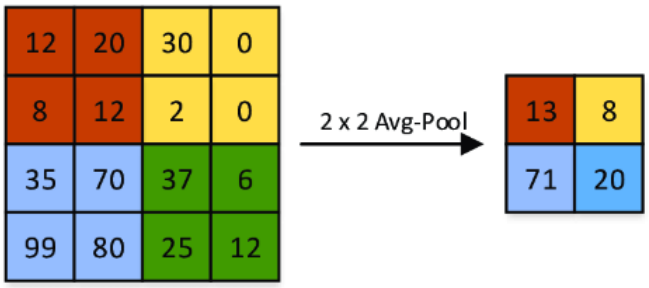


Fig. II.24 – Mean pooling [63]

II.4.1.4 Couche entièrement connectée

Les couches entièrement connectées, figure II.25, sont similaires à la couche entièrement connectée dans les réseaux neuronaux généraux.

Ceux-ci ont des connexions complètes à tous les neurones de la couche précédente. Cette couche permet d'effectuer les tâches de classification [61].

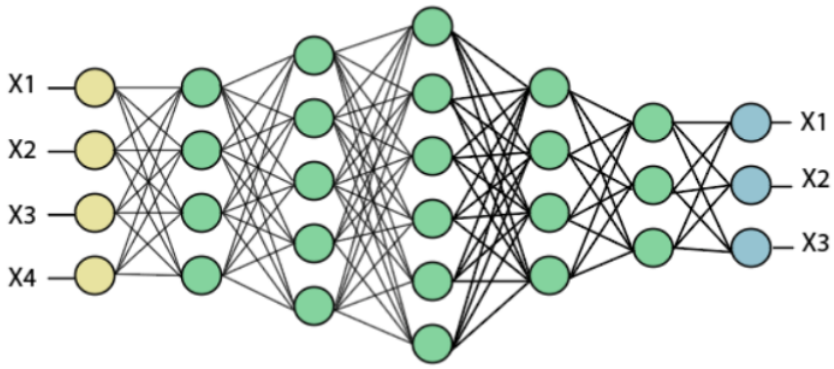


Fig. II.25 – Illustration de couches entièrement connectées [64]

II.4.2 Réseaux neuronaux récurrents

Un réseau neuronal récurrent, connu également sous le nom de RNN, est un type spécial de réseau neuronal artificiel qui permet la persistance d'informations basées sur des connaissances antérieures en utilisant un type spécial d'architecture en boucle, comme indiqué dans la figure II.26. Ce type de réseau neuronal est très utilisé dans les domaines liés aux données avec des séquences comme la prédiction du mot suivant d'une phrase. Les réseaux en boucle sont appelés récurrents car ils effectuent les mêmes opérations et calculs pour chaque élément dans une séquence de données d'entrée. Les RNN ont une mémoire qui les aide à capturer les informations des séquences passées [65].

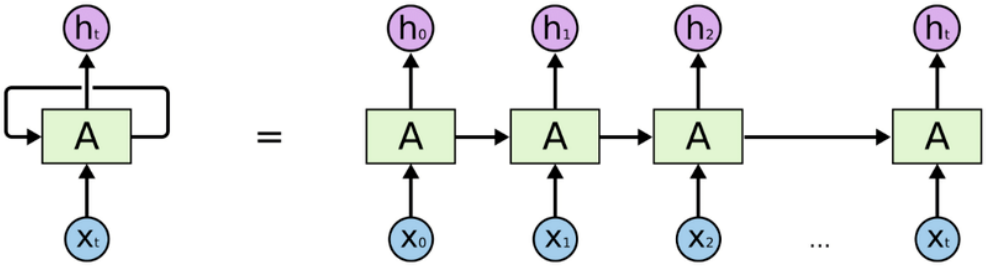


Fig. II.26 – Architecture d'un RNN [66]

II.5 Conclusion

Pour résumer, le Deep Learning s'est récemment développé dans une direction prometteuse en tant que méthode d'intelligence artificielle inspirée de la stimulation du cerveau humain. Il se développe de manière autonome pour apprendre divers ensembles de fonctionnalités complexes telles que le traitement d'images, la reconnaissance vocale et le traitement du langage naturel.

Sa large diffusion est due au développement rapide du domaine d'étude. Par conséquent ; il a un grand impact sur les nouvelles architectures et algorithmes. Nous avons discuté et expliqué le rôle de modèle principale : les réseaux convolutifs.

Il existe encore des sujets de recherche approfondie qui peuvent jouer un rôle majeur dans l'amélioration de la vision par ordinateur.

Chapitre III

Technologies et architectures utilisées

III.1 Introduction

Dans ce chapitre, nous allons voir les différentes technologies et architecture CNN que nous avons employé pour la réalisation de notre projet de fin d'études. Nous aborderons dans un premier temps, les architectures CNN auxquelles nous avons eu recours. Ensuite, nous introduirons le simulateur Udacity qui nous permettra de cloner le comportement d'une voiture. Puis, nous présenterons le langage de programmation «Python» avec ces différentes bibliothèques utilisées. Enfin, nous présenterons les environnements d'exécution Jupyter notebook et Google Colab.

III.2 Architectures CNN

Durant notre projet, nous avons eu à utiliser les deux architectures CNN suivantes : LeNet et le NVIDIA. Les structures de ces dernières sont décrites ci-après :

III.2.1 L'architecture du LeNet

La classification des panneaux de signalisation est basée sur l'architecture *LeNet* illustrée en figure III.1. Cette architecture est composée de :

- Une couche de convolution.
- Une couche de MaxPooling.
- Une couche de convolution.
- Une couche de MaxPooling.
- Une couche flatten.
- Une couche dense (ou couches de neurones entièrement connectées).
- Une couche de sortie (un ou plusieurs neurones selon le besoin).

III.2.2 L'architecture NVIDIA

L'entreprise américaine NVIDIA (réputée pour ses processeurs, cartes et autres puces graphiques pour PC) s'est lancée dans le domaine de l'IA et a conçu une architecture du même nom pour la conduite de voitures autonomes. NVIDIA a conçu un système d'apprentissage d'un modèle prédisant l'angle de braquage d'une voiture autonome, figure III.2 [67].

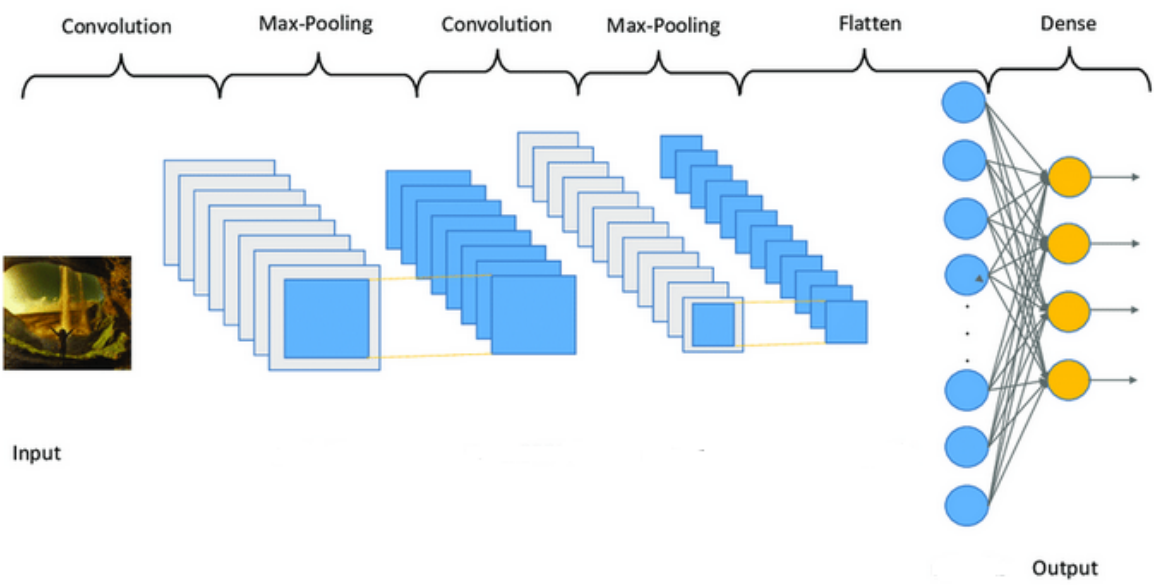


Fig. III.1 – Architecture LeNet [60]

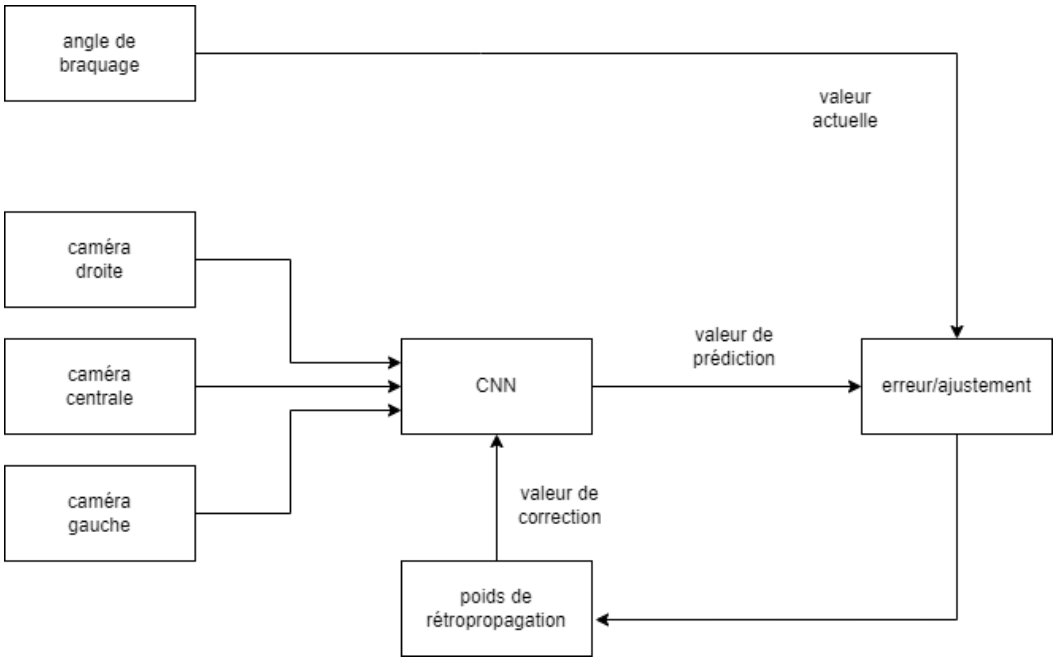


Fig. III.2 – Système d'apprentissage de conduite selon NVIDIA [68]

Il est a noté que le fait que les trois caméras soient reliées au CNN veux dire que : le modèle CNN construit, utilise comme données une seule image issues de l'une des trois caméras à la fois de manière aléatoire.

Ce modèle est basé sur l'architecture NVIDIA, figure III.3 est composée de [67] :

- Une couche de normalisation : permet prétraiter l'image avant son utilisation.
- Une couche de convolution de 24 filtres, de noyau *kernel* 5x5 et de pas *stride* de 2x2 et ayant comme entrée une image de dimension 66x200x3.
- Une couche de convolution de 36 filtres, d'un *kernel* 5x5 et d'un *stride* de 2x2.
- Une couche de convolution de 48 filtres, d'un *kernel* 5x5 et d'un *stride* de 2x2.
- Une couche de convolution de 64 filtres, d'un *kernel* 3x3.

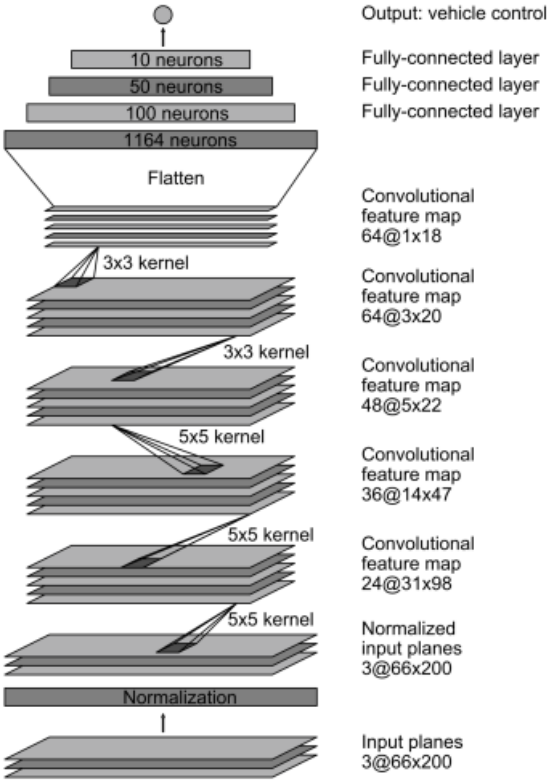


Fig. III.3 – Architecture NVIDIA [67]

- Une seconde couche de convolution de 64 filtres, d'un *kernel* 3x3.
- Une couche flatten qui convertit le résultat de sortie en un vecteur unidimensionnel.
- Trois couches de neurones entièrement connectées : chaque couche étant constituée, respectivement, de 100, 50 et de 10 neurones.
- Un neurone de sortie qui indique la valeur de l'angle de braquage prédit.

III.3 Simulateur Udacity

L'entreprise *Udacity* [1] (spécialisée dans les formations en ligne dans les domaines tels que la programmation ou l'intelligence artificielle) a développé un simulateur de voiture autonome disponible en *open source* et permettant à toute personne de travailler sur un environnement temps réel. Ce simulateur est basé sur la plateforme de jeux vidéo *Unity* [2]. Il en existe deux versions disponibles sur Linux, Mac et Windows. Nous utiliserons dans ce mémoire la version 1 sous le système Windows.

Pour commencer l'utilisation, l'utilisateur doit configurer la résolution et la qualité graphique de la simulation, puis appuyer sur le bouton «*Play !*» comme indiqué Figure III.4.

Le cœur du simulateur *Udacity* (Figure III.5) est constitué de deux pistes (illustrées figure III.6 et figure III.7) et de deux modes : *TRAINING MODE* (pour mode d'entraînement) et *AUTONOMOUS MODE* (pour mode autonome).

La piste figure III.6 est un circuit fermé avec une majorité de virages à gauche sur un terrain plat considéré comme simple sans difficultés majeures. La piste est constituée de différents

[1] https://www.udacity.com
[2] https://unity.com/

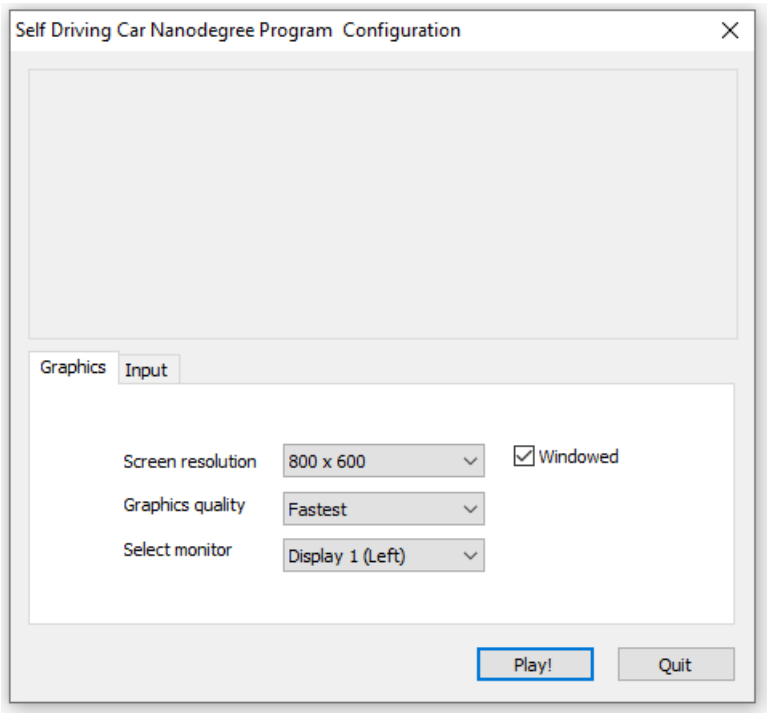


Fig. III.4 – Écran de configuration initiale du simulateur

revêtements et de bords. Ce qui rend cette piste idéale pour l'apprentissage.

La piste figure III.7 est plus difficile à la conduite avec des nombreux virages, côtes et descentes. Cette piste sera utilisée pour la validation du modèle de la voiture autonome.

Le *training mode* permet d'enregistrer la course de l'utilisateur et d'enregistrer le clonage de la conduite nécessaire pour constituer des données d'apprentissage.

L'*autonomous mode* permet de tester le modèle de conduite créé, en communiquant les données de navigation à un serveur web externe au simulateur et qui devra répondre avec les directives de conduite.

En mode d'entraînement, le simulateur permet d'enregistrer des images représentant l'environnement frontalier de la voiture à chaque seconde [69] grâce à trois caméras placées au centre, à droite et à gauche du véhicule.

Pour commencer l'enregistrement, il suffit d'appuyer sur le bouton rouge en haut à droite Figure III.6. Puis, de sélectionner le dossier dans lequel on souhaite sauvegarder les données. Enfin, pour arrêter l'enregistrement, il suffit d'appuyer sur le même bouton rouge. La voiture peut être conduite à l'aide des touches fléchées du clavier.

Le dossier de données résultant de la phase de d'apprentissage est constitué d'un dossier d'images appelé *IMG* et d'un fichier au format CSV [3] appelé *driving_log.csv*. Le dossier *IMG* est constitué de toutes les images enregistrées durant le *training*.

Le fichier *driving_log.csv* représente un tableau regroupant les informations relatives à l'enregistrement de la conduite. Les lignes du tableau représentent les instants d'enregistrement et les colonnes représentent les données suivantes :

- Colonne 1 : le chemin d'emplacement de l'image du centre.
- Colonne 2 : le chemin d'emplacement de l'image de gauche.

[3] *Comma Separated Values*, fichier texte représentant un tableau où les colonnes sont séparées par des virgules



Fig. III.5 – Choix du mode et du Track

- Colonne 3 : le chemin d'emplacement de l'image de droite.
- Colonne 4 : l'angle de braquage.
- Colonne 5 : l'accélération.
- Colonne 6 : la décélération.
- Colonne 7 : la vitesse.

III.4 Python

Python est un langage de programmation de haut niveau créé par Guido van Rossum en 1991. Il s'agit d'un langage de programmation interprété, c'est-à-dire qu'il n'est pas nécessaire de le compiler avant de l'exécuter.

Simple et puissant, le langage python permet d'écrire des morceaux de code très simples et, grâce aux nombreuses librairies disponibles, il est utilisable dans plusieurs domaines tels que : le développement web, l'intelligence artificielle, ... [70]

Ce langage facile est à apprendre : son code est plus facile à lire, donc plus facile à maintenir. Par exemple, il est 5 fois plus concis que le langage Java, ce qui augmente la productivité des développeurs et réduit le nombre d'erreurs [71].

III.4.1 Pourquoi le choix de Python ?

Python est souvent décrit comme l'un des meilleurs langages de programmation pour explorer le domaine de l'intelligence artificielle. En voici les principales raisons :



Fig. III.6 – Piste 1 du simulateur



Fig. III.7 – Piste 2 du simulateur

III.4.1.1 Syntaxe lisible

Python est un langage de programmation orienté objet qui utilise une syntaxe de script moderne et facile.

Conçu avec une lisibilité quasi humaine, la nature de script de Python permet aux codeurs et aux programmeurs de tester leurs hypothèses et d'exécuter leurs algorithmes très rapidement.

C'est pourquoi les langages de programmation structurés comme Java et C++ qui nécessitent un codage en dur ne sont généralement pas privilégiés pour l'apprentissage automatique [72].

III.4.1.2 Communauté

Python offre un support étendu grâce à sa très grande communauté d'utilisateurs, augmentant encore plus la popularité de Python [72].

III.4.2 Bibliothèques

Python est livré avec un grand nombre de bibliothèques. Beaucoup de ces bibliothèques intégrées sont destinées à l'intelligence artificielle et peuvent être facilement employées.

Avant de citer ces bibliothèques il est nécessaire de définir la notion d'API. les APIs (pour *Application Programming Interface*) d'apprentissage automatique sont des services qui fournissent un ensemble de fonctions et de procédures que les développeurs peuvent utiliser. Au cœur de ces fonctionnalités, les API et les workflows d'apprentissage automatique se trouvent des techniques statistiques utilisées pour fournir des informations d'apprentissage dans les programmes dans lesquels ces API sont utilisées [73].

Nous détaillerons dans les sous-sections suivantes les bibliothèques logicielles que nous avons utilisées durant l'élaboration de ce mémoire.

III.4.3 TensorFlow

Il s'agit d'une bibliothèque open source pour le calcul numérique et l'apprentissage automatique compatible avec le langage Python. Il simplifie le processus de collecte de données, de formation de modèles, de génération de prédictions et d'affinement des résultats futurs.

TensorFlow rassemble des modèles et des algorithmes issus du l'apprentissage automatique et du l'apprentissage approfondi. Le langage Python fournit une API d'interface utilisateur pratique et confortable pour créer des applications à l'aide de ce framework. Ces applications peuvent ensuite être exécutées en C++. Ce cadre peut être utilisé pour former et exécuter des réseaux de neurones profonds pour [74] :

- La classification manuscrite des chiffres.
- La reconnaissance d'images.
- L'intégration lexicale.
- La répétition de réseaux de neurones.
- Les modèles de « séquence - séquence » pour la traduction automatique.
- Le traitement du langage naturel.
- La simulation basée sur une équation aux dérivées partielles.

III.4.4 Keras

Keras est une API d'apprentissage en profondeur écrite en Python qui s'exécute sur la plateforme de TensorFlow. Elle a été développée dans le but de permettre des tests rapides. Être

capable de passer de l'idée au résultat le plus rapidement possible étant la clé d'une bonne expérimentation [75].

III.4.5 NumPy

NumPy est le package de base pour le calcul scientifique en Python. Il s'agit d'une bibliothèque Python qui fournit un objet tableau multidimensionnel, divers objets dérivés (tels que des tableaux et des matrices cachés) et une série de procédures pour opérations rapides sur les tableaux, y compris mathématiques, logique, manipulation de forme, tri, sélection, E/S, Fourier discret transformations, algèbre linéaire de base, opérations statistiques de base, simulations stochastiques, etc. [76].

III.4.6 Matplotlib

Matplotlib est une bibliothèque de visualisation et de traçage de données multiplateforme pour Python et son extension numérique NumPy. Par conséquent, elle fournit une alternative open source viable au logiciel propriétaire Matlab. Les développeurs peuvent également utiliser l'API de matplotlib pour intégrer des tracés dans des applications avec interface graphique ou GUI [4].

Un script Python Matplotlib est structuré de sorte que quelques lignes de code suffisent dans la plupart des cas pour produire un tracé visuel des données [77].

III.4.7 Pandas

Pandas est une bibliothèque Python pour l'analyse de données. Lancée par Wes McKinney en 2008 à la recherche d'un outil d'analyse quantitative puissant et flexible, Pandas est devenue l'une des bibliothèques Python les plus populaires. Elle possède une communauté de contributeurs extrêmement actifs. Pandas s'appuie sur deux bibliothèques Python principales, matplotlib pour la visualisation des données et NumPy pour les opérations mathématiques.

Avant Pandas, la plupart des analystes utilisaient d'abord Python pour la gestion et la préparation des données, puis passaient à un langage plus spécifique à un domaine comme R pour le reste de leur travail. Pandas a introduit deux nouveaux types d'objets pour stocker des données qui facilitent les tâches d'analyse et éliminent le besoin de changer d'outil : les Séries et les DataFrames, qui ont, respectivement, une structure de type liste et une structure de tableau semblable à une feuille de calcul [78].

III.4.8 OpenCV

OpenCV (pour *Open Computer Vision*) est une bibliothèque graphique développée par Intel. Elle est spécialisée dans le traitement d'images, aussi bien pour la photo que pour la vidéo.

Elle est disponible sur la plupart des systèmes d'exploitation et existe pour les langages Python, C++ et Java. Elle est aujourd'hui développée, maintenue, documentée et utilisée par une communauté de plus de 40 000 membres actifs !

[4] Graphical User Interface

OpenCV fournit un ensemble de plus de 2500 algorithmes de vision par ordinateur, accessibles au travers d'APIs. Ce qui permet d'effectuer toutes sortes de traitements d'images (extraction de couleurs, détection de visages, de formes, application de filtres, ...).

Ces algorithmes se basent principalement sur des calculs mathématiques complexes, concernant surtout les traitements sur les matrices : on sait qu'une image peut être considérée comme une matrice de pixels [79].

III.4.9 SciKit-Learn

Scikit-Learn est une librairie Python spécialisée en *Data Science*, initialement développée par David Cournapeau dans le cadre du projet *Google summer of code* en 2007. Il s'agit d'une bibliothèque accessible et puissante qui s'intègre naturellement dans l'écosystème plus large des outils de science des données basés sur Python [80].

III.4.10 Environnement de travail

III.4.10.1 Jupyter Notebook

JupyterLab est un environnement de développement interactif basé sur le Web pour les blocs-notes, le code et les données. Son interface flexible permet aux utilisateurs de configurer et d'organiser des flux de travail en science des données, en informatique scientifique, en journalisme informatique et en apprentissage automatique [81].

III.4.10.2 Google Colab

Colab ou «Colaboratory» nous permet d'écrire, exécuter et partager du code Python dans le navigateur avec accès gratuit à des GPU sans configuration requise.

Colab permet, par exemple, d'importer un ensemble de données d'images, d'entraîner un classificateur d'images et d'évaluer le modèle généré, avec seulement quelques lignes de code. Les notebooks Colab exécutent ce code sur les serveurs cloud de Google. Nous avons donc à notre disposition toute la puissance du matériel Google, y compris les GPU et TPU [5], quelle que soit la puissance de notre ordinateur. Un navigateur et une connexion internet suffisent [82].

III.4.11 Environnement d'exécution

Nous avons utilisé un ordinateur portable doté :

- D'un processeur Intel i5-5200U ayant une fréquence de 2.20GHz.
- D'une RAM de 8Go.
- D'un système d'exploitation Windows 10 sur une architecture 64bits.

[5] Les *Tensor Processing Unit* sont des ASIC conçus par Google spécifiquement pour l'exécution de programmes de machine learning

III.5 Conclusion

Nous avons abordé dans ce chapitre les différentes technologies utilisées lors de la réalisation de notre projet. D'abord, nous avons détaillé la composition des deux architectures CNN que nous avons utilisées. Ensuite, nous avons présenté le simulateur Udacity qui nous permettra de cloner le comportement d'une voiture, puis, le langage utilisé pour toute la programmation, «Python», avec ses différentes bibliothèques. Enfin, nous avons décrit les environnements d'exécution Jupyter notebook et Google Colab.

Chapitre IV

Travail réalisé

IV.1 Introduction

Ayant introduit les concepts de la conduite autonome et de l'intelligence artificielle ainsi que les technologies nécessaires pour la réalisation de notre étude, nous pouvons maintenant aborder le cœur de notre travail : la simulation d'une voiture autonome utilisant l'apprentissage approfondi.

Nous détaillerons, d'abord, l'ensemble des étapes suivies dans la création d'un modèle basé sur l'architecture de réseau de neurones convolutifs NVIDIA. Ce modèle prédira l'angle de braquage de notre voiture autonome. Puis, nous analyserons les résultats obtenues lors des tests sur le simulateur.

Enfin, nous implémenterons deux codes additionnels liés à la voiture autonome : Nous utiliserons des méthodes de traitement d'image pour détecter les lignes sur une voie droite et nous nous baserons sur l'architecture CNN LeNet pour la classification des panneaux de signalisation. Ces deux codes pourrons être utiliser dans des travaux réalisation des voitures autonomes.

IV.2 Simulation d'une voiture autonome

La simulation d'une voiture autonome est obtenue en clonant la conduite d'une voiture faite par un humain. Ainsi, nous allons d'abord décrire le procès qui nous a fait aboutir à la simulation : préparation des données, création du modèle, connexion entre le modèle et le simulateur, test du modèle et enfin résultats et analyses. Comme indiqué figure IV.1.

IV.2.1 Préparation des données

IV.2.1.1 Collection de l'ensemble de données

Afin de collecter les données nécessaires pour entraîner notre modèle, nous utiliserons le simulateur Udacity. Ce simulateur nous permet d'enregistrer le comportement de conduite de voiture à cloner dans un premier temps, puis, dans un second temps, après avoir créé un modèle, de pouvoir le mettre à l'épreuve.

Pour enregistrer les données, nous utiliserons le premier circuit du simulateur. Ce circuit est constitué en majorité de virages à gauche, comme mentionné en III.3 ce qui biaisera

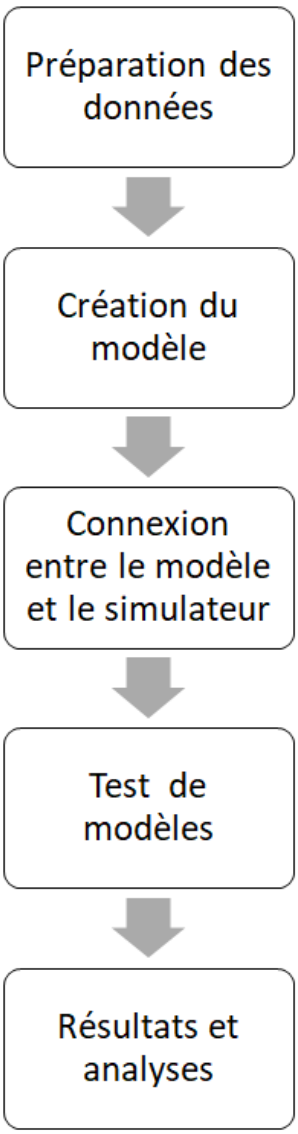
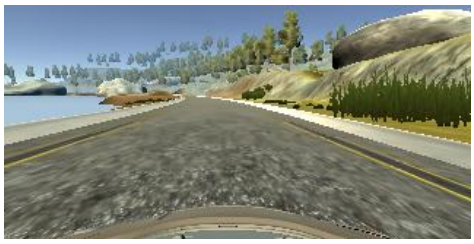


Fig. IV.1 – Approche utilisée pour la simulation

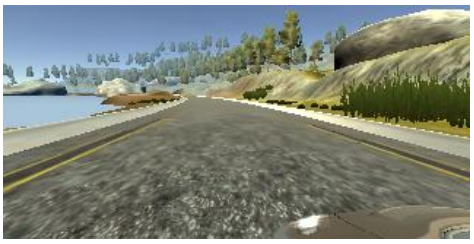
le modèle et fera qu'il tournera plus à gauche. Pour équilibrer nos données, nous avons enregistré trois tours du circuit dans un sens puis trois autres tours en sens inverse.

Rappelons que la qualité de l'apprentissage du modèle dépendra de la qualité de la conduite lors de l'enregistrement. C'est pour cette raison que nous avons essayé de garder la voiture au milieu de la route en évitant de zigzaguer durant la phase d'apprentissage.

L'ensemble final des données est constitué d'un dossier *IMG* contenant au total 12159 images enregistrées (Soit 4053 images par caméra. Un exemple de trois images du dossier prises au même moment est donné en figure IV.2) et d'un fichier *driving_log.csv*, dont un extrait est illustré en figure IV.3. Le taux d'échantillonnage de la simulation est de 10 FPS (soit 10 images par seconde) [68].



(a) Image centrale



(b) Image de gauche



(c) Image de droite

Fig. IV.2 – Illustration de 3 images prises, à un moment donné, par le simulateur

C:\Users\IMG\center_2022_05_09_13_20_19_460.jpg, C:\Users\IMG\left_2022_05_09_13_20_19_460.jpg, C:\Users\IMG\right_2022_05_09_13_20_19_460.jpg, 0, 1, 0, 30.19031
C:\Users\IMG\center_2022_05_09_13_20_19_565.jpg, C:\Users\IMG\left_2022_05_09_13_20_19_565.jpg, C:\Users\IMG\right_2022_05_09_13_20_19_565.jpg, 0, 1, 0, 30.19029
C:\Users\IMG\center_2022_05_09_13_20_19_669.jpg, C:\Users\IMG\left_2022_05_09_13_20_19_669.jpg, C:\Users\IMG\right_2022_05_09_13_20_19_669.jpg, 0, 1, 0, 30.1903
C:\Users\IMG\center_2022_05_09_13_20_19_773.jpg, C:\Users\IMG\left_2022_05_09_13_20_19_773.jpg, C:\Users\IMG\right_2022_05_09_13_20_19_773.jpg, 0, 1, 0, 30.19029
C:\Users\IMG\center_2022_05_09_13_20_19_879.jpg, C:\Users\IMG\left_2022_05_09_13_20_19_879.jpg, C:\Users\IMG\right_2022_05_09_13_20_19_879.jpg, 0, 1, 0, 30.19029
C:\Users\IMG\center_2022_05_09_13_20_19_987.jpg, C:\Users\IMG\left_2022_05_09_13_20_19_987.jpg, C:\Users\IMG\right_2022_05_09_13_20_19_987.jpg, 0, 1, 0, 30.19032
C:\Users\IMG\center_2022_05_09_13_20_20_091.jpg, C:\Users\IMG\left_2022_05_09_13_20_20_091.jpg, C:\Users\IMG\right_2022_05_09_13_20_20_091.jpg, 0, 1, 0, 30.19029
C:\Users\IMG\center_2022_05_09_13_20_20_192.jpg, C:\Users\IMG\left_2022_05_09_13_20_20_192.jpg, C:\Users\IMG\right_2022_05_09_13_20_20_192.jpg, 0, 1, 0, 30.1903
C:\Users\IMG\center_2022_05_09_13_20_20_292.jpg, C:\Users\IMG\left_2022_05_09_13_20_20_292.jpg, C:\Users\IMG\right_2022_05_09_13_20_20_292.jpg, 0, 1, 0, 30.19028
C:\Users\IMG\center_2022_05_09_13_20_20_394.jpg, C:\Users\IMG\left_2022_05_09_13_20_20_394.jpg, C:\Users\IMG\right_2022_05_09_13_20_20_394.jpg, 0, 1, 0, 30.19029

Fig. IV.3 – Extrait du fichier driving_log.csv

IV.2.1.2 Distribution des données

Le modèle que nous souhaitons construire prédira l'angle de braquage de la voiture, en se basant sur l'image en entrée.

Auparavant, il nous a fallu vérifier si toutes nos données étaient uniformément distribuées pour ne pas biaiser l'apprentissage dans un sens plus que l'autre.

Pour le vérifier, nous avons représenté nos données en histogramme selon leur angle en :

- Divisant l'ensemble des données en 25 groupes représentant les intervalles (Il est conseillé de choisir un nombre impair pour centrer l'ensemble des valeurs par rapport à 0).
- Calculant le centre de chaque intervalle.
- Définissant une épaisseur de chaque intervalle.

L'histogramme est créé par l'exécution du bout de code suivant :

```
num_bins = 25
hist, bins = np.histogram(data['steering'], num_bins)
center = (bins[:-1]+ bins[1:]) * 0.5
plt.bar(center, hist, width=0.05)
```

La distribution des angles de braquage obtenue est illustrée figure IV.4.

Nous remarquons que le nombre de données associées à l'angle 0 est très élevé, comme indiqué figure IV.4. Ceci biaisera l'apprentissage du modèle et fera que la voiture ne circulera

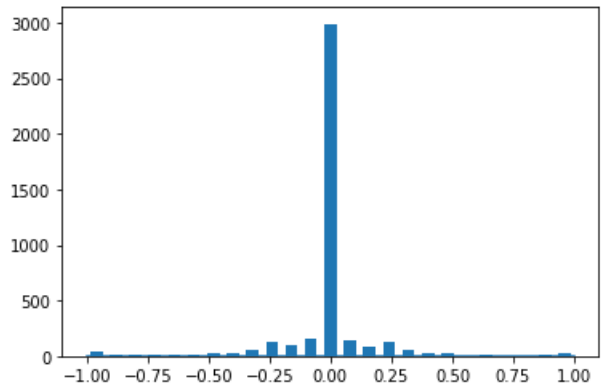


Fig. IV.4 – Répartition de notre ensemble de données selon l'angle de braquage

que tout droit. Donc, nous avons baissé le nombre de données associées à cet angle. Cette modification fait l'objet du point suivant.

IV.2.1.3 Filtrage des données associées à l'angle 0

Nous avons réduit, dans un premier temps, le nombre de données par intervalle (samples_per_bin) à 400 pour limiter le déséquilibre de l'ensemble de données.

Néanmoins, la solution évidente de ne garder que les 400 premières données de l'intervalle ferait que l'on introduirait un biais vers le début du circuit. Donc, nous avons d'abord effectuer un mélange aléatoire des données à l'aide de la fonction *shuffle* de la bibliothèque SciKit-Learn, puis garder les 400 premières valeurs.

le code ci-après détaille ce processus. La nouvelle répartition des angles est illustrée figure IV.5.

```
samples_per_bin = 400
remove_list = []
for j in range(num_bins):
  list_ = []
  for i in range(len(data['steering'])):
    if data['steering'][i] >= bins[j] and data['steering'][i] <= bins[j+1]:
      list_.append(i)
  list_ = shuffle(list_)
  list_ = list_[samples_per_bin:]
  remove_list.extend(list_)
data.drop(data.index[remove_list], inplace=True)
```

IV.2.1.4 Augmentation des données

Dans l'impossibilité d'enregistrer toutes les données provenant des différentes routes sous toutes les conditions météorologiques possibles, il nous est apparu nécessaire d'utiliser une méthode de généralisation du comportement de la voiture avec les données existantes.

L'augmentation des données aide le modèle à généraliser son apprentissage sur des routes qui lui sont inconnues. Ceci est effectué en appliquant une ou plusieurs transformations sur

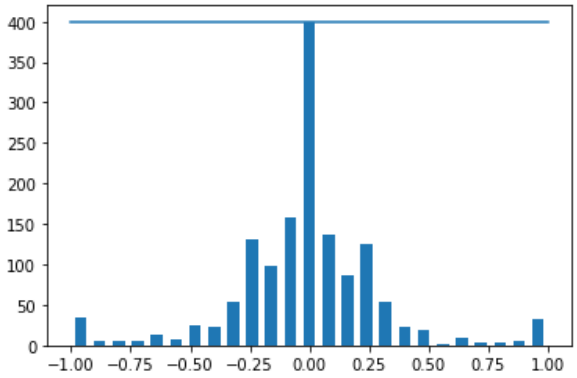


Fig. IV.5 – Nouvelle répartition des données de l'ensemble de données

les images récoltées. Les fonctions que nous avons utilisées sont les suivantes :

IV.2.1.4.a La fonction agrandissement

La fonction agrandissement, comme son nom l'indique, permet d'agrandir l'image. Pour l'implémenter nous avons créé une fonction appelée zoom décrite ci-après.

```
def zoom(image):
  zoom = iaa.Affine(scale=(1, 1.3))
  image = zoom.augment_image(image)
  return image
```

La figure IV.6 illustre l'image originale et le résultat après application du zoom.

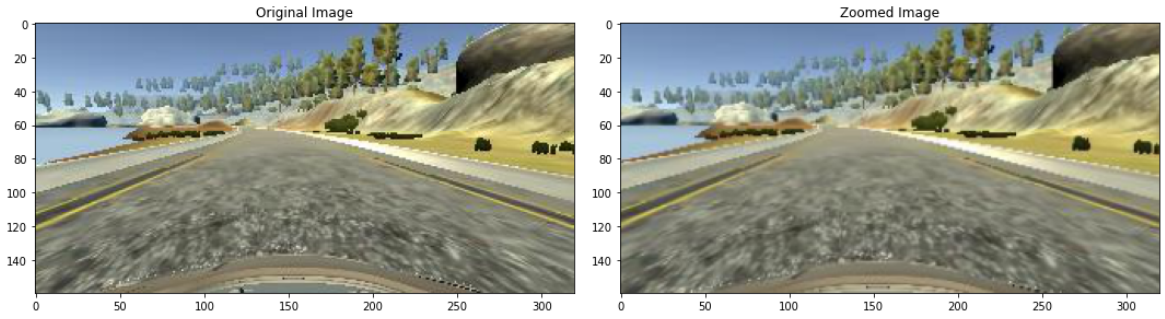


Fig. IV.6 – Application de fonction zoom

IV.2.1.4.b La fonction décalage

La fonction décalage permet de décaler l'image horizontalement et/ou verticalement. Cette fonction est décrite ci-après et est appelée pan. La figure IV.7 illustre l'image originale et le résultat après application de la fonction décalage.

```
def pan(image):
  pan = iaa.Affine(translate_percent={"x":(-0.1,0.1), "y":(-0.1,0.1)})
  image = pan.augment_image(image)
  return image
```

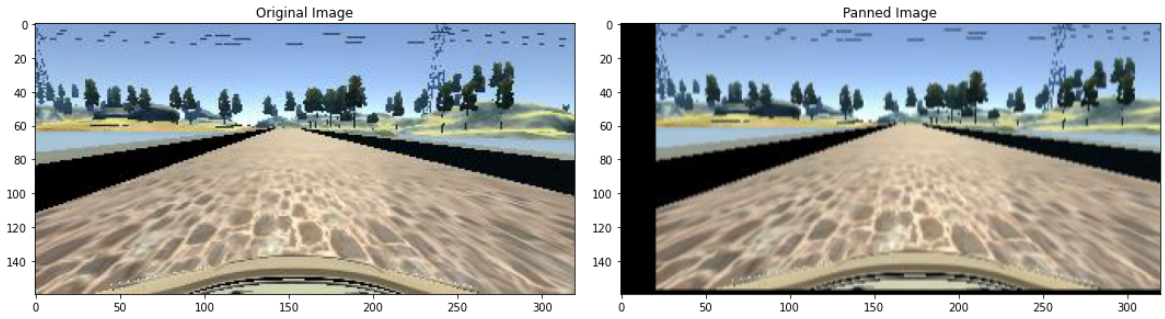


Fig. IV.7 – Application de fonction pan

IV.2.1.4.c La fonction luminosité

La fonction luminosité permet de généraliser l'image prise pour des conditions de luminosité différentes dues, par exemple, aux changements météorologiques.

La fonction qui permet d'effectuer cette généralisation est décrite ci-après par la fonction img_random_brightness. La figure IV.8 illustre l'image originale et le résultat après application du changement de luminosité.

```
def img_random_brightness(image):
    brightness = iaa.Multiply((0.2,1.2))
    image = brightness.augment_image(image)
    return image
```

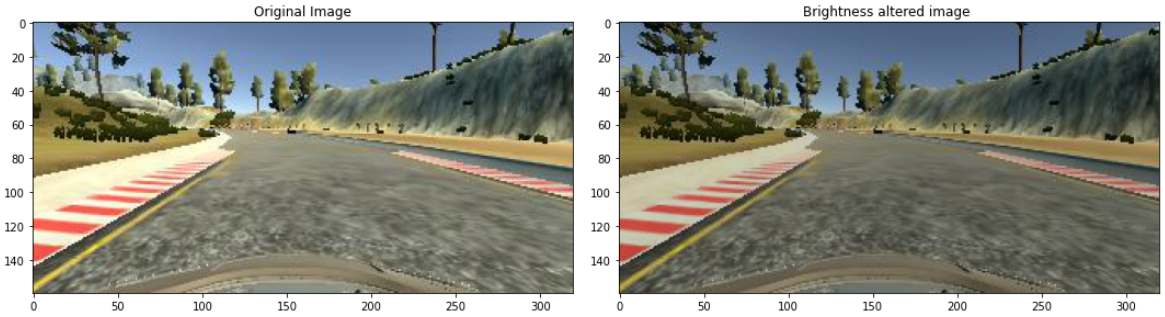


Fig. IV.8 – Application de la fonction de luminosité

IV.2.1.4.d La fonction de rotation horizontale

La rotation horizontale permet d'avoir une image miroir de l'originale. Cela permet de balancer l'ensemble de données. La fonction permettant d'effectuer cette rotation est décrite ci-après :

```
def img_random_flip(image,steering_angle):
    image = cv2.flip(image,1)
    steering_angle = -steering_angle
    return image,steering_angle
```

La figure IV.9 illustre l'image originale et le résultat après application de la rotation horizontale appelée img_random_flip.

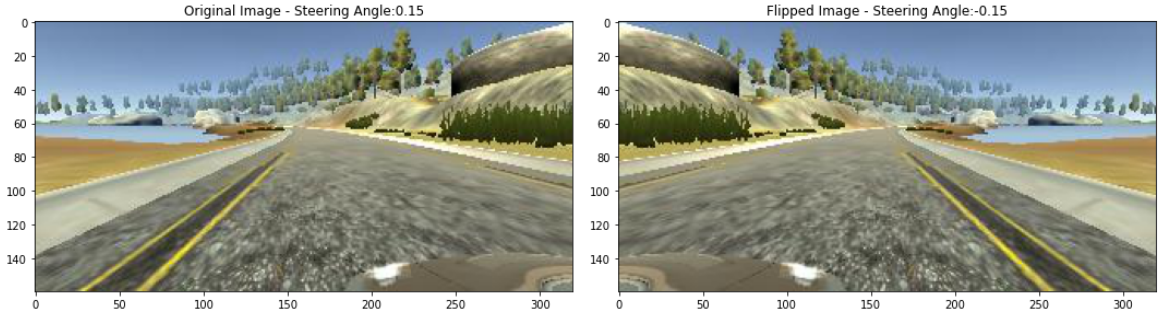


Fig. IV.9 – Application de la fonction rotation horizontale

IV.2.1.4.e La fonction d'augmentation aléatoire

La fonction d'augmentation aléatoire permet de combiner une ou plusieurs des augmentations citées précédemment à une seule et même image.

La fonction random_augment permettant d'appliquer ce traitement est décrite ci-après. La figure IV.10 illustre l'image originale et le résultat obtenu.

```
def random_augment(image,steering_angle):
    image = mpimg.imread(image)
    if np.random.rand() < 0.5:
      image = pan(image)
    if np.random.rand() < 0.5:
      image = zoom(image)
    if np.random.rand() < 0.5:
      image = img_random_brightness(image)
    if np.random.rand() < 0.5:
      image,steering_angle = img_random_flip(image, steering_angle)
    return image,steering_angle
```

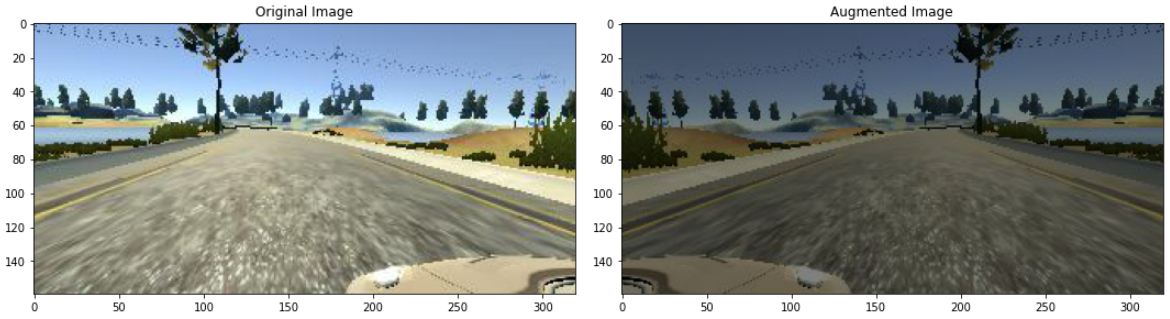


Fig. IV.10 – Augmentation aléatoire d'image

IV.2.1.5 Préparation des images

Avant de présenter les images au modèle, celles-ci sont prétraitées, à l'aide de la fonction suivante :

```
def img_preprocess(img):
    img = img[60:135,:,:]
    img = cv2.cvtColor(img,cv2.COLOR_RGB2YUV)
    img = cv2.GaussianBlur(img,(3,3),0)
    img = cv2.resize(img,(200,66))
    img = img/255
    return img
```

La fonction *img_preprocess* :

- Recadre l'image , ne laissant apparaître que la route. Ce qui permet la focalisation sur les éléments essentiels lors de l'apprentissage.
- Change le codage de couleur du RGB [1] au YUV [2] comme recommandé lors de l'utilisation de l'architecture *NVIDIA*.
- Applique un filtre gaussien de noyau 3X3 qui va lisser l'image et donc réduire le bruit.
- Rétrécit l'image à 200X66, taille indiquée pour le modèle NVIDIA, rendant par la même occasion le calcul plus rapide.
- Normalise l'image pour étaler de manière uniforme les valeurs de chaque pixel (autre prérequis de l'architecture NVIDIA).

Le résultat de la préparation de l'image est montré figure IV.11.

IV.2.1.6 Division des données en sous-ensembles d'entraînement et de validation

Après avoir augmenté les données et les avoir préparées, la prochaine étape consiste à les diviser en deux groupes : un pour l'entraînement et l'autre pour la validation.

Les données d'entraînement représentent 80 % des images. Les données de validation représentent les 20 % restantes. Le code suivant permet d'effectuer cette division :

[1] *Red Green Blue* pour Rouge Vert Bleu

[2] Format de couleur où l'Y représente la luminosité de l'image et UV correspondent à la luminance rouge et bleu respectivement tel que U = B-Y et V = R-Y. Ce format est souvent préféré au RGB pour ses capacités de compression et de réduction d'erreurs de transmission

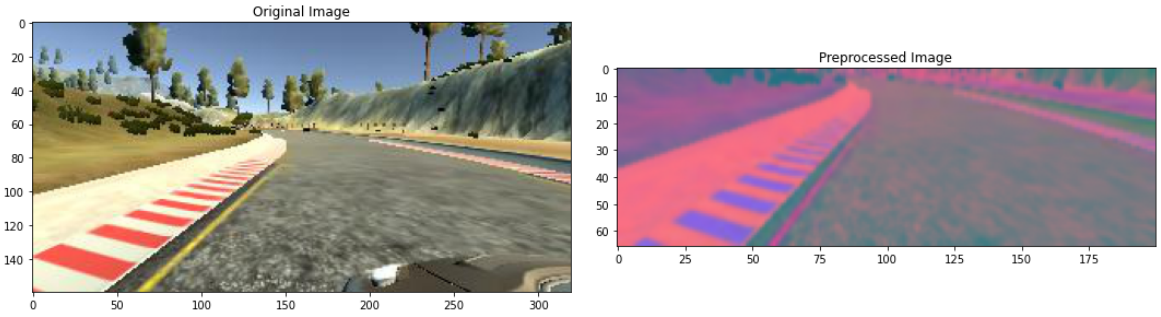


Fig. IV.11 – Prétraitement de l'image

```
X_train, X_valid, y_train, y_valid = train_test_split(image_paths,
                                                      steerings,
                                                      test_size=0.2,
                                                      random_state=6)
```

IV.2.2 Création du modèle

Après la constitution de l'ensemble de données, l'étape suivante est la construction d'un modèle NVIDIA qui permettra au véhicule de parcourir le circuit de manière autonome. Pour cela, nous pouvons soit : écrire notre code sur un Jupyter Notebook local ou bien en ligne sur la plateforme Google Colab.

En raison de la puissance de calcul limitée des machines en notre possession et le fait que Google Colab offre un accès gratuit à des GPU, nous avons opté pour cette seconde option.

IV.2.2.1 Architecture NVIDIA

L'architecture NVIDIA, précédemment décrite en III.2.2, est codée de la manière suivante :

```
def nvidia_model():
  model = Sequential()
  model.add(Convolution2D(24,5,5,subsample=(2,2),
            input_shape=(66,200,3),
            activation='elu'))
  model.add(Convolution2D(36,5,5,subsample=(2,2),activation='elu'))
  model.add(Convolution2D(48,5,5,subsample=(2,2),activation='elu'))
  model.add(Convolution2D(64,3,3,activation='elu'))
  model.add(Convolution2D(64,3,3,activation='elu'))
  model.add(Flatten())
  model.add(Dense(100,activation ='elu'))
  model.add(Dense(50,activation ='elu'))
  model.add(Dense(10,activation ='elu'))
  model.add(Dense(1))
  optimizer = Adam(lr=1e-3)
  model.compile(loss='mse',optimizer=optimizer,metrics=['accuracy'])
  return model
```

Cette architecture peut être résumée dans le tableau figure IV.12.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 31, 98, 24)        1824
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 47, 36)        21636
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 5, 22, 48)         43248
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 3, 20, 64)         27712
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 1, 18, 64)         36928
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               115300
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_3 (Dense)              (None, 10)                510
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
```

Fig. IV.12 – Résumé de l'architecture NVIDIA

Notons que : la normalisation décrite en III.2.2 est effectuée dans la fonction img_preprocess

en IV.2.1.5. Nous avons rajouté ici les lignes de code relatives à la compilation du modèle. Cette compilation se fera en utilisant les paramètres suivants :

- L'erreur quadratique moyenne comme fonction de perte (loss).
- L'optimiseur Adam avec un taux d'apprentissage de 0,001,
- Le nombre d'erreurs comme métrique d'optimisation.

IV.2.3 Connexion entre le modèle et le simulateur

Nous utilisons un serveur web pour relier le simulateur au modèle. Ce dernier est chargé par le serveur lors de son démarrage. La communication entre le simulateur et ce serveur web se fait sur le port «4567».

IV.2.3.1 Création de l'environnement de travail

Afin de pouvoir tester notre modèle, nous avons créé un environnement Python pour y installer les dépendances nécessaires, telles que : Keras, python-socketio, ...

IV.2.3.2 Approche utilisée pour la connexion

En mode autonome, la simulation se déroule de la manière suivante, figure IV.13 :

- Le simulateur envoie périodiquement, au serveur web, l'image, la vitesse, l'angle de braquage et l'accélération courants de la voiture.
- Le serveur web lance le prétraitement de l'image avant de la transmettre au modèle. Le serveur web :
  - Recadre l'image en ne laissant que la partie inférieure (du pixel 60 à 135).
  - Convertit l'image du format RGB au YUV.
  - Applique à l'image un filtre Gaussien de noyau 3X3.
  - Réduit la taille de l'image à 200X66.
  - Normalise l'image.
- Le modèle prédit le futur angle de braquage, figure IV.14.
- Le serveur web calcule la future accélération du véhicule en tenant compte de la vitesse reçue du simulateur, tel que :
  - L'accélération sera maximale si la vitesse est nulle (au début de la simulation).
  - L'accélération diminuera jusqu'à s'annuler lorsque la vitesse maximale prédéfinie est atteinte.
  - La vitesse du véhicule restera constante pour le reste de la simulation, comme indiqué par le code suivant :

```
speed_limit = 10
throttle = 1.0 - speed/speed_limit
```

- Le serveur web retourne la valeur calculée de l'accélération au simulateur, accompagnée de l'angle prédit par le modèle.
- Le simulateur applique ces commandes au véhicule.
- Ce processus se répétera durant toute la durée de la simulation.

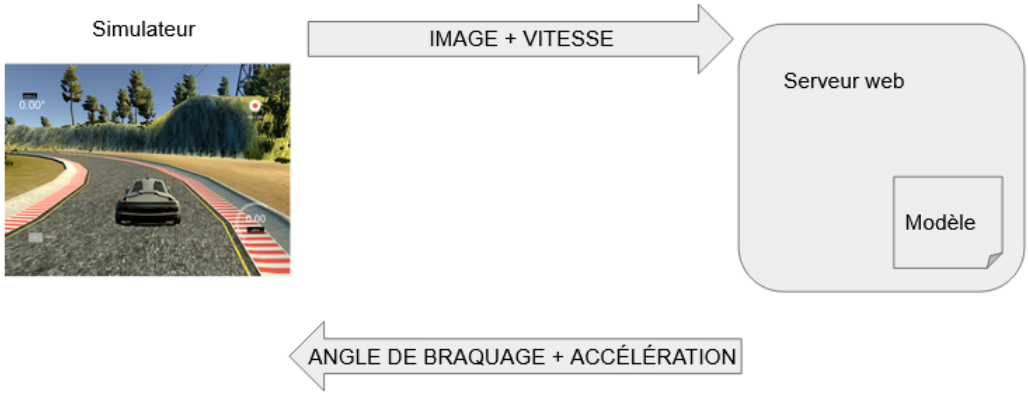


Fig. IV.13 – Approche utiliséé pour la connexion du modèle au simulateur

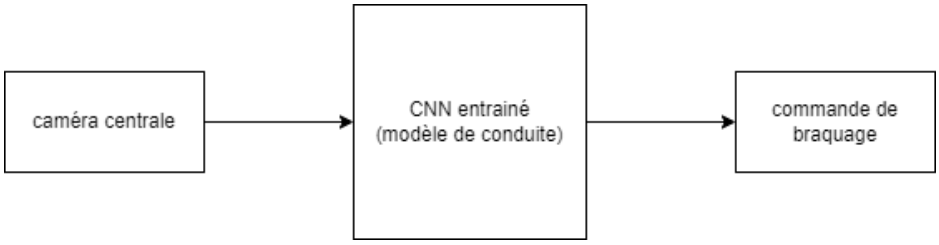


Fig. IV.14 – Processus de prédiction de l'angle de braquage [68]

IV.2.4 Test de modèles

Afin de trouver le modèle optimal pour la simulation, nous avons commencé par créer un modèle avec des paramètres que nous avons estimés comme potentiellement bons. Puis, au fur et à mesure, en analysant les résultats obtenus sur les circuits de la simulation, nous avons opéré les changements que nous jugions adéquats.

Au final, nous avons trouvé les deux modèles suivants comme les plus optimaux :

IV.2.4.1 Modèle 1

Pour le premier modèle, nous avons utilisé :

- samples_per_bin = 200
- Optimizeur Adam avec un learning rate = 0,0001
- Batch pour l'apprentissage = 200
- Le nombre d'epochs = 20
- Batch pour la validation = 500

IV.2.4.2 Modèle 2

Pour le deuxième modèle, nous avons utilisé :

- samples_per_bin = 400
- Optimizeur Adam avec un learning rate = 0.001
- Batch pour l'apprentissage = 100
- Le nombre d'epochs = 20
- Batch pour la validation = 100

IV.2.5 Résultats et analyses

IV.2.5.1 Modèle 1

À la fin du training, le premier modèle donne une valeur de loss de 0,0272 et une valeur d'accuracy de 0,0876. L'évolution de la valeur du loss et de l'accuracy durant les epochs de l'apprentissage est représentée figure IV.15.

```
500/500 [==============================] - 674s 1s/step - loss: 0.0892 - acc: 0.0749 - val_loss: 0.0651 - val_acc: 0.0724
Epoch 2/20
500/500 [==============================] - 661s 1s/step - loss: 0.0649 - acc: 0.0797 - val_loss: 0.0438 - val_acc: 0.0822
Epoch 3/20
500/500 [==============================] - 663s 1s/step - loss: 0.0543 - acc: 0.0829 - val_loss: 0.0468 - val_acc: 0.0847
Epoch 4/20
500/500 [==============================] - 665s 1s/step - loss: 0.0501 - acc: 0.0836 - val_loss: 0.0405 - val_acc: 0.0823
Epoch 5/20
500/500 [==============================] - 668s 1s/step - loss: 0.0454 - acc: 0.0856 - val_loss: 0.0323 - val_acc: 0.0828
Epoch 6/20
500/500 [==============================] - 672s 1s/step - loss: 0.0429 - acc: 0.0856 - val_loss: 0.0331 - val_acc: 0.0840
Epoch 7/20
500/500 [==============================] - 675s 1s/step - loss: 0.0403 - acc: 0.0872 - val_loss: 0.0301 - val_acc: 0.0849
Epoch 8/20
500/500 [==============================] - 671s 1s/step - loss: 0.0393 - acc: 0.0865 - val_loss: 0.0281 - val_acc: 0.0827
Epoch 9/20
500/500 [==============================] - 671s 1s/step - loss: 0.0369 - acc: 0.0879 - val_loss: 0.0294 - val_acc: 0.0813
Epoch 10/20
500/500 [==============================] - 679s 1s/step - loss: 0.0357 - acc: 0.0861 - val_loss: 0.0254 - val_acc: 0.0831
Epoch 11/20
500/500 [==============================] - 678s 1s/step - loss: 0.0347 - acc: 0.0856 - val_loss: 0.0247 - val_acc: 0.0831
Epoch 12/20
500/500 [==============================] - 680s 1s/step - loss: 0.0335 - acc: 0.0857 - val_loss: 0.0252 - val_acc: 0.0837
Epoch 13/20
500/500 [==============================] - 686s 1s/step - loss: 0.0326 - acc: 0.0856 - val_loss: 0.0249 - val_acc: 0.0836
Epoch 14/20
500/500 [==============================] - 690s 1s/step - loss: 0.0311 - acc: 0.0857 - val_loss: 0.0246 - val_acc: 0.0826
Epoch 15/20
500/500 [==============================] - 687s 1s/step - loss: 0.0303 - acc: 0.0872 - val_loss: 0.0222 - val_acc: 0.0828
Epoch 16/20
500/500 [==============================] - 690s 1s/step - loss: 0.0295 - acc: 0.0859 - val_loss: 0.0226 - val_acc: 0.0829
Epoch 17/20
500/500 [==============================] - 688s 1s/step - loss: 0.0288 - acc: 0.0864 - val_loss: 0.0227 - val_acc: 0.0828
Epoch 18/20
500/500 [==============================] - 687s 1s/step - loss: 0.0284 - acc: 0.0875 - val_loss: 0.0216 - val_acc: 0.0837
Epoch 19/20
500/500 [==============================] - 684s 1s/step - loss: 0.0275 - acc: 0.0872 - val_loss: 0.0206 - val_acc: 0.0824
Epoch 20/20
500/500 [==============================] - 688s 1s/step - loss: 0.0272 - acc: 0.0876 - val_loss: 0.0202 - val_acc: 0.0830
```

Fig. IV.15 – Valeurs du loss et de l'accuracy à chaque époque du modèle 1

Ces valeurs sont représentées en figure IV.17 et figure IV.16, pour une meilleur analyse.

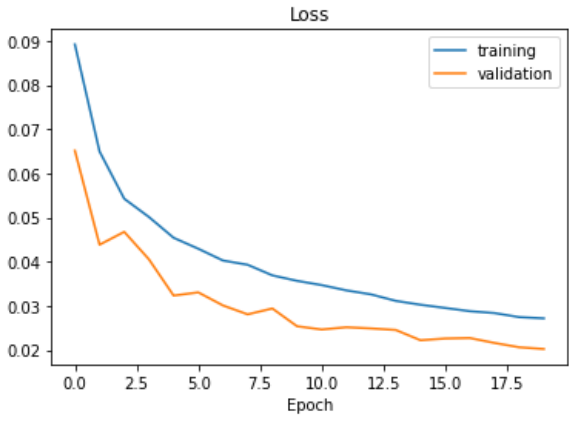


Fig. IV.16 – Représentation du loss à chaque époque du training du modèle 1

Après simulation du modèle, nous constatons que :

- Dans le circuit de training (circuit déjà connu par le modèle), la voiture arrive presque à rouler correctement sur la majorité de circuit. Elle sort du circuit comme le montre la figure IV.18.

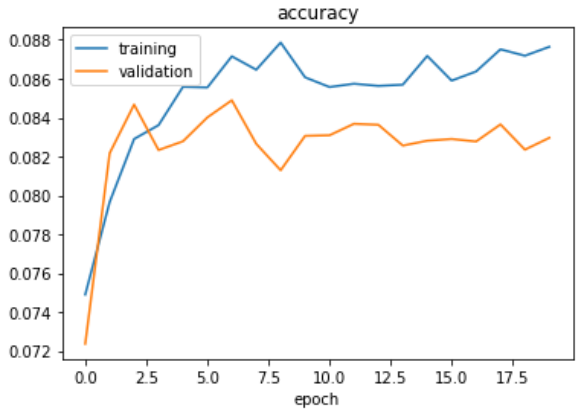


Fig. IV.17 – Représentation de l'accuracy à chaque époque du training du modèle 1



Fig. IV.18 – Simulation du modèle 1 sur le circuit de training

- Dans le circuit de test (circuit inconnu), la voiture a roulé jusqu'au moment où elle a heurté une barrière après un quart du circuit, comme indiqué figure IV.19.



Fig. IV.19 – Simulation du modèle 1 sur le circuit de validation

D'après les résultats obtenus, nous concluons que le modèle 1 n'atteint pas les performances souhaitées.

IV.2.5.2 Modèle 2

À la fin du training, le deuxième modèle donne une valeur de *loss* de 0,0420 et une valeur de l'accuracy est de 0,1168. L'évolution de la valeur du *loss* à chaque époque est indiqué figure IV.20.

```
Epoch 1/20
300/300 [==============================] - 197s 658ms/step - loss: 0.1360 - acc: 0.1028 - val_loss: 0.0883 - val_acc: 0.1085
Epoch 2/20
300/300 [==============================] - 186s 620ms/step - loss: 0.0836 - acc: 0.1053 - val_loss: 0.0718 - val_acc: 0.1135
Epoch 3/20
300/300 [==============================] - 184s 614ms/step - loss: 0.0658 - acc: 0.1089 - val_loss: 0.0480 - val_acc: 0.1171
Epoch 4/20
300/300 [==============================] - 185s 617ms/step - loss: 0.0579 - acc: 0.1135 - val_loss: 0.0455 - val_acc: 0.1192
Epoch 5/20
300/300 [==============================] - 185s 618ms/step - loss: 0.0501 - acc: 0.1132 - val_loss: 0.0421 - val_acc: 0.1208
Epoch 6/20
300/300 [==============================] - 185s 617ms/step - loss: 0.0466 - acc: 0.1175 - val_loss: 0.0371 - val_acc: 0.1196
Epoch 7/20
300/300 [==============================] - 185s 616ms/step - loss: 0.0431 - acc: 0.1168 - val_loss: 0.0334 - val_acc: 0.1196
Epoch 8/20
300/300 [==============================] - 186s 619ms/step - loss: 0.0422 - acc: 0.1185 - val_loss: 0.0312 - val_acc: 0.1229
Epoch 9/20
300/300 [==============================] - 185s 616ms/step - loss: 0.0420 - acc: 0.1168 - val_loss: 0.0328 - val_acc: 0.1214
Epoch 10/20
300/300 [==============================] - 184s 615ms/step - loss: 0.0396 - acc: 0.1179 - val_loss: 0.0303 - val_acc: 0.1248
Epoch 11/20
300/300 [==============================] - 183s 611ms/step - loss: 0.0394 - acc: 0.1177 - val_loss: 0.0351 - val_acc: 0.1210
Epoch 12/20
300/300 [==============================] - 182s 608ms/step - loss: 0.0376 - acc: 0.1151 - val_loss: 0.0278 - val_acc: 0.1219
Epoch 13/20
300/300 [==============================] - 184s 613ms/step - loss: 0.0402 - acc: 0.1160 - val_loss: 0.0284 - val_acc: 0.1191
Epoch 14/20
300/300 [==============================] - 183s 609ms/step - loss: 0.0368 - acc: 0.1114 - val_loss: 0.0262 - val_acc: 0.1190
Epoch 15/20
300/300 [==============================] - 183s 609ms/step - loss: 0.0350 - acc: 0.1177 - val_loss: 0.0365 - val_acc: 0.1192
Epoch 16/20
300/300 [==============================] - 181s 604ms/step - loss: 0.0381 - acc: 0.1132 - val_loss: 0.0286 - val_acc: 0.1241
Epoch 17/20
300/300 [==============================] - 181s 604ms/step - loss: 0.0340 - acc: 0.1187 - val_loss: 0.0245 - val_acc: 0.1222
Epoch 18/20
300/300 [==============================] - 182s 607ms/step - loss: 0.0328 - acc: 0.1171 - val_loss: 0.0274 - val_acc: 0.1237
Epoch 19/20
300/300 [==============================] - 182s 608ms/step - loss: 0.0323 - acc: 0.1178 - val_loss: 0.0260 - val_acc: 0.1218
Epoch 20/20
300/300 [==============================] - 187s 623ms/step - loss: 0.0323 - acc: 0.1152 - val_loss: 0.0276 - val_acc: 0.1242
```

Fig. IV.20 – Valeurs du loss et accuracy à chaque epochs du modèle 2

Ces valeurs sont représentées en figure IV.21 et figure IV.22, pour une meilleur analyse.

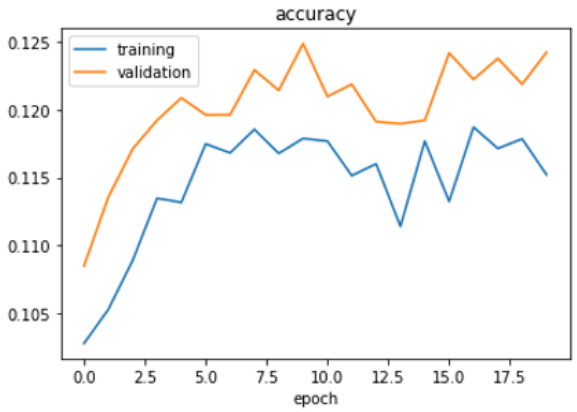


Fig. IV.21 – Représentation de l'accuracy à chaque époque du training du modèle 2

Après simulation du modèle, nous constatons que :

- Dans le circuit de training (circuit connu par le modèle), la voiture fait le tour du circuit en gardant comme position le milieu de la voie sauf à certains endroits. Figure IV.23.
- Dans le circuit de test (circuit non connu), la voiture termine parfaitement sa course tout en restant au milieu de la voie. Figure IV.24.

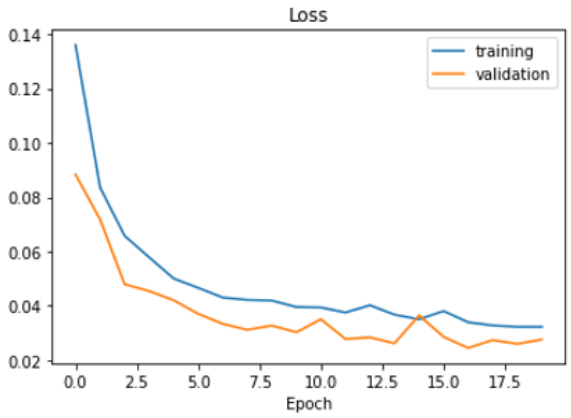


Fig. IV.22 – Représentation du loss à chaque époque du training du modèle 2



Fig. IV.23 – Simulation du modèle 2 sur le circuit de training



Fig. IV.24 – Simulation du modèle 2 sur le circuit de test

Au vu de ces résultats, nous concluons que le modèle 2 est le meilleur, compte tenu des performances réalisées sur les deux circuits du simulateur.

IV.3 Détection de lignes sur une voie droite

La détection des lignes de voie est une étape très importante lors de la conduite. Le maintien du véhicule dans sa voie garantissant une certaine sécurité. Nous nous sommes intéressés dans ce qui suit à la détection de lignes sur une voie droite en utilisant une image puis une vidéo.

Pour effectuer cette détection, nous avons choisi la bibliothèque graphique open-source OpenCV pour le traitement de l'image et l'algorithme Canny de détection de contours.

IV.3.1 Détection sur une image

Pour détecter les lignes sur une voie droite, nous avons suivi les étapes décrites ci-dessous :

IV.3.1.1 Importation de l'image

Nous avons téléchargé l'image d'une route contenant une voie dans un dossier nommé «finding-lanes» [83], figure IV.25.



Fig. IV.25 – Image de route avec une voie droite [83]

IV.3.1.2 Traitement de l'image

Dans un fichier python enregistré dans le dossier cité précédemment, nous avons défini une fonction nommée *canny* qui :

- Convertit l'image en niveau de gris pour faciliter le calcul (une image en niveau de gris possède un seul niveau alors qu'une image RGB est représentée sur trois niveaux).
- Ajoute un filtre Gaussien, pour réduire le bruit de l'image en niveau de gris.
- Applique l'algorithme Canny qui utilise la méthode de la descente du gradient pour détecter les régions d'intérêt de l'image.

```
def canny(image):
 gray = cv2.cvtColor(image,cv2.COLOR_RGB2GRAY)
 blur = cv2.GaussianBlur(gray,(5,5),0)
 canny = cv2.Canny(blur, 50 , 150)
```

```
    return canny
```

Le résultat est montré par la figure IV.26.

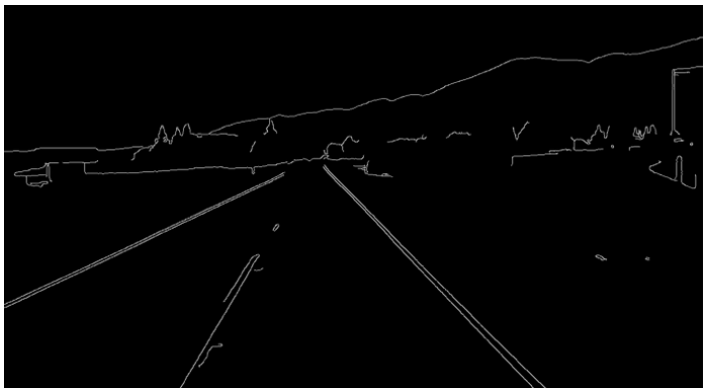


Fig. IV.26 – Résultat de l'exécution de la fonction canny

IV.3.1.3 Délimitation de la région d'intérêt

Pour délimiter la région d'intérêt, nous avons appliqué le masque figure IV.27 au résultat précédant figure IV.26 en utilisant le code suivant :

```
def region_of_interest(image):
    height = image.shape[0]
    polygons = np.array([
        [(200, height),(1100,height),(550,250)]
                        ])
    mask = np.zeros_like(image)
    cv2.fillPoly(mask, polygons,255)
    masked_image = cv2.bitwise_and(image,mask)
    return masked_image
```

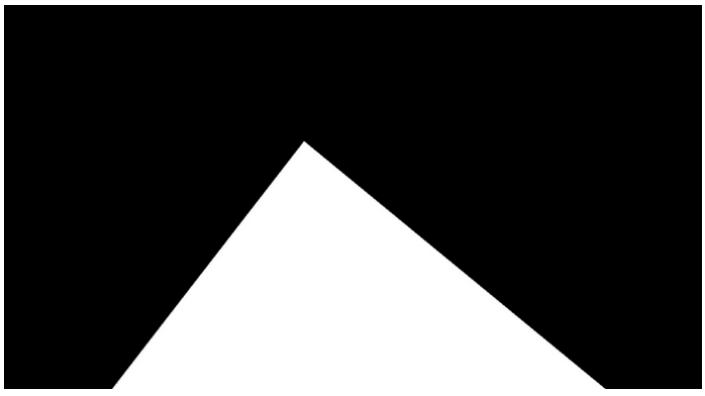


Fig. IV.27 – Masque utilisé

Le résultat que nous avons obtenu est donné sur la figure IV.28.



Fig. IV.28 – Région d'intérêt

IV.3.1.4 Identification des lignes les plus adéquates

Une fois la spécification de la région délimitant la voie effectuée, il nous faut représenter les deux droites qui délimitent les côtés de notre voie. Pour cela, nous allons calculer la moyenne des lignes ayant une pente positive (qui représentera la ligne de voie de droite). Puis, la moyenne des lignes ayant une pente négative (qui représentera la ligne de voie de gauche).

Ceci est décrit dans le code suivant :

```
def make_coordinates(image, line_parameters):
    slope,intercept = line_parameters
    y1 = image.shape[0]
    y2 = int(y1*(3/5))
    x1 = int((y1-intercept)/slope)
    x2 = int((y2-intercept)/slope)
    return np.array([x1,y1,x2,y2])

def average_slope_intercept(image,lines):
    left_fit = []
    right_fit = []
    for line in lines:
        x1,y1,x2,y2=line.reshape(4)
        parameters = np.polyfit((x1,x2),(y1,y2), 1)
        print(parameters)
        slope = parameters[0]
        intercept = parameters[1]
        if slope < 0:
            left_fit.append((slope,intercept))
        else:
            right_fit.append((slope,intercept))
    left_fit_average = np.average(left_fit, axis=0)
    right_fit_average = np.average(right_fit, axis=0)
    left_line = make_coordinates(image,left_fit_average)
    right_line = make_coordinates(image,right_fit_average)
    return np.array([left_line,right_line])
```

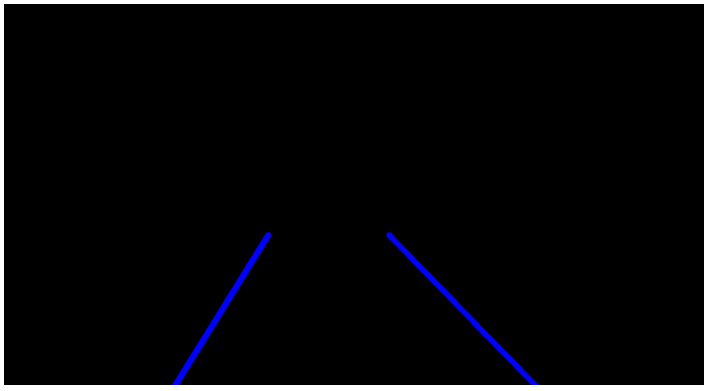


Fig. IV.29 – Lignes de voie optimisées

Notons qu'il y a une inversion des pentes, entre ce que nous avons décrit et ce que l'on voit sur la figure IV.28 car l'axe vertical de l'image est orienté vers le bas.

IV.3.1.5 Superposition du résultat obtenu avec l'image initiale

La dernière étape pour la détection de lignes sur une voie droite est la superposition de l'image du résultat précédent figure IV.29 et notre image initiale figure IV.25.

Pour ce faire, nous attribuerons à la figure IV.25 un poids de 0,8 ce qui la rendra un peu plus sombre et un poids de 1 à la figure IV.29 ce qui lui donnera 20% de poids en plus que l'image précédente, d'où une apparence plus claire lors de la superposition. Nous ajouterons par la suite une valeur scalaire de 1 (C'est un paramètre obligatoire de la fonction de superposition *addWeighted* permettant de modifier encore l'image résultante mais dont nous n'avons pas besoin ici). Ceci se traduit par le code suivant :

```
combo_image = cv2.addWeighted(lane_image, 0.8, line_image, 1, 1)
```

Le résultat d'exécution de la dernière étape est illustré figure IV.30.

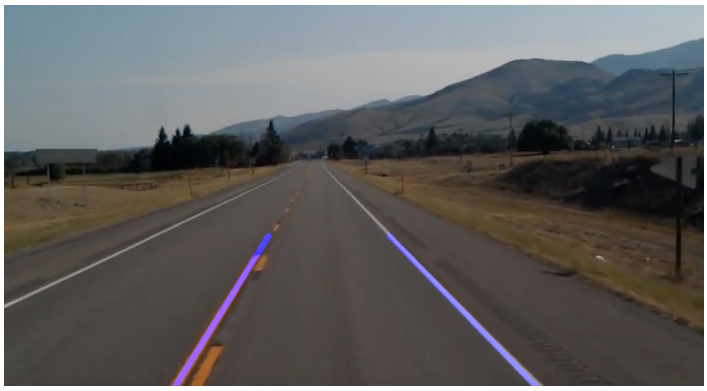


Fig. IV.30 – Détection des lignes de voie sur image

IV.3.2 Détection sur une vidéo

Après avoir généré le code qui détecte les lignes sur une voie droite, nous allons maintenant généraliser nos résultats pour une vidéo.

Après avoir téléchargé une vidéo contenant des lignes sur une voie droite [84], nous avons implémenté l'algorithme suivant :

- Lecture du fichier vidéo.
- Application de la détection de lignes sur l'image courante tant que la vidéo est ouverte avec les mêmes fonctions précédemment décrites en modifiant "line_image" par *Frame*.
- Arrêt de la boucle de lecture de la vidéo si l'utilisateur appuie sur la touche "q".
- Libération de l'espace mémoire contenant la vidéo.
- Fermeture de la fenêtre d'affichage.

Les lignes de code suivantes traduisent ses instructions :

```
cap = cv2.VideoCapture('test2.mp4')
while(cap.isOpened()):
   _, frame = cap.read()
   canny_image= canny(frame)
   cropped_image = region_of_interest(canny_image)
   lines=cv2.HoughLinesP(cropped_image, 2, np.pi/180, 100,
                         np.array([]), minLineLength=40,maxLineGap=5)
   averaged_lines = average_slope_intercept(frame,lines)
   line_image = display_lines(frame,averaged_lines)
   combo_image = cv2.addWeighted(frame,0.8, line_image,1,1)
   cv2.imshow('result',combo_image)
   if cv2.waitKey(1) == ord('q'):
      break
cap.release()
cv2.destroyAllWindows()
```

IV.4 Classification des panneaux de signalisation

La classification des panneaux de signalisation permet d'identifier les images issues du ou des capteurs dédiés à cette tâche au niveau du véhicule.

Pour ce faire, l'approche suivante a été entreprise, et sera détaillée ci-après :

1. Importation des données.
2. Construction du modèle.
3. Test du modèle.

IV.4.1 Importation des données

Le dossier de données, contenant des images (de panneaux de signalisation allemands) obtenues depuis [85], est constitué de trois fichiers de données et d'un fichier *signnames.csv*. Les fichiers de données sont :

1. Fichier *train* : contenant 34799 images de dimension 32x32x3 utilisées pour l'apprentissage du modèle.
2. Fichier *valid* : contenant 12630 images de dimension 32x32x3 utilisées pour la validation du modèle.
3. Fichier *test* : contenant 4410 images de dimension 32x32x3 utilisées pour tester le modèle.

Le fichier *signnames.csv* contient la liste des 43 classes de panneaux à reconnaître. Une classe étant un ensemble d'images représentant un seul et même panneau de signalisation. Une partie de ce fichier est présentée en figure IV.31.

	ClassId	SignName
0	0	Speed limit (20km/h)
1	1	Speed limit (30km/h)
2	2	Speed limit (50km/h)
3	3	Speed limit (60km/h)
4	4	Speed limit (70km/h)
5	5	Speed limit (80km/h)
6	6	End of speed limit (80km/h)
7	7	Speed limit (100km/h)
8	8	Speed limit (120km/h)
9	9	No passing
10	10	No passing for vechiles over 3.5 metric tons
11	11	Right-of-way at the next intersection
12	12	Priority road
13	13	Yield
14	14	Stop
15	15	No vechiles
16	16	Vechiles over 3.5 metric tons prohibited
17	17	No entry
18	18	General caution
19	19	Dangerous curve to the left
20	20	Dangerous curve to the right
21	21	Double curve
22	22	Bumpy road
23	23	Slippery road

Fig. IV.31 – Partie du fichier signnames.csv

IV.4.2 Construction du modèle

Le modèle que nous voulons construire permettra d'identifier l'image reçue en entrée en prédisant sa classe. Pour ce faire, nous avons modifié le réseau de neurones convolutifs *LeNet* de la manière suivante pour obtenir de meilleurs résultats :

1. Utilisation de 60 filtres avec un noyau de 5X5 à la première couche de convolution et de 30 filtres avec un noyau de 3X3 pour la deuxième couche de convolution.
2. Attribution d'un noyau de 2X2 à chacune des couches de MaxPooling.
3. Ajout d'une couche de convolution après chacune des deux couches précédentes avec leurs mêmes paramètres.
4. Emplois de 500 neurones dans le réseau entièrement connectés.
5. Ajout d'un dropout de coefficient 0,5 après le réseau de neurones entièrement connectés.

```
def leNet_modifmodel():
  model = Sequential()
  model.add(Conv2D(60,(5,5), input_shape=(32,32,1),
            activation='relu'))
  model.add(Conv2D(60,(5,5),activation='relu'))
  model.add(MaxPooling2D(pool_size=(2,2))
  model.add(Conv2D(30,(3,3),activation ='relu'))
  model.add(Conv2D(30,(3,3),activation ='relu'))
  model.add(MaxPooling2D(pool_size=(2,2))
  model.add(Flatten())
  model.add(Dense(500,activation='relu'))
  model.add(Dropout(0.5))
  model.add(Dense(num_classes,activation='softmax'))
  model.compile(adam_v2.Adam(learning_rate = 0.001),
                loss='categorical_crossentropy',
                metrics=['accuracy'])
  return model
```

Les paramètres de chaque couche du réseau de neurones, dans l'architecture *LeNet*, sont donnés en figure IV.32.

IV.4.3 Test du modèle

Pour tester la validité du modèle construit, l'image figure IV.33 d'un panneau de signalisation ne faisant pas partie de l'ensemble de données initial a été utilisée. Le modèle nous a prédit que cette image appartient à la classe 29, figure IV.34. Pour vérifier si le modèle a bien prédit la classe de l'image, il suffit de se référer à l'annexe page 75. Nous concluons que le modèle a bel et bien associé l'image à la classe lui correspondant.

IV.5 Conclusion

Dans ce chapitre, nous avons décrit le travail réalisé et analysé les résultats obtenus lors de la simulation du modèle qui prédit l'angle de braquage de la voiture autonome. Au vu de ces résultats, nous avons conclu que le modèle 2 est le plus performant, ayant parcouru avec succès la totalité des deux circuits. Nous avons, ensuite, crée un algorithme qui détectes les lignes sur une voie droite. Enfin, nous avons créé un modèle qui classifie les panneaux de signalisation en utilisant comme ensemble de données les panneaux de signalisation allemands.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 28, 28, 60)        1560

 conv2d_1 (Conv2D)           (None, 24, 24, 60)        90060

 max_pooling2d (MaxPooling2D  (None, 12, 12, 60)       0
 )

 conv2d_2 (Conv2D)           (None, 10, 10, 30)        16230

 conv2d_3 (Conv2D)           (None, 8, 8, 30)          8130

 max_pooling2d_1 (MaxPooling  (None, 4, 4, 30)         0
 2D)

 flatten (Flatten)           (None, 480)               0

 dense (Dense)               (None, 500)               240500

 dropout (Dropout)           (None, 500)               0

 dense_1 (Dense)             (None, 43)                21543

=================================================================
Total params: 378,023
Trainable params: 378,023
Non-trainable params: 0
_________________________________________________________________
None
```

Fig. IV.32 – Résumé du modèle modifié de LeNet



Fig. IV.33 – Image test du modèle LeNet [86]

```
predicted sign: [29]
```

Fig. IV.34 – Résultat de la prédiction du modèle LeNet

Conclusion générale

Vu le nombre alarmant et toujours croissant des accidents de la route, les constructeurs automobiles se penchent vers l'industrie des véhicules autonomes espérant de la sorte participer à la lutte contre ce fléau.

Dans ce mémoire, nous avons d'abord vu l'évolution des voitures autonomes surtout avec l'expansion fulgurante de l'intelligence artificielle. Nous avons étudié et simulé une voiture autonome. Nous avons utilisé l'architecture du réseau de convolution de bout en bout NVIDIA afin de générer un modèle qui prédit l'angle de braquage de la voiture après la fin du processus d'apprentissage. Le modèle que nous avons créé a pu finir avec succès le circuit de l'apprentissage et a pu généraliser son comportement en finissant avec succès sur un circuit lui étant inconnu. Ensuite, nous avons pu détecter les lignes issues d'une image de voie droite puis avons généralisé cette détection avec succès sur une vidéo. Par la suite, nous avons modifié l'architecture du réseau de convolution LeNet pour classifier des panneaux de signalisation.

Comme perspectives de travail, Nous pourrions aborder les points suivants :

- Ajouter d'autres fonctions d'augmentation de données. Ce qui permettra à notre modèle d'être performant quelles que soient les conditions météorologiques.
- Combiner le modèle de classification de panneaux de signalisation et de la prédiction de l'angle de braquage afin de rendre plus réaliste la simulation.
- Étudier et réaliser un prototype réel de voiture autonome implémenté du modèle.
- Intégrer d'autres capteurs à la réalisation précédente pour avoir des résultats encore plus fiables ce qui permettra de mieux détecter la route et les éventuels obstacles.

Bibliographie

[4] G. Kermorgant et O. Siary, « Is the Law Ready for Autonomous Cars ? », en, in *Energy Consumption and Autonomous Driving*, J. Langheim, éd., sér. Lecture Notes in Mobility, Cham : Springer International Publishing, 2016, p. 89-98, isbn : 978-3-319-19818-7. doi : 10.1007/978-3-319-19818-7_10.

[7] K. Bimbraw, « Autonomous Cars : Past, Present and Future - A Review of the Developments in the Last Century, the Present Scenario and the Expected Future of Autonomous Vehicle Technology : » en, in *Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics*, Colmar, Alsace, France : SCITEPRESS - Science, 2015, p. 191-198, isbn : 978-989-758-122-9. doi : 10.5220/0005540501910198. adresse : http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0005540501910198.

[9] M. McHugh, « Tesla's Cars Now Drive Themselves, Kinda », en-US, *Wired*, oct. 2015, Section : tags, issn : 1059-1028. adresse : https://www.wired.com/2015/10/tesla-self-driving-over-air-update-live/ (visité le 12/06/2022).

[11] J. Berrada, « Analyse et optimisation technico-économiques des nouveaux services de mobilité basés sur l'usage des véhicules autonomes », en, p. 255,

[17] D. Albert, M. Fassi, J.-F. Robin et al., « Vehicule Autonome : Etat de l'art, capture et traitement de l'information embarquees », fr, Institut national des sciences appliquées de Rouen, Rouen, rapp. tech., 2017, p. 27.

[18] N. Naas et S. Riad, « Détermination de la position d'un objet à l'aide d'un couple de caméras », fr, Mémoire de fin d'études pour l'obtention du diplôme de master en Informatique, Université Abd Elhamid Ibn Badis, Mostaganem, 2012. adresse : http://e-biblio.univ-mosta.dz/bitstream/handle/123456789/9359/MINF02.pdf.

[22] J. Ondruš, E. Kolla, P. Vertaľ et Ž. Šarić, « How Do Autonomous Cars Work ? », en, *Transportation Research Procedia*, LOGI 2019 - Horizons of Autonomous Mobility in Europe, t. 44, p. 226-233, jan. 2020, issn : 2352-1465. doi : 10.1016/j.trpro.2020.02.049. adresse : https://www.sciencedirect.com/science/article/pii/S2352146520300995.

[24] N. Monot, « Des systèmes d'aide à la conduite au véhicule autonome connecté », fr, Thèse de doctorat, Université de Bordeaux, 2019. adresse : https://tel.archives-ouvertes.fr/tel-02347511/document.

[25] S. J. Russell, P. Norvig et E. Davis, *Artificial intelligence : a modern approach*, 3rd ed, sér. Prentice Hall series in artificial intelligence. Upper Saddle River : Prentice Hall, 2010, isbn : 978-0-13-604259-4. adresse : https://zoo.cs.yale.edu/classes/cs470/materials/aima2010.pdf.

[26] R. Chandaliya, « Tele Stroke System for Stroke Detection », Master of Science, National University of Ireland, Galway, mai 2020. doi : 10.13140/RG.2.2.22376.90881.

[27] A. Munoz, « Machine Learning and Optimization », en-US, Courant Institute of Mathematical Science, New York, rapp. tech. adresse : https://cims.nyu.edu/~munoz/files/ml_optimization.pdf.

[28] G. Shaheamlung, H. Kaur et M. Kaur, « A Survey on machine learning techniques for the diagnosis of liver disease », in *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, juin 2020, p. 337-341. doi : 10.1109/ICIEM48762.2020.9160097.

[29] J. Schmidhuber, « Deep learning in neural networks : An overview », en, *Neural Networks*, t. 61, p. 85-117, jan. 2015, issn : 0893-6080. doi : 10.1016/j.neunet.2014.09.003. adresse : https://www.sciencedirect.com/science/article/pii/S0893608014002135.

[31] C. Touzet, *LES RESEAUX DE NEURONES ARTIFICIELS, INTRODUCTION AU CONNEXIONNISME*, fr. EC2, 1992. doi : 10/document. adresse : https://hal-amu.archives-ouvertes.fr/hal-01338010.

[32] K. Dahmane, « Analyse d'images par méthode de Deep Learning appliquée au contexte routier en conditions météorologiques dégradées », fr, thèse de doct., Université Clermont Auvergne [2017-2020], juin 2020. adresse : https://tel.archives-ouvertes.fr/tel-03022934.

[33] F. Bre, J. M. Gimenez et V. D. Fachinotti, « Prediction of wind pressure coefficients on building surfaces using artificial neural networks », en, *Energy and Buildings*, t. 158, p. 1429-1441, jan. 2018, issn : 0378-7788. doi : 10.1016/j.enbuild.2017.11.045. adresse : https://www.sciencedirect.com/science/article/pii/S0378778817325501.

[38] Y. S. Chowdhury, R. Dasgupta et S. Nanda, « Analysis of Various Optimizer on CNN model in the Application of Pneumonia Detection », in *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*, mai 2021, p. 417-421. doi : 10.1109/ICSPC51351.2021.9451768.

[40] M. Riedmiller, « Advanced supervised learning in multi-layer perceptrons — From backpropagation to adaptive learning algorithms », en, *Computer Standards & Interfaces*, t. 16, n° 3, p. 265-278, juill. 1994, Publisher : North-Holland, issn : 0920-5489. doi : 10.1016/0920-5489(94)90017-5. adresse : https://www.sciencedirect.com/science/article/abs/pii/0920548994900175.

[41] K. P. Murphy, *Machine learning : a probabilistic perspective*, sér. Adaptive computation and machine learning series. Cambridge, MA : MIT Press, 2012, isbn : 978-0-262-01802-9. adresse : http://noiselab.ucsd.edu/ECE228/Murphy_Machine_Learning.pdf.

[42] S. Sah, « Machine Learning : A Review of Learning Types », en, juill. 2020, Publisher : Preprints. doi : 10.20944/preprints202007.0230.v1. adresse : https://www.preprints.org/manuscript/202007.0230/v1 (visité le 16/06/2022).

[44] P. Bajaj, « Introduction to Machine Learning 2e Ethem Alpaydin », adresse : https://www.academia.edu/35741465/Introduction_to_Machine_Learning_2e_Ethem_Alpaydin.

[45] C. Robert, « Machine Learning, a Probabilistic Perspective », *CHANCE*, Publisher : Taylor & Francis, t. 27, n° 2, p. 62-63, avr. 2014, issn : 0933-2480. doi : 10.1080/09332480.2014.914768. adresse : https://doi.org/10.1080/09332480.2014.914768.

[46] F. Payette, « Apprentissage semi-supervisé par réduction de dimensionnalité non linéaire », fra, mém. de mast., Université de Montréal, 2004. adresse : http://hdl.handle.net/1866/16641.

[47] DARPA-IPTO, « Autonomous Off-Road Vehicle Control Using End-to-End Learning », rapp. tech., juill. 2004. adresse : https://cs.nyu.edu/~yann/research/dave/dave-final-report.pdf.

[48] S. Sheikholeslami, « Ablation Programming for Machine Learning (M.Sc. Thesis, KTH Royal Institute of Technology) », thèse de doct., juill. 2019. doi : 10.13140/RG.2.2.27959.06567.

[51] H. N. K. Al-Behadili, K. R. Ku-Mahamud et R. Sagban, « Rule pruning techniques in the ant-miner classification algorithm and its variants : A review », in *2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, avr. 2018, p. 78-84. doi : 10.1109/ISCAIE.2018.8405448.

[52] R. Gencay et M. Qi, « Pricing and hedging derivative securities with neural networks : Bayesian regularization, early stopping, and bagging », *IEEE Transactions on Neural Networks*, t. 12, n° 4, p. 726-734, juill. 2001, Conference Name : IEEE Transactions on Neural Networks, issn : 1941-0093. doi : 10.1109/72.935086.

[54] M. S. M. R. K. P. Pujari, *Practical Convolutional Neural Networks*, Undetermined. Place of publication not identified : Packt Publishing, 2018, OCLC : 1035779157, isbn : 978-1-78839-230-3. adresse : https://proquest.safaribooksonline.com/9781788392303.

[57] J. Brownlee, *Deep Learning for Computer Vision - Image Classification, Object Detection and Face Recognition in Python*, en-US, Machine Learning Mastery. 2019. adresse : https://fr.b-ok.africa/book/16857790/c2ba9f.

[59] D. Sarkar, R. Bali et T. Sharma, « Deep Learning for Computer Vision », en, in *Practical Machine Learning with Python : A Problem-Solver's Guide to Building Real-World Intelligent Systems*, D. Sarkar, R. Bali et T. Sharma, éd., Berkeley, CA : Apress, 2018, p. 499-520, isbn : 978-1-4842-3207-1. doi : 10.1007/978-1-4842-3207-1_12. adresse : https://doi.org/10.1007/978-1-4842-3207-1_12.

[60] M. García-Ordás, J. Benítez-Andrades, I. García, C. Benavides et H. Alaiz Moreton, « Detecting Respiratory Pathologies Using Convolutional Neural Networks and Variational Autoencoders for Unbalancing Data », *Sensors*, t. 20, fév. 2020. doi : 10.3390/s20041214.

[63] A. Gridi, «Un Outil de Deep Learning pour les données textuelles», fr, Mémoire de Master, Universite L'Arbi Ben M'Hidi, Oum El Bouaghi, 2019. adresse : http://bib.univ-oeb.dz:8080/jspui/bitstream/123456789/10492/1/m%C3%A9moire%20.pdf.

[66] D. F. Atrevi, «Détection et analyse des évènements rares par vision, dans un contexte urbain ou péri-urbain», fr, thèse de doct., Université d'Orléans, juin 2019. adresse : https://tel.archives-ouvertes.fr/tel-02985957 (visité le 19/06/2022).

[68] I. Sonata, Y. Heryadi, L. Lukas et A. Wibowo, «Autonomous car using CNN deep learning algorithm», *Journal of Physics : Conference Series*, t. 1869, nº 1, p. 012 071, avr. 2021, issn : 1742-6588, 1742-6596. doi : 10.1088/1742-6596/1869/1/012071. adresse : https://iopscience.iop.org/article/10.1088/1742-6596/1869/1/012071.

[70] Openclassrooms, *Apprenez à programmer en Python*. adresse : http://user.oc-static.com/pdf/223267-apprenez-a-programmer-en-python.pdf.

Webographie

[1] *Planetoscope - Statistiques : Mortalité - Morts d'accidents de la route dans le Monde*. adresse : https://www.planetoscope.com/mortalite/1270-.html (visité le 07/06/2022).

[2] Centre National de Ressources Textuelles et Lexicales, *AUTONOME : Définition de AUTONOME*. adresse : https://www.cnrtl.fr/definition/academie9/autonome (visité le 29/05/2022).

[3] É. Larousse, *Définitions : voiture - Dictionnaire de français Larousse*, fr. adresse : https://www.larousse.fr/dictionnaires/francais/voiture/82405 (visité le 06/06/2022).

[5] *SAE J3016 automated-driving graphic*, en, Last Modified : 2020-05-15T14:03:15-04:00. adresse : https://www.sae.org/site/news/2019/01/sae-updates-j3016-automated-driving-graphic (visité le 28/04/2022).

[6] *Histoire des voitures autonomes*, en. adresse : http://stringfixer.com/fr/History_of_autonomous_cars (visité le 27/05/2022).

[8] *BMW Personal CoPilot : la conduite autonome | BMW.fr*, fr-FR. adresse : https://www.bmw.fr/fr/topics/univers/bmw-conduite-autonome.html (visité le 12/06/2022).

[10] *Autonomous Driving FAQ Let's Talk Autonomous Driving*, en-US. adresse : https://ltad.com/about/autonomous-driving-faq.html (visité le 11/06/2022).

[12] Toyota, *Toyota | Touch : la magie de la mobilité : e-Palette*, fr. adresse : https://www.toyota.fr/decouvrez-toyota/start-your-impossible/solutions-de-mobilite/e-palette (visité le 14/06/2022).

[13] J. Cusack, *How driverless cars will change our world*, en. adresse : https://www.bbc.com/future/article/20211126-how-driverless-cars-will-change-our-world (visité le 04/06/2022).

[14] *Volvo Cars présente son système d'assistance à la conduite autonome Ride Pilot en Californie*, fr-fr. adresse : https://www.media.volvocars.com/fr/fr-fr/media/pressreleases/292917/volvo-cars-presente-son-systeme-dassistance-a-la-conduite-autonome-ridepilot-en-californie (visité le 12/06/2022).

[15] *Baidu et Pony.ai peuvent proposer des courses à bord de leurs véhicules autonomes sans chauffeur*. adresse : https://www.usine-digitale.fr/article/baidu-et-pony-ai-peuvent-proposer-des-courses-a-bord-de-leurs-vehicules-autonomes-sans-chauffeur.N2000042 (visité le 12/06/2022).

[16] G. T., *Qu'est-ce qu'une voiture autonome ?*, fr-FR, avr. 2022. adresse : https://www.ferrari-by-logic3.com/voiture-autonome/ (visité le 13/06/2022).

[19] *Ondes électromagnétiques : quels effets sur le corps humain ?*, fr. adresse : https://www.pediact.com/blog/ondes-electromagnetiques-quels-effets-sur-le-corps-humain (visité le 19/06/2022).

[20] *JIDU Robo-01, la voiture autonome de Baidu*, fr, juin 2022. adresse : https://www.automobile-propre.com/jidu-robo-01-la-voiture-autonome-de-baidu/ (visité le 12/06/2022).

[21] *Le match Ford-GM de la voiture autonome*, fr, Section : Tech, juin 2021. adresse : https://www.autonews.fr/hi-tech/le-match-ford-gm-de-la-voiture-autonome-98693 (visité le 13/06/2022).

[23] K. Melnick, *The Apple Car Could Feature VR Technology And No Windows*, en-US, mai 2022. adresse : https://vrscout.com/news/the-apple-car-could-feature-vr-technology-and-no-windows/ (visité le 29/05/2022).

[30] J. Hertz, A. Krogh et R. G. Palmer, *Introduction to the Theory of Neural Computation : Physics Today : Vol 44, No 12*. adresse : https://physicstoday.scitation.org/doi/10.1063/1.2810360 (visité le 16/06/2022).

[34] M. Bastien, *Fonction d'activation*, fr-FR, sept. 2018. adresse : https://deeplylearning.fr/cours-theoriques-deep-learning/fonction-dactivation/ (visité le 12/06/2022).

[35] L. R, *Focus : Le Réseau de Neurones Convolutifs*, fr-FR, jan. 2019. adresse : https://penseeartificielle.fr/focus-reseau-neurones-convolutifs/ (visité le 11/06/2022).

[36] T. Keldenich, *Fonction d'activation, comment ça marche ? - Une explication simple*, fr-FR, fév. 2021. adresse : https://www.inside-machinelearning.com/fonction-dactivation-comment-ca-marche-une-explication-simple/ (visité le 16/06/2022).

[37] R. Kwiatkowski, *Gradient Descent Algorithm — a deep dive*, en, mai 2021. adresse : https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21 (visité le 25/06/2022).

[39] V. Bushaev, *Adam — latest trends in deep learning optimization.* en, oct. 2018. adresse : https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c (visité le 25/06/2022).

[43] R. Raj, *Supervised, Unsupervised, And Semi-Supervised Learning With Real-Life Use-case*. adresse : https://www.enjoyalgorithms.com/blogs/supervised-unsupervised-and-semisupervised-learning (visité le 16/06/2022).

[49] D. Nikolaiev, *Overfitting and Underfitting Principles*, en-US. adresse : https://towardsdatascience.com/overfitting-and-underfitting-principles-ea8964d9c45c (visité le 26/06/2022).

[50] A. Bhande, *What is underfitting and overfitting in machine learning and how to deal with it.* en, mars 2018. adresse : https://medium.com/greyatom/what-is-underfitting-and-overfitting-in-machine-learning-and-how-to-deal-with-it-6803a989c76 (visité le 16/06/2022).

[53] J. Huber, *Batch normalization in 3 levels of understanding | by Johann Huber | Towards Data Science*. adresse : https://towardsdatascience.com/batch-normalization-in-3-levels-of-understanding-14c2da90a338 (visité le 25/06/2022).

[55] S. Guedira, *A scalable and component-based deep learning parallelism platform : an application to convolutional neural networks for medical imaging segmentation. - TEL - Thèses en ligne*. adresse : https://tel.archives-ouvertes.fr/tel-03011791/ (visité le 16/06/2022).

[56] Simplilearn, *Regression vs. Classification in Machine Learning for Beginners*, en, avr. 2022. adresse : https://www.simplilearn.com/regression-vs-classification-in-machine-learning-article (visité le 26/06/2022).

[58] J. Novet, *Deep learning startup Ersatz Labs suspends development of its cloud service*, oct. 2015. adresse : https://venturebeat.com/2015/10/31/deep-learning-startup-ersatz-labs-suspends-development-of-its-cloud-service/.

[61] A. Amidi et S. Amidi, *Pense-bête de réseaux de neurones convolutionnels*, fr. adresse : https://stanford.edu/~shervine/l/fr/teaching/cs-230/pense-bete-reseaux-neurones-convolutionnels.

[62] Pierre, *Classification de pages Web via Deep Learning - Réseau de Neurones Convolutif*, fr-FR, jan. 2020. adresse : https://www.anakeyn.com/2020/01/13/classification-de-pages-web-via-deep-learning-reseau-de-neurones-convolutif/ (visité le 16/07/2022).

[64] *Convolutional Neural Network - javatpoint*, en. adresse : https://www.javatpoint.com/pytorch-convolutional-neural-network (visité le 19/06/2022).

[65] Y. Touzani, *Deep Learning : les réseaux de neurones récurrents (RNN)*, fr-FR, Section : IA/Data Science, août 2021. adresse : https://datavalue-consulting.com/deep-learning-reseaux-neurones-recurrents-rnn/ (visité le 19/06/2022).

[67] *End-to-End Deep Learning for Self-Driving Cars*, en-US, août 2016. adresse : https://developer.nvidia.com/blog/deep-learning-self-driving-cars/ (visité le 11/06/2022).

[69] Udacity, *Code source du simulateur Udacity*, original-date : 2017-02-08T03:31:26Z, juin 2022. adresse : https://github.com/udacity/self-driving-car-sim/blob/522bd7ad6784ad5de1f12f593e00c7d1d79ec34d/Assets/Standard%20Assets/Vehicles/Car/Scripts/CarController.cs (visité le 13/06/2022).

[71] *Python language de programmation apprendre cours tutoriaux django informatique dévelopement web internet script*, fr. adresse : https://python.doctor/ (visité le 23/06/2022).

[72] *Why Python for Machine Learning ? - Python Tutorial*. adresse : https://pythonbasics.org/why-python-for-machine-learning/ (visité le 23/06/2022).

[73] Lizengo, *Que signifie l'API ? Exemples d'utilisation des API*, fr-FR. adresse : https://lizengo.fr/uncategorized/que-signifie-api/ (visité le 23/06/2022).

[74] B. L., *TensorFlow : le framework de Machine Learning de Google*, fr-FR, jan. 2021. adresse : https://datascientest.com/tensorflow (visité le 23/06/2022).

[75] K. Team, *Keras documentation : About Keras*, en. adresse : https://keras.io/about/ (visité le 23/06/2022).

[76] NumPy community, *NumPy User Guide*, juin 2022. adresse : https://numpy.org/doc/stable/numpy-user.pdf.

[77] *What Is Matplotlib In Python ? How to use it for plotting ?*, en-CA. adresse : https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/ (visité le 23/06/2022).

[78] Mode Analytics, *Pandas | Python Library - Mode*, mai 2016. adresse : https://mode.com/python-tutorial/libraries/pandas/ (visité le 23/06/2022).

[79] T. Chaboud, *OpenCV, c'est quoi ?*, fr, sept. 2019. adresse : https://www.axopen.com/blog/2019/09/open-cv-cest-quoi/ (visité le 23/06/2022).

[80] J. JVC, *Scikit-Learn : guide de démarrage rapide en Machine Learning avec Python*, fr-FR, sept. 2021. adresse : https://www.data-transitionnumerique.com/scikit-learn-python/ (visité le 23/06/2022).

[81] *Project Jupyter*, en. adresse : https://jupyter.org (visité le 23/06/2022).

[82] Google Colaboratory, *Bienvenue dans Colaboratory*, fr. adresse : https://colab.research.google.com/?utm_source=scs-index#scrollTo=OwuxHmxllTwN (visité le 23/06/2022).

[83] K. Ji, *Creating a Lane Detection Program Using OpenCV-Python*, en, mars 2021. adresse : https://medium.com/nerd-for-tech/creating-a-lane-detection-program-using-opencv-python-ba82b2305d26 (visité le 15/06/2022).

[84] Nigama Vykari, *Line lanes detection Video*. adresse : https://raw.githubusercontent.com/Nigama-oss/Lane-Detection/master/test2.mp4.

[85] Jad Slim, *jadslim / german-traffic-signs — Bitbucket*. adresse : https://bitbucket.org/jadslim/german-traffic-signs/src/master/ (visité le 25/06/2022).

[86] *german-road-sign-bicycles-crossing-J2MRAJ.jpg (Image JPEG, 1300 × 956 pixels) - Redimensionnée (75%)*. adresse : https://c8.alamy.com/comp/J2MRAJ/german-road-sign-bicycles-crossing-J2MRAJ.jpg (visité le 03/06/2022).

Annexes

Annexe 1: Classes de plaques de signalisation

ClassId	SignName
0	Speed limit (20km/h)
1	Speed limit (30km/h)
2	Speed limit (50km/h)
3	Speed limit (60km/h)
4	Speed limit (70km/h)
5	Speed limit (80km/h)
6	End of speed limit (80km/h)
7	Speed limit (100km/h)
8	Speed limit (120km/h)
9	No passing
10	No passing for vechiles over 3.5 metric tons
11	Right-of-way at the next intersection
12	Priority road
13	Yield
14	Stop
15	No vechiles
16	Vechiles over 3.5 metric tons prohibited
17	No entry
18	General caution
19	Dangerous curve to the left
20	Dangerous curve to the right
21	Double curve
22	Bumpy road
23	Slippery road
24	Road narrows on the right

25	Road work
26	Traffic signals
27	Pedestrians
28	Children crossing
29	Bicycles crossing
30	Beware of ice/snow
31	Wild animals crossing
32	End of all speed and passing limits
33	Turn right ahead
34	Turn left ahead
35	Ahead only
36	Go straight or right
37	Go straight or left
38	Keep right
39	Keep left
40	Roundabout mandatory
41	End of no passing
42	End of no passing by vechiles over 3.5 metric tons

Tab. IV.1 – Classes des plaques de signalisation

ملخص

يبدو أن السيارات ذاتية القيادة هي أحد الحلول لتقليل عدد حوادث الطرق. بعد سرد تاريخ وتطور السيارات ذاتية القيادة،خاصة منذ إدخال الذكاء الاصطناعي، ركزت أطروحتنا على ثلاثة جوانب مهمة لعملها: أولاً ، لمحاكاة قيادة سيارة مستقلة ، استخدمنا بنية شبكة الالتفاف NVIDIA لإنشاء نموذج يتنبأ بزاوية توجيه السيارة. اجتاز النموذج بنجاح مسار التعلم وكذلك مسار كامل لم يكن معروف له ، مما يدل على أنه يمكن أن يعمم سلوكه. بعد ذلك، اكتشفنا خطوط الممر على طريق مستقيم الناتجة عن صورة ثم عممنا هذا الاكتشاف على فيديو. أخيرًا ، قمنا بتعديل بنية شبكة الالتفاف LeNet لتصنيف إشارات المرور.

كلمات مفتاحية: سيارة ذاتية القيادة ، التعلم العميق ، CNN ، الذكاء الاصطناعي ، محاكي Udacity ، الرؤية الحاسوبية.

Résumé

Dans ce mémoire, après avoir abordé l'évolution des voitures autonomes et notamment depuis l'introduction de l'intelligence artificielle, nous avons étudié trois aspects importants de leur fonctionnement. D'abord, pour simuler la conduite d'une voiture autonome, nous avons utilisé l'architecture du réseau de convolution NVIDIA afin de générer un modèle qui prédit l'angle de braquage de la voiture. Le modèle créé a réussi à finir avec succès le circuit d'apprentissage et a pu généraliser son comportement sur l'ensemble d'un circuit qui lui était inconnu. Ensuite, nous avons développé un code qui permet la détection des lignes de voie droite de circulation issues d'une image puis nous avons généralisé cette détection à une vidéo. Enfin, nous avons modifié l'architecture du réseau de convolution LeNet pour classifier des panneaux de signalisation.

Mots-clés : Voiture autonome, apprentissage en profondeur, CNN, intelligence artificielle, simulateur Udacity, vision par ordinateur.

Abstract

In this thesis, after having discussed the evolution of autonomous cars, especially since the introduction of artificial intelligence, we studied three important aspects of their operation. First, to simulate the driving of an autonomous car, we used the NVIDIA convolutional neural network architecture to generate a model that predicts the steering angle of the car. The model created successfully completed the training circuit and was able to generalise its behaviour to an entire unknown circuit. Then, we developed a straight lane lines detector from a source image and then generalised this detection to a video. Finally, we modified the architecture of the LeNet convolutional neural network to classify traffic signs.

Keywords : Autonomous car, deep learning, CNN, artificial intelligence, Udacity simulator, computer vision.