

Université de Carthage

Ecole supérieure des communications de Tunis

Projet n°4:

**End-to-End Learning pour la conduite autonome: de la simulation au monde réel.**

**Préparé par :** Manel Jrad

Lamia Hamida Brahim

**Encadrées par :** M. Fethi Tlili

Année universitaire

2019-2020

Table des matières

[Introduction générale 1](#_Toc28880175)

[Chapitre1 : Etude préalable et présentation du projet 2](file:///C:\Users\DELL\Downloads\rapport_Tutoré.docx#_Toc28880176)

[1- Introduction 3](#_Toc28880177)

[2- Présentation du projet 3](#_Toc28880178)

[2.1. Etude de marché 3](#_Toc28880179)

[2.2. Solution proposé**e** 3](#_Toc28880180)

[2.3. Architecture générale de la solution 4](#_Toc28880181)

[a- Le simulateur Carla 4](#_Toc28880182)

[3- Conclusion 5](#_Toc28880183)

[Chapitre 2 : Etat de l’art 6](file:///C:\Users\DELL\Downloads\rapport_Tutoré.docx#_Toc28880184)

[1- Introduction 7](#_Toc28880185)

[2- Environnement du travail 7](#_Toc28880186)

[2.1. Outils matériels : 7](#_Toc28880187)

[2.2. Outils logiciels 7](#_Toc28880188)

[*2.2.1. Le simulateur Carla* 7](#_Toc28880189)

[2.3. Technologie : 8](#_Toc28880190)

[3- Conclusion 10](#_Toc28880191)

[Chapitre 3 : Réalisation 11](#_Toc28880192)

[1- Introduction 13](#_Toc28880193)

[2- Collection des données 13](#_Toc28880194)

[2.1. Procédés de collection de données 13](#_Toc28880195)

[2.2. Résultats de la collection de données 14](#_Toc28880196)

[3- Augmentation des données 14](#_Toc28880197)

[3.1. Procédés d’augmentation de données 15](#_Toc28880198)

[3.2. Comparaison de la diversification de la base de données avant et après l’augmentation des données 15](#_Toc28880199)

[3.3. Résultat de l’augmentation de données 16](#_Toc28880200)

[4- Préparation de la base de données pour l’entrainement 16](#_Toc28880201)

[5- Implémentation du réseau de neurones 16](#_Toc28880202)

[5.1. Architecture de CNN : Pilotnet 16](#_Toc28880203)

[5.2. Résultat de la prédiction 17](#_Toc28880204)

[5.3. Evaluation du modèle 17](#_Toc28880205)

[6- Implémentation du modèle sur Carla 17](#_Toc28880206)

[7- Conclusion 18](#_Toc28880207)

[Conclusion 19](#_Toc28880208)

[Bibliographie 20](#_Toc28880209)

# Table des figures

[Figure 1: Principe du End to End Learning 3](#_Toc28880213)

[Figure 2:Composition de Carla 4](#_Toc28880214)

[Figure 3: Architecture suivie dans le projet 5](#_Toc28880215)

[Figure 4: Logo de Carla 7](#_Toc28880216)

[Figure 5: Logo d'Anaconda Prompt 7](#_Toc28880217)

[Figure 6: logo de Google Colab 8](#_Toc28880218)

[Figure 7: logo du langage Python 8](#_Toc28880219)

[Figure 8: Logos de Keras et TensorFlow 9](#_Toc28880220)

[Figure 9: L'architecture Pilotnet 10](#_Toc28880221)

[Figure 10: Le serveur Carla 13](#_Toc28880222)

[Figure 11: la fenêtre Pygame pour la conduite du véhicule et la collection des données 13](#_Toc28880223)

[Figure 12: Fichier où les images routières sont stockées 14](#_Toc28880224)

[Figure 13: Les mesures d'orientation 14](#_Toc28880225)

[Figure 14: Effet de miroir 15](#_Toc28880226)

[Figure 15: Histogramme de la base de données avant l'augmentation des données 15](#_Toc28880227)

[Figure 16: Histogramme de la base de données après l'augmentation des données 15](#_Toc28880228)

[Figure 17: Les statistiques de la base de données résultante 16](#_Toc28880229)

[Figure 18: Architecture Pilotnet 16](#_Toc28880230)

[Figure 19: Les valeurs des angles de déviation prédites 17](#_Toc28880231)

[Figure 20: Evaluation des mesures prédites 17](#_Toc28880232)

# Introduction générale

Un véhicule autonome est un véhicule automobile apte à rouler, sur route ouverte, sans intervention d'un conducteur. Le concept vise à développer et produire un véhicule pouvant réellement circuler sur la voie publique dans le trafic sans intervention humaine en toutes situations, à terme. C'est une application typique du domaine de la robotique mobile dans laquelle de nombreux acteurs sont engagés.

La notion de voiture autonome peut recouvrir aussi bien un véhicule totalement autonome ou bien un véhicule « semi-autonome » disposant de différents systèmes d'aide à la conduite automatisée, par exemple en ville ou sur autoroute, ou encore de système de stationnement automatisé.

En contrepartie, Au cœur de la voiture autonome, on trouve les mêmes [algorithmes](https://dataanalyticspost.com/Lexique/algorithme/) d’[apprentissage automatique](https://dataanalyticspost.com/Lexique/machine-learning/) que ceux utilisés pour la [classification](https://dataanalyticspost.com/Lexique/classification/) des images ou la reconnaissance de la parole. Mais avant de pouvoir embarquer dans le véhicule, ces algos font l’objet de tests stricts. Ces derniers sont des algorithmes de Réseau de neurones Convolutif CNN qui ont révolutionné la reconnaissance de formes. Avant l'adoption généralisée des CNN, la plupart des tâches de reconnaissance de formes étaient effectuées à l'aide d'une étape initiale d'extraction de caractéristiques fabriquée à la main suivie d'un classificateur. La percée des CNN est que les fonctionnalités sont apprises automatiquement à partir d'exemples de formation. L'approche CNN est particulièrement puissante dans les tâches de reconnaissance d'image car l'opération de convolution capture la nature 2D des images. De plus, en utilisant les noyaux de convolution pour numériser une image entière, relativement peu de paramètres doivent être appris par rapport au nombre total d'opérations. Alors que les CNN avec des fonctionnalités apprises sont en usage commercial depuis plus de vingt ans, leur adoption a explosé ces dernières années en raison de deux développements récents. Premièrement, de grands ensembles de données étiquetés tels que le défi de reconnaissance visuelle à grande échelle (ILSVRC) sont devenus disponibles pour la formation et la validation. Deuxièmement, des algorithmes d'apprentissage CNN ont été mis en œuvre sur les unités de traitement graphique massivement parallèles (GPU), ce qui accélère considérablement l'apprentissage et l'inférence.

Dans ce rapport, nous décrivons un modèle CNN qui va au-delà de la reconnaissance des images routières. Il apprend tout le pipeline de traitement nécessaire pour diriger une automobile. Les bases de ce projet ont été dans un projet tuteuré connu sous le nom de « **End-to-End Learning pour la conduite autonome : de la simulation au monde réel** » proposé par Monsieur Fethi Tlili et Monsieur Rémi Boutteau.



# Chapitre1 : Etude préalable et présentation du projet

## Introduction

L’objectif de ce chapitre est de mettre notre travail dans son contexte général. Pour cela, on va commencer par présenter le cadre du projet. Ensuite, on va présenter l’architecture générale du sujet et enfin on va mettre l'accent sur la solution.

## Présentation du projet

### 2.1. Etude de marché

Des grandes entreprises comme Google, Uber et Tesla conceptualisent et déploient des voitures autonomes depuis plusieurs années. En effet, la voiture autonome est déjà une réalité dans certains pays, notamment en Californie aux USA. En outre, parmi les solutions proposées pour concevoir une voiture autonome est l’utilisation des réseaux de neurones.

Ces réseaux de neurones (en particulier le Deep Learning) sont entraînés avec des Datasets de plusieurs milliers d’heures de conduite et peuvent ainsi répondre à chaque cas possible. De plus, la technologie utilisable est fiable et les recherches dans ce domaine sont en croissance. Parmi les algorithmes utilisés :

* Boucle robotique classique et assez simple : asservissement commande du volant selon estimation position par rapport à route
* Détection et catégorisation/reconnaissance d’objets (voitures, piétons, vélos, panneaux, feux, …)

L’idée d’améliorer l’intelligence artificielle des véhicules continue de croître dans la tête des hauts décisionnaires des grandes entreprises comme Google.

### 2.2. Solution proposé**e**

Les véhicules n’ont cessé de gagner en autonomie ces dernières années ceci grâce aux progrès fulgurants accomplis en matière d’[intelligence artificielle](https://dataanalyticspost.com/Lexique/intelligence-artificielle-ia/), en particulier par certains algorithmes, dits de Réseau de Neurones Convolutif ou CNN. Ces méthodes d’apprentissage automatique, basées sur l’exemple, sont notamment utilisées pour la reconnaissance des routes sur les photos. Désormais, elles se retrouvent au cœur du fonctionnement des voitures sans pilote.

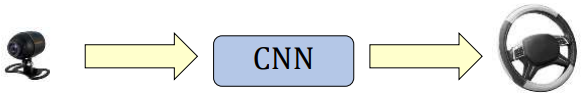
Dans ce contexte s’implique notre projet qui a pour but de développer un réseau de neurones CNN qui prend en entrée des images routières prises par des caméras embarquées sur le véhicule et donne en sortie les actions à appliquer au volant.

Figure 1: Principe du End to End Learning

### 2.3. Architecture générale de la solution

#### 2.3.1. Composants de l’architecture

##### a- Le simulateur Carla :

Le simulateur CARLA est un simulateur gratuit et open-source propulsé par Unreal Engine 4  (UE4)  qui été développé de fond en comble pour soutenir le développement, la formation et la validation de systèmes de conduite autonomes. En plus du code et des protocoles open source, CARLA fournit des actifs numériques ouverts (agencements urbains, bâtiments, véhicules) qui ont été créés à cet effet et peuvent être utilisés librement. La plate-forme de simulation prend en charge la spécification flexible des suites de capteurs, les conditions environnementales, le contrôle total de tous les acteurs statiques et dynamiques, la génération de cartes et bien plus encore.

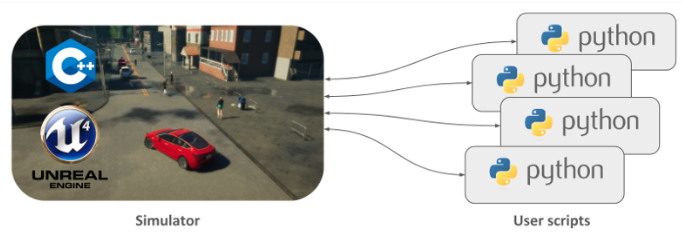
CARLA se compose principalement de deux modules, le simulateur CARLA et le module API CARLA Python. Le simulateur effectue la plupart des travaux lourds, contrôle la logique, la physique et le rendu de tous les acteurs et capteurs de la scène; il nécessite une machine avec un GPU dédié pour fonctionner. L'API CARLA Python est un module que vous pouvez importer dans vos scripts Python, il fournit une interface pour contrôler le simulateur et récupérer les données. Avec cette API Python, vous pouvez, par exemple, contrôler n'importe quel véhicule dans la simulation, y attacher des capteurs et relire les données générées par ces capteurs. La plupart des aspects de la simulation sont accessibles à partir de notre API Python, et d'autres le seront dans les prochaines versions.

Figure 2:Composition de Carla

##### b- Intelligence artificielle et Deep Learning

Le deep learning ou l’apprentissage profond est un type d'[intelligence artificielle](https://www.futura-sciences.com/tech/definitions/informatique-intelligence-artificielle-555/) dérivé de la machine learning (apprentissage automatique) où la machine est capable d'apprendre par elle-même, contrairement à la programmation où elle se contente d'exécuter à la lettre des règles prédéterminées.

Le deep Learning s'appuie sur un réseau de [neurones artificiels](https://www.futura-sciences.com/tech/actualites/intelligence-artificielle-synapse-artificielle-reproduire-fonctionnement-neurones-34674/) s'inspirant du [cerveau](https://www.futura-sciences.com/sante/definitions/biologie-cerveau-3125/) humain. Ce réseau est composé de dizaines voire de centaines de « couches » de [neurones](https://www.futura-sciences.com/sante/definitions/biologie-neurone-209/), chacune recevant et interprétant les informations de la couche précédente. Le système apprendra par exemple à reconnaître les lettres avant de s'attaquer aux mots dans un texte, ou détermine s'il y a un visage sur une photo avant de découvrir de quelle personne il s'agit.

À chaque étape, les « mauvaises » réponses sont éliminées et renvoyées vers les niveaux en amont pour ajuster le modèle mathématique. Au fur et à mesure, le programme réorganise les informations en blocs plus complexes. Lorsque ce modèle est par la suite appliqué à d'autres cas, il est normalement capable de reconnaître un chat sans que personne ne lui ait jamais indiqué qu'il n'ait jamais appris le concept de chat. Les données de départ sont essentielles : plus le système accumule d'expériences différentes, plus il sera performant.

#### 2.3.2. L’Architecture à mettre en place

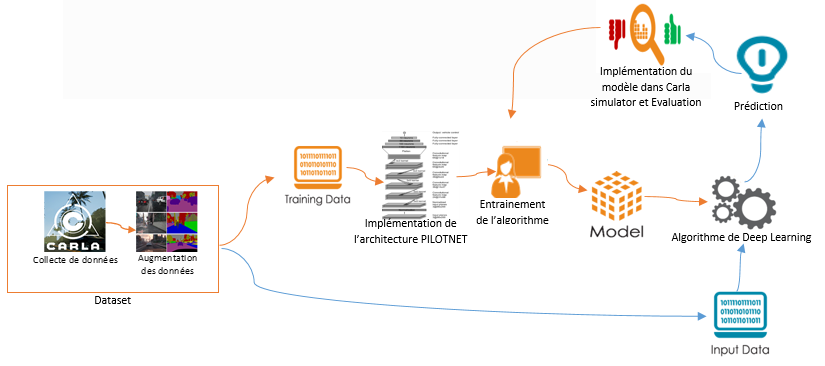


Figure 3: Architecture suivie dans le projet

Pour atteindre le but ultime du projet, et comme la figure 2 décrit, on a utilisé un simulateur appelé le simulateur Carla pour collecter les données qui sont les images des routes et les mesures d’orientation telles que le « steering angle » c’est-à-dire l’angle d’orientation et la vitesse , puis on a essayé d’augmenter la taille de la base de données collectée qu’on a les décomposées pour faire entrainer une partie en suivant une architecture appelée PILOTNET et pour utiliser l’autre partie pour la validation du modèle qui sera implémenté dans le simulateur Carla et ainsi sera évalué. On fait ça jusqu’à ce que le modèle devienne performant. Toutes ces technologies et tous ces outils mentionnés ci-dessus seront définis par détaille dans les prochaines parties.

## Conclusion

Tout au long de ce chapitre, nous avons commencé par présenter notre sujet, le but et l’architecture de notre solution. Dans le chapitre suivant on va discuter clairement notre model et les différents outils.



# Chapitre 2 : Etat de l’art

## Introduction

Dans ce chapitre, nous allons définir les différents outils matériels et logiciels nécessaires pour la conception du modèle CNN pour la conduite autonome.

## Environnement du travail

### 2.1. Outils matériels :

|  |  |
| --- | --- |
| Processeur | Intel® Core™ i5-4005 CPU @ 1.700GHz |
| Disque dur | 1 To |
| Mémoire Installée | 8.00 Go |
| Système d’exploitation | Windows 10 Professionnel (x64) |

### 2.2. Outils logiciels

#### Image result for carla simulator2.2.1. Le simulateur Carla :

Figure 4: Logo de Carla

CARLA est un simulateur open source pour la recherche de conduite autonome. Il a été développé pour soutenir le développement, la formation et la validation de systèmes de conduite autonomes. En plus du code et des protocoles open source, CARLA fournit des actifs numériques ouverts (agencements urbains, bâtiments, véhicules) qui ont été créés à cet effet et peuvent être utilisés librement.

La plate-forme de simulation prend en charge la spécification flexible des suites de capteurs et des conditions environnementales.

#### Image result for Anaconda logo2.2.2. Anaconda

Figure 5: Logo d'Anaconda Prompt

Anaconda est une distribution [libre et open source](https://fr.wikipedia.org/wiki/Free/Libre_Open_Source_Software) des langages de programmation [Python](https://fr.wikipedia.org/wiki/Python_(langage)) et [R](https://fr.wikipedia.org/wiki/R_(langage)) appliqué au développement d'applications dédiées à la science des données et à l'apprentissage automatique (traitement de données à grande échelle, analyse prédictive, calcul scientifique), qui vise à simplifier la gestion des paquets et de déploiement.

Directement à partir de cette plateforme, les scientifiques des données peuvent développer et déployer l'IA et l'apprentissage automatique. Anaconda fournit les outils nécessaires pour facilement:

• Collecter des données à partir de fichiers et de bases de données

• Gérer les environnements avec Conda (toutes les dépendances de package sont prises en charge au moment deTélécharger)

• Partager, collaborer et reproduire des projets

• Déployer des projets en production d'un simple clic sur un bouton

#### 2.2.3. Google Colaboratory

Figure 6: logo de Google Colab

Google Colab  ou Colaboratory est un service cloud, offert par Google (gratuit), basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l’apprentissage automatique. Cette plateforme permet d’entraîner des modèles de Machine Learning directement dans le cloud. Sans donc avoir besoin d’installer quoi que ce soit sur notre ordinateur à l’exception d’un navigateur.

#### 2.2.4. GPU

Le GPU (Graphics Processing Unit) est considéré comme le cœur du Deep Learning, une partie de l'intelligence artificielle. Il s'agit d'un processeur à puce unique utilisé pour des calculs graphiques et mathématiques étendus, ce qui libère des cycles de CPU (unité centrale de traitement) pour d'autres tâches.

Comme son nom l'indique, il a été développé à la base pour les calculs intensifs liés à l'affichage, mais la parallélisassions de son traitement de l'information le rend aussi particulièrement efficace pour le deep learning.

### 2.3. Technologie :

#### 2.3.1. Langage de programmation python:

Figure 7: logo du langage Python

Python est un langage de programmation puissant et facile à apprendre. Il dispose de structures de données de haut niveau et permet une approche simple mais efficace de la programmation orientée objet. Parce que sa syntaxe est élégante, que son typage est dynamique et qu'il est interprété, Python est un langage idéal pour l'écriture de scripts et le développement rapide d'applications dans de nombreux domaines et sur la plupart des plateformes.

#### Image result for keras logo2.3.2. Keras:

Figure 8: Logos de Keras et TensorFlow

Keras est une API de réseaux de neurones de haut niveau, écrite en Python et interfaçable avec TensorFlow, CNTK et Theano. Elle a été développée pour objectif de permettre des expérimentations rapides. Être capable d’aller de l’idée au résultat avec le plus faible délai possible étant la clef d’une recherche efficace.

* Permet le prototypage rapide et facile (par sa modularité et son extensibilité).
* Supporte à la fois les réseaux convolutifs et les réseaux récurrents ainsi que la combinaison des deux.
* Fonctionne de façon transparente sur CPU et GPU.

#### 2.3.3. CNN (convolutional neural network) :

En [apprentissage automatique](https://fr.wikipedia.org/wiki/Apprentissage_automatique), un réseau de neurones convolutifs ou réseau de neurones à [convolution](https://fr.wikipedia.org/wiki/Produit_de_convolution) (en anglais CNN ou ConvNet pour Convolutional Neural Networks) est un type de [réseau de neurones artificiels](https://fr.wikipedia.org/wiki/R%C3%A9seau_de_neurones_artificiels) acycliques (feed-forward), dans lequel le motif de connexion entre les [neurones](https://fr.wikipedia.org/wiki/Neurone_formel) est inspiré par le [cortex visuel](https://fr.wikipedia.org/wiki/Cortex_visuel) des animaux. Les neurones de cette région du cerveau sont arrangés de sorte qu'ils correspondent à des régions qui se chevauchent lors du pavage du [champ visuel](https://fr.wikipedia.org/wiki/Champ_visuel). Leur fonctionnement est inspiré par les processus [biologiques](https://fr.wikipedia.org/wiki/Biologie), ils consistent en un empilage multicouche de [perceptrons](https://fr.wikipedia.org/wiki/Perceptron), dont le but est de prétraiter de petites quantités d'informations. Les réseaux neuronaux convolutifs ont de larges applications dans la [reconnaissance d'image et vidéo](https://fr.wikipedia.org/wiki/Vision_par_ordinateur), les systèmes de recommandation et le [traitement du langage naturel](https://fr.wikipedia.org/wiki/Traitement_automatique_du_langage_naturel).

#### 2.3.4. Pilotnet :

Figure 9: L'architecture Pilotnet

Dans le cadre de nos recherches sur la conduite autonome, NVIDIA a créé un système basé sur l'apprentissage profond, connu sous le nom de PilotNet, qui apprend à émuler le comportement des conducteurs humains et peut être déployé en tant que contrôleur de voiture autonome. PilotNet est formé à l'aide d'images routières associées aux angles de braquage générés par un humain conduisant une voiture de collecte de données. Il tire les connaissances nécessaires du domaine des données.

Cela élimine le besoin pour les ingénieurs humains d'anticiper ce qui est important dans une image et de prévoir toutes les règles nécessaires pour une conduite sûre. Les essais sur route ont démontré que PilotNet peut réussir à maintenir la voie dans une grande variété de conditions de conduite, que des marques de voie soient présentes ou non.

The PilotNet architecture is shown in Figure 1. The network consists of 9 layers, including a normalization layer, 5 convolutional layers and 3 fully connected layers.

## Conclusion

Au cours de ce chapitre, nous avons présenté les concepts de base de notre projet et les composants. Dans le chapitre suivant on va detaille notre travaille et les resultats obtenues.



# Chapitre 3 : Réalisation

## Introduction

Nous arrivons maintenant à la phase ultime. Cette dernière partie est la plus importante puisqu’elle met en réalité toute la théorie précédente.

## Collection des données

### 2.1. Procédés de collection de données

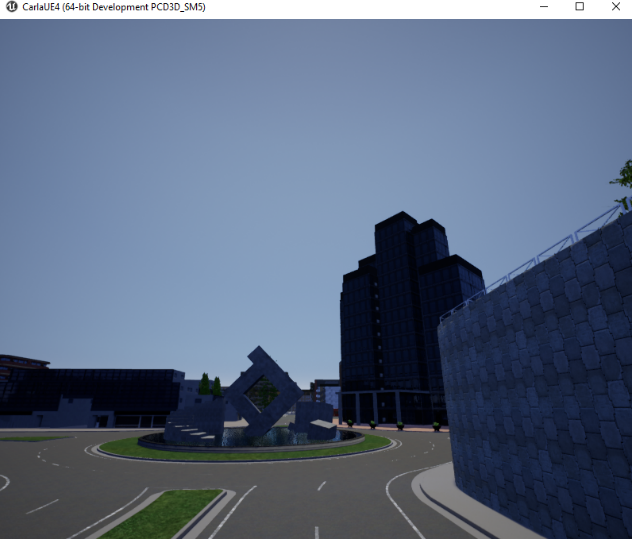
Pour collecter les données, on lance le serveur Carla à l’aide d’Anaconda Prompt qui ouvre une fenêtre de dimension (800,600) et identifie le Maps sur le Town07:

Figure 10: Le serveur Carla

Puis on exécute le fichier de configuration « manual\_controle,py » qui contient un ensemble de positions de début / fin, les conditions météorologiques et le nombre d'objets dynamiques devant apparaître à chaque épisode de collecte de données. De plus, l'utilisateur configure également un objet de [paramètres CARLA](https://carla.readthedocs.io/en/latest/cameras_and_sensors/) contenant tous les capteurs qui vont être stockés en tant que jeu de données. Donc, ce fichier sert à :

* ouvrir la fenêtre Pygame
* conduire un véhicule défini « Toyota » de façon auto pilote et prend au fur et à mesures les images chaque 2s (en moyenne) et les stocke sous format .csv et ainsi enregistre les mesures d’orientation dans un fichier .txt.

Figure 11: la fenêtre Pygame pour la conduite du véhicule et la collection des données

### 2.2. Résultats de la collection de données

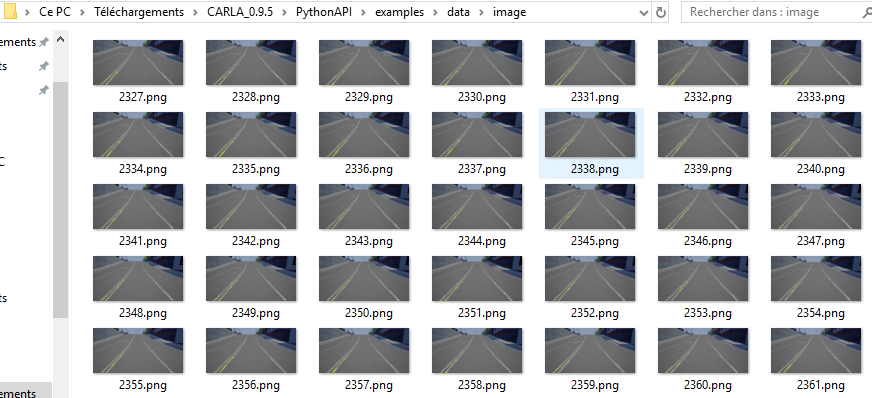
Les images sont enregistrées dans le fichier data/image de résolution 320x160x4 tel que « 4 » est 3 pour les canaux RGB et un pour un transparent.

Figure 12: Fichier où les images routières sont stockées

Les mesures d’orientation sont enregistrées dans un fichier .txt qui contient trois colonnes :

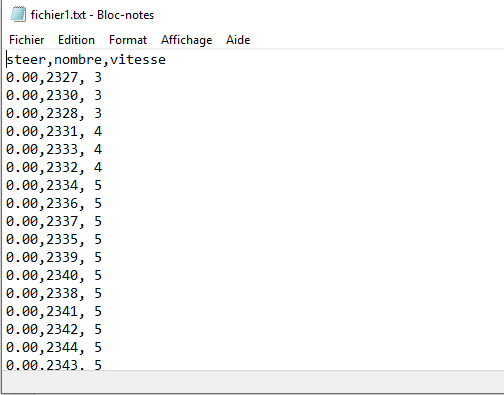
* L’angle d’orientation (steering angle) étant en degré et entre -70 et 70
* Le numéro de l’image correspondante
* La vitesse qui est une valeur entière  

Figure 13: Les mesures d'orientation

## Augmentation des données

A ce moment-là, notre base de données n’est pas suffisante pour notre réseau de neurones. En fait, l’ordre de grandeur de la taille de la base de données est 40000 images=1000scénes \* 20 seconds\*2fps(frame par second). C’est pour cela on a recouru aux procédés d’augmentation de données.

En fait, L’augmentation de données consiste à générer des données artificielles permettant d’améliorer la diversité des données d’entraînement du modèle de prédiction dans le but d’améliorer ses performances.

### 3.1. Procédés d’augmentation de données

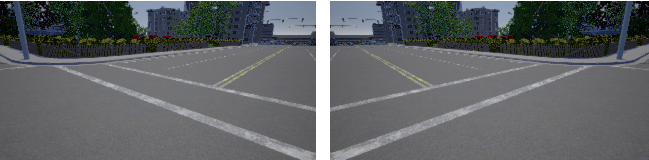
* Effet de miroir

Figure 14: Effet de miroir

* Suppression des images dupliquées : (vitesse=0)
* Utilisation de la fonction jitter() qui introduit un bruit sur certaine images

### 3.2. Comparaison de la diversification de la base de données avant et après l’augmentation des données

En faisant une comparaison entre les tailles de la base de données avant et après l’augmentation des données, on a présenté l’histogramme du pourcentage d’apparition des steering angle en fonction du steering angle :

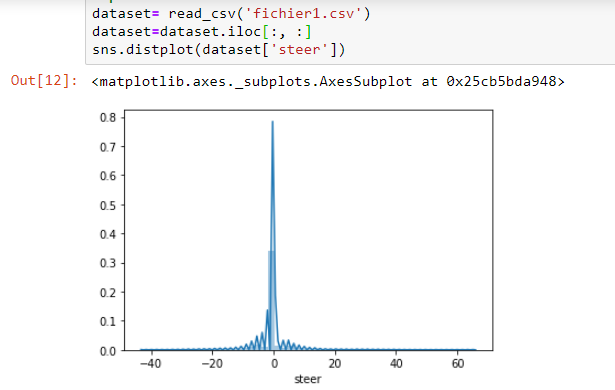
* Les données dans l’histogramme avant l’augmentation des données sont plus concentées autour de zéro donc il n’y a pas des plusieurs possibilités pour mélanger la base de données.

Figure 15: Histogramme de la base de données avant l'augmentation des données

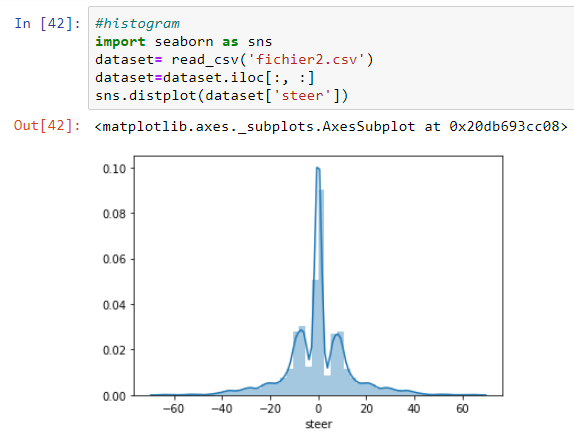
* L’histogramme après l’augmentation des données qui est plus élargis. Les valeurs sont entre -30 et 30 donc il y a plus de possibilité pour mélanger la base de données qui est plus riche et a plus de valeurs possibles.

Figure 16: Histogramme de la base de données après l'augmentation des données

### 3.3. Résultat de l’augmentation de données

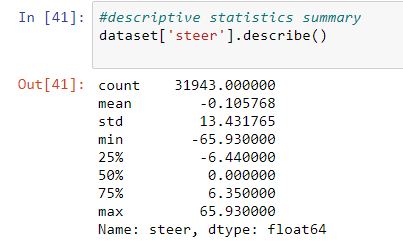
A ce stade là notre base de données contient 31943 images donc près de la valeur littéraire (40000).

Figure 17: Les statistiques de la base de données résultante

## Préparation de la base de données pour l’entrainement

Lors de l’entrainement du modèle d’apprentissage, on a procédé comme suit :

Sachant que dans la littérature, les pourcentages de décomposition de la base de données est 0.25 et 0.75, on a décomposé la base de données de telle sorte que :

* 0.8 des données pour l’entrainement dans fichier train.csv
* 0.2 des données pour la validation dans un fichier valid.csv

On a changé la révolution des images de 320x160x4 à 200\*66\*3. En fait, L’architecture PILOTNET qu’on a adoptée pour l’entrainement doit avoir en entrée des images ayant une résolution 200\*66\*3 (3 pour les canaux RGB et le transparent a été éliminé).

## Implémentation du réseau de neurones

### 5.1. Architecture de CNN : Pilotnet

NVIDIA a créé un système basé sur l'[apprentissage profond](https://developer.nvidia.com/deep-learning) , connu sous le nom de PilotNet, qui apprend à émuler le comportement des conducteurs humains et peut être déployé en tant que contrôleur de voiture autonome. PilotNet est formé à l'aide d'images routières ayant une résolution 200\*66\*3 associées aux angles de déviation générés par un le véhicule conduit de façon autopilote. Il tire les connaissances nécessaires du domaine des données.

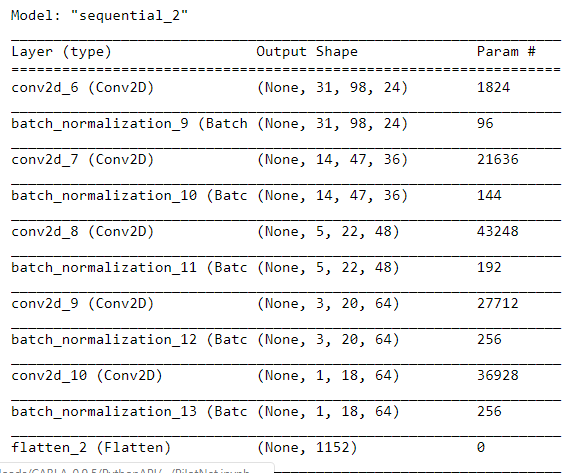


Figure 18: Architecture Pilotnet

Arrivant maintenant à l’étape de prédiction et d’évaluation de la performance du modèle conçu.

### 5.2. Résultat de la prédiction

Les valeurs des angles de déviation prédits sont presque identiques à celles réelles. En fait :

Figure 19: Les valeurs des angles de déviation prédites

### 5.3. Evaluation du modèle

En fait, évaluer la qualité de la prédiction, on a recouru à mesurer l’erreur quadratique moyenne, Mean Squared Error en anglais (MSE) qui intitulé dans notre cas « val\_loss>> dont la valeur est relativement bien car il est très faible (de l’ordre de 0.037).

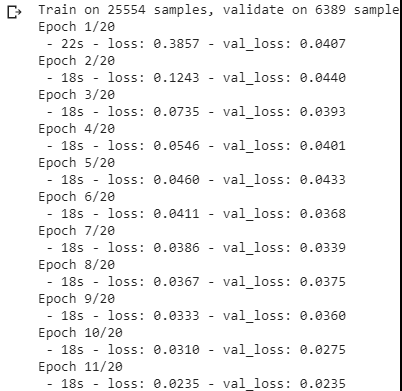


Figure 20: Evaluation des mesures prédites

## Implémentation du modèle sur Carla

A ce stade-là, on a pu concevoir un modèle relativement parfait qui à partir des images routières associées par des mesures d’angle d’orientation donne les actions à appliquer au volant qui sont les angles préditset dans cette phase on a

implémenter le modèle sur le simulateur Carla.

## Conclusion

Au cours de ce dernier chapitre, nous avons implémenté notre solution qui se déroule sur trois étapes telles qu’une étape technique pour la collection de donnée, une étape sur le développement de model et la dernière pour le tester sur Carla.

# Conclusion

Ce présent rapport a été rédigé dans le cadre du projet tuteuré n°4 intitulé par «End-to-End Learning pour la conduite autonome : de la simulation au monde réel ».

Notre objectif était de concevoir et de développer un réseau de neurones CNN capable de donner, à partir des images routières et des mesures d’orientation, les actions à appliquer au volant.

Ce travail a suivi plusieurs étapes indispensables pour la phase de conception du modèle et son implémention sur le simulateur Carla. La première étape était de collecter la base de données et les mesures d’orientation. La deuxième étape était l’augmentation des données pour qu’elles soient riches et rendent le modèle plus performent. La troisième étape était l’implémentation de l’architecture Pilotnet de NVIDIA pour l’entrainement des données après son décomposition en deux partie dont une pour l’entrainement et l’autre pour la validation et l’évaluation du modèle. La dernière étape était l’implémentation du modèle sur Carla pour qu’il soit prêt pour son implémentation sur un robot et ainsi on arrive à assurer que le modèle est parfait ou non.

Malgré tous les changements qui nous sont arrivés durant le travail, et malgré les obstacles techniques que nous avons rencontrés, nous avons pu atteindre tous les objectifs définis au préalable. Pour conclure, notre projet était totalement bénéfique etreste toujours susceptible d’être amélioré et évolué, notre modeste travail n’est que le début d’un long processus.

# Bibliographie

[1] <https://mc.ai/convolutional-neural-network-to-steer-a-vehicle-inside-a-game/>

[2] <https://pdfs.semanticscholar.org/cfe0/f2abfc97326cfd8b519e714a3e47595006da.pdf>

[3] <https://fr.wikipedia.org/wiki/Python_(langage)>

[4] <https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)>

[5] <https://fr.wikipedia.org/wiki/Patron_de_conception>

[6] <https://medium.com/asap-report/introduction-to-the-carla-simulator-training-a-neural-network-to-control-a-car-part-1-e1c2c9a056a5>

[7] <https://www.udemy.com/course/unrealcourse/>