Transactions on Machine Learning and Artificial Intelligence - Vol. 9, No. 5
Publication Date: October, 25, 2021
DOI:10.14738/tmlai.95.10795.
Nabil, A., & Kassem, A. (2021). Self-Driving Vehicles Using End to End Deep Imitation Learning. Transactions on Machine Learning and Artificial Intelligence, 9(5). 33-43.

Self-Driving Vehicles Using End to End Deep Imitation Learning

Ashraf Nabil
Aerospace Engineering, Cairo University, Cairo, Egypt

Ayman Kassem
Professor in Aerospace Engineering at Cairo University

ABSTRACT

Autonomous Driving is one of the difficult problems faced the automotive applications. Nowadays, it is restricted due to the presence of some laws that prevent cars from being fully autonomous for the fear of accidents occurrence. Researchers try to improve the accuracy and safety of their models with the aim of having a strong push against these restricted Laws. Autonomous driving is a sought-after solution which isn't easily solved by classical approaches. Deep Learning is considered a strong Artificial Intelligence paradigm which can teach machines how to behave in difficult situations. It proved its success in many differ domains, but it needs sometimes to prove itself in the automotive applications. The presented work will use the end-to-end deep machine learning field in order to reach to our goal of having Full Autonomous Driving Vehicle that can behave correctly in different scenarios. CARLA simulator will be used to initially train and test the deep neural networks. The resulting DNN models are further trained using one of the most popular real datasets of automotive that includes camera images with the corresponding driver's control action [1]. This data mixing approach shows improvements in the DNN models prediction.

Keywords: Autonomous Vehicles; Self-driving; Machine learning; Deep learning; Convolution Neural Networks; Control.

INTRODUCTION

According to the World Health Organization (WHO), approximately 1.35 million people die in the world every year due to vehicle accidents and between 20 to 50 million more people are injured/disabled. Many of these accidents are preventable, and an alarming number of them are a result of distracted driving [2].

From this perspective the automotive companies and many research centers and universities started looking for solutions for such drastic issue. An autonomous car is a vehicle that can guide itself without human interaction.

Autonomous vehicles depend on sensors, actuators, complex control algorithms, machine learning systems, and powerful processors to run software. Autonomous vehicles contain 4 main blocks from system and software point of view as shown in figure 1. First is the data sensors level followed by the perception layer then the motion planning layer and finally the control layer. Each one of those will be discussed in detail in the next section.

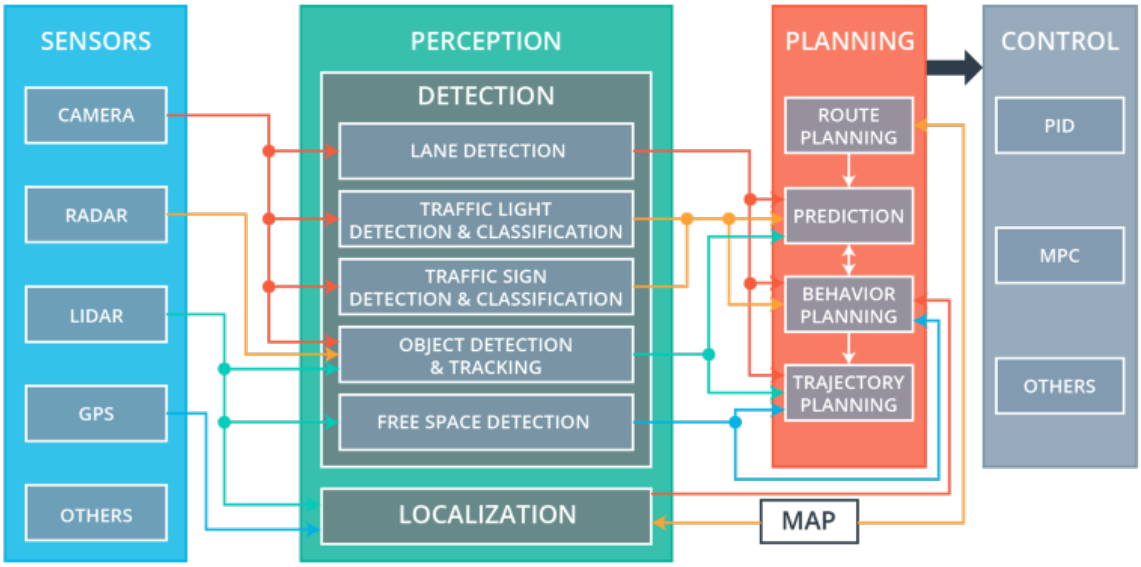


Figure 1 Self-Driving Vehicles Pipeline [3]

LITERATURE SOLUTIONS

Making vehicle autonomous mostly depends on one of 2 approaches:

1. The first is a modular pipeline that relies on dedicated subsystems for visual perception, planning, and control. This architecture is in line with most existing autonomous driving systems [4].
2. The second approach is based on a deep network trained end-to-end via imitation learning. This approach represents a long line of investigation that has recently attracted renewed interest [1, 5].

Modular Pipeline Approach

First method is a modular pipeline that decomposes the driving task among the following subsystems:

Sensors

It is the first layer in both approaches which describes how many sensors are mounted on the vehicle and what are they. Since the autonomous driving is a sensitive field and may lead to many accidents, usually vehicles must be equipped with different sensors. Each sensor has its own pros and cons. Some of the most popular sensors are:

- Radar: it is very accurate in getting the object's velocities (based on doppler effect).
- Camera: it is very precise in classifying the object's types.
- Lidar (laser scanner): it is very accurate in measuring the object's position even for a long-range object.

Perception

Frequently provide and update a map of the surroundings based on a different sensor mounted in different parts of the vehicle like radars (monitor the position and velocity of nearby vehicles), cameras (detect traffic lights, read road signs, track other vehicles and pedestrians),

LIDARs (send pulses of light to measure distances, detect road edges, and identify lane markings).

Sophisticated algorithms use all these sensors data to be fused together, filtered and tracked over time. All the previous sensor's readings are fused together in order to have accurate, robust, and sufficient information for the surrounded environment.

Motion Planning

At a high level, autonomous driving mission is to navigate from point A to point B. Point A could be a point on a map or vehicle current position, and point B is the target destination. Motion planning is the task of finding the most efficient path in the road network in terms of travel time or distance, then navigating the ego-vehicle to its destination in a safe and comfortable manner following the traffic rules such as roads structures, obstacles, and other agents on the road.

Control

Takes the trajectory from the planning module and sends instructions to the vehicle's actuators, which control acceleration, braking, and steering using any kind of control algorithms like Proportional Integral Deferential controller (PID controller) or even one of more advanced techniques like Model Predictive Control (MPC).

Deep Learning Approach

Second method is conditional imitation learning, a form of imitation learning that uses high level commands based on inputs from sensors.

- This method utilizes a dataset of driving traces recorded by human drivers in different driving scenarios. The dataset consists of tuples D = [observation, command].
- The commands are provided by drivers during data collection and indicate their intentions to the current scenario.
- The observations are data from current sensors in vehicle, in our case they are images from a forward-facing camera.
- Hours of driving data for training should be recorded to train the network how to learn in different scenarios (straight lines, curvy lines, intersections, ...etc.) and different conditions (sunny, rainy, cloudy, …etc.).

As mentioned earlier the autonomous driving problem is very complex and one of the complexities of this problem on how to use an environment that is human safe and complex enough to emulate the real life. Simulation can be very helpful in autonomous driving research. It is essential since some scenarios are too dangerous to be staged in the physical world (e.g., a passenger crossing without paying attention to the road ahead of the car).

Simulations are also used to be the first platform used by the machine learning to train and validate the models. More recently, different simulators have been used to evaluate new approaches to autonomous driving. Not only Custom simulation setups are commonly used to train and validate autonomous systems but also some commercial games have been used to in the same purpose due to their performance.

FULL SELF-DRIVING VEHICLE USING IMITATION LEARNING

Introduction

Development of full autonomous vehicles from the beginning will be very challenging and complex. It will need a huge data and effort. So, the problem was divided in to sub problems trying to solve each one separately (divide and conquer method). Several researched before discussed how we could use control the steering angle of the vehicle autonomously based on DNN using either simulator environments (like Udacity's simulator) [6] or even in the actual vehicle (like Nvidia's steering model) [7, 8].

But we will discuss how to extend the steering angle approach to the rest of full autonomous driving control functionalities (like gas pedal, braking and steering). For the sake of illustration and learning,

First Stage (Simulator based)

CARLA simulator is used to collect data, train the DNN models and finally validate them [5]. Figure 2 shows some of the features from the environment in the simulator. Several vehicles, pedestrians, traffic lights, lanes and road boundaries are available to simulate the real environment as possible.



Figure 2 Image from CARLA simulator: Acceleration DNN model behaves correctly by stopping the vehicle due to the crossing pedestrian

DATA COLLECTION, MANIPULATION AND TRAINING

Data collection

CARLA simulator has different modes for operating:

- Manual driving: the drive can drive the vehicle by hand (i.e., by keyboard, joystick, ... etc.) like normal computer games.

- Autopilot: the vehicle will be driven automatically from the simulator itself. This mode is very helpful because it will eliminate any human errors while recording the data and is human independent. So, the simulator can be left to run for hours to collect data without any human interaction or supervision.

Data Manipulation

For the data acquisition, autopilot mode has been enabled and while driving images from the camera is captured and stored with their corresponding driver inputs (like steering, acceleration, braking, …etc.). One of the most important features that the steering DNN model had to learn is the curves in the road.

There might be more curves in the road in one direction than the other direction. If the data collect is used only, the model would learn to steer on one of the directions better that the other. But by using data augmentation any using mirroring images technique using a horizontal flip, and inverting the steering angles (i.e. multiplying the values by -1). This will provide a balanced datasets where the steering DNN model would learn how to steer in both direction with the same efficiency [9].

In order to improve the robustness of the learned policy, we injected noise into the steering angles during training data collection. In other words, a perturbation (noise) is added to the steering angle provided by the driver. This simulates gradual drift from the desired trajectory, like what might happen with a poorly trained controller.

In order to reduce over fitting of the DNN model, an extra extensive data augmentation by adding Gaussian blur and additive Gaussian noise. All the previous mentioned points emulate what might happen in the real world due to sensor noises or environment changes (sunny, rainy, cloudy, ...etc.) [9].

Cropping was used in order to remove the parts of the image that do not have valuable information for autonomous driving, as to remove the sky threes and hills on the top of image and the part of hood of the vehicle on the image bottom. The raw image from the camera is 600x800 pixels while after cropping the redundant pixels it reached 200x300 pixels at the end which has more than 50% reduction.

The images from dataset were normalized by dividing each pixel of the image by 255, which is the maximum value of an image pixel. Once the image was normalized to a range between 0 and 1, mean centering of the data was applied by subtracting 0.5 per each pixel [1,9].

Training Data

There are a lot of popular DNN models that are available each one has its own purpose and application. One of the most fitting DNN models that is published and used recently by Nvidia is Pilot-Net (NVIDIA CNN) figure 3. It is designed by researchers from NVIDIA Corporation. It is the deep neural network for end-to-end autonomous driving, created with the motivation to improve the DARPA Autonomous Vehicle (DAVE) system for autonomous driving. Pilot-Net has a normalization layer, followed with five convolutional layers, followed by four fully connected layers.

Since our goal to let the vehicle to drive autonomously, we need three control action at the end steering angle, braking command and acceleration pedal command. So, a two separate DNN models are implemented since braking and acceleration are correlated. The correlation appeared also when the different layers of all the DNN models where visualized (to show how each layer could learn from the data). So as a kind of improvement to the whole process, the braking DNN model was removed totally and the system was working with just 2 DNN models, one for lateral control which is the steering and the other is longitudinal control that is used for both speed and braking (if needed).

Such enhancement has a big influence on the system overall as it reduces the run-time a lot since almost one-third of the system time has been removed. It also reduces the training time if any further training needed in the future. At the end, the performance hasn't been affected after removing the third DNN model from the system.

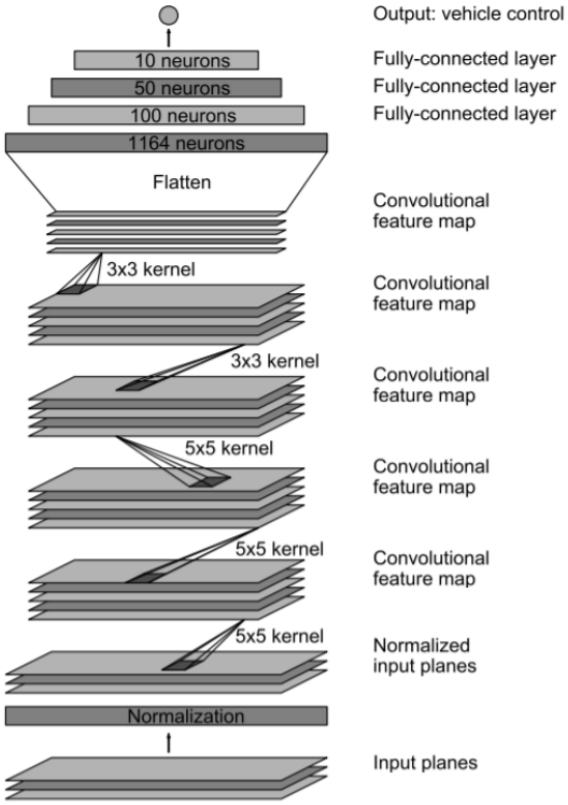


Figure 3 Nvidia (Pilot Net) DNN model architecture [10]

Second Stage (Real dataset based)

The purpose of this section is to generalize the DNN models performance not only in simulation environment like CARLA, but also on real dataset from the actual road scenarios. There are several datasets that are collected by different entities over the last couple of years as a chance to enhance and accelerate the autonomous driving research.

After some research in different online available data sets [11, 12], Comma-ai dataset is the one at the end to validate our DNN models. Comma-ai dataset used for Artificial Intelligence self-driving car, it is based on an agent that learns to clone driver behaviors and plans maneuvers

by simulating future events in the road. The dataset consists of 10 videos clips of variable size recorded at 20 Hz with a camera mounted on the windshield of an Acura ILX 2016 as shown in figure 4. In parallel to the videos, some measurements were recorded such as car's speed, acceleration, steering angle, GPS coordinates, gyroscope angles [12].

The DNN model had to be tuned a little bit since the images from the Comma-ai dataset has size of 320x160 pixels. So, the input layer in the DNN models had to be adapted to this new input images. Since the dataset consists of 10 videos, 8 of them were used for training while 2 of them were used for validation at the end.



Figure 4 One of the images from Comma-ai dataset

IMPROVEMENTS AND OPTIMIZATIONS

Introduction

Usually, the problem in DNN models is the size of the models themselves. Of course, as the model size increase, usually the DNN model learn more features which will have much better results. But it also has its drawbacks. There are benefits for reducing the DNN model in terms of layers or terms of neurons in the same layer such as:

- Reduction in the DNN model training time which opens the door to get more data for training.
- Reduction in the DNN model inference time which is very important in automotive applications. Capturing an image form the image and outputting the correct output should happens as fast as possible. It isn't acceptable for example to take around 5 seconds in one cycle (capture image, process it, pass it to the DNN model and expect the output at the end).
- Reduction in the DNN model actual size itself save some memory.

IMPROVEMENTS

Minimize image input size

The default image size from CARLA is 600 pixels height and 800 pixels width. Of course, this whole image contains unnecessary parts (for example like the sky, the buildings, ...etc.). The problem isn't only that there are huge number of pixels that we don't benefit from, but they also increase the number neurons in the input layer (800 pixels * 600 pixels * 3 layers for RGB = 1,440,000 pixels).

One of the easiest ways is to crop the image size and remove some of the redundant pixels. Of course, it should be cropped wisely as it might remove important features from the road (like traffic lights) that might affect the performance. After trying different cropping combinations, the best one that kept the performance as it is and minimized the number of pixels was (200 pixels * 300 pixels * 3 layers = 180,000). Of course, this reduces the input size by a factor of 8 which is a pretty good start.

MINIMIZE DNN MODEL PARAMETERS

As discussed before that Nvidia's neural network model for steering autonomously is used. The model consists of the different layers as shown in figure 3.

Based on the input image size (200x300x3), The total number of parameters in the DNN model is then 3,593,019 parameters, which is a huge number of parameters that takes some time while training the network and while inference time. After further investigation, the most layer with number of parameters is the first fully connected layer with 100 layers, it only contains about 3,456,100 parameters.

So, after several tunning in the number of neurons in the fully connected layers, the DNN model has been modified where still the normalization layer, 5 convolution layers, 1 flatten layer, 1 dropout layer are still the same as the Pilot-Net model. The only difference in the layer 3 fully connected layers where instead of having 100 neurons layer, 50 neurons later and finally 10 neurons layer, the system has been modified to have 70 neurons layer, 30 neurons layer and finally 10 neurons layer. Such kind of reduction in the number of neurons decrease the overall number of parameters in the DNN model to 2,553,239. This is around 30% decrease in the number of parameters in the DNN model while keeping the same performance.

PROVIDE MEASUREMENTS AS AN INPUT

One other improvement to the system is the introduction of the vehicle's input (speed specifically) as an extra input to the DNN model beside the RGB image. Such improvement was needed because the DNN model before wasn't considering the ego vehicle speed. For example, if the distance between the ego vehicle and another vehicle in front of it is around 10 meters. Then, the vehicle in front stopped suddenly, the ego vehicle shall stop. The problem here that the model won't take in consideration the ego vehicle's speed, and this is a problem since the braking command that is needed to stop the ego vehicle if is 50km/h not like the braking command needed if the ego vehicle's speed is 120km/h.

EXPERIMENTS AND ANALYSIS

The experiments and results are performed as follows:

- Use some real dataset from real road to test the DNNs performance.
- Visualize the DNN layers features to check how each layer learnt from the data.
- Visualize the error in DNN Models

DNN MODEL WITH REAL DATASET

Figures 5 and 6 shows how the two DNN models for steering angle and acceleration results after training using the Comma-ai dataset. The green curve shows the results from the DNN models while the blue curves show the output from the labels of the dataset which act as the ground-truth in our case. The results shows both how the steering in almost the same from the

DNN models output and the dataset labels. As the speed of the vehicle which is represented in the length of the curve shown in the figures. So, this shows that the two DNN models not only shows a very good results in the CARLA simulator but also it shows a promising result using one of the popular real datasets for automotive.

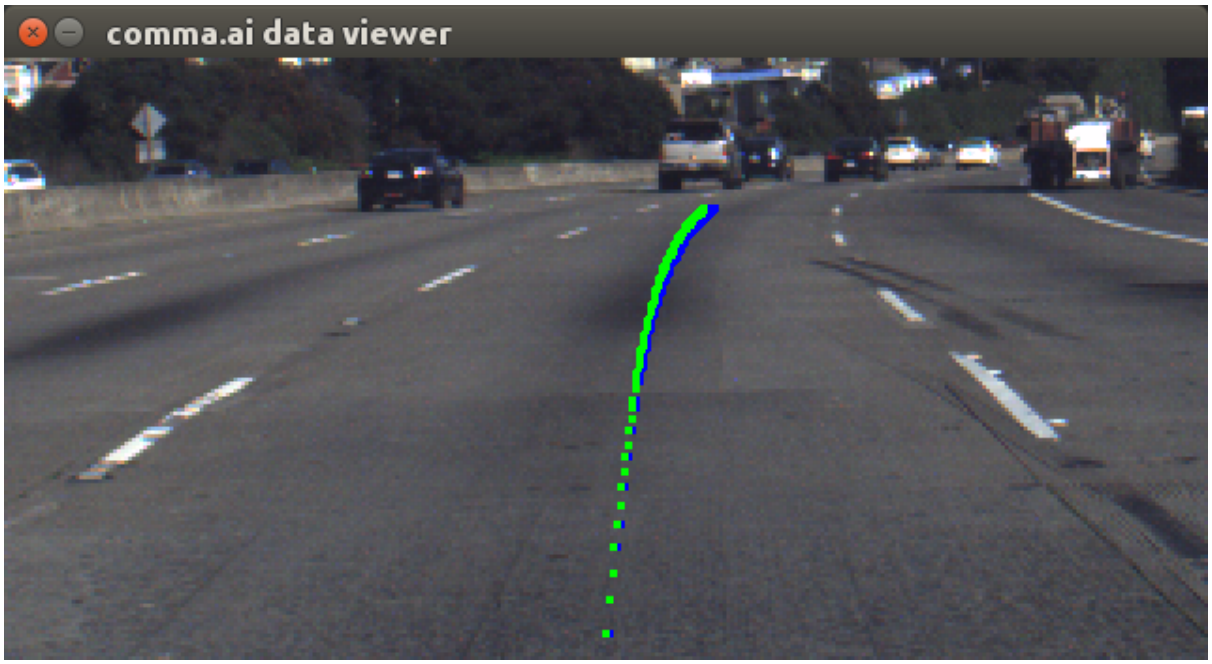


Figure 5 Results of DNN models using Comma-ai dataset in curved road

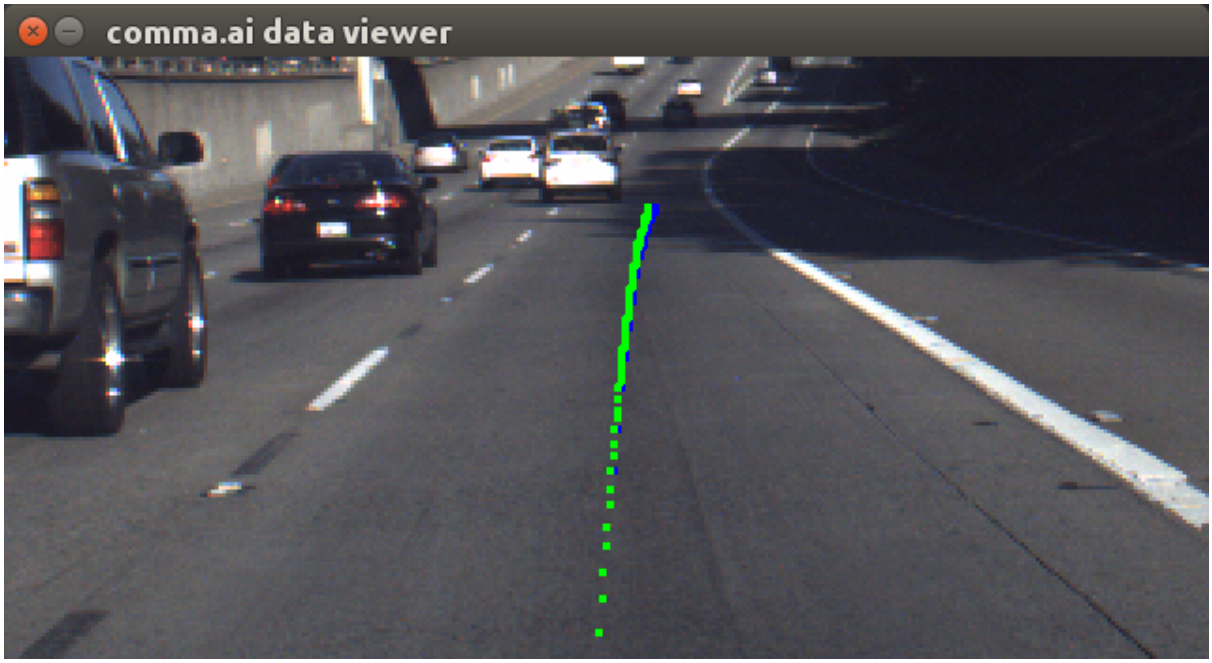


Figure 6 Results of DNN models using Comma-ai dataset in almost straight road

ERROR IN DNN MODELS

Figures 7 and 8 shows the error between the output of the two DNN models we have trained (acceleration and steering models) and the output from CARLA's autopilot which acts as reference in our case.

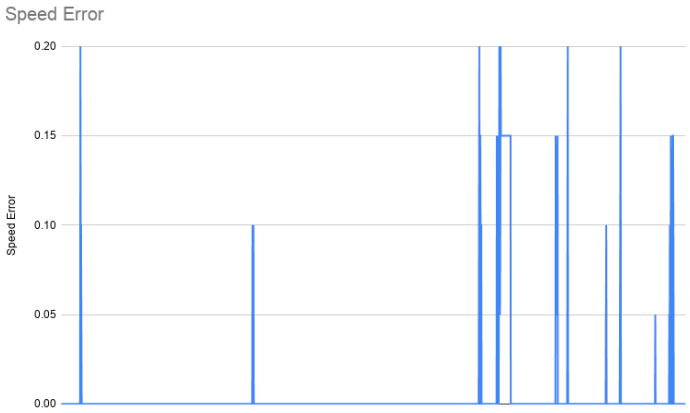


Figure 7 Error between CARLA autopilot and the DNN output for speed values



Figure 8 Error between CARLA autopilot and the DNN output for steering angle

The autopilot of CARLA was the one we used earlier to log data to train the DNN models assuming it is the perfect driver model. The plots shows that the error between them is almost unnoticeable in the steering angles values which is in degrees and is small in the speed values which varies from 0 to 1 which means no throttle up to full throttle respectively.

CONCLUSION

The two DNN models (acceleration and steering) tested in the simulator. Different scenarios were used in order to make sure that both models can behave well. The two DNN models learned how to act over:

- Other vehicles (don't crash in other vehicles)
- Pedestrians (don't crash in passing pedestrians)

- Traffic lights (detect traffic lights and make decisions either keep moving or stop)
- Lanes (detect lanes and keep moving in same lane)

Finally, using imitation learning as a deep learning solution in autonomous driving problem is a popular research solution nowadays and there is a huge break through these days as the computation capabilities is huge now (compared to the past 10 years). So, it means we can let our network learn a lot of information in a very short time.

References

Mariusz Bojarski et al., *End to End Learning for Self-Driving Cars*. In: ArXiv (2016).

World Health Organization (WHO). *Road traffic injuries*. June 2021. url: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.

Udacity Inc. Self-driving Car Engineer. url: https://www.udacity.com/course/self-driving-car-engineer-nanodegree--nd0013.

Philip E. Ross. *Robot, you can drive my car*. In: IEEE Spectrum (2014), pp. 60–90

Alexey Dosovitskiy et al. *CARLA: An Open Urban Driving Simulator*. In: (2017), pp. 1–16.

Udacity. Inc. Self-Driving Car Simulator. Nov. 2018. url: https://github. com/udacity/self-driving-car-sim.

Mariusz Bojarski et al. *Explaining how a deep neural network trained with end-to-end learning steers a car*. In: arXiv preprint arXiv:1704.07911 (2017).

Shuyang Du, Haoli Guo, and Andrew Simpson. *Self-Driving Car Steering Angle Prediction Based on Image Recognition*. In: (Dec. 2019).

Jelena Kocić, Nenad Jovičić, and Vujo Drndarević. *An End-to-End Deep Neural Network for Autonomous Driving Designed for Embedded Automotive Platforms*. In: Sensors (2019).

Navarro, Anthony & Joerdening, Jendrik & Khalil, Rana & Brown, Aaron & Asher, Zachary. (2018). Development of an Autonomous Vehicle Control Strategy Using a Single Camera and Deep Neural Networks. 10.4271/2018-01-0035.

Jannik Fritsch, Tobias Kühnl, and Andreas Geiger. *A new performance measure and evaluation benchmark for road detection algorithms*. In: (2013), pp. 1693–1700.

Eder Santana and George Hotz. *Learning a Driving Simulator*. In: CoRR abs/1608.01230 (2016). arXiv: 1608.01230. url: http://arxiv.org/abs/ 1608.01230.