DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Masterpraktikum Final Report

Simulation-Based Autonomous Driving in Crowded City

Tarik Isildar

Contents

1 Motivation and Introduction

The pursuit of autonomous driving has been a hallmark of technological innovation in recent years, captivating imaginations and revolutionizing the transportation landscape. As vehicles equipped with sophisticated sensors and artificial intelligence algorithms inch closer to true autonomy, we find ourselves at a critical juncture. Amidst the fervor surrounding autonomous vehicles, a profound shift is taking place – one that acknowledges the central role of data and the emergence of imitation learning as a catalyst for unlocking their full potential.

1.1 Introduction

Conventional ways of autonomous driving consist of treating every aspect of driving sense differently and make them contribute to the final decision making. Methods in recent years mostly break-off the problem to perception, planning [1] [2] and decision making, and try to make a better way for each of the components.

If we look at the problem in a higher-level perspective. Driving is something that humans do for more than hundred years. Rule-based methods will surely keep on improving and at some point will pass the human judgement, but until then We better of rely on human senses and reflexes.

Imitation learning, a sub-field of machine learning, thrives on data. This approach empowers machines to learn from expert demonstrations, thereby imitating their actions and decision-making processes. Its transformative potential lies in its ability to harness the knowledge embedded in data, not through rule-based programming, but by observing and emulating the expertise of humans or expert systems.

In the context of autonomous driving, imitation learning bridges the gap between data collection and intelligent decision-making. It addresses the challenges of data acquisition by automating the process through the assimilation of human or expert behavior. This not only accelerates data collection but also ensures the data's alignment with the intricate demands of autonomous systems. In essence, imitation learning acts as the conduit through which data becomes actionable intelligence.

In this project, we embark on a journey that commences with the state of autonomous driving and traverses the path towards imitation learning, highlighting the pivotal role of data collection throughout. We explore the challenges faced by autonomous driving systems in accumulating diverse, high-quality data and delve into how imitation learning can streamline this process. Additionally, we investigate the ways in which imitation learning enhances decision-making within autonomous systems, bringing us closer to the realization of safe, efficient, and adaptive autonomous vehicles.

1.2 State of The Current Research

Like mentioned before, there are two main ways to tackle the same problem. Rule based methods came a long way during the last 10 years, using various methods and sensory information to divide the problem into sub-problems and do lane-following, obstacle detection [3], path planning [1] [2]. Also detection and following of the traffic rules.

On the other hand, imitation-learning models can also have additional sensory information like LIDARs [4] as well as the real-time RGB images.

2 Methodology

2.1 Initial Setting

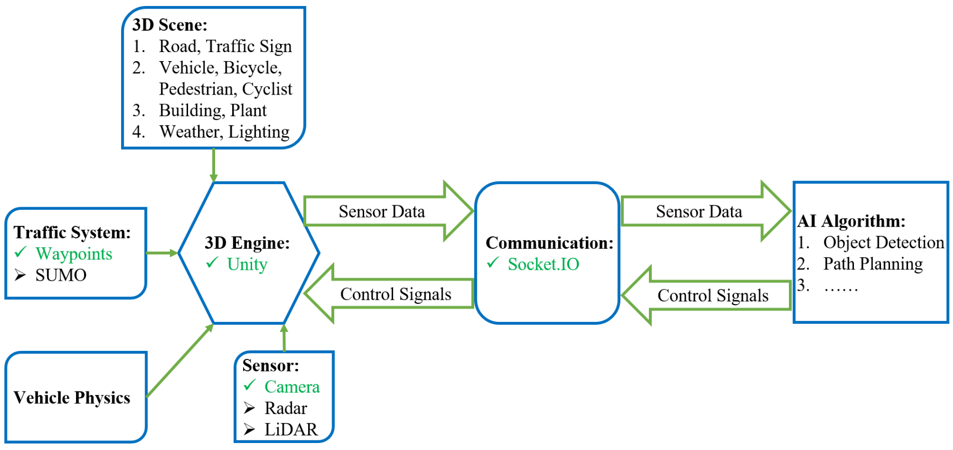


Figure 1. The Framework of Our Autonomous Driving Simulator.

Our setting is running on a Unity3D simulator that is connected through Socket.io framework to send the sensory information to the driving script. On data preperation time, the simulation saves the images and the wanted inputs. And inference time the same images sent through the socket and commands for throttle, steering angle and brake have taken as response.

2.2 Problems

It's normally better to use all the possible inputs and outputs and hopefully it'll work like a magic box since it's a transfer learning. But this setting will have couple of core problems:

- Data-size: Since the driving in a city most of the time just waiting or going straight, the data will be bloated real fast. It gets hard to train and not very successful
- Bias: Since the driving is heavily relies on the data, there will be some bias towards the driver groups' driving habits.
- Edge-cases: On a normal run, we hardly ever run-by on a case that would be considered as an edge case. That makes the model fragile to even the slightest anomalies.

2.3 Solutions

2.3.1 Input Differentiation

Instead on this method, We focused on breaking off the imitation learning to have a better data-set for each input. Each input is treated differently:

- Throttle: Fully rule-based. It's aim is to stay on a specific speed. It's decided directly from the combination of current speed, maximum speed and the steering angle.
- Steering angle: Trained an end-to-end DNN with its own data-set and parameters.
- Brake: Trained an end-to-end DNN with its own data-set and parameters.

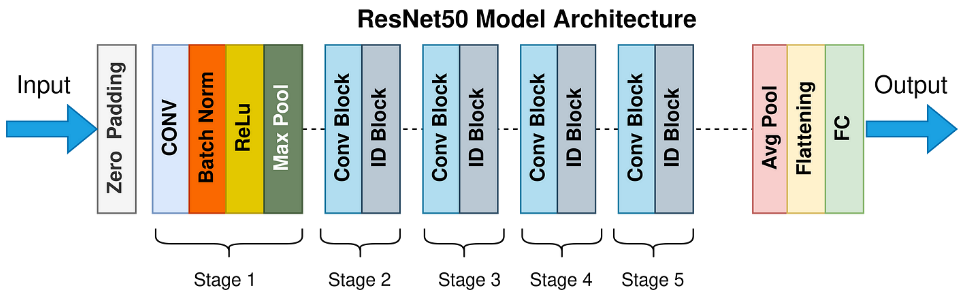
So in the end, the steering angle and the Brake has their own trained models each of them are optimized specifically for that cause. The data-set should cover every case but some samples will be more important than the others. But we saw during the development that the important samples for the steering angle and the brakes are completely different.

2.3.2 Performance Considerations

We are aware that this method won't scale nicely for the real-time applications as running a model per input per frame would be very costly. But instead of doing one model per input, the inputs can be grouped to have better representations. And for solving the performance issue we went for running only one model each frame. Since our training models now can be lighter weight and we can get away with less parameters, we got more frames per second and the total decision time is not effected drastically.

2.3.3 Network and Training

For the both steering angle and brake models I used the ResNet-50[5] architecture with a pre-trained backbone from PyTorch.



The ResNet architecture helped me to have satisfactory results during and after training. In an ideal scenario more network architecture should be trained and should compare the result. But the priority was iterating fast with a reasonable performance need and ResNet-50 offers that perfectly. The model has around 23 million parameters and the script together with the visualization can run around 20 fps on the GPU RTX 3060 Ti.

From the training mode in the simulator, gathered recording footage from various scenarios. Had around 20 thousand images that are used as base data and ran a multiple iterations of training.

The result wasn't near what was expected at this point.

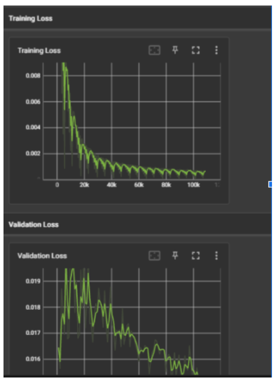


Figure 2.1: Training Graphs

2.4 Improvements

2.4.1 Data Fine-Tuning

After we had the first models, we observed some cases the vehicle was always failing. It had difficulties with keeping the lane and couldn't correct itself at all when some things went bad. We could still give manual input to correct those behaviors and keep the driving alive. At that moment we realized those corrections were the exact situations that the model needed in the training phase. So we made an extension to the driving script to record every manual correction sequence made during the inference time. From that moment we used the last model as pre-trained state and did 5 iterations with 2500 new "corrected" images for less epochs. With this system we also intentionally took the vehicle off-road in different scenarios and recorded the corrections. In the end we had a robust model that is capable to cover every edge-case.

3 Results

3.1 Failed Trials

3.1.1 Semantic Segmentation

The idea at this point is adding extra information to the training to make the imitation learning more aware of the scene. We only have the RGB images as input from the simulation, but we can always extract more information from it. The first idea was using Semantic Segmentation.

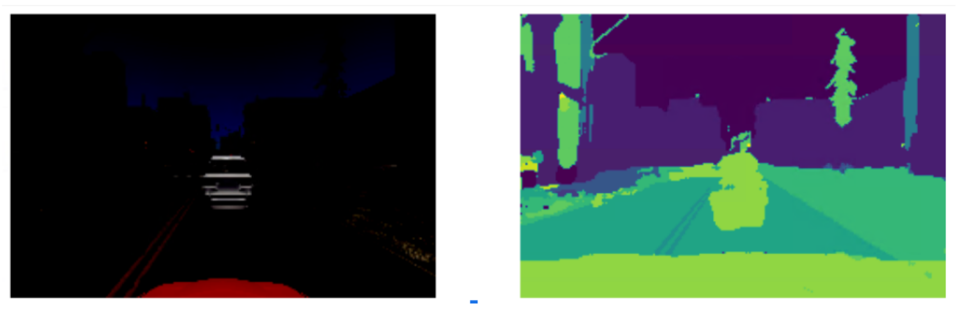


Figure 3.1: Semantic Segmentation

Using UNet[6] architecture we tried to mask out the unnecessary information and classify the road, vehicles, lights etc. So in the imitation learning phase the decision for the model would be less ambiguous. We did initial trials with this, it's promising and has a lot of room for improvement but this requires running two models in the same frame to decide on an input. Which slowed down the decision making drastically and in the end the losses outweigh the gains.

3.1.2 Object Detection

Another idea was using the YOLOv5[7] algorithm for object recognition and have a fail-safe for braking. It's a simple rule-based stopping mechanism where it orders the vehicle to stop when there are other objects too close to the driving vehicle. This was also discarded after realizing some core-problems about the difficulty of deciding which cases are actually need a failsafe and which cases are false-positives.

3.2 End Result

With the great help from the data-fine tuning method, the model could run without a problem for over 4-5 minutes on average. It's using the speed limit as 30 km/h and still can confuse when to stop on some cases. The biggest missing part for this toy setting is the Traffic-lights. They were not considered during the data capturing nor the fine tuning time. The turns are rather smooth, and the following distance to the other cars are satisfactory.

4 Discussion

Every single part of this project requires more research and more improvements if it'll need to be considered in a real-life scenario. This small case-study can only shows what is promising and which part of the problem has realistic solutions. I experienced in the first-hand how problemous can the rule-based methods can get with smaller workforce/experience whereas imitation learning is very easy to make things work but very difficult to improve later on.

5 Outlook

The results that I had were with a total of 32,000 images for steering and around 10,000 for braking which I consider very small considering the scope of the problem. It needs much bigger data-sets to have a better generalization, also even different drivers to collect those data to have less drivers' bias in the end. The results are fully using end-to-end trained models. I don't believe it would be enough for a real-world scenario. It needs extra systems to support the main model. Especially traffic rules mustn't be relied on the user-data. A better simulation environment is as crucial as the driving part. It needs improvements performance-wise. Also other sensory information can be beneficial.

Bibliography

[1] e. a. Y. Hu J. Yang. “Planning-oriented autonomous driving”. In: *CVPR* (2023).

[2] A. S. S. Casas and R. Urtasun. “Mp3: A unified model to map, perceive, predict and plan”. In: *CVPR* (2021).

[3] P. Amaradi, N. Sriramoju, L. Dang, G. S. Tewolde, and J. Kwon. “Lane following and obstacle detection techniques in autonomous driving vehicles”. In: *2016 IEEE International Conference on Electro Information Technology (EIT)*. 2016, pp. 0674–0679. DOI: `10.1109/EIT.2016.7535320`.

[4] J. Chen, B. Yuan, and M. Tomizuka. “Deep Imitation Learning for Autonomous Driving in Generic Urban Scenarios with Enhanced Safety”. In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 2884–2890. DOI: `10.1109/IROS40897.2019.8968225`.

[5] K. He, X. Zhang, S. Ren, and J. Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: `1512.03385 [cs.CV]`.

[6] O. Ronneberger, P. Fischer, and T. Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: `1505.04597 [cs.CV]`.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: `1506.02640 [cs.CV]`.