End-to-end Imitation Learning for Autonomous Vehicle Steering on a Single Camera Stream

Thomas van Orden and Arnoud Visser

Intelligent Robotics Lab, University of Amsterdam, The Netherlands*

Abstract. Vehicles can follow roads based on a forward looking camera, but this has to be done reliably in all circumstances. In daily traffic they can encounter many unforeseen situations. Training for those situations in simulations should prepare them for such encounters, but this requires simulated worlds with enough complexity. In this paper we have trained a vehicle to follow the roads in one of the most complex environments available in the simulation environment Carla: the map Town 3. Still, during training the vehicle encounters a disproportionate number of simple straight roads, so care has to be taken on the balance in the training set. End-to-end learning for autonomous vehicles have been shown before, but not for the complex worlds used in this paper. After the training the vehicle can follow the road reliably on the training map, a behavior which can be transferred to another map with circumstances it has not seen before. The learned behavior has been validated on a map which is just released with the latest version of the Carla simulator, Town10HD, with a success-rate of 77%.

Keywords: Imitation Learning · Autonomous Vehicles · CARLA simulator.

1 Introduction

The role of machine learning and in particular deep learning in the car industry is greater than ever before [2]. Well known car companies introduce more and more smart assistants and some companies like Tesla even construct cars capable of level 3 autonomous driving [11]. Those innovative products may provide many benefits, but the main objection for wide adoption is safety. Last year the Dutch *Onderzoeksraad voor Veiligheid* investigated and reported an accident where Tesla's AutoPilot system did not sufficiently recognize a truck changing lane [9]. This caused the Tesla to be caught between the crash barrier and the truck.

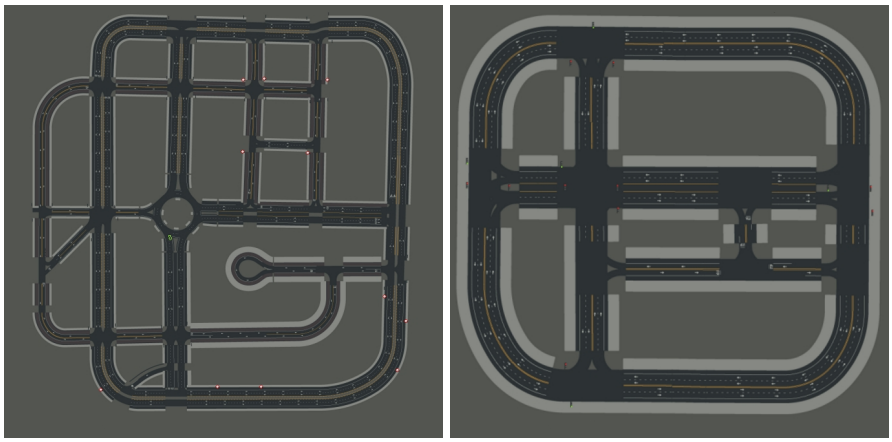
The report concluded that driving assistant systems may take unexpected actions which may confuse the driver. In addition, the system's choices are difficult to explain and even more difficult to compare with other systems. A lack of insight in a model's decisions is also known as the black box problem. With

* Supported by the Meaningful Control of Autonomous Systems initiative from TNO, CWI and UvA.

Meaningful Control for Autonomous Systems (MCAS) we aim to provide better insight in the black box of a smart system. To better understand and improve upon driving assistants a repeatable experiment is key. Therefore, we use a photo-realistic simulator as testing environment.

CARLA is a state-of-the art simulator which includes many complex aspects of the real world such as pedestrians, different junction types, multiple-lane roads, fifteen different weather conditions [5] and traffic simulation. In addition, the quality of the representation of the real world is outstanding, including a variety of assets such as houses, street lights, trees and other vehicles. The current 10 maps range from rural driving areas to highways. Figure 1 shows the layout of Town 3 which we use for training (1a) and the layout of CARLA's newest Town 10HD which we use for testing (1b) in this research.

Convolutional neural networks (CNN) have proven to be very accurate at learning features from two-dimensional images [8]. End-to-end imitation learning takes advantage of these features and tries to solve mapping from raw sensory input to an action output [6]. Therefore, the network simultaneously learns to extract features as well as a decision policy based on those features. We use a CNN network inspired by NVIDIA's approach to learn a discrete steering angle from a single raw input image [3].



(a) Town 3 for training. (b) Town 10HD for validation.

Fig. 1: Top-down layout view of used CARLA maps.

2 Related work

A lot of research has been done in the past few years on the topic of autonomous driving with end-to-end imitation learning. Bojarski *et al.* from NVIDIA have shown that an end-to-end approach is able to learn to follow the road and stay in

one lane [3]. They used a triple-cam setup in which three cameras were mounted behind the car's windshield. Figure 2 shows such an triple-cam setup, as in the recently announced Autopilot 2.0/Tesla Vision hardware suite. The two off-center cameras provided the model multiple viewpoints with according steering angles. A relatively small CNN with 250 thousand parameters was fed an input image from one of the three cameras and the current steering angle. In addition, a random shift and rotation were applied on the input. After training, the model could accurately predict the steering angle from only an input image from the center camera. Even when transferred to the real world the model obtained high autonomy scores of 98%. However, those real world tests excluded lane changes and road-crossing turns.

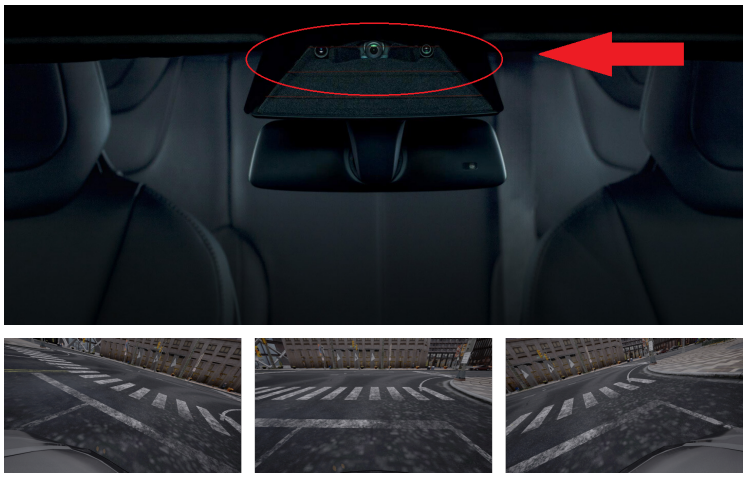


Fig. 2: Example of triple-cam setup behind car's windshield, with below this example the view of the triple camera's from left, middle to right camera.

Codevilla *et al.* from Intel Labs have combined high level human commands and CNNs [4]. In contrast to other approaches, the human interaction acts as a decision module for the network. A high level sub-module is activated upon the human's command. Besides the output of the sub-module an acceleration value is predicted. The dataset is generated in CARLA Town 1 using a triple-cam setup with human expert annotations. Crucial to the success of the model is data augmentation. By injecting noise to the steering commands of the teacher while driving, the dataset covers recovery situations. These might prevent overfitting to a perfect driver. Evaluation in Town 2 is done in a similar fashion as Bojarski *et al.* Results in simulation as well as in the real world, using a scale model radio controlled car, show that the proposed architecture is capable of reaching the goal without many failures.

Abdou *et al.* propose a network which combines Intel's and NVIDIA's approaches [1]. The CARLA Autopilot system is used to generate a large dataset of

images. The used model consists of a feature extraction part based on NVIDIA's model and is then conditioned on a CARLA command signal. This command signal determines which branch of the Intel inspired decision part is then used to predict the steering angle. Every branch corresponds to a possible task such as take a left turn, go straight. Besides, a speed branch is attached which predicts a throttle value. Evaluating is done in Town 1 and 2 of CARLA under different weather conditions. It outperforms Intel's architecture in almost all tasks, but this measurement is only based on the success rate where the model reached the destination in time. Factors such as collisions with other cars or traffic rules do not directly affect this rate.

Haavaldsen *et al.* were also inspired by NVIDIA's approach and used CARLA's Town 1 and 2 for training [6]. They augmented the CNN with a long-term short-term (LSTM) layer which is used as a feature extractor. The LSTM layer improved the robustness of the controller, with a superior reaction on sudden changes in the scenery caused by other traffic. In this study we concentrate on robust driving on the trajectory first, for the more challenging scenario of CARLA's Town 3.

3 Experiment

For this experiment we use CARLA simulator version 0.9.8 and Town 3 map for training and testing. Town 3 includes different complex situations such as 5-lane junctions, a roundabout and a tunnel. According to CARLA's documentation Town 3 is the most complex town. The recently released Town10HD was used for validation.

First of all, we generated a dataset by using the CARLA waypoint function. The selected ego-vehicle, a Toyota Prius, has a RGB camera mounted on its roof. The camera has a standard 105 degrees field of view (FOV) capturing an image at every simulator time step. A fixed velocity vector of $\begin{bmatrix}0.0001 & 0.0001 & 0.0001\end{bmatrix}$ is applied to the ego-vehicle which corresponds to approximately 0.1 km/h.

To prevent overfitting the weather conditions are cycled throughout the simulation to ensure an uniform distributed dataset per weather condition. The weather conditions change the lightning of the environment as well and can thus be seen as a form of data augmentation. To ensure a realistic path for the ego-vehicle we use the waypoints included in Town 3. Those waypoints are calculated by CARLA. At every time step we choose a random waypoint out of the set of valid next waypoints. Valid waypoints are defined by CARLA and are dynamically changing based on the ego-vehicles' position. For example, an intersection provides multiple valid waypoints for every possible turn. In addition, to prevent overfitting to a perfect path, we add noise to the location of the waypoint (1).

$$waypoint = \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} \mathcal{N}(0, 0.8) \\ \mathcal{N}(0, 0.8) \\ 0 \end{bmatrix} \tag{1}$$

Fig. 3: CARLA simulator with model inference vector (green) and vehicle forward vector (red).

The resulting dataset consists of 131 thousand 720×1280 RGB images including the corresponding steering angles. A sample with corresponding steering angle is shown in figure 4. Note that the steering angle is in the interval $\left[-1, 1\right]$.

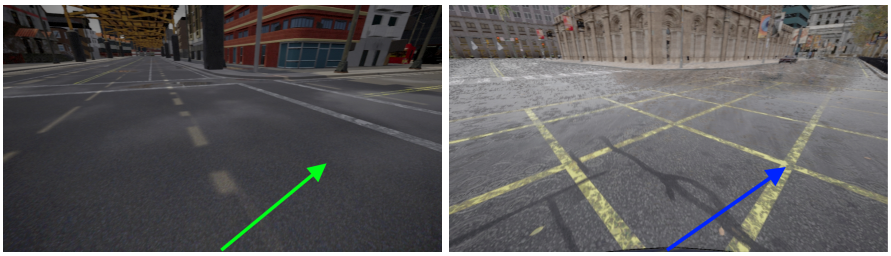


Fig. 4: Samples from encountered situation in Town 3 (left) and Town 10HD (right). Steering angles: 0.439225 (green vector) and 0.385478 (blue vector).

The CNN network (see Figure 5) consists of an input layer with shape $\left[720, 1280, 3\right]$ followed by a cropping and resizing layer which transforms the images to $\left[104, 256, 3\right]$. The transformation is done to reduce memory usage and to ensure the image only captures the road under the horizon. In addition the color space is transformed from RGB to YUV. The feature extraction part of the model is based on NVIDIA's architecture consisting of 5 convolution layers. The first 3 layers use a 3×3 convolution kernel and the last two use a 2×2 kernel. All convolution layers have Exponential Linear Unit (ELU) activation functions. The feature extraction part is followed by an decision module which

starts of with an 0.3 dropout layer. Followed by a flatten layer and two dense layers. The final output layer consists of 5 nodes with a softmax activation function. Categorical cross-entropy is used as loss function in combination with an Adam optimizer (learning rate $= 1e - 4$) [7].

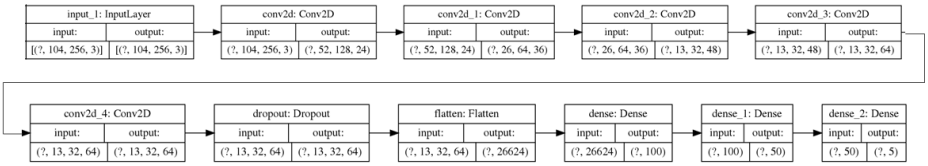


Fig. 5: Illustration of the model architecture. All question marks represent the non-fixed batch size.

To enhance insight in the model's decisions the direction vectors of the ego-vehicle and the model's prediction are plotted as overlay in the simulation. Figure 3 shows an example simulation snapshot with vector overlay. In addition, the continuous regression problem of predicting a steering angle is converted to a discrete classification problem. The model's five output nodes correspond to the action space of five possible steering angles: $\left[-20, -10, 0, 10, 20\right]$. Class balancing is done to create an equally distributed dataset along all classes.

Evaluating the model is done in two ways. First, the F1 scores are calculated for the validation set of the dataset. Second, the model is tested in the CARLA simulator itself combined with a custom benchmark. The benchmark consists of multiple episodes in which all steering is done by model inference. We use a random start position for the ego-vehicle at each episode which is a sample from the provided set of possible spawn positions by CARLA. During the benchmark a number of statistics are captured such as lane invasions and collisions. Those statistics are provided by the CARLA simulator.

4 Result

The model has trained on a GeForce RTX 2080 Ti graphics card and converged after 3 epochs in less than 3 hours (see Figure 6). Table 1 shows the model's detailed epoch F1 score and epoch loss across the training and validation set.

Table 1: Training and validation scores of model for the final epoch. Concerning precision, recall and F1 score higher is better. Only lower is better for the loss.

Dataset	Precision	Recall	F1 score	Loss
Training	0.9286	0.9078	0.9179	**0.198**
Validation	**0.9392**	**0.9182**	**0.9284**	0.2143

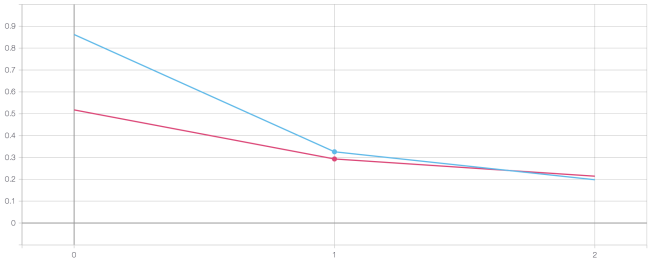


Fig. 6: Epoch loss of trained model over 3 epochs for training (blue) and test set (pink).

The benchmark introduced in section 3 has multiple configuration options. The velocity v of the ego-vehicle is fixed at 0.69 m/s. An episode is considered done when the ego-vehicle has covered a maximum distance d of 100 meters or caused a collision of any sort. In addition, to prevent endless episodes a time constraint, t, of 144 seconds is dynamically calculated (see Eq. 2). The CARLA simulation ran with the *-benchmark -fps 10* settings to ensure a constant frame rate and repeatable experiment. In addition, all other traffic was turned off.

$$t = \frac{d}{v} \tag{2}$$

Figure 7 shows the training map results from 56 episodes covering a total of 4530 meters and a run-time of 101 minutes. The high variance in duration and distance are due to the fact that a collision immediately ends the episode which causes lower distance and duration. In addition, the simulator lacks a true constant frame-rate hence the total number of frames simulated might differ per episode. However, this variance in frame-rate can also be seen as a domain randomisation. All possible lane types are shown on the x axis of Figure 7. Besides, to provide more relative perspective, the total number of episodes and the number of collisions are added at the end. Table 2 gives more details concerning collisions. The exact definition per lane type is defined by the OpenDRIVE 1.4 standard [10]. *Broken*, *NONE* and *Other* lane types may be crossed in real life circumstances. During the crossing of a junction the vehicle will always cross some of those lane marking types, since junctions contain lane markings too.

To evaluate the generalization performance of the model we used the benchmark in Town 10HD. This town is more realistic than Town 3 concerning textures, although the layout is slightly simpler. The possible traffic situations differ enough from Town 3 to make this a good validation map. This town has no roundabouts, but includes a larger junction which the model has never faced before (see Figure 4. As shown in figure 8 the number of lane invasions is almost equal to Town 3. However, the mean and variance concerning collisions is lower in comparison to Town 3. The benchmark has run for an equal number of episodes, covering 4072 meters and spanning a run-time of 114 minutes.

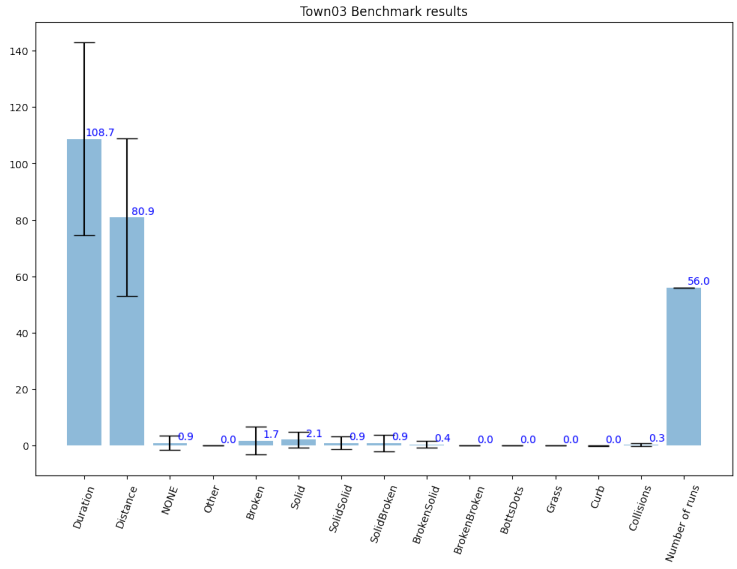


Fig. 7: Results benchmark training Town 3 over 56 episodes (4530 meters). The means are shown as blue labels and the error bars represent the variance.

To provide a more relative metric, we calculated the success-rate and summarized the lane invasions and collisions per meter. The success-rate is defined as the percentage of episodes in which the vehicle reached the end of the episode, in relation to the total number of episodes. In other words, the percentage of non-collisions episode in relation to the total number of episodes. Table 2 shows the success-rate specified for the training and validation map. In addition, the collisions and lane invasions per meter are calculated as the sum of collisions and invasions respectively divided by the total distance covered during the benchmark. Note that in the calculation of the lane invasions per meter, only the non-collision episodes are considered. It is assumed that a collision will most of the time cause more lane invasions, leading to an extremely unbalanced number of lane invasions per meter.

Table 2: Objective generalization results. Concerning success-rate higher is better. Both for collisions and lane invasions per meter lower is better.

Town	Success-rate	Collisions per meter	Lane invasions per meter
Town 3	67.86%	0.0040	**0.0510**
Town 10HD	**76.79%**	**0.0032**	0.0626

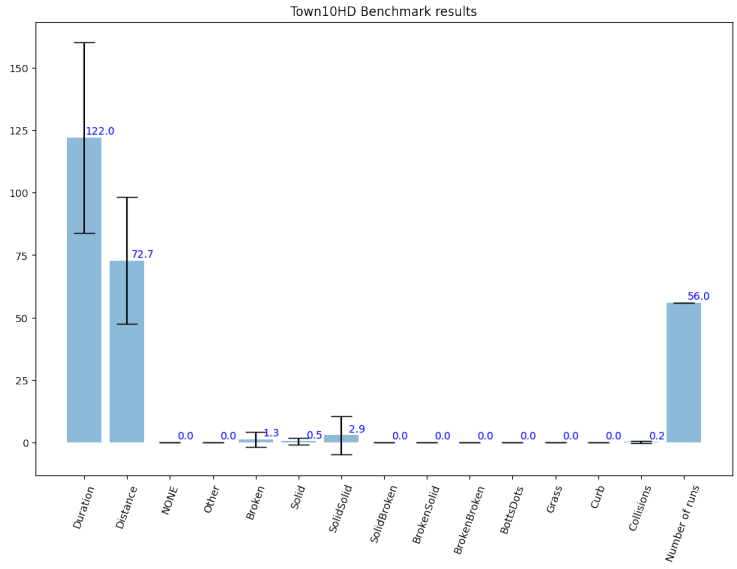


Fig. 8: Results benchmark testing Town 10 over 56 episodes (4072 meters). The means are shown as blue labels and the error bars represent the variance.

5 Discussion

As shown by Haalvaldsen *et al.* [6] adding a specialized layer suchs as a Long Short Term Memory (LSTM) to the model improves the general model's performance. Incorporating a segmentation layer in our proposed network might improve the model's performance as well. A segmentation layer should be able to mask lane markings resulting in only a few classes per image instead of viewing every pixel independently. Therefore, a better intermediate representation might be achieved which could result in further optimized policies.

To better facilitate goal-directed scenarios a high level conditional command unit as proposed by Codevilla *et al.* [4] might be necessary. Although adding such a module still relies on a robust feature extraction model. Thoroughly testing and evaluating this model's foundation should therefore always be done first.

Our proposed benchmark in combination with the simulation vector overlay gives more insight in the model's shortcomings. However, future research in the model's internal state representations could facilitate more insight in the reasoning behind the model's decisions. Regarding safety, explainable decisions should be a huge step towards safer systems.

The rapid development of CARLA causes inconsistency between studies with other training and testing environments. Other researches have used older, less complex towns such as Town 1 and Town 2 [6][1]. In contrast, we use the most complex and realistic towns CARLA currently offers. Comparing the conclu-

sion and results between different papers should therefore be done with great precision.

6 Conclusion

We have proposed a CNN network which is capable of controlling a vehicle's steering with 77% success-rate in unseen circumstances. The benchmark quantifies a set of the model's shortcomings. The potential of end-to-end imitation learning on a single camera stream therefore can be exploit even further. In addition, testing in a highly realistic environment such as Town 10HD proves the model to be robust to changes in lightning conditions and different road layouts. We propose the use of realistic simulation environments in combination with robust networks and benchmarks to provide a starting point to improve the safety of modern day driving assistants.

7 Acknowledgements

Special thanks go to Thomas Wiggers for noteworthy suggestions and helpful discussions.

References

1. Abdou, M., Kamal, H., El-Tantawy, S., Abdelkhalek, A., Adel, O., Hamdy, K., Abaas, M.: End-to-end deep conditional imitation learning for autonomous driving. In: 2019 31st International Conference on Microelectronics (ICM). pp. 346–350 (2019)
2. Badue, C., Guidolini, R., Carneiro, R.V., Azevedo, P., Cardoso, V.B., Forechi, A., Jesus, L., Berriel, R., Paixão, T.M., Mutz, F., et al.: Self-driving cars: A survey. Expert Systems with Applications p. 113816 (2020)
3. Bojarski, M., Testa, D.D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K.: End to end learning for self-driving cars. CoRR **abs/1604.07316** (2016)
4. Codevilla, F., Müller, M., López, A., Koltun, V., Dosovitskiy, A.: End-to-end driving via conditional imitation learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 1–9. IEEE (2018)
5. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)
6. Haavaldsen, H., Aasboe, M., Lindseth, F.: Autonomous vehicle control: End-to-end learning in simulated urban environments. In: Symposium of the Norwegian AI Society. pp. 40–51. Springer (2019)
7. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
9. Onderzoeksraad voor Veiligheid: Wie stuurt? Verkeersveiligheid en automatisering in het wegverkeer. Published online (2019)
10. OpenDRIVE1.4: Format specification, rev. 1.4. Published online (2015)
11. SAE International: Automated driving – levels of driving automation are defined in new sae international standard j3016. Published online (2014)