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 compare different convolutional neural networks trained 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 in the training map, a behavior that can be transferred to a non-complex map with circumstances it has not seen before. Complex situations remain difficult to learn without high-level commands. The learned behavior has been validated on a map which is just released with the latest version of the CARLA simulator, Town 10HD. The Xception network architecture performs best in our benchmark with success rates of 34% and 90% for complex validation town Town 10HD and non-complex validation town Town 6 respectively.

Keywords: End-to-end Imitation Learning · Autonomous Vehicles · CARLA simulator.

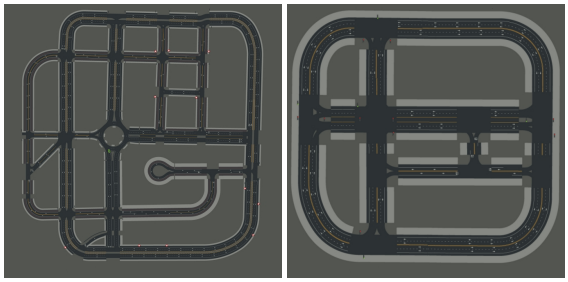
1 Introduction

The role of machine learning and in particular deep learning in the car industry is greater than ever before [3]. Well-known car companies introduce more and more smart assistants, and some companies like Tesla even construct cars capable of level 3 autonomous driving [16]. Those innovative products may provide many benefits, but the main objection for wide adoption is safety. Last year the Dutch *Onderzoeksraad voor Veiligheid* investigated several accidents at the Dutch highways. The report concluded that driving assistant systems may take unexpected actions that may confuse the driver. Besides, the system's choices are

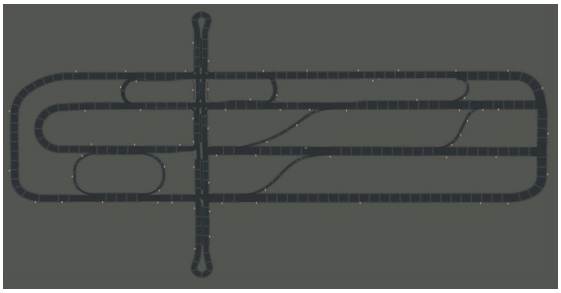
* Supported by the Meaningful Control of Autonomous Systems initiative from TNO, CWI and UvA.

difficult to explain and even more difficult to compare with other systems. A lack of insight into a model's decisions is also known as the black box problem. With Meaningful Control for Autonomous Systems (MCAS), we aim to provide better insight into 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 that includes many complex aspects of the real world such as pedestrians, different junction types, multiple-lane roads, fifteen different weather conditions [9], 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 that we use for training (1a), the layout of CARLA's newest Town 10HD that we use for testing (1b), and the layout of CARLA's highway Town 6 that we use for additional validation in this research (1c).



(a) Town 3 (b) Town 10HD



(c) Town 6

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

Convolutional neural networks (CNN) have proven to be very accurate at learning features from two-dimensional images [15]. End-to-end imitation learning takes advantage of these features and tries to solve mapping from raw sensory input to an action output [10]. Therefore, the network simultaneously learns to extract features as well as a decision policy based on those features. We compare different state-of-the-art CNN networks to learn a discrete steer-

ing angle from a single raw input image. Our baseline network is inspired by NVIDIA's approach [4].

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 can learn to follow the road and stay in one lane [4]. They used a triple-cam setup in which three cameras were mounted behind the car's windshield. Figure 2 shows such a 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 250.000 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 to 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.

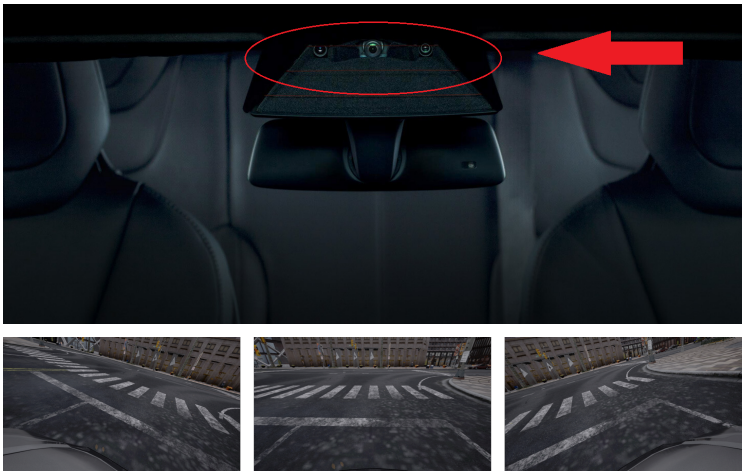


Fig. 2: Example of triple-cam setup behind car's windshield, with below this example the view of the triple camera's from left, middle to right camera.

Codevilla *et al.* from Intel Labs have combined high-level human commands and CNNs [7]. In contrast to other approaches, 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 includes 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 that 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 that 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 [10]. 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 to sudden changes in the scenery caused by other traffic.

Chen *et al.* propose a new approach to imitation learning [5]. First, an agent learns a driving policy by imitating an expert. The agent in this stage is provided with all environmental information. Second, a new agent learns to imitate the first agent but without access to the environment's state. They achieve a 100% success rate on the CARLA benchmark in Town 1 and Town 2.

In this study, we concentrate on robust driving on the trajectory first, for the more challenging scenario of CARLA's Town 3. In addition, we aim to identify the agents' strengths and weaknesses to set the outline for future work.

3 Experiment

For this experiment, we use CARLA simulator version 0.9.8 and Town 3 map for training and testing. Note that the logic inside the simulator changed significantly recently, which means that the results in this paper cannot directly be compared with results obtained from versions $< 0.9.6$. The results can be as large as a difference in completion rate of 30% [5]. 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. The recently released Town 6 and Town 10HD are used for validation.

First of all, we generated the 2020621 dataset by using the CARLA waypoint function. The selected ego-vehicle, a Toyota Prius (see Fig. 3), has an RGB camera mounted on top its windshield. 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.

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

To prevent overfitting the weather conditions are cycled throughout the simulation to ensure a 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. Also, to prevent overfitting to a perfect path, we add noise to the location of the waypoint (1).

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

The resulting dataset consists of 131 thousand 720×1280 RGB images, including the corresponding steering angles. A sample with the corresponding steering angle is shown in Figure 4. Note that the steering angle is in the interval $\begin{bmatrix}-1, 1\end{bmatrix}$.

Our baseline CNN network (see Fig. 5) consists of an input layer with shape $\begin{bmatrix}720, 1280, 3\end{bmatrix}$ followed by a cropping and resizing layer which transforms the images to $\begin{bmatrix}104, 256, 3\end{bmatrix}$. The transformation is done to reduce memory usage and

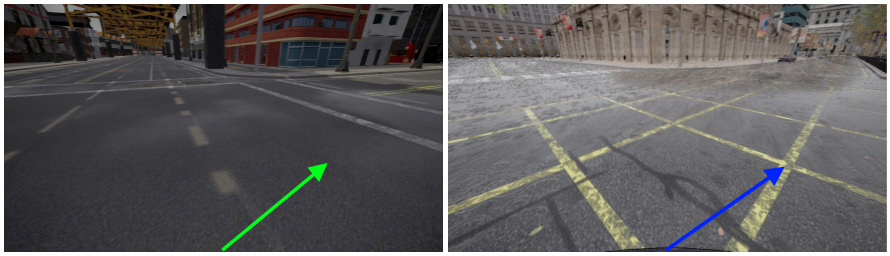


Fig. 4: Samples from encountered situation in Town 3 (left) and Town 10HD (right). Steering angles: 0.439225 (green vector) and 0.385478 (blue vector).

to ensure the image only captures the road below the horizon. Besides, 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 a decision module that starts with a 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 (with learning rate 10^{-4}) [14].

We compare our baseline to the following state-of-the-art CNN networks: ResNet50 [11], ResNet101V2 [12], DenseNet121 [13], Xception [6] and EfficientNetB7 [17]. All architectures are extended with a flatten and softmax dense layer to match the five class output. Table 1 compares the number of trainable parameters for all models. Our network is the smallest in terms of trainable parameters.

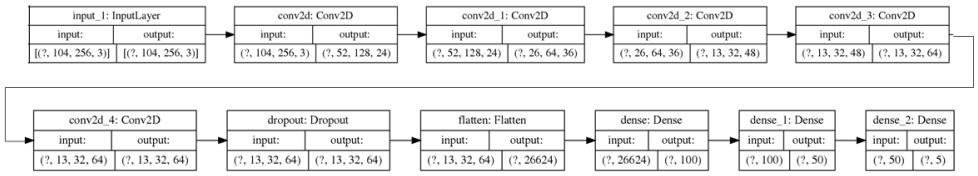


Fig. 5: Illustration of the model architecture. All question marks represent the non-fixed batch size.

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

Evaluating and comparing the models is done in two ways. First, the F1 scores are calculated for the validation set of the dataset. Second, the models are tested in the CARLA simulator combined with a custom benchmark to provide success rates and lane marking infractions.

To average out CARLA's non-determinism, the benchmark consists of a number of runs that consist of several episodes. During an episode, 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, several statistics are captured such as lane marking invasions and collisions. Those statistics are provided by the CARLA simulator.

In this study, we configured the benchmark to match the original CARLA benchmark as good as possible [9]. The velocity v of the ego-vehicle is fixed at 2.78 m/s. An episode is considered done when the ego-vehicle has covered a maximum distance d of 200 meters or caused a collision of any sort. In addition, to prevent endless episodes a time constraint, t, of 79.2 seconds is dynamically calculated (see Eq. 2).

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

We set the benchmark to 25 episodes per run for a total of 5 runs per model. This comes down to a maximum of 25 000 driven meters per model. The CARLA simulation ran with the *-benchmark -fps 25* settings to ensure a constant frame rate and repeatable experiment. Besides, all other traffic was turned off.

4 Result

Table 1 shows the models' detailed epoch F1 scores and epoch losses across the training and validation set. The number of epochs the models took to converge is mentioned as well. All models converge to an excellent validation F1-score of 0.90 or above.

Table 1: Model comparison on CARLA 20200621 dataset. All values are from the best epoch, based on validation loss. Best values are marked bold.

Model	Trainable parameters	Epochs	Train Loss	Val Loss	Train F1	Val F1
Our work	**2.799.153**	31/50	**0.002762**	**0.02298**	**0.9989**	**0.9956**
ResNet50	23.862.277	34/50	0.0085174	0.03973	0.9949	0.9887
ResNet101V2	42.856.581	9/50	0.191	0.1711	0.9172	0.9402
DenseNet121	7.076.741	5/50	0.1917	0.2192	0.9147	0.9193
Xception	21.134.637	**4/50**	0.1649	0.1539	0.9237	0.959
EfficientNetB7	64.196.565	16/50	0.1417	0.1943	0.9271	0.9336

Table 2: Benchmark results in training town, Town 3, with mean and standard deviation below. All values, except "Success rate", are per kilometer. All lane marking types are in italics. Only lane markings with non-zero values are shown. Best values are marked bold.

Model	Success rate	Infractions	*NONE*	*Broken*	*Solid*	*SolidSolid*	*SolidBroken*	*BrokenSolid*	*Curb*
Our work	0.504 ±0.0599	**0.5652** ±0.5172	**0.065** ±0.1468	0.1132 ±0.3325	0.2113 ±0.2778	0.062 ±0.1227	**0.0562** ±0.1359	0.0553 ±0.145	0.0021 ±0.0192
ResNet50	0.592 ±0.064	0.6577 ±0.4085	0.0846 ±0.1502	**0.0906** ±0.1296	0.2444 ±0.2811	0.06564 ±0.1184	0.0929 ±0.1858	0.0759 ±0.1451	0.0036 ±0.0251
ResNet101V2	0.368 ±0.1372	1.1111 ±1.6274	0.1272 ±0.3134	0.0915 ±0.162	0.4325 ±0.8442	0.161 ±0.4959	0.1957 ±0.6192	0.0983 ±0.3007	0.0049 ±0.0325
DenseNet121	0.616 ±0.0697	0.6635 ±0.5414	0.0761 ±0.1581	0.186 ±0.3218	0.2125 ±0.3239	**0.0411** ±0.0984	0.0598 ±0.1992	0.0773 ±0.2046	0.0106 ±0.0715
Xception	**0.624** ±0.0824	0.7092 ±0.5159	0.1159 ±0.1847	0.0952 ±0.2104	0.2294 ±0.2934	0.0862 ±0.1986	0.0971 ±0.221	0.0839 ±0.1847	0.0016 ±0.0122
EfficientNetB7	0.4 ±0.1131	0.6449 ±0.5851	0.0807 ±0.1838	0.1347 ±0.2267	**0.2083** ±0.3301	0.074 ±0.1764	0.0917 ±0.2467	**0.0545** ±0.2152	**0.001** ±0.0106

The results of the benchmark in the training town, Town 3, are shown in Table 2. The exact definition per lane marking type is defined by the OpenDRIVE 1.4 standard [2]. *Broken*, *NONE* and *Other* lane marking 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 provide a more relative metric, we calculated the success-rate and averaged the lane marking infractions over the lane marking types per kilometer. 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-collision episodes in relation to the total number of episodes.

Even though the models are trained and benchmarked in the same town, it proves to be difficult to reach high success rates. Our method seems to perform well in terms of infraction rate, but Xception outperforms ours in terms of success rate. DenseNet121 performs very well too, since its *Solid* and *SolidSolid* infraction rates are best or second-best in addition to a second-best success rate. DenseNet121 is about 3 times smaller than Xception in terms of trainable parameters, see Table 1.

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 1b and Figure 4. As shown in Table 3 almost every model yields a two times lower success rate in comparison to the training town. Xception is the best model in terms of success and infraction rate. Our method yields

Table 3: Benchmark results in validation town, Town 10HD, with mean and standard deviation below.

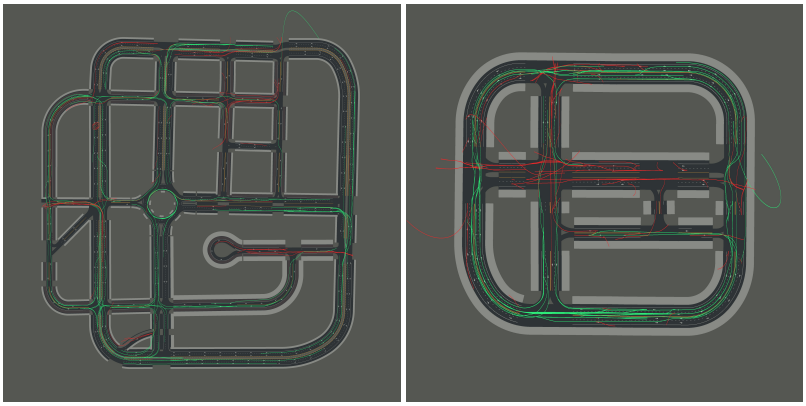
Model	Success rate	Infractions	*NONE*	*Broken*	*Solid*	*SolidSolid*
Our work	0.304 ±0.1061	0.5982 ±1.1649	0.002 ±0.0166	0.273 ±0.5517	0.1579 ±0.3514	0.1653 ±0.3935
ResNet50	0.288 ±0.0776	0.4774 ±0.5229	0.0054 ±0.0364	0.2451 ±0.3729	**0.1142** ±0.1985	**0.1128** ±0.2142
ResNet101V2	0.28 ±0.0912	0.7119 ±1.3838	0.0034 ±0.031	0.3664 ±0.7647	0.1361 ±0.3236	0.2059 ±0.4813
DenseNet121	0.24 ±0.1315	0.9242 ±1.1605	0.0059 ±0.0401	0.407 ±0.5079	0.2153 ±0.3594	0.296 ±0.6334
Xception	**0.3449** ±0.150	**0.4475** ±0.6345	**0.0** ±0.0	**0.178** ±0.2598	0.1167 ±0.2202	0.1528 ±0.4305
EfficientNetB7	0.184 ±0.0543	0.85 ±1.8165	0.0056 ±0.0443	0.3671 ±0.9452	0.2146 ±0.452	0.2626 ±0.5982

a second-best success rate but has a significantly higher infraction rate in comparison to Xception. ResNet50 performs best concerning *Solid* and *SolidSolid* infraction rates.

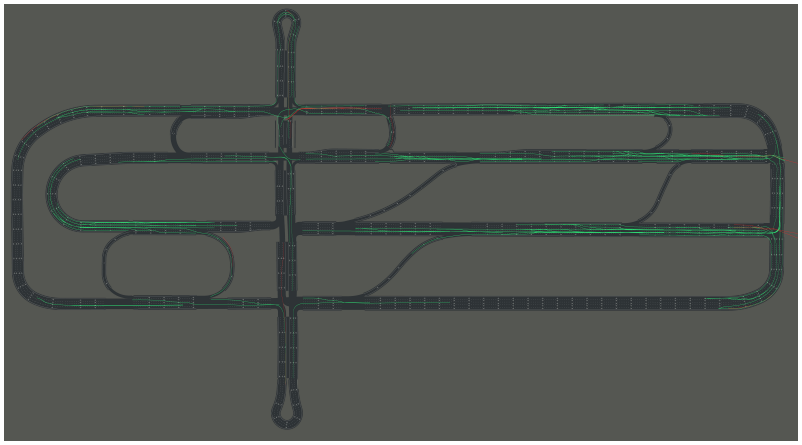
Figure 6 shows the trajectories driven by the best performing model, Xception, during the benchmark in Town 3 (6a), Town 6 (6c) and Town 10HD (6b). We used Town 6 as an additional validation benchmark since Figure 6b shows that the Xception agent lacks at handling complex situations such as large junctions in Town 10HD. Town 6 is mainly highway and therefore a good validation for non-complex situations. Table 4 shows the results of the benchmark for the Xception agent in Town 6 in comparison to Town 3 and Town 10HD. The success rate in validation town Town 6 is about 2.5 times higher, compared to validation town Town 10HD. Note that infractions and lane marking types cannot be directly compared since the distribution of lane markings differ from town to town.

Table 4: Xception Agent validation on Town 6 (Highways). Lane markings with no infractions are left out.

Town	Success rate	Infractions	*NONE*	*Broken*	*Solid*	*SolidSolid*	*SolidBroken*	*BrokenSolid*	*Curb*
Town 3	0.624 ±0.0824	0.7092 ±0.5159	0.1159 ±0.1847	0.0952 ±0.2104	0.2294 ±0.2934	0.0862 ±0.1986	0.0971 ±0.221	0.0839 ±0.1847	0.0016 ±0.0122
Town 6	**0.904** ±0.0599	**0.4218** ±0.4038	0.0346 ±0.088	0.3563 ±0.3865	0.0309 ±0.0753	- -	- -	- -	- -
Town 10HD	0.3449 ±0.150	0.4475 ±0.6345	0.0 ±0.0	0.178 ±0.2598	0.1167 ±0.2202	0.1528 ±0.4305	- -	- -	- -



(a) Town 3 (b) Town 10HD



(c) Town 6

Fig. 6: Trajectories driven during benchmark for Xception model. Green trajectories are successful. Red trajectories are failures due to a collision.

5 Discussion

We hypothesize that the feature extraction capabilities of our proposed model, in comparison to the Xception model, are correlated with the difference between our and Xception's infraction rate. Chollet [6] mentions that depthwise separable convolutions, as used in Xception, outperform standard convolution on image tasks. These depthwise separable convolutions could therefore result in better feature extraction capabilities of the Xception model in comparison to our model since we only use standard convolutions. More robust feature extraction may subsequently increase the accuracy of identifying lane markings and therefore possibly lowering infraction rates.

As shown by Haalvaldsen *et al.* [10] adding a specialized layer such 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 and improve performance in complex situations a high-level conditional command unit as proposed by Codevilla *et al.* [7] 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 be done first.

Our proposed benchmark in combination with the simulation vector overlay gives more insight into the model's shortcomings. However, future research in the model's internal state representations could facilitate more insight into the reasoning behind the model's decisions. Approaching the problem in two stages as proposed by Chen *et al.* [5] could improve this too. 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 [10][1]. Yet, according to Chen *et al* [5] this is a solved problem. Even the more recent NoCrash benchmark [8] is based on those two towns. In contrast, we use the most complex and realistic town CARLA currently offers. Comparing the conclusions and results between different papers should therefore be done with great care.

6 Conclusion

We have proposed a detailed comparison of different state-of-the-art CNN networks and our baseline. The proposed benchmark quantifies a set of the models' shortcomings in addition to the trajectory plots. Xception has proven to perform best, especially in non-complex situations with a 90% success rate. A specific set of problematic situations facilitates the potential of end-to-end imitation learning on a single camera stream to be exploited even further. In addition, testing in a highly realistic environment such as Town 10HD ensures models to be robust to changes in lighting conditions. 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.

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: 31st International Conference on Microelectronics (ICM). pp. 346–350. ICM (2019)
2. Association for Standardization of Automation and Measuring Systems: ASAM OpenDRIVE: Open dynamic road information for vehicle environment, version 1.6. Published online (March 2020)
3. 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 **165**(no. 113816) (March 2021)
4. 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)
5. Chen, D., Zhou, B., Koltun, V., Krähenbühl, P.: Learning by cheating. In: Conference on Robot Learning. vol. 100, pp. 66–75. PMLR (2020), preprint arXiv:1912.12294
6. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: IEEE conference on Computer Vision and Pattern Recognition. pp. 1251–1258. CVPR (2017), preprint arXiv:1610.02357
7. Codevilla, F., Müller, M., López, A., Koltun, V., Dosovitskiy, A.: End-to-end driving via conditional imitation learning. In: IEEE International Conference on Robotics and Automation. pp. 1–9. ICRA (2018), preprint arXiv:1710.02410
8. Codevilla, F., Santana, E., Lopez, A.M., Gaidon, A.: Exploring the limitations of behavior cloning for autonomous driving. In: IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019), preprint arXiv:1904.08980
9. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: 1st Annual Conference on Robot Learning. pp. 1–16 (2017), preprint arXiv:1711.03938
10. 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), preprint arXiv:1905.06712
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: IEEE conference on Computer Vision and Pattern Recognition. pp. 770–778. CVPR (2016), preprint arXiv:1512.03385
12. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: European Conference on Computer Vision. pp. 630–645. Springer (2016), preprint arXiv:1603.05027
13. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: IEEE conference on Computer Vision and Pattern Recognition. pp. 4700–4708. CVPR (2017), preprint arXiv:1608.06993
14. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representation. ICLR (2015), preprint arXiv:1412.6980
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105. NIPS (2012)
16. SAE International: Automated driving – levels of driving automation. Defined in SAE International Standard J3016. Published online (2014)
17. Tan, M., Le, Q.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning. pp. 6105–6114. PMLR (2019), preprint arXiv:1905.11946