Training traffic light behavior with end-to-end learning

Maël Wildi[1], Alexandre Alahi[1], and Arnoud Visser[2]

[1] EPFL, Visual Intelligence for Transportation Laboratory,
Bâtiment GC, Station 18, CH-1015 Lausanne, Switzerland
[2] Universiteit van Amsterdam, Intelligent Robotics Lab,
Science Park 904, 1098 XH Amsterdam, The Netherlands

Abstract. In this work, we study neural network architectures that will reduce the number of infractions made by autonomous-driving agents. These agents control vehicles by providing future waypoints directly from a forward-facing camera. Building on top of the teacher-student approach of *Cheating by Segmentation*, we investigate the impact of Pyramid Pooling Module and Feature Pyramid Network with the aim to learn more representative features. We run our experiment with CARLA simulator and show that pyramid perception modules have a positive impact in reducing the number of traffic light infractions and collisions.

Keywords: conditional imitation learning, feature pyramid network

1 Introduction

In 2021, Honda in Japan and Mercedes in Germany received the authorization to deploy a vehicle where the driver is allowed to let the car drive itself, as long as they do not exceed a speed of 60km/h and are able to take back control if needed.

Most self-driving vehicles are based on a modular approach, where perception, planning and control are separated from one another [13]. In the last decade, the use of convolutional neural networks (CNN) has exploded, thanks to the increased power of computers. It is now widely used for perception tasks, as it enables to learn a model capable of recognizing objects from a large amount of annotated images. For autonomous driving, this enabled the possibility of end-to-end learning, an alternative to the modular method [10]. The control commands for the autonomous car are directly derived from the sensory input (such as a forward-looking camera), letting the network learn the intermediate features relevant for this task.

Imitation learning is, alongside the emerging reinforcement learning approach, the main method to perform end-to-end learning for autonomous cars [9]. It consists in training a model to reproduce the actions an expert would have performed if presented to the same situation.

Many models rely on data collected by a human driver [1], however driving simulators such as Car Learning to Act (CARLA) [6] are more and more used as they are getting very realistic. Most importantly, a simulator allows to put

Fig. 1: Illustration of CARLA simulator, widely used for autonomous driving [5]

the autopilot in constant varying situations, which is essential for efficient learning. Secondly, it enables to have scenarios where other cars and pedestrians are disrespecting traffic rules, which is important in order to learn a robust policy trained for worst-case scenarios.

Learning by Cheating (LBC) [3], one of the best performing driving policy on CARLA simulator, was modified in *Cheating by Segmentation* (CBS) [8] to rely only on images coming from a driver perspective instead of a bird's-eye view (BEV). It led to promising results, however the number of collisions and traffic lights infractions increased due to the less informative viewpoint.

Thus, this study pursues the goal of generating a robust end-to-end imitation learning driving policy using RGB and semantically segmented images from a driver perspective as only inputs. The focus is on reducing the number of traffic light infractions and collisions. In order to do this, we rely on pyramid perception modules that are described in the Method section.

2 Related work

Although there are several interesting developments in learning to drive based on existing datasets (see for instance [12], [4] and [9]), we concentrate for conciseness on methods that train in an end-to-end fashion an autonomous agent to drive using CARLA simulator.

2.1 Conditional imitation learning

Learning by Cheating *Learning by Cheating* (LBC) [3] proposes a novel way of training an agent to learn a robust driving policy. It separates the learning process in two: learning to act and learning to perceive using a teacher-student network approach.

The teacher agent is trained offline with the supervision of a dataset of expert trajectories and is privileged in the sense that it has access to a bird's-eye view (BEV) semantically segmented image. The student agent is trained under the

supervision of the teacher and has only access to a standard forward-facing RGB picture.

Both the teacher and student agents have multiple prediction heads connected to the backbone, which outputs for each of the possible directional command (turn left, turn right, go straight, follow lane) heatmaps that are converted into waypoints using a soft-argmax operation. These waypoints are then converted to vehicle commands by a PID controller.

The dataset of expert trajectories is collected directly in the simulator using an agent based on CARLA autopilot. It collects both the BEV semantically segmented image and the forward-facing RGB image. Data augmentation is performed by rotating the BEV to simulate steering noise. Multiple driving episodes are gathered in different traffic and weather conditions.

Cheating by Segmentation Although the student of LBC could be transferred to the real-world without too much difficulties according to the authors, the teacher used to train it relies on the ground-truth BEV semantic segmentation image which could be difficult and expensive to gather. *Cheating by Segmentation* (CBS) [8] addresses this by replacing it by a 120deg field-of-view forward-facing camera (although still semantic segmented for the teacher). This would enable to use pre-existing large datasets to train the student.

CBS gives promising results, but has difficulty anticipating braking actions, which results in a much higher amount of collisions and traffic light infractions than LBC on the *NoCrash* [5] benchmark. According to its author, it could be a consequence of the change of perspective in the segmentation camera from BEV to forward-facing: in the perspective view, close objects appear bigger which results in a shorter-term behaviour (a strong signal at a late moment, resulting in late braking).

2.2 Reinforcement learning

World on rails *Learning to drive from a world on rails* (WoR) [2], published in 2021 by the same authors as LBC, proposes a model-based reinforcement learning method. It relies on a model of the ego-vehicle, which enables to simulate the outcome of the agent's actions. This model is learned by training a network to predict the next agent's state (location, orientation, speed) given its initial state and an action (steer, throttle, brake). This is done using pre-recorded driving logs from CARLA.

The main assumption of the paper is that the world is on rails, meaning that the agent has no influence on its environment. Consequently, the latter does not depend on the agent's state or actions which means the initial world state determines the entire sequence of world states. As the agent is unable to change this sequence, the state transitions for the world are simply the ordered sequence of pre-recorded world states from the driving logs.

The model of the ego-vehicle (and the known sequence of world states) are used alongside a reward function to compute the Q-value, $Q_t(s, a)$, for all possible combination of agent state s and action a at timestep t. The Q-value is the

reward received for taking a given action in the current state plus the discounted estimated optimal return on the long term that it will get in the state it ends up in. The reward function is designed in a way to encourage the vehicle to stay within the target lane as well as to stop for a pedestrian or a traffic light, and is penalized otherwise. The Q-value table is used to supervise a visuomotor agent that takes a RGB image and the vehicle speed as input.

3 Method

The approach taken in this study uses as starting point *Cheating by Segmentation* (CBS): a teacher learns to drive from a dataset of forward-facing semantically segmented images and predicts waypoints in the image frame. It then supervises a student, that has the same image perspective as the teacher but with an RGB image input. Finally, the learned student model is used alone in the environment and its predicted waypoints are converted by a controller into vehicle commands.

In this study, the goal is to modify the student network architecture using two pyramid perception modules, PPM and FPN, which will be described later in this section.

3.1 Network Architecture

This section presents the network architecture on which this study relies. The learning process uses a teacher-student approach. The teacher is supervised by ground truth waypoints retrieved from expert driving logs and, once trained, is used to supervise a student. Figure 2 illustrates the whole framework. In the following sub-sections, the teacher and student are presented in detail.

Teacher The teacher architecture is unchanged from CBS. The network gets as input the forward speed of the ego-vehicle `spd` and the forward-facing semantically segmented image `SSI` representing roads, road lines, red traffic lights, cars, and pedestrians. The image is fed into a Resnet18 backbone and its output at the 4th layer, just before the fully connected layers, is retrieved. It is then concatenated with a repeated version of the forward speed `spd` and fed to one of the specialized waypoint prediction head, according to the directional command `cmd`.

These prediction heads perform deconvolution to predict heatmaps that are converted into waypoints in the image frame using a spatial arg-softmax. The model is supervised by the ground truth waypoints coming from the dataset. The loss function is the mean squared error (MSE) between the network predictions and the ground truth waypoints.

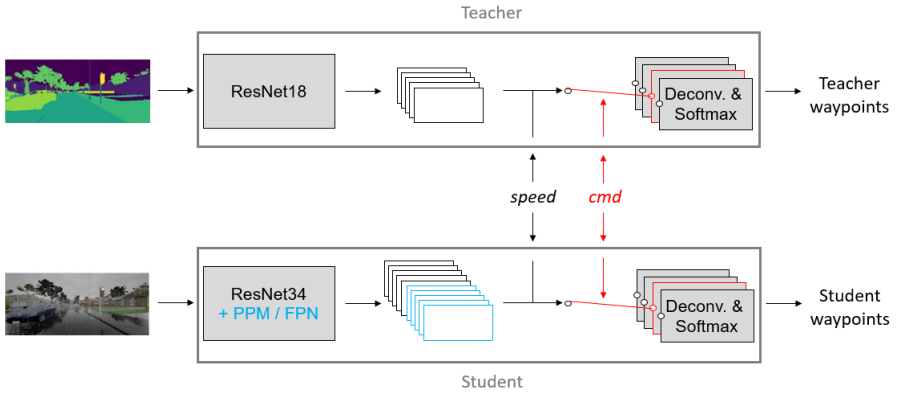

Fig. 2: Illustration of the network architecture. The teacher takes a semantically segmented image as input, while the student gets an RGB image. The teacher is supervised by the ground truth waypoints and the student by the teacher's.

Student The student uses the same framework as the teacher, except that it takes an RGB image as input and uses a ResNet34 backbone instead of a ResNet18. It is supervised by the waypoints predictions of the teacher. The student uses the same loss function as the teacher.

The main objective of this study is to improve the traffic light perception in the network, with the two modules detailed hereafter.

Pyramid Pooling Module Pyramid Scene Parsing Network (PSPNet) [14] was introduced in 2017 and aim to increase the effective receptive field and add context to the final feature maps of a feature extractor backbone such as ResNet. The heart of the PSPNet is the Pyramid Pooling Module (PPM). For different scales, adaptive average pooling is applied on these feature maps, followed by an up-sampling step to bring them back to their original dimensions. Then, the newly obtained maps are concatenated to the original ones. This provides for each pixel multiple level of context: from global context (whole image) to local context (small sub-region around the pixel). In other words, by looking at all the feature maps at a same specific location, information about features concerning the pixels outside its receptive field is also provided.

This PPM module is integrated into the architecture by placing it after the ResNet34 backbone but before the concatenation with the ego-vehicle speed. It thus takes as input the feature maps of the last ResNet34 convolutional layer.

Using PPM should provide a more robust representation of the image by fusing together features from different sub-regions. Its authors have established its effectiveness against mismatched relationships. In our case, a potential mismatched relationship would be to confound a car rear light for a red traffic light.

Feature Pyramid Network Feature Pyramid Network (FPN) [7], also introduced in 2017, is quite similar to PSPNet. However, its feature maps are coming from different layers, not only the last one. They are obtained by concatenating the intermediate feature maps of one layer with the up-sampled feature maps of the next layer. Thus, combining the good spatial resolution of the earliest layers with the higher semantic level in the features of the latest. This enables to include more complex features in spatially more accurate maps, as well as considering small objects that would not have been detected with the feature maps of the latest layer only.

This FPN module takes as input the feature maps of each layer of the backbone, in our case ResNet34. Thus, it gets feature maps with different numbers of channels and dimensions. The first step is to convolve each of these inputs with 256 kernels of size (*feature_ maps_ in* x 1 x 1) in order to get for each backbone layer the same amount of channels, while keeping their original dimensions.

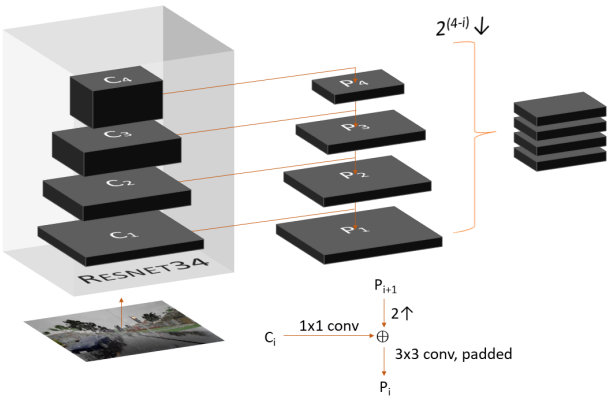

Fig. 3: Illustration of the proposed Feature Pyramid Network architecture.

To be able to integrate it to the existing student network without needing to change it, we propose to add a down-sampling step so that each set of feature maps has the same dimensions as the last backbone convolutional layer. Finally, we concatenate them to end up with a unique output of the same dimensions as PPM. The resulting architecture is illustrated in Figure 3.

By fusing together high semantic - low resolution features of the latest layers with low semantic - high resolution of the earliest layers, a more accurate representation of the image is constructed, which should help the waypoint prediction heads in their task.

3.2 PID Controller

The controller responsible for transforming waypoints to vehicle commands is the same as the one used for LBC and CBS. The target velocity is the average speed needed to go from one waypoint to the next one. Given the current speed

of the vehicle, a longitudinal PID controller must then compute the throttle and brake commands that shall make it reach its target speed. The steer command is computed by fitting an arc through the predicted waypoints. The angle formed by the current position and the projection of one of the waypoints on the arc is fed to a lateral PID controller.

Compared to LBC and CBS, the braking threshold is incremented from 2.0m/s to 3.8m/s in order to adapt the controller to the sensitivity our model has to obstacles. Moreover, the target speed is clipped to be in a range from 0m/s to 5m/s as in CBS.

4 Experimental setup

The approach having been described, now we present the setup and dataset used to train our networks, as well as the evaluation process used to measure the performances of the different models. We compare two different modules that can be added to the existing student network architecture, in order to favour a better recognition of traffic lights. The process remains end-to-end and no classification is made. Both modules can be introduced between the backbone and the waypoint prediction heads. The architecture and code used for the study is described in more detail in the thesis [11].

4.1 Simulation environment

We used CARLA 0.9.10.1 which allows to have reproducible runs with a fixed random seed responsible for traffic generation and an evaluator that is now used by the latest state-of-the-art implementations, allowing us to compare performances with them on common metrics. Note that an earlier version 0.9.6 of CARLA was employed to evaluate CBS algorithm (with a non-fixed random seed), which means that some of those results had to be recreated in CARLA 0.9.10.1 to get a fair comparison.

4.2 Dataset

The dataset is collected using a collector agent which follows the approach of WoR, but has been collected especially for this work. It receives a route through a given CARLA town and receives rewards for attaining intermediate waypoints, for breaking at red lights as well as for pedestrians and cars. Information about the agent's surroundings is based on the semantically segmented BEV retrieved from the simulator. For example, attributing a non-zero reward for braking for a pedestrian is possible only if the latter is visible in the BEV. The BEV is centered on the ego-vehicle and has a range of 32 meters in each cardinal direction. The collected dataset is then split into two distinct datasets used respectively for training and validation, with a 80%-20% ratio.

4.3 Evaluation

Our models are evaluated on the *NoCrash* benchmark, in which a run is stopped as soon as the agent collides. It is also the case if it deviates by more than 30 meters from the route or if it does not move for 180 seconds (timeout). *NoCrash* assesses the performances of the agent in train (*Town01*) and test (*Town02*) environments, as well as train (#1, #3, #6, #8) and test (#10, #14) weather conditions. For all of these combinations, the agent is always evaluated in three different traffic conditions: *Empty*, *Regular* and *Dense*, detailed in Table 1.

Config.	**Vehicles**	**Pedestrians**	**Config.**	**Vehicles**	**Pedestrians**
T1-Empty	0	0	T2-Empty	0	0
T1-Regular	20	50	T2-Regular	15	50
T1-Dense	100	250	T2-Dense	70	150

Table 1: Number of vehicles and pedestrians under the three traffic conditions, *Empty*, *Regular* and *Dense*, in Town01 (left) and Town02 (right).

There are 25 predefined routes in each town that are evaluated in every traffic and weather conditions. Each of these runs is called an episode. For instance, in the test town with test weather conditions, there are 25 routes x 4 weathers x 3 traffics resulting in 300 episodes. Therefore, with three algorithms to be evaluated, the complete benchmark consists in total in 900 episodes.

We compare the performances not on the *Success Rate*, the metric used for WoR, but zoom in on underlying metrics like the *Route Completion*, *In Lane Rate*, *No Collision Rate*, *No Block Rate* and finally the metric which is the focus of the study: *Lights ran per hour*. Note that on average in each episode (depending on the route) two traffic lights are encountered. Those performance measures are compared with WoR, the leading implementation on *NoCrash* and the former state-of-the-art, LBC.

5 Results

For the results, first the learning process is described and followed by the evaluation of the selected models on the *NoCrash* benchmark.

5.1 Learning curves

Figure 4 shows the training and validation curves of all the models. The teacher's 50th epoch checkpoint (loss of respectively 0.0028 and 0.015) is chosen to supervise the student network. Training it for longer lead to an increased validation loss that signify an overfit of the training dataset. Three student models are trained with the supervision of the teacher previously obtained. One without additional module (Original), one with a PPM, and one relying on a FPN. The chosen model is the one with the lowest validation loss after 100 epochs at

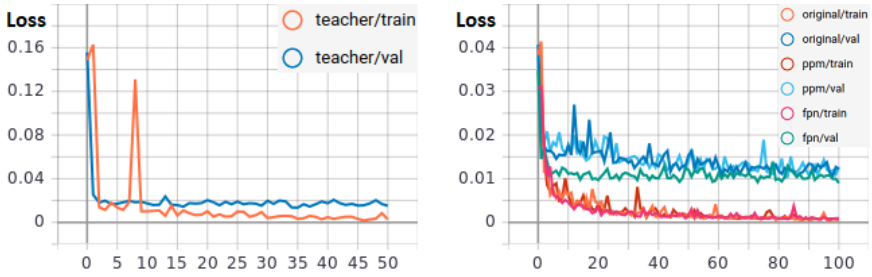

Fig. 4: Evolution of the training and validation loss of the teacher (left) and of the three students (right) with the number of epochs. The loss function is the mean L2 distance between predicted and supervision waypoints.

most. Their validation loss is similar, which, although it is not increasing, stays quite constant and maintain a non-negligeable gap with the training curve. This could indicate a small overfit of the dataset. To overcome this in the future, the regularization could be strengthened for instance by performing more data augmentation or by introducing a weight decay in the optimizer.

5.2 *NoCrash* results

We evaluate the three student models selected in the previous section on *NoCrash* and compare them with the results of WoR and LBC as published in [2], if available. For a fair comparison, the results for CBS marked as 'Original', are the results of the original CBS algorithm trained in the exact same circumstances as FPN and PPM, so these are not based on the previous reported results.

Overall results As shown in Figure 5 (left), the *Route Completion* results are still well below the state-of-the-art. This is not due to the vehicle leaving the road or even its lane. Our three student models have actually learned to drive and to follow a route through the city. The performances on lane keeping are excellent in every configuration, as illustrated in Figure 5 (right).

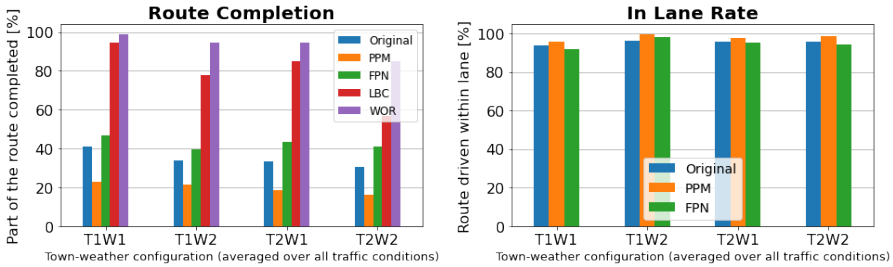

Fig. 5: Left: Percentage of the route completed without collision. Right: Percentage of the route driven within the lane. Metrics are measured according to the town-weather configuration (1:train, 2:test).

Error sources When staying within the lane, still three error sources for not completing the route in the *NoCrash* benchmark remain. The agent can collide with cars and pedestrians in their lane (*No Collision Rate*), stop due to a "false obstacle" or fail to restart when a traffic light becomes green (*No Block Rate*) or, finally, run a red light (*Lights ran per hour*). This latter was a major error source for CBS, but with PPM and FPN this is no longer the case, as demonstrated at the end of this section.

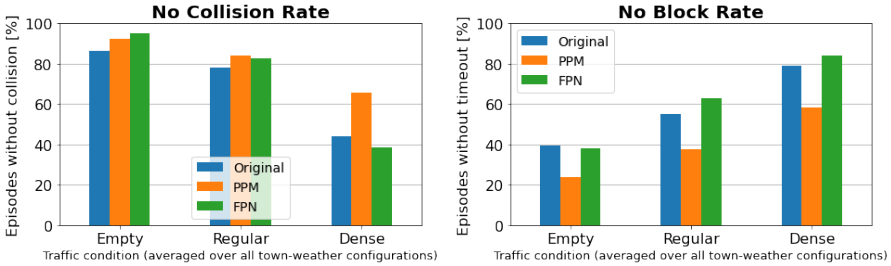

Fig. 6: Left: Percentage of episodes ended without collision according to the traffic condition. Right: Percentage of episodes ended without being blocked. This is defined to be the case when the vehicle does not move for 180 seconds.

Unfortunately, the *No Collision Rate* and *No Block Rate* are hard to balance. The student models' sensitivity to other vehicles is moderate. Other vehicles are detected, because the network reduces its target speed, but too often that is not sufficiently to avoid a collision. This can be repaired by adjusting the braking threshold of the PID controller, described in Section 3.2. Yet, this also influence the risk of getting stuck due to a "false obstacle" such as a puddle of water. This trade-off can be clearly seen in Figure 6.

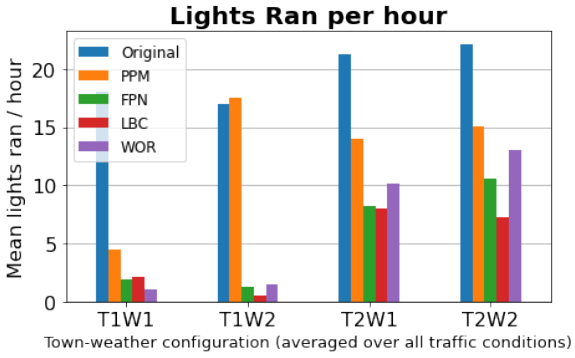

Fig. 7: Traffic light ran per hour of driving according to the town and weather.

We can observe that the agent is less likely to be blocked in heavy traffic, as the resulting change in the visible environment is the only reason for it to make a change in its waypoint predictions, thereby giving it a chance to move. Moreover, it is clear that the PPM model is more often stuck. This could be explained by the fact that it is more capable than the others to differentiate objects from the road even if they have a similar texture. Nonetheless, this higher sensitivity also has the effect that PPM is better at perceiving cars. This is illustrated on the left of Figure 6, where the percentage of episodes ended without collision is depicted.

Correct behavior The original goal and the positive result of this study is that, with the FPN algorithm, the traffic light infractions are no longer the major source of error. The major improvement against the original CBS algorithm can be clearly seen in Table 2 and summarized in Figure 7.

	Train town						Test town					
	Train weather			Test weather			Train weather			Test weather		
	Empty	Reg	Dense	Empty	Reg	Dense	Empty	Reg	Dense	Empty	Reg	Dense
LBC	1.35	1.89	3.27	0.36	0.81	**0.52**	**8.45**	8.22	**7.26**	**8.17**	**8.61**	**4.87**
WoR	**0.00**	**0.43**	**2.61**	**0.00**	**0.00**	4.29	10.68	**6.95**	12.90	14.46	11.30	13.28
Original	15.07	18.45	20.7	16.74	18.06	16.1	19.27	20.1	24.34	25.23	24.94	16.28
PPM	3.37	5.71	4.37	14.9	16.66	20.9	17.84	13.87	10.12	**2.53***	24.62	18.05
FPN	**1.15**	**2.12**	**2.51**	**0.49**	**1.65**	**1.57**	**6.70**	**9.13**	**8.69**	7.74	**10.70**	**13.27**

Table 2: Number of traffic lights infractions per hour

FPN provides a robust traffic light handling, competitive to WoR and close to LBC[3]. Fusing feature maps from different layers of the backbone enables it to recognize more precisely where and which features in the image are responsible for stopping or restarting. Also, it generalizes well to unseen environments, with a number of infractions per hour of only 10.57 in the test town - test weather configuration. Besides, it seems that the weather conditions have a very limited impact on its performances.

6 Conclusion

In this study, we trained an agent end-to-end to imitate an expert using only images taken from a driver perspective. Two different student network architectures were tested. Both the Pyramid Pooling Module and the Feature Pyramid Network resulted in an agent presenting a very good lane keeping. PPM enabled to react more strongly to cars but also to inoffensive obstacles.

[3] Note: the infraction score of the PPM agent in the *Empty* scenario of the test town and weather (marked with *) is underestimated due to it being significantly more stuck (72% of the episodes) compared to the other traffic scenarios.

FPN enabled to significantly reduce the number of traffic lights ran, reaching a number of infractions per hour of 10.6% in test conditions on *NoCrash*. This brings it close to the state-of-the-art implementations *Learning to drive from a world on rails* and *Learning by Cheating*. Diversifying the distance at which the expert stops for a traffic light could further improve the model.

This study has shown insights on the potential of pyramid perception modules for conditional imitation learning and hopefully their perception abilities will contribute to the improvement of autonomous driving.

References

1. 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 (2016)
2. Chen, D., Koltun, V., Krähenbühl, P.: Learning to drive from a world on rails. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021)
3. Chen, D., Zhou, B., Koltun, V., Krähenbühl, P.: Learning by cheating. Conference on Robot Learning (CoRL) (2019)
4. Chi, L., Mu, Y.: Deep steering: Learning end-to-end driving model from spatial and temporal visual cues. preprint arXiv:1708.03798 (2017), https://arxiv.org/abs/1708.03798
5. Codevilla, F., Santana, E., López, A.M., Gaidon, A.: Exploring the limitations of behavior cloning for autonomous driving. Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2019)
6. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. Proceedings of the 1st Annual Conference on Robot Learning (PMLR) (2017)
7. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
8. van Orden, T.: Cheating by segmentation. Bachelor thesis, Universiteit van Amsterdam (June 2021)
9. Pan, Y., Cheng, C.A., Saigol, K., Lee, K., Yan, X., Theodorou, E.A., Boots, B.: Imitation learning for agile autonomous driving. The International Journal of Robotics Research **39**(2-3), 286–302 (2020). https://doi.org/10.1177/0278364919880273
10. Tampuu, A., Matiisen, T., Semikin, M., Fishman, D., Muhammad, N.: A survey of end-to-end driving: Architectures and training methods. IEEE Transactions on Neural Networks and Learning Systems (TNNLS) (2020)
11. Wildi, M.: Conditional imitation learning with pyramid perception modules. Master's thesis, École Polytechnique Fédérale de Lausanne (January 2022)
12. Xu, H., Gao, Y., Yu, F., Darrell, T.: End-to-end learning of driving models from large-scale video datasets. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
13. Yurtsever, E., Lambert, J., Carballo, A., Takeda, K.: A survey of autonomous driving: Common practices and emerging technologies. IEEE Access **8**, 58443–58469 (2020). https://doi.org/10.1109/ACCESS.2020.2983149
14. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)