Using Semantic Information to Improve Generalization of Reinforcement Learning Policies for Autonomous Driving

Florence Carton
Université Paris-Saclay
CEA, List, F-91120,
Palaiseau, France
florence.carton@cea.fr

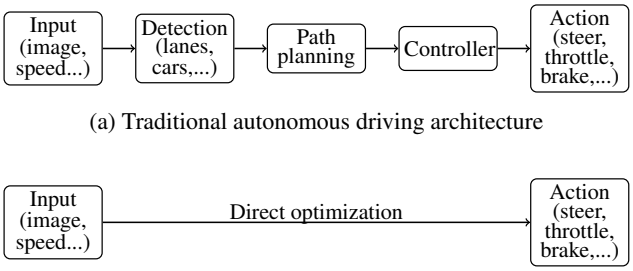
David Filliat
U2IS, ENSTA Paris
INRIA FLOWERS
Institut Polytechnique de Paris
Palaiseau, France
david.filliat@ensta.fr

Jaonary Rabarisoa
Université Paris-Saclay
CEA, List, F-91120,
Palaiseau, France
jaonary.rabarisoa@cea.fr

Quoc Cuong Pham
Université Paris-Saclay
CEA, List, F-91120,
Palaiseau, France
quoc-cuong.pham@cea.fr

Abstract

The problem of generalization of reinforcement learning policies to new environments is seldom addressed but essential in practical applications. We focus on this problem in an autonomous driving context using the CARLA simulator and first show that semantic information is the key to a good generalization for this task. We then explore and compare different ways to exploit semantic information at training time in order to improve generalization in an unseen environment without finetuning, showing that using semantic segmentation as an auxiliary task is the most efficient approach.



(a) Traditional autonomous driving architecture

(b) End-to-end autonomous driving architecture

Figure 1: Traditional vs end-to-end learning

1. Introduction

Since ALVINN [23] in the 90s, autonomous driving based on visual input has seen significant advances over the past few years. Traditionally, two approaches can be distinguished (see Figure 1). The first one consists of a modular pipeline, which breaks down driving into intermediate tasks, such as perception, path planning, and control. This allows monitoring all stages of the decision-making process but needs individual optimization for every subsystem, making it difficult to find an overall optimal solution. The second approach that has recently become popular is end-to-end driving, where the system directly maps raw input, such as road images, to driving controls. The advantage of end-to-end learning is the reduction of intermediate operation, especially as these are mostly designed for human drivers. It can reduce time and complexity not only during training, but also in later application, as no extra features are computed. Besides, no human bias is added.

Many of the end-to-end autonomous driving approaches rely on imitation learning [2, 23, 6, 25], where the system aims at imitating expert behavior. Although relatively stable, it requires a large amount of annotated data and suffers from distributional shift. Indeed the system sees only good behaviors, learns to imitate them, and will never learn how to react in problematic situations, which are rare or even absent from the dataset. Some extend their dataset to add erroneous trajectories (drone crashes in [10], or extra camera to simulate a come off the road in [2]), but these are specific cases and it seems impossible to predict all error cases.

Reinforcement learning on the other hand, makes the

agent learn by trial and errors, only sending a reward signal to indicate good or bad behavior. The agent experiments by itself and thus learns how to act with less human intervention than imitation learning, and also sees a lot more of error cases. With the success of Alpha Go in 2016 [29], reinforcement learning has experienced huge success in machine learning research, but only few approaches are using it for autonomous driving task [22, 30, 14, 16]. Reinforcement learning has indeed several drawbacks such as long training time, poor data efficiency and instability. However in several domains, such as Go, reinforcement learning has proven to outperform supervised learning, and recent work has established that learning to drive using reinforcement learning could achieve high performance [30, 16], even if most of the approaches are still in simulation, for example using the CARLA driving simulator [9].

Yet very often in reinforcement learning framework, the test environment is the same as the training environment,[5], which leaves aside the problem of overfitting and generalization. However in the case of autonomous driving, generalization is essential. We need autonomous cars to handle unseen situations, from unseen towns to different driving conditions.

In this paper, we explore the relative interest of various ways to increase the generalization performance of a driving agent having learned with reinforcement learning. In a first part, we show that semantic segmentation contains almost all the necessary information, except for traffic signs and traffic light state, and that using semantic segmentation as input (instead of RGB image) is enough to achieve perfect score in generalization on the CARLA CoRL benchmark [6]. This shows that the main generalization problem is to learn to correctly segment the target environment, and not to adapt the decision part going from the semantic segmentation to the controls.

In a second part, we therefore explore how to use semantic information in the training environment. We will then measure the improvement of generalization in an unfamiliar environment of an agent using input images with reinforcement learning. To explore properly the different options, we chose to simplify the problem, and deal with urban driving without obstacles in a first step.

2. Related Work

2.1. Autonomous Driving with Reinforcement Learning and Generalization

Reinforcement learning (RL) consists in learning by trial and error. For obvious safety reasons, most of the work in autonomous driving with reinforcement learning is conducted in simulation. The two most widely used simulators in the literature are TORCS [31] and CARLA [9], and the latter is increasingly used since TORCS is a racing game and CARLA does offer various urban environments.

When introducing CARLA driving simulator [9], the authors compared the performance on goal-directed driving task between supervised and reinforcement learning. Their baseline in reinforcement learning, trained with A3C, was much worse than imitation learning. A more recent approach called CIRL combines both supervised learning and reinforcement learning by pretraining the network with supervised learning [16]. Although they use classical training approaches such as data augmentation, the unseen town's performance remains a lot lower than the ones in training conditions.

Even if the number of work on autonomous driving keeps growing, only a handful concentrates on autonomous driving with reinforcement learning. Kendall et al. in [14] are among the first to tackle this problem. Their agent learns to perform lane following in both simulation and real world. A few data augmentation is performed in the simulation experiment (road texture, lane marking, and road topology), but test roads are used as validation to stop training. In *Virtual to Real Reinforcement Learning for Autonomous Driving*[20], Pan et al. propose a framework to transfer a virtually trained agent to real world. They use a translation network to convert virtual frame to realistic ones, and the driving agent is trained on the realistic images, which allows a smooth transfer to real world driving. Both [14] and [20] agents drive in real world, but they only deal with the lane following task, whereas generalization seems to be more critical in navigation task. The performance achieved by CIRL [16] when it comes to going straight ahead is identical in training and test town, but a huge drop in performance can be observed in the navigation task (-40%).

Recent work [30] showed significant improvement on autonomous driving with reinforcement learning, training a network with affordances (like semantic segmentation, or traffic light state), and then use these affordances to train a reinforcement learning agent. An ablation study is made to compare the performance on unseen town when using one or several training town(s). Unsurprisingly, performance increase with the addition of data, but the highest performance achieved in an unseen town reaches 58.4% of successful driving episodes. Therefore, even with the high performance of their agent in the CARLA challenge, we argue that generalization is still an open problem for autonomous driving with reinforcement learning. Moreover, fixing the backbone network comes down to separately training different sub-system (one for perception, one for driving), and gets a little bit closer to a modular pipeline approach, where each subsystem needs to be optimized separately.

Another approach with reinforcement learning reaches high score on the CoRL benchmark : *Learning to drive using waypoints* [1]. It generalizes well on unseen town, but uses bird-view segmented image as well as waypoints as

input for the RL algorithms, which is unrealizable in real world. Perfect score is also achieved by *Learning by cheating* [4], which is an imitation learning approach. A privileged agent is trained with expert trajectories, and has access to the layout of the environment (birdview, other actors positions, etc...). This privileged agent then trains an agent that only has access to raw image input and no other groundtruth information. This framework has shown effective generalization in the unseen town, but privileged information such as birdview or other actors states are again not available in real environment. All these algorithms certainly achieve a high generalization score, however we want to focus on approaches that exploit only visual input for reinforcement learning and are therefore easily transferable to other fields.

In a broader context, generalization is a key element in computer vision. However, as mentioned in *Quantifying Generalization in Reinforcement Learning* [5], it is common to use the same environment for training and test in reinforcement learning benchmarks. Both [15] and [5] study the problem of generalization and the way to apply the classical methods (data augmentation, L2 regularization, dropout...) to reinforcement learning training. They both conclude that classical techniques used in supervised learning can be successfully applied in reinforcement learning with some notable success of data augmentation and batch-normalization.

2.2. Use of Additional Information to Improve Training and Generalization

Learning to drive is a very difficult task. An autonomous vehicle must operate in a dynamic environment, and take into account many external elements to make a decision. Therefore using extra information is a commonly used technique. A very standard practice is to pretrain the neural network, either with the same task (autonomous driving in [9]), or with different task (collision probability in [26] for drone navigation).

To simplify the driving problem, learning affordances, i.e. learning intermediate representation of the input, is becoming more and more used in both supervised and reinforcement learning for autonomous driving [32, 30, 27, 17]. Indeed, using affordances is extracting useful information, and thus reduces the state space size. Nevertheless it requires human intervention, for both choosing the affordances that are necessary, and training the supervised affordances.

To improve generalization, data augmentation is a widely used practice in deep learning. However even if it is unavoidable in supervised learning, it is less the case in reinforcement learning. When comparing the baselines in both imitation and reinforcement learning in autonomous driving, CARLA authors [6] do apply data augmentation and regularization techniques (dropout) during imitation learning training, and not during reinforcement learning.

Last but not least, an auxiliary loss (or multi-task training) was first used in [13] with deep reinforcement learning, and it showed improved performance. However, no study on generalization capabilities is made. The idea is to train several task at the same time, the main one is trained with reinforcement learning, and the auxiliary tasks are trained with supervised or unsupervised learning. Auxiliary loss is also used with reinforcement learning in *Learning to navigate in complex environment* [18] to predict depth to help an agent navigate in a maze.

After showing that the availability of the semantic segmentation information in the target environment essentially solves the generalization problem in autonomous driving, we propose here to compare these methods to exploit this information during training only and analyze their impact on generalization.

3. Methodology

We present here the details of the components used in our experiments where we trained RL agents with image or segmentation as input. Seeing the good results of the latter, we trained a network to predict the segmentation from the image. This network was used for two different purposes: first, to train an RL agent with this learned segmentation, and second, we used the encoder weights as pre-training for an agent using the image as input. Finally, we experimented with segmentation as an auxiliary task, which consists in learning in parallel the segmentation and the driving, and thus building an intermediate representation that would be efficient for both. We also detail our data augmentation.

3.1. RL Algorithm and Network

In reinforcement learning, an agent interacts with an environment. The environment produces a state s_t at each timestep t, and when receiving the current state s_t, the agent reacts with an action a_t. The agent acts according to a policy $\pi(a_t|s_t)$, which represents the probability to take an action a when being in state s.

After the agent has taken the action a_t, the environment provides a new state s_{t+1} alongside a reward r_t, which indicates how good the new state is. The goal of the agent is to maximize the cumulative rewards, the *return*, often using a discount factor γ to avoid exploding sums (eq. 1).

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \quad (1)$$

For this study, we chose the PPO algorithm [28], as it showed good results on classical problems with image input and continuous actions. PPO is an actor-critic reinforcement learning algorithm, where the policy modification is

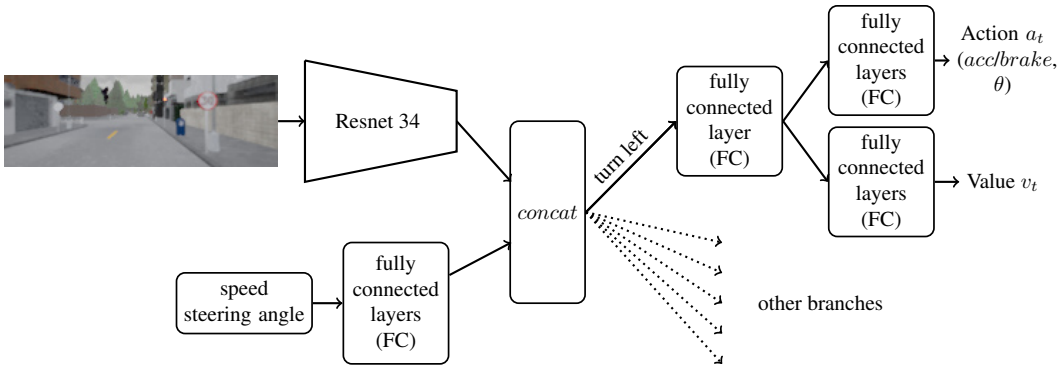


Figure 2: Reinforcement learning architecture with input image

constrained in a trust region. In other words, the step size is forced to stay in a specific range for each policy update to improve stability and reliability. To stay as close as possible to human driving, we decided to work with continuous actions. The state is a combination of the current RGB image of the road and some measurement (current speed and angle of the wheels). The agent outputs acceleration/brake, and variation of steering angle. We use a conditional network like [9] to take the driving command in account. This means that the model consists of branches at the output of the CNN. Each branch corresponds to a control command (turn right, follow lane, ...). The branches are composed of fully connected layers (FC).

The backbone network used is a well-studied Resnet34 [11] since it has proven to be effective in the autonomous driving task in [7]. The general architecture is detailed in Figure 2. For the experiment with semantic segmentation as input, the network is a much smaller CNN from [19], which we call NatureCNN, with 3 convolutionnal layers. We use stable baselines [24] implementations for the reinforcement learning algorithm.

Since we use an on-policy algorithm, no replay buffer is used, but to accelerate training, several instances of CARLA simulator run in parallel to collect images. Input images are small ($192 \times 64 \times 3$) to speed up training. Details on the hyperparameters used are presented in the supplementary material.

3.2. Reward Function

Our reward function is the weighted sum of several terms: the difference between current speed and objective speed, the cross-track error cte (i.e. distance to road center), and the angle between the car and the road (in degrees). The objective speed is variable, 35km/h in general, and 15km/h at crossings. There is also a negative reward when the car collides with an obstacle, runs off the track, or stops for no reason.

$$r_t = \begin{cases} r_{speed} + r_{angle} + r_{cte} \\ r_{punish} \text{ when collision, offroad, bad stop.} \end{cases} \quad (2)$$

We modified the reward so that it is mostly positive (it experimentally leads to better results). Details on the reward can be found in supplementary material. The reward is scaled during training using log scaling [12] :

$$r_{agent} = sign(r) log(1 + |r|). \quad (3)$$

Moreover, we use partial episodes bootstrapping from [21]. It consists in modifying the reward for a terminal state if the termination is due to time limit and it leads to a better convergence of the value loss. More precisely, the reward is modified with value of the next state $v(s')$ (weighted with the discount factor γ) when episode termination is due to a timeout :

$$r_{terminal\_state} = \begin{cases} r & \text{at environmental} \\ & \text{termination} \\ r + \gamma v(s') & \text{at timeout.} \end{cases} \quad (4)$$

3.3. Semantic Segmentation Training

The network used for semantic segmentation of RGB images is a Linknet [3] since it is based on Resnet34 we already use for RL. Data for training are collected in training conditions only, in Town01 with training weathers. 140 000 images are used, divided in train and validation set with the ration $0.8/0.2$. Images are collected with three cameras to perform viewpoint augmentation. We used one forward-looking camera, and two with a $\pm 30°$ angle like in *End to End Learning for Self-Driving Cars* [2]. Five categories are trained : road, roadlines, vehicles, sidewalk and background. Loss function is categorical cross-entropy, and optimizer is Adam. The learning rate is 10^{-3}, and we used a batch size of 32.

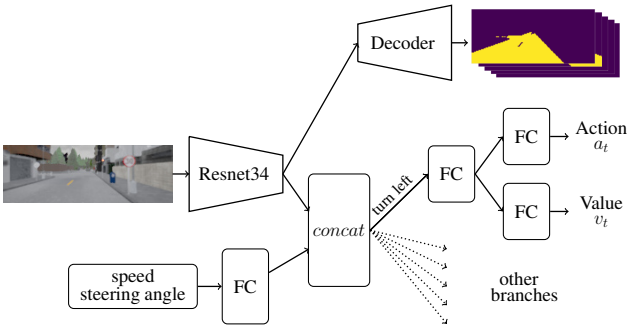


Figure 3: Reinforcement learning architecture with auxiliary task

3.4. Auxiliary Task

Auxiliary tasks consists in training a different task than the main one at the same time in order to improve its performance. In our case, we train semantic segmentation along with driving. The general architecture (see Figure 3), as well as the inputs and outputs of the reinforcement training are identical to the previous one (see Figure 2). The Linknet [3] decoder is added to compute the segmentation of the image. The decoder outputs $S_t \in [0,1]^{W*H*c}$ with c the number of classes to segment. For a given pixel on the original image, the output is the probability to belong to the different classes. The 5 classes are the same as previously : road, roadlines, vehicle, sidewalk and background. One of the challenge is to combine both learnings at the same time, and not having one prevailing on the other. The simplest loss is a weighted sum of the different losses :

$$l = \lambda_{rl} l_{rl} + \lambda_{aux} l_{aux}. \tag{5}$$

In equation 5, l_{rl} is PPO loss, l_{aux} the semantic cross entropy loss, and λ_{rl} and λ_{aux} their respective weights. However, in [14] is described a more efficient loss combination to perform automatic loss balance in multi-task learning. We set σ_{rl} and σ_{aux} two trainable parameters, and the loss function becomes :

$$l = e^{-\sigma_{rl}} \times l_{rl} + \sigma_{rl} + e^{-\sigma_{aux}} \times l_{aux} + \sigma_{aux}. \tag{6}$$

This homoscedastic loss was used for training reinforcement learning with auxiliary task.

3.5. Data Augmentation

Our data augmentation includes random variation in hue and saturation, brightness and contrast, along with conversion to gray scale, partial erasing of the image, salt and pepper noise, and Gaussian blur. This data augmentation is inspired from the one used in the original CARLA paper for the supervised learning training [9]. We also explore a semantic data augmentation, that consists in randomly changing the color of one semantic part of the image (Figure 4). The data augmentation is applied to each image independently, so it changes along an episode. The different data augmentation are combined in a random order, and also in a random way, the most frequent being the variation of contrast, hue and saturation (half of the time), and the least frequent being Gaussian blur and gray-scale (10% of the images).

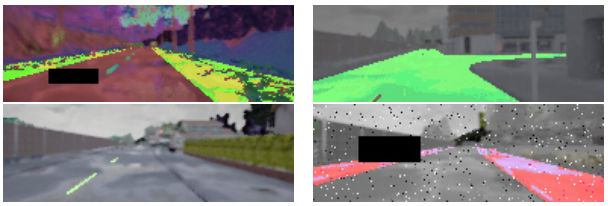


Figure 4: Examples of data augmentation

4. Experiments and Results

We now present our different experiments. Performance is measured using the original CoRL benchmark from [6]. We focus on the navigation task only, and measure the percentage of successful episode in training conditions (training town and training weathers) and in new weathers, new town, and new town and new weathers conditions. Some examples of training and test conditions are shown on figure 5.

The first experiments are made with semantic segmentation as input - instead of the RGB image. We compare the generalization performance using the perfect segmentation generated by CARLA simulator and the segmentation we trained with a Linknet [3]. Then we explore the influence of data augmentation, and notably the data augmentation with semantic information. Finally, we compare the relative influence of pre-training and auxiliary task on the generalization ability of our agent. Training are made up to 20 Millions steps, and we report the best performance for each model.

4.1. Segmentation

Training	Training Conditions	New weather	New town	New Town & weather
Ground truth seg	100%	100%	100%	100%
Learned seg	100%	100%	89%	86%

Table 1: Performances with semantic segmentation as input.

This first experiment shows the importance of the semantic information in generalization. Indeed, training an algo-

Figure 5: Examples of training and test environments. Right : training town and training weathers. Left : test town and test weathers.

rithm with the semantic image as an input achieves perfect training and generalization performance (first line in table 1). This approach is not useful in practice as perfect semantic segmentation is usually not available in the target environments, but shows that the bottleneck in generalization is mainly on the image analysis part, and not in the driving decision that generalizes perfectly.

A natural solution to this problem is to train a semantic segmentation in the training conditions and rely on its generalization capacity to the new environment. However, this approach shows that this is not sufficient to keep the generalization capacity of the previous approach (second line in table 1). It is also probably an upper bound on the generalization performances that could be reached using this reinforcement learning approach.

4.2. Data Augmentation

Training	Training conditions	New weather	New town	New Town & weather
No da	34%	6%	9%	2%
Classic da	57%	60%	22%	4%
Da w/ seg	67%	60%	34%	28%

Table 2: Baseline results for navigation task with different data augmentation

We compare in table 2 the results between training with no data augmentation (no da), and 2 different data augmentations : the first one is classical (classic da), adding noise in the image (gaussian blur, salt and pepper, etc...), and the second one (da w/ seg) includes in addition the semantic component, where a semantic part of the image changes color randomly. We first notice that learning without data augmentation is not efficient at all (we are barely higher than the RL baseline from the original CARLA paper [6]), even under training conditions. Classic data augmentation allows to increase the performance, both in training conditions and in the new city, but does not really reduce the gap between the training city and the new city. If we look closer at the results of training without data augmentation, we notice that with a few exceptions, only the episodes with the easiest weather (ClearNoon) were successful. The addition of noise (classic da) in the input image has therefore made it possible to better learn the different meteorological conditions, even if they are all seen during the training. The addition of segmentation in the data augmentation allows to reduce the generalization gap a bit, even if it is still important. Highlighting the semantic elements of the image allows the agent to make the link with the reward function, and thus to better generalize. However, even if progress is made compared to the training without data augmentation, the gap between training town and test town is still huge.

4.3. Pretraining and Auxiliary Task

Training	Training conditions	New weather	New town	New Town & weather
Pretraining	82%	**98%**	49%	40%
Auxiliary task	**90%**	92%	**78%**	**68%**

Table 3: Pretraining and auxiliary task performance

In table 3, we compare the relative effects of pre-training and auxiliary task. Concerning the pre-training, we trained a Linknet to segment the image, and the weights of the Resnet34 were initialized with it. The network is then trained for the driving task in the same way as before. We notice an improvement of the results, both on the training conditions and on the generalization to the new city. However, the pre-training does not reduce the large gap that still exists between the train and the test. It simply improves performances. The auxiliary task seems more promising, as it significantly reduces the gap between the training city and the test city. Moreover, contrary to [8] for example, we noticed that training auxiliary task simultaneously tends to stabilize the training.

5. Conclusion and Future Work

We analyzed the influence on performance, and in particular on the generalization, of different methods based on semantic segmentation. Indeed, semantic segmentation allows an agent to learn how to drive and to generalize to roads it has not seen. However, basing oneself solely on the generalization capacity of segmentation means separating perception and control, which is not in the spirit of end to end driving. We have therefore experimented with different ways of adding this segmentation information: in data

augmentation, through pre-training, and finally with an auxiliary task. We have shown that it is only the latter method that reduces the gap between the training conditions and the test environments. Pre-training or applying a rigorously chosen data augmentation allows us to improve the performance of our agent in general, but does not solve the problem of generalization. Learning at the same time the image segmentation, while sharing a part of the network, has on the other hand improved generalization.

	Training Condition	New Weather	New Town	New Town & Weather
RL [9]	14%	2%	3%	6%
CIRL [16]	**93%**	86%	53%	68%
Auxiliary task	90%	**92%**	**78%**	**68%**

Table 4: CoRL Benchmark and comparison with SOTA on the task navigation

To conclude, table 4 compares our results with the RL baseline from the original CARLA paper [9], and CIRL [16]. Our training with auxiliary task has outperformed the original baseline by far, and is competitive with CIRL, knowing that CIRL has been trained with both supervision and reinforcement. Our approach even outperforms CIRL in terms of generalization to unseen environment, but needs however longer training, since no supervised learning for driving is used. In a future work, we plan to improve the performance of the auxiliary task by studying the effect of the architecture (How many blocks should be shared between the auxiliary task and the driving task? Should we use the Linknet skip connections?) and more complex data augmentation.

References

[1] Tanmay Agarwal, Hitesh Arora, Tanvir Parhar, Shubhankar Deshpande, and Jeff Schneider. Learning to Drive using Waypoints. (NeurIPS), 2019.

[2] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to End Learning for Self-Driving Cars. pages 1–9, 2016.

[3] Abhishek Chaurasia and Eugenio Culurciello. LinkNet: Exploiting encoder representations for efficient semantic segmentation. *2017 IEEE Visual Communications and Image Processing, VCIP 2017*, 2018-Janua:1–4, 2018.

[4] Dian Chen, Zhou Brady, Koltun Vladlen, and Krähenbühl Hilipp. Learning by Cheating. *CoRL*, (CoRL):1–10, 2019.

[5] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:2280–2293, 2019.

[6] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end Driving via Conditional Imitation Learning. *Proceedings ICRA, 2018*, 2017.

[7] Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. Exploring the Limitations of Behavior Cloning for Autonomous Driving. (Cvc), 2019.

[8] Tim De Bruin, Jens Kober, Karl Tuyls, and Robert Babuska. Integrating State Representation Learning into Deep Reinforcement Learning. *IEEE Robotics and Automation Letters*, 3(3):1394–1401, 2018.

[9] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An Open Urban Driving Simulator. (CoRL):1–16, 2017.

[10] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. *IEEE International Conference on Intelligent Robots and Systems*, 2017-Septe:3948–3955, 2017.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:770–778, 2016.

[12] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep Q-learning from Demonstrations. 2017.

[13] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement Learning with Unsupervised Auxiliary Tasks. pages 1–14, 2016.

[14] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to Drive in a Day. 2018.

[15] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement Learning with Augmented Data. 2020.

[16] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric Xing. CIRL: Controllable Imitative Reinforcement Learning for Vision-based Self-driving. 2018.

[17] Ashish Mehta, Adithya Subramanian, and Anbumani Subramanian. Learning End-to-end Autonomous Driving using Guided Auxiliary Supervision. 2018.

[18] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to Navigate in Complex Environments. 2016.

[19] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–33, 2015.

[20] Xinlei Pan, Yurong You, Ziyan Wang, and Cewu Lu. Virtual to Real Reinforcement Learning for Autonomous Driving. 2017.

[21] Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time Limits in Reinforcement Learning. 2017.

[22] Etienne Perot, Maximilian Jaritz, Marin Toromanoff, and Raoul De Charette. End-to-End Driving in a Realistic Racing Game with Deep Reinforcement Learning. *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017-July:474–475, 2017.

[23] Dean A. Pomerleau. ALVINN: an autonomous land vehicle in a neural network. pages 305–313, 1989.

[24] Antonin Raffin and Ashley Hill. Stable baselines. `https://stable-baselines.readthedocs.io/en/master/`.

[25] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep Imitative Models for Flexible Inference, Planning, and Control. pages 1–12, 2018.

[26] Fereshteh Sadeghi and Sergey Levine. CAD2RL: Real Single-Image Flight without a Single Real Image. 2016.

[27] Axel Sauer, Nikolay Savinov, and Andreas Geiger. Conditional Affordance Learning for Driving in Urban Environments. (CoRL):1–16, 2018.

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. pages 1–12, 2017.

[29] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–9, 2016.

[30] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. End-to-End Model-Free Reinforcement Learning for Urban Driving using Implicit Affordances. (Il), 2019.

[31] Bernhard Wymann, Eric Espié, Christophe Guionneau, Dimitrakakis Christos, Coulom Rémi, and Sumner Andrew. TORCS: The Open Racing Car Simulator. pages 1–5, 2014.

[32] Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. *IEEE International Conference on Intelligent Robots and Systems*, 2017-Septe:2371–2378, 2017.

A. Supplementary Material

A.1. Reward Details

Here we present the reward function used in more detail. It is an additive reward function :

$$r_t = \begin{cases} r_{speed} + r_{angle} + r_{cte} \\ r_{punish} \text{ when collision, offroad, bad stop.} \end{cases} \quad (7)$$

The weights for every component were determined experimentally, and constants are used to make the reward mostly positive (for instance, when the car has a cross-track error that is less than one meter, r_{cte} will remain positive).

$$r_{speed} = w_{speed}(s_{obj} - |s_{obj} - s_t|) = 1.0 \times (s_{obj} - |s_{obj} - s_t|), \quad (8)$$

with s_t the current speed in km/h,, s_{obj} the objective speed, set at 35km/h (15 at intersections)

$$r_{angle} = w_{angle}(m_a - a_t) = 0.1 \times (15 - a_t), \quad (9)$$

with a_t the current angle of the car with the road in degrees

$$r_{cte} = w_{cte}(m_{cte} - cte_t) = 10 \times (1 - cte_t), \quad (10)$$

with cte_t the current cross-track error (ie the distance in meters from the road center) and

$$r_{punish} = -100. \quad (11)$$

A.2. Hyperparameters for Reinforcement Learning Training

In reinforcement learning, hyperparameters tuning is crucial. We go through all the hyperparameters that are used during RL training, using notation from [28] : ϵ is the clipping parameter, γ is the discount factor, c_1 is the value function loss weight, N is the number of environments, M the number of minibatch used for backpropagation, and T the number of collected data per environment :

hyperparameter	value
ϵ	0.2
γ	0.99
c_1	0.5
N	16
M	16
T	128

The standard deviation (stdev) of action distribution is also set (i.e. not trainable) in a such way that the log(stdev) is linearly decreasing over the training. We also set an exponentially decreasing learning rate (from 5×10^{-4} to 5×10^{-8} over training)