Supplementary materials: End-to-End Model-Free Reinforcement Learning for Urban Driving using Implicit Affordances

Marin Toromanoff[1,2,3], Emilie Wirbel[2,3], Fabien Moutarde[1]

[1]Center for Robotics, MINES ParisTech, PSL [2]Valeo Driving Assistance Research [3]Valeo.ai

[1]name.surname@mines-paristech.fr [2,3]name.surname@valeo.com

1. Implementation details

In this section, we will detail the hyper-parameters and the architecture of both the Supervised and the Reinforcement Learning training.

1.1. Supervised phase of affordances training: architecture and hyper-parameters

Our encoder architecture is mainly based on Resnet-18 [5] with two main differences. First, we changed the first convolutional layer to take 12 channels as input (we stack 4 RGB frames). Secondly, we changed the kernel size of downsample convolutional layers from 1x1 to 2x2. Indeed as mentionned in the paper Enet [7], *When downsampling, the first 1x1 projection of the convolutional branch is performed with a stride of 2 in both dimensions, which effectively discards 75% of the input. Increasing the filter size to 2x2 allows to take the full input into consideration, and thus improves the information flow and accuracy.*. We also removed the two last layers: the average pooling layer and the last fully connected. Finally, we added a last downsample layer taking 512x7x7 feature maps as input and outputting our RL state of size 512x4x4.

For the loss computation, we add a weight of 10 for the part of the loss around traffic light state detection, and 1 for all other losses.

Table 1. Supervised training hyperparameters

Parameter	Value
Learning rate	5.10^{-5}, eps 3.10^{-4} (Adam)
Batchsize	32
Epochs	20

For the semantic decoder, each layer consists of an upsample layer with a nearest neighbor interpolation, then 2 convolutional layers with batchnorm. All the other losses are build with fully connected layers with one hidden layer of size 1024. See Table 1 for more details on other hyperparameters used in the supervised phase.

To train our encoder, we used a dataset of around 1M frames with associated ground-truth label (e.g. semantic segmentation, traffic light state and distance). This dataset was collected mainly in 2 cities of the CARLA [4] simulator: Town05 (US) and Town02 (EU).

1.2. Reinforcement Learning phase: architecture and hyper-parameters

In all our RL trainings, we used our encoder trained on affordances learning as a frozen image encoder: the actual RL state is the 8162 features coming from this frozen encoder. We then give this state to one fully connected layer of size 8162x1024. Then from these 1024 features concatenated with the 4 previous speed and steering angle values, we use a gated network to handle different orders as presented in CIL [2]. All the 6 heads have the same architecture but different weights, they are all made with 2 fully connected layers with one hidden layer of size 512.

Table 2. RL training hyperparameters for our *Single Town* and *Multi-Town* experiments: all parameters not mentioned come from the open-source implementation of Rainbow-IQN [10].

Parameter	Single Town / Multi-Town
Learning rate	5.10^{-5}, eps 3.10^{-4} (Radam)
Batchsize	32
Memory capacity	90 000 / 450 000
Number actors	3 / 9
Number steps	20M (23 days) / 50M (57 days)
Synchro. actors/learner	Yes / No

All hyperparameters used in our Rainbow-IQN training are the same as the one used in the open-source implementation [10] but for the replay memory size and for the optimiser. We use the really recent Radam [6] optimiser as it is giving consistent improvement on standard supervised training. Some comparisons were made with the Adam optimiser but did not show any significant difference. For all our *Single Town* experiments, we used Town05 (US) as environment. For our *Multi-Town* training, we used Town02

(EU), Town04 (US) and Town05 (US). Table 2 details the hyper-parameters used in our RL training.

2. Experiments

2.1. Stability study

One RL training of 20M steps was taking more than one week on a Nvidia 1080 Ti. That is why we did not have time nor computational resources to run an extensive study on the stability for all our experiments. Moreover evaluating our saved snapshot was also taking time, around 2 days to evaluate performance each million of steps as in Figure 7 of the main paper. Still, we performed multiple runs for 3 experiments presented in Table 1: *No TL state*, *No segmentation* and *All Affordances*. We evaluated those seeds at 10M and at 20M steps and the results (mean and standard deviation) can be found in the following Table 3.

Encoder used	10M steps		20M steps	
	Inters.	Nb seeds	Inters.	Nb seeds
No TL state	17.9% $\pm$ 7.3	6	27% $\pm$ 5.7	5
No segmentation	27.7% $\pm$ 9.3	5	41.7% $\pm$ 0.1	2
All affordances	24.9% $\pm$ 8.2	6	64.4% $\pm$ 2.5	2

Table 3. Mean and standard deviation of agents performance with regards to encoder training loss (trained without traffic light loss, without semantic segmentation loss, or with all affordance losses)

Even if we just have few different runs, those experiments on stability support the fact that our training are roughly stable and our results are significant. At 20M steps the "best" seed of *No TL state* perform worse than both seeds of *No segmentation*. More importantly, both seeds of *No segmentation* perform way worse than both seeds of *All affordances*.

2.2. Additional experiments

We made one experiment, *4 input one output*, to know the impact of predicting only one semantic segmentation instead of predicting 4 at the same time. Indeed, we stack 4 frames as our input and we thought it would give more information to learn from, if we train using all 4 semantic segmentations. We also tried to remove temporality in the input: taking only one frame as input and thus predicting only one semantic segmentation, *One input one output*. Finally, we made an experiment, *U-net Skip connection*, on which we used a standard U-net like architecture [8] for the semantic prediction. Indeed we did not use skip connections in all our experiments to prevent the semantic information to flow in this skip connections. Our intuition was that the semantic information could not be present in our final RL state (the last features maps of 4x4) if using skip connections.

The results of this 3 experiments are described in Table 4.

Encoder used	Inters.	TL	Ped.
One input one output	29.6%	95%	85%
4 input one output	64.3%	93.8%	70.7%
U-net Skip connection	58.6%	95%	69.8%
All affordances	64.4%	98.1%	76.2%

Table 4. Additional experiments to study impact of temporality both as input and as output of our Supervised phase. Also experiments with skip connection for the semantic prediction (U-net like skip connection [8]).

We can see from this results that using only one frame as input has a large impact on the final performance (going from 64% intersections crossed with our standard scheme *All Affordances* to 29% when using only one image as input). The impact of predicting only one semantic segmentation instead of 4 is marginal on our main metric (*Inters.*) but we can see that the performance on traffic lights (*TL*) and on pedestrians (*Ped.*) are slightly lower. Finally, the impact of using U-net like skip connections seems to be relatively small on the number of intersection crossed. However, there is still a difference with our normal system particularly on the pedestrians metric.

As a conclusion, those additional experiments confirmed our intuitions first about adding temporality both as input and output of our encoder and secondly to not use standard U-net skip connection is our semantic segmentation decoder to prevent semantic information to flow away from our final RL state. However, the impact of those intuitions are relatively small and we conducted only one seed which could not be representative enough.

2.3. Description of our test scenario

Each of our scenario is defined by a starting waypoint and 10 orders one for each intersection to cross. An example of one of our 10 scenario can be found on Figure 1. We also spawn 50 vehicles in the whole Town05 while testing. Finally, we spawn randomly pedestrian ahead of the agent every 20/30 seconds.

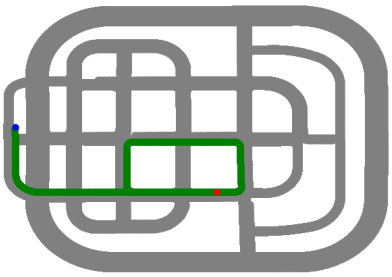

Figure 1. Sample of one of our scenario in Town05. The blue point is the starting point, the red is the destination.

2.4. Comparison on CARLA Benchmark: Implementation Details and Test Weathers Results

2.4.1 Test weathers results (train and test town)

As mentioned in the main paper, we did not have time to re-implement our training setup for the really recently released [1] implementation of the CARLA benchmark on the newer version of CARLA (0.9.6), particularly regarding the weather condition. At submission time, all our training were done under all possible weather conditions. That's why we reported our results only for training weathers in the main paper. We only had time to train our whole pipeline in the exact condition of the CARLA benchmark (i.e. only Town01 and train weathers for training and Town02 and test weathers for test) after acceptance. That's why we give our results for test weather only in the Supplementary Materials.

	CoRL2017 (train town)					NoCrash (train town)		
Task	RL	CAL	CILRS	LBC	Ours	Task	LBC	Ours
Straight	86	100	96	100	100	Empty	87	36
One turn	16	96	96	96	100	Regular	87	34
Navigation	2	90	96	100	100	Dense	63	26
Nav. dynamic	2	82	96	96	100			

	CoRL2017 (test town)					NoCrash (test town)		
Task	RL	CAL	CILRS	LBC	Ours	Task	LBC	Ours
Straight	68	94	96	100	100	Empty	70	24
One turn	20	72	92	100	100	Regular	62	34
Navigation	6	88	92	100	100	Dense	39	18
Nav. dynamic	4	64	90	100	100			

Table 5. Success rate comparison (in % for each task and scenario, more is better) with baselines [4, 9, 3, 1] on test weathers.

We can see from Table 5 that we are the only approach reaching a perfect score on all the tasks under test weathers. However, we can see that our results on the *NoCrash* benchmark fall far behind LBC [1] baseline under test weathers (even if our results were similar under train weathers). We found that the test weathers on the *NoCrash* benchmark are actually really different from the train weathers, particularly regarding sun reflection on the ground. We discovered that our frozen encoder trained only on Town01/train weathers was predicting sun reflection as "moving obstacles" and thus in this situation the RL agent is just braking for ever, acting like if a car was ahead. Most of our failure under test weathers on *NoCrash* benchmark are in fact timeout because our agent is not moving anymore when he faces sun reflection on the ground. Handling diverse weather conditions is a known issue for perception algorithms and we think that improving our supervised performance (particularly the semantic segmentation) would probably manage this issue but this is left as future work.

2.4.2 Implementation Details for the CARLA benchmark

To train our new encoder in the exact condition of the CARLA benchmark, we used a new dataset of around 500K frames with associated ground-truth label (e.g. semantic segmentation, traffic light state and distance). This dataset was collected only in Town01 and under training weathers. Then we trained our RL agent with the *implicit affordances* coming from this new encoder for around 40M steps using 9 actors with all actors on Town01 under training weathers. We used a slightly bigger field of view (from 90° to 100°) and we cropped the sky (from 288x288x3 images to 288x168x3) as the EU traffic lights are less high than the US traffic lights (the CARLA benchmark contains only EU traffic lights). Finally, we removed all the change lane orders because all towns in CARLA benchmark are single lane (the CARLA benchmark setup is actually simpler than the CARLA challenge for which this paper has been initially done).

2.5. Training infrastructure

The training of the agents was split over several computers and GPUs, containing in total:

- 3 Nvidia Titan X and 1 Nvidia Titan V (training computer)
- 1 Nvidia 1080 Ti (local workstation)
- 2 Nvidia 1080 (local workstations)
- 3 Nvidia 2080 (training computer)

References

[1] Dian Chen, Brady Zhou, and Vladlen Koltun. Learning by Cheating. Technical report, 2019. 3

[2] Felipe Codevilla, Matthias Miiller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018. 1

[3] Felipe Codevilla, Eder Santana, Antonio M. López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving, 2019. 3

[4] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1, 3

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. 1

[6] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond, 2019. 1

[7] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation, 2016. 1

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 2

[9] Axel Sauer, Nikolay Savinov, and Andreas Geiger. Conditional affordance learning for driving in urban environments. *arXiv preprint arXiv:1806.06498*, 2018. 3

[10] Marin Toromanoff and Emilie Wirbel. Is Deep Reinforcement Learning Really Superhuman on Atari? Leveling the playing field. Technical report, 2019. 1