Learning by Cheating - Supplementary Material

Dian Chen UT Austin
Brady Zhou Intel Labs, UT Austin
Vladlen Koltun Intel Labs
Philipp Krähenbühl UT Austin

1 Additional details

Map-view perspective transformation. Given a predicted waypoint $\tilde{\mathbf{w}} = (\tilde{w}_x, \tilde{w}_y)$ in camera coordinates, we compute its projection into the ground plane $(\hat{w}_x, \hat{w}_y, 0)$ using cameras horizontal field of view (fov), $f = \frac{w}{2\tan(fov/2)}$, height p_y, and canvas center $c_x = \frac{w}{2}, c_y = \frac{h}{2}$. We assume the camera always faces forward, as the map is anchored at the agents position and in its local coordinate frame. We always project points onto a constant ground plane $z = 0$ to avoid depth estimation: $\hat{w}_y = \frac{f}{c_y - \tilde{w}_y} p_y, \hat{w}_x = \frac{\tilde{w}_x - c_x}{c_y - \tilde{w}_y} p_y$. To make the projection a one-to-one mapping, we additionally move the projected points back by 4 meters, to prevent points closer to the ego-vehicle getting clipped at the image bottom. This transformation is differentiable, and the sensorimotor agent can be end-to-end trained by map coordinate waypoints. The camera is placed at $p = (2, 0, 1.4)$ in the vehicle coordinate, at the hood position. The camera faces forward, and has a resolution of 384×160 with horizontal fov $90°$.

Data collection We use 157K training frames and 39K validation frames at 10 fps collected by a hand-crafted autopilot to train the privileged agent. We use 174K training frames in our 0.9.6 implementation. For both our offline dataset collection and privileged rollouts, we collect the frames using four training weather conditions uniformly sampled in the training town. We add 100 other vehicles to share the traffic with the ego-vehicle. We add 250 pedestrians in our 0.9.6 implementation.

Hyperparameters We use the Adam optimizer with initial learning rate 10^{-4}, no weight decay to train all our models. We use batch size 32 to train all of our models in 0.9.5, and we use batch size 128 to train the privileged model, and batch size 96 to train the image model in 0.9.6. We used the batch augmentation trick [5] with $m = 4$ for our image model training in 0.9.6. For the spatial argmax layers in both privileged and sensorimotor agent, we fix temperature $\beta = 1$ instead of a learnable parameter. We use PyTorch 1.0 to train and evaluate our models.

Image model warm-up Since a randomly initialized network returns the canvas center at the end of the spatial argmax layer in the image coordinate, it corresponds to infinitely far when projected. This causes exploding gradients in the backward pass. To address this issue, we warm-up our image model by first supervising it with loss in the projected image coordinate space for 1K iterations before the two-stage training.

2 Additional experiments

If an agent can perform near perfectly using a map representation, why not simply try to predict the map representation from raw pixels, then act on that? This approach resembles that of Müller et al. [7], where perception and control are explicitly decoupled and trained separately.

We train two networks - the first network is used for perception and directly predicts the privileged representation from an RGB image. We resize the RGB image to 192×192 and feed this into a ResNet34 [4] backbone, followed by five layers of bilinear-upsampling + convolution + ReLU to produce a map of the original resolution of 192×192. The perception network is trained using an L1 loss between the network's output, and the ground truth privileged representation. The second network is used for action and predicts waypoints from the output of the first network. We use the same architecture and training procedure as the privileged agent in our main experiments, and we freeze the weights of the perception network during training.

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

We use a dataset of 150k frames collected from an expert in training conditions, with no trajectory noise. The offline map predictions on the training and validation sets are quite good, but we notice that during evaluation, even slight out of distribution observations produce erroneous map predictions, causing the waypoint network to fail. To address this, we collect another dataset of the same size and employ trajectory noise [1] in 20% of the frames, to broaden the states seen by the perception network. Table 1 shows the results. Both map prediction agents performs significantly worse than our two-stage agent.

	Train Town		Test Town	
Method	Train Weather	Test Weather	Train Weather	Test Weather
No augmentation	39	34	30	32
Trajectory noise	58	62	65	62
LBC	100	100	100	100

Table 1: Map prediction baseline with and without trajectory noise compared to our two stage LBC, evaluated on the Navigation task of the *CoRL2017* benchmark on CARLA 0.9.5.

3 Benchmark results

For completeness, Table 2 and Table 3 show the training town performance in the CARLA CoLR 2017 and NoCrash benchmark respectively. We again compare to MP [3], CIL [1, 3], CAL [8], CIRL [6], and CILRS [2].

Table 4 compares the different CARLA versions 0.8 and 0.9.5. We compare the performance of CILRS on the Navigation Dynamic task between with (version 0.8), and without a working pedestrian autopilot (version 0.9.5). The CILRS performance in 0.9.5 matches the older CARLA version in testing weathers, and is slightly lower in the training weathers. This indicates that CARLA 0.9.5 does not make the task easier. We report the higher numbers from the CILRS paper [2].

Task	Weather	MP[3]	CIL[1]	CIRL[6]	CAL[8]	CILRS[2]	**LBC**	**LBC†**
Straight	train	98	98	98	**100**	96	**100**	**100**
One Turn		82	89	97	97	92	**100**	**100**
Navigation		80	86	93	92	95	**100**	**100**
Nav. Dynamic		77	83	82	83	92	**100**	**100**
Straight	test	**100**	98	**100**	**100**	96	**100**	**100**
One Turn		95	90	94	**96**	**96**	**100**	**96**
Navigation		94	84	86	90	96	**100**	**100**
Nav. Dynamic		89	82	80	82	**96**	**96**	**96**

Table 2: Quantitative results on the training town in the CoRL2017 CARLA benchmark in the training town. LBC† denotes our agent trained and evaluated on our customized CARLA based on 0.9.6, the most up-to-data CARLA version. Note that CARLA 0.9.6 has different graphics comparing to 0.8 and 0.9.5, while the latter two share the same.

References

[1] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy. End-to-end driving via conditional imitation learning. In *ICRA*, 2018.

[2] F. Codevilla, E. Santana, A. López, and A. Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *ICCV*, 2019.

[3] A. Dosovitskiy, G. Ros, F. Codevilla, A. López, and V. Koltun. CARLA: An open urban driving simulator. In *CoRL*, 2017.

Task	Weather	CARLA ≤0.9.5 CIL[1]	CAL[8]	CILRS[2]	LBC	CARLA 0.9.6 LBC	PV	AT
Empty	train	79 ± 1	81 ± 1	87 ± 1	**100** ± 0	**97** ± 1	100 ± 1	100 ± 0
Regular	train	60 ± 1	73 ± 2	83 ± 0	**99** ± 1	**93** ± 1	96 ± 3	99 ± 1
Dense	train	21 ± 2	42 ± 1	42 ± 2	**95** ± 2	**71** ± 5	80 ± 5	86 ± 3
Empty	test	83 ± 2	85 ± 2	87 ± 1	**100** ± 0	87 ± 4	100 ± 0	100 ± 0
Regular	test	55 ± 5	68 ± 5	88 ± 2	**99** ± 1	**87** ± 3	97 ± 3	99 ± 1
Dense	test	13 ± 4	33 ± 2	70 ± 3	**97** ± 2	**63** ± 1	81 ± 6	83 ± 6

Table 3: Quantitative results on the training town in the NoCrash benchmark. The methods were run on CARLA <=0.9.5. LBC† denotes our agent trained and evaluated on our customized CARLA based on 0.9.6, the most up-to-data CARLA version. Note that CARLA 0.9.6 has different graphics comparing to 0.8 and 0.9.5, while the latter two share the same.

	CARLA version	Train Town Train Weather	Train Town Test Weather	Test Town Train Weather	Test Town Test Weather
CILRS [2]	0.8.4	92	96	66	90
CILRS	0.9.5 (no ped)	84	96	53	92

Table 4: Comparison of CILRS [2] on different versions of CARLA on the Navigation Dynamic task of the *CoRL2017* benchmark.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[5] E. Hoffer, T. Ben-Nun, I. Hubara, N. Giladi, T. Hoefler, and D. Soudry. Augment your batch: better training with larger batches. *arXiv:1901.09335*, 2019.

[6] X. Liang, T. Wang, L. Yang, and E. Xing. CIRL: Controllable imitative reinforcement learning for vision-based self-driving. In *ECCV*, 2018.

[7] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun. Driving policy transfer via modularity and abstraction. In *CoRL*, 2018.

[8] A. Sauer, N. Savinov, and A. Geiger. Conditional affordance learning for driving in urban environments. In *CoRL*, 2018.