Supplementary

1. Additional Details on CARLA Experiments

In Sec 5, we specified the experimental setup for our experiments in the CARLA driving environment, which closely follows standard protocol from Codevilla et al. (2018; 2019); Chen et al. (2020). We provide additional details here.

Data Collection. The CARLA100 dataset (Codevilla et al., 2019) in our experiments is collected with noise injection (Laskey et al., 2017) to perturb 10% of expert actions. It also uses three cameras: a forward-facing one and two lateral cameras facing 30 degrees away towards left or right (Bojarski et al., 2016), for data augmentation to guard against distributional shift.

Architectures and Training Details. All our baseline models use ImageNet-pretrained Resnet34 as our perception model (Codevilla et al., 2019). We use the state-of-the-art conditional imitation learning model *CILRS* (Codevilla et al., 2019) as our backbone, with a weighted control loss (Codevilla et al., 2018) that assigns weights of 0.5 to steer, 0.45 to throttle and 0.05 to brake. We train all models for 10^5 training iterations with minibatch size 120. We use Adam optimizer, set the initial learning rate to 1×10^{-4} and decay the learning rate by 0.1 whenever the loss value no longer decreases for 5000 iterations. We use L_1 loss as our loss function.

Defining the Metrics. We now more thoroughly define all the metrics we used to measure the performance of the CARLA experiments, following standard protocol (Codevilla et al., 2018; 2019; Chen et al., 2020):

- The %success is the number of episodes fully completed by the ego vehicle among all the 100 predefined test episodes. Higher is better.

- #collision reports the total times the ego car crashes into the pedestrians, vehicles and other obstructions (over 100 test episodes). Lower is better.

- For %progress, we calculate the euclidean distance from the start point to the target point at the beginning of each episode (initial distance) and then compute the distance from the start point to the final point after the episode is over (final distance); and the %progress is 1 minus the ratio between final distance and initial distance. Higher is better.

- The avg. speed is calculated by dividing the path length that the ego car traveled by the travel time. Higher is better in general, but approaches that drive the vehicle straight into pedestrians at high speed will still get avg. speed, so this metric is not very informative without the context of the other metrics.

Reporting Results In Terms of All Four Metrics. In Sec 5 in the paper, we only had space to report the quantitative results in terms of %success, but we remarked on broad trends in terms of the other metrics above too. We now report those complete results in Table 1 (for unmodified CARLA, including the agent velocity inputs, corresponding to Table 1 in the paper) and Table 2 (for the CARLA-w/o-speed setting with increased partial observability).

Expanded Fig 6. Figure 6 in the paper compares the imitation errors of various models on changepoint and non-changepoint samples. We now report those same results in tabular form for increased precision, in Table 3.

2. Additional Details on MuJoCo Experiments

Architectures and Training Details. We use ResNet18 as our backbone and a two-layer MLP with 300 hidden units on top of it to get the predicted action. We use Adam optimizer with initial learning rate 2×10^{-4} and reduce the learning rate by a factor of 10 every 100k iterations during training. We use a batchsize of 128. We train the imitation agent for 300k iterations until convergence. We use L_2 as our loss function.

3. Hyperparameter Choices

Since there are only few hyperparameters in our method, we can perform grid search to find the best set of hyperparameters based on evaluation reward. For $\mathrm{softmax}$ weighting function, we select its temperature τ from $\{0.1, 0.2, 0.5, 1, 5, 10\}$ and for step weighting function, we select its threshold THR from $\{10\%, 20\%\}$ and its weight W from $\{3, 5, 10\}$.

Table 1. CARLA test results in Nocrash-Dense benchmark. BC-SO is significantly better than BC-OH, verifying the causal confusion phenomenon. Our method outperforms both BC-SO and BC-OH with higher success rate. Moreover, our method significantly reduces the #collision, which is very important for urban driving tasks. Comparing with the baselines, we do better in all the four metrics than the two offline algorithms and even beats the Dagger which require online query.

METHOD	%SUCCESS (/100) (↑)	#COLLISION (↓)	%PROGRESS (↑)	AVG. SPEED (↑)
BC-SO (CODEVILLA ET AL., 2019)	**42.667 ± 8.668**	48.444 ± 9.044	**0.580 ± 0.055**	15.559 ± 3.035
BC-OH	33.000 ± 4.190	52.111 ± 5.878	0.497 ± 0.042	11.775 ± 3.225
OURS (STEP)	**43.444 ± 0.786**	**42.615 ± 2.228**	**0.580 ± 0.040**	14.938 ± 2.759
FCA (WEN ET AL., 2020)	35.667 ± 3.559	50.333 ± 4.643	0.551 ± 0.030	13.927 ± 2.905
HISTORYDROPOUT (BANSAL ET AL., 2019)	34.000 ± 2.625	60.222 ± 3.119	0.506 ± 0.029	17.847 ± 2.120
DAGGER 120K (ROSS ET AL., 2011)	35.222 ± 3.067	60.000 ± 2.625	0.512 ± 0.026	**18.515 ± 1.586**
BCPD	28.667 ± 2.494	36.667 ± 8.260	0.440 ± 0.057	10.071 ± 6.668
ACTFREQ	20.333 ± 5.825	26.000 ± 5.354	0.410 ± 0.089	9.229 ± 3.353
BOOSTING	3.000 ± 1.414	19.333 ± 4.110	0.101 ± 0.026	1.933 ± 2.174

Table 2. CARLA-w/o-speed results in Nocrash-Dense benchmark. In this case, BC-SO's performance is much worse than BC-OH for lack of historical information and it is pretty difficult to infer speed from a single frame. Our method still performs well in such an information-constrained situation.

METHOD	%SUCCESS (/100) (↑)	#COLLISION (↓)	%PROGRESS (↑)	AVG. SPEED (↑)
BC-SO (CODEVILLA ET AL., 2019)	9.222 ± 2.380	84.667 ± 4.769	0.204 ± 0.033	35.182 ± 1.594
BC-OH	25.667 ± 0.981	60.889 ± 1.911	0.467 ± 0.049	15.543 ± 3.714
OURS (STEP)	**36.778 ± 5.808**	**45.333 ± 4.690**	**0.540 ± 0.050**	14.416 ± 3.145
FCA (WEN ET AL., 2020)	27.444 ± 4.113	59.222 ± 5.996	0.453 ± 0.052	13.856 ± 2.729
HISTORYDROPOUT (BANSAL ET AL., 2019)	25.333 ± 5.375	66.778 ± 5.865	0.449 ± 0.047	16.086 ± 3.889
DAGGER 120K (ROSS ET AL., 2011)	28.333 ± 3.496	62.667 ± 3.771	0.457 ± 0.014	**16.988 ± 2.757**
BCPD	20.000 ± 1.414	44.000 ± 9.899	0.400 ± 0.0626	5.379 ± 2.259
ACTFREQ	14.667 ± 1.764	24.667 ± 3.972	0.313 ± 0.031	3.649 ± 3.073
BOOSTING	10.000 ± 2.160	49.667 ± 4.028	0.223 ± 0.026	12.502 ± 3.242

Our method We use step function for CARLA environment. In both of *CARLA* and *CARLA-w/o-speed*, we set THR to 10% and W to 5, i.e., upweight the top 10% of samples by a factor of 5.

In *MuJoCo-Image*, we experiment with both step and softmax function. For the softmax function, we use a temperature of 0.2 for Hopper and Walker2D and 0.1 for HalfCheetah. For the step function, we set the THR to 10% and W to 5.

Baselines In HistoryDropout, we set the dropout rate to 0.5 for all environments. In FCA, we set different adversarial loss weight for different environment to balance the adversarial loss and the imitation loss. More specifically, the adversarial loss weights for the different environments are *CARLA* and *CARLA-w/o-speed*: 0.2, Hopper: 0.1, HalfCheetah and Walker2D: 0.5.

4. Sensitivity Study

To study the sensitivity of out method to the hyperparameter choices, we ran some additional experiments by perturbing the hyperparameters. For Walker 2D, we set the softmax temperature to 0.1, 0.2, and 0.5, and the test rewards are 765 ± 103, 769 ± 97, and 724 ± 61. And for CARLA, the weight W of step function is set to 4, 5 and 6, producing success rate 41.89, 43.44, and 41.33%. These results suggest that our method is pretty robust to the hyperparameter choices.

5. Additional Sandbox Environment: ToyCar

In addition to the standard environments in the paper, we constructed a simple sandbox environment for fast experimentation, that we will refer to as ToyCar. The environment consists of a car on a straight road with a traffic light that stays on red or green for random lengths of time as shown in Fig 1. The car must successfully drive through the road from left to right without breaking red lights. The car fol-

Table 3. The loss of different types of samples. Our method significantly reduces the loss of hard samples, thus reducing the empirical risks among the whole dataset. The standard deviation values are smaller than 1×10^{-3} so we don't put them here.

Experiment	Method	Unweighted loss $\times 10^{-2}$		Changepoint loss $\times 10^{-2}$		Copycat sample loss $\times 10^{-2}$	
		train	val	train	val	train	val
w/-speed	BC-SO	1.769	1.464	4.995	4.222	1.410	1.157
	BC-OH	1.549	1.395	5.721	4.808	1.086	1.016
	Ours (step)	1.742	1.537	4.545	3.731	1.431	1.294
w/o-speed	BC-SO	5.350	4.989	11.213	11.059	4.710	4.316
	BC-OH	2.172	2.055	8.194	7.588	1.504	1.441
	Ours (step)	1.671	1.491	4.985	4.132	1.303	1.198

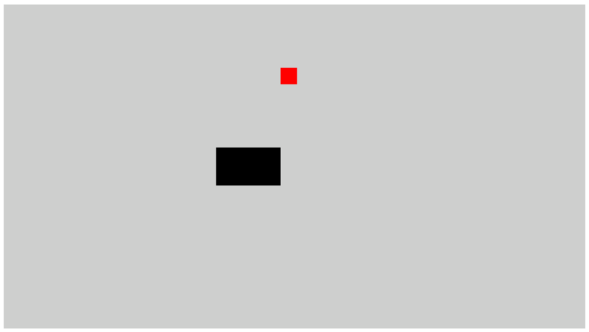
Figure 1. Snapshot of the ToyCar environment.

lows point mass double integrator dynamics, and has two actions, throttle, and brake, that apply fixed positive and negative accelerations. There is a fixed upper limit on the velocity. The expert is a rule-based agent and has access to the full observation, i.e. the car position, the velocity, the traffic light position, the traffic light status and the time remaining for the current status. All the imitator policies act on 3x128x128 images.

Table 4. ToyCar results.

	BC-SO	BC-OH	Ours(step)
%Success Rate	97.2 ± 0.7	95.1 ± 4.5	97.8 ± 0.5

Results We train the BC-SO, BC-OH and our method with 1K expert samples, and test them in the environment. The test success rates are shown in the Table 4. All results are reported over 5 trials. Quantitative results mirror those reported in the paper, but gains are relatively small due to the fact that the environment is very simple. Qualitatively, we observed instances of similar problems with BC-OH to the inertia problem reported before in Codevilla et al. (2019), namely, the car sometimes stops at a seemingly random location and refuses to start again. Our method successfully removes those failure cases.

6. Low-Dimensional Environments: MuJoCo-State

Finally, while in the paper, we showed results for image-based MuJoCo settings, we now report results in low-dimensional partially observed MuJoCo settings. Following Wen et al. (2020), we remove the velocity and external force information from the full state to make the environment partially observed.

The environmental rewards are shown in Tab 5. As in the image-based settings, our method successfully outperforms BC-OH easily in all these settings. Further, our results in these environments are comparable to FCA (Wen et al., 2020). We clearly outperform FCA on Walker2D, perform on par with FCA on HalfCheetah, and perform worse on Hopper. However, the FCA method, reliant on adversarial training, scales poorly to higher-dimensional image-based environments as mentioned in Wen et al. (2020), and also verified throughout our other results.

Table 5. MuJoCo-State results.

	Hopper	HalfCheetah	Walker2D
BC-SO	275 ± 40	-38 ± 36	363 ± 86
BC-OH	293 ± 83	820 ± 60	592 ± 124
FCA	**1086 ± 262**	**1250 ± 42**	1296 ± 288
Ours	641 ± 7	**1023 ± 75**	**1460 ± 169**

7. Additional Baseline: Category Frequency Weighting with Discovered Categories

As we mentioned in Sec 2 in the paper, prior approaches (Cui et al., 2019; Cao et al., 2019) have proposed rebalancing data in supervised learning settings based on the frequencies of categories, so that low-frequency categories are upsampled to correct data imbalance issues. We now report the results of a baseline that discovers action categories in an unsupervised manner and apply a similar strategy. This evaluates a different reweighting strategy to the APE-based strategy proposed in the paper.

We assume that there are 6 typical scenarios in urban autonomous driving tasks, i.e. going straight, turning right, turning left, accelerating, slowing down and parking, so we use the k-means algorithm to cluster the actions in the training dataset into $k = 6$ clusters as categories. And we use category frequency based weighting function $w_i = \frac{\sum_{j=1}^{6} n_j}{n_i}$ to reweight each sample in the cluster i, where n_j is the number of samples in cluster j.

This naive reweighting strategy-based baseline does not perform very well. The results are shown in the last rows in Table 1 and Table 2.

References

Mayank Bansal, Alex Krizhevsky, and Abhijit Ogale. ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst. *Robotics: Science & Systems (RSS)*, art. arXiv:1812.03079, 2019.

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. *CoRR*, abs/1604.07316, 2016.

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

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.

Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9329–9338, 2019.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9268–9277, 2019.

Michael Laskey, Anca Dragan, Jonathan Lee, Ken Goldberg, and Roy Fox. Dart: Optimizing noise injection in imitation learning. In *Conference on Robot Learning (CoRL)*, volume 2, page 12, 2017.

Stéphane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. *Journal of Machine Learning Research*, 15:627–635, 2011. ISSN 15324435.

Chuan Wen, Jierui Lin, Trevor Darrell, Dinesh Jayaraman, and Yang Gao. Fighting copycat agents in behavioral cloning from observation histories. In *Advances in Neural Information Processing Systems*, volume 33, 2020.