

End-to-End Urban Driving by Imitating a Reinforcement Learning Coach

Zhejun Zhang¹, Alexander Liniger¹, Dengxin Dai^{1,2}, Fisher Yu¹ and Luc Van Gool^{1,3}

¹Computer Vision Lab, ETH Zürich, ²MPI for Informatics, ³PSI, KU Leuven

{zhejun.zhang, alex.liniger, dai, vangoool}@vision.ee.ethz.ch, i@yf.io

Abstract

End-to-end approaches to autonomous driving commonly rely on expert demonstrations. Although humans are good drivers, they are not good coaches for end-to-end algorithms that demand dense on-policy supervision. On the contrary, automated experts that leverage privileged information can efficiently generate large scale on-policy and off-policy demonstrations. However, existing automated experts for urban driving make heavy use of hand-crafted rules and perform suboptimally even on driving simulators, where ground-truth information is available. To address these issues, we train a reinforcement learning expert that maps bird’s-eye view images to continuous low-level actions. While setting a new performance upper-bound on CARLA, our expert is also a better coach that provides informative supervision signals for imitation learning agents to learn from. Supervised by our reinforcement learning coach, a baseline end-to-end agent with monocular camera-input achieves expert-level performance. Our end-to-end agent achieves a 78% success rate while generalizing to a new town and new weather on the NoCrash-dense benchmark and state-of-the-art performance on the challenging public routes of the CARLA LeaderBoard.

1. Introduction

Even though nowadays, most autonomous driving (AD) stacks [31, 49] use individual modules for perception, planning and control, end-to-end approaches have been proposed since the 80’s [36] and the success of deep learning brought them back into the research spotlight [5, 51]. Numerous works have studied different network architectures for this task [3, 17, 53], yet most of these approaches use supervised learning with expert demonstrations, which is known to suffer from covariate shift [37, 41]. While data augmentation based on view synthesis [2, 5, 36] can partially alleviate this issue, in this paper, we tackle the problem from the perspective of expert demonstrations.

Expert demonstrations are critical for end-to-end AD algorithms. While imitation learning (IL) methods directly

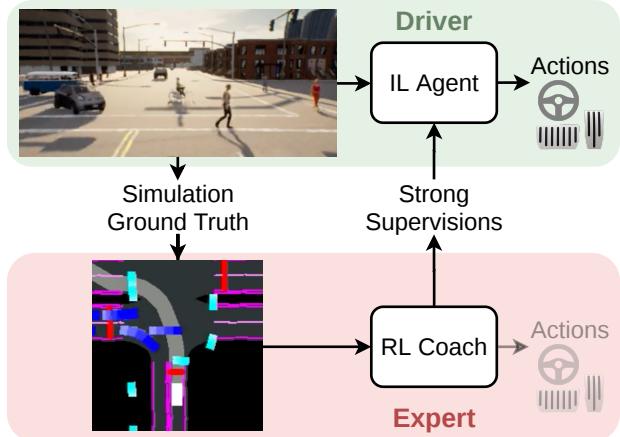


Figure 1: **Roach:** RL coach allows IL agents to benefit from dense and informative on-policy supervisions.

mimic the experts’ behavior [3, 11], reinforcement learning (RL) methods often use expert demonstrations to improve sample efficiency by pre-training part of the model via supervised learning [28, 48]. In general, expert demonstrations can be divided into two categories: (i) *Off-policy*, where the expert directly controls the system, and the state/observation distribution follows the expert. Off-policy data for AD includes, for example, public driving datasets [7, 23, 52]. (ii) *On-policy*, where the system is controlled by the desired agent and the expert “labels” the data. In this case, the state/observation distribution follows the agent, but expert demonstrations are accessible. On-policy data is fundamental to alleviate covariate shift as it allows the agent to learn from its own mistakes, which the expert in the off-policy data does not exhibit. However, collecting adequate on-policy demonstrations from humans is non-trivial. While trajectories and actions taken by the human expert can be directly recorded during off-policy data collection, labeling these targets given sensor measurements turns out to be a challenging task for humans. In practice, only sparse events like human interventions are recorded, which, due to the limited information it contains, is hard to use for training and better suited for RL [2, 24, 25] than for IL methods.

In this work we focus on automated experts, which in

contrast to human experts can generate large-scale datasets with dense labels regardless of whether they are on-policy or off-policy. To achieve expert-level performance, automated experts may rely on exhaustive computations, expensive sensors or even ground truth information, so it is undesirable to deploy them directly. Even though some IL methods do not require on-policy labeling, such as GAIL [21] and inverse RL [1], these methods are not efficient in terms of on-policy interactions with the environment.

On the contrary, automated experts can reduce the expensive on-policy interactions. This allows IL to successfully apply automated experts to different aspects of AD. As a real-world example, Pan et al. [35] demonstrated end-to-end off-road racing with a monocular camera by imitating a model predictive control expert with access to expensive sensors. In the context of urban driving, [37] showed that a similar concept can be applied to the driving simulator CARLA [13]. Driving simulators are an ideal proving ground for such approaches since they are inherently safe and can provide ground truth states. However, there are two caveats. The first regards the “expert” in CARLA, commonly referred to as the Autopilot (or the roaming agent). The Autopilot has access to ground truth simulation states, but due to the use of hand-crafted rules, its driving skills are not comparable to a human expert’s. Secondly, the supervision offered by most automated experts is not informative. In fact, the IL problem can be seen as a knowledge transfer problem and just learning from expert actions is inefficient.

To tackle both drawbacks and motivated by the success of model-free RL in Atari games [19] and continuous control [15], we propose Roach (RL coach), an RL expert that maps bird’s-eye view (BEV) images to continuous actions (Fig. 1 bottom). After training from scratch for 10M steps, Roach sets the new performance upper-bound on CARLA by outperforming the Autopilot. We then train IL agents and investigate more effective training techniques when learning from our Roach expert. Given that Roach uses a neural network policy, it serves as a better coach for IL agents also based on neural networks. Roach offers numerous informative targets for IL agents to learn from, which go far beyond deterministic action provided by other experts. Here we demonstrate the effectiveness of using action distributions, value estimations and latent features as supervisions.

Fig. 1 shows the scheme of learning from on-policy supervisions labeled by Roach on CARLA. We also record off-policy data from Roach by using its output to drive the vehicle on CARLA. Leveraging 3D detection algorithms [27, 50] and extra sensors to synthesize the BEV, Roach could also address the scarcity of on-policy supervisions in the real world. This is feasible because on the one hand, BEV as a strong abstraction reduces the sim-to-real gap [32], and on the other hand, on-policy labeling does not have to happen in real-time or even onboard. Hence 3D de-

tection becomes easier given the complete sequences [38].

In summary, this paper presents Roach, an RL expert that sets a new performance upper-bound on CARLA. Moreover, we demonstrate the state-of-the-art performance on both the NoCrash benchmark and the public routes of CARLA LeaderBoard using a single camera based end-to-end IL agent, which is supervised by Roach using our improved training scheme. Our repository is available at <https://github.com/zhejz/carla-roach>

2. Related Work

Since our methods are trained and evaluated on CARLA, we mainly focus on related works also done on CARLA.

End-to-End IL: Dosovitskiy et al. [13] introduced the CARLA driving simulator and demonstrated that a baseline end-to-end IL method with single camera input can achieve a performance comparable to a modular pipeline. After that, CIL [11] and CILRS [12] addressed directional multimodality in AD by using branched action heads where the branch is selected by a high-level directional command. While the aforementioned methods are trained via behavior cloning, DA-RB [37] applied DAGGER [41] with critical state sampling to CILRS. Most recently, LSD [33] increased the model capacity of CILRS by learning a mixture of experts and refining the mixture coefficients using evolutionary optimization. Here, we use DA-RB as the baseline IL agent to be supervised by Roach.

Mid-to-X IL: Directly mapping camera images to low-level actions requires a large amount of data, especially if one wants generalization to diverse weather conditions. Mid-to-X approaches alleviate this issue by using more structured intermediate representation as input and/or output. CILRS with coarse segmentation masks as input was studied in [4]. CAL [42] combines CIL and direct perception [8] by mapping camera images to driving affordances which can be directly used by a rule-based low-level controller. LBC [9] maps camera images to waypoints by mimicking a privileged mid-to-mid IL agent similar to Chauffeurnet [3], which takes BEV as input and outputs future waypoints. Similarly, SAM [54] trained a visuomotor agent by imitating a privileged CILRS agent that takes segmentation and affordances as inputs. Our Roach adopts BEV as the input representation and predicts continuous low-level actions.

RL: As the first RL agent on CARLA, an A3C agent [30] was demonstrated in [13], yet its performance is lower than that of other methods presented in the same paper. CIRL [28] proposed an end-to-end DDPG [29] agent with its actor network pre-trained via behavior cloning to accelerate online training. To reduce the problem complexity, Chen et al. [10] investigated DDQN [16], TD3 [14] and SAC [15] using BEV as an input and pre-trained the image encoder with a variational auto-encoder [26] on expert tra-

jectories. State-of-the-art performance is achieved in [48] using Rainbow-IQN [47]. To reduce the number of trainable parameters during online training, its image encoder is pre-trained to predict segmentation and affordances on an off-policy dataset. IL was combined with RL in [40] and multi-agent RL on CARLA was discussed in [34]. In contrast to these RL methods, Roach achieves high sample efficiency without using any expert demonstrations.

IL with Automated Experts: The effectiveness of automated experts was demonstrated in [35] for real-world off-road racing, where a visuomotor agent is trained by imitating on-policy actions labeled by a model predictive control expert equipped with expensive sensors. Although CARLA already comes with the Autopilot, it is still beneficial to train a proxy expert based on deep neural networks, as shown by LBC [9] and SAM [54]. Through a proxy expert, the complex to solve end-to-end problem is decomposed into two simpler stages. At the first stage, training the proxy expert is made easier by formulating a mid-to-X IL problem that separates perception from planning. At the second stage, the end-to-end IL agent can learn more effectively from the proxy expert given the informative targets it supplies. To provide strong supervision signals, LBC queries all branches of the proxy expert and backpropagates all branches of the IL agent given one data sample, whereas SAM matches latent features of the proxy expert and the end-to-end IL agent. While the proxy expert addresses planning, it is also possible to address perception at the first stage as shown by FM-Net [22]. Overall, two-stage approaches achieve better performance than direct IL, but using proxy experts inevitably lowers the performance upper-bound as a proxy expert trained via IL cannot outperform the expert it imitates. This is not a problem for Roach, which is trained via RL and outperforms the Autopilot.

3. Method

In this section we describe Roach and how IL agents can benefit from diverse supervisions supplied by Roach.

3.1. RL Coach

Our Roach has three features. Firstly, in contrast to previous RL agents, Roach does not depend on data from other experts. Secondly, unlike the rule-based Autopilot, Roach is end-to-end trainable, hence it can generalize to new scenarios with minor engineering efforts. Thirdly, it has a high sample efficiency. Using our proposed input/output representation and exploration loss, training Roach from scratch to achieve top expert performance on the six LeaderBoard maps takes less than a week on a single GPU machine.

Roach consists of a policy network $\pi_\theta(\mathbf{a}|\mathbf{i}_{\text{RL}}, \mathbf{m}_{\text{RL}})$ parameterized by θ and a value network $V_\phi(\mathbf{i}_{\text{RL}}, \mathbf{m}_{\text{RL}})$ parameterized by ϕ . The policy network maps a BEV image \mathbf{i}_{RL}

and a measurement vector \mathbf{m}_{RL} to a distribution of actions \mathbf{a} . Finally the value network estimates a scalar value v , while taking the same inputs as the policy network.

Input Representation: We use a BEV semantic segmentation image $\mathbf{i}_{\text{RL}} \in [0, 1]^{W \times H \times C}$ to reduce the problem complexity, similar to the one used in [3, 9, 10]. It is rendered using ground-truth simulation states and consists of C grayscale images of size $W \times H$. The ego-vehicle is heading upwards and is centered in all images at D pixels above the bottom, but it is not rendered. Fig. 2 illustrates each channel of \mathbf{i}_{RL} . Drivable areas and intended routes are rendered respectively in Fig. 2a and 2b. In Fig. 2c solid lines are white and broken lines are grey. Fig. 2d is a temporal sequence of K grayscale images in which cyclists and vehicles are rendered as white bounding boxes. Fig. 2e is the same as Fig. 2d but for pedestrians. Similarly, stop lines at traffic lights and trigger areas of stop signs are rendered in Fig. 2f. Red lights and stop signs are colored by the brightest level, yellow lights by an intermediate level and green lights by a darker level. A stop sign is rendered if it is active, i.e. the ego-vehicle enters its vicinity and disappears once the ego-vehicle has made a full stop. By letting the BEV representation memorize if the ego-vehicle has stopped, we can use a network architecture without recurrent structure and hence reduce the model size of Roach. A colored combination of all channels is visualized in Fig. 1. We also feed Roach a measurement vector $\mathbf{m}_{\text{RL}} \in \mathbb{R}^6$ containing the states of the ego-vehicle not represented in the BEV, these include ground-truth measurements of steering, throttle, brake, gear, lateral and horizontal speed.

Output Representation: Low-level actions of CARLA are $\text{steering} \in [-1, 1]$, $\text{throttle} \in [0, 1]$ and $\text{brake} \in [0, 1]$. An effective way to reduce the problem complexity is predicting waypoint plans which are then tracked by a PID-controller to produce low-level actions [9, 40]. However, a PID-controller is not reliable for trajectory tracking and requires excessive parameter tuning. A model-based controller would be a better solution, but CARLA’s vehicle dynamics model is not directly accessible. To avoid parameter tuning and system identification, Roach directly predicts action distributions. Its action space is $\mathbf{a} \in [-1, 1]^2$ for steering and acceleration, where positive acceleration corresponds to throttle and negative corresponds to brake. To describe actions we use the Beta distribution $\mathcal{B}(\alpha, \beta)$, where $\alpha, \beta > 0$ are respectively the concentration on 1 and 0. Compared to the Gaussian distribution, which is commonly used in model-free RL, the support of the Beta distribution is bounded, thus avoiding clipping or squashing to enforce input constraints. This results in a better behaved learning problem since no tanh layers are needed and the entropy and KL-divergence can be computed explicitly. Further, the modality of the Beta distribution is also suited for driving,

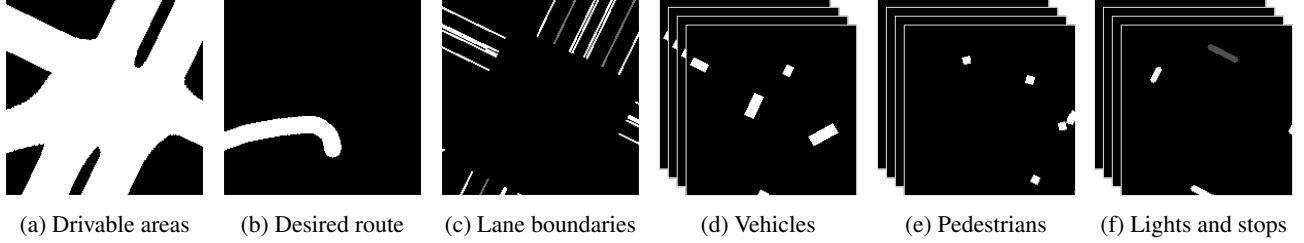


Figure 2: The BEV representation used by our Roach.

where extreme maneuvers may often be taken, for example, emergency braking or taking a sharp turn.

Training: We use proximal policy optimization (PPO) [44] with clipping to train the policy network π_θ and the value network V_ϕ . To update both networks, we collect trajectories by executing π_{θ_k} on CARLA. A trajectory $\tau = \{(\mathbf{i}_{RL,k}, \mathbf{m}_{RL,k}, \mathbf{a}_k, r_k)^T_{k=0}, z\}$ includes BEV images \mathbf{i}_{RL} , measurement vectors \mathbf{m}_{RL} , actions \mathbf{a} , rewards r and a terminal event $z \in \mathcal{Z}$ that triggers the termination of an episode. The value network is trained to regress the expected returns, whereas the policy network is updated via

$$\theta_{k+1} = \arg \max_{\theta} \mathbb{E}_{\tau \sim \pi_{\theta_k}} [\mathcal{L}_{\text{ppo}} + \mathcal{L}_{\text{ent}} + \mathcal{L}_{\text{exp}}]. \quad (1)$$

The first objective \mathcal{L}_{ppo} is the clipped policy gradient loss with advantages estimated using generalized advantage estimation [43]. The second objective \mathcal{L}_{ent} is a maximum entropy loss commonly employed to encourage exploration

$$\mathcal{L}_{\text{ent}} = -\lambda_{\text{ent}} \cdot H(\pi_\theta(\cdot | \mathbf{i}_{RL}, \mathbf{m}_{RL})). \quad (2)$$

Intuitively \mathcal{L}_{ent} pushes the action distribution towards a uniform prior because maximizing entropy is equivalent to minimizing the KL-divergence to a uniform distribution,

$$H(\pi_\theta) = -\text{KL}(\pi_\theta \| \mathcal{U}(-1, 1)), \quad (3)$$

if both distributions share the same support. This inspires us to propose a generalized form of \mathcal{L}_{ent} , which encourages exploration in sensible directions that comply with basic traffic rules. We call it the exploration loss and define it as

$$\begin{aligned} \mathcal{L}_{\text{exp}} &= \lambda_{\text{exp}} \cdot \mathbb{1}_{\{T-N_z+1, \dots, T\}}(k) \\ &\cdot \text{KL}(\pi_\theta(\cdot | \mathbf{i}_{RL,k}, \mathbf{m}_{RL,k}) \| p_z), \end{aligned} \quad (4)$$

where $\mathbb{1}$ is the indicator function and $z \in \mathcal{Z}$ is the event that ends the episode. The terminal condition set \mathcal{Z} includes collision, running traffic light/sign, route deviation and being blocked. Unlike \mathcal{L}_{ent} which imposes a uniform prior on the actions at all time steps regardless of which z is triggered, \mathcal{L}_{exp} shifts actions within the last N_z steps of an episode towards a predefined exploration prior p_z which encodes an “advice” to prevent the triggered event z from happening again. In practice, we use $N_z = 100, \forall z \in \mathcal{Z}$. If

z is related to a collision or running traffic light/sign, we apply $p_z = \mathcal{B}(1, 2.5)$ on the acceleration to encourage Roach to slow down while the steering is unaffected. In contrast, if the car is blocked we use an acceleration prior $\mathcal{B}(2.5, 1)$. For route deviations, a uniform prior $\mathcal{B}(1, 1)$ is applied on the steering. Despite being equivalent to maximizing entropy in this case, the exploration loss further encourages exploration on steering angles during the last 10 seconds before the route deviation.

Implementation Details: Our implementation of PPO-clip is based on [39] and the network architecture is illustrated in Fig. 3a. We use six convolutional layers to encode the BEV and two fully-connected (FC) layers to encode the measurement vector. Outputs of both encoders are concatenated and then processed by another two FC layers to produce a latent feature \mathbf{j}_{RL} , which is then fed into a value head and a policy head, each with two FC hidden layers. Trajectories are collected from six CARLA servers at 10 FPS, each server corresponds to one of the six LeaderBoard maps. At the beginning of each episode, a pair of start and target location is randomly selected and the desired route is computed using the A* algorithm. Once the target is reached, a new random target will be chosen, hence the episode is endless unless one of the terminal conditions in \mathcal{Z} is met. We use the reward of [47] and additionally penalize large steering changes to prevent oscillating maneuvers. To avoid infractions at high speed, we add an extra penalty proportional to the ego-vehicle’s speed. More details are in the supplement.

3.2. IL Agents Supervised by Roach

To allow IL agents to benefit from the informative supervisions generated by Roach, we formulate a loss for each of the supervisions. Our training scheme using Roach can be applied to improve the performance of existing IL agents. Here we use DA-RB [37] (CILRS [12] + DAGGER [41]) as an example to demonstrate its effectiveness.

CILRS: The network architecture of CILRS is illustrated in Fig. 3b, it includes a perception module that encodes the camera image \mathbf{i}_{IL} and a measurement module that encodes the measurement vector \mathbf{m}_{IL} . Outputs of both modules are concatenated and processed by FC layers to generate a bottleneck latent feature \mathbf{j}_{IL} . Navigation instructions are given

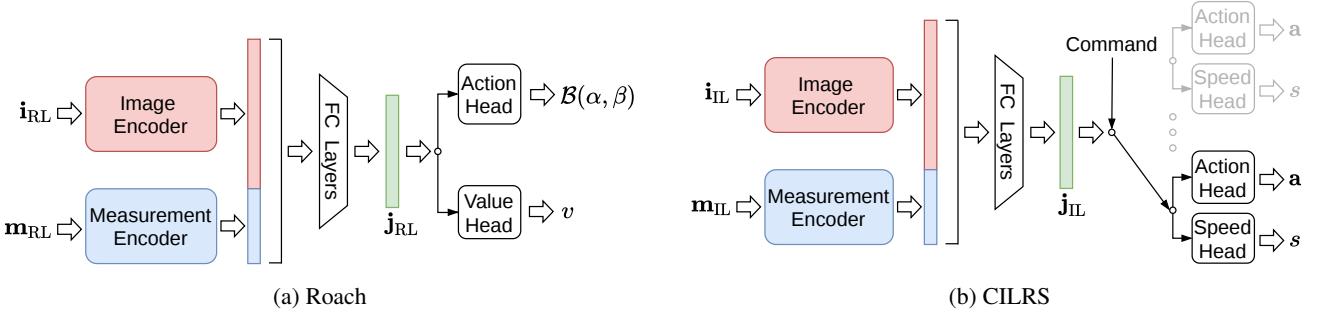


Figure 3: Network architecture of Roach, the RL expert, and CILRS, the IL agent.

as discrete high-level commands and for each kind of command a branch is constructed. All branches share the same architecture, while each branch contains an action head predicting continuous actions \mathbf{a} and a speed head predicting the current speed s of the ego-vehicle. The latent feature \mathbf{j}_{IL} is processed by the branch selected by the command. The imitation objective of CILRS consists of an L1 action loss

$$\mathcal{L}_A = \|\hat{\mathbf{a}} - \mathbf{a}\|_1 \quad (5)$$

and a speed prediction regularization

$$\mathcal{L}_S = \lambda_S \cdot |\hat{s} - s|, \quad (6)$$

where λ_S is a scalar weight, $\hat{\mathbf{a}}$ is the expert’s action, \hat{s} is the measured speed, \mathbf{a} and s are action and speed predicted by CILRS. Expert actions $\hat{\mathbf{a}}$ may come from the Autopilot, which directly outputs deterministic actions, or from Roach, where the distribution mode is taken as the deterministic output. Besides deterministic actions, Roach also predicts action distributions, values and latent features. Next we will formulate a loss function for each of them.

Action Distribution Loss: Inspired by [20] which suggests soft targets may provide more information per sample than hard targets, we propose a new action loss based on the action distributions as a replacement for \mathcal{L}_A . The action head of CILRS is modified to predict distribution parameters and the loss is formulated as a KL-divergence

$$\mathcal{L}_K = \text{KL}(\hat{\pi} \| \pi) \quad (7)$$

between the action distribution $\hat{\pi}$ predicted by the Roach expert and π predicted by the CILRS agent.

Feature Loss: Feature matching is an effective way to transfer knowledge between networks and its effectiveness in supervising IL driving agents is demonstrated in [22, 54]. The latent feature \mathbf{j}_{RL} of Roach is a compact representation that contains essential information for driving as it can be mapped to expert actions using an action head consists of only two FC layers (cf. Fig. 3a). Moreover, \mathbf{j}_{RL} is invariant to rendering and weather as Roach uses the BEV representation. Learning to embed camera images to the latent space

of \mathbf{j}_{RL} should help IL agents to generalize to new weather and new situations. Hence, we propose the feature loss

$$\mathcal{L}_F = \lambda_F \cdot \|\mathbf{j}_{RL} - \mathbf{j}_{IL}\|_2^2. \quad (8)$$

Value Loss: Multi-task learning with driving-related side tasks could also boost the performance of end-to-end IL driving agents as shown in [51], which used scene segmentation as a side task. Intuitively, the value predicted by Roach contains driving relevant information because it estimates the expected future return, which relates to how dangerous a situation is. Therefore, we augment CILRS with a value head and regress value as a side task. The value loss is the mean squared error between \hat{v} , the value estimated by Roach, and v , the value predicted by CILRS,

$$\mathcal{L}_V = \lambda_V \cdot (\hat{v} - v)^2. \quad (9)$$

Implementation Details: Our implementation follows DARB [37]. We choose a Resnet-34 pretrained on ImageNet as the image encoder to generate a 1000-dimensional feature given $\mathbf{i}_{RL} \in [0, 1]^{900 \times 256 \times 3}$, a wide-angle camera image with a 100° horizontal FOV. Outputs of the image and the measurement encoder are concatenated and processed by three FC layers to generate $\mathbf{j}_{IL} \in \mathbb{R}^{256}$, which shares the same size as \mathbf{j}_{RL} . More details are found in the supplement.

4. Experiments

Benchmarks: All evaluations are completed on CARLA 0.9.11. We evaluate our methods on the NoCrash [12] and the offline LeaderBoard benchmark¹ [46]. Each benchmark specifies its training towns and weather, where the agent is allowed to collect data, and evaluates the agent in new towns and weather. The NoCrash benchmark considers generalization from Town 1, a European town composed of solely one-lane roads and T-junctions, to Town 2, a smaller version of Town 1 with different textures. By contrast, the LeaderBoard considers a more difficult generalization task in six maps that cover diverse traffic situations, including

¹In contrast to the Leaderboard online ranking, this benchmark is evaluated offline on the Leaderboard public routes (50 training, 26 testing).

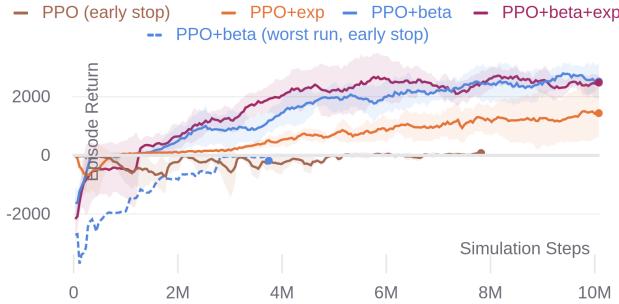


Figure 4: **Learning curves of RL experts** trained in CARLA Town 1-6. Solid lines show the mean and shaded areas show the standard deviation of episode returns across 3 seeds. The dashed line shows an outlier run that collapsed.

freeways, US-style junctions, roundabouts, stop signs, lane changing and merging. Following the NoCrash benchmark, we test generalization from four training weather types to two new weather types. But to save computational resources, only two out of the four training weather types are evaluated. The NoCrash benchmark comes with three levels of traffic density (empty, regular and dense), which defines the number of pedestrians and vehicles in each map. We focus on the NoCrash-dense and introduce a new level between regular and dense traffic, NoCrash-busy, to avoid congestion that often appears in the dense traffic setting. For the offline LeaderBoard the traffic density in each map is tuned to be comparable to the busy traffic setting.

Metrics: Our results are reported in success rate, the metric proposed by NoCrash, and driving score, a new metric introduced by the CARLA LeaderBoard. The success rate is the percentage of routes completed without collision or blockage. The driving score is defined as the product of route completion, the percentage of route distance completed, and infraction penalty, a discount factor that aggregates all triggered infractions. For example, if the agent ran two red lights in one route and the penalty coefficient for running one red light was 0.7, then the infraction penalty would be $0.7^2=0.49$. Compared to the success rate, the driving score is a fine-grained metric that considers more kinds of infractions and it is better suited to evaluate long-distance routes. More details about the benchmarks and the complete results are found in the supplement.

4.1. Performance of Experts

We use CARLA 0.9.10.1 to train RL experts and fine-tune our Autopilot, yet all evaluations are still on 0.9.11.

Sample Efficiency: To improve the sample efficiency of PPO, we propose to use BEV instead of camera images, Beta instead of Gaussian distributions, and the exploration loss in addition to the entropy loss. Since the benefit of using a BEV representation is obvious, here we only ablate

the Beta distribution and the exploration loss. As shown in Fig. 4, the baseline PPO with Gaussian distribution and entropy loss is trapped in a local minimum where staying still is the most rewarding strategy. Leveraging the exploration loss, PPO+exp can be successfully trained despite relatively high variance and low sample efficiency. The Beta distribution helps substantially, but without the exploration loss the training still collapsed in some cases due to insufficient exploration (cf. dashed blue line in Fig. 4). Our Roach (PPO+beta+exp) uses both Beta distribution and exploration loss to ensure stable and sample efficient training. The training takes around 1.7M steps in each of the six CARLA servers, this accounts for 10M steps in total, which takes roughly a week on an AWS EC2 g4dn.4xlarge or 4 days on a 2080 Ti machine with 12 cores.

Driving Performance: Table 1 compares different experts on the NoCrash-dense and on all 76 LeaderBoard routes under dynamic weather with busy traffic. Our Autopilot is a strong baseline expert that achieves a higher success rate than the Autopilot used in LBC and DA-RB. We evaluate three RL experts - (1) Roach, the proposed RL coach using Beta distribution and exploration prior. (2) PPO+beta, the RL coach trained without using the exploration prior. (3) PPO+exp, the RL coach trained without using the Beta distribution. In general, our RL experts achieve comparable success rates and higher driving scores than Autopilots because RL experts handle traffic lights in a better way (cf. Table 3). The two Autopilots often run red lights because they drive over-conservatively and wait too long at the junction, thus missing the green light. Among RL experts, PPO+beta and Roach, the two RL experts using a Beta distribution, achieve the best performance, while the difference between both is not significant. PPO+exp performs slightly worse, but it still achieves better driving scores than our Autopilot.

4.2. Performance of IL Agents

The performance of an IL agent is limited by the performance of the expert it is imitating. If the expert performs poorly, it is not sensible to compare IL agents imitating that expert. As shown in Table 1, this issue is evident in the NoCrash new town with dense traffic, where Autopilots do not perform well. To ensure a high performance upper-bound and hence a fair comparison, we conduct ablation studies (Fig. 5 and Table 3) under the busy traffic setting such that our Autopilot can achieve a driving score of 80% and a success rate of 90%. In order to compare with the state-of-the-art, the best model from the ablation studies is still evaluated on NoCrash with dense traffic in Table 2.

The input measurement vector \mathbf{m}_{IL} is different for the NoCrash and for the LeaderBoard. For NoCrash, \mathbf{m}_{IL} is just the speed. For the LeaderBoard, \mathbf{m}_{IL} contains additionally a 2D vector pointing to the next desired waypoint. This vector is computed from noisy GPS measurements and the desired

Suc. Rate % ↑	NCd-tt	NCd-tn	NCd-nt	NCd-nn	LB-all
PPO+exp	86 ± 6	86 ± 6	79 ± 6	77 ± 5	67 ± 3
PPO+beta	95 ± 3	95 ± 3	83 ± 5	87 ± 6	72 ± 5
Roach	91 ± 4	90 ± 7	83 ± 3	83 ± 3	72 ± 6
AP (ours)	95 ± 3	95 ± 3	83 ± 5	81 ± 2	75 ± 8
AP-lbc [9]	86 ± 3	83 ± 6	60 ± 3	59 ± 8	N/A
AP-darb [37]	71 ± 4	72 ± 3	41 ± 2	43 ± 2	N/A
Dri. Score % ↑	NCd-tt	NCd-tn	NCd-nt	NCd-nn	LB-all
PPO+exp	92 ± 2	92 ± 2	88 ± 3	86 ± 1	83 ± 0
PPO+beta	98 ± 2	98 ± 2	90 ± 3	92 ± 2	86 ± 2
Roach	95 ± 2	95 ± 3	91 ± 3	90 ± 2	85 ± 3
AP (ours)	86 ± 2	86 ± 2	70 ± 2	70 ± 1	78 ± 3

Table 1: **Success rate and driving score of experts.** Mean and standard deviation over 3 evaluation seeds. NCd: NoCrash-dense. tt: train town & weather. tn: train town & new weather. nt: new town & train weather. nn: new town & weather. LB-all: all 76 routes of LeaderBoard with dynamic weather. AP: CARLA Autopilot. For RL experts the best checkpoint among all training seeds and runs is used.

Success Rate % ↑	NCd-tt	NCd-tn	NCd-nt	NCd-nn
LBC [9] (0.9.6)	71 ± 5	63 ± 3	51 ± 3	39 ± 6
SAM [54] (0.8.4)	54 ± 3	47 ± 5	29 ± 3	29 ± 2
LSD [33] (0.8.4)	N/A	N/A	30 ± 4	32 ± 3
DA-RB ⁺ (E) [37]	66 ± 5	56 ± 1	36 ± 3	35 ± 2
DA-RB ⁺ [37] (0.8.4)	62 ± 1	60 ± 1	34 ± 2	25 ± 1
Our baseline, $\mathcal{L}_A(AP)$	88 ± 4	29 ± 3	32 ± 11	28 ± 4
Our best, $\mathcal{L}_K + \mathcal{L}_F(c)$	86 ± 5	82 ± 2	78 ± 5	78 ± 0

Table 2: **Success rate of camera-based end-to-end IL agents on NoCrash-dense.** Mean and standard deviation over 3 seeds. Our models are from DAGGER iteration 5. For DA-RB, + means triangular perturbations are added to the off-policy dataset, (E) means ensemble of all iterations.

route is specified as sparse GPS locations. The LeaderBoard instruction suggests that it is used to disambiguate situations where the semantics of left and right are not clear due to the complexity of the considered map.

Ablation: Fig. 5 shows driving scores of experts and IL agents at each DAGGER iteration on NoCrash and offline LeaderBoard with busy traffic. The baseline $\mathcal{L}_A(AP)$ is our implementation of DA-RB⁺ supervised by our Autopilot. Given our improved Autopilot, it is expected that $\mathcal{L}_A(AP)$ can achieve higher success rates than those reported in the DA-RB paper, but this is not observed in Table 2. The large performance gap between the Autopilot and $\mathcal{L}_A(AP)$ (cf. Fig. 5), especially while generalizing to a new town and new weather, indicates the limitation of this baseline.

By replacing the Autopilot with Roach, \mathcal{L}_A performs bet-

ter overall than $\mathcal{L}_A(AP)$. Further learning from the action distribution, \mathcal{L}_K generalizes better than \mathcal{L}_A on the NoCrash but not on the offline LeaderBoard. Feature matching only helps when j_{IL} is provided with the necessary information needed to reproduce j_{RL} . In our case, j_{RL} contains navigational information as the desired route is rendered in the BEV input. For the LeaderBoard, navigational information is partially encoded in m_{IL} , which includes the vector to the next desired waypoint, so better performance is observed by using \mathcal{L}_F . But for NoCrash this information is missing as m_{IL} is just the speed, hence it is impractical for j_{IL} to mimic j_{RL} and this causes the inferior performance of $\mathcal{L}_K + \mathcal{L}_F$ and $\mathcal{L}_K + \mathcal{L}_F + \mathcal{L}_V$. To confirm this hypothesis, we evaluate a single-branch network architecture where the measurement vector m_{IL} is augmented by the command encoded as a one-hot vector. Using feature matching with this architecture, $\mathcal{L}_K + \mathcal{L}_F(c)$ and $\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F(c)$ achieve the best driving score among IL agents in the NoCrash new town & weather generalization test, even outperforming the Autopilot.

Using value supervision in addition to feature matching helps the DAGGER process to converge faster as shown by $\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F$ and $\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F(c)$. However, without feature matching, using value supervision alone $\mathcal{L}_K + \mathcal{L}_V$ does not demonstrate superior performance. This indicates a potential synergy between feature matching and value estimation. Intuitively, the latent feature of Roach encodes the information needed for value estimation, hence mimicking this feature should help to predict the value, while value estimation could help to regularize feature matching.

Comparison with the State-of-the-art: In Table 2 we compare the baseline $\mathcal{L}_A(AP)$ and our best performing agent $\mathcal{L}_K + \mathcal{L}_F(c)$ with the state-of-the-art on the NoCrash-dense benchmark. Our $\mathcal{L}_A(AP)$ performs comparably to DA-RB⁺ except when generalizing to the new weather, where there is an incorrect rendering of after-rain puddles on CARLA 0.9.11 (see supplement for visualizations). This issue does not affect our best method $\mathcal{L}_K + \mathcal{L}_F(c)$ due to the stronger supervision of Roach. By mimicking the weather-agnostic Roach, the performance of our IL agent drops by less than 10% while generalizing to the new town and weather. Hence if the Autopilot is considered the performance upper-bound, it is fair to claim our approach saturates the NoCrash benchmark. However, as shown in Fig. 5, there is still space for improvement on NoCrash compared to Roach and the performance gap on the offline LeaderBoard highlights the importance of this new benchmark.

Performance and Infraction Analysis: Table 3 provides the detailed performance and infraction analysis on the NoCrash benchmark with busy traffic in the new town & weather setting. Most notably, the extremely high “Agent blocked” of our baseline $\mathcal{L}_A(AP)$ is due to reflections from after-rain puddles. This problem is largely alleviated by im-

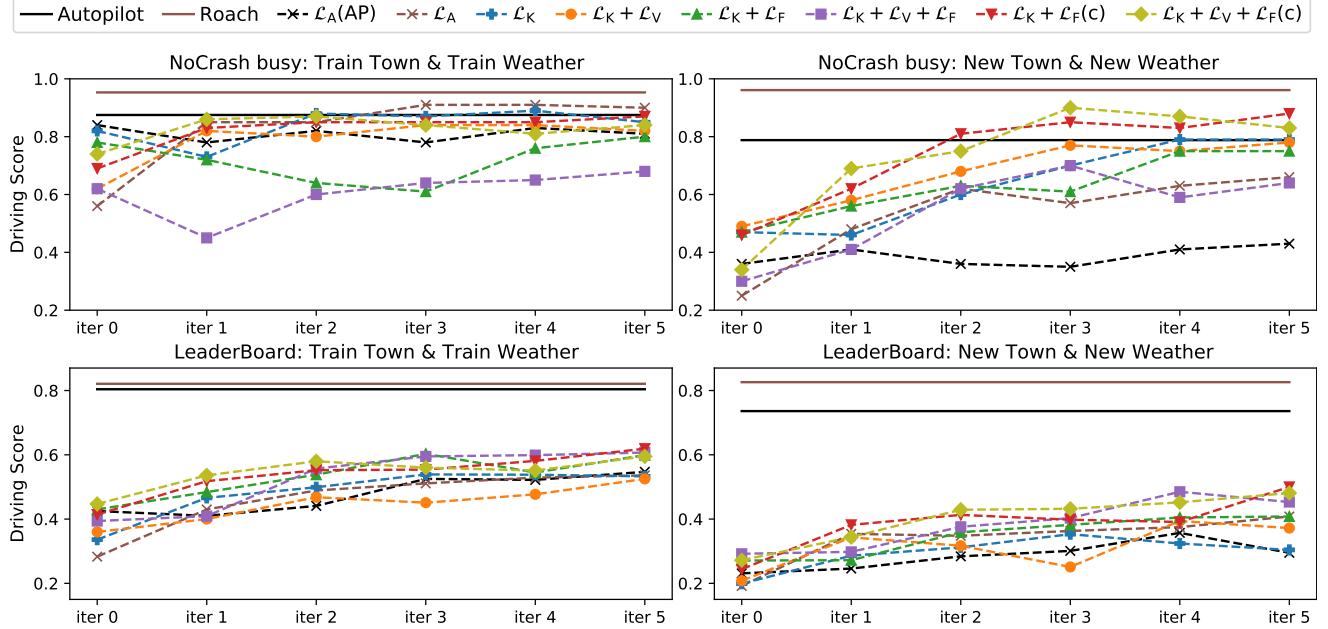


Figure 5: **Driving score of experts and IL agents.** All IL agents (dashed lines) are supervised by Roach except for $\mathcal{L}_A(AP)$, which is supervised by our Autopilot. For IL agents at the 5th iteration on NoCrash and all experts, results are reported as the mean over 3 evaluation seeds. Others are evaluated with one seed. The offline Leaderboard benchmark is used here.

	Success rate	Driving score	Route compl.	Infrac. penalty	Collision others	Collision pedestrian	Collision vehicle	Red light infraction	Agent blocked
iter 5	% , ↑	% , ↑	% , ↑	% , ↑	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓
$\mathcal{L}_A(AP)$	31 ± 7	43 ± 2	62 ± 6	77 ± 4	0.54 ± 0.53	0 ± 0	0.63 ± 0.50	3.33 ± 0.58	19.4 ± 14.4
\mathcal{L}_A	57 ± 7	66 ± 3	84 ± 3	76 ± 1	2.07 ± 1.37	0 ± 0	1.36 ± 1.10	1.4 ± 0.2	2.82 ± 1.45
\mathcal{L}_K	74 ± 3	79 ± 0	91 ± 2	86 ± 1	0.50 ± 0.25	0 ± 0	0.53 ± 0.18	0.68 ± 0.08	3.39 ± 0.20
$\mathcal{L}_K + \mathcal{L}_F(c)$	87 ± 5	88 ± 3	96 ± 0	91 ± 3	0.08 ± 0.04	0.01 ± 0.02	0.23 ± 0.08	0.61 ± 0.23	0.84 ± 0.04
Roach	95 ± 2	96 ± 3	100 ± 0	96 ± 3	0 ± 0	0.11 ± 0.07	0.04 ± 0.05	0.16 ± 0.20	0 ± 0
Autopilot	91 ± 1	79 ± 2	98 ± 1	80 ± 2	0 ± 0	0 ± 0	0.18 ± 0.08	1.93 ± 0.23	0.18 ± 0.08

Table 3: **Driving performance and infraction analysis of IL agents on NoCrash-busy, new town & new weather.** Mean and standard deviation over 3 evaluation seeds.

itating Roach, which drives more naturally, and \mathcal{L}_A shows an absolute improvement of 23% in terms of driving score. In other words this is the gain achieved by using a better expert, but the same imitation learning approach. Further using the improved supervision from soft targets and latent features results in our best model $\mathcal{L}_K + \mathcal{L}_F(c)$, which demonstrates another 22% absolute improvement. By handling red lights in a better way, this agent achieves 88%, an expert-level driving score, using a single camera image as input.

5. Conclusion

We present Roach, an RL expert, and an effective way to imitate this expert. Using the BEV representation, Beta distribution and the exploration loss, Roach sets the new performance upper-bound on CARLA while demonstrating

high sample efficiency. To enable a more effective imitation, we propose to learn from soft targets, values and latent features generated by Roach. Supervised by these informative targets, a baseline end-to-end IL agent using a single camera image as input can achieve state-of-the-art performance, even reaching expert-level performance on the NoCrash-dense benchmark. Future works include performance improvement on simulation benchmarks and real-world deployment. To saturate the LeaderBoard, the model capacity shall be increased [3, 18, 33]. To apply Roach to label real-world on-policy data, several sim-to-real gaps have to be addressed besides the photorealism, which is partially alleviated by the BEV. For urban driving simulators, the realistic behavior of road users is of utmost importance [45]. **Acknowledgements:** This work was funded by Toyota Motor Europe via the research project TRACE Zurich.

References

- [1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, page 1, 2004. 2
- [2] Alexander Amini, Igor Gilitschenski, Jacob Phillips, Julia Moseyko, Rohan Banerjee, Sertac Karaman, and Daniela Rus. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters (RA-L)*, 5(2):1143–1150, 2020. 1
- [3] Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. In *Robotics: Science and Systems XV*, 2019. 1, 2, 3, 8
- [4] Aseem Behl, Kashyap Chitta, Aditya Prakash, Eshed Ohn-Bar, and Andreas Geiger. Label efficient visual abstractions for autonomous driving. In *International Conference on Intelligent Robots and Systems (IROS)*, 2020. 2
- [5] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 1
- [6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. 12
- [7] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liang, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*, 2019. 1
- [8] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2722–2730, 2015. 2
- [9] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning (CoRL)*, pages 66–75, 2020. 2, 3, 7
- [10] Jianyu Chen, Bodi Yuan, and Masayoshi Tomizuka. Model-free deep reinforcement learning for urban autonomous driving. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2765–2771, 2019. 2, 3
- [11] Felipe Codevilla, Matthias Müller, Antonio López, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4693–4700, 2018. 1, 2
- [12] 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 (ICCV)*, pages 9329–9338, 2019. 2, 4, 5, 14
- [13] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the Conference on Robot Learning (CoRL)*, pages 1–16, 2017. 2
- [14] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, pages 1587–1596, 2018. 2
- [15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, pages 1861–1870, 2018. 2
- [16] Hadou van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, page 2094–2100. AAAI Press, 2016. 2
- [17] Simon Hecker, Dengxin Dai, Alexander Liniger, Martin Hahner, and Luc Van Gool. Learning accurate and human-like driving using semantic maps and attention. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 2346–2353, 2020. 1
- [18] Simon Hecker, Dengxin Dai, and Luc Van Gool. End-to-end learning of driving models with surround-view cameras and route planners. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 435–453, 2018. 8
- [19] Matteo Hessel, Joseph Modayil, Hadou Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2018. 2
- [20] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 5
- [21] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, 2016. 2
- [22] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning to steer by mimicking features from heterogeneous auxiliary networks. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2019. 3, 5
- [23] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska. One thousand and one hours: Self-driving motion prediction dataset. <https://level5.lyft.com/dataset/>, 2020. 1
- [24] Gregory Kahn, Pieter Abbeel, and Sergey Levine. LaND: Learning to navigate from disengagements. *IEEE Robotics and Automation Letters (R-AL)*, 2021. 1
- [25] 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. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254, 2019. 1
- [26] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2
- [27] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7345–7353, 2019. 2

- [28] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric Xing. Cirl: Controllable imitative reinforcement learning for vision-based self-driving. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 584–599, 2018. 1, 2
- [29] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2016. 2
- [30] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 1928–1937, 2016. 2
- [31] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 25(9):569–597, 2008. 1
- [32] Matthias Mueller, Alexey Dosovitskiy, Bernard Ghanem, and Vladlen Koltun. Driving policy transfer via modularity and abstraction. In *Proceedings of the Conference on Robot Learning (CoRL)*, volume 87, pages 1–15, 29–31 Oct 2018. 2
- [33] Eshed Ohn-Bar, Aditya Prakash, Aseem Behl, Kashyap Chitta, and Andreas Geiger. Learning situational driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11296–11305, 2020. 2, 7, 8
- [34] Praveen Palanisamy. Multi-agent connected autonomous driving using deep reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020. 3
- [35] Yunpeng Pan, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, Evangelos A. Theodorou, and Byron Boots. Agile autonomous driving using end-to-end deep imitation learning. In *Robotics: Science and Systems (RSS)*, 2018. 2, 3
- [36] Dean Pomerleau. ALVINN: An autonomous land vehicle in a neural network. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 305–313, December 1989. 1
- [37] Aditya Prakash, Aseem Behl, Eshed Ohn-Bar, Kashyap Chitta, and Andreas Geiger. Exploring data aggregation in policy learning for vision-based urban autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11763–11773, 2020. 1, 2, 4, 5, 7, 14
- [38] Charles R. Qi, Yin Zhou, Mahyar Najibi, Pei Sun, Khoa Vo, Boyang Deng, and Dragomir Anguelov. Offboard 3D Object Detection from Point Cloud Sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6134–6144, 2021. 2
- [39] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3. <https://github.com/DLR-RM/stable-baselines3>, 2019. 4
- [40] Nicholas Rhinehart, Rowan McAllister, and Sergey Levine. Deep imitative models for flexible inference, planning, and control. In *International Conference on Learning Representations (ICLR)*, 2020. 3
- [41] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 627–635, 2011. 1, 2, 4
- [42] Axel Sauer, Nikolay Savinov, and Andreas Geiger. Conditional affordance learning for driving in urban environments. In *Conference on Robot Learning (CoRL)*, pages 237–252, 2018. 2
- [43] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016. 4
- [44] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4
- [45] Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. TrafficSim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 8
- [46] CARLA team. Carla autonomous driving leaderboard. <https://leaderboard.carla.org/>, 2020. Accessed: 2021-02-11. 5
- [47] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. Is deep reinforcement learning really superhuman on atari? In *Deep Reinforcement Learning Workshop of the Conference on Neural Information Processing Systems*, 2019. 3, 4, 14
- [48] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7153–7162, 2020. 1, 3
- [49] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008. 1
- [50] Dequan Wang, Coline Devin, Qi-Zhi Cai, Philipp Krähenbühl, and Trevor Darrell. Monocular plan view networks for autonomous driving. In *International Conference on Intelligent Robots and Systems (IROS)*, 2019. 2
- [51] Huazhe Xu, Yang Gao, Fisher Yu, and Trevor Darrell. 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)*, pages 2174–2182, 2017. 1, 5
- [52] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2636–2645, 2020. 1

- [53] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8660–8669, 2019. 1
- [54] Albert Zhao, Tong He, Yitao Liang, Haibin Huang, Guy Van den Broeck, and Stefano Soatto. Sam: Squeeze-and-mimic networks for conditional visual driving policy learning. In *Conference on Robot Learning (CoRL)*, 2020. 2, 3, 5, 7

A. Summary

In the appendix, we provide (1) an overview of supplementary videos and codes, (2) implementation details of the RL experts and the IL agents, (3) details regarding benchmarks, and (4) additional experimental results.

B. Other Supplementary Materials

B.1. Videos

To investigate how different agents actually drive, we provide three videos. **roach.mp4** shows the driving performance of Roach, and highlights that it has a natural driving style and that it can handle complex traffic scenes. In **autopilot.mp4** we demonstrate the rule-based CARLA Autopilot. This agent uses unnatural brake actuation, i.e. it only uses emergency braking. Further, this video also highlights that in dense traffic, the rule-based agent can get stuck due to conservative danger predictions. For more details about the Autopilot and changes we made see Section C.3. Finally, in **il_agent.mp4** we demonstrate our best roach-supervised IL agent, showing that the agent can handle complex traffic scenes but also highlighting failure cases. In detail:

- **roach.mp4** is an *uncut* evaluation run recorded from Roach driving in Town03 (LeaderBoard-busy under dynamic weather). This video demonstrates the natural driving style of Roach even in challenging situations such as US-style traffic lights, unprotected left turns, roundabouts and stop signs.
- **autopilot.mp4** is an *uncut* evaluation run recorded from Autopilot driving in Town02 (NoCrash-dense, new town & new weather). This video demonstrates the over-conservative behavior of the Autopilot while driving in dense traffic. This often leads to red light infractions and blockage (both are present in the video).
- **il_agent.mp4** is a *highlight* video recorded from our best roach-supervised IL agent $\mathcal{L}_K + \mathcal{L}_F(c)$. This video includes multiple challenging situations often encountered during urban driving, such as EU and US-style junctions, unprotected left turns, roundabouts and reacting to pedestrians walking into the street. Furthermore, we highlight some of the failure modes of our camera-based IL agent, including not coming to a full stop for stop signs, collisions at overcrowded intersections and oscillation in the steering if the lane markings are not visible due to sun glare. We believe that including memory in the IL agent policy can help in most of these issues, due to a better understanding of the ego-motion (stop sign and oscillations) and other agents' motion (collisions).

B.2. Code

To reproduce our results, we provide four python scripts:

- *train_rl.py* for training Roach.
- *train_il.py* for training DA-RB (CILRS + DAGGER).
- *benchmark.py* for benchmarking agents.
- *data_collect.py* for collecting on/off-policy data.

It is recommended to run our scripts through bash files contained in the folder *run*. All configurations are in the folder *config*. Our repository is composed of two modules:

- *carla_gym*, a versatile OpenAI gym [6] environment for CARLA. It allows not only RL training with synchronized rollouts, but also data collection and evaluation. The environment is configurable in terms of weather, number of background pedestrians and vehicles, benchmarks, terminal conditions, sensors, rewards for the ego-vehicle and etc.
- *agents*, which includes our implementation of Autopilot (in *agents/expert*), Roach (in *agents/rl_birdview*) and DA-RB (in *agents/cilrs*).

B.3. Rendering issues

As illustrated in Fig. 6, on CARLA 0.9.11 reflections from after-rain puddles are sometimes wrongly rendered as black pixels. When the black pixels are accumulated, for example in the middle of Fig. 6a, they are often recognized as obstacles by the camera-based agents. Since this kind of reflection only appears under the testing weather but not under the training weather, generalizing to testing weather is exceptionally hard on CARLA 0.9.11 for the camera-based end-to-end IL agents.

C. Implementation Details

C.1. Roach

The network architecture of Roach can be found in Table 6 and the hyper-parameter values are listed in Table 8.

BEV: Cyclists and pedestrians are rendered larger than their actual sizes, this allows us to use a smaller image encoder with less parameters for Roach. Additionally, increasing the size naturally adds some caution when dealing with these vulnerable road users.

Update: The policy network and the value network are updated together using one Adam optimizer with an initial learning rate of 1e-5. The learning rate is scheduled based on the empirical KL-divergence between the policy before and after the update. If the KL-divergence is too large after an update epoch, the update phase will be interrupted



(a) Reflections from after-rain puddles in front of the ego-vehicle are incorrectly rendered as black pixels.



(b) Reflections are correctly rendered if the puddle is not directly in front of the ego-vehicle.

Figure 6: **Rendering issue of CARLA 0.9.11 running on Ubuntu with OpenGL.**

and a new rollout phase will start. Furthermore, a patience counter will be increased by one and the learning rate will be reduced once the patience counter reaches a threshold.

Rollout: Before each update phase a fixed-size buffer will be filled with trajectories collected on six CARLA servers, each corresponds to one of the six LeaderBoard maps (Town1-6).

Terminal Condition: An episode is terminated if and only if one of the following event happens.

- Run red light: examination code taken from the public repository of LeaderBoard. Terminal reward: $-1 - s$.
- Run stop sign: examination code taken from the public repository of LeaderBoard. Terminal reward: $-1 - s$.
- Collision registered by CARLA: based on the physics engine. Any collision with intensity larger than 0 is considered. Terminal reward: $-1 - s$.
- Collision detected by bounding box overlapping in the BEV. Terminal reward: $-1 - s$.
- Route deviation: triggered if the lateral distance to the lane centerline of the desired route is larger than 3.5 meters. Terminal reward: -1 .

- Blocked: speed of the ego-vehicle is slower than 0.1 m/s for more than 90 consecutive seconds. Terminal reward: -1 .

with s is the ego-vehicle's speed. The terminal reward is the reward given to the very last observation/action pair before the termination. For non-terminal samples, the terminal reward is 0.

Reward Shaping: The reward is the sum of the following components.

- r_{speed} : equals to $1.0 - |s - s_{\text{desired}}|/s_{\max}$, where s is the measured speed of the ego-vehicle, s_{\max} is the maximum speed and s_{desired} is the desired speed. We use a constant maximum speed $s_{\max} = 6$ m/s. The desired speed is a variable and is explained below.
- r_{position} : equals to $-0.5\Delta_p$, where Δ_p is the lateral distance (in meters) between the ego-vehicle's center and the center line of the desired route.
- r_{rotation} : equals to $-\Delta_r$, where Δ_r is the absolute value of the angular difference (in radians) between the ego-vehicle's heading and the heading of the center line of the desired route.

- r_{action} : equals to -0.1 if the current steering differs more than 0.01 from the steering applied in the previous step.
- r_{terminal} : the aforementioned terminal reward.

The desired speed, as proposed in [47], depends on rule-based obstacle detections. If there's no obstacle detected, the desired speed equals to the maximum speed. If an obstacle is detected, based on the distance to the obstacle the desired speed is linearly decreased to 0. As obstacle detector we use the hazard detection of Autopilot (cf. Section C.3). As a dense and informative reward, r_{speed} helps substantially to train our Roach and the camera-based end-to-end RL agent [47]. However, using rule-based obstacle detections inevitable introduces bias, the trained RL agent can be over-aggressive or over-conservative depending on the false positive and false negative rate of the detector. For example, during multi-lane freeway driving, our Roach decelerates for vehicles on the neighbouring lanes because those vehicles are detected as obstacles during training. Another example, Roach tends to collide after a right turn, this is related to the sector shaped (around 40 degrees) detection area used by the obstacle detection; vehicles and pedestrians on the right are not covered in the detection area. To further improve the performance of Roach, this r_{speed} should be modified, either using a better obstacle detector, or completely remove the rule-based obstacle detection, and build a less artificial reward based on simulation states.

Mode of Beta Distribution: We take the distribution mode as a deterministic output. The mode of the Beta distribution $\mathcal{B}(\alpha, \beta)$ is defined as

$$M = \begin{cases} \frac{\alpha-1}{\alpha+\beta-2} & \text{if } \alpha > 1, \beta > 1 \\ 0 & \text{if } \alpha \leq 1, \beta > 1 \\ 1 & \text{if } \alpha > 1, \beta \leq 1 \\ \text{bimodal } \{0, 1\} & \text{if } \alpha < 1, \beta < 1 \\ \text{any value in } [0, 1] & \text{if } \alpha = 1, \beta = 1 \end{cases} \quad (10)$$

For a natural driving behavior, we use the mean $\frac{\alpha}{\alpha+\beta}$ as the deterministic output when the mode is not uniquely defined, i.e. when $\alpha < 1, \beta < 1$ or $\alpha = 1, \beta = 1$.

C.2. IL Agent Supervised by Roach

The network architecture of our IL agent is found in Table 7 and the hyper-parameter values are listed in Table 9.

Network Architecture: We use six branches: turning left, turning right and going straight at the junction, following lane, changing to the left lane and changing to the right lane.

Off-policy Data Collection: Following CILRS [12], triangular perturbations on actions are applied while collecting the off-policy expert dataset to alleviate the covariate shift.

The off-policy dataset for NoCrash includes 80 episodes and for LeaderBoard it includes 160 episodes. Each episode is at most 300 seconds and at least 30 seconds long. The episode will be terminated if the expert violates any traffic rules, including red light infractions, stop sign infractions and collisions. In such a case, we remove the last 30 seconds of that episode so as to ensure that the off-policy dataset includes only correct demonstrations. Data is not collected using the given training routes but from randomly spawned start and target locations.

On-policy Data Collection: We follow DA-RB [37] for DAGGER with critical state sampling and replay buffer. New DAGGER-data will replace the old data in the replay buffer, while the buffer size is fixed. The same number of frames are contained in the replay buffer as in the off-policy dataset. At each DAGGER iteration, around 15-25% of the replay buffer is filled with new DAGGER-data, whereas at least 20% of the replay buffer is filled with off-policy data. Identical to the off-policy data collection, we use randomly spawned start and target locations while collecting DAGGER datasets. Following DA-RB, we did not use a mixed agent/expert policy to collect DAGGER datasets. However, our code allows this kind of rollout for DAGGER.

Training Details: Since we take the ResNet-34 pre-trained on ImageNet, the input image is normalized as suggested. In case the IL agent uses a distributional action head and/or a value head, the corresponding weights will be loaded from the Roach model at the first training iteration (the behavior cloning iteration). At each DAGGER iteration, the training continuous from the last epoch of the previous DAGGER iteration. We apply image augmentations using code modified from CILRS. The image augmentation methods are applied in random order and include Gaussian blur, additive Gaussian noise, coarse and block-wise dropouts, additive and multiplicative noise to each channel, randomized contrast and grayscale. All models are trained for 25 epochs using the ADAM optimizer with an initial learning rate of 2e-4. The learning rate is halved if the validation loss has not decreased for more than 5 epochs.

C.3. Autopilot

The CARLA Autopilot (also called roaming agent) is a simple but effective automated expert based on hand-crafted rules and ground-truth simulation states. The Autopilot is composed of two PID controllers for trajectory tracking and hazard detectors for emergency brake. Hazards include

- pedestrians/vehicles detected ahead,
- red lights/stop sings detected ahead,
- negative ego-vehicle speed, for handling slopes.

Locations and states of pedestrians, vehicles, red lights and stop signs are provided as ground-truth by the CARLA

API. If any hazard appears in a trigger area ahead of the ego-vehicle, Autopilot will make an emergency brake with $\text{throttle} = 0$, $\text{steering} = 0$, $\text{brake} = 1$. If no hazard is detected, the ego-vehicle will follow the desired path using two PID controllers, one for speed and one for steering control. The PID controller takes as input the location, rotation and speed of the ego-vehicle and the desired route specified as dense (1 meter interval) waypoints. The speed PID yields $\text{throttle} \in [0, 1]$ and the steering PID yields $\text{steering} \in [-1, 1]$. We tuned the parameters for PID controllers and hazard detectors manually, such that the Autopilot is a strong baseline. The target speed is 6 m/s.

D. Benchmarks

Scope: The scope of the NoCrash and the offline LeaderBoard benchmark are illustrated in Table 4. The offline LeaderBoard benchmark considers more traffic scenarios and longer routes in six different maps.

Weather: Following the NoCrash benchmark, we use *ClearNoon*, *WetNoon*, *HardRainNoon* and *ClearSunset* as the training weather types, whereas new weather types are *SoftRainSunset* and *WetSunset*. To save computational resources, only two out of the four training weather types are evaluated, they are *WetNoon* and *ClearSunset*.

Background Traffic: The number of vehicles and pedestrians spawned in each map of different benchmarks are listed in Table 5. Vehicles and pedestrians are spawned randomly from the complete blueprint library of CARLA 0.9.11. This stands in contrast to several previous works where for example two-wheeled vehicles are disabled.

Pros and cons of the online and the offline Leaderboard:

Online Leaderboard: (+) All methods are evaluated under exactly the same condition. (+) No need to re-evaluate other methods. (-) No restriction on how the method is trained and how the training data is collected.

Offline Leaderboard: (+) Strictly prescribes both the training and testing environment. (+) Full control and observation over the benchmark. (-) You will have to re-evaluate other methods, if any setup of the benchmark has changed, for example CARLA version and etc.

One can use the offline Leaderboard if a thorough study on the generalization ability of the method is desired.

E. Additional Experimental Results

To verify IL agents trained using the feature loss indeed embed camera images to the latent space of Roach, we report the feature loss at test time in Fig. 7. In the first row of Fig. 7, the IL agent trained without feature loss, \mathcal{L}_K , learns a latent space independent of the one of Roach. Hence, the test feature loss is effectively noise that is invariant to the test condition. In the second row, $\mathcal{L}_K + \mathcal{L}_F(c)$ is trained

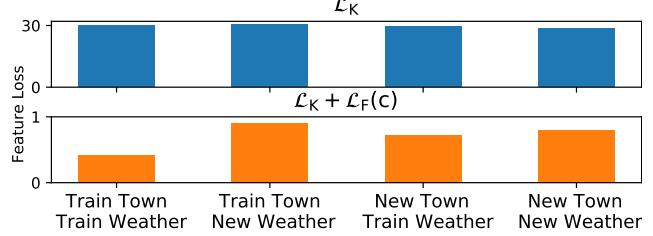


Figure 7: **Feature loss w.r.t. Roach** on one of the NoCrash-dense route. The y-axis of both charts have different scale.

Map	# Routes	Total Km	# Traffic lights	# Stop signs
NoCrash Train				
Town01	25	17.4	110	0
NoCrash Test				
Town02	25	8.9	94	0
LeaderBoard Train				
Town01	10	7.9	47	0
Town03	20	30.7	140	63
Town04	10	24.1	72	13
Town06	10	19.5	58	1
LeaderBoard Test				
Town02	6	5.5	54	0
Town04	10	24.1	72	13
Town05	10	12.4	82	29

Table 4: **Scope of the Nocrash benchmark and the offline LeaderBoard benchmark.** Total kilometers, number of traffic lights and stop signs are measured using Roach.

with the feature loss. The test feature loss of this agent is much smaller (less than 1) and increases as expected during the generalization tests.

To complete Fig. 5 of the main paper, driving scores of experts and IL agents at each DAGGER iterations are in Fig. 8 (NoCrash-busy) and Fig. 9 (LeaderBoard-busy).

To complete Table 3 of the main paper, detailed driving performance and infraction analysis of our experts and IL agents (5th DAGGER iteration) are listed in

- Table 10: NoCrash-busy, train town & train weather.
Table 11: NoCrash-busy, train town & new weather.
Table 12: NoCrash-busy, new town & train weather.
Table 13: NoCrash-busy, new town & new weather.
- Table 14: LeaderBoard, train town & train weather.
Table 15: LeaderBoard, train town & new weather.
Table 16: LeaderBoard, new town & train weather.
Table 17: LeaderBoard, new town & new weather.

Map	# Vehicles	# Pedestrians
NoCrash dense		
Town01	100	250
Town02	70	150
NoCrash busy		
Town01	120	120
Town02	70	70
LeaderBoard busy		
Town01	120	120
Town02	70	70
Town03	70	70
Town04	150	80
Town05	120	120
Town06	120	80

Table 5: **Background traffic settings for different benchmarks.**

Layer Type	Filters	Size	Strides	Activation
Image Encoder				
Conv2d	8	5x5	2	ReLU
Conv2d	16	5x5	2	ReLU
Conv2d	32	5x5	2	ReLU
Conv2d	64	3x3	2	ReLU
Conv2d	128	3x3	2	ReLU
Conv2d	256	3x3	1	-
Flatten				
Measurement Encoder				
Dense	256			ReLU
Dense	256			ReLU
FC Layers after Concatenation				
Dense	512			ReLU
Dense	256			ReLU
Action Head				
Dense (shared)	256			ReLU
Dense (shared)	256			ReLU
Dense (for α)	2			Softplus
Dense (for β)	2			Softplus
Value Head				
Dense	256			ReLU
Dense	256			ReLU
Dense	1			-

Table 6: **The network architecture used for Roach.** Around 1.53M trainable parameters.

Layer Type	Filters	Activation	Dropout
Image Encoder			
ResNet-34			
Measurement Encoder			
Dense	128	ReLU	
Dense	128	ReLU	
FC Layers after concatenation			
Dense	512	ReLU	
Dense	512	ReLU	
Dense	256	ReLU	
Speed Head			
Dense	256	ReLU	
Dense	256	ReLU	0.5
Dense	1		
Value Head			
Dense	256	ReLU	
Dense	256	ReLU	0.5
Dense	1		
Deterministic Action Head			
Dense	256	ReLU	
Dense	256	ReLU	0.5
Dense	2		
Distributional Action Head			
Dense (shared)	256	ReLU	
Dense (shared)	256	ReLU	0.5
Dense (for α)	2	Softplus	
Dense (for β)	2	Softplus	

Table 7: **The network architecture used for our IL agent.** Around 23.4M trainable parameters.

Notation	Description	Value
BEV Representation		
W	Width	192 px
H	Height	192 px
C	Number of channels	15
K	Size of the temporal sequence	4
D	Timestamps of images in the temporal sequence	{-1.5, -1, -0.5, 0} sec
	Distance from the ego-vehicle to the bottom	40 px
	Pixels per meter	5 px/m
	Minimum width/height of rendered bounding boxes	8 px
	Scale factor for bounding box size of pedestrians	2
Rollout		
	Buffer size for six environments	12288 frames
	Value bootstrap for the last non-terminal sample	True
	Synchronized	True
	Reset at the beginning of a new phase	False
	Weather	dynamic
	Range of vehicle/pedestrian number in Town 1	[0, 150]/[0, 300]
	Range of vehicle/pedestrian number in Town 2	[0, 100]/[0, 200]
	Range of vehicle/pedestrian number in Town 3	[0, 120]/[0, 120]
	Range of vehicle/pedestrian number in Town 4	[0, 160]/[0, 160]
	Range of vehicle/pedestrian number in Town 5	[0, 160]/[0, 160]
	Range of vehicle/pedestrian number in Town 6	[0, 160]/[0, 160]
Update		
	Number of epochs	20
λ_{ent}	Weight for the entropy loss	0.01
λ_{exp}	Weight for the exploration loss	0.05
	Weight for value loss	0.5
	γ for GAE	0.99
	λ for GAE	0.9
	Clipping range for PPO-clip	0.2
	Max norm for gradient clipping	0.5
	Batch size	256
	Initial learning rate	1e-5
	KL-divergence threshold for learning rate schedule	0.15
	Patience for learning rate schedule	8
	Factor for learning rate schedule	0.5

Table 8: **The hyper-parameter values used for Roach.**

Description	Value
Inputs	
Camera type	RGB
Camera image width	900 px
Camera image height	256 px
Camera location $[x, y, z]$ relative to the ego-vehicle	$[-1.5, 0, 2]$
Camera rotation $[roll, pitch, yaw]$ relative to the ego-vehicle	$[0, 0, 0]$
Camera horizontal FOV	100°
Mean for image normalization	$[0.485, 0.456, 0.406]$
Standard deviation for image normalization	$[0.229, 0.224, 0.225]$
Speed measurement	Forward speed in m/s
Normalization factor for speed	12
Data Collection	
Episode length	300 sec
Triangular perturbation for off-policy data	20%
Number of episodes (NoCrash, off-policy)	80
Number of episodes (LeaderBoard, off-policy)	160
Number of episodes (NoCrash, on-policy, Autopilot)	80
Number of episodes (LeaderBoard, on-policy, Autopilot)	160
Number of episodes (NoCrash, on-policy, Roach)	40
Number of episodes (LeaderBoard, on-policy, Roach)	80
DA-RB critical state sampling criterion	difference in acceleration
DA-RB critical state sampling threshold	0.2
Weather	Same as NoCrash train weathers
Range of vehicle/pedestrian number in NoCrash train town 1	$[0, 150]/[0, 200]$
Range of vehicle/pedestrian number in LeaderBoard train town 1	$[80, 160]/[80, 160]$
Range of vehicle/pedestrian number in LeaderBoard train town 3	$[40, 100]/[40, 100]$
Range of vehicle/pedestrian number in LeaderBoard train town 4	$[100, 200]/[40, 120]$
Range of vehicle/pedestrian number in LeaderBoard train town 6	$[80, 160]/[40, 120]$
Training	
Number of epochs at each DAGGER iteration	25
λ_S , weight for the speed regularization	0.05
λ_V , weight for the value loss, if applied	0.05
λ_F , weight for the feature loss, if applied	0.001
Batch size	48
Initial learning rate	0.0002
Patience for reduce-on-plateau learning rate schedule	5
Factor for learning rate schedule	0.5
Pre-trained distributional action head	True
Pre-trained value head	True
Image augmentation	True

Table 9: The hyper-parameter values used for our IL agent.

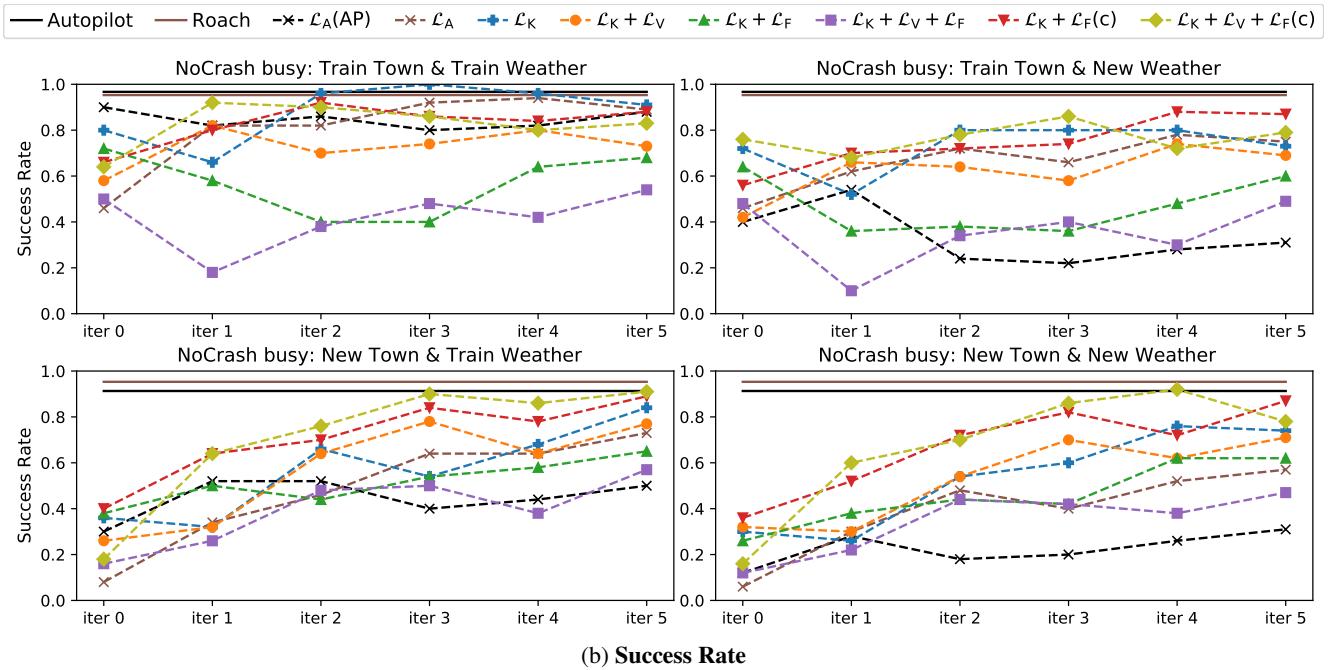
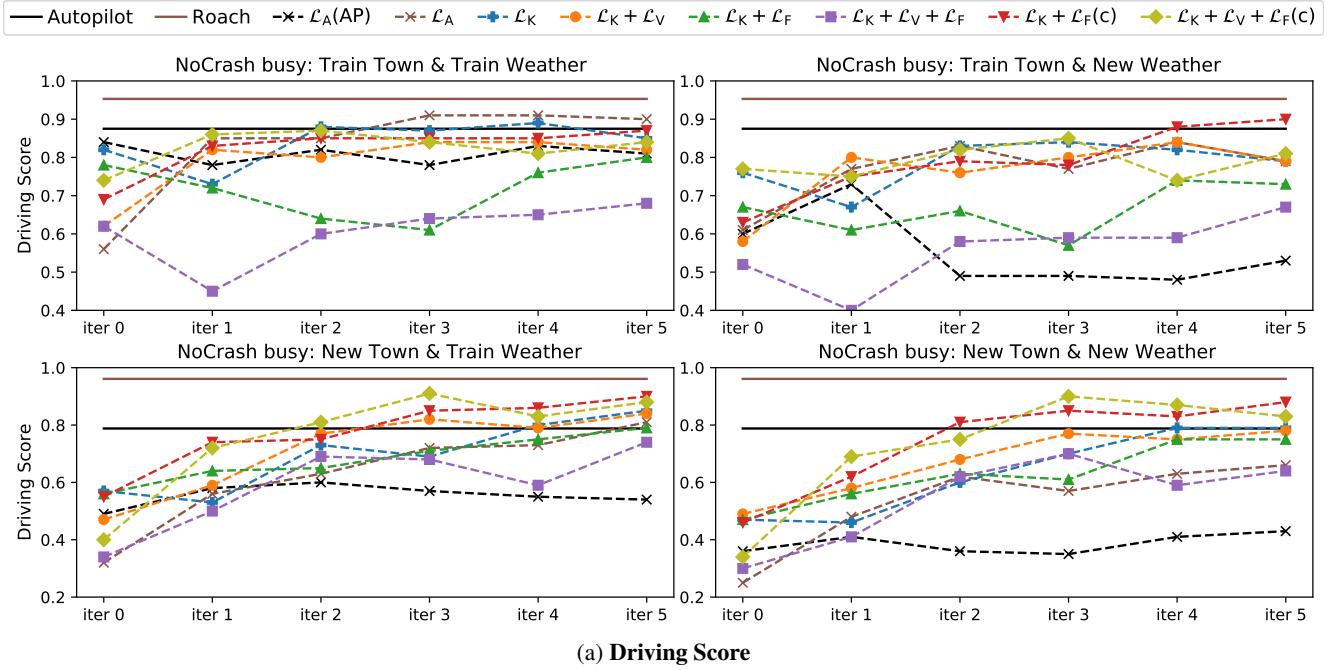


Figure 8: **Driving performance of experts and IL agents on the NoCrash-busy benchmark.** All IL agents (dashed lines) are supervised by Roach except for $\mathcal{L}_A(AP)$, which is supervised by the CARLA Autopilot. For IL agents at the 5th iteration and all experts, results are reported as the mean over 3 evaluation seeds. Others agents are evaluated only once.

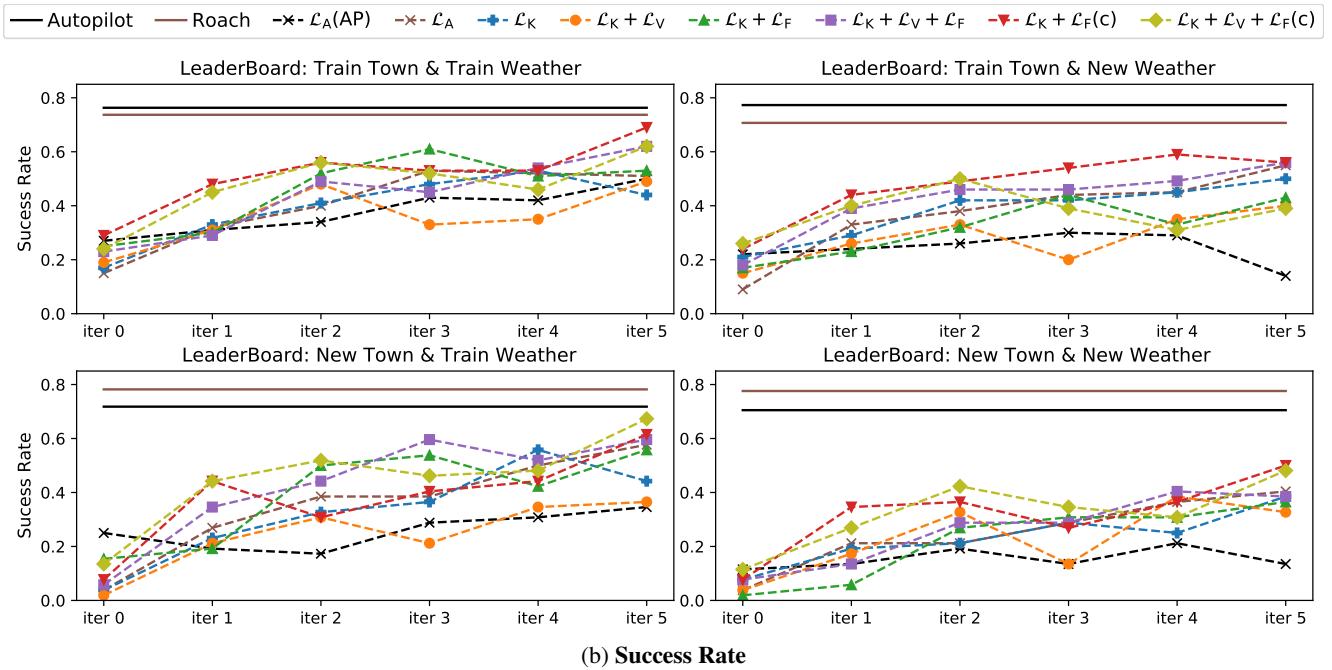
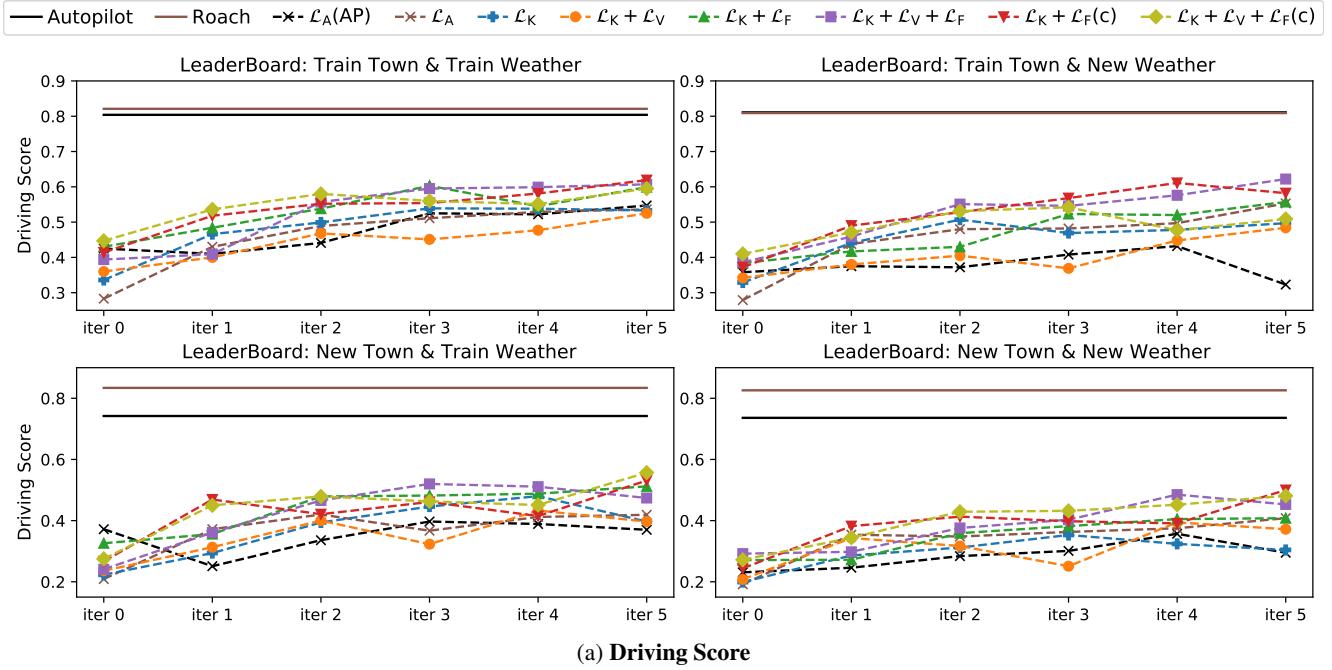


Figure 9: **Driving performance of experts and IL agents on the offline LeaderBoard-busy benchmark.** All IL agents (dashed lines) are supervised by Roach except for $\mathcal{L}_A(\text{AP})$, which is supervised by the CARLA Autopilot. For all experts, results are reported as the mean over 3 evaluation seeds. Results of IL agents are evaluated only once.

	Success rate	Driving score	Route compl.	Infrac. penalty	Collision others	Collision pedestrian	Collision vehicle	Red light infraction	Agent blocked
iter 5	%, ↑	%, ↑	%, ↑	%, ↑	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓
\mathcal{L}_A (AP)	88 ± 2	81 ± 2	94 ± 2	86 ± 1	0 ± 0	0 ± 0	0.08 ± 0.11	1.02 ± 0.33	1 ± 0.28
\mathcal{L}_A	89 ± 5	90 ± 2	99 ± 1	90 ± 1	0.06 ± 0.04	0.05 ± 0.02	0.06 ± 0.04	0.29 ± 0.03	0.05 ± 0.06
\mathcal{L}_K	91 ± 10	85 ± 6	99 ± 2	85 ± 5	0.1 ± 0.18	0.03 ± 0.04	0.1 ± 0.11	0.58 ± 0.07	0.07 ± 0.12
$\mathcal{L}_K + \mathcal{L}_V$	73 ± 4	82 ± 3	91 ± 2	91 ± 2	0.07 ± 0.07	0.02 ± 0.02	0.18 ± 0.12	0.27 ± 0.06	0.6 ± 0.2
$\mathcal{L}_K + \mathcal{L}_F$	68 ± 11	80 ± 6	89 ± 3	89 ± 4	0.15 ± 0.03	0.02 ± 0.01	0.05 ± 0.06	0.41 ± 0.13	0.12 ± 0.02
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F$	54 ± 2	68 ± 3	80 ± 2	87 ± 3	0.22 ± 0.34	0.06 ± 0.05	0.08 ± 0.05	0.53 ± 0.08	0.91 ± 0.32
$\mathcal{L}_K + \mathcal{L}_F(c)$	88 ± 2	87 ± 2	98 ± 1	88 ± 2	0.05 ± 0.08	0.07 ± 0.02	0.1 ± 0.07	0.41 ± 0.05	0.33 ± 0.49
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F(c)$	83 ± 1	84 ± 2	95 ± 1	89 ± 3	0 ± 0	0.04 ± 0.03	0.06 ± 0.06	0.5 ± 0.16	0.06 ± 0.06
Roach	95 ± 5	95 ± 1	100 ± 0	95 ± 1	0 ± 0	0.04 ± 0.04	0.03 ± 0.04	0.13 ± 0.11	0 ± 0
Autopilot	97 ± 2	87 ± 4	99 ± 2	88 ± 3	0 ± 0	0 ± 0	0.33 ± 0.55	0.89 ± 0.54	0.35 ± 0.58

Table 10: **Performance and infraction analysis on NoCrash-busy, train town & train weather.** Mean and std. over 3 seeds.

	Success rate	Driving score	Route compl.	Infrac. penalty	Collision others	Collision pedestrian	Collision vehicle	Red light infraction	Agent blocked
iter 5	%, ↑	%, ↑	%, ↑	%, ↑	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓
\mathcal{L}_A (AP)	31 ± 3	53 ± 2	61 ± 1	87 ± 1	0 ± 0	0 ± 0	0.35 ± 0.23	1.31 ± 0.36	5.75 ± 0.11
\mathcal{L}_A	75 ± 4	79 ± 5	92 ± 2	87 ± 4	0.13 ± 0.18	0.03 ± 0	0.06 ± 0.01	0.69 ± 0.19	0.79 ± 0.32
\mathcal{L}_K	73 ± 5	79 ± 5	91 ± 3	87 ± 3	0.02 ± 0.04	0 ± 0	0.24 ± 0.37	0.6 ± 0.18	0.95 ± 0.45
$\mathcal{L}_K + \mathcal{L}_V$	69 ± 4	79 ± 3	91 ± 1	87 ± 3	0.03 ± 0.05	0.04 ± 0.03	0.14 ± 0.07	0.5 ± 0.1	0.7 ± 0.05
$\mathcal{L}_K + \mathcal{L}_F$	60 ± 5	73 ± 2	80 ± 3	92 ± 1	0.05 ± 0.08	0.1 ± 0.16	0.09 ± 0.05	0.38 ± 0.03	0.02 ± 0.03
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F$	49 ± 8	67 ± 4	75 ± 4	90 ± 1	0.07 ± 0.13	0.03 ± 0.05	0.86 ± 1.41	0.88 ± 0.61	0.73 ± 0.17
$\mathcal{L}_K + \mathcal{L}_F(c)$	87 ± 5	90 ± 2	97 ± 2	93 ± 1	0 ± 0	0.01 ± 0.03	0.03 ± 0.06	0.37 ± 0.03	0.23 ± 0.13
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F(c)$	79 ± 3	81 ± 0	92 ± 1	89 ± 1	0 ± 0	0.01 ± 0.01	0.02 ± 0.02	0.57 ± 0.06	0.39 ± 0.17
Roach	95 ± 5	95 ± 1	100 ± 0	95 ± 1	0 ± 0	0.04 ± 0.04	0.03 ± 0.04	0.13 ± 0.11	0 ± 0
Autopilot	97 ± 2	87 ± 4	99 ± 2	88 ± 3	0 ± 0	0 ± 0	0.33 ± 0.55	0.89 ± 0.54	0.35 ± 0.58

Table 11: **Performance and infraction analysis on NoCrash-busy, train town & new weather.** Mean and std. over 3 seeds.

	Success rate	Driving score	Route compl.	Infrac. penalty	Collision others	Collision pedestrian	Collision vehicle	Red light infraction	Agent blocked
iter 5	%, ↑	%, ↑	%, ↑	%, ↑	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓
\mathcal{L}_A (AP)	50 ± 5	54 ± 1	79 ± 3	72 ± 3	0.88 ± 0.86	0 ± 0	0.08 ± 0.06	3.24 ± 0.35	3.76 ± 0.8
\mathcal{L}_A	73 ± 4	81 ± 4	94 ± 4	85 ± 2	1.03 ± 1.09	0.09 ± 0.05	0.72 ± 0.8	0.79 ± 0.12	1.24 ± 0.88
\mathcal{L}_K	84 ± 7	85 ± 4	97 ± 1	88 ± 4	0.25 ± 0.13	0.02 ± 0.03	0.3 ± 0.31	0.74 ± 0.18	0.37 ± 0.04
$\mathcal{L}_K + \mathcal{L}_V$	77 ± 10	84 ± 5	97 ± 3	86 ± 3	0.25 ± 0.28	0.02 ± 0.03	0.49 ± 0.13	0.73 ± 0.18	0.19 ± 0.24
$\mathcal{L}_K + \mathcal{L}_F$	65 ± 2	79 ± 2	88 ± 1	90 ± 3	0.31 ± 0.47	0.07 ± 0.07	0.37 ± 0.16	0.6 ± 0.19	0.3 ± 0.45
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F$	57 ± 4	74 ± 4	82 ± 1	90 ± 4	0.96 ± 0.2	0.04 ± 0.05	0.22 ± 0.16	0.43 ± 0.21	0.93 ± 0.23
$\mathcal{L}_K + \mathcal{L}_F(c)$	89 ± 5	90 ± 3	100 ± 1	90 ± 2	0.02 ± 0.03	0.08 ± 0.07	0.23 ± 0.11	0.59 ± 0.12	0.04 ± 0.08
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F(c)$	91 ± 5	88 ± 4	98 ± 1	89 ± 3	0.06 ± 0.06	0.01 ± 0.03	0.19 ± 0.08	0.78 ± 0.25	0.06 ± 0.06
Roach	95 ± 2	96 ± 3	100 ± 0	96 ± 3	0 ± 0	0.11 ± 0.07	0.04 ± 0.05	0.16 ± 0.2	0 ± 0
Autopilot	91 ± 1	79 ± 2	98 ± 1	80 ± 2	0 ± 0	0 ± 0	0.18 ± 0.08	1.93 ± 0.23	0.18 ± 0.08

Table 12: **Performance and infraction analysis on NoCrash-busy, new town & train weather.** Mean and std. over 3 seeds.

	Success rate	Driving score	Route compl.	Infrac. penalty	Collision others	Collision pedestrian	Collision vehicle	Red light infraction	Agent blocked
iter 5	%, ↑	%, ↑	%, ↑	%, ↑	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓
\mathcal{L}_A (AP)	31 ± 7	43 ± 2	62 ± 6	77 ± 4	0.54 ± 0.53	0 ± 0	0.63 ± 0.50	3.33 ± 0.58	19.4 ± 14.4
\mathcal{L}_A	57 ± 7	66 ± 3	84 ± 3	76 ± 1	2.07 ± 1.37	0 ± 0	1.36 ± 1.10	1.4 ± 0.2	2.82 ± 1.45
\mathcal{L}_K	74 ± 3	79 ± 0	91 ± 2	86 ± 1	0.50 ± 0.25	0 ± 0	0.53 ± 0.18	0.68 ± 0.08	3.39 ± 0.20
$\mathcal{L}_K + \mathcal{L}_V$	71 ± 9	78 ± 3	91 ± 1	85 ± 3	0.55 ± 0.22	0.11 ± 0.06	0.34 ± 0.31	0.72 ± 0.09	1.14 ± 0.10
$\mathcal{L}_K + \mathcal{L}_F$	62 ± 2	75 ± 1	85 ± 0	87 ± 2	0.79 ± 0.61	0.03 ± 0.05	0.73 ± 0.16	0.63 ± 0.02	2.04 ± 1.33
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F$	47 ± 9	64 ± 6	72 ± 5	89 ± 3	0.9 ± 0.73	0.03 ± 0.06	0.38 ± 0.26	0.79 ± 0.42	1.29 ± 0.9
$\mathcal{L}_K + \mathcal{L}_F(c)$	87 ± 5	88 ± 3	96 ± 0	91 ± 3	0.08 ± 0.04	0.01 ± 0.02	0.23 ± 0.08	0.61 ± 0.23	0.84 ± 0.04
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F(c)$	78 ± 3	83 ± 1	94 ± 2	89 ± 2	0.21 ± 0.14	0 ± 0	0.16 ± 0.05	0.79 ± 0.15	0.46 ± 0.13
Roach	95 ± 2	96 ± 3	100 ± 0	96 ± 3	0 ± 0	0.11 ± 0.07	0.04 ± 0.05	0.16 ± 0.20	0 ± 0
Autopilot	91 ± 1	79 ± 2	98 ± 1	80 ± 2	0 ± 0	0 ± 0	0.18 ± 0.08	1.93 ± 0.23	0.18 ± 0.08

Table 13: **Performance and infraction analysis on NoCrash-busy, new town & new weather.** Mean and std. over 3 seeds.

	Success rate	Driving score	Route compl.	Infrac. penalty	Collision others	Collision pedestrian	Collision vehicle	Red light infraction	Stop Sign infraction	Agent blocked
iter 5	%, ↑	%, ↑	%, ↑	%, ↑	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓
\mathcal{L}_A (AP)	50	55	82	68	0.24	0.01	0.38	0.53	0.22	1.39
\mathcal{L}_A	51	54	87	60	0.46	0.19	0.30	0.50	0.39	0.48
\mathcal{L}_K	44	53	86	63	0.14	0.07	0.35	0.42	0.38	0.77
$\mathcal{L}_K + \mathcal{L}_V$	49	53	81	66	0.39	0.04	0.30	0.36	0.40	1.35
$\mathcal{L}_K + \mathcal{L}_F$	53	60	85	71	0.11	0.10	0.20	0.25	0.32	0.47
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F$	62	61	94	65	0.01	0.05	0.30	0.37	0.42	0.47
$\mathcal{L}_K + \mathcal{L}_F(c)$	69	62	94	66	0.05	0.04	0.15	0.35	0.59	0.40
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F(c)$	62	59	95	63	0.04	0.41	0.21	0.33	0.50	0.45
Roach	74 ± 1	82 ± 2	95 ± 1	86 ± 2	0.03 ± 0.02	0.04 ± 0.03	0.12 ± 0.04	0.13 ± 0.05	0 ± 0.01	0.13 ± 0.04
Autopilot	76 ± 1	80 ± 1	96 ± 1	84 ± 2	0 ± 0	0 ± 0	0.16 ± 0.05	0.3 ± 0.05	0 ± 0.01	0.16 ± 0.07

Table 14: **Performance and infraction analysis on the offline LeaderBoard, train town & train weather.** Mean and std. over 3 seeds.

	Success rate	Driving score	Route compl.	Infrac. penalty	Collision others	Collision pedestrian	Collision vehicle	Red light infraction	Stop Sign infraction	Agent blocked
iter 5	%, ↑	%, ↑	%, ↑	%, ↑	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓
$\mathcal{L}_A(AP)$	14	32	47	79	0.23	0.00	0.31	0.55	0.32	31.79
\mathcal{L}_A	55	55	87	64	0.14	0.03	0.26	0.37	0.47	0.43
\mathcal{L}_K	50	50	87	58	0.08	0.06	0.42	0.57	0.62	0.61
$\mathcal{L}_K + \mathcal{L}_V$	40	48	79	64	0.13	0.02	0.37	0.48	0.45	0.80
$\mathcal{L}_K + \mathcal{L}_F$	43	56	82	70	0.11	0.03	0.20	0.34	0.31	0.66
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F$	56	62	91	69	0.06	0.05	0.15	0.29	0.39	0.31
$\mathcal{L}_K + \mathcal{L}_F(c)$	56	58	90	66	0.07	0.05	0.19	0.36	0.60	0.36
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F(c)$	39	51	88	59	0.10	0.03	0.29	0.38	0.54	0.47
Roach	71 ± 2	81 ± 1	95 ± 1	85 ± 0	0.02 ± 0.02	0.04 ± 0.03	0.14 ± 0.02	0.12 ± 0.04	0 ± 0.01	0.14 ± 0.07
Autopilot	77 ± 2	81 ± 1	96 ± 1	85 ± 2	0 ± 0	0 ± 0	0.16 ± 0.04	0.28 ± 0.06	0 ± 0.01	0.22 ± 0.13

Table 15: **Performance and infraction analysis on the offline LeaderBoard, train town & new weather.** Mean and std. over 3 seeds.

	Success rate	Driving score	Route compl.	Infrac. penalty	Collision others	Collision pedestrian	Collision vehicle	Red light infraction	Stop Sign infraction	Agent blocked
iter 5	%, ↑	%, ↑	%, ↑	%, ↑	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓
$\mathcal{L}_A(AP)$	35	37	75	55	0.17	0.00	1.52	1.00	0.50	3.64
\mathcal{L}_A	58	42	92	46	0.17	0.04	0.42	0.82	0.75	0.29
\mathcal{L}_K	44	40	91	44	0.91	0.04	0.36	1.21	0.75	0.94
$\mathcal{L}_K + \mathcal{L}_V$	37	40	76	58	0.13	0.05	0.45	0.80	0.40	0.33
$\mathcal{L}_K + \mathcal{L}_F$	56	51	91	56	0.22	0.07	0.34	0.45	0.61	0.16
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F$	60	47	95	50	0.00	0.09	0.81	0.64	0.74	1.29
$\mathcal{L}_K + \mathcal{L}_F(c)$	62	53	94	56	0.00	0.04	1.22	0.71	0.70	1.04
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F(c)$	67	56	95	58	0.03	0.05	0.31	0.38	0.72	0.11
Roach	78 ± 4	83 ± 2	97 ± 1	86 ± 2	0 ± 0	0.03 ± 0.02	0.13 ± 0.1	0.16 ± 0.03	0 ± 0	0.09 ± 0.04
Autopilot	72 ± 13	74 ± 5	95 ± 2	78 ± 3	0 ± 0	0 ± 0	0.14 ± 0.07	0.57 ± 0.13	0 ± 0	0.18 ± 0.14

Table 16: **Performance and infraction analysis on the offline LeaderBoard, new town & train weather.** Mean and std. over 3 seeds.

	Success rate	Driving score	Route compl.	Infrac. penalty	Collision others	Collision pedestrian	Collision vehicle	Red light infraction	Stop Sign infraction	Agent blocked
iter 5	%, ↑	%, ↑	%, ↑	%, ↑	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓	#/Km, ↓
$\mathcal{L}_A(\text{AP})$	14	30	42	80	0.06	0.05	1.11	0.63	0.49	28.27
\mathcal{L}_A	40	41	90	45	0.36	0.10	0.61	0.88	0.67	0.35
\mathcal{L}_K	39	30	86	39	0.17	0.03	0.42	1.31	0.81	0.51
$\mathcal{L}_K + \mathcal{L}_V$	33	37	78	53	0.09	0.06	0.47	0.97	0.54	0.40
$\mathcal{L}_K + \mathcal{L}_F$	37	41	81	54	0.29	0.03	0.79	0.61	0.68	1.23
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F$	39	45	85	56	0.02	0.01	1.54	0.73	0.64	2.30
$\mathcal{L}_K + \mathcal{L}_F(c)$	50	50	86	60	0.01	0.02	0.48	0.60	0.63	2.64
$\mathcal{L}_K + \mathcal{L}_V + \mathcal{L}_F(c)$	48	48	90	56	0.02	0.04	0.18	0.60	0.81	0.47
Roach	78 ± 4	83 ± 2	97 ± 1	85 ± 2	0 ± 0	0.04 ± 0.02	0.13 ± 0.1	0.18 ± 0.06	0 ± 0	0.09 ± 0.04
Autopilot	71 ± 11	74 ± 4	95 ± 1	78 ± 3	0 ± 0	0 ± 0	0.14 ± 0.07	0.58 ± 0.12	0 ± 0	0.2 ± 0.12

Table 17: **Performance and infraction analysis on the offline LeaderBoard, new town & new weather.** Mean and std. over 3 seeds.