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

Zhejun Zhang[1], Alexander Liniger[1], Dengxin Dai[1,2], Fisher Yu[1] and Luc Van Gool[1,3]
[1]Computer Vision Lab, ETH Zürich, [2]MPI for Informatics, [3]PSI, KU Leuven
{zhejun.zhang,alex.liniger,dai,vangool}@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 more challenging CARLA LeaderBoard.

1. Introduction

Even though nowadays, most autonomous driving (AD) stacks [30, 48] use individual modules for perception, planning and control, end-to-end approaches have been proposed since the 80's [35] and the success of deep learning brought them back into the research spotlight [5, 50]. Numerous works have studied different network architectures for this task [3, 16, 52], yet most of these approaches use supervised learning with expert demonstrations, which is known to suffer from covariate shift [36, 40]. While data augmentation based on view synthesis [2, 5, 35] 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

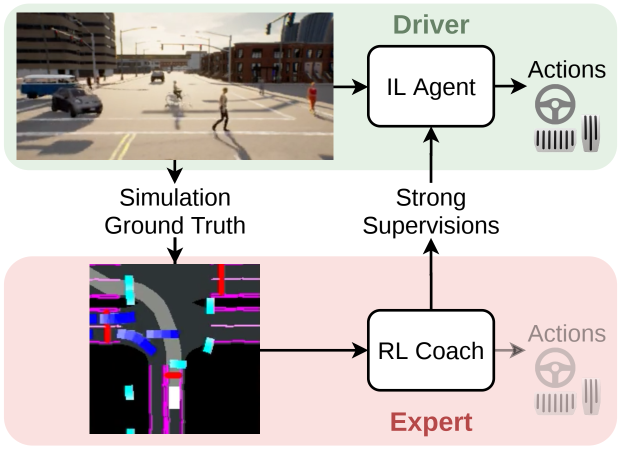


Figure 1: **Roach: RL coach** allows IL agents to benefit from dense and informative on-policy supervisions.

mimic the experts' behavior [3, 10], reinforcement learning (RL) methods often use expert demonstrations to improve sample efficiency by pre-training part of the model via supervised learning [27, 47]. 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 [6, 22, 51]. *(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, 23, 24] 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 [20] 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. [34] 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, [36] showed that a similar concept can be applied to the driving simulator CARLA [12]. 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 [18] and continuous control [14], 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 [26, 49] 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 [31], and on the other hand, on-policy labeling does not have to happen in real-time or even onboard. Hence 3D detection becomes easier given the complete sequences [37].

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 CARLA LeaderBoard and the CARLA NoCrash benchmark using a single camera based end-to-end IL agent, which is supervised by Roach using our improved training scheme. Our repository is publically 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. [12] 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 [10] and CILRS [11] addressed directional multi-modality 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 [36] applied DAGGER [40] with critical state sampling to CILRS. Most recently, LSD [32] 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 [41] combines CIL and direct perception [7] by mapping camera images to driving affordances which can be directly used by a rule-based low-level controller. LBC [8] 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 [53] 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 [29] was demonstrated in [12], yet its performance is lower than that of other methods presented in the same paper. CIRL [27] proposed an end-to-end DDPG [28] agent with its actor network pre-trained via behavior cloning to accelerate online training. To reduce the problem complexity, Chen et al. [9] investigated DDQN [15], TD3 [13] and SAC [14] using BEV as an input and pre-trained the image encoder with a variational auto-encoder [25] on expert trajectories. State-of-the-art performance is achieved in [47] using Rainbow-IQN [46]. 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 [39] and multi-agent RL on CARLA was discussed in [33]. 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 [34] 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 [8] and SAM [53]. 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 [21]. 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 $\mathbf{i}_{\text{RL}}$ and a measurement vector $\mathbf{m}_{\text{RL}}$ 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, 8, 9]. 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 $\mathbf{i}_{\text{RL}}$. 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 $\mathit{steering} \in [-1,1]$, $\mathit{throttle} \in [0,1]$ and $\mathit{brake} \in [0,1]$. 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 [8, 39]. 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,

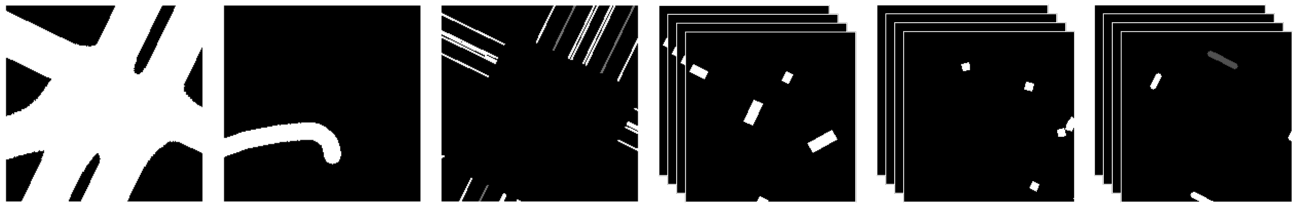


Figure 2: **The BEV representation used by our Roach.**

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

Training: We use proximal policy optimization (PPO) [43] 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 $\tau = \{(\mathbf{i}_{\text{RL},k}, \mathbf{m}_{\text{RL},k}, \mathbf{a}_k, r_k)_{k=0}^T, z\}$ includes BEV images $\mathbf{i}_{\text{RL}}$, measurement vectors $\mathbf{m}_{\text{RL}}$, 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

$$\theta_{k+1} = \arg\max_\theta \mathop{\mathrm{E}}_{\tau\sim\pi_{\theta_k}} \left[\mathcal{L}_{\text{ppo}} + \mathcal{L}_{\text{ent}} + \mathcal{L}_{\text{exp}}\right]. \tag{1}$$

The first objective $\mathcal{L}_{\text{ppo}}$ is the clipped policy gradient loss with advantages estimated using generalized advantage estimation [42]. The second objective $\mathcal{L}_{\text{ent}}$ is a maximum entropy loss commonly employed to encourage exploration

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

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

$$\mathrm{H}\left(\pi_\theta\right) = -\mathrm{KL}\left(\pi_\theta \parallel \mathcal{U}(-1,1)\right), \tag{3}$$

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

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

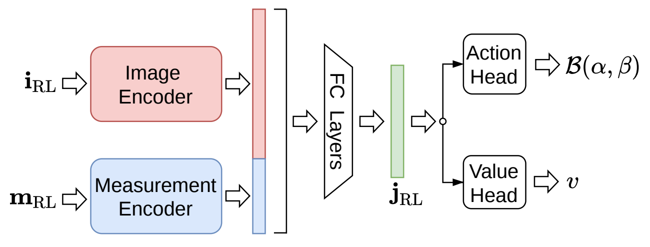
where $\mathbb{1}$ is the indicator function and $z \in \mathcal{Z}$ is the event that ends the episode. The terminal condition set $\mathcal{Z}$ includes collision, running traffic light/sign, route deviation and being blocked. Unlike $\mathcal{L}_{\text{ent}}$ which imposes a uniform prior on the actions at all time steps regardless of which z is triggered, $\mathcal{L}_{\text{exp}}$ 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 [38] 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 $\mathbf{j}_{\text{RL}}$, 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 [46] 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.

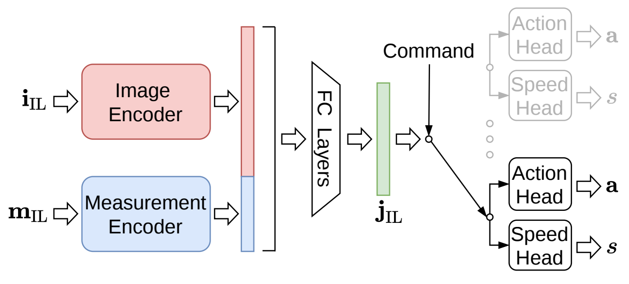
3.2. IL Agents Supervised by Roach

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 [36] (CILRS [11] + DAGGER [40]) 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 $\mathbf{i}_{\text{IL}}$ and a measurement module that encodes the measurement vector $\mathbf{m}_{\text{IL}}$. Outputs of both modules are concatenated and processed by FC layers to generate a bottleneck latent feature $\mathbf{j}_{\text{IL}}$. Navigation instructions are given



(a) Roach



(b) CILRS

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 $\mathbf{j}_{\text{IL}}$ is processed by the branch selected by the command. The imitation objective of CILRS consists of an L1 action loss

$$\mathcal{L}_{\text{A}} = \|\hat{\mathbf{a}} - \mathbf{a}\|_1 \tag{5}$$

and a speed prediction regularization

$$\mathcal{L}_{\text{S}} = \lambda_{\text{S}} \cdot |\hat{s} - s|, \tag{6}$$

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 [19] 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 $\mathcal{L}_{\text{A}}$. The action head of CILRS is modified to predict distribution parameters and the loss is formulated as a KL-divergence

$$\mathcal{L}_{\text{K}} = \text{KL}(\hat{\pi} \| \pi) \tag{7}$$

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 [21, 53]. The latent feature $\mathbf{j}_{\text{RL}}$ 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, $\mathbf{j}_{\text{RL}}$ is invariant to rendering and weather as Roach uses the BEV representation. Learning to embed camera images to the latent space of $\mathbf{j}_{\text{RL}}$ should help IL agents to generalize to new weather and new situations. Hence, we propose the feature loss

$$\mathcal{L}_{\text{F}} = \lambda_{\text{F}} \cdot \|\mathbf{j}_{\text{RL}} - \mathbf{j}_{\text{IL}}\|_2^2. \tag{8}$$

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 [50], 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,

$$\mathcal{L}_{\text{V}} = \lambda_{\text{V}} \cdot (\hat{v} - v)^2. \tag{9}$$

Implementation Details: Our implementation follows DA-RB [36]. We choose a Resnet-34 pretrained on ImageNet as the image encoder to generate a 1000-dimensional feature given $\mathbf{i}_{\text{RL}} \in [0,1]^{900\times256\times3}$, 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 $\mathbf{j}_{\text{IL}} \in \mathbb{R}^{256}$, which shares the same size as $\mathbf{j}_{\text{RL}}$. 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 [11] and the latest LeaderBoard benchmark[1] [45]. 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 freeways, US-style junctions, roundabouts, stop signs, lane

[1]We use the 50 public training routes and the 26 public testing routes.

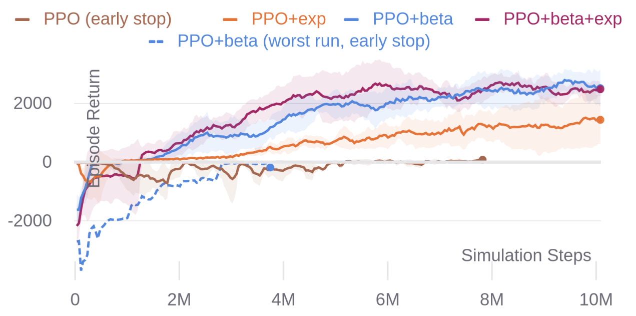


Figure 4: **Learning curves of RL experts** trained in CARLA Town 1-6. Solid lines show the mean and shaded areas show the standard deviation of episode returns across 3 seeds. The dashed line shows an outlier run that collapsed.

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 CARLA 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 $\mathbf{m}_{\mathrm{IL}}$ is different for the NoCrash and for the LeaderBoard. For NoCrash, $\mathbf{m}_{\mathrm{IL}}$ is just the speed. For the LeaderBoard, $\mathbf{m}_{\mathrm{IL}}$ contains additionally a 2D vector pointing to the next desired waypoint. This vector is computed from noisy GPS measurements and the desired route is specified as sparse GPS locations. The LeaderBoard

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	$\mathbf{95} \pm 3$	$\mathbf{95} \pm 3$	83 ± 5	$\mathbf{87} \pm 6$	72 ± 5
Roach	91 ± 4	90 ± 7	$\mathbf{83} \pm 3$	83 ± 3	72 ± 6
AP (ours)	$\mathbf{95} \pm 3$	$\mathbf{95} \pm 3$	83 ± 5	81 ± 2	$\mathbf{75} \pm 8$
AP-lbc [8]	86 ± 3	83 ± 6	60 ± 3	59 ± 8	N/A
AP-darb [36]	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	$\mathbf{98} \pm 2$	$\mathbf{98} \pm 2$	90 ± 3	$\mathbf{92} \pm 2$	$\mathbf{86} \pm 2$
Roach	95 ± 2	95 ± 3	$\mathbf{91} \pm 3$	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 [8] (0.9.6)	71 ± 5	63 ± 3	51 ± 3	39 ± 6
SAM [53] (0.8.4)	54 ± 3	47 ± 5	29 ± 3	29 ± 2
LSD [32] (0.8.4)	N/A	N/A	30 ± 4	32 ± 3
DA-RB$^+$(E) [36]	66 ± 5	56 ± 1	36 ± 3	35 ± 2
DA-RB$^+$ [36] (0.8.4)	62 ± 1	60 ± 1	34 ± 2	25 ± 1
Our baseline, $\mathcal{L}_\text{A}$(AP)	$\mathbf{88} \pm 4$	29 ± 3	32 ± 11	28 ± 4
Our best, $\mathcal{L}_\text{K} + \mathcal{L}_\text{F}(\text{c})$	86 ± 5	$\mathbf{82} \pm 2$	$\mathbf{78} \pm 5$	$\mathbf{78} \pm 0$

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.

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 the NoCrash and LeaderBoard with busy traffic. The baseline $\mathcal{L}_\text{A}(\text{AP})$ is our implementation of DA-RB$^+$ supervised by our Autopilot. Given our improved Autopilot, it is expected that $\mathcal{L}_\text{A}(\text{AP})$ 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 $\mathcal{L}_\text{A}(\text{AP})$ (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, $\mathcal{L}_\text{A}$ performs better overall than $\mathcal{L}_\text{A}(\text{AP})$. Further learning from the action distribution, $\mathcal{L}_\text{K}$ generalizes better than $\mathcal{L}_\text{A}$ on the NoCrash but not on the LeaderBoard. Feature matching only helps when $\mathbf{j}_\text{IL}$ is provided with the necessary information needed to reproduce $\mathbf{j}_\text{RL}$. In our case, $\mathbf{j}_\text{RL}$ contains navigational information as the desired route is rendered in the BEV input. For the LeaderBoard, navigational information is partially encoded in $\mathbf{m}_\text{IL}$, which includes the vector to the next desired waypoint, so better performance is observed by using $\mathcal{L}_\text{F}$. But for NoCrash this information is missing as $\mathbf{m}_\text{IL}$ is just the speed, hence it is impractical for $\mathbf{j}_\text{IL}$ to mimic $\mathbf{j}_\text{RL}$ and this causes the inferior performance of $\mathcal{L}_\text{K} + \mathcal{L}_\text{F}$ and $\mathcal{L}_\text{K} + \mathcal{L}_\text{F} + \mathcal{L}_\text{V}$. To confirm this hypothesis, we evaluate a single-branch network architecture where the measurement vector $\mathbf{m}_\text{IL}$ is augmented by the command encoded as a one-hot vector. Using feature matching with this architecture, $\mathcal{L}_\text{K} + \mathcal{L}_\text{F}(\text{c})$ and $\mathcal{L}_\text{K} + \mathcal{L}_\text{V} + \mathcal{L}_\text{F}(\text{c})$ 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 $\mathcal{L}_\text{K} + \mathcal{L}_\text{V} + \mathcal{L}_\text{F}$ and $\mathcal{L}_\text{K} + \mathcal{L}_\text{V} + \mathcal{L}_\text{F}(\text{c})$. However, without feature matching, using value supervision alone $\mathcal{L}_\text{K} + \mathcal{L}_\text{V}$ 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 $\mathcal{L}_\text{A}(\text{AP})$ and our best performing agent $\mathcal{L}_\text{K} + \mathcal{L}_\text{F}(\text{c})$ with the state-of-the-art on the NoCrash-dense benchmark. Our $\mathcal{L}_\text{A}(\text{AP})$ 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 $\mathcal{L}_\text{K} + \mathcal{L}_\text{F}(\text{c})$ 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 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 $\mathcal{L}_\text{A}(\text{AP})$ is due to reflections from after-rain puddles. This problem is largely alleviated by imitating Roach, which drives more naturally, and $\mathcal{L}_\text{A}$ shows

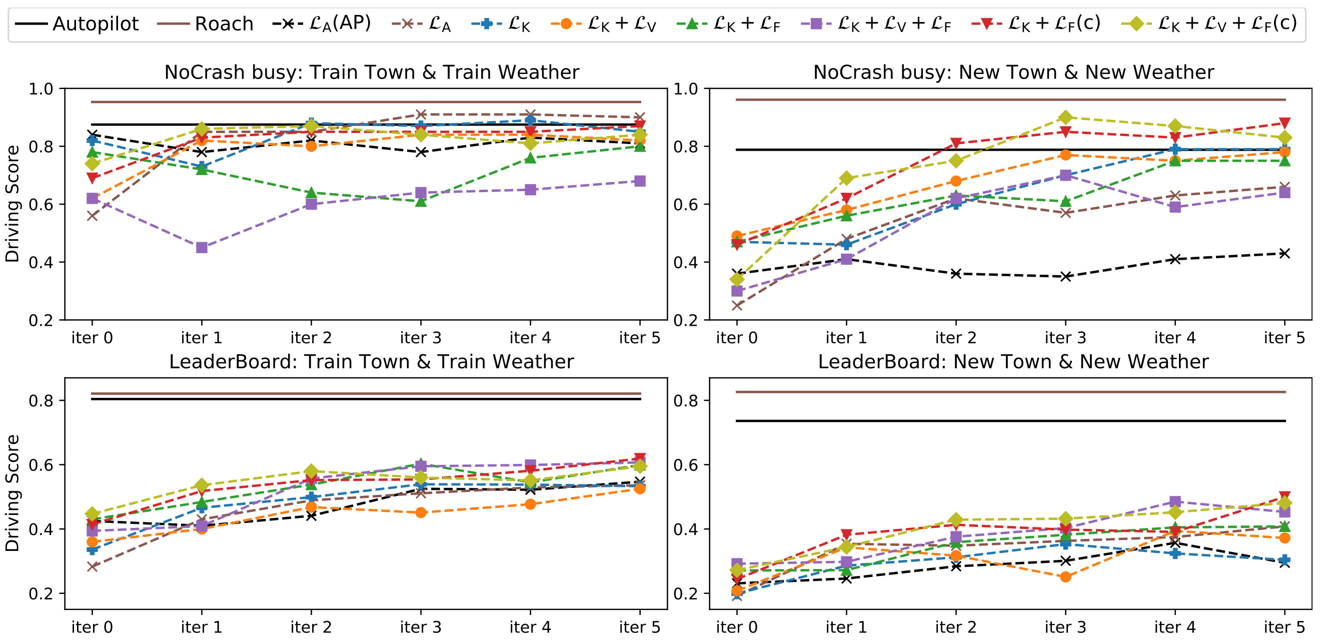


Figure 5: **Driving score of experts and IL agents.** All IL agents (dashed lines) are supervised by Roach except for $\mathcal{L}_A$(AP), which is supervised by our Autopilot. For IL agents at the 5th iteration on NoCrash and all experts, results are reported as the mean over 3 evaluation seeds. Others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

	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	$\mathbf{0} \pm 0$	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	$\mathbf{0} \pm 0$	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	$\mathbf{0} \pm 0$	0.53 ± 0.18	0.68 ± 0.08	3.39 ± 0.20
$\mathcal{L}_K + \mathcal{L}_F$(c)	$\mathbf{87} \pm 5$	$\mathbf{88} \pm 3$	$\mathbf{96} \pm 0$	$\mathbf{91} \pm 3$	$\mathbf{0.08} \pm 0.04$	0.01 ± 0.02	$\mathbf{0.23} \pm 0.08$	$\mathbf{0.61} \pm 0.23$	$\mathbf{0.84} \pm 0.04$
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.

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, 17, 32]. 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 [44].

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] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, 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

[7] 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

[8] 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

[9] 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

[10] 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

[11] 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

[12] 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

[13] 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

[14] 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

[15] Hado 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

[16] 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

[17] 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

[18] Matteo Hessel, Joseph Modayil, Hado 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

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 5

[20] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, 2016. 2

[21] 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

[22] 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

[23] Gregory Kahn, Pieter Abbeel, and Sergey Levine. LaND: Learning to navigate from disengagements. *IEEE Robotics and Automation Letters (R-AL)*, 2021. 1

[24] 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

[25] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[26] 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

[27] 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

[28] 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

[29] 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

[30] 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

[31] 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

[32] 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

[33] Praveen Palanisamy. Multi-agent connected autonomous driving using deep reinforcement learning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2020. 3

[34] 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

[35] 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

[36] 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

[37] 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

[38] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3. https://github.com/DLR-RM/stable-baselines3, 2019. 4

[39] 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

[40] 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

[41] 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

[42] 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

[43] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 4

[44] 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

[45] CARLA team. Carla autonomous driving leaderboard. https://leaderboard.carla.org/, 2020. Accessed: 2021-02-11. 5

[46] 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

[47] 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

[48] 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

[49] 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

[50] 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

[51] 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

[52] 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

[53] 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