Reinforcement Learning based Control of Imitative Policies for Near-Accident Driving

Zhangjie Cao*[1], Erdem Bıyık*[2], Woodrow Z. Wang[1],
Allan Raventos[3], Adrien Gaidon[3], Guy Rosman[3], Dorsa Sadigh[1,2]
[1]Computer Science, Stanford University, [2]Electrical Engineering, Stanford University, [3]Toyota Research Institute
Emails: {caozj18, ebiyik, wwang153, dorsa}@stanford.edu, {allan.raventos, adrien.gaidon, guy.rosman}@tri.global
* First two authors contributed equally to this work.

***Abstract*—Autonomous driving has achieved significant progress in recent years, but autonomous cars are still unable to tackle high-risk situations where a potential accident is likely. In such near-accident scenarios, even a minor change in the vehicle's actions may result in drastically different consequences. To avoid unsafe actions in near-accident scenarios, we need to fully explore the environment. However, reinforcement learning (RL) and imitation learning (IL), two widely-used policy learning methods, cannot model rapid phase transitions and are not scalable to fully cover all the states. To address driving in near-accident scenarios, we propose a hierarchical reinforcement and imitation learning (H-REIL) approach that consists of low-level policies learned by IL for discrete driving modes, and a high-level policy learned by RL that switches between different driving modes. Our approach exploits the advantages of both IL and RL by integrating them into a unified learning framework. Experimental results and user studies suggest our approach can achieve higher efficiency and safety compared to other methods. Analyses of the policies demonstrate our high-level policy appropriately switches between different low-level policies in near-accident driving situations.**

I. INTRODUCTION

Recent advances in learning models of human driving behavior have played a pivotal role in the development of autonomous vehicles. Although several milestones have been achieved (see [1]–[12] and references therein), the current autonomous vehicles still cannot make safe and efficient decisions when placed in a scenario where there can be a high risk of an accident (a near-accident scenario). For example, an autonomous vehicle needs to be able to coordinate with other cars on narrow roads, make unprotected left turns in busy intersections, yield to other cars in roundabouts, and merge into a highway in a short amount of time. The left panel of Fig. 1 shows a typical near-accident scenario: The ego car (red) wants to make an unprotected left turn, but the red truck occludes the oncoming blue car, making the ego car fail to notice the blue car, which can potentially result in a collision. Clearly, making suboptimal decisions in such near-accident scenarios can be dangerous and costly, and is a limiting factor on the road to safe wide-scale deployment of autonomous vehicles.

One major challenge when planning for autonomous vehicles in near-accident scenarios is the presence of *phase transitions* in the car's policy. Phase transitions in autonomous driving occur when small changes in the critical states – the ones we see in near-accident scenarios – require dramatically different actions of the autonomous car to stay safe. For example, the speed of the blue car in Fig. 1 can determine the ego car's policy: if it slows down, the ego car can proceed forward and make the left turn; however, a small increase in its speed would require the ego car to stop and yield. The rapid phase transition requires a policy that can handle such non-smooth transitions. Due to the non-smooth value function, an action taken in one state may not generalize to nearby states. Hence, when training a policy, our algorithms must be able to visit and handle all the critical states individually, which can be computationally inefficient.

Reinforcement learning (RL) [12]–[14] and imitation learning (IL) [1]–[3], [15]–[24] are two promising learning-based approaches for autonomous driving. RL explores the state-action space to find a policy that maximizes the reward signals while IL imitates the behavior of the agent from expert demonstrations. However, the presence of rapid phase transitions makes it hard for RL and IL to capture the policy because they learn a smooth policy across states. Furthermore, to achieve full coverage, RL needs to explore the full environment while IL requires a large amount of expert demonstrations covering all states. Both are prohibitive since the state-action space in driving is continuous and extremely large.

In this paper, *our key insight is to model phase transitions as optimal switches, learned by reinforcement learning, between different modes of driving styles, each learned through imitation learning.* In real world driving, various factors influence the behaviors of human drivers, such as efficiency (time to destination), safety (collision avoidance), etc. Different modes characterize different trade-offs of all factors. For example, the aggressive mode cares more about efficiency so it always drives fast in order to reach the destination in minimal time. The timid mode cares more about safety, so it usually drives at a mild speed and pays attention to all potential threats. Switching from one mode to another can model the rapid phase transition conditioned on the environment changes.

Using these modes, we propose a new algorithm **Hierarchical Reinforcement and Imitation Learning** (H-REIL), which is composed of a high-level policy learned with RL that switches between different modes and low-level policies learned with IL, each of which represents a different mode.

Using our proposed approach, the low-level policy for each mode can be efficiently learned with IL even with only a few expert demonstrations, since IL is now learning a much simpler and specific policy by sticking to one driving style with little phase transition. We emphasize that RL would not

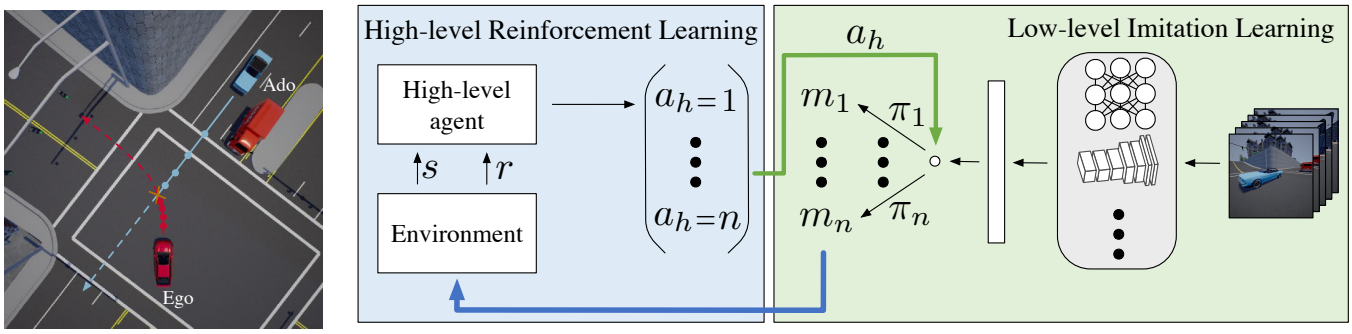

Fig. 1. The left part of the figure is a typical near-accident scenario: The ego car (red car) turns left but the truck occludes the blue car, which causes the ego car to overlook the blue car and collide with it at time step 5. The right part of the figure is the overall architecture of the proposed hierarchical reinforcement learning and imitation learning model. The right green square shows the low-level imitation learning part, where the low-level policies are learned by the conditional imitation framework. All the policies share the same feature extractor and split to different branches in later layers for action prediction, where each corresponds to one mode. The branch is selected by external input a_h from high-level reinforcement learning. The low-level policies are learned from expert demonstrations by imitation learning. The left blue square shows the high-level reinforcement learning part, where the high-level agent interacts with the environment to learn the high-level policy, which selects the best low-level policy branch through the high-level action a_h at different states.

be a reasonable fit for learning the low-level policies as it is difficult to define the reward function. For example, designing a reward function for the aggressive mode that exactly matches an aggressive human driver's behavior is non-trivial.

For the high-level policy, RL is a better fit since we need to learn to maximize the return based on a reward that contains a trade-off between various terms, such as efficiency and safety. Furthermore, the action space is now reduced from a continuous space to a finite discrete space. IL does not fit to the high-level policy, because it is not natural for human drivers to accurately demonstrate how to switch driving modes.

We therefore combine RL at the high-level and IL at the low-level in our proposed hierarchical model, which can utilize both approaches and learn driving policies in a wide variety of settings, including near-accident driving scenarios.

Our main contributions in this paper are three-fold:

- We develop a **Hierarchical Reinforcement and Imitation Learning (H-REIL)** approach composed of a high-level policy learned with RL, which switches optimally between different modes, and low-level policies learned with IL, which represent driving in different modes.
- We demonstrate and assess our proposed H-REIL model on two different driving simulators in a set of near-accident driving scenarios. Our simulations demonstrate that the learned hierarchical policy outperforms imitation learning policies, the individual policies learned for each mode, and a policy based on random mixtures of modes, in terms of efficiency and safety.
- We finally conduct a user study in which human subjects compare trajectories generated by H-REIL and the compared methods to demonstrate H-REIL's ability to generate safe and efficient policies. The results show the users significantly prefer the H-REIL driving policies compared to other methods in near-accident scenarios.

II. Related Work

Rule-based Methods. Traditional autonomous driving techniques are mostly based on manually designed rules [25]–[27]. However, it is tedious, if not impossible, to enumerate all the driving rules and norms to deal with all the states. Therefore, rule-based methods often cause the vehicle to drive in an unnatural manner or completely fail in unexpected edge cases.

Imitation Learning (IL). ALVINN was one of the first instances of IL applied to driving [1]. Following ALVINN, Muller *et al.* [28] solved off-road obstacle avoidance using behavior cloning. IL learns driving policies on datasets consisting of off-policy state-action pairs. However, they suffer from potential generalization problems to new test domains due to the distribution shift. Ross *et al.* [29] address this shortcoming by iteratively extending the base dataset with on-policy state-action pairs, while still training the base policy offline with the updated dataset. Bansal *et al.* [17] augment expert demonstrations with perturbations and train the IL policy with an additional loss penalizing undesired behavior. Generative Adversarial Imitation Learning [30], [31] proposes to match the state-action occupancy between trajectories of the learned policy and the expert demonstrations.

A major shortcoming of IL is that it requires a tremendous amount of expert demonstrations. Conditional imitation learning (CoIL) [15] extends IL with high-level commands and learns a separate IL model for each command. Although it improves data-efficiency, high-level commands are required at test time, e.g., the direction at an intersection. In our setting, each high-level command corresponds to a different driving mode. Instead of depending on drivers to provide commands, we would like to learn the optimal mode-switching policy.

Inverse Reinforcement Learning (IRL). Originally proposed to address the learning problem in a Markov decision process (MDP) without an explicitly given reward function [32], IRL aims to recover the reward function from expert demonstrations. The reward is typically represented by a weighted sum of several reward features relevant to the task. IRL learns those weights by observing how experts perform the task. Abbeel and Ng [32] tune the weights to match the expected return of the expert trajectories and the optimal policy. Ziebart *et al.* [33] further add a maximum entropy regularization. Following [34], Finn *et al.* [35] improve the optimization in [33].

Similar to IL, IRL also suffers from the requirement of a

large amount of expert demonstrations. It is also difficult and tedious to define reward features that accurately characterize efficiency and safety in all scenarios. Thus, IRL is not fit for learning driving policies in near-accident scenarios.

Reinforcement Learning (RL). RL has been applied to learn autonomous driving policies [14], [36]–[38]. RL explores the environment to seek the action that maximizes the expected return for each state based on a pre-defined reward function. However, it suffers from the fact that the state-action space for driving is extremely large, which makes it very inefficient to explore. Chen *et al.* [36] try to alleviate this problem by augmenting RL with Model Predictive Control (MPC) to optimally control a system while satisfying a set of constraints. Tram *et al.* [39] combine RL with MPC to shrink the action space, however the MPC is based on driving rules, which are difficult to exhaustively define and enumerate. Finally, Gupta *et al.* [40] proposed using RL to fine-tune IL policies for long-horizon, multi-stage tasks, different than our problem setting.

Hierarchical Reinforcement Learning. Hierarchical RL is motivated by feudal reinforcement learning [41], which first proposes a hierarchical structure for RL composing of multiple layers: the higher layer acts as a manager to set a goal for the lower layer, which acts as a worker to satisfy the goal. Hierarchical RL enables efficient exploration for the higher level with a reduced action space, i.e. goal space, while making RL in the lower level easier with an explicit and short-horizon goal. Recent works extended hierarchical RL to solve complex tasks [42]–[46]. Le *et al.* [47] proposed a variant of hierarchical RL, which employs IL to learn the high-level policy to leverage expert feedback to explore the goal space more efficiently. Recently, more related to our work, Qureshi *et al.* [48] proposed using deep RL to obtain a mixture between task-agnostic policies. However in our case, low-level policies are not task-agnostic and are produced by IL on the same tasks, so it is arguably sufficient to discretely switch between them. Finally, Nair *et al.* [49] use expert demonstrations to guide the exploration of RL.

However, for near-accident scenarios, most off-the-shelf hierarchical RL techniques do not address the problem of driving safely and efficiently, because it is difficult to define the reward function for low-level RL. We instead construct a hierarchy of RL and IL, where IL is in the low-level to learn a basic policy for each mode and RL is in the high-level, similar to [50], to learn a mode switching policy that maximizes the return based on a simpler pre-defined reward function.

III. Model

A. Problem Setting

We model traffic as a partially observable Markov decision process (POMDP): $P_l = \langle \mathcal{S}, \Omega, O, \mathcal{A}, f, R \rangle$ where the agent is the ego car. The scenario terminates either by a collision, by reaching the destination, or by a time-out, which forces the POMDP to be finite horizon. $\mathcal{S}$ is the set of states, Ω is the set of observations, O is the set of conditional observation probabilities, $\mathcal{A}$ is the set of actions, and f is the transition function. Each state $s^t \in \mathcal{S}$ consists of the positions and velocities of all the vehicles at time step t. Each action $a^t \in \mathcal{A}$ is the throttle and the steering control available for the ego car. At each time step t, all vehicles move and the state s^t is transitioned to a new state s^{t+1} according to f, which we model as a probability distribution, $P(s^{t+1}|s^t, a^t) = f(s^t, a^t, s^{t+1})$, where the stochasticity comes from noise and the uncertainty about the other vehicles' actions. The agent receives an observation $o^t \in \Omega$ with a likelihood conditioned on the state s^t, *i.e.* $O(o^t|s^t)$. For example, if some vehicles are occluded behind a building, their information is missing in the observation. Finally, the agent receives a reward $R(s^t, a^t)$ at each time step t, which encodes desirable driving behavior.

B. H-ReIL *Framework*

We design the H-ReIL framework using a set of n experts, each representing its own mode of driving. Following different modes, the experts are not necessarily optimal with respect to the true reward function. For example, the modes can be aggressive or timid driving. We denote the corresponding policies by $\pi_1, \ldots, \pi_n$; where $\pi_i : \Omega \rightarrow \mathcal{A}$, $\forall i$. Our goal is to learn a policy Π that switches between the modes to outperform all π_i in terms of cumulative reward.

As shown in the right panel of Fig. 1, we divide the problem into two levels where $\pi_i \,|_{i=1}^n$ are low-level policies learned with IL using the data coming from experts, and the high-level agent learns Π with RL using a simulator of the POMDP.

Low-Level Imitation Learning Policy. Unlike [42] and [47], which employ RL in the low-level of the hierarchy, we employ IL to learn low-level policies π_i, because each low-level policy sticks to one driving style, which behaves relatively consistently across states and requires little rapid phase transitions. Hence, the actions taken in nearby states can generalize to each other easily. Therefore, the simpler policy can be learned by IL easily with only a few expert demonstrations $\mathcal{H}_i = \{o_i^t, a_i^t\}|_{t=1}^K$, consisting of observation-action pairs for each mode m_i. Here we use Conditional Imitation Learning (CoIL) [15] as our IL method. We define the loss as

$$l_{IL} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{K} \sum_{t=1}^{K} \ell_1(a_i^t, \pi_i(o_i^t)), \tag{1}$$

where we take the mean over $L1$ distances. As in CoIL, we model $\pi_i \,|_{i=1}^n$ using a neural network with branching at the end. Each branch corresponds to an individual policy π_i. We present the details of the networks in Section IV-F.

High-Level Reinforcement Learning Policy. After training the low-level policies, we build the high-level part of the hierarchy: We train a high-level policy Π to select which of the policies from $S_\pi = \{\pi_i\}_{i=1}^n$ the ego car should follow. This high-level decision is made every t_s time steps of P_l.

We model this high-level problem as a new POMDP, called $P_h^{t_s}$, where the states and observations are the same as the original POMDP P_l, but the actions choose which driving mode to follow. For example, if the action is 2, then the ego car follows π_2 for the next t_s time steps in P_l, which is a single time step in $P_h^{t_s}$. Formally, $P_h^{t_s} = \langle \mathcal{S}, \Omega, \mathcal{O}, \mathcal{A}_h, f_h^{t_s}, R_h^{t_s} \rangle$ and the new action space $\mathcal{A}_h$ is a discrete space, $\{1, 2, ..., n\}$, representing

the selection of low-level policies. The new transition function $f_h^{t_s}(s^t, a_h, s^{t+1})$ gives the probability of reaching s^{t+1} from s^t by applying policy π_{a_h} for t_s consecutive time steps in P_l. Similarly, the new reward function accumulates the reward from P_l over t_s time steps in which the policy π_{a_h} is followed.

Then, our goal in this high-level hierarchy is to solve:

$$\arg\max_{\Pi} \mathbb{E}\left[\sum_j \sum_{o^j} O(o^j \mid s^j) R_h^{t_s}(s^j, \Pi(o^j))\right]$$
$$\text{subject to } s^{j+1} \sim f_h^{t_s}(s^j, \Pi(o^j), s^{j+1}) \text{ for } \forall j \tag{2}$$

where we use indexing j to denote the time steps of $P_h^{t_s}$. As shown in Fig. 1, we attempt to solve (2) using RL. In $P_h^{t_s}$, the action space is reduced from continuous to discrete, which eases the efficient exploration of the environment. Furthermore, it is now much easier to define a reward function because the ego car already satisfies some properties by following the policies learned from expert demonstrations. For example, with a high enough t_s, we do not need to worry about jerk, because the experts naturally give low-jerk demonstrations. Therefore, we design a simple reward function consisting of the efficiency and safety terms (R_e and R_s). R_e is negative in every time step, so that the agent will try to reach its destination as quickly as possible. R_s gets an extremely large negative value if a collision occurs. Otherwise, it is 0.

Besides, setting $t_s > 1$ reduces the number of time steps in an episode and makes the collision penalty, which appears at most once per episode, less sparse. With the new action space, transitions, and reward function, we can train the high-level policy with any RL algorithm (PPO [51] in this paper). Algorithm 1 outlines our training algorithm.

Algorithm 1: H-ReIL Training Algorithm

Input: Expert demonstrations $\mathcal{H}_1, ..., \mathcal{H}_n$, POMDP $P_h^{t_s} = \langle \mathcal{S}, \Omega, \mathcal{O}, \mathcal{A}_h, f_h^{t_s}, R_h^{t_s} \rangle$
Output: Low-level policies $\pi_i \mid_{i=1}^N$, high-level policy Π
Train low-level policies $\pi_i \mid_{i=1}^N$ with demonstrations $\mathcal{H}_i \mid_{i=1}^N$ to minimize the loss in Eqn. (1).
Train high-level policy Π using $\pi_i \mid_{i=1}^N$ and $P_h^{t_s}$ according to (2) with PPO.
return $\pi_i \mid_{i=1}^N$ and Π

C. Analysis of H-ReIL

Proposition 1. *Let's consider a POMDP with a fixed finite horizon T, for which we have n low-level policies. Let's call the expected cumulative reward for the optimal and worst high-level control sequences U^* and U', respectively. If there exists a scalar $a > U^* - U'$ such that the expected cumulative rewards of keeping the same low-level policy are smaller than $U^* - a + a/n^T$; then there exists a probability distribution p such that randomly switching the policies with respect to p is better than keeping any of the n low-level policies.*

Proof: Let p be a uniform distribution among the low-level policies. Then, each possible control sequence has a $1/n^T$ probability of being realized. This guarantees that the expected cumulative reward of this random policy is larger than: $\frac{n^T-1}{n^T}(U^* - a) + \frac{1}{n^T}(U^*) = U^* - a + a/n^T$.

While this is a worst-case bound, it can be shown that the expected cumulative reward of a random policy can be higher if the optimal high-level control sequence is known to be imbalanced between the modes. In that case, a better lower bound for random switching is obtained by a p maximizing the probability of the optimal sequence being realized. ■

For a different interpretation of H-ReIL, one can think of the true driving reward R as a sum of n different terms; such as, for $n = 2$, $R(s^t, a^t) = R_e(s^t, a^t) + R_s(s^t, a^t)$ where R_e denotes the part of the reward that is more associated with efficiency, and R_s with safety. Then, strictly aggressive drivers optimize for $\alpha R_e(s^t, a^t) + (2-\alpha) R_s(s^t, a^t)$ for some $1 < \alpha \leq 2$, whereas strictly timid drivers try to optimize the same reward with $0 \leq \alpha < 1$. One may then be tempted to think there exists a high-level stationary random switching distribution p that outperforms both the aggressive and timid drivers, because the true reward function is in the convex hull of the individuals' reward functions for each $s^t \in \mathcal{S}, a^t \in \mathcal{A}$. However, even with this reward structure and hierarchy, the existence of such a p is not guaranteed without the assumptions of Proposition 1 (or other assumptions).

Remark 1. *With the reward structure that can be factorized such that each mode weighs some terms more than the others and the true reward is always in the convex hull of them, there may not exist a high-level stationary random-switching strategy that outperforms keeping a single low-level policy.*

Proof: Consider the 4-state deterministic MDP with a finite-horizon T shown in Fig. 2. There are only two actions, represented by solid ($a = 1$) and dashed ($a = 2$) lines. The rewards for each state-action pair are given in a tuple form $r = (R_e, R_s)$ where the true reward is $R(s^t, a^t) = R_e(s^t, a^t) + R_s(s^t, a^t)$. Consider two modes optimizing $\alpha R_e(s^t, a^t) + (2-\alpha) R_s(s^t, a^t)$, one for $\alpha = 1.8$ and the other for $\alpha = 0.2$. While the former will always take $a = 1$, the latter will keep $a = 2$. Both policies will achieve a true cumulative reward of 0. Let $t_s = 1$. A stationary random switching policy cannot outperform those individual policies, because they will introduce a risk of getting $R = -2$ from s_2 and s_4. In fact, any such policy that assigns strictly positive probabilities to each action will perform worse than the individual policies. On the other hand, a policy that outperforms the individual policies by optimally switching between the modes exists and achieves T cumulative reward. ■

Unfortunately, the assumptions of Proposition 1 may not hold for driving in general, and Remark 1 shows that a stationary random switching strategy may perform poorly. Next, we show that the solution to (2) yields a good policy.

Proposition 2. *The optimal solution to* (2) *is at least as good as keeping the same low-level policy throughout the episode in terms of the expected cumulative reward.*

Proof: Since $\Pi(o^j) = i$ for $\forall o^j \in \Omega$ for any i is a feasible solution to (2), the optimal solution is guaranteed to be at least

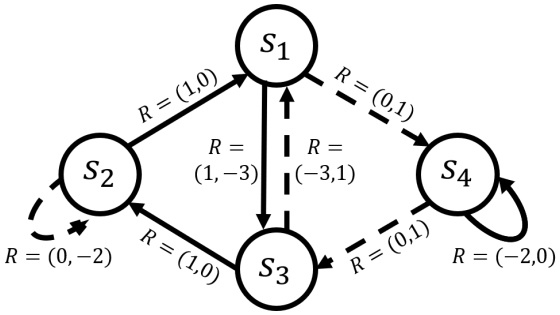

Fig. 2. While random switching cannot guarantee better performance, an intelligent switching policy outperforms individual low-level policies.

as good as keeping the same low-level policy in terms of the objective, i.e. the expected cumulative reward. ■

In H-REIL, we decompose the complicated task of driving in near-accident scenarios into two levels, where the low-level learns basic policies with IL to realize relatively easier goals, and the high-level learns a meta-policy using RL to switch between different low-level policies to maximize the cumulative reward. **The mode switching can model rapid phase transitions. With the reduced action space and fewer time steps, the high-level RL can explore all the states efficiently to address state coverage.** The two-level architecture makes both IL and RL much easier, and learns a policy to drive efficiently and safely in near-accident scenarios.

IV. EXPERIMENTS

A video giving an overview of our experiments, as well as the proposed framework, is at https://youtu.be/CY24zlC_HdI. Below, we describe our experiment settings.

A. Environment

We consider the environment where the ego car navigates in the presence of an ado car. The framework extends to cases with multiple environment cars easily. In order to model near-accident scenarios, we let the ado car employ a policy to increase the possibility of collision with the ego car.

B. Scenarios

We design five near-accident scenarios, each of which is visualized in Fig. 3 and described subsequently.

1) Cross Traffic. The ego car seeks to cross the intersection, but a building occludes the ado car (Fig. 3, row 1).

2) Halting Car. The ego car drives behind the ado car, which occasionally stops abruptly (Fig. 3, row 2).

3) Wrong Direction. The ado car, which drives in the opposite direction, cuts into the ego car's lane (Fig. 3, row 3).

4) Unprotected Turn. The ego car seeks to make a left turn, but a truck occludes the oncoming ado car (Fig. 3, row 4).

5) Merge. The ego car wants to cut between the ado car and another car in the front, who follows a fixed driving policy. However, the ado car can aggressively accelerate to prevent it from merging (Fig. 3, row 5).

For each scenario, we have two settings: difficult and easy. The difficult setting is described above where the ado car acts carelessly or aggressively, and is likely to collide with the ego car. The easy setting either completely removes the ado car from the environment or makes it impossible to collide with the ego car. In simulation, we sample between these two settings uniformly at random for each scenario. In addition, we also perturb the initial positions of both cars with some uniform random noise in their nominal directions.

C. Simulators

CARLO[1] is our in-home 2D driving simulator that models the physics and handles the visualizations in a simplistic way (see Fig. 5). Assuming point-mass dynamics model as in [52], CARLO simulates vehicles, buildings and pedestrians.

While CARLO does not provide realistic visualizations other than two-dimensional diagrams, it is useful for developing control models and collecting large amounts of data. Therefore, we use CARLO as a simpler environment where we assume perception is handled, and so we can directly use the noisy measurements of other vehicles' speeds and positions (if not occluded) in addition to the state of the ego vehicle.

CARLA [53] is an open-source simulator for autonomous driving research, which provides realistic urban environments for training and validation of autonomous driving systems. Specifically, CARLA enables users to create various digital assets (pedestrians, buildings, vehicles) and specifies sensor suites and environmental conditions flexibly. We use CARLA as a more realistic simulator than CARLO.

For both CARLO and CARLA, the control inputs for the vehicles are throttle/brake and steering.

D. Modes

While H-REIL can be used with any finite number of modes, we consider two in this paper ($n = 2$): aggressive and timid modes. In the former, the ego car favors efficiency over safety: It drives fast and frequently collides with the ado car. In the timid mode, the ego car drives in a safe way to avoid all potential accidents: It slows down whenever there is even a slight risk of an accident. The high-level agent learns to switch between the two modes to achieve our final goal: *driving safely and efficiently in near-accident scenarios*.

For the near-accident driving setting, having two modes of driving – aggressive and timid – is arguably the most natural and realistic choice. Since humans often do not optimize for other nuanced metrics, such as comfort, in a near-accident scenario and the planning horizon of our high-level controller is extremely short, there is a limited amount of diversity that different modes of driving would provide, which makes having extra modes unrealistic and unnecessary in our setting.

For our simulations on the first four scenarios (other than Merge), we collect data from the hand-coded aggressive and timid modes for the ego car based on rules around the positions and velocities of the vehicles involved. While both modes try to avoid accidents and reach destinations; their reaction times, acceleration rates and willingness to take risks differ.

For the Merge scenario, we collected real driving data from a driver who tried to drive either aggressively or timidly. We collected human data only in CARLA due to its more realistic visualizations and dynamics model.

[1]Publicly available at https://github.com/Stanford-ILIAD/CARLO.

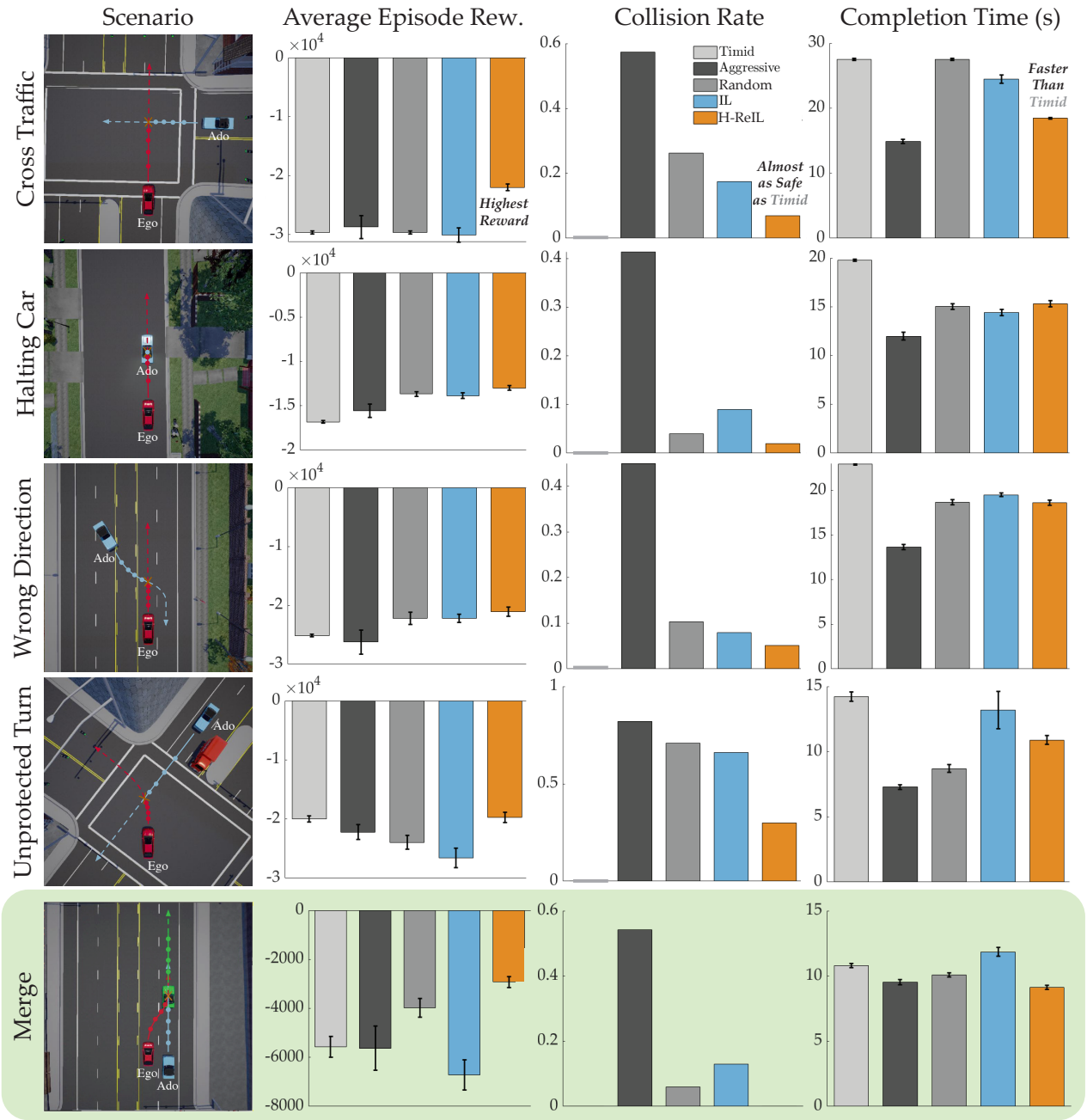

Fig. 3. The scenario illustration, average episode reward, collision rate, completion time for each scenario and each policy in CARLA simulator. In the scenario visualizations, the ego car is always red and the ado car is blue.

In each of the first four scenarios, we separately collect aggressive and timid driving data as expert demonstrations for the aggressive and timid modes, denoted by $\mathcal{H}_{\text{agg}}$ and $\mathcal{H}_{\text{tim}}$, respectively. In CARLO, which enables fast data collection, we collected 80000 episodes per mode. In CARLA, which includes perception data, we collected 100 episodes per mode.

E. Compared Methods

We compare H-REIL with the following policies:

1) IL. π^{IL} trained on the mixture of aggressive and timid demonstrations $\mathcal{H}_{\text{agg}}$ and $\mathcal{H}_{\text{tim}}$.
2) AGGRESSIVE. π^{agg} trained only on $\mathcal{H}_{\text{agg}}$ with IL.
3) TIMID. π^{tim} trained only on $\mathcal{H}_{\text{tim}}$ with IL.
4) RANDOM. Π^{rand} which selects π^{agg} or π^{tim} at every high-level time step uniformly at random.

F. Implementation Details

CARLO. The observations include ego car location and velocity. They also include the location and the velocity of the ado car, if not occluded, perturbed with Gaussian noise.

These are then fed into a neural network policy with two fully-connected hidden layers to output the high-level decision. The same information are also fed into a neural network with only a single fully-connected hidden layer to obtain features. Depending on the high-level mode selection, these features are then passed into another fully connected neural network with a single hidden layer, which outputs the controls.

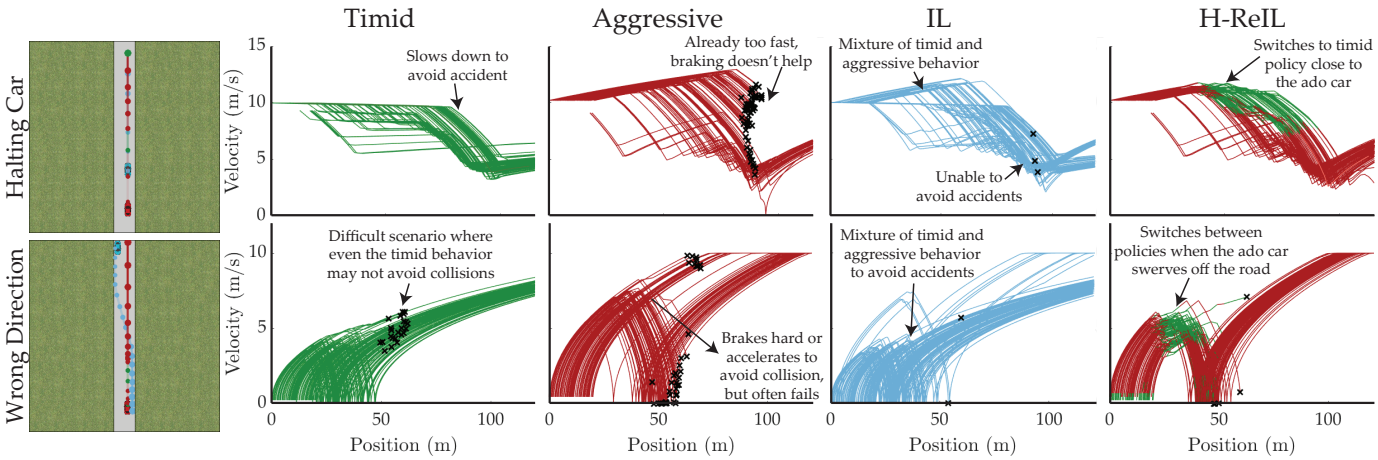

Fig. 4. The plots of velocity vs position of the ego car under Halting Car and Wrong Direction scenarios with TIMID, AGGRESSIVE, IL and H-REIL in CARLO. The green and red colors correspond to the selections of timid and aggressive modes, respectively. The black crosses show collisions where the episode terminates. The episode also terminates when the ego car arrives at the predefined destinations.

CARLA. The observations consist of ego car location, velocity and a front-view image for the first four scenarios. Merge scenario has additional right-front and right-view images to gain necessary information specific to the scenario.

For the first four scenarios, we use an object detection model, Faster-RCNN with R-50-FPN backbone [54], to detect the cars in the front-view images and generate a black image with only the bounding boxes colored white, which we call the detection image. It provides information of the ado car more clearly and alleviates the environmental noise. We do not apply this technique to the Merge scenario because the ado car usually drives in parallel with the ego car and its shape is only partially observable in some views. Instead, we use the original RGB images for the Merge scenario.

We then design another network consisting of a convolutional neural network encoder and a fully-connected network encoder. The convolutional encoder encodes the detection image and the fully-connected encoder encodes the location and velocity information (of the ego car) into features.

The high-level RL policy feeds these features into a fully-connected network to output which mode the ego car will follow. We then feed the features to the chosen low-level IL policy composed of fully-connected layers, at the next t_s low-level time steps to obtain the controls. We use Proximal Policy Optimization (PPO) [51] for the high-level agent of H-REIL.

For IL, we use a network structure similar to our approach but without branching since there is no mode selection.

V. RESULTS

A. Simulations

We compare the *average episode reward*, *collision rate*, and *completion time* of different methods under all scenarios with both simulators. We compute these metrics for each model and scenario averaged over 100 test runs.

For the simple reward of the high-level agent, we select the trade-off between efficiency (time/distance penalty) and safety (collision penalty) such that the high-level policy cannot naïvely bias to a single low-level policy. The collision rate is only computed for the episodes with the difficult setting.

As shown in Fig. 3 for CARLA, our H-REIL framework is better than or comparable to other methods in terms of the average episode reward under all five scenarios, which demonstrates the high-level RL agent can effectively learn a smart switching between low-level policies. H-REIL framework usually outperforms IL with a large margin, supporting the claim that in near-accident scenarios, training a generalizable IL policy requires a lot of demonstrations. *Inadequate demonstrations cause the IL policy to fail in several scenarios*.

In terms of collision rate and completion time, H-REIL achieves a collision rate lower than IL, AGGRESSIVE and RANDOM while comparable to TIMID. H-REIL also achieves a completion time shorter than IL and TIMID while comparable to RANDOM. These demonstrate H-REIL achieves a good trade-off between efficiency and safety.

B. User Studies

Having collected real driving data in CARLA for the Merge scenario, we generated a test set that consists of 18 trajectories for each of AGGRESSIVE, TIMID, IL and H-REIL. We then recruited 49 subjects through Amazon Mechanical Turk to evaluate how good the driving is on a 7-point Likert scale (1 - least preferred, 7 - most preferred). Figure 6 shows the users prefer H-REIL over the other methods. The differences between H-REIL and the other methods are statistically significant with $p < 0.005$ (two-sample t-test).

VI. ANALYSIS

Velocity Analysis. We visualize the relation between the velocity and the position of the ego car in its nominal direction in Fig. 4 for the Halting Car and the Wrong Direction scenarios in CARLO. We selected these two scenarios for visualization as the ego does not change direction.

We observe TIMID always drives with a relatively low speed while AGGRESSIVE drives fast but collides with the ado car more often. Compared with these two, H-REIL and IL drive at a medium speed while H-REIL achieves a relatively higher speed than IL with comparable number of accidents.

In particular, there is an obvious phase transition in both scenarios (about $[35, 75]$ for the Halting Car and $[25, 45]$ for

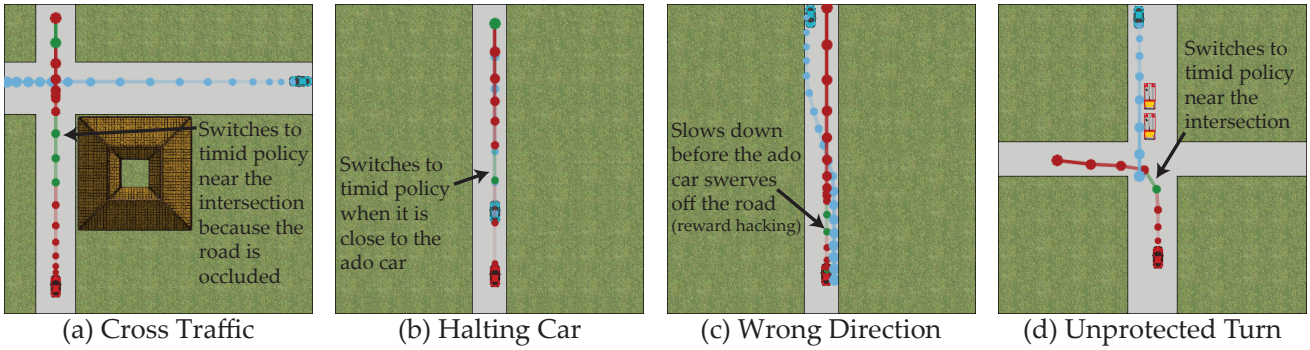

Fig. 5. Visualization of locations at each time step of the ego car and the ado car in CARLO simulator. The blue color shows trajectory of the ado car. Green means selecting the timid policy while red means selecting the aggressive policy.

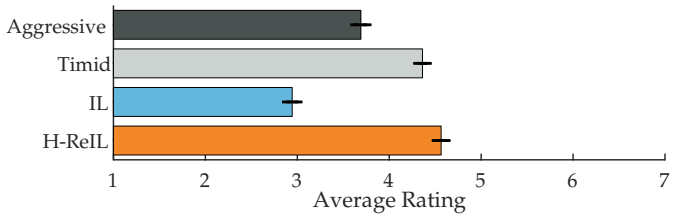

Fig. 6. User study results are shown. Users rate H-REIL significantly higher than the other methods ($p < 0.005$).

the Wrong Direction) where a collision is very likely to occur. Baseline models learned by plain IL, cannot model such phase transitions well. Instead, H-REIL switches the modes to model such phase transitions: it selects the timid mode in the risky states to ensure safety while selecting the aggressive policy in other states to maximize efficiency. This intelligent mode switching enables H-REIL to drive reasonably under different situations: slowly and cautiously under uncertainty, and fast when there is no potential risk.

Policy Visualization. We visualize the locations of the cars in Fig. 5 in CARLO. We observe that H-REIL usually chooses the timid policy at the areas that have a collision risk while staying aggressive at other locations when it is safe to do so. These support that our high-level policy makes correct decisions under different situations.

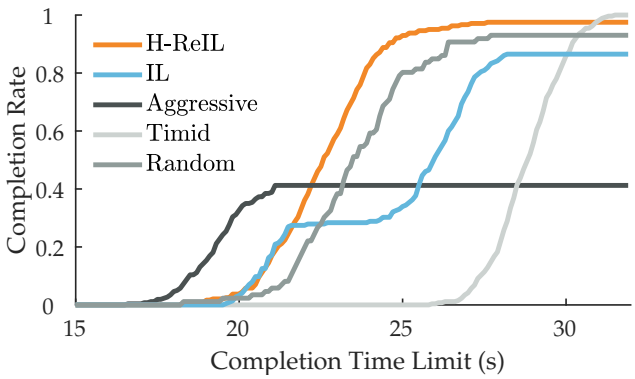

Fig. 7. The completion rate with varying time limits. The completion rate is the proportion of the trajectories in which the ego car safely reaches the destination within the time limit.

Completion within Time Limit. We plot the completion rate with respect to varying time limits for the ego car in Fig. 7 in CARLA for the Cross Traffic scenario. The completion rate is the portion within 500 runs that the ego car reaches the destination within the time limit. Overall, we observe H-REIL achieves the best trade-off. While AGGRESSIVE achieves higher completion rates for the low time limits, it cannot improve further with the increasing limit with collisions.

We also observe the trajectories of IL are divided into two clusters. The group that achieves lower time limit (20-22s) imitates the aggressive policy more but has lower completion rate. The other group that corresponds to the higher time limit (25-28s) imitates the timid policy more but has better completion rate. This demonstrates IL directly imitates the two modes and learns a mild aggressive or a mild timid policy while it does not learn when to use each mode. On the other hand, H-REIL consistently achieves higher or comparable completion rate than IL and RANDOM, showing that our high-level RL agent can learn when to switch between the modes to safely arrive at the destination efficiently.

VII. CONCLUSION

Summary. In this work, we proposed a novel hierarchy with reinforcement learning and imitation learning to achieve safe and efficient driving in near-accident scenarios. By learning low-level policies using IL from drivers with different characteristics, such as different aggressiveness levels, and training a high-level RL policy that makes the decision of which low-level policy to use, our method H-REIL achieves a good trade-off between safety and efficiency. Simulations and user studies show it is preferred over the compared methods.

Limitations and Future Work. Although H-REIL is generalizable to any finite number of modes, we only considered $n=2$. Having more than 2 modes, for which our preliminary experiments have given positive results, can be useful for other robotic tasks. Also, we hand-designed the near-accident scenarios in this work. Generating them automatically as in [55] could enable broader evaluation in realistic scenarios.

ACKNOWLEDGMENTS

The authors thank Derek Phillips for the help with CARLA simulator, Wentao Zhong and Jiaqiao Zhang for additional experiments with H-REIL, and acknowledge funding by FLI grant RFP2-000. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

[1] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989, pp. 305–313.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[3] A. Amini, G. Rosman, S. Karaman, and D. Rus, "Variational end-to-end navigation and localization," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8958–8964.

[4] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Proceedings of Robotics: Science and Systems (RSS)*, 2016. DOI: 10.15607/RSS.2016.XII.029.

[5] D. Sadigh, S. S. Sastry, S. A. Seshia, and A. Dragan, "Information gathering actions over human internal state," in *Proceedings of the IEEE, /RSJ, International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2016, pp. 66–73. DOI: 10.1109/IROS.2016.7759036.

[6] D. Sadigh, N. Landolfi, S. S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for cars that coordinate with people: Leveraging effects on human actions for planning and active information gathering over human internal state," *Autonomous Robots (AURO)*, vol. 42, no. 7, pp. 1405–1426, 2018, ISSN: 1573-7527. DOI: 10.1007/s10514-018-9746-1.

[7] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 336–345.

[8] E. Biyik and D. Sadigh, "Batch active preference-based learning of reward functions," in *Proceedings of the 2nd Conference on Robot Learning (CoRL)*, ser. Proceedings of Machine Learning Research, vol. 87, PMLR, 2018, pp. 519–528.

[9] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 9329–9338.

[10] M. Kwon, E. Biyik, A. Talati, K. Bhasin, D. P. Losey, and D. Sadigh, "When humans aren't optimal: Robots that collaborate with risk-aware humans," in *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2020. DOI: 10.1145/3319502.3374832.

[11] C. Basu, E. Biyik, Z. He, M. Singhal, and D. Sadigh, "Active learning of reward dynamics from hierarchical queries," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019. DOI: 10.1109/IROS40897.2019.8968522.

[12] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[13] K. Makantasis, M. Kontorinaki, and I. Nikolos, "A deep reinforcement learning driving policy for autonomous road vehicles," *arXiv preprint arXiv:1905.09046*, 2019.

[14] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "Deep reinforcement learning framework for autonomous driving," *Electronic Imaging*, vol. 2017, no. 19, pp. 70–76, 2017.

[15] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1–9.

[16] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end simulated driving," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[17] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018.

[18] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2722–2730.

[19] M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining how a deep neural network trained with end-to-end learning steers a car," *arXiv preprint arXiv:1704.07911*, 2017.

[20] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2017, pp. 204–211.

[21] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," *arXiv preprint arXiv:1704.03952*, 2017.

[22] X. Huang, S. G. McGill, B. C. Williams, L. Fletcher, and G. Rosman, "Uncertainty-aware driver trajectory prediction at urban intersections," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 9718–9724.

[23] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," *arXiv preprint arXiv:1804.09364*, 2018.

[24] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on intelligent vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[25] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, 2018.

[26] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[27] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[28] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in neural information processing systems*, 2006, pp. 739–746.

[29] S. Ross, G. J. Gordon, and J. A. Bagnell, "No-regret reductions for imitation learning and structured prediction," in *In AISTATS*, Citeseer, 2011.

[30] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in neural information processing systems*, 2016, pp. 4565–4573.

[31] J. Song, H. Ren, D. Sadigh, and S. Ermon, "Multi-agent generative adversarial imitation learning," in *Advances in Neural Information Processing Systems (NIPS)*, Curran Associates, Inc., 2018, pp. 7461–7472.

[32] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proceedings of the twenty-first international conference on Machine learning*, ACM, 2004, p. 1.

[33] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning.," in *Aaai*, Chicago, IL, USA, vol. 8, 2008, pp. 1433–1438.

[34] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," *arXiv preprint arXiv:1206.4617*, 2012.

[35] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International Conference on Machine Learning*, 2016, pp. 49–58.

[36] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," *arXiv preprint arXiv:1904.09503*, 2019.

[37] F. Youssef and B. Houda, "Deep reinforcement learning with external control: Self-driving car application," in *Proceedings of the 4th International Conference on Smart City Applications*, ACM, 2019, p. 58.

[38] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[39] T. Tram, I. Batkovic, M. Ali, and J. Sjöberg, "Learning when to drive in intersections by combining reinforcement learning and model predictive control," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 3263–3268.

[40] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning," in *Proceedings of the 3rd Conference on Robot Learning (CoRL)*, 2019.

[41] P. Dayan and G. E. Hinton, "Feudal reinforcement learning," in *Advances in neural information processing systems*, 1993, pp. 271–278.

[42] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Advances in neural information processing systems*, 2016, pp. 3675–3683.

[43] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu, "Feudal networks for hierarchical reinforcement learning," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, pp. 3540–3549.

[44] F. Stulp and S. Schaal, "Hierarchical reinforcement learning with movement primitives," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*, IEEE, 2011, pp. 231–238.

[45] R. Strudel, A. Pashevich, I. Kalevatykh, I. Laptev, J. Sivic, and C. Schmid, "Combining learned skills and reinforcement learning for robotic manipulations," *arXiv preprint arXiv:1908.00722*, 2019.

[46] B. Wu, J. K. Gupta, and M. Kochenderfer, "Model primitives for hierarchical lifelong reinforcement learning," *Autonomous Agents and Multi-Agent Systems*, vol. 34, no. 1, pp. 1–38, 2020.

[47] H. M. Le, N. Jiang, A. Agarwal, M. Dudík, Y. Yue, and H. Daumé III, "Hierarchical imitation and reinforcement learning," *arXiv preprint arXiv:1803.00590*, 2018.

[48] A. H. Qureshi, J. J. Johnson, Y. Qin, T. Henderson, B. Boots, and M. C. Yip, "Composing task-agnostic policies with deep reinforcement learning," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=H1ezFREtwH.

[49] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 6292–6299.

[50] G. Comanici and D. Precup, "Optimal policy switching algorithms for reinforcement learning," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1 - Volume 1*, ser. AAMAS 10, Toronto, Canada: International Foundation for Autonomous Agents and Multiagent Systems, 2010, 709714, ISBN: 9780982657119.

[51] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[52] D. Sadigh, A. D. Dragan, S. S. Sastry, and S. A. Seshia, "Active preference-based learning of reward functions," in *Proceedings of Robotics: Science and Systems (RSS)*, 2017. DOI: 10.15607/RSS.2017.XIII.053.

[53] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Proceedings of the 1st Conference on Robot Learning (CoRL)*, 2017.

[54] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[55] M. O'Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," in *Advances in Neural Information Processing Systems*, 2018, pp. 9827–9838.