B-GAP: Behavior-Guided Action Prediction for Autonomous Navigation

Preprint · November 2020

CITATIONS
0

READS
340

3 authors, including:

Angelos Mavrogiannis
University of Maryland, College Park
2 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Dinesh Manocha
University of Maryland, College Park
1,006 PUBLICATIONS **31,216** CITATIONS
SEE PROFILE

B-GAP: Behavior-Guided Action Prediction and Navigation for Autonomous Driving

Angelos Mavrogiannis, Rohan Chandra, and Dinesh Manocha
University of Maryland, College Park
(Code and Videos at https://gamma.umd.edu/bgap)

***Abstract*— We present an algorithm for behaviorally-guided action prediction and local navigation for autonomous driving in dense traffic scenarios. Our approach classifies the driver behavior of other vehicles or road-agents (aggressive or conservative) and considers that information for decision-making and safe driving. We present a behavior-driven simulator that can generate trajectories corresponding to different levels of aggressive behaviors, and we use our simulator to train a reinforcement learning policy using a multilayer perceptron neural network. We use our reinforcement learning-based navigation scheme to compute safe trajectories for the ego-vehicle accounting for aggressive driver maneuvers such as overtaking, over-speeding, weaving, and sudden lane changes. We have integrated our algorithm with the OpenAI gym-based "Highway-Env" simulator and demonstrate the benefits of our navigation algorithm in terms of reducing collisions by $3.25 - 26.90\%$ and handling scenarios with $2.5\times$ higher traffic density.**

I. Introduction

The navigation problem for autonomous vehicles (AVs) corresponds to computing the optimal actions that enable the AV to start from starting point A and reach destination B via a smooth trajectory while avoiding collisions with dynamic obstacles or traffic agents. A key aspect of navigation is safety because the AVs are expected to keep a safe distance from other vehicles along with making driving more fuel- and time-efficient. In addition to being a central task in autonomous driving, navigation problems have also been studied extensively in the contexts of motion planning and mobile robots.

There is considerable research on designing action prediction and navigation algorithms for autonomous driving, but they are currently primarily deployed in specific driving scenarios or low-density traffic. Modern action prediction and navigation algorithms must be able to handle various driving scenarios to comply with real-world situations. One of these scenarios is dense traffic, which is commonly observed in city centers or in the vicinity of frequent or popular destinations. There are many challenges in terms of handling dense traffic environments, including computing safe and collision-free trajectories, as well as modeling the interactions between the traffic-agents. Some key issues are related to evaluating the driving behaviors of human drivers and ensuring that the driving pattern of the AV is consistent with the traffic norms. It has been observed that current AVs tend to drive hyper-cautiously or in ways that can frustrate other human drivers [1], potentially leading to fender-benders. Moreover, it is important to handle the unpredictability or aggressive nature of human drivers. For example, human drivers may act irrationally and move in front of other vehicles by suddenly changing lanes or aggressively overtaking. In 2016, Google's self-driving car had a collision with an oncoming bus during a lane change [2]. In this case, the ego-vehicle assumed that the bus driver was going to yield; instead, the bus driver accelerated. Overall, we need better action prediction and navigation methods that can account for such behaviors.

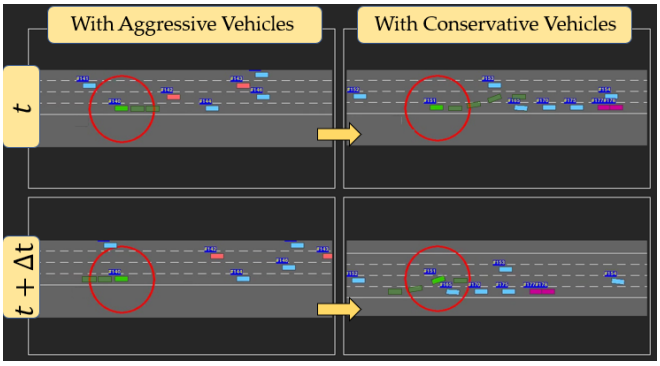


Fig. 1: **Behavior-guided Action Prediction and Navigation:** We highlight the predicted actions and the trajectory for the ego-vehicle (green) in aggressive (left) or conservative (right) environments. *(Left)* At time t, the ego-vehicle identifies the aggressive vehicle (red) ahead in the left lane. The ego-vehicle predicts that, at time $t + \Delta t$, the aggressive vehicle will make a lane-change and our action prediction algorithm decides to slow down. *(Right)* The ego-vehicle identifies a group of conservative vehicles (cyan) in its path ahead. Based on this conservative behavior, our action prediction makes a decision to change lanes by speeding up and overtaking the group of vehicles at time $t + \Delta t$. Overall, our approach results in fewer collisions and improved navigation in complex scenarios.

There is considerable work on classifying driver behaviors or style in traffic psychology [3], [4], [5]. In most cases, driving style is defined with respect to either aggressiveness [6] or fuel consumption [7]. Many recent methods have been proposed to classify the driving behaviors using trajectories [8], [9], [10] to use them for predicting the vehicle's trajectory [11], [12]. On the other hand, recent techniques for action prediction and navigation have been based on reinforcement learning, and they tend to learn optimal policies with suitable rewards for collision handling, lane changing, and following traffic rules. One of our goals is to extend these learning methods to account for driver behaviors.

Main Results: We present an algorithm for behaviorally-guided action prediction and navigation in dense traffic that takes into account the behavior of traffic agents in terms of conservativeness or aggressiveness. We use a Markov Decision process to formulate the prediction of a high-level action for an AV (e.g., making a left or right lane change) in traffic environments consisting of aggressive drivers. We present a technique to train a behavior-rich policy that uses

an enhanced traffic simulator that can generate lateral and longitudinal behaviors. Furthermore, we use the CMetric measure [10] to generate trajectories with varying levels of aggressiveness. Overall, the novel components of our work include:

1) Generating driver behaviors with varying levels of aggressiveness based on the CMetric measure. Our behavior-rich simulator is general and can generate traffic scenarios corresponding to varying traffic densities and driving styles.
2) Providing a behavior-based policy training scheme that implicitly models the interactions between traffic agents and leads to improved navigation in dense traffic. Our formulation can automatically model the behavioral interactions between aggressive and conservative traffic agents, and the ego-vehicle.
3) Using our policy to generate behavior-aware ego-vehicle trajectories in dense traffic compared to non-behavioral policies. Our resulting navigation algorithm can handle challenging traffic scenarios and results in reduced collision rates compared to non-behavior policies.

We have integrated our enhanced simulator and policy training scheme into an OpenAI gym-based highway simulator [13] and have evaluated its performance in different kinds of traffic scenarios by varying the traffic density and the behaviors of traffic agents. We observe that our approach based on the behavior-guided simulator exhibits improved performance in dense traffic scenarios over the default OpenAI gym-based simulator. We highlight the benefits of our algorithm over prior work in terms of reduction in collisions by $3.25 - 26.90\%$ and handling scenarios with $2.5\times$ higher traffic density.

II. Related Work

A. Navigation Algorithms

There is extensive work on navigation and planning algorithms in robotics and autonomous driving. At a broad level, they can be classified as techniques for vehicle control [14], techniques for motion planning, and end-to-end learning-based methods. Vehicular control methods rely on a very accurate model of the vehicle, which needs to be known a priori for navigation at high speeds or during complex maneuvers. Motion planning methods are either algebraic or probabilistic search-based [15], [16] or use non-linear control optimization [17]. Recently, many learning-based techniques have been proposed [18], [19], [20]. These include reinforcement learning algorithms that aim to find an optimal policy that directly maps the sensor measurements to control commands such as velocity or acceleration and steering angle. Optimal policies for navigation can be learned with suitable rewards for collision handling, lane-changing, and following traffic rules and regulations. However, the learned policies may not consider the behavior of other road entities or traffic entities. Some methods [19] use a linear constant velocity model that assumes that road-agents move mostly in straight lines with fixed velocities. Other methods use deep learning-based trajectory prediction models to predict the future state [20]. However, the noisy inputs obtained from this model degrade the accuracy of the navigation method [21]. Our behavior-based formulation can be integrated with most of these methods.

B. Driver Behavior Modeling

There is considerable work in traffic literature on modeling or classifying driver behavior based on driver attributes [22], [8], [9]. Other sets of classification methods depend on natural factors, including climate or traffic conditions [23] and the mental health of the drivers [24].

Many approaches have also been proposed for modeling driver behavior [25]. These strategies are based on partially observable Markov decision processes (POMDPs) [26], [27], data mining techniques [28], game theory [29], [30], graph theory [10], and imitation learning [31].

C. Trajectory Prediction and Action Prediction

There has been extensive work over the past few years on low-level trajectory prediction [32], [33], [34], [35], [36], [11], which involves directly forecasting the future positions of the ego-vehicle on the ground or in the 2D Euclidean space. In contrast, action prediction is the task of predicting the high-level semantic action ("left," "right," etc.) defined by the action-space, which is the set of all available actions. While some trajectory prediction approaches have taken driver behavior into account [11], [12], most research in action prediction has not yet considered driver behaviors. Our approach is similar in scope to [37] and [38], which use different encoder-decoder LSTM formulations to predict the high-level actions of a vehicle. However, their ultimate goal and main result is trajectory prediction and, unlike our method, they do not consider a wide range of different driving behaviors. Li et al. [12] perform action prediction using the proximity relationship between agents along with their visual features. Action prediction has also been studied in the context of risk assessment [39].

III. Background and Problem Formulation

Our goal is to classify the behaviors of traffic-agents and use them for action prediction in and trajectory computation in dense traffic. Given the current state of the environment, which may consist of the positions and velocities of all the traffic-agents at current time t, our goal is to predict the action for the ego-vehicle at the next time step $t+1$ using a trained policy. These actions typically belong to a finite set of longitudinal (speed up, slow down, maintain speed) and lateral (turn left/right) actions.

A. Markov Decision Processes (MDP) and Deep Q Networks

A standard model for sequential decision-making and planning is the Markov Decision Process (MDP). Autonomous vehicles must be able to predict high-level actions such as "left" or "right" to make safe and efficient decisions in real-time traffic. An MDP $\mathcal{M}$ consists of states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, a reward function $\mathcal{R}(s,a)$, and a state transition matrix $\mathcal{T}(s'|s,a)$ that computes the probability of the next state given the current state and action. A policy $\pi(a|s)$ represents an action distribution for each state. The goal in an MDP is to find a policy that obtains high future rewards. For each state $s \in \mathcal{S}$, the agent takes an action $a \in \mathcal{A}$. Upon taking this action, the agent receives a reward $\mathcal{R}(s,a)$ and reaches a new state s', determined from the

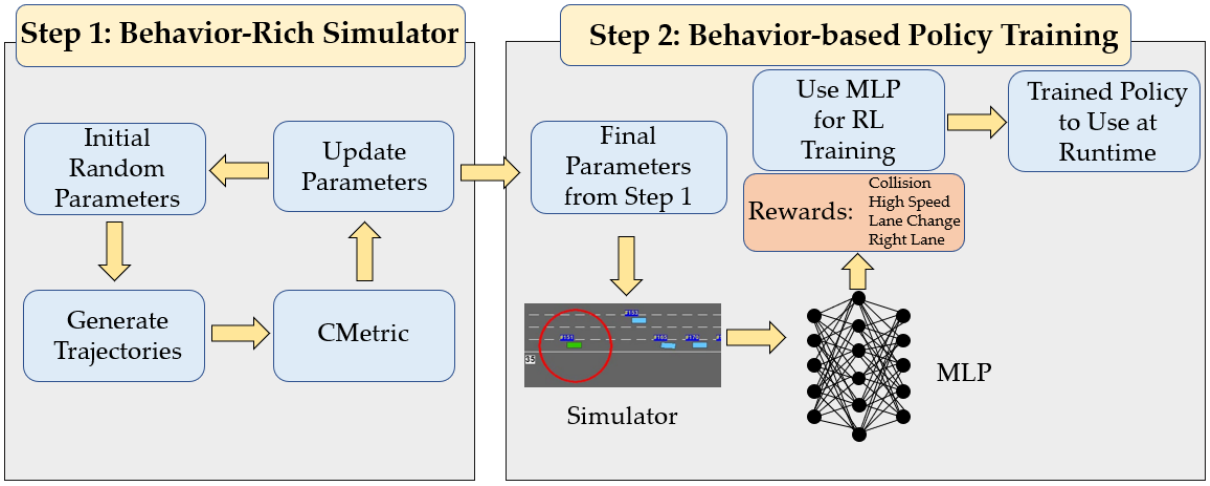


Fig. 2: **Offline Training:** We highlight our behavior-guided navigation policy for autonomous driving. We use a behavior-rich simulator that can generate aggressive or conservative driving styles. In *Step 1*, we use the CMetric behavior classification algorithm to compute a set of parameters that characterize aggressive behaviors such as over-speeding, overtaking, and sudden lane changing. In *Step 2*, we use these parameters to train a behavior-based action class navigation policy for action prediction and local navigation.

transition matrix $\mathcal{T}(s'|s,a)$. A policy $\pi(a|s)$ specifies the action that the agent will take in each state. The goal of the agent is to find the policy $\pi(a|s)$ that maps states to actions to maximize the expected discounted total reward over the agent's lifetime. The value $Q^{\pi}(s,a)$ of a given state-action pair (s,a) is an estimate of the expected future reward that can be obtained from (s,a) when following policy π. The optimal value function $Q^{*}(s,a)$ provides maximal values in all states and is determined by solving the Bellman equation:

$$Q^{*}(s,a) = \mathbb{E}\left[R(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^{*}(s',a')\right]$$

The optimal policy π is then $\pi(s) = \arg\max_{a \in \mathcal{A}} Q^{*}(s,a)$. Deep Q Networks (DQNs) [40] approximate the value function $Q(s,a)$ with a deep neural network that outputs a set of action values $Q(s,\cdot;\theta)$ for a given state input s, where θ are the parameters of the network.

B. Problem Formulation

Our goal is to design an MDP for navigation in dense traffic environments consisting of aggressive drivers. The MDP characterizes a policy $\pi(a|s)$ that outputs the optimal action a for an autonomous vehicle given the state of the environment s.

IV. Offline Training

In this section, we present our algorithm for training a behavior-based navigation policy. In order to learn such a policy, it is necessary to use a simulation environment with traffic agents that have varying behaviors, ranging from conservative to aggressive. These behaviors can be controlled using different parameters within a simulator, which are computed offline using a state-of-the-art driver behavior model [10]. Our behavior-enriched simulator is used within a deep reinforcement learning (DRL) paradigm to learn a policy for behavior-guided navigation. We highlight this offline training setup in Figure 2.

A. Enriching the Simulator with Driver Behaviors

Many techniques have been proposed to classify driver behaviors [25]. In this work, we use the CMetric [10] measure to generate aggressive driver behaviors in a simulated environment. A vehicle's behavior in a simulator can be controlled through specific parameters. Depending on the values of these parameters, a vehicle's trajectory can exhibit aggressive or conservative behavior. In our parametric simulator, the problem of generating aggressive behaviors reduces to finding the appropriate range of parameter values that can generate realistically aggressive behaviors.

In the CMetric measure, the nearby traffic entities are represented using a *proximity graph*. The vertices denote the positions of the vehicle, and the edges denote the Euclidean distances between the vehicles in the global coordinate frame. Using centrality functions [41], CMetric presents techniques to model both longitudinal (along the axis of the road) and lateral (perpendicular to the axis of the road) aggressive driving behaviors.

1) Lateral Behaviors: Sudden lane-changing, overtaking, and weaving are some of the most typical lateral maneuvers made by aggressive vehicles. We model these maneuvers using the closeness centrality, which measures how centrally placed a given vertex is within the graph. The closeness centrality is computed by calculating the reciprocal of the sum of shortest paths from the given vertex to all other vertices,

Definition IV.1. ***Closeness Centrality:*** *In a connected traffic-graph at time t with adjacency matrix A_t, let $\mathcal{D}_t(v_i, v_j)$ denote the minimum total edge cost to travel from vertex i to vertex j, then the discrete closeness centrality measure for the i^{th} vehicle at time t is defined as*

$$\zeta_c^i[t] = \frac{N-1}{\sum_{v_j \in \mathcal{V}(t) \setminus \{v_i\}} \mathcal{D}_t(v_i, v_j)}, \tag{1}$$

If a vertex is located more centrally, it results in a smaller sum, and thereby a higher value of the closeness centrality. The centrality for a given vehicle thus increases as the vehicle moves towards the center and decreases as it moves away

from the center. Lateral driving styles (i.e., styles executed perpendicular to the axis of the road) are modeled using the closeness centrality.

2) Longitudinal Behaviors: Longitudinal aggressive driving behavior consists of over-speeding. We model the over-speeding by using the degree centrality, which measures the number of neighbors of a given vertex,

Definition IV.2. ***Degree Centrality:*** *In a connected traffic-graph at time t with adjacency matrix A_t, let $\mathcal{N}_i(t) = \{v_j \in \mathcal{V}(t),\ A_t(i,j) \neq 0, \nu_j \leq \nu_i\}$ denote the set of vehicles in the neighborhood of the i^{th} vehicle with radius μ, then the discrete degree centrality function of the i^{th} vehicle at time t is defined as,*

$$\zeta_d^i[t] = \left|\{v_j \in \mathcal{N}_i(t)\}\right| + \zeta_d^i[t-1] \quad \text{such that } (v_i, v_j) \notin \mathcal{E}(\tau), \tau = 0, \ldots, t-1 \tag{2}$$

where $|\cdot|$ denotes the cardinality of a set and ν_i, ν_j denote the velocities of the i^{th} and j^{th} vehicles, respectively.

Intuitively, an aggressive or over-speeding vehicle will observe new vehicles (increasing degree) at a higher rate than a neutral or conservative vehicle. We use this property to distinguish between over-speeding and non-over-speeding vehicles. The overall method for generating these longitudinal and lateral behaviors is described as follows:

Algorithm 1:

1) In the first episode (iteration), we begin by randomly choosing $\Xi(A)$, which represents the set of simulation parameters for generating aggressive agents.
2) $\Xi(A)$ is then passed as an input to the simulator to generate the trajectories for aggressive vehicles. These vehicles perform both longitudinal and lateral aggressive maneuvers.
3) Next, we compute the likelihood of these trajectories being aggressive using centrality values obtained using CMetric. Based on this likelihood feedback, we update the parameters, $\Xi(A)$, for the next episode.
4) We repeat this process until the Time Deviation Error (TDE) [10] is below an acceptable threshold.

B. Training a Behavior-Rich Policy

We frame the underlying action prediction and navigation problem as a Markov Decision Process (MDP) represented by $\mathcal{M} := \{\mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma, \mathcal{R}\}$. The sets of possible states and actions are denoted by $\mathcal{S}$ and $\mathcal{A}$, respectively. $\mathcal{T} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ captures the state transition dynamics, γ is the discounting factor, and $\mathcal{R}$ is the set of rewards defined for all of the states in the environment. We explain each of these further below:

State Space: The state of the world S at any time step is equal to a matrix $F \times V$, which includes the state s of every vehicle in the environment. V is the number of vehicles considered and F is the number of features used to represent the state of a vehicle: F includes the following features:

- *presence*: a binary variable denoting whether a vehicle is observable in the vicinity of the ego-vehicle.
- x: the longitudinal coordinate of the center of a vehicle.
- y: the lateral coordinate of the center of a vehicle.
- v_x: the longitudinal velocity of a vehicle.
- v_y: the lateral velocity of a vehicle.

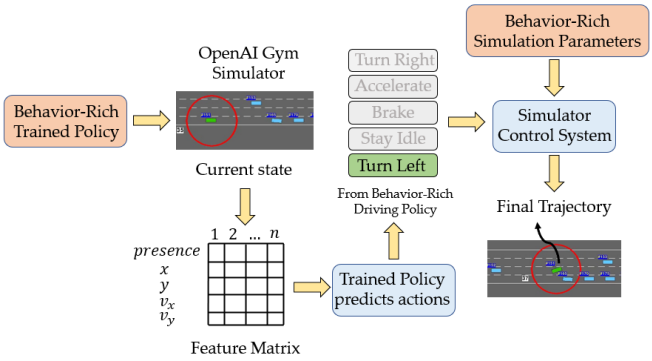


Fig. 3: **Runtime:** We use our behavior-guided trained policy and the final simulation parameters computed using offline training. During an episode at runtime, we use the trained policy to predict the next action of the ego-vehicle given the current state of the traffic environment, which is represented in the form of a feature matrix. The predicted action (in this case, "turn left") is converted into the final local trajectory using the internal controls of the simulator, modified by the parameters that take into account the behavior of traffic agents.

Environment: The environment consists of a highway road with four single-direction lanes, along with conservative and aggressive traffic agents that are generated using CMetric.

Action Space: The ego-vehicle can take five different actions: $\mathcal{A}$ ={"accelerate", "decelerate", "right lane-change", "left lane-change", "idle"}.

Reward Distribution: The philosophy for orchestrating the reward function $\mathcal{R}$ is based on our objective for training an agent that can safely and efficiently navigate in dense traffic while respecting other road agents in its neighborhood. To this end, we use four rules to determine the reward of a state. At every time step t, collision with any (conservative or aggressive) road-agent earns the ego-vehicle r_C points, lane changing earns r_{LC} points, staying in the rightmost lane gives r_{RL} points, and maintaining high speed earns r_{HS} points:

$$\mathcal{R}^t = r_C^t + r_{LC}^t + r_{RL}^t + r_{HS}^t.$$

State Transition Probability: The state transition matrix $\mathcal{T}$ boils down to a state transition probability $P(s'|s)$, which is defined as the probability of beginning from a current state s and arriving at a new state s'. This probability is calculated by the kinematics of the simulator, which depend on the underlying motion models (described further in Section V), and thus it is equal to 1, establishing a deterministic setting.

C. Reinforcement Learning-based Navigation

Learning to navigate autonomously in dense traffic environments is a difficult problem due to both the number of agents involved and the inherent uncertainty in their road behavior. The problem becomes even more challenging when the agents demonstrate varying behaviors ranging from conservative behaviors, which can disrupt the smoothness of the traffic flow, to aggressive maneuvers, which increase the probability of an accident. We integrate these behaviors into the simulator through the use of CMetric and thereby generate more realistic traffic scenarios.

Using these behaviors, we employ a Deep Reinforcement Learning setup to achieve autonomous navigation. Specifi-

TABLE I: We use CMetric to obtain the simulation parameters that define the conservative and aggressive vehicle classes.

Model	Parameter	Conservative	Aggressive
IDM [42]	Time gap (T)	1.5s	1.2s
	Min distance (s_0)	5.0 m	2.5 m
	Max comfort acc. (a)	3.0 m/s^2	6.0 m/s^2
	Max comfort dec. (b)	6.0 m/s^2	9.0 m/s^2
MOBIL [43]	Politeness (p)	0.5	0
	Min acc gain (Δa_{th})	0.2 m/s^2	0 m/s^2
	Safe acc limit (b_{safe})	3.0 m/s^2	9.0 m/s^2

cally, we use a Multilayer Perceptron (MLP) that receives an observation of the state space as input and implicitly models the behavioral interactions between aggressive and conservative agents and the ego-vehicle. Ultimately, the MLP learns a function that receives a feature matrix that describes the current state of the traffic as input and provides us with the optimal Q values of the state space. Finally, the ego-vehicle can use the learned model during evaluation time in order to navigate its way around the traffic by choosing the best action that corresponds to the maximum Q value for its state at every time step.

V. Runtime Navigation and Trajectory Computation

At runtime, our goal is to perform behaviorally-guided navigation using the RL policy trained offline. The traffic simulator provided by the Open AI Gym environment is modified to account for driver behaviors in a fashion similar to the offline training phase (explained in Section IV). At each time step, we record the state of the environment, which consists of the positions and velocities of all the vehicles in that frame, as described in Section IV-B. We store the state in a feature matrix that is passed to the trained behavior-guided policy, computed from the offline training phase. The policy is then used to predict the optimal action for the next time step, which is passed to the simulator control system and executed to generate the final trajectory. We present an overview in Figure 3.

The motion models and controllers used by the simulator compute the final local trajectory, guided by the high-level action predicted by the trained policy. The linear acceleration model is based on the Intelligent Driver Model (IDM) [42] and is computed via the following kinematic equation,

$$\dot{v}_\alpha = a\left[1 - (\tfrac{v_\alpha}{v_0^\alpha})^4 - (\tfrac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha})^2\right]. \quad (3)$$

Here, the linear acceleration, $\dot{v}_\alpha$, is a function of the velocity v_α, the net distance gap s_α, and the velocity difference Δv_α between the ego-vehicle and the vehicle in front. The deceleration term depends on the ratio of the desired minimum gap ($s^*(v_\alpha, \Delta v_\alpha)$) and the actual gap ($s_\alpha$), where $s^*(v_\alpha, \Delta v_\alpha) = s_0 + vT + \frac{v\Delta v}{2\sqrt{ab}}$. s_0 is the minimum distance in congested traffic, vT is the distance while following the leading vehicle at a constant safety time gap T, and a, b correspond to the comfortable maximum acceleration and comfortable maximum deceleration, respectively.

Complementary to the longitudinal model, the lane changing behavior is based on the MOBIL [43] model. As per this model, there are two key aspects to keep in mind:

1) *Safety Criterion*: This condition checks if, after a lane change to a target lane, the ego-vehicle has enough room to accelerate. Formally, we check if the deceleration of the successor a_{target} in the target lane exceeds a pre-defined safe limit b_{safe}:
$$a_{\text{target}} \geq -b_{safe}.$$
2) *Incentive Criterion*: This criterion determines the total advantage to the ego-vehicle after the lane change, measured in terms of total acceleration gain or loss. It is computed with the formula
$$\tilde{a}_{\text{ego}} - a_{\text{ego}} + p(\tilde{a}_n - a_n + \tilde{a}_o - a_o) > \Delta a_{th},$$
where $\tilde{a}_{\text{ego}} - a_{\text{ego}}$ represents the acceleration gain that the ego-vehicle would receive after the lane change. The second term denotes the total acceleration gain/loss of the immediate neighbors (the new follower in the target, a_n, and the original follower in the current lane, a_o) weighted with the politeness factor p. By adjusting p, the intents of the drivers can be changed from purely egoistic ($p = 0$) to more altruistic ($p = 1$). We refer the reader to [43] for further details.

The lane change is executed if both the safety criterion is satisfied and the total acceleration gain is more than the defined minimum acceleration gain Δa_{th}. We use CMetric to adjust the parameters $p, \Delta a_{th}, b_{safe}$, to introduce aggressive and conservative driving behaviors (Table I).

VI. Experiments and Results

A. Simulator

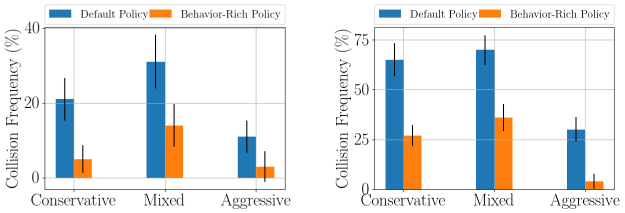
We used the OpenAI gym-based highway-env simulator [13] and enriched it with various driving behaviors using CMetric (as shown in Figure 2). In our benchmarks, we consider dense traffic scenarios on a four-lane, one-way highway environment populated by vehicles with various driving behaviors.

B. Training

We use the Deep Q Learning implementation provided in [44] and train a Multilayer Perceptron (MLP) model using PyTorch. To avoid sampling correlated samples during the training of the Deep Q network agent, we use Prioritized Experience Replay [45]. The MLP model receives as input a feature matrix that includes the features for each vehicle present in the current simulation. The simulator provides an observation from the environment in the form of a $F \times V$ matrix, where V is the number of vehicles in the simulation and F is the number of features that constitutes the state space of each vehicle.

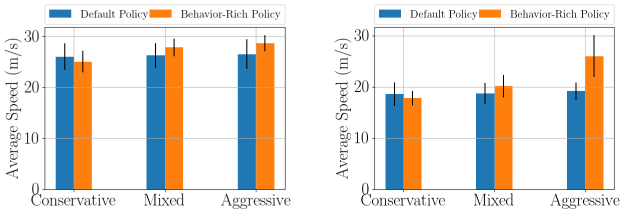
C. Reward Value Computation

We observe that increasing the collision penalty tends to slow down the ego-vehicle, highlighting the importance of avoiding collisions and reinforcing the importance of respecting the surroundings. Therefore, to achieve the right balance between high average speed and collision avoidance, we increase the reward for the lane-changing behavior to motivate the agent to prioritize lane changing rather than decelerating to avoid collisions. However, overshooting this



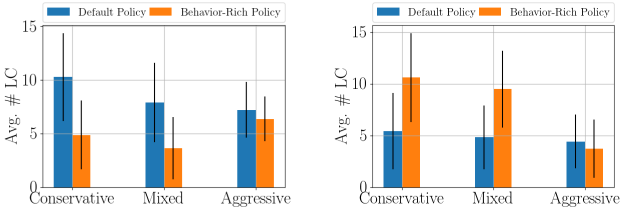
(a) Collision frequency for $n = 5$ vehicles.

(b) Collision frequency for $n = 40$ vehicles.



(c) Average speed for $n = 5$ vehicles.

(d) Average speed for $n = 40$ vehicles.



(e) Average number of lane changes for $n = 5$ vehicles.

(f) Average number of lane changes for $n = 40$ vehicles.

Fig. 4: **Evaluation:** The behavior-rich policy is compared with the default policy in traffic scenarios of varying density and driver behavior. The behavior-rich policy leads to improved navigation and reduced number of collisions (Figures 4a, 4b) by adjusting the speed (Figures 4c, 4d) of the ego-vehicle to the behaviors of its neighbors (higher average speed for higher percentages of aggressive neighbors) and decreasing its lane changes (Figures 4e, 4f) to respect unpredictable drivers. When $n = 5$ in the aggressive scenario (Figure 4e), the number of lane changes is higher, as the ego-vehicle is more confident to perform lane-changing maneuvers due to the sparsity of the traffic.

reward causes the ego-vehicle to perform unnecessary lane changes, which can lead to collisions.

D. Implementation

We use the Deep Q Learning algorithm to train the 3-layer MLP model for 3000 episodes (episode duration=60 seconds) in order to tune the reward values using the Epsilon Greedy policy. We set $\epsilon = 1$ initially with 0.05 decay. We save the reward values for every 50 episodes and test the model for 20 episodes. Our main focus is to achieve a low collision frequency, while maintaining the speed of the ego-vehicle at satisfactory levels that guarantee efficient navigation and smooth traffic flow. To this end, we manually tune the reward values monitoring the collision frequency and the average speed of the ego-vehicle at test time. We increase the reward value r_{HS} when the ego-vehicle is moving too slowly and decrease the reward for collisions r_C, thereby penalizing collisions more, when the ego-vehicle is prioritizing higher speed over collision avoidance. We also keep the value of r_{RL} at a low level so that the ego-vehicle does not stay too long in the right lane. Furthermore, we tune r_{LC} based on the pattern, according to which the ego-vehicle performs a lane change. More specifically, we visually inspect the performance at test time, and increase r_{LC} when the ego-vehicle tends to overstay behind other vehicles while the adjacent lanes are empty, to encourage overtaking maneuvers. On the contrary, we decrease r_{LC} when the ego-vehicle performs many "unnecessary" lane changes, since they disrupt the smoothness of the generated trajectory and increase the possibility of an accident. Finally, we use the Adam optimizer with a learning rate of $\eta = 0.0005$ and the Mean Squared Error (MSE) loss function.

E. Evaluation and Results

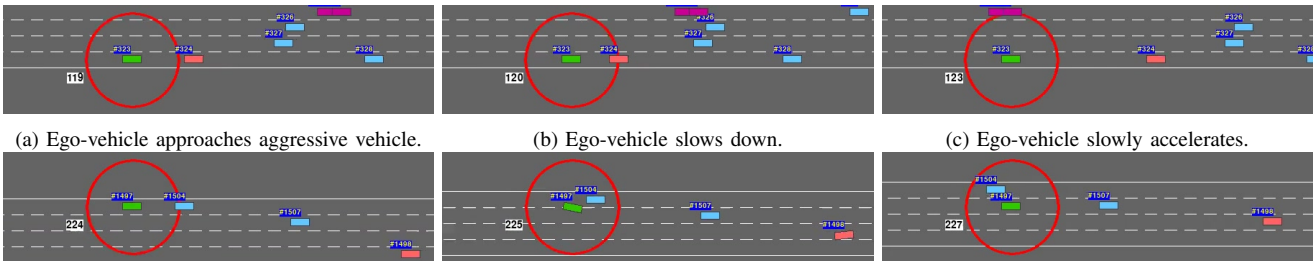
We evaluate the performance of our method in both sparse and dense traffic scenarios by varying the number of aggressive and conservative agents. We apply three metrics for evaluation averaged over multiple 100-episode runs,

- *Collision frequency* (%) of the ego-vehicle measured as a percentage over the total test runs.
- *Average speed* (m/s) of the ego-vehicle, since it captures the distance per second covered in a varying time interval.
- *Number of lane changes* performed by the ego-vehicle on average during the given duration.

In terms of density, we consider a sparse traffic scenario ($n = 5$) and a dense traffic scenario ($n = 40$). In terms of driver behavior, we examine three distinct types of traffic *(Conservative, Mixed, Aggressive)*. Conservative traffic corresponds to fully conservative agents, mixed traffic corresponds to 50% aggressive agents and 50% conservative agents, and aggressive traffic corresponds to 100% aggressive agents. Figure 4 shows the performance of our behavior-rich policy compared to the default policy generated by the simulator which does not take into account driver behavior.

In both the sparse (Figure 4a) and the dense (Figure 4b) scenario, our behavior-rich policy leads to a significantly lower collision rate compared to the baseline policy of the simulator. We observe fewer collisions when all the road-agents are either aggressive or conservative, as compared to when the traffic is mixed. This occurs because the ego-vehicle gets confused by the combination of slow-moving vehicles and aggressive agents.

The average speed of the ego-vehicle increases (Figures 4c, 4d) as the traffic ranges from conservative to aggressive with our behavior-rich policy as slow-moving conservative vehicles tend to slow down the ego-vehicle. On the contrary, the speed in the default policy remains almost unchanged indicating that the ego-vehicle is unable to adjust to the behaviors of the road-agents and maintains a uniform behavior. Finally, the number of lane changes performed by the ego-vehicle decreases as the traffic becomes more aggressive for both policies (Figures 4e, 4f), with the exception of the sparse aggressive scenario (4e), where the behavior-rich policy leads to an increased number of lane changes. In that case, the ego-vehicle leverages the sparsity of the traffic and, influenced by its aggressive neighbors, attempts lane-changing maneuvers more confidently. We show qualitative results in Figure 5.



(a) Ego-vehicle approaches aggressive vehicle. (b) Ego-vehicle slows down. (c) Ego-vehicle slowly accelerates.

(d) Ego-vehicle approaches conservative vehicle. (e) Ego-vehicle switches to adjacent lane. (f) Ego-vehicle overtakes conservative vehicle.

Fig. 5: **Behavior-Guided Action Prediction & Navigation:** Interaction with aggressive (*top*) and conservative (*bottom*) vehicles. We indicate the ego-vehicle, aggressive and conservative vehicles by green, red, and blue colors, respectively. (*Top*) The ego-vehicle senses that the red vehicle is aggressive and therefore decides to slow-down instead of overtaking. (*Bottom*) The ego-vehicle notices the conservative vehicle in front and decides to confidently overtake it. Our approach can automatically generate such trajectories for the ego-vehicle.

TABLE II: **Comparison with Prior Work:** We compare B-GAP with prior DRL-based navigation methods. For fair comparison, we compare % collisions along with the number of vehicles, the scenario used for testing, and the number of episodes used for training. '-' indicates that the corresponding information is unspecified.

Methods	Scenario	# vehicles	# episodes	% Collisions
DDPG [46]	Highway	4	4000	10.13
PS-DDPG [47]	Highway	5	600	11.40
Liu et al. [48]	Intersection	15	4000	6.25
Leurent et al. [49]	Intersection	-	3000	29.90
B-GAP	Highway	5	3000	**3.00**

In summary, we observe that our behavior-guided action prediction and navigation algorithm offers the following benefits:

- *Increased Safety*: Our behavior-rich policy results to a significantly lower collision frequency compared to the default policy baseline.
- *Behavior-Aware Trajectories*: Our behavior-rich policy generates behavior-aware trajectories. The ego-vehicle learns to adjust its behavior based on the behavior of its neighbors, both in terms of speed regulation and the number of executed lane changes.

F. Comparison with DRL-based Navigation Methods

We compare with state-of-the-art methods that broadly follow two kinds of approaches. The first set of approaches [48], [49] consists of training a navigation policy using the Open AI Gym simulator. However, these methods are tested in simple intersection scenarios with at most 4 vehicles crossing the intersection. The second set of approaches [46], [50] consists of training the navigation policy in highway environments with $4-5$ vehicles with a maximum density of 15 vehicles. However, they use a different simulator for example SUMO [51], and do not consider aggressive agents. It is not clear whether these methods can handle complex scenarios with high number of vehicles and varying behaviors, as we have evaluated (with $n = 40$) in Figure 4. Furthermore, many prior works [52], [53], [54] assume a controller or an oracle to guarantee collision-free trajectories. Our approach makes no such assumptions. Instead, our navigation method results in very few collisions in high density scenarios ($n = 40$ as compared to maximum 15 used in [47]) with aggressive agents (Figure 4).

In Table II, we compare our approach with these methods in terms of % collisions. For fair comparison, we use $n = 5$ in the aggressive environment since prior methods [46], [47] for the highway environment have also tested with $4 - 5$ vehicles. Since the performance of these methods depends on various factors, including the number of vehicles and the number of training episodes, the type of the simulation environment and other hyperparameters, we include some of these details as well.

VII. Conclusion, Limitations, and Future Work

We proposed a new approach for action prediction and AV navigation in dense traffic environments composed of conservative and aggressive human drivers. Our algorithm is based on deep reinforcement learning and considers driver behavior to perform *behaviorally-guided* navigation. We enrich an existing simulator with driver behavior and use that to train a policy that accounts for these varying behaviors. We have integrated our algorithm in the OpenAI gym-based "Highway-Env" simulator, and our preliminary results in many benchmarks are encouraging.

Our method has several limitations. For example, in some cases, the ego-vehicle avoids aggressive vehicles by decelerating and performing fewer lane changes, thereby acting conservatively. In some scenarios, it may be necessary to execute non-conservative actions such as increasing the frequency of lane changes or following local traffic norms. In terms of future work, we plan to extend our approach to different environments like roundabouts, intersections, and parking lots and perform more detailed evaluation. Furthermore, we plan to develop efficient navigation techniques to handle heterogeneous traffic-agents such as pedestrians or bicycles. Our current approach assumes perfect state information of other agents on the road and in the future we need to account for perception errors.

Acknowledgements

This work was supported in part by ARO Grants W911NF1910069 and W911NF1910315, Semiconductor Research Corporation (SRC), and Intel. We would also like to thank Abhinav Modi for technical discussions.

References

[1] D. Tesla, "25 miles of full self driving, tesla challenge 2, autopilot," https://www.youtube.com/watch?v=Rm8aPR0aMDE, 2019.

[2] A. Davies, "Google's self-driving car caused its first crash," 2016.

[3] S. Anthony, "Self-driving cars still can't mimic the most natural human behavior," https://qz.com/1064004/self-driving-cars-still-cant-mimic-the-most-natural-human-behavior/, 2017.

[4] E. Vinkhuyzen and M. Cefkin, "Developing socially acceptable autonomous vehicles," in *Ethnographic Praxis in Industry Conference Proceedings*, vol. 2016, no. 1. Wiley Online Library, 2016, pp. 522–534.

[5] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proceedings of the National Academy of Sciences*, vol. 116, no. 50, pp. 24 972–24 978, 2019.

[6] A. Doshi and M. M. Trivedi, "Examining the impact of driving style on the predictability and responsiveness of the driver: Real-world and simulator analysis," in *2010 IEEE Intelligent Vehicles Symposium*. IEEE, 2010, pp. 232–237.

[7] A. Corti, C. Ongini, M. Tanelli, and S. M. Savaresi, "Quantitative driving style estimation for energy-oriented applications in road vehicles," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2013, pp. 3710–3715.

[8] J. Rong, K. Mao, and J. Ma, "Effects of individual differences on driving behavior and traffic flow characteristics," *Transportation research record*, vol. 2248, no. 1, pp. 1–9, 2011.

[9] K. H. Beck, B. Ali, and S. B. Daughters, "Distress tolerance as a predictor of risky and aggressive driving." *Traffic injury prevention*, vol. 15 4, pp. 349–54, 2014.

[10] R. Chandra, U. Bhattacharya, T. Mittal, A. Bera, and D. Manocha, "Cmetric: A driving behavior measure using centrality functions," *arXiv preprint arXiv:2003.04424*, 2020.

[11] R. Chandra, T. Guan, S. Panuganti, T. Mittal, U. Bhattacharya, A. Bera, and D. Manocha, "Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms," *IEEE Robotics and Automation Letters*, 2020.

[12] C. Li, Y. Meng, S. H. Chan, and Y.-T. Chen, "Learning 3d-aware egocentric spatial-temporal interaction via graph convolutional networks," *arXiv preprint arXiv:1909.09272*, 2019.

[13] E. Leurent, "An environment for autonomous driving decision-making," https://github.com/eleurent/highway-env, 2018.

[14] A. De Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot motion planning and control*. Springer, 1998, pp. 171–253.

[15] D. Manocha, *Algebraic and numeric techniques in modeling and robotics*. University of California at Berkeley, 1992.

[16] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[17] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[18] T. Fan, X. Cheng, J. Pan, D. Monacha, and R. Yang, "Crowd-move: Autonomous mapless navigation in crowded scenarios," *arXiv preprint:1807.07870*, 2018.

[19] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 1343–1350.

[20] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," *arXiv preprint arXiv:1809.08835*, 2018.

[21] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[22] E. R. Dahlen, B. D. Edwards, T. Tubré, M. J. Zyphur, and C. R. Warren, "Taking a look behind the wheel: An investigation into the personality predictors of aggressive driving," *Accident Analysis & Prevention*, vol. 45, pp. 1–9, 2012.

[23] A. H. Jamson, N. Merat, O. M. Carsten, and F. C. Lai, "Behavioural changes in drivers experiencing highly-automated vehicle control in varying traffic conditions," *Transportation research part C: emerging technologies*, vol. 30, pp. 116–125, 2013.

[24] P. Jackson, C. Hilditch, A. Holmes, N. Reed, N. Merat, and L. Smith, "Fatigue and road safety: a critical analysis of recent evidence," *Department for Transport, Road Safety Web Publication*, vol. 21, 2011.

[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] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Transactions on control systems technology*, vol. 26, no. 5, pp. 1782–1797, 2017.

[27] Z. Cao, E. Bıyık, W. Z. Wang, A. Raventos, A. Gaidon, G. Rosman, and D. Sadigh, "Reinforcement learning based control of imitative policies for near-accident driving," 2020.

[28] Z. Constantinescu, C. Marinoiu, and M. Vladoiu, "Driving style analysis using data mining techniques," *International Journal of Computers Communications & Control*, vol. 5, no. 5, pp. 654–663, 2010.

[29] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proceedings of the National Academy of Sciences*, vol. 116, no. 50, pp. 24 972–24 978, 2019.

[30] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions." in *Robotics: Science and Systems*, vol. 2. Ann Arbor, MI, USA, 2016.

[31] 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.

[32] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," *arXiv preprint arXiv:1805.06771*, 2018.

[33] R. Chandra, U. Bhattacharya, A. Bera, and D. Manocha, "Traphic: Trajectory prediction in dense and heterogeneous traffic using weighted interactions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8483–8492.

[34] F.-C. Chou, T.-H. Lin, H. Cui, V. Radosavljevic, T. Nguyen, T.-K. Huang, M. Niedoba, J. Schneider, and N. Djuric, "Predicting motion of vulnerable road users using high-definition maps and efficient convnets," 2018.

[35] X. Li, X. Ying, and M. C. Chuah, "Grip: Graph-based interaction-aware trajectory prediction," *arXiv preprint arXiv:1907.07792*, 2019.

[36] Y. Ma, X. Zhu, S. Zhang, R. Yang, W. Wang, and D. Manocha, "Trafficpredict: Trajectory prediction for heterogeneous traffic-agents," *arXiv preprint arXiv:1811.02146*, 2018.

[37] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/IVS.2018.8500493

[38] H. Song, W. Ding, Y. Chen, S. Shen, M. Y. Wang, and Q. Chen, "Pip: Planning-informed trajectory prediction for autonomous driving," *Lecture Notes in Computer Science*, p. 598–614, 2020. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-58589-1_36

[39] C. Li, S. H. Chan, and Y.-T. Chen, "Who make drivers stop? towards driver-centric risk assessment: Risk object identification via causal inference," *arXiv preprint arXiv:2003.02425*, 2020.

[40] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[41] F. A. Rodrigues, "Network centrality: An introduction," *A Mathematical Modeling Approach from Nonlinear Dynamics to Complex Systems*, p. 177, 2019.

[42] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[43] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[44] E. Leurent, "rl-agents: Implementations of reinforcement learning algorithms," https://github.com/eleurent/rl-agents, 2018.

[45] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2016.

[46] M. Kaushik, V. Prasad, K. M. Krishna, and B. Ravindran, "Overtaking maneuvers in simulated highway driving using deep reinforcement learning," in *2018 ieee intelligent vehicles symposium (iv)*. IEEE, 2018, pp. 1885–1890.

[47] M. Kaushik, K. M. Krishna, *et al.*, "Parameter sharing reinforcement learning architecture for multi agent driving behaviors," *arXiv preprint arXiv:1811.07214*, 2018.

[48] T. Liu, X. Mu, B. Huang, X. Tang, F. Zhao, X. Wang, and D. Cao, "Decision-making at unsignalized intersection for autonomous vehicles: Left-turn maneuver with deep reinforcement learning," *arXiv preprint arXiv:2008.06595*, 2020.

[49] E. Leurent, D. Efimov, and O.-A. Maillard, "Robust-adaptive control of linear systems: beyond quadratic costs," in *NeurIPS 2020-34th Conference on Neural Information Processing Systems*, 2020.

[50] M. Kaushik and K. Madhava Krishna, "Learning driving behaviors for automated cars in unstructured environments," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 0–0.

[51] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

[52] M. Huegle, G. Kalweit, M. Werling, and J. Boedecker, "Dynamic interaction-aware scene understanding for reinforcement learning in autonomous driving," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4329–4335.

[53] T. Li, J. Wu, and C.-Y. Chan, "Evolutionary learning in decision making for tactical lane changing," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 1826–1831.
[54] E. Galceran, A. G. Cunningham, R. M. Eustice, and E. Olson, "Multipolicy decision-making for autonomous driving via changepoint-based behavior prediction: Theory and experiment," *Autonomous Robots*, vol. 41, no. 6, pp. 1367–1382, 2017.