2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)
October 23-27, 2022, Kyoto, Japan

Planning for Negotiations in Autonomous Driving using Reinforcement Learning

Roi Reshef[1]

***Abstract*— Planning autonomous driving behaviors in dense traffic is challenging. Human drivers are able to influence their road environment to achieve (otherwise unachievable) goals, by communicating their intents to other drivers. An autonomous system that is required to drive in the presence of human traffic must thus possess this fundamental *negotiation* capability. This work presents a novel benchmark that includes a stochastic driver negotiation model and a framework for training policies to drive and negotiate based on reinforcement learning. It is shown that driving policies trained in this framework lead to greater safety, higher mission accomplishment rates and more driving comfort, and can generalize across scenarios.**

I. INTRODUCTION

Autonomous Vehicle (AV) software needs to transform raw sensory inputs into commands to the vehicle actuators to generate driving behavior that blends naturally with traffic. Most commercial *advanced driver assistance systems* (ADAS) are *reactive* to the environment, as they are able to maintain set headway or speed, brake for obstacles, react to traffic control devices, or change lane to existing gaps. Under these simplified requirements, ADAS systems composed of rules and heuristics work reasonably well. However, these approaches quickly reach their limits in more complex scenarios that include stochastic interactions between road users or involve unobservable information impinging on decision making, since these systems currently do not carry out *proactive* reasoning or optimize for the abundance of all possible outcomes.

For instance, while a vehicle approaching an on-ramp that merges into a densely populated main road is expected to give way to traffic in the target lane, waiting for an open gap may result in overly cautious behavior that fails to blend into traffic (see link at [1] for an illustrative example). However, in such situations human drivers are taught to ”negotiate” their way through communicating a cut-in intent by *explicitly* using blinker lights, or *implicitly* through lateral movements, exploiting the potentially cooperative nature of the drivers around them. Since the level of cooperation from drivers is unknown apriori, the likelihood of the multiple possible outcomes and the remaining slack for maneuvering should be (e.g. distance to the end of the lane) assessed and plans should be adapted accordingly.

This paper presents a methodology for planning in challenging driving scenarios that are stochastic and/or include unobservable information, require long-term reasoning and necessitate negotiation with other drivers to be solved efficiently. It introduces a novel model for interactions between drivers, associated with a set of highly interactive scenarios to illustrate the challenges of AV driving at different levels of traffic density. Effective state and action abstractions are used in a hybrid training pipeline of Reinforcement Learning (RL) and verifiable rules, to enable the agent to optimize its plans in order to negotiate with other drivers, while maintaining safety at all times. The primary contributions of this work are: (a) a novel formulation of interaction between drivers (b) policy structure and a RL-based training pipeline for safe negotiation and jerk-minimization (c) a reward function design that comprehensively captures: task completion, efficiency and comfort (d) a neural network architecture suitable for policy training for challenging driving scenarios.

The results show that it is possible to frame these problems as MDPs and harness a model-free training pipeline to solve them efficiently. The findings underscore that negotiation is crucial for achieving good performance in highly demanding scenarios. These new policies can implement this behavior without sacrificing either driving comfort nor efficiency.

[1]Roi Reshef is with NVIDIA, Yitzhak Sade 4, Tel Aviv, Israel `roireshef@hotmail.com, rreshef@nvidia.com`

II. RELATED WORK

The list of works on autonomous driving policies has grown considerably over the past few years with tailwinds coming from commercial products and services at different levels of autonomy. Nonetheless, only a few publications have dealt with training driving policies to negotiate with other drivers. The remainder of this section is organized to allow comparison of works across different aspects of driving policies and their applicability to the real world.

Policy training: Different paradigms implementing machine learning to optimize driving decisions have evolved. *Imitation learning*, a supervised learning paradigm for training policies to imitate expert drivers, was described in [2] who reported it was insufficient for highly interactive scenarios and suggested RL as a viable alternative. Similar conclusions were drawn in [3]–[5]. Machine learning approaches that train a policy to maximize rewards are roughly divided into two categories. In *model-based reinforcement learning*, an explicit transition model is specified and its parameters are learned during the optimization process. While works such as [6], [7] report improved sample-efficiency, this approach is mostly effective when the model is simple and straightforward. Conversely, in works based on *model-free reinforcement learning* [8], [9] a policy is optimized directly by sampling the environment dynamics model, which is an advantageous approach for problems with complex dynamics.

State encoding and neural architecture: Encoding environmental inputs compactly and efficiently constitutes an active area of research. In *end-to-end learning*, the policy is trained to control the vehicle directly from raw sensory signals [10]–[12]. While this purely machine learning approach is appealing, it is an unpopular choice for real world applications due to various limitations detailed in [13], [14]. A common practice is to use *mid-level representations* to mediate between scene understanding modules and planning. For example, a top-view rasterization followed by convolutional neural networks was applied in [2], [15], [16]. Other works have used object-to-object affordances [17], [18] that are not straightforward to generalize to complex road structures, Cartesian coordinates or the distances between them [19]–[21]. Since a representation based on Cartesian coordinate frame is sensitive to variations in road geometry, it may increase training complexity, or hurt policy generalization. To mediate that, representations based on road's coordinate-frame were combined with fully connected or convolutional network architectures [9], [22]–[25]. Although these methods use object-to-lane assignments to reduce training complexity, they rely on the manual assignment of spatial relations between objects, some of which are difficult to model in complex road structures. Finally, publications by [6], [8] suggest a permutation-invariant neural architecture which is easy to scale to complex scenarios. Their approach was adopted here and extended with self-attention [26] and residual blocks [27] to improve training stability.

Action space: There is a fairly large body of works on solving continuous actions-space problems with deep RL [28]–[30]. Although theoretically suitable for problems of any length, those methods tend to fail in practice on problems with a hierarchical structure or long horizons [31]. This issue is commonly mitigated by using an external low-level controller for trajectory tracking. To complement the latter, works such as [32], [33] train policies to output trajectories, although they lack consistency, comfort and safety guarantees. Discrete action spaces of constant actuation (acceleration and steering values) and complete lane changes at constant speeds are reported in articles such as [8], [9], [24], [34]. An integration with an external adaptive-cruise controller was used in [6]. Their simplistic design makes the vehicle prone to dynamics issues and limits maneuver coverage [35]. In contrast, here, the motion planning process of [36] is expanded with lane change abstractions to enable negotiation, increased comfort and verifiable safety.

Optimization objective: The underlying objective function that represents a naturalistic driving behavior is unknown. In its most basic form, an agent is rewarded with some positive constant for successful episode termination and zero or a negative constant for failure, as in [8], [34], [37]. Some extensions include motivating efficiency by rewarding for distance travelled [8] or penalizing for low speeds [2], [6], [9]. To motivate comfort, works such as [23], [38] penalize for changing lanes, which may be unnecessary in the presence of a general purpose comfort criterion as in [38], [39].

Enforcing safety: While some previous works such as [38]–[40] represent safety as a soft optimization objective, the impracticality of this approach is discussed in [41], which suggests an alternative approach that expands upon the widely adopted legal guidelines from The Vienna Convention on Road Traffic [42] by introducing a comprehensive accountability model for safety in driving, using formal verification.

III. BACKGROUND

The ultimate goal of the planning module is to create motion plans that guide the vehicle to its target. Since this can create a long and heavy optimization problem, conventional AV planners are often structured in hierarchy [43], [44]: A *route planner* (RP) is responsible for planning a route (coarse sequence of road segments) to follow, based on road and lane connectivity, essentially setting mid-term goals for a downstream component. Then, a *behavioral planner* (BP) optimizes for a sequence of behavioral decisions that lead to the goals it is provided with. Every decision is translated into a motion plan (or a *trajectory*), which can later be fined-tuned by a *trajectory planner* (TP).

Traditionally, a common practice in AV software stacks is to run prediction and planning sequentially to reason about how the world will evolve and optimize the agent's plans accordingly. Running these two modules sequentially can create sub-optimal results especially in highly dynamic scenarios where negotiation is required [8], [45]. This points to the need to condition predictions on the host agent's plans (and vice versa) or to the use of planning approaches that do not rely on a prediction model explicitly, such model-free RL. In this work, a goal-conditioned POMDP [46] is used to formulate the behavioral planning problem, defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, g, \gamma)$ with state space $\mathcal{S}$, action space $\mathcal{A}$ and observation space $\mathcal{O}$. A goal context $g \in \mathbb{R}^n$ is drawn at the beginning of each episode and is accessible by the agent in every frame. At each discrete time t an agent taking action $\alpha_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$ transitions to a new state $s_{t+1} \in \mathcal{S}$ according to a stochastic transition model $T(s_{t+1}|s_t, \alpha_t)$ and receives an immediate reward $r_t \in \mathbb{R}$ according to the reward model $R(r_t|s_t, \alpha_t, g)$. In a POMDP, the agent only has access to an observation $o_t \in \mathcal{O}$ drawn from an observation model $O(o_t|s_t)$ which may obstruct some of the information of s_t. Ultimately, the goal of the agent is to find the optimal policy π^* that maximizes the expected sum of future rewards, as shown in Eq. 1. While different approaches were developed for solving POMDPs, a common practice in modern deep RL setups is to use k-Markov policies where the inputs consist of k historical observations from the environment. As in [8], this was set at $k = 1$, which conceptually marginalizes out the unobserved information based on its statistics during training. Then, standard RL techniques for MDPs can be used to solve for π^* with state space $\mathcal{O} \times \mathbb{R}^n$.

$$\pi^*(\alpha_t|o_t, g) = \arg\max_{\pi} E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}|o_t, g\right] \qquad (1)$$

IV. HYBRID FRAMEWORK FOR DRIVING WITH NEGOTIATIONS

The proposed planning framework's structure is hierarchical, as detailed in previous section, as it lessens the exclusive responsibility of a behavioral policy to optimize for highest expected sum of rewards over a single road segment. In the rest of this section we describe its components and flow.

A. *Action Abstractions for Safe Negotiation*

Based on [3], [41] the action space presented here is driven by semantic targets (desires). Each desire $\delta_i = (v_i, u_i, h_i)$ in the desires set $\mathcal{D} = [0, v_{max}] \times \mathcal{U} \times \{\leftarrow, \uparrow, \rightarrow\}$ is a tuple of three components: a target longitudinal velocity $v_i \in [0, v_{max}]$ from a predefined grid of velocities, an urgency level $u_i \in \mathcal{U}$ used to create different time horizons for velocity changes and a lateral shift $h_i \in \{\leftarrow, \uparrow, \rightarrow\}$ used to shift the lateral target discretely over a fixed set of offsets. An exception is the additional emergency brake action desire $\delta_{emergency}$ that represents constant deceleration a^{max}_{decel} and is required for safety. A validation layer is employed to mask actions for feasibility and safety, based on RSS formulas from [41].

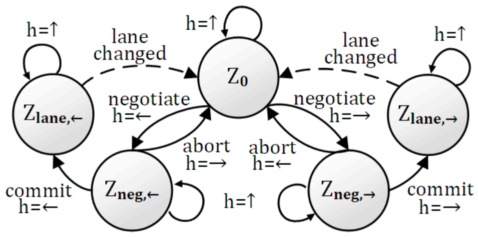
Motion planning: For local planning of motion primitives the jerk-optimization procedure in [36] is used. Formally, let trajectory $\tau_i = (s_i(t), d_i(t))$ be represented by a pair of longitudinal and lateral polynomials $s_i(t)$ and $d_i(t)$ in road's coordinate frame. Since $s_i(t)$ and $d_i(t)$ are fourth and fifth order polynomials solving the boundary conditions in Eq. 2, they are jerk-optimal. An optimal longitudinal transition time T_s is found by minimizing the cost function in Eq. 3 that trades-off squared-jerk and time. For each action, the weights that match its urgency level u_i are used such that urgency levels control transition times dynamically. Finally, all the trajectories generated belong to the class of jerk-optimal trajectories with a terminal pose of velocity converged to v_i and a lateral offset to Δ_i with the vehicle rotated parallel to the road. The lateral transition time T_d is predefined for each LFSM state transition.

$$\begin{aligned} \dot{s}_i(T_s) = v_i, \quad \ddot{s}_i(T_s) = 0, \\ d_i(T_d) = \Delta_i, \quad \dot{d}_i(T_d) = \ddot{d}_i(T_d) = 0 \end{aligned} \tag{2}$$

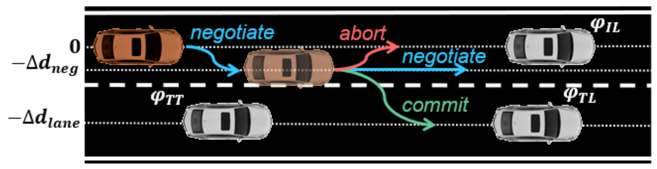
$$T_s^* = \arg\min_T \left[w_J \int_{t=0}^{T} J^2(t)dt + w_T T \right] \tag{3}$$

This framework has two noticeable advantages for local optimization of motions trajectories: (a) it provides jerk-minimization procedure for both speed and headway control (although headway control isn't dealt with here, for simplicity), which is expandable for different levels of urgency (b) it is computationally lightweight and easily parallelizable. While jerk minimization is provided for a given set of boundary conditions, a higher-level policy π is required to supply consistent boundary conditions as an input to this motion planning procedure in order to maintain a jerk-minimizing motion profile over time.

Lane change state machine: The Lateral Finite State Machine (LSFM), shown in Fig. 1a, is an abstraction used for controlling the target lateral offset of motion plans over the discrete set $\{-\Delta d_{lane}, -\Delta d_{neg}, 0, \Delta d_{neg}, \Delta d_{lane}\}$ of lateral targets relative to the current lane. At the initial state z_0 the vehicle is performing lane-centering by setting the boundary condition $d_i(t_d^*) = 0$. Then, applying any action corresponding to a desire $\delta_i = (v_i, u_i, h_i)$ with $h_i = \uparrow$ leaves the LFSM state unchanged, while $h_i \in \{\leftarrow, \rightarrow\}$ results in LSFM state transitions. More specifically, applying an action with $h_i = \leftarrow$ while in z_0 results in a transition to LFSM state $z_{neg,\leftarrow}$ denoting the new lateral target $d_i(t_d^*) = \Delta d_{neg}$, the negotiation offset towards the left adjacent lane. Consequently, while in $z_{neg,\leftarrow}$, during the physical transition or after has been completed, the agent may choose to apply any action with $h_i = \uparrow$ to stay in $z_{neg,\leftarrow}$, apply an action with $d_i = \rightarrow$ to abort the maneuver by transitioning back to z_0, or apply an action with $h_i = \leftarrow$ to commit to and initiate a full lane-change maneuver into the left adjacent lane by transitioning into LFSM state $z_{lane,\leftarrow}$, during which actions with $h_i \in \{\leftarrow, \rightarrow\}$ are masked out (to represent the commitment to complete the lane change). Finally, an LFSM state transition to z_0 is triggered once the agent has completed changing lanes. The logic applies symmetrically for right lane changes.



(a)



(b)

Fig. 1. Lane changes with LFSM (a) LFSM states and transitions (b) LFSM illustration; top-down view.

B. *State Encoding and Neural Architecture*

The kinematic state of the host vehicle is given to the agent in o_t, as well as its internal LFSM state. The road structure is assumed to be known, as well as vehicle assignments to lanes. The longitudinal position is represented relative to a predefined anchor in the scene (i.e. merging point in lane merge scenarios, end of road segment in highway scenario) for better generalization. In scenarios where a perturbation of map features is used (such as merging lane length or speed limit), the features are added to the host inputs vector. The goal context g is also provided to the agent whenever applicable, and is represented as a one-hot vector, shifted dynamically so it is centered around the agent as it

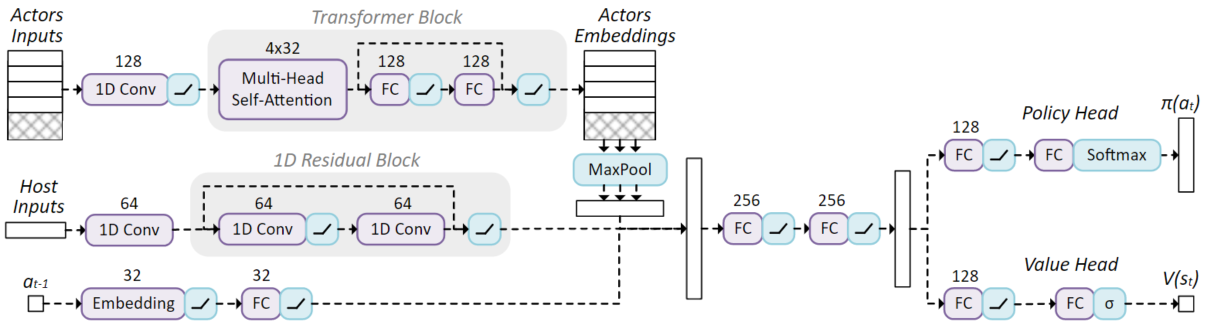


Fig. 2. The neural network architecture used in this work. Numbers represent the output sizes of trained layers.

transitions between lanes. For encoding surrounding vehicles, ego-centric projections in road-coordinates are used.

The neural architecture in this work (shown in Fig. 2) is tasked with mapping the observation o_t and goal context g to policy $\pi(\alpha_t|o_t, g)$ and state-value $V(o_t, g)$ outputs. The first stream of inputs includes a padded 2D matrix that represents the features of the dynamically-sized list of actors. It is fed into a 1D convolutional layer followed by a transformer self-attention block [26] and a max-pooling operator over the channel dimension to create permutation invariance across actors and to output a fixed-sized embedding vector. The second stream of inputs is fixed-sized and includes a concatenation of the state of the host vehicle, map attributes and goal context. These are fed into a 1D convolutional layer and a 1D residual block [27] to generate a host vehicle embedding vector. Finally, a third stream processes the action from the last step α_{t-1} with an embedding and a fully-connected layers. All outputs are concatenated to a scene embedding vector, which then goes through two fully connected layers and splits to policy and value heads, both are composed of fully connected layers. All layers in the network are followed by ReLU activations except for the output layers of the policy and value heads, which have Softmax and Sigmoid activations, respectively.

C. *Universal Reward Function*

Since non-learned safety layer is used, the responsibility of the learned policy π is reduced to optimizing action selection for task completion, efficiency and comfort. To motivate efficiency and comfort, in each simulation step the agent is penalized for deviating from the target velocity and for the jerk applied. The velocity deviation is normalized between 0 and 1 (assuming $v_h <= 2v_{target}$), skewed and rescaled with c_d and c_v respectively. Similarly, the sum of the squared jerk is extracted from the trajectory and rescaled by c_j. Since all actions are restricted to terminate with zero acceleration, both the acceleration and jerk are inherently minimized.

$$r_t = 1 - c_v \left[\frac{|v_h - v_{target}|}{v_{target}}\right]^{c_d} - c_j \int J^2(t)dt \\ + \frac{1}{1-\gamma}\left[\mathbb{1}_s(s_{t+1}) + c_h \cdot \mathbb{1}_h(s_{t+1})\right] \tag{4}$$

The agent is also rewarded upon successful episode termination; i.e., when $\mathbb{1}_s(s_{t+1}) = 1$. As in [6] the terminal reward is rescaled by the sum of the infinite geometric series $^1/_{1-\gamma}$ so that upon a successful episode termination, the agent is rewarded as though it were driving forever with no penalties. In addition, when an episode terminates in a handover event; i.e., when $\mathbb{1}_h(s_{t+1}) = 1$, the agent is rewarded proportionally to $c_h \in [0, 1]$. Note that in the complete formulation in Eq. 4, the reward function parameters control three intuitive behavioral trade-offs:

- **Aggressiveness:** c_v and c_j control the extent to which the agent is motivated to accelerate vs. to avoid jerk.
- **Risk-tolerance** is controlled by c_h with high values increasing the handover rate to prevent failures (which also decreases the success rate), and vice-versa.
- **Short-term vs. long-term:** the magnitude of terminal rewards relative to immediate rewards is controlled by γ (similar to its original reward discounting role).

V. EXPERIMENTS

To simulate the driving environment the SUMO simulator [47] and FLOW python package [48] were used with all agents controlled at 10Hz and the host agent's decisions at 1Hz. Policies were trained with the A3C on-policy model-free RL algorithm [28] implemented in the RLlib library [49], and neural networks implemented in the PyTorch library [50]. Training was done on a single machine (32-cores), with a linear schedule for entropy decay and the hyper-parameters from Table I, unless stated otherwise.

Two scenarios were generated for training in which the agent was required to perform long-term reasoning with urgency and in different traffic densities (see Fig. 3): a four-lane highway scenario in which the host agent enters from leftmost lane and is tasked with getting to a designated exit (out of 4 options) by controlling itself in both dimensions through multiple lane changes before the road ends, and a lane merge scenario where the host vehicle is tasked with merging into a main lane filled with traffic.

Actor behavior model: The longitudinal motions of other actors were governed by the widely used Intelligent Driver Model (IDM) [51] throughout all scenarios. Inspired by [34], this model was extended to include different levels of cooperation. By contrast, actor cooperativeness was reflected through increased/decreased target headway with respect to

TABLE I

HYPER-PARAMETERS USED ACROSS THE EXPERIMENTS

Environment and Abstractions		
Perception horizon $[m]$	$\Delta s_{horizon}$	150
Reaction distance $[m]$	Δs_{react}	75
Reaction lateral offset $[m]$	Δd_{react}	0.3
Negotiation offset $[m]$	Δd_{neg}	0.7
Full lane change duration $[s]$	$t_d(\Delta d_{lane}\|0)$	6
Time to negotiation offset $[s]$	$t_d(\Delta d_{neg}\|0)$	2
Time to commit lane change $[s]$	$t_d(\Delta d_{lane}\|\Delta d_{neg})$	5
Time to abort lane change $[s]$	$t_d(0\|\Delta d_{neg})$	2
Max. actors in state	N_{max}	30
Target velocity (host) $[m/s]$	v_{target}	25
Max velocity (host) $[m/s]$	v_{max}	25
Highway road length $[m]$	$L_{highway}$	600
Merging lane length $[m]$	L_{merge}	250
Reward		
Velocity magnitude coeff.	c_v	2.0
Velocity decay coeff.	c_d	3.0
Jerk penalty coeff.	c_j	0.1
Training		
Learning rate	η	1e-4
Discount factor	γ	0.99
Batch size	N_{batch}	250
Initial entropy coeff.	ϵ_0	0.01
Final entropy coeff.	$\epsilon_{N_{end}}$	0

TABLE II

IDM MODEL PARAMETERS AND CLASS DISTRIBUTIONS

	Default/ Agnostic	Cooperative	Adversary
Target velocity v_0	25±10%	-	30±10%
Target headway T	1.0±20%	2.2±20%	0.5±20%
Minimal gap d_0	2.0	-	-
Maximal acc. a	1.5	1.0	2.0
Desired acc. b	2.0	2.0	3.0
Non-reactive P_n	100%	0%	0%
Reactive P_r	33.3%	33.3%	33.3%

the leading vehicle in its own lane (as opposed to headway vis à vis the host agent, which the RL agent learns to abuse).

Actors react to the host agent's trigger stochastically, based on parameters from Table II. More specifically, each actor samples two sets of IDM parameters when it enters a scene: one of which is a *default behavior* and the other its *target behavior*. Actors normally drive according to their default behavior. When the host vehicle is at a lateral offset of at least Δd_{react} from its lane's center towards any lane, the immediate tailing actor within Δs_{react} distance in that lane switches to its target behavior. To capture as much of the complexity of real world interactions as possible, the multi-modal distribution of the target behavior had three classes of actors: cooperative (slows down to increase headway), agnostic (maintains headway) and adversary (accelerates to decrease headway), with added uniform variance around their mean target headways and target velocities.

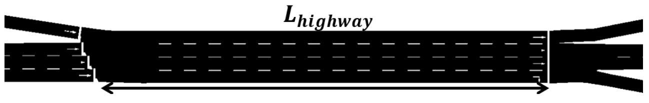
VI. RESULTS

This section presents the results obtained for training policies on benchmarks from the previous section. The performance of the agents was quantified using four measures: *Success Rate*, the percentage of episodes in which the agent achieved its goal successfully; *Episode Duration*, the simulated time it took the agent to reach a terminal state (either successful or not); Discomfort metrics *Sum of Squared Jerk (SSJ)* and *Sum of Squared Acceleration (SSA)* accumulated over the course of an episode, following studies that show their direct effect on passenger discomfort [52], [53]. Each experimental setup was repeated 3 times and its measurements were averaged. Figures also include minimum and maximum values (shaded).

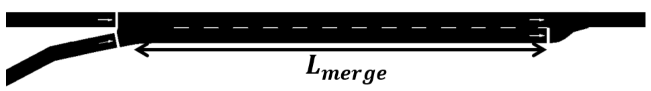
A. The Importance of Abstractions

The first experiment was conducted on the *Highway with Exit* scenario to evaluate the performance of neural architectures, action and state abstractions compared to leading works on RL-based driving policies. For a reliable comparison and since the works chosen for comparison lack the crucial components of the policy structure here (e.g. goal-directing mechanism, reward criteria, verifiable safety layer), the policy structure, reward function design and training algorithm discussed in the previous sections were used across all instances. Since the action abstractions compared to were not created with negotiations in mind, the first experiment used distribution P_n with only agnostic actors.

The state representation implemented the *Relational Grid* method in [22] (which is generalized from [9]) with their fully-connected neural network architecture (referred to as *FC*), while duplicating its last layer for the policy and value heads. The second state representation method, the *Unsorted List* in [6], [8] was implemented with the closest $N_{max} = 30$ actors to host agent, longitudinally. The neural network *PointNet* in [6] was selected, and a larger version of *PointNet+* with double layer sizes, and the *Transformer*-based architecture developed here. Positions, velocities and accelerations of other vehicles were included in all of them. Different combinations of action abstractions were also considered. For the longitudinal dimension, *Constant Acceleration* actions $(-5.5[m/s^2], -2[m/s^2], 0, 2[m/s^2])$ we applied for the duration of a single decision frame as in [8], [22], or *Jerk-Optimal* motion profiles were generated, parameterized by 6 target velocities ($0[m/s]$ to $25[m/s]$) and 3 urgency levels. The longitudinal abstractions were combined with



(a)



(b)

Fig. 3. Scenarios: (a) Highway with Exit and (b) Lane Merge. $L_{highway}$ and L_{merge} control their lengths, respectively.

TABLE III

COMPARATIVE SUMMARY OF THE FINAL RESULTS

Agent	Actors Encoding	Neural Architecture (# Parameters)	Longitudinal Actions	Lateral Actions	Success Rate [%]	Episode Duration $[s]$	SSJ $[m/s^3]^2$	SSA $[m/s^2]^2$
A1	Relational Grid	FC (∼482K)	Constant Acc.	Lane Change	29.44	41.8	-	195.59
A2	Relational Grid	FC (∼482K)	Jerk-Optimal	Lane Change	43.79	**39.2**	**4.52**	**9.24**
A3	Unsorted List	PointNet (∼15K)	Constant Acc.	Lane Change	29.81	41.9	-	189.63
A4	Unsorted List	PointNet+ (∼195K)	Constant Acc.	Lane Change	30.51	42.3	-	188.08
A5	Unsorted List	PointNet (∼15K)	Jerk-Optimal	Lane Change	53.90	44.7	9.69	21.40
A6	Unsorted List	PointNet+ (∼195K)	Jerk-Optimal	Lane Change	60.61	45.9	10.33	24.29
A7	Unsorted List	PointNet (∼15K)	Jerk-Optimal	LFSM	85.93	51.2	8.32	26.51
A8	Unsorted List	PointNet+ (∼195K)	Jerk-Optimal	LFSM	93.20	47.8	13.25	33.71
Ours	Unsorted List	Transformer (∼309K)	Jerk-Optimal	LFSM	**94.11**	44.8	14.17	29.54

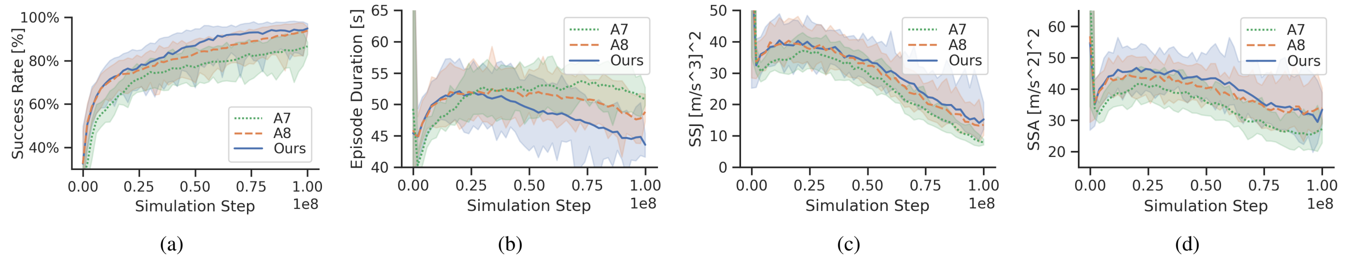


Fig. 4. Comparison of training metrics for different abstractions in the Highway with Exit scenario (top three agents are plotted): (a) success rate (b) episode duration (c) sum of squared jerk (d) sum of squared acceleration.

either *Lane Changes* executed to their entirety at constant velocity as in [8], [22], or the *LFSM* from above.

The results in Table III and Fig. 4 show that despite having the largest neural network model, the agents based on the relations grid representation and the fully-connected architecture were outperformed by agents with smaller models. Moreover, agents implementing a combination of jerk-optimal longitudinal motion with LFSM exhibited better performance, likely due to a higher coverage of motion profiles and usage of lateral offsets to decrease the duration between the lane change decision time and actual cut-in time, both allow an agent to squeeze safely into tighter gaps between vehicles.

B. The Benefits of Negotiation

Whereas previous experiment was conducted with non-reactive actors, the next experiment explored the potential benefits of an agent trained to exploit the reactive nature of other drivers. The performance of this agent was compared on the target distributions P_n and P_r in the Highway with Exit. The evaluation was conducted across 3 different levels of traffic density, controlled by the simulator's *Inflow rate* parameter, a binomial probability of vehicle injection in each frame: "light" (0.3), "medium" (0.4) and "heavy" (0.5). Typical road occupancy in the different traffic setups is shown in Fig. 5. Since the real-world class distribution is unknown, a uniform distribution for P_r was used.

Although results in Fig. 6 show similar agents' performance in light traffic, a significant performance gap can be seen in higher traffic densities, in which the agent in the reactive setup is able to get to the exit faster. In the heavy traffic setup, the agent in the non-reactive setup also

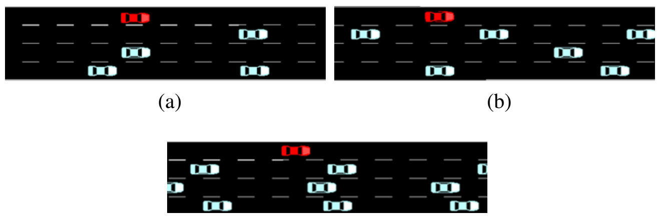


Fig. 5. Snapshots of typical road occupancy for different levels of traffic: (a) low (b) medium (c) heavy

suffers from a low success rate. This suggests that an agent trained for negotiations is superior, especially in scenarios of dense traffic. Finally, the difference in the results also hints at the significant challenge heavy traffic can introduce into decision making, and by extension the importance of evaluating planning agents using heavy traffic benchmarks.

C. Policy Versatility

To demonstrate the versatility of this framework the agent was trained on the Lane Merge scenario in two different setups. In the first setup, highway on-ramp characteristics with values from Table I were used. Full safety layer was enabled. Episodes were set to terminate with a "handover" event once the agent's velocity fell below 10[m/s], to illustrate the flexibility of defining a limit at which a human driver would override the system (by taking control) to drive less conservatively. While an average success of 95% (5% handovers) was observed for low traffic (inflow=0.3), medium traffic (inflow=0.4) and heavy traffic (inflow=0.5) resulted in lower success rates of 92% (8% handovers) and 87% (13%

handovers). This decline in performance in heavier traffic was expected, although it was mainly the outcome of using a worst-case safety model logic for yielding to traffic in high-velocity scenarios, due to the squared relationship of braking distance to driving velocity.

A different design choice for an L2 ADAS application with a driver-in-the-loop for high speeds might disable the safety layer at for yielding to merge and rely on the human driver as a fallback for safety (while still enabling safety verification for early lane changes). In these circumstances, it is still important for the system to maximize the success rate and avoid driver interventions. In another series of experiments, different values of the handover reward coefficient c_h were tested with the safety logic disabled for yielding to merge. The findings showed that as c_h increased, agents tended to have recourse to handovers more often, which significantly lowered the failure rate. For instance, in heavy traffic with $c_h = 0.2$ the agent rarely made any handovers (<0.01%) with a success rate of˜98.3% and a failure rate of˜1.7%. Conversely, setting $c_h = 0.8$ reduces the failure rate by more than 40% to 1%, but with 97.8% success and 1.2% handovers. These results for the different setups of safety logic and c_h values are indicative of the flexibility of this approach and its applicability to different levels of automation.

Finally, an agent was trained on the Lane Merge scenario in an urban setting where velocities were reduced by 10[m/s], with perturbations to merging lane length L_{merge} in the range 0-100$[m]$ and where the host agent was allowed to come to a full stop. For brevity, a qualitative demonstration of the trained agent on a *totally unfamiliar scenario of a single-lane roundabout* is presented in Fig. 8. This was made possible by selecting a state encoding method based on the road's coordinate frame. Although a top-down view of this scenario looks different than other scenarios, it exhibits similar attributes to the Lane Merge scenario with merging lane length set to 0$[m]$.

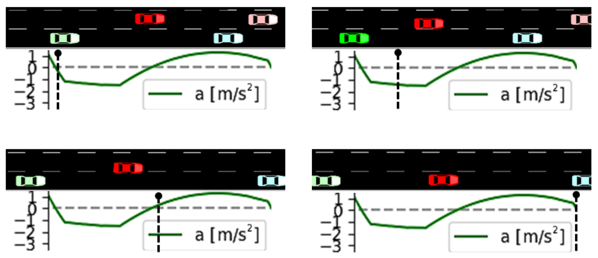


Fig. 7. Top-down view from the simulation of the agent negotiating to blend into traffic, with the longitudinal acceleration profile underneath the figures.

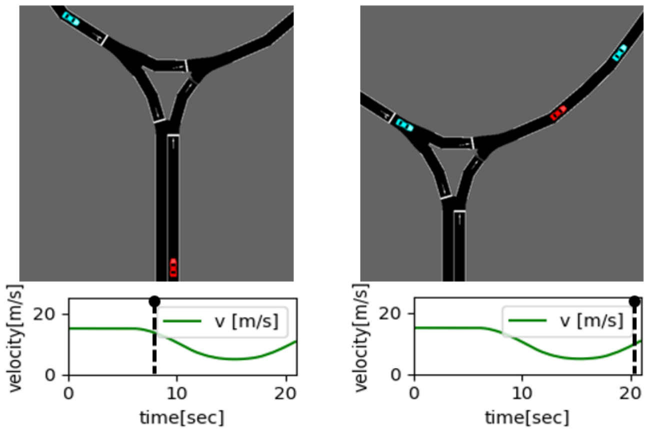


Fig. 8. Top-down view from simulation of an unfamiliar roundabout, with the longitudinal velocity profile underneath the figures

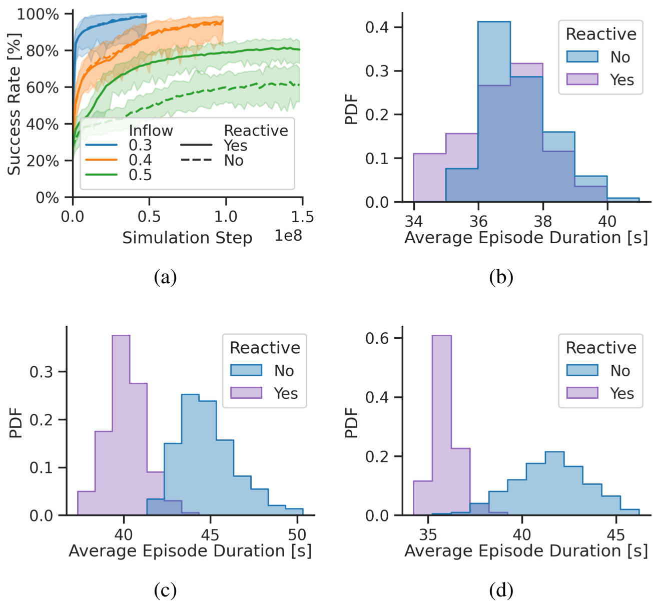


Fig. 6. Agent performance on training with reactive vs. non-reactive target behavior: (a) success rate for all traffic levels, and episode duration histograms for (b) light traffic (c) medium traffic (d) heavy traffic

VII. Conclusions

This work presented benchmarks for stochastic negotiations between drivers with different levels of cooperation. It also proposed a framework based on reinforcement learning and novel abstractions for training driving policies to negotiate. The findings confirm that policies trained to negotiate are able to perform better in traffic. The selection of abstractions can impact the performance and generalization capabilities of a driving policy, and policies trained with this new framework are able to demonstrate versatile behaviors.

Future work will focus on achieving a smooth transfer to real world applications. Using real world data for training more accurate driver models or enhancing policies directly with offline-RL could potentially improve the policy applicability to the real world. Finally, training a single policy to generalize across scenarios will require an efficient representation of map structure and attributes.

Acknowledgment

This work was fully funded by General Motors. We wish to thank Gabi Guetta and Vlad Goldner for helpful brainstorming sessions.

References

[1] Waymo, "Waymo merging into traffic," https://streamable.com/atjakr, Waymo, 2018.

[2] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," 2018.

[3] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," 2016.

[4] F. Codevilla, M. Müller, *et al.*, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 4693–4700.

[5] F. Codevilla, E. Santana, *et al.*, "Exploring the limitations of behavior cloning for autonomous driving," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9328–9337.

[6] C.-J. Hoel, K. Driggs-Campbell, *et al.*, "Combining planning and deep reinforcement learning in tactical decision making for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, p. 294–305, Jun 2020.

[7] P. Kapoor, A. Balakrishnan, and J. V. Deshmukh, "Model-based reinforcement learning from signal temporal logic specifications," 2020.

[8] C.-J. Hoel, K. Wolff, and L. Laine, "Automated speed and lane change decision making using deep reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2148–2155.

[9] B. Mirchevska, C. Pek, *et al.*, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2156–2162.

[10] L. Chi and Y. Mu, "Deep steering: Learning end-to-end driving model from spatial and temporal visual cues," 2017.

[11] W. Zeng, W. Luo, *et al.*, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[12] Y. Xiao, F. Codevilla, *et al.*, "Multimodal end-to-end autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, p. 1–11, 2020.

[13] M. Müller, A. Dosovitskiy, *et al.*, "Driving policy transfer via modularity and abstraction," 2018.

[14] J. Hawke, R. Shen, *et al.*, "Urban driving with conditional imitation learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 251–257.

[15] N. Djuric, V. Radosavljevic, *et al.*, "Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving," 2020.

[16] F.-C. Chou, T.-H. Lin, *et al.*, "Predicting motion of vulnerable road users using high-definition maps and efficient convnets," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 1655–1662.

[17] C. Chen, A. Seff, *et al.*, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2722–2730.

[18] A. Mehta, A. Subramanian, and A. Subramanian, "Learning end-to-end autonomous driving using guided auxiliary supervision," 2018.

[19] N. Rhinehart, R. McAllister, *et al.*, "Precog: Prediction conditioned on goals in visual multi-agent settings," 2019.

[20] S. Casas, C. Gulino, *et al.*, "Spatially-aware graph neural networks for relational behavior forecasting from sensor data," 2019.

[21] J. Mercat, T. Gilles, *et al.*, "Multi-head attention for multi-modal joint vehicle motion forecasting," 2019.

[22] P. Wolf, K. Kurzer, *et al.*, "Adaptive behavior generation for autonomous driving using deep reinforcement learning with compact semantic states," *2018 IEEE Intelligent Vehicles Symposium (IV)*, Jun 2018.

[23] K. Makantasis, M. Kontorinaki, and I. Nikolos, "Deep reinforcement-learning-based driving policy for autonomous road vehicles," *IET Intelligent Transport Systems*, vol. 14, no. 1, pp. 13–24, 2020.

[24] G. Wang, J. Hu, *et al.*, "Cooperative lane changing via deep reinforcement learning," 2019.

[25] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," 2018.

[26] A. Vaswani, N. Shazeer, *et al.*, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[27] K. He, X. Zhang, *et al.*, "Deep residual learning for image recognition," 2015.

[28] V. Mnih, A. P. Badia, *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1928–1937.

[29] J. Schulman, S. Levine, *et al.*, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015.

[30] T. P. Lillicrap, J. J. Hunt, *et al.*, "Continuous control with deep reinforcement learning," 2019.

[31] Y. Duan, X. Chen, *et al.*, "Benchmarking deep reinforcement learning for continuous control," in *International conference on machine learning*. PMLR, 2016, pp. 1329–1338.

[32] H. Xu, Y. Gao, *et al.*, "End-to-end learning of driving models from large-scale video datasets," 2017.

[33] Á. Fehér, S. Aradi, *et al.*, "Hybrid ddpg approach for vehicle motion planning," in *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics - Volume 1: ICINCO,*, INSTICC. SciTePress, 2019, pp. 422–429.

[34] M. Bouton, A. Nakhaei, *et al.*, "Cooperation-aware reinforcement learning for merging in dense traffic," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3441–3447.

[35] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," 2020.

[36] M. Werling, J. Ziegler, *et al.*, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 987–993.

[37] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end autonomous driving," 2016.

[38] D. Lenz, T. Kessler, and A. Knoll, "Tactical cooperative planning for autonomous highway driving using monte-carlo tree search," in *2016 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2016, pp. 447–453.

[39] P. Wang and C.-Y. Chan, "Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge," *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017.

[40] H. Zhao, Y. Zhang, *et al.*, "Towards safety-aware computing system design in autonomous vehicles," 2019.

[41] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," 2018.

[42] I. T. Committee *et al.*, "Convention on road traffic," *Economic Commission for Europe, Vienna, Austria*, 1968.

[43] B. Paden, M. Cap, *et al.*, "A survey of motion planning and control techniques for self-driving urban vehicles," 2016.

[44] S. Zhang, W. Deng, *et al.*, "Dynamic trajectory planning for vehicle autonomous driving," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*, 2013, pp. 4161–4166.

[45] J. Liu, W. Zeng, *et al.*, "Deep structured reactive planning," *CoRR*, vol. abs/2101.06832, 2021. [Online]. Available: https://arxiv.org/abs/2101.06832

[46] A. W. Drake, "Observation of a markov process through a noisy channel," Ph.D. dissertation, Massachusetts Institute of Technology, 1962.

[47] P. A. Lopez, M. Behrisch, *et al.*, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, November 2018.

[48] C. Wu, A. Kreidieh, *et al.*, "Flow: A modular learning framework for autonomy in traffic," 2020.

[49] P. Moritz, R. Nishihara, *et al.*, "Ray: A distributed framework for emerging ai applications," 2018.

[50] A. Paszke, S. Gross, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[51] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, p. 1805–1824, Aug 2000. [Online]. Available: http://dx.doi.org/10.1103/PhysRevE.62.1805

[52] J. Forstberg, "Ride comfort and motion sickness in tilting trains," Ph.D. dissertation, Royal Institute of Technology (KTH), 2000.

[53] L. Svensson and J. Eriksson, "Tuning for ride quality in autonomous vehicle: Application to linear quadratic path planning algorithm," Ph.D. dissertation, UPPSALA University, 2015. [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-257387