Graph-Based Autonomous Driving with Traffic-Rule-Enhanced Curriculum Learning

Lars Frederik Peiss, Elias Wohlgemuth, Fan Xue, Eivind Meyer, Luis Gressenbuch, and Matthias Althoff

***Abstract*— Training reinforcement learning (RL) agents for motion planning in heavily constrained solution spaces may require extensive exploration, leading to long training times. In automated driving, RL agents have to learn multiple skills at once, such as collision avoidance, traffic rule adherence, and goal reaching. In this work, we decompose this complicated learning task by applying curriculum learning for the first time onto an RL agent based on graph neural networks. The curriculum's sequence of sub-tasks gradually increases the difficulty of the longitudinal and lateral motion planning problem for the agent. Each of our sub-tasks contains a set of rewards, including novel rewards for temporal-logic-based traffic rules for speed, safety distance, and braking. Unlike prior work, the agent's state is extended by map and traffic rule information. Its performance is evaluated on prerecorded, real-world traffic data instead of simulations. Our numerical results show that the multi-stage curricula let the agent learn goal-seeking highway driving faster than in baseline setups trained from scratch. Including traffic rule information in both the RL state and rewards stabilizes the training and improves the agent's final goal-reaching performance.**

I. Introduction

Motion planning algorithms for autonomous driving face multiple constraints that restrict their solution space. These include the need to find collision-free trajectories while adhering to further traffic rules and optimizing driving performance. One commonly researched method of data-driven motion planning for autonomous vehicles is reinforcement learning (RL) [1]. RL algorithms train agents to maximize rewards obtained from interacting with an environment over time [2], such as driving through traffic. If RL agents are faced with a difficult task from the beginning of their training, e.g., due to the many constraints in motion planning, they may need a long time to converge [3], if at all. Curriculum learning (CL) can be used to ease the learning process of difficult tasks. CL schemes subdivide the training process of machine learning algorithms into a sequence of sub-tasks of increasing difficulty [4]. It has been shown that CL simplifies and accelerates various machine learning problems [3], including RL for autonomous driving [5]–[8].

As visualized in Fig. 1, RL relies on a state representation of the environment that the agent processes to derive a reward-optimal policy [2]. By using graphs as state representations for RL agents, arbitrary road topologies and numbers of surrounding vehicles can be flexibly captured [9], [10]. Graph neural networks (GNNs) have successfully been used to encode the graph-based traffic input into states for the RL agent [9]–[11] (GNN-RL). GNNs are invariant to the order and number of vehicles (nodes) in an environment and model relational information between the vehicles explicitly in their edges [9]. Hence, this vehicle-to-vehicle (V2V) relationship does not have to be learned [9].

Department of Computer Engineering, Technical University of Munich, Garching, Germany. {lf.peiss, elias.wohlgemuth, fan98.xue, eivind.meyer, luis.gressenbuch, althoff}@tum.de

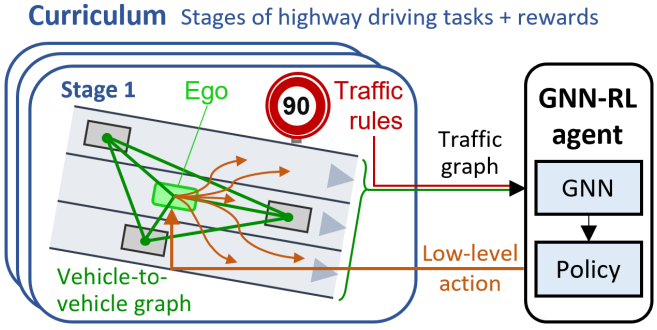


Fig. 1: High-level overview of our graph neural network (GNN)-based reinforcement learning problem to learn a curriculum of driving tasks that incorporate traffic rules.

A. Related work

As our work combines GNN-RL with CL including traffic rule evaluations, we summarize related works applied to motion planning for autonomous driving in these three areas.

1) Graph Neural Network Reinforcement Learning: In autonomous driving, several works have employed GNN-based RL motion planners [9]–[23]. A relationally interpretable GNN input representation is the V2V graph that connects the road users around the agent [9], as shown in Fig. 1. The effectiveness of V2V-GNN-based states for RL agents over non-GNN ones has been shown for a simplified highway lane change scenario in dense simulated traffic [9]. Here, the V2V-GNN contains dynamic properties in the vehicle nodes and V2V distances in the edges. It outputs an ego node embedding to serve as the state of an RL agent that controls the ego vehicle acceleration and steering angle.

Similar V2V-GNN performance advantages as in [9] are reported for high-level lane changing or keeping commands on simulated highway scenarios, except for low traffic densities with few V2V interactions [11]. In line with this, prior work has not yet evaluated the performance of V2V-GNNs for very low traffic densities. Such traffic scenes without any vehicle around the agent yield empty V2V graphs. The largely map-unaware agents cannot orient within this non-representative input as they lack explicit lane geometry features in their states. Furthermore, prior work has only evaluated GNN-RL agents in unrealistic simulation setups

where the simulated other vehicles change lanes cooperatively, if at all, and always brake when the agent cuts in.

2) RL Curriculum Learning: CL in RL has been applied to several autonomous driving tasks [5]–[8], [24]–[26]. All works rely only on a few manually specified curriculum stages with respective task and reward specifications for RL agents, while GNN-RL is not yet employed. Several works set the initial CL stage as an empty road without any vehicles beyond the agent [5], [6], [8], [24].

In [24] and [8], a driving policy is trained in urban simulations with multimodal features. RGB images and different feature vectors related to vehicles and roads are processed by specific network layers and concatenated into the agent's state vector [8]. After an empty initial CL stage, the number of dynamic obstacles (vehicles, pedestrians) and spawning locations for the agent/obstacles is increased over five stages, while the weather-based image quality is degraded [8]. The CL-trained agent performs comparably to a non-CL agent w.r.t. collision rate (which remains high) and total episode reward over tested scenarios [8].

In [5], CL is used to learn overtaking other vehicles in a racing simulation. The agent first learns to drive on empty tracks, before the second stage adds other vehicles and an overtaking reward, and finally the third stage adds increasing collision penalties [5]. While CL increases the rewards significantly to human-level driving performance, the work does not compare the collision rate against a non-CL baseline. In [7], the overtaking task is approached with CL for simulated highway driving. The state vector of hand-crafted vehicular and road features is directly fed into the agent [7]. Similar to [5], the agent is first tasked with lane keeping among other vehicles, before the second stage adds an overtaking reward [7]. The CL-based agent learns overtaking with fewer collisions and faster than a non-CL baseline, although a non-CL ablation study is shown to also improve overtaking by only adding a lane keeping reward [7].

3) Traffic Rules: Teaching an RL agent to adhere to traffic rules requires incorporating them during or after the learning process. Following simple traffic rules, such as the speed limit, can be easily achieved by masking violating actions [27] or applying penalties to the reward signal [28], although the latter is not provably safe. More complex rules are formalized using temporal logic formulas [29]. The results of linear temporal logic specifications have been integrated into RL agents as rewards for grid world scenarios [30], as selectors of rule-compliant policies for lane changing scenarios [31], [32], or as violation rewards [31]. However, the degrees of traffic rule satisfaction or violation have not yet been incorporated into the state or rewards of GNN-RL agents nor into CL schemes for autonomous driving.

B. Contributions

This work uses a CL scheme to train a GNN-RL agent in safe highway driving by incorporating lane-based spatial information and traffic rules into a V2V-GNN. Our manually specified curricula create sequences of increasingly difficult highway driving tasks with appropriately shaped rewards. These guide the RL agent towards desirable driving behavior, including traffic rule adherence. Our contribution is four-fold:

- We extend V2V graph-based GNN-RL agents from prior work with road geometry-related features and rewards to make them applicable to low traffic densities;
- to further improve driving, we integrate for the first time satisfaction degrees of temporal logic-based traffic rules into the GNN-RL agent's features and rewards;
- we specify two novel GNN-RL-based curricula with and without traffic rules to learn highway driving on progressively increasing traffic densities; and
- unlike prior work on simulations, we evaluate our GNN-RL agent and curricula against baselines on prerecorded real-world traffic requiring reactive driving styles.

To the best of our knowledge, we are the first to propose a CL scheme for motion planning that integrates traffic rules and map-awareness for GNN-RL agents. Our implementation is based on CommonRoad-Geometric [33], which offers customizable graph representations of traffic scenes, and is available online at http://github.com/CommonRoad/crgeo-cl.

II. Preliminaries

A. Graph Neural Networks (GNNs)

We use GNNs as feature extractors to output latent state vectors for RL policies. GNNs transfer the notion of common deep neural networks into the graph domain [34]. In this work, we convert traffic information about a vehicle v into directed graphs $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathbf{h}_v \in \mathcal{X}_\mathcal{V}$ represents the node features of each v and $\mathbf{e}_{vw} \in \mathcal{X}_\mathcal{E}$ denotes the edge features of the V2V relation between two vehicles v and w. Multi-layer GNNs are able to share and aggregate feature vectors $(\mathcal{X}_\mathcal{V}, \mathcal{X}_\mathcal{E})$ in a local graph neighborhood using the message passing paradigm defined by the learned differentiable message functions M_k and node update functions U_k [35]. During the message passing at each layer $k \leq K$, features $\mathbf{h}_v^k$ of each node are updated to $\mathbf{h}_v^{k+1} = U_k(\mathbf{h}_v^k, \mathbf{m}_v^{k+1})$ based on the messages $\mathbf{m}_v^{k+1}$, which are aggregated by [35]

$$\mathbf{m}_v^{k+1} = \sum_{w \in \mathcal{N}(v)} M_k(\mathbf{h}_v^k, \mathbf{h}_w^k, \mathbf{e}_{vw}^k), \tag{1}$$

where $\mathcal{N}(v)$ represents the neighbors of v in graph $\mathcal{G}$ [35]. By running multiple message passing steps in several GNN layers, information is distributed among increasingly large parts of the graph [36]. The resulting graph can be converted into a latent feature vector $\mathbf{z}$ to serve as the RL state.

B. Reinforcement Learning (RL)

RL maximizes the expected reward of Markov Decision Processes (MDPs) without knowledge of the environment by learning from experience [2]. An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, R, P, \gamma)$ where $s_t \in \mathcal{S}$ is the state at time t, $a_t \in \mathcal{A}$ is the action, r_t is the immediate reward at t based on a reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, $P : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ returns transition probabilities, and γ is the discount factor [2]. One way to solve MDPs with RL is the actor-critic method, where two different networks are implemented for learning the policy $\pi(s_t)$ and the state-value function $V(s_t)$, respectively [2].

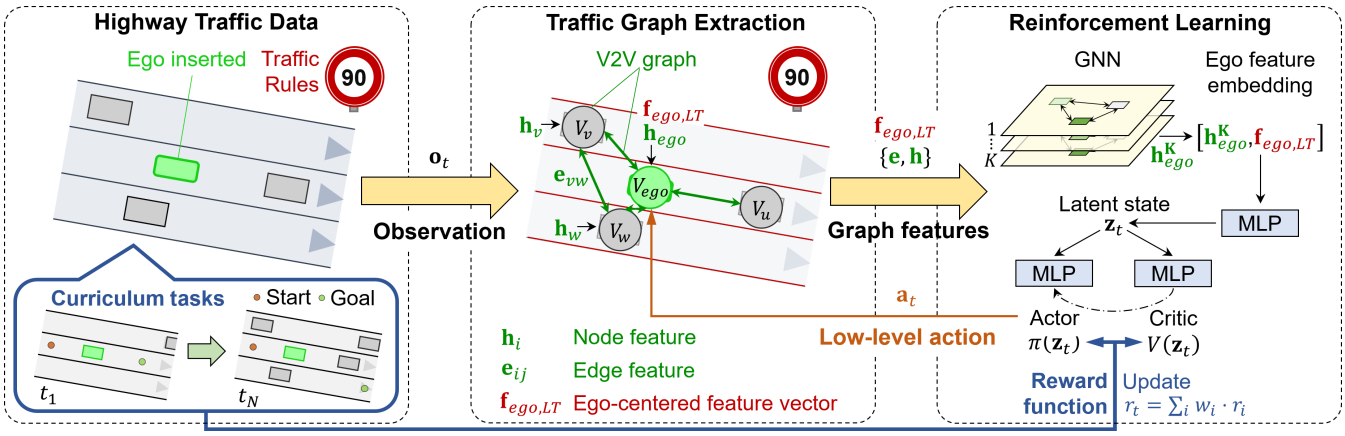


Fig. 2: Structure of our GNN-RL-based CL approach. The curriculum selects tasks on real-world highway data and reward functions r_t per stage. From the data, a traffic graph is constructed whose node and edge features are processed by GNN layers. The ego node embedding $\mathbf{h}^K_{ego}$ is concatenated with the ego feature vector $\mathbf{f}_{ego,LT}$ and embedded into the latent state $\mathbf{z}_t$ for the actor-critic RL agent.

C. Curriculum Learning (CL)

CL refers to a technique that accelerates an agent's training or enhances its performance by optimizing the sequence of tasks during learning [3]. A task-level sequence curriculum orders tasks $t_i = (\mathcal{S}_i, \mathcal{A}_i, R_i, P_i, \gamma_i)$ as $[t_1, \ldots, t_N]$, so the RL agent is trained first on samples from a simple task t_1 and last on the most difficult target task samples t_N [3], e.g., complex highway scenarios. The task difficulty is often defined manually by human experts [3]. Each curriculum specifies by stopping criteria whether a task t_i is ended after sufficient convergence or after a fixed episode count e_i, before t_{i+1} is started [3]. In this work, we use the latter and refer to the tuple $S_i = (t_i, e_i)$ as a stage.

D. Traffic Rules

The satisfaction of traffic rules can be defined by the robustness degree of signal temporal logic (STL) [37]. STL is a logic for specifying properties of signals in continuous time [37]. Out of the large number of traffic rules in different legal systems, we formalize three German traffic rules on highways according to [29]: The safety distance of v to the preceding vehicle w considering vehicles that cut in (G1), avoiding unnecessary braking of v, unless the safety distance to preceding vehicle w is violated (G2), and adhering to the maximum speed limit (G3). The robustness degrees of these rules in STL are based on several predicates (*CutIn*, *InFrontOf*, *InSameLane*, *KeepsSafeDistPrec*, *Precedes*, *RelBrakesAbruptly*) covering all relevant prior timesteps of a scenario [29], [38]. Some predicates, such as *InFrontOf*, directly evaluate lane-based position relations between vehicles, while others also factor in their current relative speeds, such as *KeepsSafeDistPrec* [29]. We adopt the scaling scheme from [38] that bounds the scalar robustness degrees to $[-1, 1]$, where a negative (positive) value represents the degree of a fulfilled logical false (true) value.

III. Approach

Our goal is to train an RL agent in safely driving through highway environments to reach given goal positions. To that end, we process a V2V graph with a multi-layer GNN and extend the ego node embedding with lane and traffic rule information. The processed output serves as the latent state $\mathbf{z}_t$ of our agent that commands the steering angle and acceleration. To simplify the training of our GNN-RL agent, we set up task-level sequence curricula with task-specific rewards, which also evaluate the influence of traffic rules on driving performance.

A. Graph Neural Networks (GNNs)

Due to a lack of lane geometry features, V2V-GNNs from prior work cannot be used as states for RL agents at low traffic densities, as these models can only infer the drivable area from a joint motion of many vehicles around the agent. Since our CL approach is based on challenging real-world highway traffic data, a lowered traffic density through sparse vehicle sampling is key to building simpler initial tasks for the agent. To make V2V-GNNs also applicable to low traffic densities, we design an ego-centered feature vector $\mathbf{f}_{ego,LT}$ with lane and road geometry information that is concatenated to the ego node embedding $\mathbf{h}^K_{ego}$. The latter is extracted from the GNN after V2V message passing as in [9]. Next, a multi-layer perceptron (MLP) embeds $[\mathbf{h}^K_{ego}, \mathbf{f}_{ego,LT}]$ to obtain the latent state $\mathbf{z}_t$ for the RL agent, as shown in Fig. 2. The features used in $\mathbf{f}_{ego,LT}$ are specified along with corresponding scaling functions applied for normalizing the feature domain and $\mathbf{h}^k_v$ in Tab. I.

The combination of a traditional fixed-size feature vector combined with a V2V-GNN embedding enables the agent to directly extract navigation cues from $\mathbf{f}_{ego,LT}$ at low traffic densities while also integrating V2V interactions from $\mathbf{h}^K_{ego}$ at higher traffic densities. We opt for this simple V2V-based architecture since the drivable space on highways can be modeled as a homogeneous 2D surface on which vehicles can change lanes freely. As the agent can directly infer its feasible actions from V2V interactions and/or $\mathbf{f}_{ego,LT}$, more complex representations are not needed for learning safe highway driving, such as heterogeneous vehicle-to-lane-graph-based GNNs [10]. To speed up the learning process

TABLE I: Scaled node, edge, and vector features of our V2V-GNN. "xy" denotes 2D features, unlike the other scalar features. $^{\text{L}}$ features used both in the V2V$^{\text{L}}$ and V2V$^{\text{LT}}$ GNNs. $^{\text{T}}$ features used only in the V2V$^{\text{LT}}$ GNN. * features taken from [9].

Feature	Feature name	Scaling function
General vehicle node features $\boldsymbol{h}_v^k$		
$\boldsymbol{f}_{p,xy}$*	*PosEgoFrame*	$f_{p,x/y}$ / 50
$\boldsymbol{f}_{v,xy}$*	*Velocity*	($f_{v,x/y}$ − 15) / 20
Vehicle-to-vehicle edge features $\boldsymbol{e}_{vw}^k$		
$\boldsymbol{f}_{rp,xy}$*	*RelativePosEgo*	$f_{rp,x/y}$ / 50
Additional ego vehicle feature vector $\boldsymbol{f}_{ego,LT}$		
$\boldsymbol{f}_{v,xy}$	*Velocity*	($f_{v,x/y}$ − 15) / 20
$\boldsymbol{f}_{a,xy}$	*Acceleration*	$f_{a,x/y}$ / 20
f_y	*YawRate*	$min(max(f_y, -1), 1)$
$f_{h,L}$	*HeadingError*	$min(max(f_{h,L}, -\pi/4), \pi/4)$
$f_{lb,L}$	*DistToLeftLaneBoundary*	$f_{lb,L}$ / 2
$f_{rb,L}$	*DistToRightLaneBoundary*	$f_{rb,L}$ / 2
$f_{lrb,L}$	*DistToLeftRoadBoundary*	($f_{lrb,L}$ + $f_{lb,L}$) / 12
$f_{rrb,L}$	*DistToRightRoadBoundary*	($f_{rrb,L}$ + $f_{rb,L}$) / 12
$f_{la,L}$	*DistToGoalLateral*	$log(\lvert f_{la,L}\rvert + 1)\, f_{la,L} \,/\, \lvert f_{la,L}\rvert$
$f_{lo,L}$	*DistToGoalLongitudinal*	$log(\lvert f_{lo,L}\rvert + 1)\, f_{lo,L} \,/\, \lvert f_{lo,L}\rvert$
$f_{g1,T}$	*G1Robustness*	$f_{g1,LT}$
$f_{g2,T}$	*G2Robustness*	$f_{g2,LT}$
$f_{g3,T}$	*G3Robustness*	$f_{g3,LT}$

TABLE II: Reward components r_i weighted by w_i and summed up over the episode to the agent's total reward r. $^{\text{L}}$ rewards used both in the V2V$^{\text{L}}$ and V2V$^{\text{LT}}$ GNNs. $^{\text{T}}$ rewards used only in the V2V$^{\text{LT}}$ GNN. * reward types taken from [9]. † termination criteria.

Reward	Name	Value r_i	Trigger
Vehicle-pose-based components			
r_{reGo} * †	*ReachedGoal*	4	`is_at_goal()`
r_{coll} * †	*Collision*	−4	`is_colliding()`
r_{road}†	*Offroad*	−4	`is_offroad()`
$r_{route,L}$	*Offroute*	−1	`is_offroute()`
$r_{offLC,L}$	*OffLaneCenter*	$max(-o_t, -0.05)$	–
$r_{hdErr,L}$	*HeadingError*	$-(\cdot)^2$	–
Vehicle-dynamics-based components			
r_{accel}*	*Acceleration*	$-(\cdot)^2$	–
r_{sAngl}*	*SteeringAngle*	$-\lVert\cdot\rVert_1$	–
r_{velo}*	*Velocity*	$-\left\lVert\lVert\dot{\mathbf{p}}_v\rVert_2 - 50\right\rVert_1$	$\lVert\dot{\mathbf{p}}_v\rVert_2 > 50$
r_{stand}	*StillStanding*	−0.01	$\lVert\dot{\mathbf{p}}_v\rVert_2 < 2$
$r_{ttc,a}$	*TimeToCollision*	$-e^{-min(t_{TC,t})/2}$	–
$r_{ttc,b}$	*TTC-Closeness*	$-max(c_t, 0)/0.08$	$min(\lVert\Delta\mathbf{p}_{vw}\rVert_2) < 4$
$r_{trajP,L}$	*TrajectoryProgr*	$min(\Delta s_v^L, 0.2)$	$\Delta s_v^L > 0$
$r_{lnCh,L}$	*LaneChange*	−2	$L_{v,t} \neq L_{v,t-1}$
Traffic-rule-based components			
$r_{g1,T}$	*G1Robustness*	$f_{g1,T}$	–
$r_{g2,T}$	*G2Robustness*	$f_{g2,T}$	–
$r_{g3,T}$	*G3Robustness*	$f_{g3,T}$	–

of safe highway driving, we integrate for the first time robustness values $f_{g1\text{-}3,T}$ of the three traffic rules G1-G3 into $\mathbf{f}_{ego,LT}$ for the GNN-RL agent. This creates a V2V$^{\text{LT}}$ GNN instead of the only lane feature-extended V2V$^{\text{L}}$ GNN. Based on [29] and [38], we compute $f_{g1\text{-}3,T} \in [-1, 1]$ for the ego vehicle in relation to surrounding vehicles at each time step t of a scenario. Unlike the prior work on a-priori-known vehicle trajectories, our rule robustness calculation needs to update at each t to reflect the dynamic actions of the agent. Due to the computational complexity of robustness calculations, we restrict them to the most relevant agent-to-vehicle-relations and do not consider pairs of other vehicles.

B. Reinforcement Learning (RL)

Key to learning safe highway driving with RL is the aggregated reward function R of carefully weighted rewards r_i. We apply the r_i defined in Tab. II and explain their contribution to aspects of safe driving subsequently. For further details, we refer to our online available implementation. Rewards denoted by † act as termination criteria, providing a single penalty when triggered and terminating the episode.

As a core safety aspect, the rewards for *Collision*, *TimeToCollision*, safety-distance-based *G1Robustness*, and braking-based *G2Robustness* incentivize collision avoidance. While *Collision* only penalizes the actual collision, *TimeToCollision* helps the agent learn to anticipate collisions, i.e., to predict when its trajectory intersects with others. The time-to-collision metric t_{TC} represents the time after which two vehicles would collide if they keep moving on their current path and with their current velocity [39]. As a proxy to the time-to-collision penalty $-e^{-\min(t_{TC,t})/2}$, we also explore the scaled inverse distance of the agent to the closest surrounding vehicle $c_t = 1/\min(\lVert\Delta\mathbf{p}_{vw}\rVert_2) - 0.92$, which is similar to distance potential functions [18]. The scaling in c_t ensures to only consider $\lVert\Delta\mathbf{p}_{vw}\rVert_2 < 4\,\text{m}$ in the reward, which avoids penalizing the agent for passing by other vehicles on the left or right lane, as it could discourage it from moving forward in dense traffic. The rewards of traffic rules G1 and G2 are calculated based on the respective GNN features available to the agent as direct feedback about rule satisfaction. The agent is rewarded with up to $+1$ for fulfilling rules with different degrees ($f_{g1\text{-}3,LT} > 0$) and penalized with up to -1 for violating them ($f_{g1\text{-}3,LT} < 0$).

The safety aspect of driving close to the lane center is covered by the rewards for *Offroute*, *Offroad*, *HeadingError*, and *OffLaneCenter*. *Offroute* returns a penalty per timestep if the agent is not on the direct route from the start to the goal lane. *Offroad* returns a termination penalty for leaving the road. *HeadingError* penalizes the agent's deviation from the orientation of the traversed lanelet and *OffLaneCenter* the weighted offset from the closest lane centerline $o_t = w_{offLC,t}\,\lVert f_{lb,L} - f_{rb,L}\rVert_1$. The latter also incentivizes the agent to avoid unnecessary lane changes, which is further reinforced by the *LaneChange* penalty given for each change from the ego lane L_v. Safely adhering to speed limits is fostered by penalizing exceeding a prespecified *Velocity* of $50\,\text{m/s}$, *StillStanding* (defined with a safety margin as below $2\,\text{m/s}$), and violating the traffic rule-based *G3Robustness* value using the official speed limit of the traversed map segment. Similar to [9], the agent's normalized control commands are squared in *Acceleration* and *SteeringAngle* to penalize shaky RL control outputs, as observed by others [18], [19]. This behavior is strengthened by *G2Robustness*

penalizing unnecessary braking maneuvers.

Lastly, the generally preferred goal-seeking behavior is introduced by the rewards of *ReachedGoal*, *StillStanding*, and *TrajectoryProgr*. The latter encourages forward motion through rewards per timestep based on the traveled distance Δs_v^L [33]. It is similar to the goal distance reward of [9], but also gives rewards when the agent has missed the goal laterally and continues driving along the highway. As in [9], we apply the rewards to an actor-critic RL agent based on PPO [40].

C. Curriculum Learning (CL)

Our task-level sequence curriculum aims to make the agent learn multiple driving skills. These are required for the driving tasks of increasing difficulty with correspondingly weighted, stage-specific rewards, optionally including traffic rules. Since our training data contains fixed prerecorded trajectories, the task difficulty can mainly be adjusted by manually reducing the vehicle density through undersampling, which acts as an upper bound of how many vehicles are kept depending on traffic.

Furthermore, the RL driving difficulty in real-world data can be controlled by selecting the start and goal positions. Unlike in works with uniformly simulated traffic over all lanes, such as [9], real-world German highway traffic exhibits different vehicle distributions of cars and trucks over the lanes, which causes density and velocity gradients from the left to the right lane [41]. As a result, our lateral choice of varying start and goal lanes not only forces the agent to change lanes, but also to deal with a specified degree of density and velocity-based driving difficulties. Lastly, longer start-to-goal distances can complicate the driving task.

At both the undersampling and start-goal position choice, the curriculum has to ensure that V2V message passing and the ego-centered feature vector can provide useful learning signals to the RL agent. For instance, sufficiently many vehicles should be kept so that the V2V graph does not collapse, and the episode count e_i should be manually limited to a value that enforces the agent to learn the task of a stage sufficiently, but not to overfit. For the final evaluation, the CL-based RL agent with traffic rules `T` and without them `C` is to be compared against a corresponding non-CL baseline `B` on the target task t_N. During training, however, a reward-based performance evaluation is difficult, as the reward function changes per stage. Inspired by the CommonRoad cost function [42], we therefore set up a stage-independent performance evaluation function p_t. It judges the overall quality of the current driving skills throughout all stages by a constant set of rewards hidden to the RL agent.

IV. Experiments

We evaluate our GNN-RL-based motion planner on real-world highway traffic data through two different curricula and investigate the effect of including traffic rule information.

A. Setup

1) Data: We train and evaluate our approach on the highD dataset of highway driving recorded in Germany [43]. The dataset contains two-lane or three-lane, unidirectional main carriageways. 19366 scenarios are extracted at a 90%/10% train/test split, where all initial and goal poses correspond to real-world trajectories of cars and trucks. Different from many GNN-RL works [9], [11], [14], [17], [23], the highD traffic is replayed open-loop in sequences of 150 steps ($\Delta t = 0.04\,s$). Unlike in closed-loop simulations, surrounding vehicles perform neither emergency braking for the agent cutting in nor cooperative lane change maneuvers. The agent has thus to learn to make space for other vehicles to avoid collisions.

TABLE III: Major hyperparameters of the V2V$^{\text{L}}$ GNN-RL setup

Ego vehicle start state (l_s, v_s)	middle lane, $30\,\mathrm{m/s}$
V2V graph construction	3-nearest neighbor $\forall \|\Delta \mathbf{p}_{vw}\|_2 < 50$
GNN message passing layers (K)	3
Aggregation, activation functions	$\max(\cdot)$, $\tanh(\cdot)$
Feature dimensions ($\|\mathbf{h}_v^{k>0}\|$, $\|\mathbf{z}_t\|$)	80, 80
GNN ego node-feature embedder	$MLP(\|\mathbf{h}_{ego}^3\| + \|\mathbf{f}_{ego,LT}\|, \|\mathbf{z}_t\|)$
RL Actor-critic networks (π, V)	Each $MLP(256, 128, 64)$
Rollout steps, batch size	256, 32
Discount factor (γ)	0.99

2) GNN-RL: The traffic graph extraction and GNN processing are orchestrated by CommonRoad-Geometric [33]. For computational speedup, we construct the graph with the three nearest neighbors of the ego vehicle for the GNN and traffic rule calculation. For training our GNN-RL agent, we use the PPO implementation of SB3 [44] with the Adam optimizer at a learning rate of 10^{-5} with a weight decay of 10^{-3}. Details are provided in Tab. III, and for the curricula and the reward functions subsequently.

3) Curriculum design: We manually design two curricula `C1` and `C2` without traffic rules (V2V$^{\text{L}}$ GNN) and two traffic-rule-enhanced curricula `T1` and `T2` (V2V$^{\text{LT}}$ GNN), as specified by the carefully tuned task parameters and weights shown in Tab. IV. All follow the principle of having intermediate tasks deemed relevant for the final task, which help the agent transfer knowledge to the final task.

To this end, stage 1 of `C1` starts with a simplified setting illustrated in Fig. 2 by t_1, where start and goal lanes are corresponding (middle lane), the distance to the goal is low (compared to final distance), all vehicles but one are filtered out from the scenario, and the components *LaneChange* and *Offroute* have zero weight. This allows the agent to focus on the basic skills of goal-seeking and learning to drive within the highway bounds. This behavior is further trained in stage 2 with a slightly more difficult setting by loading up to 10 vehicles into the scenario (distributed over the 150 time steps), which requires the agent to obtain collision avoidance skills. Stage 3 introduces the task of lane keeping and further hardens the setting by including up to 20 vehicles. Also, the goal distance is increased to the final value. Lastly, the final stage adds the task of smooth lane center driving in a scenario with up to 30 vehicles, which is close to the total number of vehicles typically contained in highD scenarios. Additionally, the final task of seeking the goal placed on a

TABLE IV: Parametrization of our curricula with reward notation from Tab. II and a short summary of the driving task sequence to be learned. Reward weights not listed default to 1. `[mid, rnd]` for the goal lane l_g abbreviate `[middle, random]`, a and b in w_{ttc} denote the time-to-collision and closeness metric, respectively, and $(w_{g1/g3}, w_{g2})$ only apply to the V2V$^{\text{LT}}$ curricula `T1` and `T2`.

$\mathbf{S_j}$	$\mathbf{e_{max}}$	$\mathbf{d_{s,g}}$	$\mathbf{l_g}$	$\mathbf{n_{veh}}$	$\mathbf{w_{coll}}$	$\mathbf{w_{route}}$	$\mathbf{w_{offLC}}$	$\mathbf{w_{hdErr}}$	$\mathbf{w_{accel/sAngl}}$	$\mathbf{w_{velo}}$	$\mathbf{w_{ttc}}$	$\mathbf{w_{trajP}}$	$\mathbf{w_{lanCh}}$	$\mathbf{w_{g1/g3}}$	$\mathbf{w_{g2}}$
Curriculum 1 (`C1`/`T1`): *at increasing traffic: basic goal-seeking → collision avoidance → lane keeping → smooth lane center driving/lane changing*															
1	3000	$70\,m$	`mid`	≤ 1	1	0	0	0.2	0	0.002	0.1_a	0.3	0	0.02	0
2	7000	$70\,m$	`mid`	≤ 10	1	0	0	0.2	0	0.002	0.1_a	0.3	0	0.02	0
3	11000	$100\,m$	`mid`	≤ 20	1	0.1	0	0.2	0	0.002	0.05_a	0.3	1	0.02	0
4	10^7	$100\,m$	`rnd`	≤ 30	1	0.1	0.01	0.2	0.005	0.002	0.05_a	0.3	1	0.02	0.02
Curriculum 2 (`C2`/`T2`): *at dense traffic: basic goal-seeking & collision avoidance → lane keeping → smooth lane center driving → lane changing*															
1	5000	$70\,m$	`mid`	$\leq \infty$	1	0	0	0.2	0	0.002	0.2_b	0.3	0	0.02	0
2	9000	$70\,m$	`mid`	$\leq \infty$	0.5	0.1	0	0.2	0	0.002	0.1_b	0.3	1	0.02	0
3	13000	$100\,m$	`mid`	$\leq \infty$	0.5	0.1	0.01	0.2	0.005	0.002	0.1_b	0.3	1	0.02	0
4	10^7	$100\,m$	`rnd`	$\leq \infty$	0.5	0.1	0.01	0.2	0.005	0.002	0.1_b	0.3	0.25	0.02	0.02

random lane is introduced, which enforces lane changes in some cases, as illustrated in Fig. 2 by t_N.

`C2` pursues a similar strategy of `C1` but does not modify the number of vehicles in the scenario. This follows the intuition that guaranteeing a dense V2V graph from the start of training lets the agent learn to avoid collisions in complex traffic from the early stages on. Additionally, unlike `C1`, the tasks of smooth lane center driving and random goal-seeking are distributed into two stages to ease the learning process.

For `T1` and `T2`, all three traffic rule-based rewards are activated. Due to their immediate relevance to basic driving skills, such as collision avoidance and adhering to speed limits, we activate the safety distance (G1) and speed limit rewards (G3) from the first stage. In contrast, the reward for unnecessary breaking (G2) encourages smooth driving behavior, similar to the acceleration penalty. Therefore, we only activate it in the final stages concerned with smooth lane center driving and lane changing.

To evaluate the effectiveness of all curricula, the baselines `B1` and `B2` are trained from scratch with the target task setup of the final stage 4. Additionally, the performance function p_t is activated for `C1`, resembling the reward function of the final stage of `C1` with the rewards r_{reGo} and r_{coll} excluded, as they are analyzed separately in Fig. 3b and Fig. 4. Thus, p_t evaluates smooth and lane-compliant driving behavior.

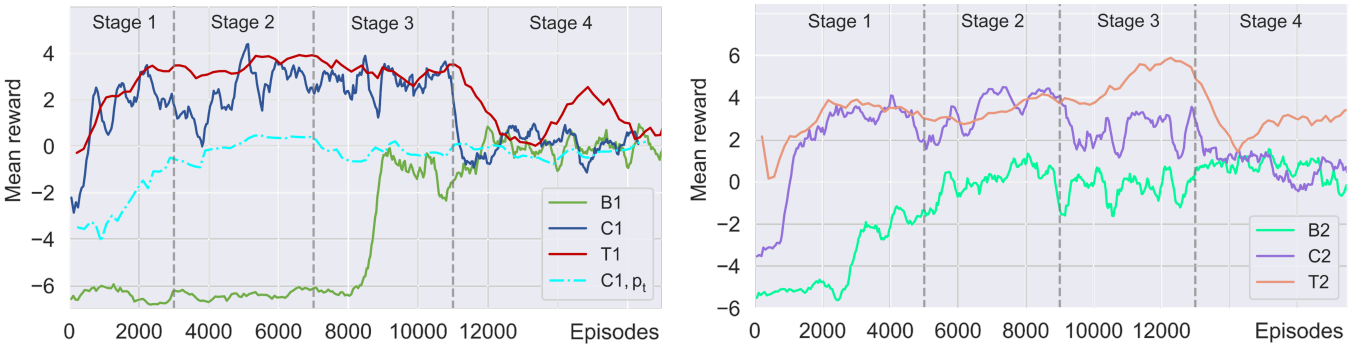
B. Evaluation

1) CL vs. baseline: We compare `C1`/`C2` with `B1`/`B2` on the metrics shown in Fig. 3 and Fig. 4, which visualize the learning progress of the corresponding agents. In Fig. 3a, `C2` exhibits a jumpstart over `B2` in mean rewards during initial training, reaching $r_t > 0$ about 6000 episodes earlier. As shown in Fig. 3b, this is attributed to the agent learning the subtasks of goal reaching and navigating in the road bounds faster. It indicates that the easier task of stage 1 in `C2` accelerates the learning process. However, a sharp drop in the *ReachedGoal* ratio in stage 4 (random goal lane) over stage 3 (goal in same lane as start) suggests that the data distributions to be learned for fixed and random goal seeking are too different, causing the `C2` agent to have a slightly higher collision rate than `B2` and similar mean rewards, as seen in Fig. 3 and Fig. 4. Presumably, the agent has converged to the same-lane tasks of the stages 1-3, and has not adapted to the multi-lane task of stage 4.

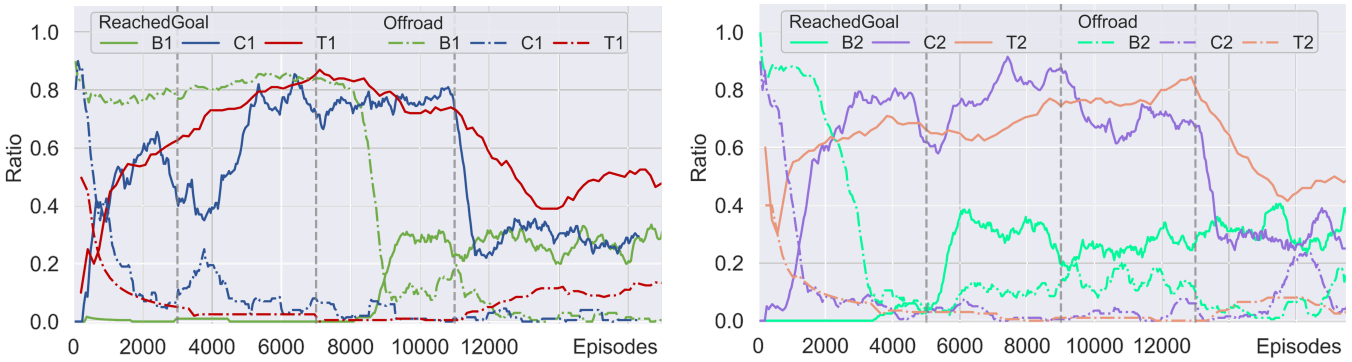
Results from `C1` show a similar pattern, where the initial jumpstart advantage over `B1` is significantly more evident. This suggests that the reduced traffic density due to `C1` eased and accelerated the initial learning task. At the end, `C1` achieves a collision rate on par with that of `B1`. The development of p_t in Fig. 3a indicates that driving aspects other than goal seeking and collision avoidance are already learned early until mid-stage 2, from where p_t stagnates.

Although our GNN is based on the model of [9], a direct performance comparison is not feasible, as the task setups are too different. In the two-lane simulation of [9], the agent always performs a merge from the right into the left lane in dense traffic. The other vehicles are simulated as highly reactive to avoid collisions when the agent cuts in, whereas in our real-world data the prerecorded trajectories of other vehicles do not react to the RL agent at all. In our setup, the task of collision avoidance lies fully with the agent that additionally has to deal with lane-dependent speed distributions, constituting a task of higher difficulty. This results in higher final collision rates of `B1`/`B2` and `C1`/`C2` than those reported in [9]. We reimplemented the GNN-RL agent of [9] and evaluated it on our task of `B2`. Despite having a specific goal distance reward [9], the agent was unable to learn basic goal-reaching behavior, indicating the benefits of our hybrid model architecture. In contrast, Fig. 5 shows an exemplary lane changing scenario at high traffic density that our agent managed to navigate successfully. In general, the agent learns to correctly and quickly change to the left or right lane depending on the randomly specified goal. Since the agent executes left lane changes into the fast lane with the same speeds as right lane changes into the slow lane, the different per-lane speed distributions of our highD data have not yet fully materialized.

2) Traffic rule-enhanced CL: Enhancing the curricula `C1` and `C2` by traffic rules improves the goal-reaching capabilities and rewards in the final stage, observed for both `T1` and `T2`. It also reduces reward fluctuations at stage changes. E.g., in Fig. 3, the start of stage 2 of `T1` does not cause a sharp



(a) Mean accumulated rewards per episode. Both C1/C2 show accelerations in learning over B1/B2, while the traffic rule-enhanced T1/T2 further exhibit more stable rewards at stage switches and some performance gains on the final task.



(b) Termination ratios of success (goal reached) and lane geometry-related failure (driven offroad) per episode.

Fig. 3: Numerical evaluation of our two CL setups with the V2V$^{\text{L}}$ curricula C1/C2, the V2V$^{\text{LT}}$ curricula T1/T2, and baselines B1/B2.

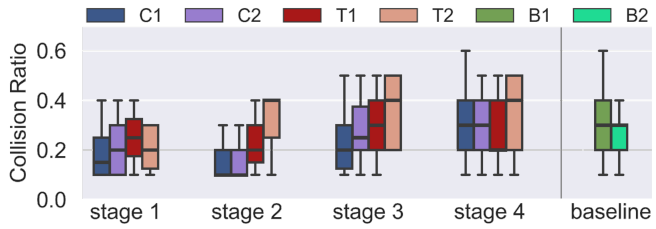


Fig. 4: Ratio of episodes of runs 1 and 2 that ended with *Collision*, computed for the last 25% of each stage for C1/C2 and T1/T2 and of last 25% for B1/B2.

drop in *ReachedGoal* ratio present for C1. T2 manages to improve over C2 by keeping the *Offroad* ratio low in the final stage. However, the collision avoidance deteriorates slightly for T2, while staying comparable for T1 and C1.

In T1/T2, we only calculate the traffic rule robustness between the agent and other vehicles $f_{g1\text{-}3,T}$ in $\mathbf{f}_{ego,LT}$ and omit pairs of other vehicles to reduce the computational load. As an extension, we compared this simplified setup with a V2V$^{\text{LT}}$ GNN with $f_{g1\text{-}3,T}$ in $\mathbf{h}_v^k$ and the predicates in $\mathbf{e}_{vw}^k$ for all vehicles. The results showed that the computational simplification did not significantly affect the agent's performance and traffic rule violation effects over multiple other vehicles are hence less relevant to our planning problem.

V. Conclusion

In this work, we have introduced a novel CL scheme for motion planning in autonomous driving that incorporates traffic rule and map information into a GNN-RL agent. Extending V2V graph-based GNN-RL agents for highway driving from previous works with features and rewards related to the road geometry enables the agents to also navigate at low traffic densities. This allows for setting up task curricula for highway driving that progressively increase the traffic density along with making the agent's longitudinal and lateral driving route more complex. By starting with simpler tasks before attempting more complex ones, our curriculum-based agents can learn basic driving skills such as goal-reaching behavior faster than non-curriculum baselines trained from scratch on the most complex target task. Different from several previous simulation-based works, our curriculum-based agents are evaluated on prerecorded real-world traffic from multi-lane highways. To increase the safety of the learned driving behavior, we incorporate for the first time three temporal logic-based traffic rules about safety distance, unnecessary braking, and speed limits as features and rewards into the agent. Introducing traffic-rule features and rewards in the traffic-rule-enhanced curricula has shown to stabilize performance at stage switches and to ultimately improve final driving performance.

Acknowledgment

This research was funded by the German Federal Ministry for Digital and Transport in the KoSi project (grant 01MM20011A) and by the Federal Ministry of Education and Research in the MCube project (grants 03ZU1105BA and 03ZU1105KA).

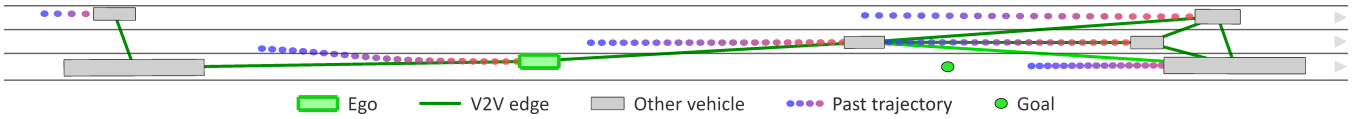


Fig. 5: Example of the agent's learned motion planning capabilities to reach the goal (green circle) by a lane change into the slower lane.

REFERENCES

[1] B. R. Kiran, I. Sobh, V. Talpaert, *et al.*, "Deep reinforcement learning for autonomous driving: A survey," *IEEE T-ITS*, vol. 23, no. 6, pp. 4909–4926, 2021.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[3] S. Narvekar, B. Peng, M. Leonetti, *et al.*, "Curriculum learning for reinforcement learning domains: A framework and survey," *JMLR*, vol. 21, no. 1, pp. 7382–7431, 2020.

[4] Y. Bengio, J. Louradour, R. Collobert, *et al.*, "Curriculum learning," in *ICML*, 2009, pp. 41–48.

[5] Y. Song, H. Lin, E. Kaufmann, *et al.*, "Autonomous overtaking in Gran Turismo Sport using curriculum reinforcement learning," in *IEEE ICRA*, 2021, pp. 9403–9409.

[6] A. Ozturk, M. B. Gunel, R. Dagdanov, *et al.*, "Investigating value of curriculum reinforcement learning in autonomous driving under diverse road and weather conditions," in *IEEE IV Workshops*, 2021, pp. 358–363.

[7] M. Kaushik, V. Prasad, K. M. Krishna, *et al.*, "Overtaking maneuvers in simulated highway driving using deep reinforcement learning," in *IEEE IV*, 2018, pp. 1885–1890.

[8] L. Anzalone, P. Barra, S. Barra, *et al.*, "An end-to-end curriculum learning approach for autonomous driving scenarios," *IEEE T-ITS*, vol. 23, no. 10, pp. 19 817–19 826, 2022.

[9] P. Hart and A. Knoll, "Graph neural networks and reinforcement learning for behavior generation in semantic environments," in *IEEE IV*, 2020, pp. 1589–1594.

[10] E. Meyer, L. F. Peiss, and M. Althoff, "Deep occupancy-predictive representations for autonomous driving," in *IEEE ICRA*, 2023, pp. 5610–5617.

[11] M. Hügle, G. Kalweit, M. Werling, *et al.*, "Dynamic interaction-aware scene understanding for reinforcement learning in autonomous driving," in *IEEE ICRA*, 2020, pp. 4329–4335.

[12] J. Dong, S. Chen, P. Y. J. Ha, *et al.*, "A DRL-based multiagent cooperative control framework for CAV networks: A graphic convolution Q network," 2020. arXiv: 2010.05437.

[13] X. Gao, X. Li, Q. Liu, *et al.*, "Multi-agent decision-making modes in uncertain interactive traffic scenarios via graph convolution-based deep reinforcement learning," *Sensors*, vol. 22, no. 12, 4586, 2022.

[14] Q. Liu, Z. Li, X. Li, *et al.*, "Graph convolution-based deep reinforcement learning for multi-agent decision-making in interactive traffic scenarios," in *IEEE ITSC*, 2022, pp. 4074–4081.

[15] Q. Liu, X. Li, Z. Li, *et al.*, "Graph reinforcement learning application to co-operative decision-making in mixed autonomy traffic: Framework, survey, and challenges," 2022. arXiv: 2211.03005.

[16] M. Xiaoqiang, Y. Fan, L. Xueyuan, *et al.*, "Graph convolution reinforcement learning for decision-making in highway overtaking scenario," in *IEEE ICIEA*, 2022, pp. 417–422.

[17] S. Wang, H. Fujii, and S. Yoshimura, "Generating merging strategies for connected autonomous vehicles based on spatiotemporal information extraction module and deep reinforcement learning," *Phys. A*, vol. 607, 128172, 2022.

[18] P. C. Hart, "Learning, evaluating and optimizing behavior policies for autonomous vehicles," Ph.D. thesis, TU Munich, 2021.

[19] X. Mei, Y. Sun, Y. Chen, *et al.*, "Autonomous navigation through intersections with graph convolutional networks and conditional imitation learning for self-driving cars," 2021. arXiv: 2102.00675.

[20] M. Klimke, B. Völz, and M. Buchholz, "Cooperative behavior planning for automated driving using graph neural networks," in *IEEE IV*, 2022, pp. 167–174.

[21] M. Klimke, J. Gerigk, B. Völz, *et al.*, "An enhanced graph representation for machine learning based automatic intersection management," in *IEEE ITSC*, 2022, pp. 523–530.

[22] M. Klimke, B. Völz, and M. Buchholz, "Automatic intersection management in mixed traffic using reinforcement learning and graph neural networks," in *IEEE IV*, 2023, pp. 1–8.

[23] S. Chen, J. Dong, P. Ha, *et al.*, "Graph neural network and reinforcement learning for multi-agent cooperative control of connected autonomous vehicles," *COMPUT-AIDED CIV INF*, vol. 36, no. 7, pp. 838–857, 2021.

[24] L. Anzalone, S. Barra, and M. Nappi, "Reinforced curriculum learning for autonomous driving in CARLA," in *IEEE ICIP*, 2021, pp. 3318–3322.

[25] R. Gutiérrez-Moreno, R. Barea, E. López-Guillén, *et al.*, "Reinforcement learning-based autonomous driving at intersections in CARLA simulator," *Sensors*, vol. 22, no. 21, 8373, 2022.

[26] S. Khaitan and J. M. Dolan, "State dropout-based curriculum reinforcement learning for self-driving at unsignalized intersections," in *IEEE IROS*, 2022, pp. 12 219–12 224.

[27] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *IEEE ITSC*, 2020, pp. 1–7.

[28] P. Wolf, K. Kurzer, T. Wingert, *et al.*, "Adaptive behavior generation for autonomous driving using deep reinforcement learning with compact semantic states," in *IEEE IV*, 2018, pp. 993–1000.

[29] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, *et al.*, "Formalization of interstate traffic rules in temporal logic," in *IEEE IV*, 2020, pp. 752–759.

[30] Q. Gao, D. Hajinezhad, Y. Zhang, *et al.*, "Reduced variance deep reinforcement learning with temporal logic specifications," in *IEEE ICCPS*, 2019, pp. 237–248.

[31] J. Rong and N. Luan, "Safe reinforcement learning with policy-guided planning for autonomous driving," in *IEEE ICMA*, 2020, pp. 320–326.

[32] J. Lin, W. Zhou, H. Wang, *et al.*, "Road traffic law adaptive decision-making for self-driving vehicles," in *IEEE ITSC*, 2022, pp. 2034–2041.

[33] E. Meyer, M. Brenner, B. Zhang, *et al.*, "Geometric deep learning for autonomous driving: Unlocking the power of graph neural networks with CommonRoad-Geometric," in *IEEE IV*, 2023, pp. 1–8.

[34] F. Scarselli, M. Gori, A. C. Tsoi, *et al.*, "The graph neural network model," *IEEE TNN*, vol. 20, no. 1, pp. 61–80, 2008.

[35] J. Gilmer, S. S. Schoenholz, P. F. Riley, *et al.*, "Neural message passing for quantum chemistry," in *ICML*, 2017, pp. 1263–1272.

[36] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016. arXiv: 1609.02907.

[37] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *FTRTFT*, 2004, pp. 152–166.

[38] L. Gressenbuch and M. Althoff, "Predictive monitoring of traffic rules," in *IEEE ITSC*, 2021, pp. 915–922.

[39] J. C. Hayward, "Near-miss determination through use of a scale of danger," *HRR*, vol. 384, pp. 24–34, 1972.

[40] J. Schulman, F. Wolski, P. Dhariwal, *et al.*, "Proximal policy optimization algorithms," 2017. arXiv: 1707.06347.

[41] W. Knospe, L. Santen, A. Schadschneider, *et al.*, "Single-vehicle data of highway traffic: Microscopic description of traffic phases," *Phys. Rev. E*, vol. 65, no. 5, 056133, 2002.

[42] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *IEEE IV*, 2017, pp. 719–726.

[43] R. Krajewski, J. Bock, L. Kloeker, *et al.*, "The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems," in *IEEE ITSC*, 2018, pp. 2118–2125.

[44] A. Raffin, A. Hill, A. Gleave, *et al.*, "Stable-Baselines3: Reliable reinforcement learning implementations," *JMLR*, vol. 22, no. 268, pp. 1–8, 2021.