Deep Reinforcement Learning for Autonomous Driving using High-Level Heterogeneous Graph Representations

Maximilian Schier[1], Christoph Reinders[1] and Bodo Rosenhahn[1]

***Abstract*— Graph networks have recently been used for decision making in automated driving tasks for their ability to capture a variable number of traffic participants. Current high-level graph-based approaches, however, do not model the entire road network and thus must rely on handcrafted features for vehicle-to-vehicle edges encompassing the road topology indirectly. We propose an entity-relation framework that intuitively models the road network and the traffic participants in a heterogeneous graph, representing all relevant information. Our novel architecture transforms the heterogeneous road-vehicle graph into a simpler graph of homogeneous node and edge types to allow effective training for deep reinforcement learning while introducing minimal prior knowledge. Unlike previous approaches, the vehicle-to-vehicle edges of this reduced graph are fully learnable and can therefore encode traffic rules without explicit feature design, an important step towards a holistic reinforcement learning model for automated driving. We show that our proposed method outperforms precomputed handcrafted features on intersection scenarios while also learning the semantics of right-of-way rules.**

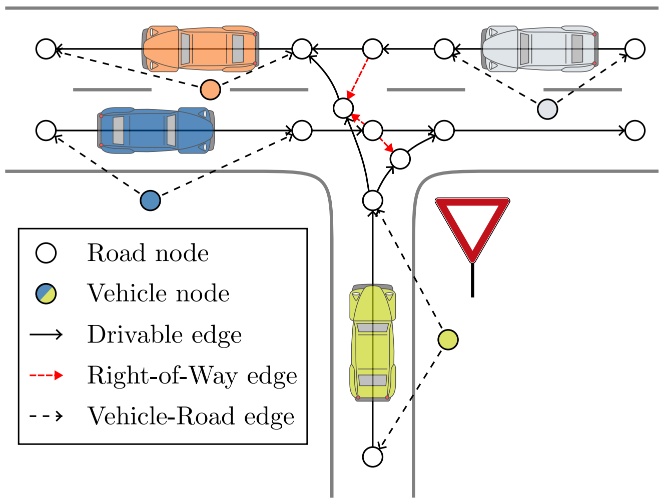


Fig. 1: Our proposed representation models full road topology and vehicles in a heterogeneous directed graph. Road nodes are connected through drivable or relational edges and vehicles linked to two road nodes to represent their position on their connecting drivable edge. This heterogeneous graph is then transformed by our framework into a simpler vehicle graph with learnable edges based on the connecting routes between vehicles. Here, the ego vehicle (light green) must safely join the prioritized road by adjusting velocity.

I. INTRODUCTION

In recent years, model-free approaches for decision making in automated driving have enjoyed growing success, with research focusing on various methods of reinforcement learning (RL) for problems like unsignalized intersection navigation [1], [2]. A fundamental problem for RL in automated driving remains finding a suitable representation of the environment that is both abstract enough to generalize well to scenarios not seen during training as well as of the required expressiveness.

Graphs have been used previously to represent traffic participants in automated driving [3], [4], [5], [6], as participants can intuitively be modeled as nodes. The advantages of graphs are well-known: they are both abstract, contain little redundant information compared to, for example, bird-eye view images, are permutation invariant, and support an arbitrary number of participants compared to, for example, vector representations [3], [7]. The choice of edges connecting vehicle nodes is, however, less natural with both the question of which vehicles to connect and which edge features to select depending on the scenario to be solved.

The applicability of graphs is not limited to modeling vehicles. They can naturally express the underlying road networks, for example by setting a node at every lane joint of the road network and connecting with edges such that every edge is a drivable connection. Such graph representations are in fact already used for road networks in the simulator SUMO [8], that is employed in many recent works [1], [9], [10]. Despite this natural mapping from the road network to a graph, previous research has skipped modeling the road on top of the vehicle graph and instead opted to include road topology in the form of handcrafted features on the vehicle-to-vehicle edges [3], [4], [5]. While such feature-crafting allows in principle easier training of an RL agent on the scenarios the features were designed for, it is still desirable to have the agent deduce semantics from the entire road topology for general applicability and to move towards a holistic reinforcement agent comprising all situations in automated driving. In this paper, we propose joining the vehicle and road topology together into a heterogeneous graph, containing both the road nodes and road network edges, as well as the vehicle nodes. We omit the vehicle-to-vehicle edges, as such edges must rely on precomputed features like relative velocity, relative position [3], the inverse of a Mahalanobis distance, or the type of relation between vehicles, i.e. whether they are crossing or following [5]. Such features are chosen specifically for the task at hand, and do not necessarily generalize to different scenarios. Instead, we express the vehicle position relative to the road network

[1]Leibniz University Hannover, Institute for Information Processing {schier,reinders,rosenhahn}@tnt.uni-hannover.de

and only add vehicle-to-road edges. An illustration of the whole heterogeneous graph is given in Fig. 1. A problem remains the complexity of the graph and the number of graph convolutions required to exchange information between vehicles, as it is difficult to train an agent that requires multiple message-passing steps in the graph. Therefore, we propose a novel framework and network architecture that is capable of learning useful vehicle-to-vehicle edges by reducing a heterogeneous vehicle-road graph to a tree with learned edge features.

The main contributions of our work are:

- We propose, to the best of our knowledge, the first graph-based environment representation for automated driving containing the entire road topology and vehicle information in a heterogeneous high-level graph
- We introduce a new model architecture for reducing the complexity of such a graph representation to levels suitable for reinforcement learning by folding the vehicle-to-vehicle paths into learnable vehicle-to-vehicle edges
- We train multiple agents based on our representation and model using reinforcement learning and show that our model is able to both generalize and learn the inherent right-of-way rules while outperforming precomputed relative vehicle-to-vehicle features
- Upon acceptance, we release the source code and configuration of our model and used environments

II. RELATED WORK

Reinforcement learning for high-level driving can be categorized into different levels of cooperativeness between vehicles and different key scenarios. For uncooperative scenarios, all agents but the ego agent execute a fixed policy that generally obeys traffic rules like right-of-way but otherwise ignores the ego agent. For cooperative scenarios, agents are controlled by a jointly-learned policy to optimize a metric like average flow rate. While this work learns a policy for an uncooperative environment, due to its focus on the novel architecture and representation we will briefly outline related work from both domains.

For uncooperative lane-changing and on-ramp merging, Wolf *et al.* [14] propose training a deep Q-network (DQN) using a relational grid encoding. The vehicles and road topology surrounding the agent are encoded on a tabular grid with each vehicle occupying a cell. The number of vehicles is limited by the grid size. Huegle *et al.* [7] propose using a deep set [15] to solve the same scenario while being unbounded in the total number of observed vehicles compared to a fixed size grid, and later refine their work to support multiple object classes [10] and surrogate learning [9]. Hart and Knoll [3] propose a graph representation optimized for the scenario of lane-changing where vehicles are encoded as vertices and relative location as edges.

For the uncooperative crossing of unsignalized intersections, Kai *et al.* [2] formulate intersection navigation as a multi-task problem, representing the intersection as a fixed-size vector, thus limiting the number of observable vehicles. Kurzer *et al.* [1] propose encoding the environment as a vector of path segments, with each segment containing precalculated current and future occupancy information. This representation generalizes to unseen intersection layouts but requires precomputation of the predicted occupancy information for the entirety of the discretized look-ahead path.

Klimke *et al.* [5] propose a graph-based model for fully cooperative intersection flow optimization, with graph nodes representing vehicles and graph edges only connecting conflicts in the form of crossing and following relations. The distance to the closest vehicle is encoded per vehicle as a node value. In their follow-up work [4], the distance to each conflicting vehicle is encoded as edge features.

Most other recent work on RL not focused on the representation relies on fixed-size vectors, e.g. for merging [16], unsignalized intersections [2], and lane-changing/highway-driving [17], [18], rarely also convolutional neural networks [19].

Graph networks have also been employed for low level scene representation for automated driving. For example, Gao *et al.* [20] propose an encoding of HD maps using two graph networks, first aggregating the low level polygonal object descriptors as polylines, which are then processed by a second graph network to model high-level relationships.

III. APPROACH

We formulate the problem of vehicle control as a Markov Decision Process (MDP). The agent's goal is controlling the ego vehicle's speed while navigating through a given scenario. During each time step, the agent chooses an action using the current environment observation, receives a reward based on current environment state and selected action, and the environment is simulated until the next time point. Such an MDP may be described as the tuple $(\mathcal{S},\mathcal{A},\mathcal{T},\mathcal{R},\gamma)$ with the state space $\mathcal{S}$, the action space $\mathcal{A}$, the transition rate function $\mathcal{T}:\mathcal{S}\times\mathcal{A}\rightarrow\mathcal{S}$, the reward function $\mathcal{R}:\mathcal{S}\times\mathcal{A}\rightarrow\mathbb{R}$, and the discount factor γ reducing future rewards.

The agent's policy π maps from the set of available actions for a given state to the probability of each action $\pi:\mathcal{S}\times\mathcal{A}\rightarrow[0,1]$. The objective of the agent is maximizing the expected total reward G over all future time steps t when starting from an initial state with $G=\sum_t \mathrm{E}[\gamma^t\mathcal{R}(s_t,a_t)]$. Let $Q^*(s',a')$ be the Q-value of the optimal policy for the next time step with $s'=\mathcal{T}(s,a)$, $a'\in\mathcal{A}$. Then the optimal Q-value for the current state s and some action a is:

$$Q^*(s,a)=\mathcal{R}(s,a)+\gamma\max_{a'\in\mathcal{A}}Q^*(s',a'). \tag{1}$$

Deep Q-Learning approximates the optimal policy Q^* using a neural network Q_θ with learned parameters θ which are iteratively updated using the loss:

$$\mathcal{L}=(r+\gamma\max_{a'\in\mathcal{A}}Q_\theta(s',a')-Q_\theta(s,a))^2 \tag{2}$$

minimizing the difference between predicted reward and the sum of immediate reward and discounted future predicted reward. The best action according to the learned policy π_θ can then be determined for a state s by $\mathrm{argmax}_{a\in\mathcal{A}}Q_\theta(s,a)$.

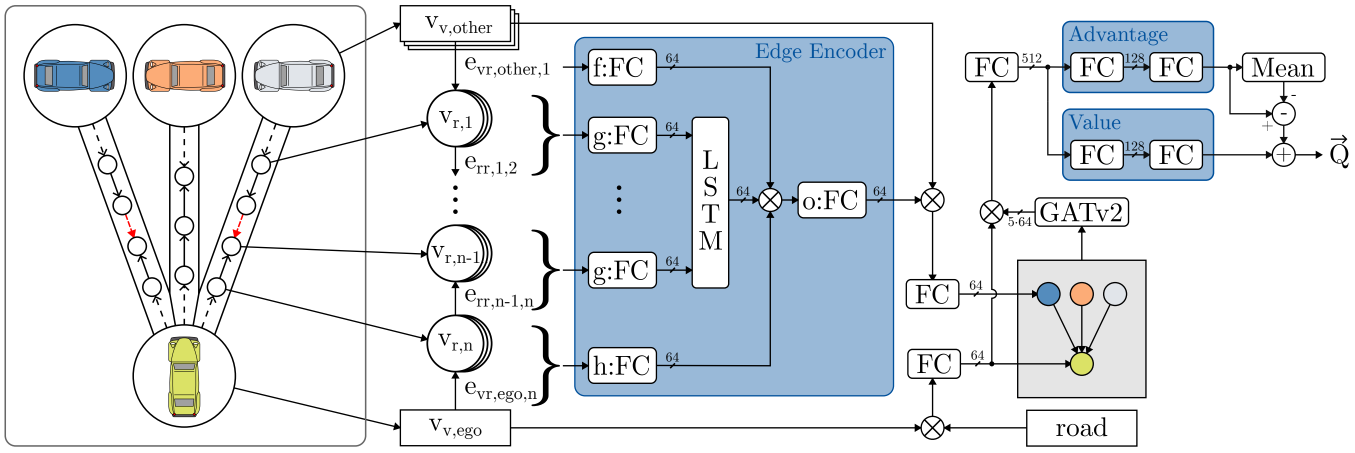


Fig. 2: We transform the heterogeneous graph into a simpler vehicle graph with edges described by the connecting paths between vehicles. Here, the situation from Fig. 1 is used as an example. Our proposed architecture encodes these paths using an LSTM [11]. We construct a bipartite graph (in grey) from the edge encoding, the observed and ego vehicle information, and road features. After transforming the graph using a GATv2 [12] convolution, a duelling DQN head [13] estimates the action rewards. Concatenation is denoted with $\otimes$, all layers except the two final layers of the head use ReLU activations.

A. Environment Representation

To represent traffic participants on a road network, we propose a novel heterogeneous graph representation. A heterogeneous directed graph $G=(\mathcal{V},\mathcal{E},\mathcal{A},\mathcal{R})$ consists of a set of vertices $\mathcal{V}$, an edge relation $\mathcal{E}\subseteq\{(x,y)\mid(x,y)\in\mathcal{V}\times\mathcal{V}\}$, and vertex and relation types $\mathcal{A}$ and $\mathcal{R}$, respectively. Each vertex and edge is mapped to its respective type using the type functions $a:\mathcal{V}\rightarrow\mathcal{A}$ and $r:\mathcal{E}\rightarrow\mathcal{R}$. Such a graph is easily extensible for new participant types, like vulnerable road users. In this work, we model vehicles and road topology. We denote $v_{v,i}$ as the i-th vehicle node, $v_{r,i}$ as the i-th road node, $e_{rr,i,j}$ as the edge connecting from the i-th to the j-th road node, and $e_{vr,i,j}$ as the edge connecting from the i-th vehicle to the j-th road node. Both nodes and edges have additional attributes. Let $\underline{\mathbf{v}}_{t,i}$ be the attribute vector of the i-th node of type t, edge attributes are notated accordingly.

For each vehicle node, the velocity in the current and previous time step, maximum capable velocity, and state of the left and right indicators are encoded. For each road node, we encode the maximum allowed velocity and whether this road node is a goal, which is only necessary for the agent to estimate when it will receive a positive terminal reward.

For the vehicle-to-road edges, the encoding consists of relative and absolute distance between vehicle and road node and whether the vehicle is traveling towards or from the node. For example, if a vehicle is 200m away from the previous node and 50m from the upcoming node, the relative distances would be 80% and 20%, respectively.

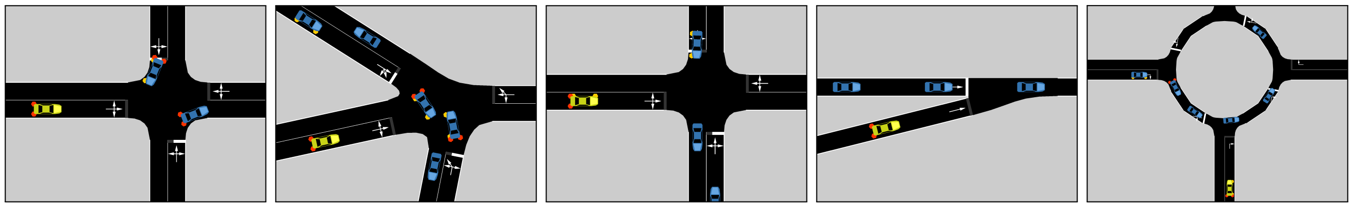
For road-road edges, we encode the type of edge and the distance covered by the edge if it is of drivable type. Drivable types are those edges the vehicle may travel along, which are *Continuation* if a road segment is split into multiple parts and *LinkLeft*, *LinkRight*, and *LinkStraight* for junction links requiring the left indicator, right indicator, or no indicator to be set during junction crossing respectively. Unpassable types encode traffic rule limitations and are either *CrossingWithYield* connecting from a junction point with lower priority to a junction point with higher priority and *CrossingWithRightOfWay* in the reverse direction since a directed graph is formed. All attributes of the road topology except goal information can be directly extracted from the road map and are static, i.e. invariant to the state of all vehicles. Thus the road graph can be efficiently initialized once and be reused for every observation without modification.

All categorical features are encoded as binary class matrices. Continuous features are normalized to a range of $[-1,1]$ through division by $50\,\mathrm{m\,s^{-1}}$ for velocities and $200\,\mathrm{m}$ for distances, followed by clipping.

B. Graph Model

Given a heterogeneous graph of the ego and other vehicles on a road network, a naïve approach to predicting the Q-values for each action would be graph folding with an attention mechanism using a graph convolution with support for edge features and heterogeneous nodes like GATv2 [12]. However, we have found that such an approach does not yield good results as the multiple message-passing steps combined with the sparse signal of the Q-learning loss prevent learning to attend to correct nodes and features. This may also be explained by the large mostly irrelevant space of the environment representation for the control problem.

Therefore, we propose reducing the heterogeneous vehicle-road graph to a tree of vehicles with maximum height of one and the ego vehicle as root. For that, the shortest path from each observed vehicle to the ego vehicle is constructed using Dijkstra's algorithm [21] with each road node equally weighted except for edges along the route, which have reduced cost. Such a path P connecting vehicle nodes $v_{v,\mathrm{other}}$ to $v_{v,\mathrm{ego}}$ over road nodes with indices $\{1,\ldots,n\}$ consists of the initial vehicle-to-road edge from the other vehicle to a road node $e_{vr,\mathrm{other},1}$, then for every but the last



(*S1*) Basic Junction (*S2*) Curved Junction (*S3*) Junction Left Turn (*S4*) Junction Merge (*S5*) Roundabout

Fig. 3: Scenarios used in the experiments on the SUMO simulator [8]. On junctions *S1* through *S4* two right-of-way variants are used. On *S5* merging traffic must always yield. The ego vehicle (light green) always starts on the same leg.

visited road node with index i the road node itself $v_{r,i}$ and the next road-road edge $e_{rr,i,i+1}$, followed by the final road node $v_{r,n}$ and the vehicle-to-road edge of the ego vehicle $e_{vr,\text{ego},n}$:

$$P = (\underline{\mathbf{e}}_{vr,\text{other},1}, \underline{\mathbf{v}}_{r,1}, \underline{\mathbf{e}}_{rr,1,2}, \dots, \underline{\mathbf{v}}_{r,n-1}, \underline{\mathbf{e}}_{rr,n-1,n}, \underline{\mathbf{v}}_{r,n}, \underline{\mathbf{e}}_{vr,\text{ego},n}). \tag{3}$$

Road-to-road edges of the directed graph can be traversed in both directions. To differentiate, we pad the feature vector to twice its size with zeros either in front or back depending on the direction of travel. P is split into the fixed length start and end, and variable length center section. The dynamic part of P is encoded by an LSTM [11] with ReLU activation to a fixed-size vector. With the segment encoders $f(\cdot)$, $g(\cdot)$, $h(\cdot)$, and output encoder $o(\cdot)$ consisting of a learnable affine transformation followed by a ReLU, the edge encoding $\mathbf{y}$ is:

$$\mathbf{y} = o([f(\underline{\mathbf{e}}_{vr,\text{other},1})||\mathbf{x}||h([\underline{\mathbf{v}}_{r,n}||\underline{\mathbf{e}}_{vr,\text{ego},n}])]), \text{ with} \tag{4}$$

$$\mathbf{x} = \text{LSTM}(g([\underline{\mathbf{v}}_{r,1}||\underline{\mathbf{e}}_{rr,1,2}]), \dots, g([\underline{\mathbf{v}}_{r,n-1}||\underline{\mathbf{e}}_{rr,n-1,n}])). \tag{5}$$

Here, $[\cdot||\cdot]$ marks concatenation. In this way, we create a similar graph to previous work using graph networks with reinforcement learning for vehicle control [3], [5], however, instead of situation-dependent handcrafted edge features our approach uses implicitly learned features describing the vehicle-to-vehicle relations. To encode the entire scene, we construct a bipartite graph. The concatenation of ego vehicle and road features is the destination node. Each observed vehicle's features concatenated with its encoded edge features is a source node. For our proposed framework, forward road information is encoded as the current forward road node of the ego vehicle concatenated with the average feature of all succeeding road nodes, as we have found this sufficient. The road nodes may, however, also be integrated as a separate reduced graph similar to the observed vehicles if required. Source and destination node values are transformed with a fully connected layer respectively and the graph is convolved once using the GATv2 [12] operator with five heads. We concatenate orignal and convolved features of the destination node, as self-links are not used on the bipartite graph. An affine transform is applied and a standard duelling head [13] consisting of two fully connected layers with ReLU activation for both the advantage and value calculation estimates $\vec{Q}$, the vector of Q-value per action.

C. Observed Vehicles

We set the vision radius of the ego vehicle to 100 m. All vehicles outside this radius are not observable and not included in the environment representation. As noted by previous work [10], the complexity of graph representations should be kept minimal to reduce computational effort. Therefore, we propose limiting the set of observed vehicles to vehicles that are reachable through a floodfill with limited depth on the road graph. Starting from the ego vehicle's node, all road nodes are added to the flood fill frontier if the connecting edge is vehicle free. If vehicles are on the connecting edge, the closest one is added to the observation space, but the following road node is not added to the frontier of the flood fill. We later show in an ablation study that our agent performs well with both this approach and all vehicles observed.

D. Reward and Actions

The reward function $r(\cdot)$ should encourage safe and comfortable driving while completing the episode quickly. We design $r(s,a)$ for the next state s given an action a as:

$$r(s,a) = r_{\text{collision}}(s) + r_{\text{success}}(s) + r_{\text{velocity}}(s) + r_{\text{acceleration}}(a). \tag{6}$$

The terminal transitions for collision and completion of an episode are rewarded with a full positive or negative reward:

$$r_{\text{collision}}(s) = -\mathbb{1}_{\text{collision}}(s) \tag{7}$$

$$r_{\text{success}}(s) = \mathbb{1}_{\text{success}}(s). \tag{8}$$

Driving at a slower speed v than the allowed speed v_{allowed} is penalized:

$$r_{\text{velocity}}(s) = -k_v \cdot \max(0, v_{\text{allowed}} - v) \tag{9}$$

and any acceleration is penalized for smooth driving:

$$r_{\text{acceleration}}(a) = -k_a \cdot |a|. \tag{10}$$

We set $k_v = 0.001$ and $k_a = 0.0002$ in all experiments, these values were empirically determined.

The agent controls the longitudinal velocity of the ego vehicle by selecting one of the three discrete accelerations $+3\,\text{m}\,\text{s}^{-2}$, $0\,\text{m}\,\text{s}^{-2}$, and $-3\,\text{m}\,\text{s}^{-2}$ per time step.

IV. EXPERIMENTS

Experiments are conducted on the five scenarios shown in Fig. 3, which are based on Kurzer *et. al.* [1] with an additional roundabout. Note, however, that the results are not directly comparable due to the difference in traffic rules, environment setup, and the statistical methods used

TABLE I: Interquartile mean of the performance using our learnable edge features compared to precomputed edge features on all scenarios. Best results highlighted in bold. Improvements on training sets are significant with $p \leq 0.05$. Our architecture performs better in success rate (SR) and collision rate (ETR) both on seen training and unseen evaluation scenarios.

	Training				Evaluation			
Edge Features	SR [%] ↑	CI95 SR [%]	ETR [%] ↓	CI95 ETR [%]	SR [%] ↑	CI95 SR [%]	ETR [%] ↓	CI95 ETR [%]
Precomputed	94.57	[94.15, 94.87]	5.36	[5.05, 5.71]	81.67	[78.91, 83.55]	17.94	[16.06, 19.89]
Learned (ours)	**95.76**	[95.36, 96.19]	**4.18**	[3.75, 4.58]	**83.17**	[79.61, 85.93]	**16.54**	[13.88, 19.76]

for evaluation. For the simulation of the environment, the traffic simulator SUMO [8] is used as in previous works [1], [7], [10]. The ego vehicle always starts on the same road segment as shown in Fig. 3, whereas the traffic flow on all other routes is randomly spawned. The agent's goal is to exit opposite to the entry, with the exception of *S3*, where it must turn left. For scenarios *S1* to *S4*, we employ two variants, one where the ego vehicle's road is prioritized, and one where it must yield. If the ego vehicle has to yield, other vehicles ignore any vehicles, such that SUMO's driver model enforces right-of-way by causing a collision, similar to previous work [1]. This is important to prevent the agent from exploiting SUMO's defensive default driver model. If, however, the ego vehicle has right-of-way, other vehicles use the standard SUMO driver model, which gives way. We simulate a macro step length of 0.4 s by repeating the selected action four times for 0.1 s in the simulator. This step length is consistent with previous work [1], but a smaller simulator step length is used as we have found 0.4 s to cause some collisions not to be detected in SUMO allowing vehicles to completely move through each other at some junction layouts within a time step when driving quickly.

TABLE II: Hyperparameters for the training of the DQN

Parameter	**Value**
Batch size	512
Replay buffer size	100,000
Gradient steps	2,000,000
Steps per gradient step	4
Optimizer	Adam [22] (lr $= 2 \cdot 10^{-5}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$)
Discount factor γ	0.9
Exploration probability ε	1.0 linearly decaying to 0.02
PER parameters	$\alpha = 0.6$, β linearly increasing from 0.4 to 1.0

A. Experimental Setup

The agents are trained similarly to the original DQN [23], but use common enhancements like Double Q-Learning [24] for more stable reward estimation, Duelling Q-Learning [13] for generalized action reward estimation, and Prioritized Experience Replay [25] to focus learning towards rare but important transitions, such as collisions. An ε-greedy strategy with linear decay of the exploration probability ε is used. The hyperparameters are given in Table II.

The performance is analyzed using two established metrics [1]. Success rate (SR) measures the rate of successfully completed episodes over all episodes, and early termination rate (ETR) the rate of early terminations through collision. As episodes are also terminated after a maximum of 600 steps, SR and ETR do not always add up to 1.

B. Evaluation

Since many previous methods have not published their implementation, we implement an agent using handcrafted precomputed features for comparison. This agent uses vehicle-to-vehicle features in the form of relative position vector and relative velocity vector in the ego vehicle's reference coordinate frame, an encoding motivated by previous work [3]. The agent is based on our architecture from Fig. 2 for better comparability, with the input path sequence replaced by the relative features and our recurrent edge encoder replaced by three fully connected layers with ReLU activation of dimensions 256, 128, and 16. This design is comparable in the number of trainable parameters to our edge encoder. The rest of the architecture is not modified.

First, we analyze overall performance of our proposed architecture using learnable vehicle-to-vehicle edges compared to the model using precomputed handcrafted features. For each scenario, ten agents of each method are trained on the other four scenarios. The performance is evaluated on both the four scenarios seen during training and the held out evaluation scenario. This approach is motivated by the generalization experiments of Kurzer *et al.* [1]. We follow current best practice [26] for reliable evaluation of RL-methods and report the interquartile mean (IQM) over all scenarios combined, which reduces uncertainty and eliminates outlier runs. The 95% confidence interval is estimated using stratified bootstrapping [26] with 50000 iterations, see Table I for the results. Our proposed architecture outperforms the precomputed edge features in both metrics on the training and evaluation performance, achieving 1.19 pp better SR and 1.18 pp better ETR on the training sets. The improvements on the training scenarios are significant with $p \leq 0.05$, showing that the features learned by our proposed architecture are more powerful in capturing the vehicle relations than the precomputed relative distance and velocity vector. Differences between *S1* to *S5* are large, with only *S3* containing a left turn, and *S5* containing a right turn. Therefore, generalizing to completely new behavior when withholding one scenario during training is challenging. The comparative method achieves 81.67% SR and ours completes 83.17% of episodes successfully. We refrain from carrying out more repetitions to improve the p-values on the held out evaluation set due to the high environmental impact and cost of a large amount of training runs in reinforcement learning.

We test more gradual changes by rotating the ego vehicle's road in *S4* counter-clockwise, thus increasing the angle between the merging roads. All previously trained agents are evaluated on the changed road geometry. The

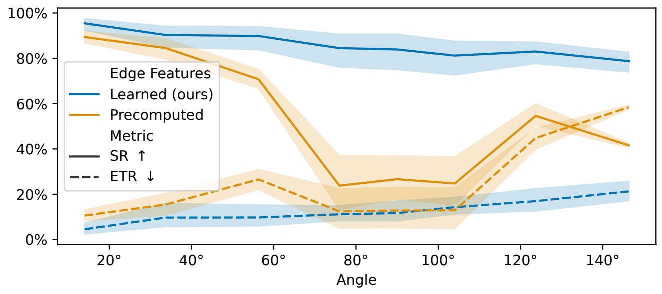


Fig. 4: Increasing the merge angle in *S4* causes a distribution shift of observations. All agents are trained on the original *S4* with an angle of 15. Using precomputed edge features leads to a collapse of behavior, whereas our learned features generalize well to the unseen geometry.

results are presented in Fig. 4. The edge features learned by our proposed method generalize well, with SR decreasing slightly to 80% as the road geometry moves further away from the training distribution. In contrast, the performance of the agents with handcrafted precomputed features decreases significantly, completing less than 30% of the episodes from 80 to 100. As the angle increases further ETR surpasses SR. While more robust handcrafted features for generalizing on intersections were proposed [1], [4], [5], this still highlights the natural ability of our approach to implicitly learn well-generalizing edge features.

Next, we analyze whether our architecture learns right-of-way semantics from the representation. We simulate episodes on all scenarios with two right-of-way variants (*S1* to *S4*) until we have 2000 relations per environment to other vehicles approaching the junction in front of the ego vehicle, thus 16 000 relations total. For each trained architecture from the main experiment, we train a single neuron on top of the edge encoder using binary cross-entropy to predict the right-of-way priority. Using a batch size of 512 only 1000 training steps are performed with the weights of the edge encoder frozen. Our learned vehicle-to-vehicle edge encoders achieve a near-perfect mean prediction accuracy of 99.5%, the relative feature-based encoders 71.12%. These results are significant with $p \leq 0.01$, showing our architectures ability to extract semantically meaningful edge features.

C. Ablations and Variations

We study the influence of the graph convolution and recurrent network used. The LSTM is substituted with an Elman-RNN [27] and the GATv2 graph convolution with the standard GAT convolution [28] as well as the SAGE operator [29]. The edge encoder is also completely removed (*No Edges*). Three agents of each architecture are trained holding out *S1*. The IQM of all metrics is reported in Table III. Removing the edge encoder from the architecture impairs the SR and ETR drastically, showing the necessity of the good vehicle-to-vehicle edges learned by our method for high performance. All other configurations perform well, with statistical noise causing some variance. This highlights the robustness of our framework to the selection of these operators. For the edge encoder, we attribute this to the short length of the sequences fed to the RNN, thus an LSTM is not required to prevent vanishing gradients. As the edge encoder encodes a relation between ego and observed vehicle, which is naturally a relative feature, the dynamic attention of GATv2 is not strictly required and the framework performs well using less attentive graph convolution operators.

Finally, we analyze the influence of the observed vehicles. The reduction of vehicles through the flood fill from Sect. III-C to increase the inference throughput is compared to including all vehicles within the vision radius. The results are shown in Table IV. The policy learned with all vehicles observable performs similar to the one with our vehicle selection. Thus, our architecture also works well when having to attend to all vehicles. Limiting the vehicles to those closest to the ego vehicle does not benefit the agent through introduction of prior knowledge, which would be undesirable, while increasing the inference throughput.

TABLE III: The graph convolution and recurrent network are substituted with different methods. Our framework is robust regarding these changes. Removing the edge encoder completely (*No Edges*) leads to significantly lower performance.

Edge Encoding	Graph Convolution	Training SR ↑	Training ETR ↓	Evaluation SR ↑	Evaluation ETR ↓
No Edges	GATv2	56.75 %	16.75 %	69.30 %	27.25 %
RNN	SAGE	95.60 %	4.40 %	85.70 %	14.30 %
RNN	GAT	94.75 %	5.25 %	87.00 %	13.00 %
RNN	GATv2	94.55 %	5.45 %	89.20 %	10.40 %
LSTM	SAGE	96.10 %	3.90 %	88.15 %	11.85 %
LSTM	GAT	95.25 %	4.75 %	85.65 %	14.35 %
LSTM	GATv2	95.30 %	4.70 %	88.65 %	11.35 %

TABLE IV: When removing the flood fill vehicle selection from Sect. III-C, performance is comparably good, but inference time increases slightly.

Vehicles Observed	Training SR ↑	Training ETR ↓	Evaluation SR ↑	Evaluation ETR ↓	Inference Time ↓
Flood fill	95.94 %	4.06 %	89.13 %	10.80 %	0.934 ms
All	95.86 %	4.14 %	89.74 %	10.26 %	0.975 ms

V. CONCLUSIONS

In this work, we presented a novel reinforcement learning framework for automated driving operating on high-level heterogeneous scene graphs representing road topology while requiring minimal feature engineering and prior knowledge. The experiments showed that our architecture generalizes well to unseen road topology, learns semantic rules like right-of-way, and outperforms precomputed vehicle-to-vehicle features on intersection scenarios. Relations that previously were expressed using handcrafted features depending on the application, in particular vehicle-to-vehicle edge features, can be implicitly learned with our method. We consider this an important step in moving towards more holistic high-level driving using graph networks. In the future, we want to extend our approach to more scenarios than intersections and plan to incorporate the vehicle-to-vehicle relations between other vehicles.

References

[1] K. Kurzer, P. Schörner, A. Albers, H. Thomsen, K. Daaboul, and J. M. Zöllner, "Generalizing decision making for automated driving with an invariant environment representation using deep reinforcement learning," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 994–1000.

[2] S. Kai, B. Wang, D. Chen, J. Hao, H. Zhang, and W. Liu, "A multi-task reinforcement learning approach for navigating unsignalized intersections," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1583–1588.

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

[4] M. Klimke, J. Gerigk, B. Völz, and M. Buchholz, "An enhanced graph representation for machine learning based automatic intersection management," *ArXiv*, vol. abs/2207.08655, 2022.

[5] M. Klimke, B. Völz, and M. Buchholz, "Cooperative behavior planning for automated driving using graph neural networks," *2022 IEEE Intelligent Vehicles Symposium (IV)*, pp. 167–174, 2022.

[6] X. Ma, J. Li, M. J. Kochenderfer, D. Isele, and K. Fujimura, "Reinforcement learning for autonomous driving with latent state inference and spatial-temporal relationships," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6064–6071, 2021.

[7] M. Huegle, G. Kalweit, B. Mirchevska, M. Werling, and J. Boedecker, "Dynamic input for deep reinforcement learning in autonomous driving," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7566–7573.

[8] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *2018 21st international conference on intelligent transportation systems (ITSC)*. IEEE, 2018, pp. 2575–2582.

[9] M. Huegle, G. Kalweit, M. Werling, and J. Boedecker, "Deep surrogate q-learning for autonomous driving," *2022 International Conference on Robotics and Automation (ICRA)*, pp. 1578–1584, 2022.

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

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] S. Brody, U. Alon, and E. Yahav, "How attentive are graph attention networks?" in *International Conference on Learning Representations*, 2022.

[13] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[14] P. Wolf, K. Kurzer, T. Wingert, F. Kuhnt, and J. M. Zollner, "Adaptive behavior generation for autonomous driving using deep reinforcement learning with compact semantic states," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 993–1000.

[15] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Póczos, R. Salakhutdinov, and A. Smola, "Deep sets," in *NIPS*, 2017.

[16] S. Triest, A. Villaflor, and J. M. Dolan, "Learning highway ramp merging via reinforcement learning with temporally-extended actions," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1595–1600.

[17] A. Alizadeh, M. Moghadam, Y. Bicer, N. K. Ure, U. Yavas, and C. Kurtulus, "Automated lane change decision making using deep reinforcement learning in dynamic and uncertain highway environment," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 1399–1404.

[18] A. Baheri, S. Nageshrao, H. E. Tseng, I. Kolmanovsky, A. Girard, and D. Filev, "Deep reinforcement learning with enhanced safety for autonomous highway driving," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1550–1555.

[19] U. Yavas, T. Kumbasar, and N. K. Ure, "A new approach for tactical decision making in lane changing: Sample efficient deep q learning with a safety feedback reward," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1156–1161.

[20] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, and C. Schmid, "Vectornet: Encoding hd maps and agent dynamics from vectorized representation," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11 522–11 530, 2020.

[21] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[24] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[25] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *CoRR*, vol. abs/1511.05952, 2016.

[26] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. G. Bellemare, "Deep reinforcement learning at the edge of the statistical precipice," in *NeurIPS*, 2021.

[27] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, pp. 179–211, 1990.

[28] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2018.

[29] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.