DQ-GAT: Towards Safe and Efficient Autonomous Driving With Deep Q-Learning and Graph Attention Networks

Peide Cai, Hengli Wang, *Graduate Student Member, IEEE*, Yuxiang Sun, *Member, IEEE*, and Ming Liu, *Senior Member, IEEE*

***Abstract*— Autonomous driving in multi-agent dynamic traffic scenarios is challenging: the behaviors of road users are uncertain and are hard to model explicitly, and the ego-vehicle should apply complicated negotiation skills with them, such as yielding, merging and taking turns, to achieve both safe and efficient driving in various settings. Traditional planning methods are largely rule-based and scale poorly in these complex dynamic scenarios, often leading to reactive or even overly conservative behaviors. Therefore, they require tedious human efforts to maintain workability. Recently, deep learning-based methods have shown promising results with better generalization capability but less hand engineering efforts. However, they are either implemented with supervised imitation learning (IL), which suffers from dataset bias and distribution mismatch issues, or are trained with deep reinforcement learning (DRL) but focus on one specific traffic scenario. In this work, we propose DQ-GAT to achieve scalable and proactive autonomous driving, where graph attention-based networks are used to implicitly model interactions, and deep Q-learning is employed to train the network end-to-end in an unsupervised manner. Extensive experiments in a high-fidelity driving simulator show that our method achieves higher success rates than previous learning-based methods and a traditional rule-based method, and better trades off safety and efficiency in both seen and unseen scenarios. Moreover, qualitative results on a trajectory dataset indicate that our learned policy can be transferred to the real world for practical applications with real-time speeds. Demonstration videos are available at https://caipeide.github.io/dq-gat/.**

***Index Terms*— Autonomous driving, reinforcement learning, motion planning, graph neural networks.**

Manuscript received 11 August 2021; revised 31 March 2022; accepted 2 June 2022. Date of publication 7 July 2022; date of current version 7 November 2022. The work of Ming Liu was supported in part by the Zhongshan Science and Technology Bureau Fund, under Project 2020AG002; in part by the Foshan-HKUST under Project FSUST20-SHCIRI06C; and in part by the Guangdong Basic and Applied Basic Research Foundation under Project 2020A0505090008. The Associate Editor for this article was S. Sun. *(Corresponding author: Ming Liu.)*

Peide Cai and Hengli Wang are with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong, SAR, China (e-mail: pcaiaa@connect.ust.hk; hwangdf@connect.ust.hk).

Yuxiang Sun is with the Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hong Kong, SAR, China (e-mail: yx.sun@polyu.edu.hk; sun.yuxiang@outlook.com).

Ming Liu is with the Thrust of Robotics and Autonomous Systems, The Hong Kong University of Science and Technology (Guangzhou), Nansha, Guangzhou, Guangdong 511400, China, also with the Department of ECE, The Hong Kong University of Science and Technology, Hong Kong, SAR, China, and also with the HKUST Shenzhen-Hong Kong Collaborative Innovation Research Institute, Futian, Shenzhen 518045, China (e-mail: eelium@ust.hk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TITS.2022.3184990, provided by the authors.

Digital Object Identifier 10.1109/TITS.2022.3184990

I. Introduction

AUTONOMOUS driving (AD) technology has made substantial progress in the last decade, moving from academic research [1] to industrial practice [2]. Nevertheless, reliable and robust autonomous driving in urban areas remains an important challenge, primarily due to the following reasons: 1) There are diverse road topologies and structures (e.g., roundabouts, multi-lane streets, and intersections) with different traffic densities to consider [3]; 2) The complex and coupled interactions among multiple road agents are hard to model explicitly [4]; 3) The agent vehicle needs to intelligently make decisions in these uncertain scenarios to properly balance two contradictory driving objectives: safety (collision avoidance) and efficiency (time to goal).

Traditional planning methods are mainly based on hand-engineered heuristics [1], such as finite state machine (FSM) [5], [6]. However, they are usually designed for a narrow set of particular use-cases, and require extremely tedious human efforts to maintain a rule database so that safety can be ensured [7]. Moreover, new problems may arise as the rules increase, for example, how to solve new situations without forgetting the old ones, and how to balance cost functions in countless hard-to-model scenarios with conflicting objectives (e.g., safety and efficiency). Therefore, rule-based methods often lead to unnatural driving behaviors, or they completely fail in unexpected edge cases [8]. For example, autonomous vehicles may slow down and stop in highly interactive scenarios (as shown in Fig. 1) to ensure safety [9], known as the *freezing robot problem*. However, such an overly conservative solution also causes confusion to other road users and even leads to traffic accidents.[1] Due to the these limitations, traditional rule-based methods are "*not robust to a varied world*", even according to their own authors [1].

In recent years, as an alternative, deep learning has advanced AD technology to a great extent. The ability to *learn and self-optimize* its behavior from data alleviates the laborious engineering maintenance required to model all foreseeable

[1]In California in 2018, 86% of autonomous vehicle crashes were caused by other cars, resulting from the conservatism of the autonomous vehicles [10].

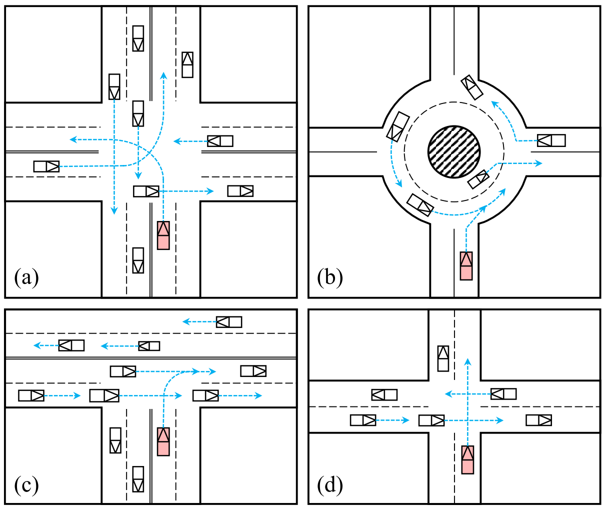


Fig. 1. Sample unsignalized driving scenarios considered in this work: (a) unprotected left turns, (b) roundabouts, (c) merging, and (d) crossing. Rule-based methods have difficulties in tuning for complex interactions, often leading to reactive or even overly conservative behaviors. In this work we aim to train DRL-based agents that can act proactively amidst other vehicles to better trade off safety and efficiency of autonomous driving. The red and white boxes denote the ego and other vehicles, respectively.

scenarios, making a deep driving model well suited to AD problems in high-dimensional, nonlinear and dynamic environments [11]–[16]. Most of these are implemented with supervised imitation learning (IL), which can efficiently extract driving knowledge from human demonstrations. However, this approach suffers from dataset bias [14] and distribution mismatch (a.k.a., *covariate shift*) [17] problems. Another learning-based approach is deep reinforcement learning (DRL) [17]–[21], where the agent proactively *interacts* with the environment and learns knowledge from trial-and-error. However, current DRL-based methods have not been well designed for scalable AD in a uniform setup. Particularly, most work focuses on a specific network design tailored for scattered traffic scenarios such as single lanes [19], specific intersections [20] and roundabouts [21], all with no [17], [18] or low-level traffic dynamics, leaving it unclear if these models can generalize to more complex or unseen environments.

In this paper we propose DQ-GAT for autonomous driving in complex and dynamic scenarios. To avoid a model tailored for specific scenarios, we first design a graph attention-based network, which can process heterogeneous traffic information for generic driving scenarios. Afterwards, we design rewards and extend the model-free DRL method dueling double deep Q-learning (D3QN) into an asynchronous version. Finally, the network is trained at scale to adaptively balance safety and efficiency (i.e., *ensuring driving safety while improving the efficiency as much as possible*) by proactively interacting with the environments.

This paper significantly extends our recent conference paper [22], and the improvements are multi-fold. First, we extend the original training pipeline from supervised learning to reinforcement learning, and demonstrate its superiority quantitatively in this work. Second, we conduct more thorough experiments with related works on DRL, IL and rule-based methods and provide more in-depth discussions. Moreover, we examine the zero-shot generalization performance and runtime speeds of our model in the real-world traffic dataset.

The main contributions are summarized as follows:

1) We propose a novel graph attention-based network architecture to encode heterogeneous traffic information (i.e., road structures and vehicle states) and implicitly model inter-vehicle interactions in generic driving scenarios.
2) We develop a parallel DRL framework to provide asynchronous and scalable training of the proposed network for autonomous driving without relying on supervisions.
3) We conduct extensive evaluation in a high-fidelity driving simulator and show that our method balances safety and efficiency better than previous learning-based and rule-based methods, in both training and unseen scenarios.
4) We show that our method can achieve sim-to-real policy transfer in an interaction-intensive real-world dataset, where real-time performance is also satisfied.

II. Related Work

A. Imitation Learning

IL is the dominant approach for learning-based AD due to its sample efficiency, where a deep neural network is trained to mimic expert driving behaviors using supervised learning [11]–[16]. Based on the collected demonstrations (i.e., observation-action pairs), the model compares the error between its prediction and the labelled data from human drivers, then adjusts its weights using gradient decent.

1) End-to-End Driving: With the powerful representation capabilities of deep neural networks, *end-to-end* driving approaches [11]–[16] directly take as input the raw sensor readings (e.g., LiDAR point clouds and camera images) to output control commands or future trajectories. For example, Codevilla *et al.* [13] proposed a conditional imitation learning approach that splits the network into multiple branches for discrete tasks such as *follow lane* and *turn left/right*. Follow-up works include [12] and [15]. However, these methods cannot handle complex road topologies such as multi-lane streets or roundabouts. Recently, Cai *et al.* [11] used global routes as direction to achieve robust end-to-end navigation in complex dynamic environments with multi-modal sensor fusion. However, as with previous methods, the learned policy is reactive without efficient interaction with other road users.

To summarize, it is quite challenging to learn a direct mapping from high dimensional sensory observations to low dimensional motion plans, as the end-to-end approaches conflate two aspects of driving: *learning to see* and *learning to act*. Therefore, they suffer from the domain gap problem in the perception stage, which leads to poor generalization performance in new environments [22].

2) Learning to Drive by Semantic Abstraction: Recently, another stream of work has arisen that uses *semantic* information to learn driving policies (e.g., HD maps [3], brid-eye-views (BEVs) [23], and occupancy maps [24]). Compared to redundant sensory observations, this semantic information is a kind of concise and informative abstraction of perceptual results, and has better environmental consistency. These properties help the training process to focus on *learning to act*, which is more efficient and generalizable. For example, the policy network of [24] takes as input hybrid features composed of the roadmap, traffic lights, route plans and dynamic objects to produce waypoints to follow. This has the advantage of helping the network to learn meaningful contextual cues behind the human driver's action and achieve more complex driving behaviors. However, [24] mainly shows its results on offline *logged data*, and only performs closed-loop evaluation in simple environments with at most two obstacles. Such a problem also exists in other similar works [23]. According to [25], the driving performance can vary significantly between offline open-loop and online closed-loop tests, and the latter can better reveal the driving quality. In this work, the observation is similar to those of [23] and [24] but, differently, we focus more on dynamic, interactive and large-scale *closed-loop* driving performance.

3) Limitations: Although the paradigm of IL is appealing, there still exist three major shortcomings that prevent IL from being applied to broader applications: 1) The cost of human driving data collection on a large scale can be prohibitive [20]; 2) IL approaches are particularly sensitive to the *dataset bias* issue, as different drivers, or even the same driver in different moods, might have different driving preferences, thus the learning objective might be dominated by the main modes in the training data [14]; 3) IL performs well for states that are covered by the training distribution, but it generalizes poorly to new states due to compounding action errors [26], which is also referred to as *distribution mismatch*.

B. Deep Reinforcement Learning

DRL is another popular training paradigm, where the agent interacts with the environment and gains knowledge through trial-and-error, aiming to maximize the sum of expected future rewards [17]–[21]. Therefore, it does not require expert labels and thus eliminates the dataset bias issue associated with IL. Furthermore, its online training paradigm also allows avoiding the distribution mismatch problem. Due to these advantages, DRL has shown promising results in various areas in decision making. For example, Wu *et al.* [27] proposed a triplet-average deep deterministic (TADD) policy gradient algorithm to reduce the estimation bias, and it achieves superior performance in the OpenAI gym environment. In addition, Dong *et al.* [28] improved the performance of visual object tracking by dynamically optimizing its hyperparameters for changing sequences with deep Q-learning.

1) Applications and Limitations: DRL has been applied to learn autonomous driving policies [17]–[21]. For example, [19] trained an end-to-end policy for lane-following tasks on a slow vehicle, and [18] achieved high-speed autonomous vehicle racing using model-free DRL methods. However, these works only consider static environments without interaction with other vehicles. By contrast, [20], [21], [29] and [30] consider dynamic traffic scenarios and thus they are more applicable for urban driving. However, current methods have not been well designed for scalable AD in a uniform setup. Particularly, most works focus on specialized network design (e.g., input representations and reward) tailored for scattered traffic scenarios such as intersections [20], roundabouts [21], merging [29] and highway [31] scenarios, where one model cannot generalize across different scenarios due to the varied requirements of observation space. For example, [31] used a list of vehicle state vectors, such as position and velocity, to depict the environment. Although this representation is concise and precise, it lacks the information of driving contexts and would fail in complex scenarios such as multi-lane intersections. For example, in Fig. 1, the vehicles should drive by obeying traffic rules according to the road structures. By contrast, processed BEVs [30] are more suitable for context-aware driving, where all necessary information, including static road and dynamic vehicles, can be rasterized into pixels and jointly processed with convolution neural netowrks (CNNs). However, the drawback of this method is the information loss during rasterization. In this work, we combine the benefits of both pixel- and state-based methods to train agents that can drive in diverse urban scenarios.

C. Graph Representation Learning

Many real-world problems can be modeled with graphs where the nodes contain features of different entities, and edges represent interactions between entities. For example, in the field of video object segmentation, Lu *et al.* used graphs to store frames as nodes and capture cross-frame correlations by edges [32]. A challenge in learning on graphs is to find an effective way to get a meaningful aggregated feature representation to facilitate downstream tasks. Recently, graph neural networks (GNNs) have been shown to be effective in many applications such as social networks, personalized recommendation, video object segmentation [32] and detection [33]. In this area, graph convolutional networks (GCNs) generalize the 2D convolution on grids to graph-structured data. When training a GCN, a fixed adjacency matrix is commonly adopted to aggregate the feature information of neighboring nodes. On the other hand, a graph attention network (GAT) [34] is a GCN variant that aggregates node information with weights learned in a self-attention mechanism. Such adaptiveness of GATs makes them commonly more effective than GCNs in graph representation learning.

1) Applications and Limitations: Recently, graph neural networks (GNNs) have also been shown to be effective in robotics, such as in crowd robot navigation [4], [35], where the robot can navigate safely in human crowds of various sizes. However, these works totally neglect the environmental structures, which is not particularly important for indoor robot navigation in restricted areas, but is non-negligible for outdoor driving problems, as introduced in Sec. II-B. In this work, we borrow the idea of the GNN to model the traffic scenes as graphs, and use a context-aware GAT for autonomous driving in dense traffic.

III. Preliminaries

In this paper, a value-based reinforcement learning algorithm called dueling double deep Q-learning (D3QN) [36],

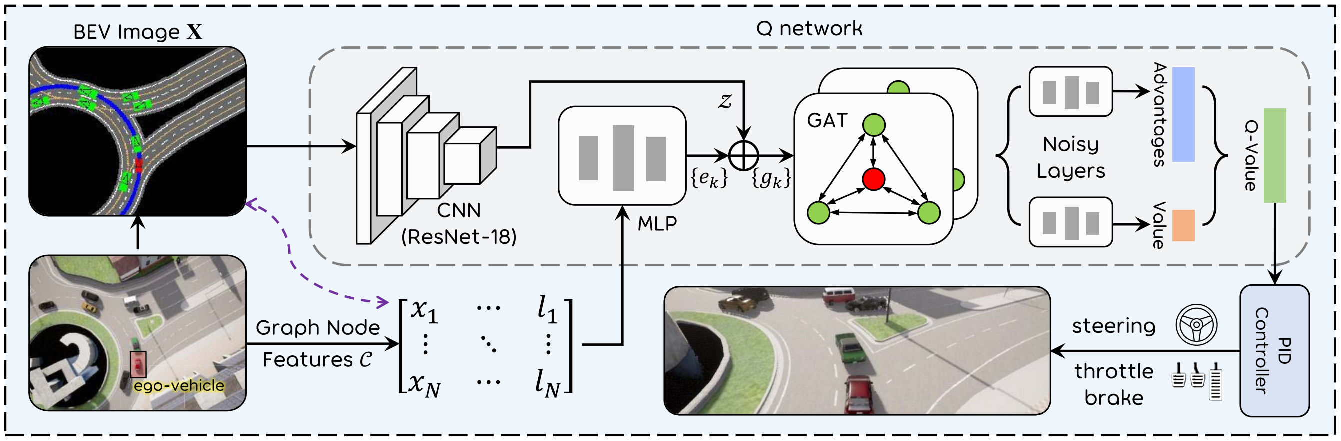


Fig. 2. Schematic overview of the proposed method DQ-GAT for autonomous driving. We assume the input information is accessible with a functioning perception system or with a vehicle-to-everything (V2X) module, and focus on *learning to act* of autonomous vehicles. The semantic BEV image and vehicle states (locations, velocities, etc.) are first encoded respectively by a CNN backbone (ResNet-18) and MLP layers. The derived feature vectors are then concatenated to construct a heterogenous scene-level graph $\{g_k\}$, which is further processed by a two-layer GAT and *noisy* MLP layers (see Sec. III-.5) to compute Q-values for controlling the ego-vehicle.

is applied to autonomous driving.[2] In the following, we introduce the notation, terminology and algorithm for the system modeling and training.

2) Markov Decision Process (MDP): The MDP process is the theoretical basis of reinforcement learning, and it can be formulated as a tuple containing five elements: $<\mathcal{S}, \mathcal{A}, R, f, \gamma>$, which denote the state space, action space, immediate reward, state transition model and the discount factor, respectively. Within this formulation, the agent interacts with environment and learns the policy π by maximizing the expected discounted return $R_t = \sum_{\tau=t}^{\infty} \gamma^{\tau-t} r_\tau$, where $\gamma \in [0,1]$ trades-off the importance of immediate and future rewards.

Given a policy π, the action-value (Q-value) of a state-action pair is defined as

$$Q_\pi(s_t, a_t) = \mathbb{E}[R_t \mid s_t, a_t, \pi], \tag{1}$$

which can be computed using the Bellman equation:

$$Q_\pi(s_t, a_t) = \mathbb{E}[r_t + \gamma \mathbb{E}[Q_\pi(s_{t+1}, a_{t+1})] \mid s_t, a_t, \pi]. \tag{2}$$

Finally, the optimal Q-value function can be written as

$$Q^*(s_t, a_t) = \mathbb{E}[r_t + \gamma \max_{a_{t+1}} Q^*(s_{t+1}, a_{t+1}) \mid s_t, a_t]. \tag{3}$$

3) Double Deep Q-Learning: It can be seen that once Eq. (3) is computed, we can choose the optimal action a_t^* with the largest Q-value to execute at state s_t. However, traditional tabular methods cannot scale to large state spaces, like images. In deep Q-learning, the $Q^*(s,a)$ in Eq. (3) is approximated by a deep neural network $Q(s,a;\theta)$ with parameters θ. We use *double* deep Q-learning [38] to estimate this network, and optimize the following sequence of loss function at iteration i based on the temporal-difference (TD) error:

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}[(y_i - Q(s,a;\theta_i))^2], \tag{4}$$

$$y_i = r + \gamma \hat{Q}(s', \arg\max_{a'} Q(s', a'; \theta_i); \theta^-), \tag{5}$$

[2]In this work we choose to use value-based DRL as it has shown advanced intelligence beyond the level of human beings in certain areas like Go [37]. In addition, it is also known to be more data efficient than the policy-based method, which is another family of DRL.

where $\hat{Q}$ denotes the *target network* with paramters θ^-, which are updated by copying the weights of $Q(s,a;\theta)$ every T gradient steps and are frozen in other intervals. In this work, Q and $\hat{Q}$ share the same CNN encoder, and we set $T = 1500$.

4) Dueling Network Architecture: Based on the double DQN algorithm introduced above, Wang *et al.* [36] further proposed the *dueling* network architecture for the Q network, named D3QN, where two streams of sub-networks are built to compute the state value $V_\pi(s)$ (scalar) and advantage functions $A_\pi(s,a)$ (vector of $|\mathcal{A}|$-dimensional) separately, as shown in Fig. 2. These two branches are finally combined to compute the action values:

$$Q(s,a;\theta) = V(s;\theta) + (A(s,a;\theta) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a';\theta)). \tag{6}$$

The decoupling operation of value and advantage in deep Q-networks has shown dramatic improvements in the challenging Atari gaming domain in terms of task performance and training speed. In this work, we extend this method into the area of autonomous driving, which will be methodologically introduced in the next section.

5) Noisy Networks for Exploration: Classical DRL methods use ϵ-greedy strategies to randomly perturb the agent's policy and induce novel behaviors for exploration. However, these methods require hyperparameter tuning, and the induced *local* dithering perturbations make it hard to generate diverse behaviors for efficient exploration. Therefore, we adopt the idea of *noisy nets* [39] for exploration in this work. This uses a noisy linear layer that combines a deterministic and noisy stream:

$$\boldsymbol{y} = (\boldsymbol{b} + \mathbf{W}\boldsymbol{x}) + \left(\boldsymbol{b}_{\text{noisy}} \odot \epsilon^b + \left(\mathbf{W}_{\text{noisy}} \odot \epsilon^w\right)\boldsymbol{x}\right), \tag{7}$$

where the parameters $\boldsymbol{b}, \mathbf{W}, \boldsymbol{b}_{\text{noisy}}$ and $\mathbf{W}_{\text{noisy}}$ are learnable, while ϵ^b and ϵ^w are random variables. In this way, the amount of noise injected in the network is tuned automatically by the RL algorithm, allowing state-conditioned and self-annealing exploration. During testing, $\boldsymbol{b}_{\text{noisy}}$ and $\mathbf{W}_{\text{noisy}}$ are set to zero for stable policy deployment.

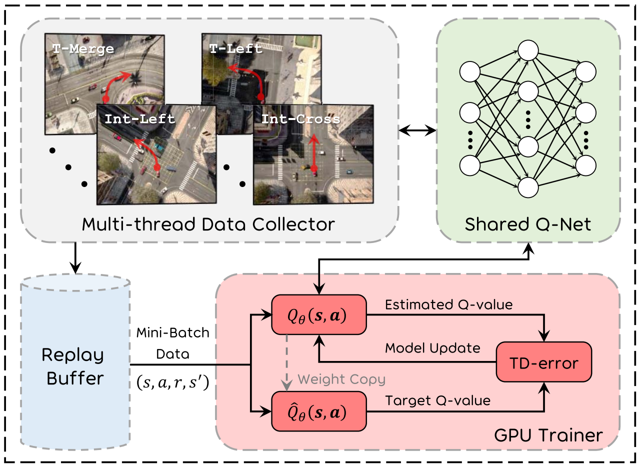


Fig. 3. Overview of our training system design. The training process is divided into two iterative parts until convergence: *exploration* and *model update*. At the exploration stage, multiple agents sharing the same Q-network collect experiences in their separate CARLA threads. Different threads are synchronized using the message passing interface (MPI), and their transition tuples (s, a, r, s') are pushed into the replay buffer $\mathcal{D}$ in parallel, enabling high throughput. At the training stage, mini-batches of transition data are sampled from $\mathcal{D}$ to calculate the TD-loss (Eq. 4) to update the Q-network.

IV. Methodology

The structure of the proposed DQ-GAT for autonomous driving is shown in Fig. 2. The goal is to drive safely and efficiently in complex and dynamic urban environments with different road layouts and other vehicles.

A. Training Scenarios

Our system is trained and evaluated in the open-source CARLA simulator (v0.9.12) [40] since it possesses abundant vehicle models and maps close to the real world. We choose four different *unsignalized* traffic scenarios, where the agent has to drive safely and efficiently according to the intent of other vehicles, being neither too conservative nor too aggressive. These scenarios are shown in Fig. 3. The `T-Merge` scenario involves making a right turn for lane merging at a T-shaped junction. The `T-Left` and `Int-Left` scenario involves making an unprotected left turn at a T-shaped junction and a four-lane intersection, respectively. `Int-Cross` involves driving straight to cross the four-lane intersection.

We use the AI engine of CARLA to form realistic and dynamic traffic flows with several random properties, in terms of vehicle types (cars, big trucks, ambulance, etc.), destinations, densities and speeds. This randomness provides a large state space to explore, which can produce generic driving policies and avoid overfitting the policy to a specific case. The simulation step is set to 0.1 seconds, meaning the control frequency is 10 Hz for all our experiments.

B. Semantic Abstraction of Driving Scenes

In order to learn good driving policies, we use semantic BEV images as the representation of driving scenes to reduce the dimensionality and redundency of raw sensory data. Furthermore, with such representation, there is no domain difference between the simulation and real world, thus the policy transfer problem [41] can be alleviated. Specifically, we rasterize different semantic elements (e.g., lane marking and obstacles) into RGB channels to form a concise and informative scene representation. As shown in Fig. 2, our BEV input is composed of the following two parts: 1) *High-definition (HD) map*: The HD map contains the drivable area, lane markings, and the route to follow. Leveraging map information to learn driving policies is very helpful because it provides valuable structural priors on the motion of surrounding road agents. For example, vehicles normally drive on lanes rather than on sidewalks, and they should not cross solid lane markings; 2) *Road vehicles*: We render the ego and other vehicles on the HD map to provide more spatial information.

In this work, our region of interest is W=70 m wide (35 m to each side of the ego-vehicle) and H=50 m long (35 m in front and 15 m behind the ego-vehicle). The image resolution is set to 0.25 m/pixel, which finally results in a binary BEV input $\mathbf{X}$ of size $200 \times 280 \times 3$, anchored at the ego-vehicle's current position. We use the CNN backbone ResNet-18 to project $\mathbf{X}$ into a lower-dimensional vector $z \in \mathbb{R}^{512}$ for further operation.

C. Graph Modeling of Driving Scenes

1) Network Architecture: As shown in Fig. 2, we use a GAT to model the interaction among road agents during driving, and it is composed of multiple graph layers. The input to the i-th layer is a set of node features, $\{\boldsymbol{h}_1^i, \boldsymbol{h}_2^i, \ldots, \boldsymbol{h}_N^i\}$, $\boldsymbol{h}_k^i \in \mathbb{R}^{F^i}$, where N is the number of nodes (agents, including the ego-vehicle), and F^i is the dimensions of features in each node. Then, the information of each node k is propagated to the neighboring nodes $\mathcal{N}_k$ and is used to update the node features via a self-attention mechanism, which produces the output of the layer:

$$\boldsymbol{h}_k^{i+1} = \sigma\Big(\sum_{j \in \mathcal{N}_k} \alpha_{kj}(\boldsymbol{h}_k^i, \ \boldsymbol{h}_j^i)\mathbf{W}\boldsymbol{h}_j^i\Big), \tag{8}$$

where $\sigma(\cdot)$ is the ReLU activation function, $\mathbf{W} \in \mathbb{R}^{F^{i+1} \times F^i}$ is a shared weight matrix to be applied to each node for expressive feature transformation, $\alpha_{kj}(\cdot, \cdot)$ means the importance of node j to node k, which is the normalized attention coefficients computed with shared weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F^{i+1}}$:

$$\alpha_{kj}(\boldsymbol{h}_k^i, \ \boldsymbol{h}_j^i) = \frac{\exp(\sigma(\vec{\mathbf{a}}^T[\mathbf{W}\boldsymbol{h}_k^i || \mathbf{W}\boldsymbol{h}_j^i]))}{\sum_{m \in \mathcal{N}_k} \exp(\sigma(\vec{\mathbf{a}}^T[\mathbf{W}\boldsymbol{h}_k^i || \mathbf{W}\boldsymbol{h}_m^i]))}, \tag{9}$$

where $||$ represents the concatenation operation. Furthermore, we follow the *multi-head attention* method in [34] to stabilize the learning process. Specifically, S^i independent graph networks execute the transformation of (8) and their features are concatenated to produce the output of the i-th layer:

$$\boldsymbol{h}_k^{i+1} = \mathop{\Big\Vert}_{s=1}^{S^i} \sigma\Big(\sum_{j \in \mathcal{N}_k} \alpha_{kj}^s(\boldsymbol{h}_k^i, \ \boldsymbol{h}_j^i)\mathbf{W}^s\boldsymbol{h}_j^i\Big). \tag{10}$$

For the final layer, we employ *averaging* among multiple heads rather than concatenation.

2) Implementation Details: In this work, we adopt a two-layer GAT and set $S^1, S^2 = 4$, $F^1, F^2 = 256$. The input features $\mathcal{C}$ include motion state information for each node (road agent) in the ego-vehicle's local coordinates. For node $k \in \{1, 2, \ldots, N\}$, the input feature is a 10-dimensional vector:

$$\boldsymbol{c}_k = \{x, y, d, \psi, vx, vy, ax, ay, w, l\}, \tag{11}$$

which includes its location (x, y), distance to the ego-vehicle (d), yaw angle (ψ), velocity (vx, vy), acceleration (ax, ay) and size (width w and length l). Inspired by [4], we first pass each node state $\boldsymbol{c}_k \in \mathcal{C}$ through a multilayer perceptron (MLP) to produce a feature vector $\boldsymbol{e}_k \in \mathbb{R}^{128}$ for sufficient expressive power. For context-aware graph modeling, we then concatenate $\boldsymbol{e}_k$ with $\boldsymbol{z}$ derived from the CNN module introduced in Section IV-B to generate the mixed vector $\boldsymbol{g}_k$. Then, the set $\{\boldsymbol{g}_k\}$ is sent to the GAT to output the final aggregated feature $\boldsymbol{h}_k^o \in \mathbb{R}^{256}$, which represents the internal interactions on each node k. We are interested in the result $\boldsymbol{h}_1^o$ of the first node, which represents the influence on the ego-vehicle.

D. Driving Policy Training With D3QN

With the components defined above, the derived feature vector $\boldsymbol{h}_1^o$ is processed with two MLPs to generate the advantage functions and the state value, separately (as shown in Fig. 2). Finally, the Q values are computed based on Eq. (6). During deployment, the action with the highest Q value will be executed to control the vehicle. In the following, we introduce the implementation details of this part.

1) Action Space: In this work, the agent chooses among a set of target speeds in the action space $\mathcal{A} = \{0, 10, 20, 30, 40\}$ (*km/h*) to navigate the vehicle longitudinally. The chosen target speed is translated into throttle and brake actions based on a low-level PID controller. The steering control is implemented with another PID to track the route.

2) Reward Design: To enable *safe* autonomous driving, the reward is set to -50 as punishment for collision events, and to $v/40 \in [0, 1]$ elsewhere to stimulate driving *efficiency*, where v is the speed of the agent vehicle in *km/h*.

3) Asynchronous Training: Inspired by [41], we extend the original D3QN algorithm to an *asynchronous* version with many experience collection threads working in parallel. Each thread randomly chooses a scenario to simulate for every new episode. In this way, the experience generation is decoupled from the parameter learning, which can provide higher throughput and thus improve the training efficiency. Moreover, interacting with different environments simultaneously also decorrelates the agent's data and makes the training process more stable [42]. To show the effectiveness of asynchronous training, we track the agent's mean reward during training using different numbers of threads. The results are shown in Fig. 4. It can be seen that the asynchronous DQ-GAT with 6 threads converges much faster than the single-thread version.

4) Overall Pipeline: The overall training pipeline of our DQ-GAT is shown in Fig. 3. The learning process is divided into two iterative parts: data collection and model update. During data collection, the agent selects and executes the actions according to the estimated Q values from $Q(s, a; \theta)$,

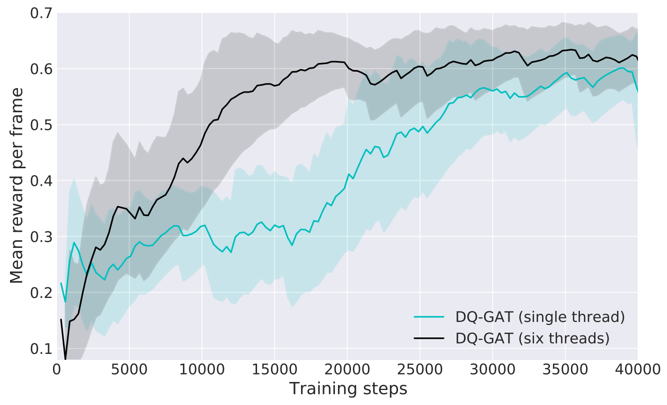


Fig. 4. Training performance of our method with different numbers of experience collection threads. The solid curve indicates the mean, and the shaded region corresponds to the standard deviation.

where the noisy MLP layers are activated for exploration. Related experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ are accumulated into the buffer $\mathcal{D} = \{e_1, e_2, \ldots, e_t\}$. During training, we first sample mini-batches of experiences from $\mathcal{D}$ using prioritized experience replay (PER) [43], then update the parameters of the Q network through stochastic gradient descent.

V. EXPERIMENTS

A. Training Setup

The replay buffer size is set to 500K. At each new episode, the ego-vehicle is placed in a random training scenario. The GPU trainer samples mini-batches of experiences every 4000 steps to update the Q-net for 300 rounds with the Adam optimizer. We further use a grid search to empirically find the best hyperparameters on learning rate $\alpha \in \{0.001, 0.0001\}$, reward discount $\gamma \in \{0.9, 0.99\}$ and batch size $N \in \{32, 128\}$. We found that setting $\alpha = 0.0001$, $\gamma = 0.99$ and $N = 128$ performs best in this work.

B. Training Performance With Different DRL Methods

We first compare our DQ-GAT with several other DRL-based methods for autonomous driving to verify the effectiveness of our model design.

- **GCN(U)**. This adopts a two-layer GCN to process the node features $\{\boldsymbol{g}_k\}$. We refer to the baseline *U-GCNRL* introduced in [4] and set the adjacency matrix of GCN with uniform weights.
- **GCN(D)**. It is similar to GCN(U) but uses distance-related weights in its adjacency matrix. It adopts a straightforward intuition that obstacles closer to the ego-vehicle should exert a stronger influence. This network follows the idea of *D-GCNRL* introduced in [4].
- **DenseBEV**. Following [30], we use the occupancy-grid style BEV image to form the observation, including information about the dynamic states of neighbouring vehicles and road layouts. Then, a CNN backbone Resnet-18 is used to handle such input.

The results are shown in Fig. 5. It can be seen that our method performs better than others with higher final rewards. First, DenseBEV has information loss when rasterizing the

TABLE I

EVALUATION RESULTS OF DIFFERENT MODELS IN BOTH TRAINING AND NEW SCENARIOS. ↑ MEANS LARGER NUMBERS ARE BETTER, ↓ MEANS SMALLER NUMBERS ARE BETTER. THE BOLD FONT HIGHLIGHTS THE BEST RESULTS IN EACH COLUMN

	Training Scenarios								New Scenarios			
Scenarios	`T-Left`		`T-Merge`		`Int-Cross`		`Int-Left`		`Five-Way`		`Roundabout`	
Models	S.R.	C.T.	S.R.	C.T.	S.R.	C.T.	S.R.	C.T.	S.R.	C.T.	S.R.	C.T.
regular traffic	(%) ↑	(*s*) ↓	(%) ↑	(*s*) ↓	(%) ↑	(*s*) ↓	(%) ↑	(*s*) ↓	(%) ↑	(*s*) ↓	(%) ↑	(*s*) ↓
IL-Conser.	84.00	9.64	96.00	8.62	78.33	8.49	83.33	9.27	80.00	6.89	55.67	30.52
IL-Aggr.	75.00	6.64	84.67	5.31	76.67	5.88	79.33	6.45	83.00	5.50	26.67	22.79
H-REIL [8]	79.67	6.77	94.33	5.48	81.67	**5.83**	82.33	6.53	85.00	5.52	28.33	22.16
FSM-TTC [44]	93.33	8.15	99.67	6.48	98.33	7.13	97.00	7.40	98.67	7.12	**87.67**	25.66
DQ-GAT*(ours)*	**99.00**	**6.34**	**100.00**	**4.99**	**100.00**	5.84	**98.33**	**6.15**	**99.67**	**5.40**	87.33	**21.15**
dense traffic												
IL-Conser.	77.00	11.12	92.67	11.99	69.33	8.40	77.67	11.16	65.33	7.59	21.00	35.57
IL-Aggr.	51.33	6.73	79.00	5.36	68.67	6.09	66.00	**6.49**	67.67	**5.39**	2.00	25.32
H-REIL [8]	70.00	6.85	91.33	5.71	69.00	**6.02**	76.00	6.54	81.33	5.63	1.00	**22.23**
FSM-TTC [44]	84.33	10.59	99.33	9.03	94.33	8.48	89.67	8.50	92.33	10.92	57.00	41.34
DQ-GAT*(ours)*	**96.67**	**6.72**	**99.67**	**5.24**	**99.00**	6.22	**98.33**	6.63	**99.67**	5.93	**77.67**	31.57

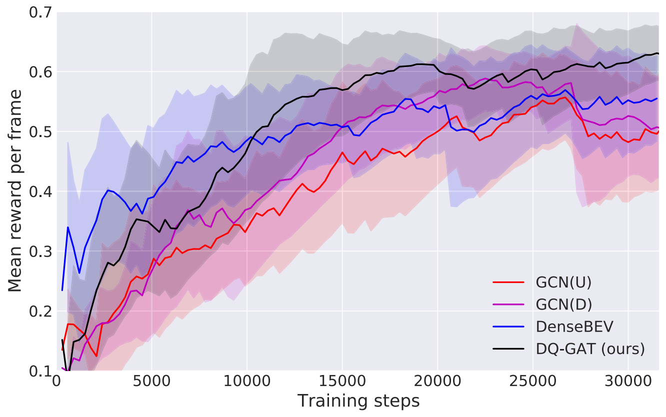


Fig. 5. Training performance with different DRL methods. The solid curve indicates the mean, and the shaded region means the standard deviation.

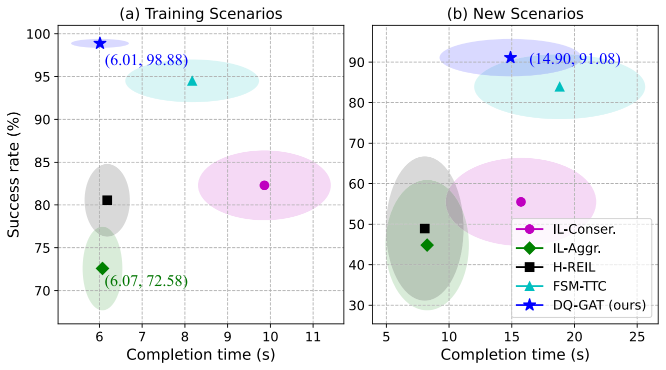


Fig. 6. The average success rate and the completion time for different methods on the (a) training scenarios and (b) unseen new scenarios. The solid marker indicates the mean value and the shaded area means the standard deviation.

dynamic states of vehicles into pixels. Second, the influences of road agents are not always equal (GCN(U)) or related to distance (GCN(D)). Our model addresses these problems through a heterogeneous graph attention network, where inter-vehicle influences are automatically learned and adjusted through data.

C. Evaluation Methods

In order to cover as many driving scenarios as possible to thoroughly evaluate different methods, we set two levels of traffic densities, namely *regular* and *dense*. Then, we run 300 trials on each scenario setup for each model with 200 random seeds *different from those in the training stage.*

1) Metrics: We use the following two metrics, which are averaged over all episodes, to measure the driving performance: (1) S.R. (success rate): An episode is considered to be successful if the agent reaches a certain goal without any collision. The episode will be recounted if there is a traffic jam. (2) C.T. (completion time): The average time cost for the *successful* trials. The *failed* trials where collision happens are not counted for this metric.

2) Baselines: We consider two driving modes in this paper, namely aggressive and conservative. The former favors efficiency over safety. For example, it drives fast (40 km/h) so as to reach the goal in minimal time, but tends to collide with other vehicles. By contrast, the conservative mode drives the car very cautiously (20∼30 km/h), and slows down or stops to avoid all potential accidents. Accordingly, we collect human driving data in the training scenarios per mode, leading to demonstrations $\mathcal{H}_{aggr}$ and $\mathcal{H}_{conser}$, respectively. Then, we compare DQ-GAT with the following policies:

- **IL-Conser**. An IL agent trained on $\mathcal{H}_{conser}$.
- **IL-Aggr**. An IL agent trained on $\mathcal{H}_{aggr}$.
- **H-REIL**. We follow [8] and train a high-level DRL-based mode switcher, which selects the low-level agent, i.e., IL-Conser. or IL-Aggr., every 0.5 s to balance safety and efficiency.
- **FSM-TTC**. This is a traditional rule-based method for crossing intersections [44]. It is implemented by an FSM with the time-to-collision (TTC) safety indicator [45].

D. Quantitative Analysis

The left column of Table I shows the evaluation results in four training scenarios. The average performance is shown in Fig. 6-(a). We have the following main findings:

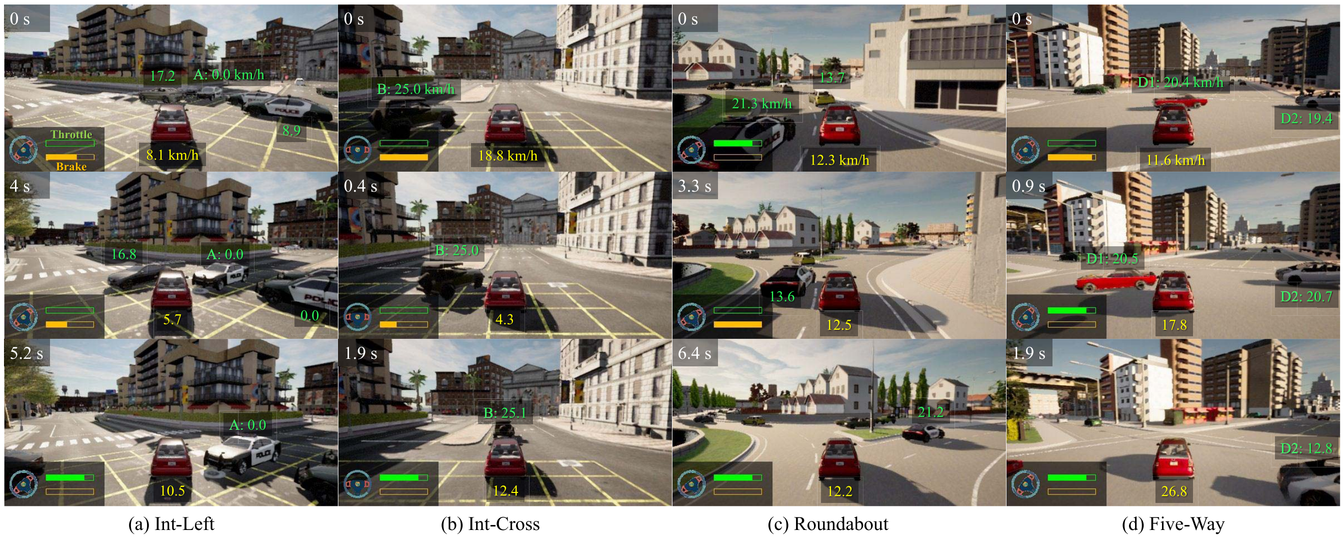


Fig. 7. Evaluation results of our DQ-GAT in the CARLA simulator. We show several driving clips with camera images in four scenarios, where `Roundabout` and `Five-Way` are unseen scenarios for the agent. We label the speed of key vehicles, and render the output control commands of the ego-vehicle for better understanding. The sample driving behaviors are: (a) and (d) creeping forward to safely and efficiently drive through the traffic when taking unprotected left turns at intersections; (b) timely slowing down to avoid an accident when an aggressive vehicle beside suddenly cuts into the lane of the ego-vehicle; and (c) yielding to the vehicle (which is departing the roundabout) on the left side for collision avoidance.

1) Since IL-Cons. favors more safety than IL-Aggr., it achieves higher success rates at the cost of efficiency. For example, in `T-Merge` with dense traffic, IL-Cons. reaches a success rate of 92.67%, higher than that of IL-Aggr. (79%). However, it requires much more time for the task (11.99 v.s. 5.36 s).

2) After training a high-level DRL-based mode switcher, the H-REIL model achieves success rates close to those of IL-Cons., and completion time close to those of IL-Aggr., as shown in Fig. 6-(a), meaning it better trades off safety and efficiency than the two base agents.

3) The rule-based method FSM-TTC achieves much higher success rates than the above methods, but it is not as efficient as IL-Aggr. with more task completion time. This is because FSM-TTC controls the vehicle in a *reactive* manner, and always waits for a gap to go.

4) As shown in Fig. 6-(a), on average, our DQ-GAT model not only achieves the highest success rate (98.88%), but also achieves a comparable completion time to the IL-Aggr. model (6.01 v.s. 6.07 s). These results demonstrate that DQ-GAT can better trade off safety and efficiency than previous methods. We accredit this improvement to our DRL-based pipeline, which is an unsupervised training framework enabling *end-to-end* optimization (compared with H-REIL) and self-learning (compared to IL-Coners., IL-Aggr. and FSM-TTC) of driving policies.

E. Qualitative Analysis

The qualitative results of DQ-GAT in diverse dynamic environments with different road structures and traffic flows are shown in Fig. 7. For example, in (a) `Int-Left`, the ego-vehicle is taking an unprotected left turn at a four-lane intersection, with many other vehicles driving towards different directions in front. The ego-vehicle first applies a brake to slow down for collision avoidance at t=0 s. In the meantime,

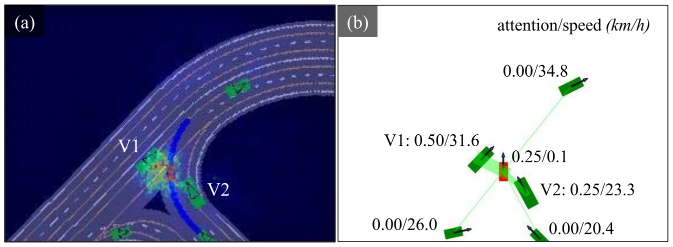


Fig. 8. Policy visualization. (a) Saliency map of the input BEV image **X**, where the computed Jacobian of Q is masked on the **X** for better visualization. Lighter pixels indicate more salient parts with larger values. (b) Attention distribution, where thicker lines indicate larger attentions.

another vehicle A is also waiting to cross the intersection. Rather than stopping at the intersection and waiting other vehicles to leave, the ego-vehicle releases the brake to slowly creep forward at 5.7 km/h in an *exploratory* manner (t=4 s). This interactive manner informs the other vehicles of its intent. Therefore, vehicle A continues to wait and yields to the ego-vehicle. Finally at t=5.2 s when the ego-vehicle arrives in front of vehicle A, it starts to accelerate to finish the turn efficiently.

In some cases, instant safety is more important than efficiency. For example, in (b) `Int-Cross`, when the ego-vehicle is driving straight down the road, an aggressive vehicle B suddenly cuts into the lane of the ego-vehicle at 25 km/h. To ensure safety, the ego-vehicle applies a full brake to lower the speed from 18.8 km/h to 4.3 km/h at t=0.4 s, and then accelerates at t=1.9 s to drive behind vehicle B.

In summary, these exploratory and interactive driving styles are quite similar to how humans drive, leading to a better trade-off between safety and efficiency, which can also explain the advantage of DQ-GAT on the quantitative results in Sec. V-D.

F. Policy Visualization

To better understand the roles of Q-learning and the attention mechanism in this work, in the following, we compute and

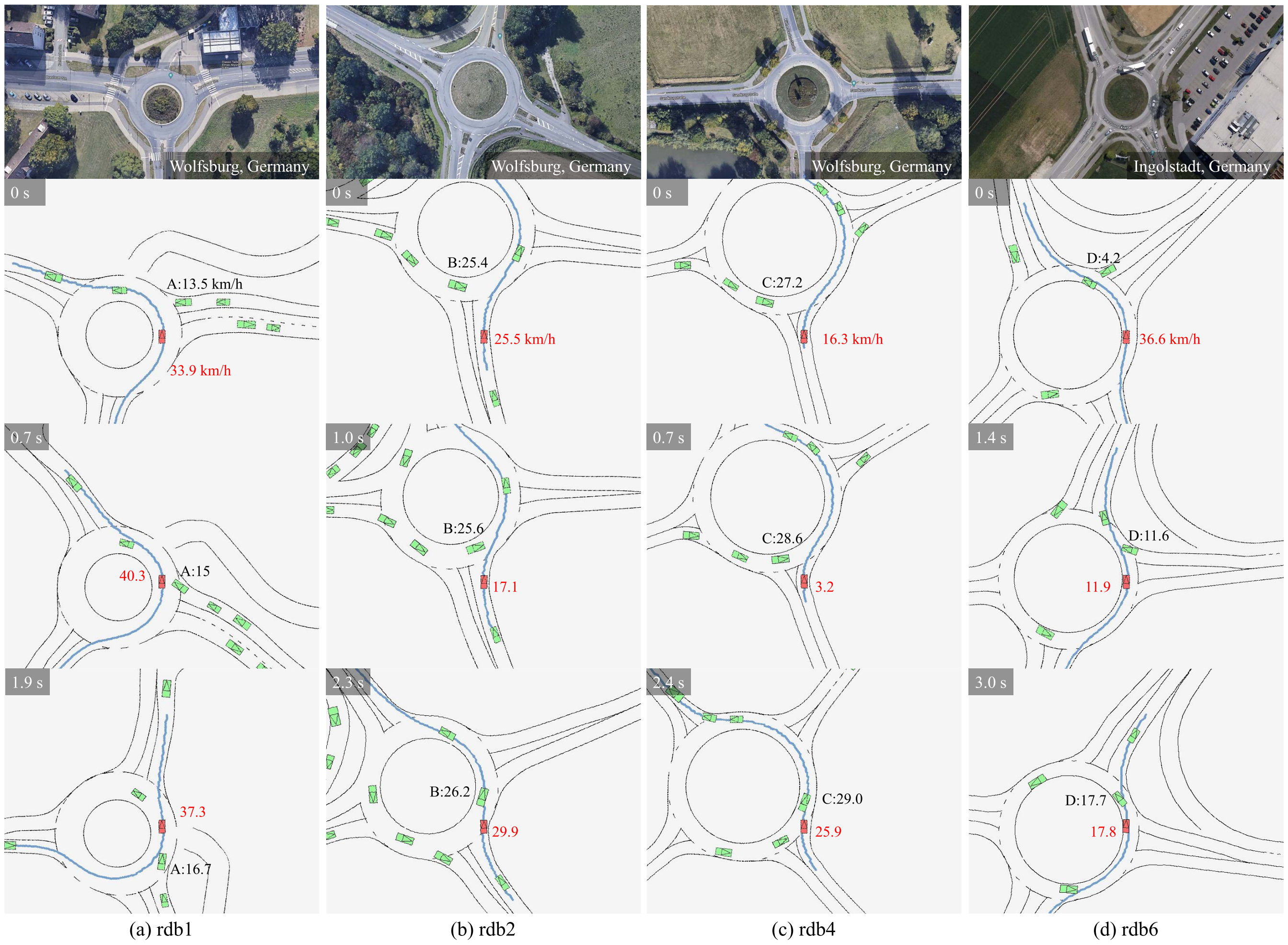


Fig. 9. Qualitative results on the real-world openDD dataset. (a) Maintaining the speed to take the front way of a mild slow-driving vehicle, and decelerating to yield to the front vehicle which is (b,c) at relatively high speeds or (d) accelerating to enter the roundabout.

visualize the salient part [36] of the input BEV image $\mathbf{X}$, and show attention values from the graph networks. Specifically, to show the saliency map of $\mathbf{X}$, we compute the absolute value of the Jacobian of the estimated Q value with respect to the input BEVs: $|\nabla_X Q(s, \arg\max_{a'} Q(s, a'); \theta)|$.

A sample driving case in the `T-Merge` scenario is shown in Fig. 8. We can see that the ego-vehicle is yielding to the front vehicle V1, which drives at high speeds (31.6 km/h) and exerts a strong influence with higher attention (0.5) than the others. Accordingly, as observed in the saliency map, the Q-value also cares more about V1. Note that there is another vehicle V2 in the right lane, which is about the same distance from the ego agent as is V1. However, V2 is assigned with lower attention (0.25), because it is turning into a different lane and does not have much influence on the ego agent. This phenomenon demonstrates that the ideas of GCN(U) and GCN(D) are not reasonable in some occasions because the influences of road agents should be measured within specific contexts and are not always equal or related to distance. By contrast, without external supervisions, our DQ-GAT can still learn to dynamically and reasonably focus on different parts of the environment, like humans do, thanks to the self-attention mechanism. Such a difference also explains the performance gap between GCN methods and our DQ-GAT in Fig. 5.

G. Zero-Shot Generalization Performance

1) New Scenarios in the CARLA Simulator: To examine whether our method can generalize to unseen environments, we further conduct benchmark tests as stated in Sec. V-C in two new scenarios, namely `Roundabout` and `Five-Way`, where the vehicle should drive around a roundabout and take a left turn at a five-way intersection, respectively. The quantitative results are shown in the right column of Table I. The average performance is shown in Fig. 6-(b). We can see that IL-based methods generalize poorly in new scenarios. For example, in `Roundabout` with regular traffic, the success rate of IL-Aggr. is only 26.67%. By contrast, DQ-GAT achieves higher success rates (77.67∼99.67%), with similar completion time to the IL-Aggr. model, in most scenarios.[3]

For qualitative analysis, we demonstrate two driving cases in columns (c and d) of Fig. 7. In (c) `Roundabout`, the ego-vehicle timely slows down at t=3.3 s to yield to the vehicle on the left side (which is departing the roundabout)

[3]An exception is in `Roundabout` with dense traffic. We observe that the IL-Aggr./H-REIL model exhibits low-level interactive driving skills, and tends to collide with other vehicles, leading to a very low success rate (1∼2%). Therefore, it can only aggressively finish a few random cases where safe behaviors that take time (e.g., yielding, vehicle-following, etc.) are not needed, leading to a shorter average completion time than ours.

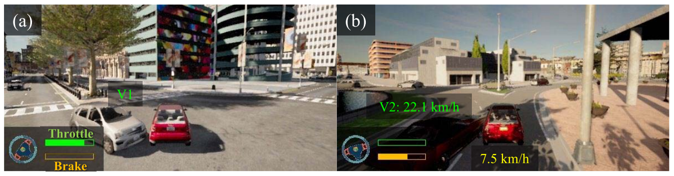


Fig. 10. Sample failure cases of our method: (a) scratching accidents; (b) collisions caused by other aggressive vehicles.

for collision avoidance. In (d) `Five-Way`, the ego-vehicle first slows down to yield to vehicle D1 for collision avoidance, then accelerates at t=0.9 s to inform vehicle D2 to slow down (20.7→12.8 km/h), and finally crosses the traffic flow at t=1.9 s. In summary, similar to the performance in seen environments, DQ-GAT can still dynamically adjust its driving style in new environments according to specific driving contexts.

2) New Scenarios in the Real World: For practical application, we further test the driving performance of DQ-GAT in openDD [46], which is a real-world trajectory dataset focusing on interaction-intensive unregulated roundabouts with varying topologies in Wolfsburg and Ingolstadt, Germany. Qualitative results on four roundabouts in openDD, rdb_1, rdb_2, rdb_4 and rdb_6 are shown in Fig. 9. We can see that the agent makes two different decisions in these scenarios: (a) maintaining the speed against a mild vehicle for *efficiency*, and (b-d) decelerating to yield to the front aggressive vehicle for *safety*.

In addition, the proposed model also observes a notable inference rate of 150 Hz on the NVIDIA RTX 2060 mobile GPU, and 260 Hz on the GTX 1080 Ti GPU, essential for real-time driving applications.

H. Failure Cases

The typical failure cases of our model are shown in Fig. 10. We observe that they can be divided into two categories: inter-vehicle scratching accidents (Fig. 10-(a)), and collisions caused by other aggressive cars (Fig. 10-(b)). For example, in scenario (a), the ego-vehicle is taking an unprotected left turn, but its rear left panel slightly collides with the stopped vehicle V1. In scenario (b), the rear-left vehicle V2 suddenly turns right to depart the roundabout at a relative higher speed of 22.1 km/h. The ego-vehicle timely infers its intent and starts to decelerate by applying large brake values. However, V2 continues to drive without slowing down. Finally, these two vehicles collide. The limitation of our model in handling these near-accident cases leaves possible avenues for future research on safer autonomous driving.

VI. Conclusion and Future Work

In this work, we proposed DQ-GAT to achieve safe and efficient autonomous driving in various urban environments. It is a graph-based network using the self-attention mechanism to encode heterogeneous information in generic traffic scenarios. We extended the original D3QN to an asynchronous version to train the network without relying on human labels. Then, by setting various traffic flows in different scenarios (e.g., unprotected left turns at intersections, merging, and crossing), we extensively evaluated different methods and demonstrated that our DQ-GAT can dynamically adjust its driving styles according to specific driving contexts like human drivers do, finally achieving a better trade-off between safety and efficiency than previous rule-based and learning-based methods. Afterwards, we showed that our method generalizes well in totally unseen scenarios like roundabouts and five-way intersections, where the performance of baseline learning-based methods degrades a lot in terms of safety. Furthermore, we qualitatively tested the zero-shot generalization performance of DQ-GAT, which is trained in a simulator, on the real-world dataset openDD and demonstrated its potential for practical applications.

This work makes a common assumption on perfect perception results. However, measurement noise is inevitable in the real world. In the future, we will investigate how to handle the perceptual uncertainties within the training framework.

References

[1] C. Urmson *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *J. Field Robot.*, vol. 25, no. 8, pp. 425–466, 2008.
[2] S. Kuutti, R. Bowden, Y. Jin, P. Barber, and S. Fallah, "A survey of deep learning applications to autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 2, pp. 712–733, Feb. 2021.
[3] J. Chen, B. Yuan, and M. Tomizuka, "Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Nov. 2019, pp. 2884–2890.
[4] Y. Chen, C. Liu, B. E. Shi, and M. Liu, "Robot navigation in crowds by graph convolutional networks with attention learned from human gaze," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2754–2761, Apr. 2020.
[5] M. Montemerlo *et al.*, "Junior: The Stanford entry in the urban challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, 2008.
[6] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 33–55, Jun. 2016.
[7] W. Zeng *et al.*, "End-to-end interpretable neural motion planner," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8652–8661.
[8] Z. Cao *et al.*, "Reinforcement learning based control of imitative policies for near-accident driving," in *Proc. Robot., Sci. Syst.*, 2020, pp. 1–10.
[9] M. Zhou *et al.*, "SMARTS: Scalable multi-agent reinforcement learning training school for autonomous driving," in *Proc. 4th Conf. Robot Learn.*, 2020, pp. 1–20.
[10] J. Stewart. *Why People Keep Rear-Ending Self-Driving Cars*. Accessed: Jul. 14, 2021. [Online]. Available: https://www.wired.com/story/self-driving-car-crashes-rear-endings-why-charts-statistics/
[11] P. Cai, S. Wang, Y. Sun, and M. Liu, "Probabilistic end-to-end vehicle navigation in complex dynamic environments with multimodal sensor fusion," *IEEE Robot. Autom. Lett.*, vol. 5, no. 3, pp. 4218–4224, Jul. 2020.
[12] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," in *Proc. 3rd Conf. Robot Learn.*, 2019, pp. 66–75.
[13] F. Codevilla *et al.*, "End-to-end driving via conditional imitation learning," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2018, pp. 1–9.
[14] F. Codevilla, E. Santana, A. Lopez, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9329–9338.
[15] P. Cai, Y. Sun, H. Wang, and M. Liu, "VTGNet: A vision-based trajectory generation network for autonomous vehicles in urban environments," *IEEE Trans. Intell. Vehicles*, vol. 6, no. 3, pp. 419–429, Sep. 2021.
[16] C. Liu, Y. Chen, M. Liu, and B. E. Shi, "Using eye gaze to enhance generalization of imitation networks to unseen environments," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 5, pp. 2066–2074, May 2021.
[17] P. Cai, H. Wang, H. Huang, Y. Liu, and M. Liu, "Vision-based autonomous car racing using deep imitative reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7262–7269, Oct. 2021.
[18] P. Cai, X. Mei, L. Tai, Y. Sun, and M. Liu, "High-speed autonomous drifting with deep reinforcement learning," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 1247–1254, Apr. 2020.

[19] A. Kendall *et al.*, "Learning to drive in a day," in *Proc. Int. Conf. Robot. Autom.*, 2019, pp. 8248–8254.
[20] E. Leurent and J. Mercat, "Social attention for autonomous decision-making in dense traffic," 2019, *arXiv:1911.12250*.
[21] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Oct. 2019, pp. 2765–2771.
[22] P. Cai, H. Wang, Y. Sun, and M. Liu, "DiGNet: Learning scalable self-driving policies for generic traffic scenarios with graph neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2021, pp. 8979–8984.
[23] M. Bansal *et al.*, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," in *Proc. Robot., Sci. Syst.*, Jun. 2019, pp. 1–20.
[24] A. Sadat *et al.*, "Perceive, predict, and plan: Safe motion planning through interpretable semantic representations," in *Proc. Eur. Conf. Comput. Vis.*, vol. 12368, 2020, pp. 414–430.
[25] F. Codevilla *et al.*, "On offline evaluation of vision-based driving models," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 246–262.
[26] Z. Zhu and H. Zhao, "A survey of deep RL and IL for autonomous driving policy learning," *IEEE Trans. Intell. Transp. Syst.*, early access, Dec. 22, 2021, doi: 10.1109/TITS.2021.3134702.
[27] D. Wu, X. Dong, J. Shen, and S. C. Hoi, "Reducing estimation bias via triplet-average deep deterministic policy gradient," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4933–4945, Nov. 2020.
[28] X. Dong, J. Shen, W. Wang, L. Shao, H. Ling, and F. Porikli, "Dynamical hyperparameter optimization via deep reinforcement learning in tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 5, pp. 1515–1529, May 2019.
[29] M. Bouton, A. Nakhaei, D. Isele, K. Fujimura, and M. J. Kochenderfer, "Reinforcement learning with iterative reasoning for merging in dense traffic," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–6.
[30] D. M. Saxena, S. Bae, A. Nakhaei, K. Fujimura, and M. Likhachev, "Driving in dense traffic with model-free reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 5385–5392.
[31] M. Kaushik, V. Prasad, K. M. Krishna, and B. Ravindran, "Overtaking maneuvers in simulated highway driving using deep reinforcement learning," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1885–1890.
[32] X. Lu *et al.*, "Video object segmentation with episodic graph memory networks," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 661–679.
[33] J. Yin, J. Shen, X. Gao, D. Crandall, and R. Yang, "Graph neural network and spatiotemporal transformer attention for 3D video object detection from point clouds," *IEEE Trans. Pattern Anal. Mach. Intell.*, early access, Nov. 9, 2021, doi: 10.1109/TPAMI.2021.3125981.
[34] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12.
[35] L. Manso *et al.*, "Graph neural networks for human-aware social navigation," in *Proc. Int. Workshop Phys. Agents*, 2020, pp. 167–179.
[36] Z. Wang *et al.*, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.
[37] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[38] H. Van Hasselt *et al.*, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2094–2100.
[39] M. Fortunato *et al.*, "Noisy networks for exploration," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–21.
[40] A. Dosovitskiy *et al.*, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, vol. 78, Nov. 2017, pp. 1–16.
[41] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 31–36.
[42] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn.*, vol. 48. Jun. 2016, pp. 1928–1937.
[43] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–21.
[44] A. Cosgun *et al.*, "Towards full automated drive in urban environments: A demonstration in gomentum station, California," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2017, pp. 1811–1818.
[45] M. M. Minderhoud and P. H. Bovy, "Extended time-to-collision measures for road traffic safety assessment," *Accident Anal. Prevention*, vol. 33, no. 1, pp. 89–97, Jan. 2001.
[46] A. Breuer *et al.*, "openDD: A large-scale roundabout drone dataset," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst. (ITSC)*, Sep. 2020, pp. 1–6.

Peide Cai received the bachelor's degree in automation from School of Control Science and Engineering, Zhejiang University, in 2018. He is currently pursuing the Ph.D. degree with the Department of Electronic and Computer Engineering, Robotics Institute, Hong Kong University of Science and Technology (HKUST), Hong Kong, China.

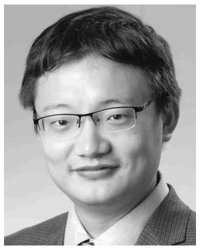
His research interests include autonomous driving, robotics and autonomous systems, and deep learning.

Hengli Wang (Graduate Student Member, IEEE) received the B.E. degree in mechatronics engineering from Zhejiang University, Hangzhou, China, in 2018. He is currently pursuing the Ph.D. degree with the Department of Electronic and Computer Engineering, Robotics Institute, Hong Kong University of Science and Technology (HKUST), Hong Kong, China.

His research interests include computer vision, robot navigation, and deep learning.

Yuxiang Sun (Member, IEEE) received the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong, in 2017.

He is currently a Research Assistant Professor with the Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hong Kong. His research interests include autonomous driving, robotics and artificial intelligence, deep learning, and mobile robots. He is an Associate Editor of IEEE ROBOTICS AND AUTOMATION LETTERS.



Ming Liu (Senior Member, IEEE) received the Ph.D. degree from the Department of Mechanical and Process Engineering, ETH Zürich, Zurich, Switzerland, in 2013.

He is currently an Associate Professor with the Thrust of Robotics and Autonomous Systems, The Hong Kong University of Science and Technology (Guangzhou), China, and also with the Department of ECE, The Hong Kong University of Science and Technology, Hong Kong, SAR, China. His research interests include dynamic environment modeling, 3-D mapping, machine learning, and visual control.