Reinforcement Leaning-Enhanced Simulated Driving Strategies with Large Language Models

Xiangyu Li, Ruochen Jiao, Qi Zhu

Abstract—**This project investigates the transformative potential of large language models (LLMs) in enhancing vehicle-to-everything (V2X) communication and cooperative autonomous driving systems (CADS) in complex urban environments. By integrating LLMs with cyber-physical systems (CPS) and digital twin technologies, this research aims to create realistic and controllable virtual environments that replicate real-world challenges such as dense traffic, high-rise buildings, and diverse driving behaviors. LLMs' capabilities in real-time decision-making, context-aware communication, and adaptive coordination will be leveraged to address critical road safety and efficiency issues. Additionally, the synergy between LLMs and deep reinforcement learning (DRL) will be explored to evaluate the feasibility of training autonomous vehicles in these virtual simulations. The project aims to provide novel insights into optimizing V2X communication systems, improving autonomous vehicle behavior, and advancing smart city infrastructure for safer and more efficient transportation systems.**

Index Terms—**Large language models, Cyber-physical systems, Reinforcement Learning, Vehicle-to-everything, Autonomous driving systems.**

I. Introduction

A Serves as a virtual (often 3D) representation of a real-world system, leveraging real-world data about the characteristics and attributes of physical objects as input to generate simulations or predictions. These outputs adapt dynamically to the input data, enabling the CPS to mirror and analyze real-world behavior. Frequently, there is real-time data communication between the physical system and its virtual twin. The sensors embedded in the real-world system continuously update the virtual twin's depiction, allowing real-time monitoring and synchronization with the real-world system's state. Beyond monitoring, the virtual twin can simulate hypothetical scenarios and optimize parameters to achieve desired outcomes, providing insights for system improvement. This concept was first introduced in 2003 by Professor Grieves from the University of Michigan, initially for the maintenance and health management of aerospace vehicles [1].

In the context of intelligent transportation systems (ITS), advanced driver-assistance systems (ADAS) have emerged as one of the most impactful applications for enhancing road safety and traffic efficiency. ADAS leverages Internet-of-Things (IoT) technologies, such as sensors, cameras, and vehicle-to-everything (V2X) communication, to detect nearby vehicles, identify potential driver errors, and respond proactively to mitigate risks. By utilizing a human-machine interface (HMI), these systems provide real-time assistance in driving, braking, and decision-making functions. Examples of advanced ADAS applications include adaptive cruise control, collision warning systems, lane-keeping assist, and emergency braking.

By integrating CPS with V2X communication and leveraging the predictive and decision-making capabilities of large language models (LLMs), this research seeks to advance the development of cooperative autonomous driving systems (CADS). CPS and digital twin environments provide a platform to simulate, evaluate, and optimize various driving scenarios in controlled virtual settings, addressing the challenges of dense traffic, urban obstacles, and real-time communication. These advancements, combined with the sophisticated reasoning capabilities of LLMs, are poised to transform ADAS and CADS by enabling more adaptive, efficient, and safer autonomous driving systems. Modern advancements in intelligent transportation systems (ITS) are driving the integration of cyber-physical systems (CPS), vehicle-to-everything (V2X) communication, and advanced driver-assistance systems (ADAS) to enhance road safety and optimize traffic efficiency. ADAS, as a critical component of ITS, employs IoT-enabled sensors, cameras, and onboard processing units to assist drivers in functions like braking, lane changing, and collision avoidance. These technologies represent microscopic applications aimed at mitigating risks and improving driving experiences. Examples of advanced ADAS functionalities include:

1) Adaptive Cruise Control (ACC): Automatically adjusts the vehicle's speed based on the distance to the vehicle ahead [2].
2) Collision Avoidance Systems (CAS): Uses radar sensors to detect obstacles and notify the driver of potential crash risks [3].
3) Lane Change Assistance: Monitors blind spots and nearby vehicles to facilitate safe lane changes [4].

While traditional ADAS relies on onboard sensors and cameras, V2X communication extends its capabilities by enabling vehicles to interact with infrastructure, pedestrians, and other vehicles. This broader information exchange results in significant improvements over sensor-limited systems:

1) Expanded Information Sources: V2X-aided ADAS is not confined to vehicle-mounted sensors; it receives data from external sources like roadside units (RSUs) and other vehicles.

Xiangyu Li is the Department of Civil and Environmental Engineering, Northwestern University, Evanston, United States

Ruochen Jiao and Qi Zhu are the Department of Electrical and Computer Engineering, Northwestern University, Evanston, United States

2) Extended Field of Vision: Onboard units (OBUs) and RSUs provide a more comprehensive view of the driving environment, enabling better-informed decisions.
3) Enhanced Efficiency and Performance: V2X enables ADAS to disseminate real-time information over a broader range, improving system responsiveness and reliability.

The growing number of vehicles on global roadways has led to an alarming increase in accident rates. Understanding the relationship between driving behavior, risk perception, and traffic incidents is crucial for developing effective strategies to reduce collisions and improve safety. Existing research often relies on Driving Behavior Questionnaires (DBQ) and driving performance metrics like the standard deviation of speed (SDspeed) to evaluate driver behavior [6].

In this project, the CPS framework will be utilized to simulate real-world driving scenarios, incorporating V2X messaging to study its impact on driving behavior. By inviting participants of varying age groups and driving proficiencies, we will use DBQs to evaluate the acceptance and effectiveness of V2X messages. Driving behavior data collected during these simulations will serve as a foundation for further analysis.

Example Scenario: Illegal Pedestrian Crossing (IPC) To demonstrate a practical application, the project will simulate a high-risk illegal pedestrian crossing (IPC) scenario, which accounts for 93.5% of dangerous road behaviors. Within the CPS, IPC events will occur randomly, requiring vehicles equipped with OBUs to broadcast emergency V2X messages to trailing vehicles. These messages will instruct rear vehicles to decelerate urgently, preventing potential collisions. Participants' responses to these scenarios will be analyzed across different demographics, such as:

- Age Groups: Divided into ranges like [18–28], [29–39], [39–49], etc.
- Driving Proficiency: Measured by driving experience in ranges like [0–10], [10–20], [20–30] years, and so on.

Both DBQ responses and driving performance data will be collected during these experiments, providing invaluable insights into the role of V2X communication in influencing driving behavior.

Deep Reinforcement Learning for Driving Optimization Once a large volume of driving data is collected, it will be used to train reinforcement learning models. Deep reinforcement learning (DRL) has shown significant promise in traffic simulation and driving behavior modeling [7–8]. By defining appropriate states, actions, and reward functions, DRL can optimize vehicle behaviors such as route planning, speed control, car-following, and lane-changing maneuvers.

Contributions of the CPS-Based Framework This project builds upon a CPS-based platform for vehicular networks and ITS to bridge the gap between theoretical models and real-world applications. By implementing V2X messaging scenarios within the CPS, participants can react to dynamic driving conditions while their responses and performance are recorded using a physical driving simulator. The driving data collected will serve multiple purposes:

Evaluating Algorithm Efficiency: Assess the proposed algorithms and V2X messaging systems for safety and performance. Driver Behavior Analysis: Study how drivers perceive and react to V2X messages in varied scenarios. Autonomous Vehicle Training: Utilize the collected data to train deep learning models, enabling autonomous vehicles to respond effectively to V2X messages. To illustrate the potential of this approach, an emergency broadcast message scenario has been implemented on the CPS's digital twin platform, as shown in the figure below. This project demonstrates the synergy between CPS, V2X communication, and DRL, providing a robust foundation for advancing connected and autonomous driving systems (CADS) in smart cities.

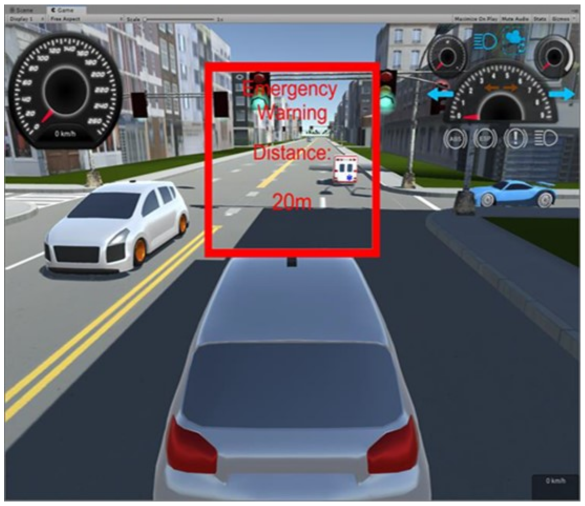

Fig. 1. Historical Trend for Time Spent in Highway Congestion - Bay Area
.

We want to train an automatic driving system that can perfectly adapt to the emergency vehicle situation, and improve the efficiency while ensuring the safety of the overall system, that is, the average waiting time. We collected the standard deviation of speed (SDspeed), standard deviation of heading error (SDHE), mean heading error (meanHE), standard deviation of lateral position (SDLP) data for the simulations on the Logitech G29 driving simulator. We use these data to train the autonomous driving system, optimize these four parameters to ensure the safety of each autonomous vehicle, and ultimately reduce the average waiting time of the entire system.

II. Methodology

A. The development of the cyber-physical system

1) Driving behavior capturing with the physical driving simulator: The physical driving simulator plays an important role in this research because we use it to acquire human driver inputs to the virtual environment and collect real-time driving data from it. Some racing game controllers available in the market such as the Logitech G29 steering suite would be suitable for constructing the driving simulator, because it provides real-time data collection and near-real driving experience.

2) Driving behavior capturing with the cyber system: To the best of our knowledge, there are few existing joint traffic and communication simulation platforms for ITS, and the commercial platforms are mainly autonomous driving platform, such as Baidu Apollo [9]. It is also hard to find an existing digital twin platform for vehicular networks, the only references are IBM Digital Twin Exchange, Gazebo, and Microsoft Azure for Industry 4.0.

In the prototype, the proposed physical and cyber systems will be connected bidirectionally in real time via a communication protocol (e.g., TCP socket). The physical system captures human driving data in real-time, such as speed, acceleration, direction. The cyber system consists of the traffic and V2X simulators. For the traffic simulation, the simulation of urban mobility (SUMO) and Aimsun are common vehicular traffic simulators. Unity 3D and Unreal engine 4 are also good choices for 3D traffic simulation, as they can provide the integration of 3D models, scripts, and viewpoints. For the V2X simulation, OMNeT++, NS3, and Matlab wireless communication toolbox will be exploited in this project.

Eclipse simulation of urban mobility (SUMO) [10] is a microscopic and continuous multi-modal traffic simulation package designed to handle large traffic. The driving parameter can be exported to other software in real-time via traffic control interface (TraCI) [11] via TCP socket. SUMO can serve as the traffic simulation server and provide data exporting and importing to and from other simulators. Open Street Map (OSM) is a database of street layout, building information, and road information in the world, which can be integrated with SUMO for realistic traffic simulation.

Objective Modular Network Testbed in C++ (OMNeT++) Discrete-Event Simulator is a modular, component-based C++ simulation library and framework, primarily for building network simulators [12]. VEINS is an open-source event-based framework [13] for running vehicular networks simulations between OMNeT++ and SUMO via TraCI. They will be used as the network simulator to calculate and transmit V2X messages to the game engine.

Unity is a cross-platform 3D game engine [14] for creating a 3D virtual world to compile the joint traffic and network simulation. Unity also serves as the gateway between the physical and cyber systems, where human drivers can control virtual vehicles in the large-scale 3D virtual simulation environment. The integration of these systems is mainly based on TCP connections, the traffic simulator is the server while other simualtors are clients that establish real-time connections with it. Traffic and network data are transmitted bidirectionally among these simulators for the virtual V2X-aided ADAS platform to operate.

Features:

1) Design V2X message in network simulator
2) Connect to the physical part environment in real time
3) Multi-user access possible
4) High data rate transmission
5) Obtain driving data from the software

B. Cross-layer protocol design for safety message dissemination

Dedicated short-range communications (DSRC) [15] and Cellular-V2X [16] are existing networking technologies for connected vehicles. For the transmission of the safety and warning messages, there is the wireless access in vehicular environments (WAVE) [17] protocol in DSRC for real-time message transmission, including the basic safety message (BSM), WAVE short message (WSM). Self-driving vehicles or drivers can make driving decisions based on these messages. We will propose more efficient and secure message dissemination protocols based on BSM and WSM, which can have faster urgent message transmission and better channel utilization. In WAVE (IEEE 1609.4 standard), multiple channel operation may result in high packet contention in the vehicular network, especially when the vehicular density is high. A solution is to design a suitable channel switching mechanism to balance the contention. In addition, the DSRC protocol stack lacks the mechanism of message verification and retransmission, and the receiver (driver) will not know about the lost messages. This may lead to critical hidden danger in V2X-aided ADAS, and we will address this flaw in the protocol design from a cross-layer perspective based on the analyses conducted with the developed virtual platform.

Traffic modeling is a popular area in transportation and ITS studies, it usually contains microscopic traffic mobility, driver behaviors, traffic control algorithms, etc. For research in vehicular networks, macroscopic stochastic traffic mobility is usually considered for evaluating the performance of different V2X communication protocols. However, it might not be realistic enough as there are different types of traffic scenarios in real-world vehicular networks, such as highway, ramp, intersection, urban street, etc. Under different traffic scenarios and driving strategies, it will lead to different communication impact and hence the performance in ITS applications. With the virtual V2X-aided ADAS platform, the existing V2X protocols can be rigorourly and conveniently evaluated and optimized with respect to different traffic situations.

Features:

1) Design V2X message in network simulator
2) Use of existing V2X protocol stacks (DSRC, CV2X)
3) No interference with existing wireless networks
4) Multi-user access possible
5) Real-time high-speed data exchange of traffic-relevant content with the cloud

C. Autonomous driving deep reinforcement learning models

After implementing the algorithms in the CPS, the driver (player) can react according to different V2X messages, and we can collect a huge amount of driving data through the physical driving simulator. These driving data are invaluable for evaluating the efficiency of the proposed algorithms and models, studying the impact on drivers, as well as training deep learning models for autonomous vehicles that are sensitive to V2X messages. Such deep learning model plays an important role in the research of self-driving vehicles to improve existing models such as the cooperative adaptive cruise control (CACC) model commonly used in many traffic simulators.

The experiment is divided into multiple tasks in different scenarios. We will first invite drivers of different backgrounds to conduct the experiment. According to their driving proficiency, we will test the efficiency of our V2X-aided CPS for experienced, general, and novice drivers. They will use different levels of V2X-aided CPS to drive freely in multiple driving scenarios in the virtual world, requiring safety and compliance with respective to different traffic rules. There will be stochastic traffic conditions in each trial, such as traffic accidents, running traffic lights, speeding, pedestrian's rushing out, etc. After each experiment, drivers will be asked to rate the user experience of the assisted driving system, and we will adapt the amount and frequency of V2X messages accordingly to optimize the effectiveness and driving comfort in specific road conditions.

With the collected dataset from the driving simulator, we will investigate deep learning models for V2X communication-sensitive autonomous driving and compare the performance with generic sensor-based models. Such model can be further imported into the traffic simulator for optimized cooperative autonomous driving. Regarding the training of the deep neural network (DNN) model, we can employ the most commonly used machine learning frameworks such as TensorFlow [18] and Keras [19]. In addition, the Unity game engine also listed the latest machine learning agent [20], which can be exploited to train autonomous driving models.

Features:

- Use of machine learning techniques
- Data collection from both the software and hardware

D. Driving behavior modelling and analysis

In the driving behavior questionnaire (DBQ), we will add the degree of adaptation to the V2X message warning, for example, the level of feeling very useful is 10 to 1, and 10 is the highest. Thus we can draw whether the effect of such notification messages is useful for those with higher driving proficiency. Similarly, this setting is also effective in studying age groups, and we can know whether V2X message promotion is more helpful to the elderly. Several questions in this section gauge participants' unsafe and aggressive driving behavior, which may include speeding, tailgating, distracted driving, and failing to buckle up. Other inquiries looked into the participant's response to encountering rude or hostile behavior from other drivers or users of the road. To mathematically analyse the driving behavior, we will propose several metrics to evaluate it, such as SDspeed. Others can be standard deviation of the acceleration, heading erorr, and so on.

Features:

- Mathematic analysis of driving behavior
- Data collection from different settings of experiment group

III. System Model

A. Traffic Mobility Model Generation

In the research of the Internet of vehicles, if we use real vehicles and real roads for experiments, it will be a great expense, and the controllability is not high, such as adjusting the average speed of vehicles. For this reason, we need to build a traffic mobility model based on real vehicles and roads, in this way, we can directly analyze the performance of different traffic scenarios by modifying various parameters and locations in the mobility model. The existing traffic mobility models (e.g., the Random Waypoint Model, Manhattan Model), cannot fully reflect the real traffic conditions, nor can they capture the real vehicle action. In other words, the real modeling of vehicle trajectory, speed, and other mobility parameters have a very important impact on the performance of the vehicular networks. There are many methods to generate a real traffic mobility model. In the design of firmware distribution, we use open street map (OSM), SUMO, ns3 software to generate traffic mobility models and simulate them. The specific steps to generate a mobility model in Tsim Sha Tsui, Hong Kong is given below.

B. Realistic Tsim Sha Tsui Traffic Mobility Model

Nowadays, the urban traffic model is the communication carrier in the research of the Internet of Vehicles. In urban road topology, there is a strong relationship between vehicles, and their movements are also related. When selecting the location of the urban traffic model, we need to consider the following points:

1. Vehicle types and their trajectory Different vehicle types determine the vehicle's trajectory, speed limit, and other parameters. For example, the trajectory of private cars is mostly different, and the randomness is very strong. But the bus, the school bus's traveling track has a fixed route. Emergency vehicles, such as police cars, ambulances, and fire engines. Their mobility is protected by law. They can violate traffic rules when performing emergency tasks, such as running red lights and exceeding the speed limit on the road.

2. Traffic density The traffic density of different places in the city is different, and the traffic density of the same place at different times is also different. Some urban roads are crowded for a long time every day, some urban roads are idle, and some roads are only crowded during rush hours. In some specific areas like bus stops and parking lots, the traffic density is always high.

3. Characteristics of high-speed motion vehicles: The biggest difficulty in vehicular networks is the high mobility of vehicles, which will cause the vehicle nodes to disconnect from communication from time to time. Even in the high-density traffic flow model, the communication link will be easily damaged. Besides, high-speed vehicles will introduce inter-carrier interference (ICI), which will weaken the communication signal.

The above points are taken into account when building a real traffic mobility model. To meet the above requirements and for the sake of local traffic conditions in Hong Kong, we selected Tsim Sha Tsui, a famous landmark in Kowloon, Hong Kong,

TABLE I
MULTI-INTERSECTION DQN'S ACTION SPACE LOGIC TABLE

Location	Traffic conditions	Vehicle number in this area
Tsim Sha Tsui (474.86 × 418.99 m2)	Sparse	123 (bus+vehicle)
Tsim Sha Tsui (474.86 × 418.99 m2)	Moderate	408 (bus+vehicle)
Tsim Sha Tsui (474.86 × 418.99 m2)	Congestion	513 (bus+vehicle)

as the realistic area for simulation analysis. The software we used is SUMO and Open Street Map (i.e., osmWebwizard in the SUMO library).

The above points are taken into account when building a real traffic mobility model. To meet the above requirements and for the sake of local traffic conditions in Hong Kong, we selected Tsim Sha Tsui, a famous landmark in Kowloon, Hong Kong, as the realistic area for simulation analysis. The software we used is SUMO and Open street map (i.e., osm Webwizard in SUMO library). First of all, the traffic scenes we designed are summarized in Table 2.1 below. We consider three different types of traffic conditions in Tsim Sha Tsui (i.e., Sparse, Moderate, Congestion). The number of buses and vehicles is 123, 408, and 513, respectively.

C. Project setting

The data is collected from different types of participants on the driving simulator, and preprocessing was made on the data. After that we used the Simulation of Urban MObility (SUMO) simulator platform, on a windows-based operating system, where I made single agent simulations, and multi agent (2 agents) simulations, which we can see further in this document. The data of cars flow was generated in the SUMO simulator, and was transferred to python with the Traci library.

D. Single agent case

In this setting we first create a traffic simulation which only contains a single intersection. The traffic is coming from all sides of the road, and we named each side of the road as North, East, West and South. Each road has three options to cross the intersection – Going straight, turning left or right. For traffic generation we use Simulation of Urban Mobility (SUMO), as shown in the picture below.

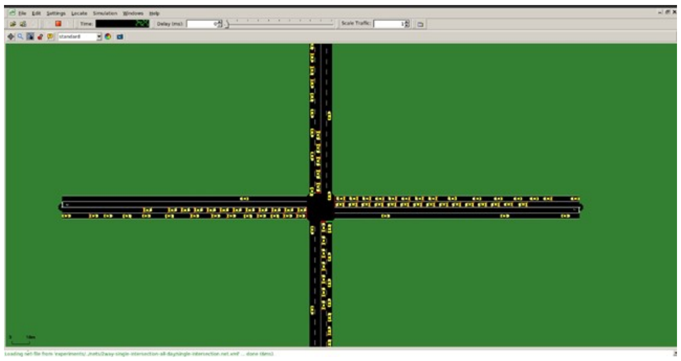

Fig. 2. Historical Trend for Time Spent in Highway Congestion - Bay Area.

E. WHAT IS Reinforcement Learning?

Reinforcement Learning is a branch of Machine Learning with a specific structure and flow, as described in the chart. An RL agent performs an action in an environment, which causes some kind of reaction/result in the system. The result that appeared yields a reward – whether the consequences were good or bad. These results are being transferred back to the agent, which adapts itself corresponding to the reward it got from his last action. RL algorithms can be divided into two types: model-free RL algorithms and model-based RL algorithms. In model-free RL algorithms, we do not have an exact model of the environment, which means that we do not know what will happen in the next time step after we performs an action. On the other hand, in model-based RL algorithms, agent must have to learn the model of the environment which may not be available in most of the real-world problems.

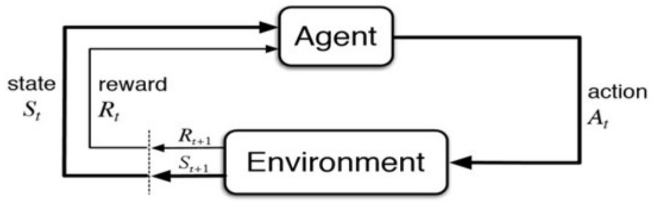

Fig. 3. DQN process.

In this work I used model-free RL algorithms to control the overall automatic driving system. Model-free algorithms further can be classified as value-based (Q-learning and QL with Neural Networks = DQN) and policy gradient (A2C, PPO) algorithms. In this work we will work with both value based and policy gradient algorithms, and examine their success on our problem.

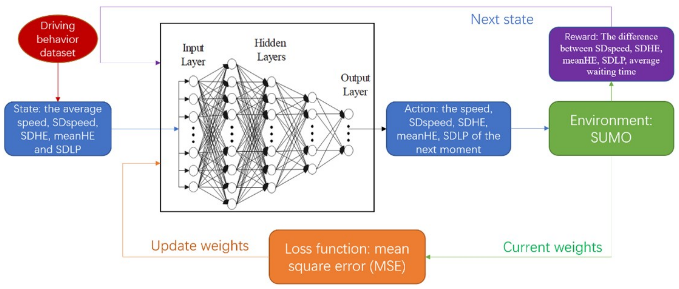
Driving behavior dataset
Input Layer
Hidden Layers
Output Layer
Next state
Reward: The difference between SDspeed, SDHE, meanHE, SDLP, average waiting time
State: the average speed, SDspeed, SDHE, meanHE and SDLP
Action: the speed, SDspeed, SDHE, meanHE, SDLP of the next moment
Environment: SUMO
Update weights
Loss function: mean square error (MSE)
Current weights

Fig. 4. DQN.

Since we didn't learn these in class, I'll elaborate: A2C – The Advantage Actor Critic algorithm is somewhat of a merge between the value-based and policy-gradient methods. Each iteration, it computes two values – the Critic function, which is the same function the DQN goes by, to minimize the error, and the Actor function, which takes the Critic's result in consideration and updates the policy the agent goes by.

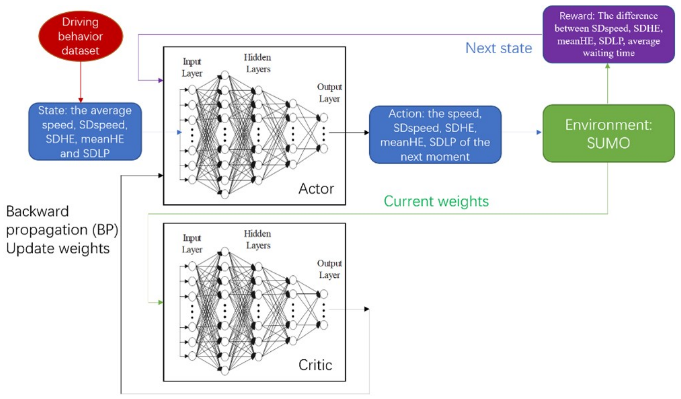
Driving behavior dataset
State: the average speed, SDspeed, SDHE, meanHE and SDLP
Input Layer
Hidden Layers
Output Layer
Actor
Action: the speed, SDspeed, SDHE, meanHE, SDLP of the next moment
Environment: SUMO
Reward: The difference between SDspeed, SDHE, meanHE, SDLP, average waiting time
Next state
Current weights
Backward propagation (BP) Update weights
Critic

Fig. 5. A2C.

PPO - The Proximal Policy Optimization algorithm does what on-policy algorithms do, which is to continuously update the policy, so that the agents using the policy will be guided to making the optimal decisions for the entire system, to maximize our reward OVERALL (opposed to maximizing it in the short term, like DQN)

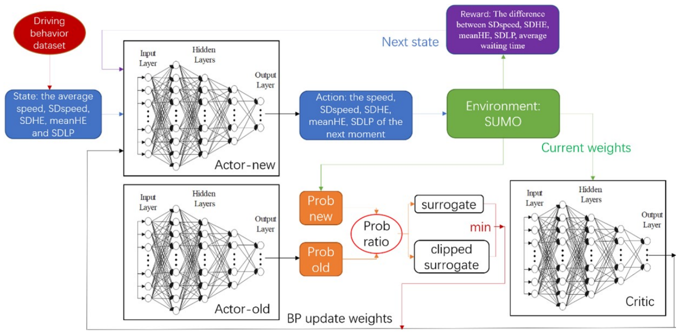
Driving behavior dataset
State: the average speed, SDspeed, SDHE, meanHE and SDLP
Input Layer
Hidden Layers
Output Layer
Actor-new
Action: the speed, SDspeed, SDHE, meanHE, SDLP of the next moment
Environment: SUMO
Reward: The difference between SDspeed, SDHE, meanHE, SDLP, average waiting time
Next state
Current weights
Prob new
Prob old
Prob ratio
surrogate
clipped surrogate
min
Actor-old
BP update weights
Critic

Fig. 6. PPO.

F. State setting

We used the average speed, SDspeed, SDHE, meanHE and SDLP in the automatic driving system as the state input.

G. Action setting

We used the speed, SDspeed, SDHE, meanHE, SDLP of the next moment as the action space.

H. Reward setting

The difference between SDspeed, SDHE, meanHE, SDLP, average waiting time at the next moment and SDspeed, SDHE, meanHE, SDLP, average waiting time at the previous moment. The reward values are the differences.

IV. Expected Results and Implication of Results

A. Expected Results

The proposed Cyber-physical system and V2X message can meet the needs of smart city and can be commercialized. They can fully demonstrate the advantages of CPS technology in CATS systems and the advantages compared with traditional technique/measurement.

For simulation and data analysis, they can fully demonstrate the feasibility and high reliability of deep reinforcement learning and driving behavior improvement.

We expect that the CPS system can effectively help collect the data of the family members of the participants, which can be used to analyze the effectiveness of our proposed V2X message reminder and help improve road safety. Secondly, these data can also be used to train self-driving vehicles to optimize the metric of driving behavior.

B. Practical Implications

- Findings from this proposal can provide references for cyber-physical system design.
- The methodologies used in this project can contribute to further CPS and V2X communication development.
- Potential deep reinforcement learning algorithm (e.g., PPO, DDPG) for the CPS will be discovered.

C. Theoretical Implications

- Theoretical findings from this proposal can help strengthen the existing theory in CPS technology and create new idea for further research.
- Theoretical findings can help us understand the relationship between driving behavior and CPS.

D. Model Comparison Conclusion: We've seen that PPO and A2C work better than DQN on this case of problem. Why?

A – A2C tends to succeed in scenarios where information needs to be transferred between agents in an environment.

B – This problem fits the on-policy model better, since it's very behavior-based, and doesn't converge well when actions are always taken in a greedy manner (like DQN).

C – Having a policy that is learned and can guide the agents what to do next helps us here, since it provides a lot of information to the agents, who can now make their decisions based on the entire environment instead of just making sure they do the best thing for their own "small world" in the specific moment in time. Also, a policy to guide the agents provide us with a macro view that is capable of capturing the whole scene, instead of each agent optimizing itself just by the next move.

This project addresses the challenge of optimizing decision-making strategies for autonomous vehicles in complex multi-agent traffic scenarios, particularly where interactions between vehicles involve cooperation or competition. Traditional approaches in autonomous driving often rely on supervised learning methods that require large-scale labeled datasets, which can be costly and limited in representing dynamic, real-world traffic interactions. Furthermore, existing models may lack adaptability and reasoning capabilities when facing novel or adversarial behaviors from other agents.

To overcome these limitations, we propose a novel framework that integrates Reinforcement Learning (RL) from self-playing with Large Language Models (LLMs) as decision makers. Inspired by the paradigm introduced in "Self-playing Adversarial Language Game Enhances LLM Reasoning," this approach leverages self-playing to simulate interactions between autonomous vehicles, allowing the models to dynamically generate diverse and realistic training data. Through this self-playing mechanism, LLMs are trained to predict and adapt to the strategies of other agents, enabling robust and efficient decision-making in multi-agent traffic environments.

In this framework, autonomous vehicles equipped with LLMs engage in iterative self-play, where one vehicle simulates cooperative or competitive strategies while the other optimizes its responses based on real-time reasoning. For example, the Ramp Autonomous Vehicle (RAV) aims to merge into the main road, while the Main Road Autonomous Vehicle (MAV) evaluates whether to cooperate or compete, considering its own objectives and predictions of the RAV's behavior. Meanwhile, the RAV predicts the MAV's decisions based on the MAV's historical actions and determines its own course of action based on the environment. This dynamic process fosters mutual adaptation of strategies and enhances the LLMs' reasoning capabilities.

The proposed method has the following main contributions:

- **Advancement in Autonomous Driving Strategies:** This work pioneers the application of reinforcement learning from self-play in the autonomous driving domain, offering a novel, efficient, and scalable solution to optimize driving strategies in multi-agent transportation scenarios.
- **Enhancement of LLM Reasoning Capabilities:** Training Large Language Models (LLMs) within this task not only improves their problem-solving skills specific to autonomous driving but also enhances their general reasoning abilities, enabling better performance and transferability across diverse domains.
- **Efficient Data Utilization via Self-Play:** By employing self-play reinforcement learning, this approach eliminates the dependency on large-scale supervised datasets. The self-play paradigm generates training data through simulated interactions, significantly reducing computational and data acquisition costs.

V. Problem Definition

A. Scenario Description

In this scenario, an autonomous vehicle on a ramp named ramp autonomous vehicle (RAV) aims to merge into the main road, while an autonomous vehicle named main road autonomous vehicle (MAV) on the main road faces a decision:

- **Cooperation:** The MAV can choose to yield by changing lanes or reducing its speed, facilitating the RAV's merge. The RAV may decelerate and wait for the MAV to pass before merging.
- **Competition:** The MAV can maintain its current lane and speed, or accelerate, focusing more on its own efficiency without providing convenience to the RAV. The RAV may accelerate to merge.

The RAV determines its strategy based on a prediction of the MAV's behavior:

- If the RAV predicts that the MAV will **cooperate**, it may choose to merge aggressively.
- If the RAV predicts that the MAV will **compete**, it may choose to delay its merge or adjust its acceleration.

Both vehicles use their respective large language models (LLMs) to analyze the situation and make decisions:

- The MAV uses its LLM to analyze the historical behavior of the RAV, assess whether cooperation or competition would optimize its objectives, such as maintaining efficiency or avoiding delays.
- The RAV leverages its LLM to analyze the historical behavior of the MAV, predict its current strategy, and decide its own merging approach.

This interaction leads to a dynamic decision-making process where both AVs adapt their strategies in real-time to achieve their respective goals of safety, efficiency, and merging success.

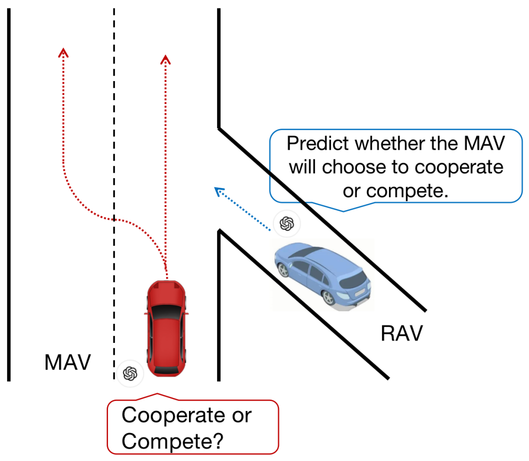
Predict whether the MAV will choose to cooperate or compete.
RAV
MAV
Cooperate or Compete?

Fig. 7. Transportation scenario setup

3. Game Setup

Players:

- **RAV:** Chooses acceleration/merging strategy based on MAV's predicted behavior.
- **MAV:** Decides whether to cooperate or compete based on historical observation of RAV.

State

In this simulation environment, we define the states for the two vehicles as follows:

- **RAV State:** The state of RAV, denoted as $s_{r,t}$, includes:

$$s_{r,t} = [x_{r,t}, lane_{r,t}, v_{r,t}, a_{r,t}, P_{cooperate,r}],$$

where:

- $x_{r,t}$: position of the RAV in $lane_r$ at timestep t.

- $lane_{r,t}$: The lane number where the RAV is located at timestep t.
- $v_{r,t}$: The speed of the RAV at timestep t.
- $a_{r,t}$: The accelerations of the RAV at timestep t.
- $P_{cooperate,r}$: Probability of cooperation as predicted by the LLM for the MAV at timestep t.

- **MAV State:** The state of MAV, denoted as $s_{m,t}$, includes:

$$s_{m,t} = [x_{m,t}, lane_{m,t}, v_{m,t}, a_{m,t}, P_{cooperate,m}],$$

where:

- $x_{m,t}$: position of the MAV in $lane_r$ at timestep t.
- $lane_{m,t}$: The lane number where the MAV is located at timestep t.
- $v_{m,t}$: The speed of the MAV at timestep t.
- $a_{m,t}$: The accelerations of the MAV at timestep t.
- $P_{cooperate,m}$: Probability of cooperation as predicted by the LLM for the RAV at timestep t.

Dynamic Updates

The states for both vehicles evolve over time based on their respective dynamics:

$$s_{t+1} = f(s_t, u_t)$$

$$s_t = [s_{r,t}, s_{m,t}]$$

where:

- s_t: Current state of the vehicle.
- $u_t = [a_{m,t}, a_{r,t}, lane_{m,t}, lane_{r,t}]$: Control inputs, representing the accelerations and lane-changing behavior of MAV and RAV.
- f: Vehicle dynamics model.

Lane-Specific Behavior

For lane information, we define:

$$\mathrm{l}ane_{m,t} = \begin{cases} 0 & \text{if the vehicle is in the left lane of the main road.} \\ 1 & \text{if the vehicle is in the right lane of the main road.} \\ 2 & \text{if the vehicle is located at the ramp.} \end{cases}$$

LLM Integration for Predictions

The LLM predicts the probability of cooperation based on historical observations:

$$P_{cooperate,r} = \text{LLM}\,(\text{history of MAV's actions}),$$
$$P_{cooperate,m} = \text{LLM}\,(\text{history of RAV's actions}).$$

4. Prompt Framework for Action Generation by LLMs

The decision is based on analyzing lane positions, velocities, accelerations, and the predicted cooperation probabilities.

MAV Action Generation Prompt

The MAV's decision-making process involves the following steps:

$$\text{Action}_{\text{MAV}} = \arg \max_{a_{\text{MAV}} \in \{\text{Cooperate, Compete}\}} R_{\text{MAV}}(s_t, a_{\text{MAV}}, a_{\text{RAV}}) + \beta_1 \cdot R_{\text{system}},$$

where:

- $R_{\text{MAV}}(s, a_{\text{MAV}}, a_{\text{RAV}})$: The reward function for the MAV, dependent on the state s, its own action a_{MAV}, and the RAV's predicted action a_{RAV}.
- $\beta_1 \cdot R_{\text{system}}$: The action of the MAV considers the system's reward, where β_1 is the weight assigned to the system's consideration.

a) LLM Prompt for MAV Action::

- Input: $s_{m,input} =$
$\mathrm{x}_{m,t}, lane_{m,t}, v_{m,t}, a_{m,t}, P_{\text{cooperate},m}, x_{r,t}, lane_{r,t}, v_{r,t}, a_{r,t},$

- Query:

Acceleration: $\mathrm{a}_{m,t}, a_{r,t}$
Predicted cooperation probability from RAV: $P_{\text{cooperate},m}$,
Should MAV cooperate or compete?

- Output: Action recommendation: *Cooperate* or *Compete*? *Accelerate*, *Decelerate*, or *Lane-changing*.

B. RAV Action Generation Prompt

The RAV's decision-making process involves:

$$\text{Action}_{\text{RAV}} = \arg \max_{a_{\text{RAV}} \in \{\text{Accelerate, Decelerate, Merge}\}} R_{\text{RAV}}(s, a_{\text{RAV}}, a_{\text{MAV}}) + +\beta_2 \cdot R_{\text{system}},$$

where:

- $R_{\text{RAV}}(s, a_{\text{RAV}}, a_{\text{MAV}})$: The reward function for the RAV, dependent on the state s, its own action a_{RAV}, and the MAV's predicted action a_{MAV}.
- $\beta_2 \cdot R_{\text{system}}$: The action of the RAV considers the system's reward, where β_2 is the weight assigned to the system's consideration.

a) LLM Prompt for RAV Action::

- Input:
$s_{m,input} =$
$\mathrm{x}_{m,t}, lane_{m,t}, v_{m,t}, a_{m,t}, P_{\text{cooperate},r}, x_{r,t}, lane_{r,t}, v_{r,t}, a_{r,t},$

- Query:

Based on the current state:
Lane: $lane_{m,t}, lane_{r,t}$
Position: $x_{m,t}, x_{r,t}$ Velocity: $v_{m,t}, v_{r,t}$ Acceleration: $a_{m,t}, a_{r,t}$
Predicted cooperation probability from MAV: $P_{\text{cooperate},r}$,
Should RAV cooperate, compete?

- Output: Action recommendation: *Cooperate* or *Compete*? *Accelerate*, *Decelerate*, or *Merge*.

C. Risk of Collision Using Time-to-Collision (TTC)

The **Risk of Collision** is represented using Time-to-Collision (TTC), defined as:

Risk of Collision

$$(x_{m,t}, x_{r,t}, v_{m,t}, v_{r,t}) = \begin{cases} \frac{|x_{r,t}-x_{m,t}|}{|v_{m,t}-v_{r,t}|}, \\ \\ \text{if } lanem,t = 1, lane_{r,t} = 1 \text{ and } x_{r,t} >= 0, \\ \text{else if } lanem,t = 1, lane_{r,t} = 2 \text{ and } x_{m,t} >= 0, x_{r,t} < 0 \\ \infty, \end{cases}$$

- $x_{m,t}, x_{r,t}$: Positions of two vehicles (e.g., MAV and RAV).
- $v_{m,t}, v_{r,t}$: Velocities of two vehicles.
- ϵ: A small threshold to avoid division by zero, here we use 10^{-6}.

When TTC falls below a threshold τ_{TTC}, a penalty is applied:

$$\text{Penalty}_{\text{Collision}} = \begin{cases} \frac{\tau_{\text{TTC}}-\text{TTC}}{\tau_{\text{TTC}}}, & \text{if TTC} < \tau_{\text{TTC}=2s}, \\ 0, & \text{otherwise.} \end{cases}$$

D. MAV Reward Function

The reward function for the Mainline Autonomous Vehicle (MAV) is defined as:

$$R_{\text{MAV}}(s, a_{\text{MAV}}, a_{\text{RAV}}) = -\lambda_1 \cdot \text{Time Delay} - \lambda_2 \cdot \text{Penalty}_{\text{Collision}} - \lambda_3 \cdot \text{Cost of Yielding}(a_{\text{MAV}})$$

where:

- Time Delay: Time lost due to yielding or competing actions.
- $\text{Penalty}_{\text{Collision}}$: Derived from TTC, penalizing high collision risk.
- Cost of Yielding(a_{MAV}): A penalty for cooperative actions such as deceleration or lane change.

E. RAV Reward Function

The reward function for the Ramp Autonomous Vehicle (RAV) is defined as:

$$R_{\text{RAV}}(s, a_{\text{RAV}}, a_{\text{MAV}}) = +\gamma_1 \cdot \text{Successful Merge}$$

-γ_2 · Time Delay

-γ_3 · $\text{Penalty}_{\text{Collision}}$where:

- Successful Merge: A binary reward for successfully merging into the main lane without long waiting time.
- Time Delay: Time lost during the merging process.
- $\text{Penalty}_{\text{Collision}}$: Derived from TTC, penalizing high collision risk.

F. Overall Rewards for MAV and RAV

MAV Overall Reward: The MAV's overall reward is:

$$\mathcal{R}_{\text{MAV}} = \mathbb{E}\left[R_{\text{MAV}}(s, a_{\text{MAV}}, a_{\text{RAV}})\right]$$

1) RAV Overall Reward: The RAV's overall reward is:

$$\mathcal{R}_{\text{RAV}} = \mathbb{E}\left[R_{\text{RAV}}(s, a_{\text{RAV}}, a_{\text{MAV}})\right]$$

G. Combined System Reward

The combined reward for evaluating the system's overall performance is:

$$\mathcal{R}_{\text{system}} = \mathcal{R}_{\text{MAV}} + \mathcal{R}_{\text{RAV}}$$

if $v_{m,t} \neq v_{r,t}$ and $|v_{m,t} - v_{r,t}| > \epsilon$,

This formulation encourages both safe and efficient interactions between the vehicles.

otherwise,

H. Dynamic Updates for State Transitions

The state transitions are defined as:

$$x_{t+1} = x_t + v_t \cdot \Delta t,$$
$$v_{t+1} = v_t + a_t \cdot \Delta t,$$
$$l_{t+1} = l_t + \Delta l,$$

where:

- x_t, v_t, a_t: Current position, velocity, and acceleration.
- Δt: Time step.
- l_t: Current lane index.
- Δl: Lane change indicator.

VI. Reinforcement Learning Framework

Policy Representation with LLMs

$$\pi_\theta(a|s) = P(a|f(s))$$

where $f(s)$ is the LLM-generated context feature based on the state.

Markov Decision Process (MDP)

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$$

- $\mathcal{S}$: State space.
- $\mathcal{A}$: Action space.
- P: Transition probabilities.
- R: Reward function.
- γ: Discount factor.

A. PPO Training Steps

Policy Gradient Objective

$$L^{\text{PPO}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t\right)\right]$$

where:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

B. Advantage Estimation

$$A_t = \sum_{l=0}^{\infty}(\gamma\lambda)^l \delta_{t+l}, \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$$

VII. Training Workflow

1) **Initialization:** Train an imitation learning policy $\pi_{\theta_{\text{init}}}$ using expert trajectories.
2) **Self-Play Training:**
$$\theta \leftarrow \theta + \alpha \nabla_\theta L^{\text{PPO}}(\theta)$$
3) **Iterative Updates:** Compute win/loss rates and adjust reward shaping as necessary.

VIII. Experiment Description

This experiment uses the large language model Llama 3.1 (pretrained model version 7B) to infer and simulate the decision-making behaviors of MAV (Mobile Autonomous Vehicle) and RAV (Following Vehicle) in an in-vehicle system. The objective is to model the historical and current states of MAV and RAV and generate corresponding decision-making and action recommendations based on this information.

IX. Experimental Procedure

A. Model Selection and Loading

The pretrained Llama 3.1 model was selected and loaded into the inference pipeline. To accelerate the inference process, configurations such as `torch_dtype=torch.bfloat16` and `device_map="auto"` were applied, enabling the model to allocate resources automatically across available devices.

B. Historical Data Simulation

a) MAV Historical Data:: The MAV's historical data includes its position, lane, speed, acceleration, decisions, and actions (e.g., cooperation, competition, deceleration, or acceleration) at previous time steps.

b) RAV Historical Data:: Similarly, the RAV's historical data consists of position, lane, speed, acceleration, decisions, and actions (e.g., deceleration, acceleration, maintaining speed).

Example:

```
# MAV historical states
mav_history = [
    {'x_m_t': 100, 'lane_m_t': 1, 'v_m_t': 30,
    ↪   'a_m_t': 2, 'decision_m_t':
    ↪   'cooperate', 'action_m_t':
    ↪   'decelerate'},
    {'x_m_t': 105, 'lane_m_t': 1, 'v_m_t': 32,
    ↪   'a_m_t': 2.1, 'decision_m_t':
    ↪   'compete', 'action_m_t':
    ↪   'accelerate'},
    {'x_m_t': 110, 'lane_m_t': 1, 'v_m_t': 33,
    ↪   'a_m_t': 1.9, 'decision_m_t':
    ↪   'cooperate', 'action_m_t': 'Lane
    ↪   Change'}
]

# RAV historical states
rav_history = [
    {'x_r_t': 110, 'lane_r_t': 1, 'v_r_t': 28,
    ↪   'a_r_t': 1.5, 'decision_r_t':
    ↪   'cooperate', 'action_r_t':
    ↪   'Decelerate'},
    {'x_r_t': 112, 'lane_r_t': 1, 'v_r_t': 30,
    ↪   'a_r_t': 1.6, 'decision_r_t':
    ↪   'compete', 'action_r_t':
    ↪   'accelerate'},
    {'x_r_t': 115, 'lane_r_t': 1, 'v_r_t': 31,
    ↪   'a_r_t': 1.7, 'decision_r_t':
    ↪   'compete', 'action_r_t': 'Maintain
    ↪   Speed'}
]
```

C. Current State Simulation

The current states of MAV and RAV include position, lane, speed, acceleration, and decision probabilities.

```
# MAV current state
mav_state = {
    'x_m_t': 120,
    'lane_m_t': 1,
    'v_m_t': 35,
    'a_m_t': 2.5,
    'P_cooperate_m': 0.8
}

# RAV current state
rav_state = {
    'x_r_t': 125,
    'lane_r_t': 1,
    'v_r_t': 33,
    'a_r_t': 1.8,
    'decision_r_t': 'cooperate',
    'action_r_t': 'Lane Change'
}
```

D. Prompt Construction

Using the historical and current state data, prompts are generated for the MAV system using the `generate_mav_prompt` function. These prompts incorporate MAV and RAV historical behavior and current states to guide the model in making appropriate decisions.

E. Inference Process

a) MAV Decision Inference:: Using the generated prompts, the inference pipeline (`pipe`) predicts MAV's behavior, determining whether it should cooperate and what actions to take (e.g., accelerate, decelerate, or change lanes).

b) RAV Decision Inference:: Similarly, the `infer_rav_decision` function predicts RAV's behavior, such as whether it will cooperate and what specific actions it will take.

X. Inference Results

A. RAV Inference

a) Behavior Prediction (Step 3):: Based on the historical states of the RAV, the model predicts that the RAV will choose to cooperate. The RAV has cooperated in two out of three instances, indicating a pattern of prioritizing safety and cooperation.

b) Action Output (Step 4):: Given the predicted behavior, the model recommends the RAV to cooperate. This decision aligns with the RAV's history, which shows a tendency toward safety. To support this cooperation, the action `Decelerate` is recommended to create a safe gap for merging and avoid collisions.

c) Decision Explanation:: The decision to cooperate is based on the predicted behavior of the RAV. Deceleration ensures safety and aligns with the cooperative behavior pattern. This recommendation prioritizes a safe and smooth merging process.

B. MAV Inference

a) Decision Explanation:: Based on the historical and current states of the RAV, the model predicts that the RAV will compete. The RAV's behavior history shows competition in two out of three instances, suggesting a tendency to prioritize self-interest. Its higher speed and acceleration further reinforce this competitive behavior.

Given this prediction, the MAV should also choose to compete. The recommended action is `Maintain Speed`, which prevents escalation while avoiding giving the RAV an opportunity to merge safely. This decision minimizes collision risk while maintaining the MAV's position.

C. Experimental results- Single Agent (Intersection) problem:

We first run Random agent, which makes random actions, which aren't a result of his previous actions & rewards, and fixed agent, which gives each vehicle a fixed action such as "speed+=5", and compare their performance. Figure 7 clearly

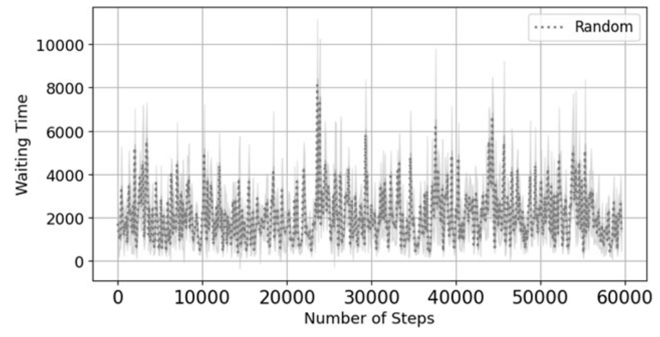

Fig. 8. Random action space.

shows that random action space does not learn anything (as expected), and the waiting time doesn't improve overtime. As shown in figure 8 and 9, fixed action space has much better performance than random action space in terms of speed (e.g., avg. waiting time), but still – as it's fixed, it doesn't improve overtime, and doesn't learn about the real flow of the traffic in the intersection.

Now, we move on to trying to manage the driver behaviors (SDspeed, SDHE, meanHE, SDLP) to be responsive of the actual traffic in the intersection, using DQN: DQN seems to converge and improve its results overtime, but the final results aren't outstanding. The DQN algorithm is an off-policy algorithm, which tries to maximize the reward it gets in each iteration separately. This approach gets us pretty good results, but not optimal. Optimal results can be achieved with algorithms that learn a POLICY to go by (like a manual for the agents), that way we achieve a MACRO view instead of a MICRO view in the DQN algorithm.

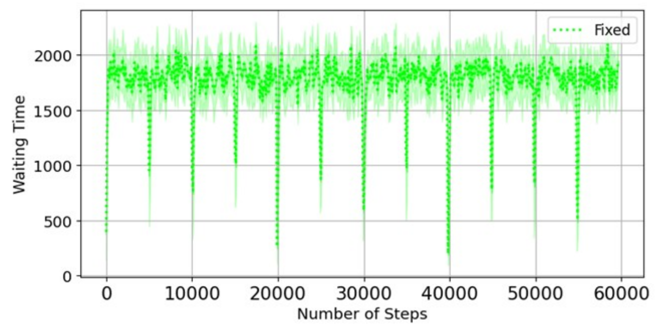

Fig. 9. Fixed action space.

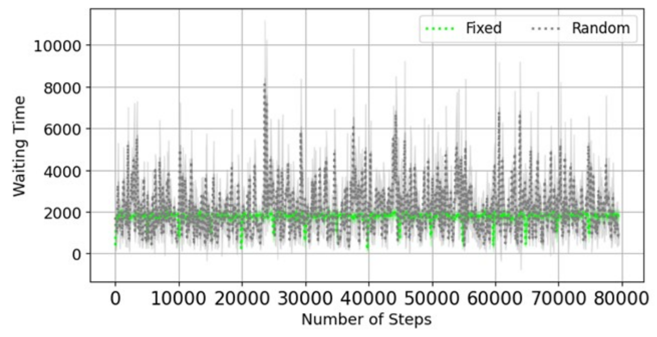

Fig. 10. DQN result of single agent traffic.

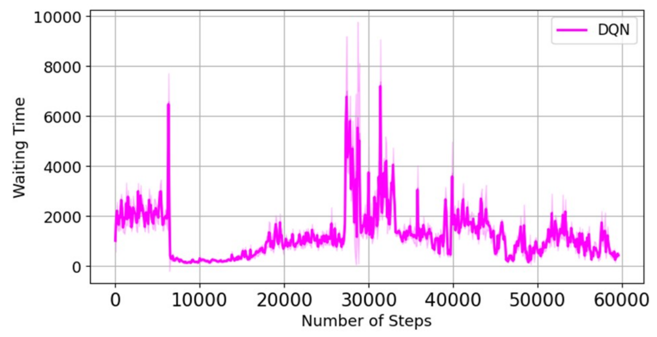

Fig. 11. DQN result of single agent traffic.

After realizing on-policy algorithms might perform better in this problem, let's take a look on their result, in the next page. Let's check the A2C performance: As we can see in the chart, A2C algorithm manages to successfully deal with the single intersection driving behavior modelling problem and controls each vehicle in a way that minimizes almost completely the cars' waiting time.

Let's also check PPO performance: As they are relatively similar, PPO algorithm also performs well in the single agent problem.

Single agent conclusion:

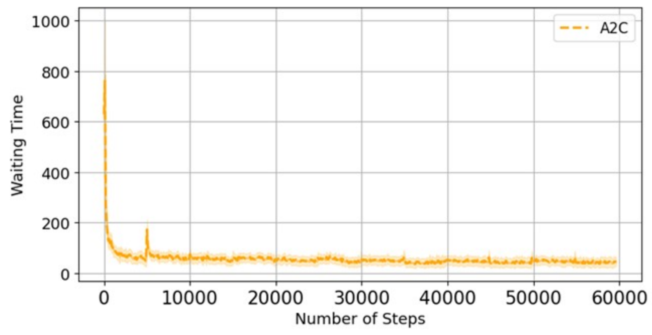

Fig. 12. A2C result of single agent traffic.

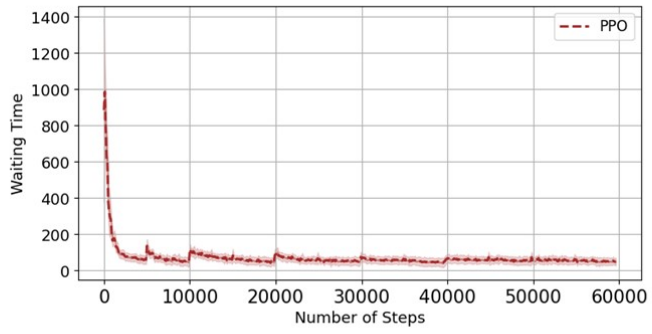

Fig. 13. PPO result of single agent traffic.

As shown in Figure 7:

Random Action – cannot learn any driving behavior.

Fixed Action – cannot learn any driving behavior, the result don't improve overtime.

DQN – Converges successfully overtime and is better than fixed, but still performs worse than on-policy algorithms.

A2C, PPO – Waiting time decreases massively and end up minimized – solves and deals well with one intersection. Multi-Agent:After finding a good driving behavior model and

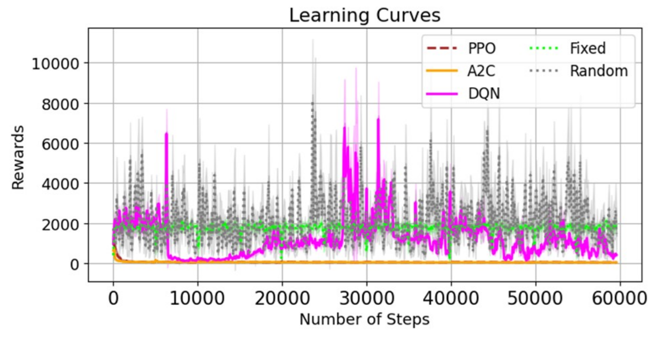

Fig. 14. Reward comparison of different RL of the single agent.

minimizing waiting time in the Single Agent problem (just one intersection), we move on to the multi-agent problem – The real world. In real life, there are a lot of intersections, where one road you take in an intersection leads you to another in the next one. We'll attempt to deal with this problem as well. In the picture above, we can see two intersections (therefore

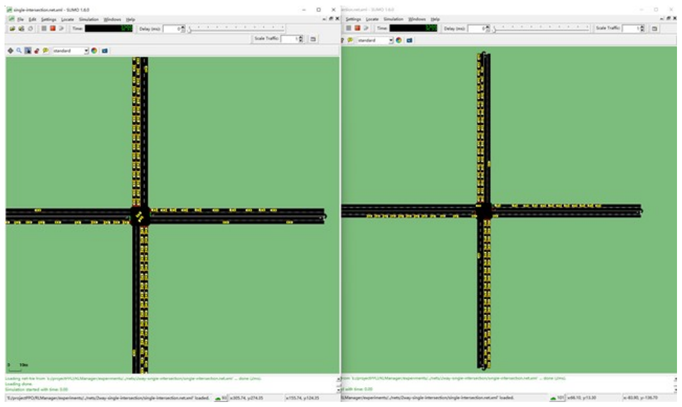

Fig. 15. Reward comparison of different RL of the single agent.

we have two agents), where each agent is controlling its own intersection. Here are the results:

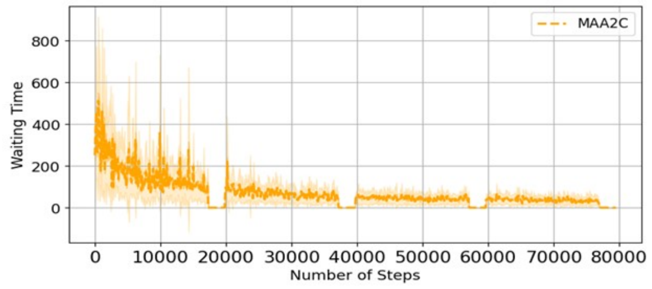

Fig. 16. A2C result of the multiple agent.

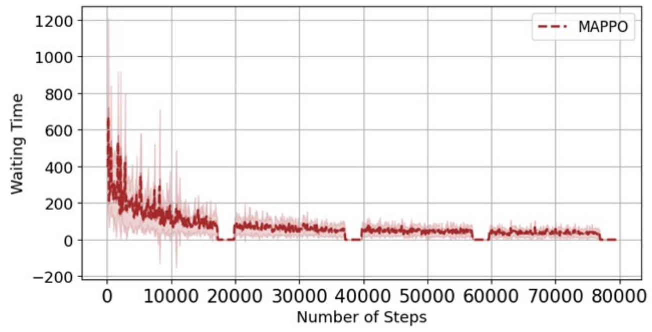

Fig. 17. PPO result of the multiple agent.

D. *What can we see?*

DQN again successfully converges after a decent amount of learning, but is again worse than A2C and PPO (look at the y axis' scale).

As for A2C and PPO, not only we succeed at solving this problem, the waiting time for all cars in all intersections combined is the same as it was in the single agent problem.

E. *How is this possible?*

The cause of this is the fact that in single agent, the agent has no idea where the traffic will come from. In Multi Agent

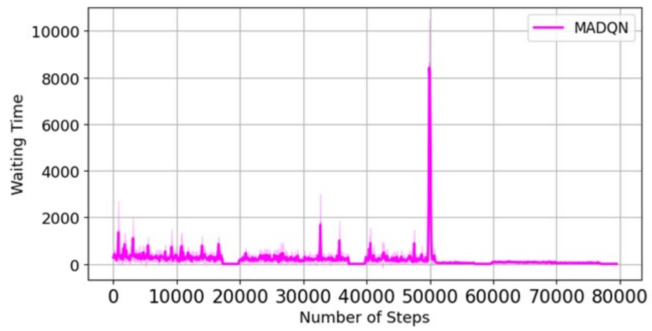

Fig. 18. DQN result of the multiple agent.

we have a big advantage – the agents communicate, and tell each other – "Cars are coming from your side, be aware". This way the Multi Agent problem handles far more cars and intersections, but can maintain similarly low total waiting times. After seeing the great result with 1 2 intersections, we can look at the real world, where there's a net of intersections, all connected to each other, where cars flow in every direction.

XI. Conclusion

This research demonstrates the application of self-play training to improve the logical reasoning capabilities of large language models (LLMs) in autonomous driving systems. By simulating interactions between multiple virtual agents (vehicles) in controlled environments, self-play enables LLMs to iteratively refine their decision-making and coordination strategies. These simulated interactions allow LLMs to explore complex multi-agent scenarios, such as negotiating lane changes, resolving traffic conflicts, or optimizing route planning, without relying on large-scale labeled datasets.

Through self-play, LLMs gain a deeper understanding of contextual reasoning and dynamic adaptation. This enhanced logical ability translates into more effective vehicle behavior predictions and better management of cooperative and competitive driving scenarios. Additionally, self-play training aligns the objectives of individual agents (vehicles) with broader system-wide goals, promoting collaboration and improving traffic flow.

The benefits of self-play extend beyond the LLM itself. By incorporating self-play-trained LLMs into autonomous driving systems, the study observes significant improvements in vehicle safety and efficiency. Autonomous vehicles trained with LLMs demonstrate smoother navigation, reduced collision risks, and more efficient use of road infrastructure. These outcomes align with the study's broader objective of creating smarter, safer, and more efficient transportation networks using cutting-edge AI technologies.

In conclusion, the use of self-play training enhances the logical reasoning capabilities of LLMs, making them more adept at handling complex autonomous driving tasks. This, in turn, improves the efficiency and safety of autonomous vehicles, paving the way for advanced intelligent transportation systems and cooperative autonomous driving solutions in urban environments.

Acknowledgments

This should be a simple paragraph before the References to thank those individuals and institutions who have supported your work on this article.

References

[1] J. Wu, Y. Yang, X. Cheng, H. Zuo and Z. Cheng, "The Development of Digital Twin Technology Review," 2020 Chinese Automation Congress (CAC), Shanghai, China, 2020, pp. 4901-4906, doi: 10.1109/CAC51589.2020.9327756.

[2] MarkVollrath. Schleicher. Susanne. Gelau. Christhard. 19 December 2010. The influence of Cruise Control and Adaptive Cruise Control on driving behaviour -A driving simulator study. Accident Analysis & Prevention. 43. 3. 1134–1139. 10.1016/j.aap.2010.12.023. 21376911. 0001-4575.

[3] How Pre-Collision Systems Work, 2022. [Online]. Available: https://auto.howstuffworks.com/car-driving-safety/safety-regulatory-devices/pre-collision-systems.htm

[4] Habenicht, Stefan; Winner, Hermann; Bone, Sven; Sasse, Fabian; Korzenietz, Peter (5 July 2011). "A maneuver-based lane change assistance system". 2011 IEEE Intelligent Vehicles Symposium (IV): 375–380. doi:10.1109/IVS.2011.5940417. ISBN 978-1-4577-0890-9. S2CID 9690965.

[5] Arena, Fabio; Pau, Giovanni (2019-01-24). "An Overview of Vehicular Communications". Future Internet. 11 (2): 27. doi:10.3390/fi11020027. ISSN 1999-5903.

[6] Sayed, I., Abdelgawad, H. & Said, D. Studying driving behavior and risk perception: a road safety perspective in Egypt. J. Eng. Appl. Sci. 69, 22 (2022). https://doi.org/10.1186/s44147-021-00059-z

[7] C. Wu et al., "Framework for control and deep reinforcement learning in traffic," 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017, pp. 1-8, doi: 10.1109/ITSC.2017.8317694.

[8] Song, Yaofeng & Zhao, Han & Luo, Ruikang & Huang, Liping & Zhang, Yicheng & Su, R.. (2022). A SUMO Framework for Deep Reinforcement Learning Experiments Solving Electric Vehicle Charging Dispatching Problem. 10.48550/arXiv.2209.02921.

[9] Baidu Apollo, 2022. [Online]. Available: https://github.com/ApolloAuto/apollo

[10] Tonguz, Ozan K. (25 Sep 2018). "How Vehicle-to-Vehicle Communication Could Replace Traffic Lights and Shorten Commutes". IEEE Spectrum.

[11] Chowdhury, D., Santen, L., & Schadschneider, A. (2000). Statistical physics of vehicular traffic and some related systems. Physics Reports, 329(4-6), 199-329.

[12] Information Resources Management Association. Networking and Telecommunications: Concepts, Methodologies, Tools, and Applications. Idea Group Inc (IGI); ISBN 978-1-60566-987-8. p. 592.

[13] Christoph Sommer, Reinhard German and Falko Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," IEEE Transactions on Mobile Computing (TMC), vol. 10 (1), pp. 3-15, January 2011.

[14] "Unity 2022.1a". Unity. Retrieved November 5, 2021.

[15] Harvey J. Miller and Shih-Lung Shaw (2001). Geographic Information Systems for Transportation. Oxford University Press. ISBN 978-0-19-512394-4.

[16] "Cellular V2X as the Essential Enabler of Superior Global Connected Transportation Services". IEEE 5G Tech Focus. IEEE. 1 (2). June 2017.

[17] Morgan, Y. L. (2010). Notes on DSRC & WAVE standards suite: Its architecture, design, and characteristics. IEEE Communications Surveys & Tutorials, 12(4), 504-518.

[18] TensorFlow, 2022. [Online]. Available: https://www.tensorflow.org

[19] Keras, 2022. [Online]. Available: https://keras.io

[20] Unity ML-Agents, 2022. [Online]. Available: https://github.com/Unity-Technologies/ml-agents.