DYNA-STYLE LEARNING WITH A MACROSCOPIC MODEL FOR VEHICLE PLATOONING IN MIXED-AUTONOMY TRAFFIC

Yichuan Zou
Shanghai Jiao Tong University
zouyichuan@sjtu.edu.cn

Li Jin
Shanghai Jiao Tong University
li.jin@sjtu.edu.cn

Xi Xiong
Tongji University
xi_xiong@tongji.edu.cn

ABSTRACT

Platooning of connected and autonomous vehicles (CAVs) plays a vital role in modernizing highways, ushering in enhanced efficiency and safety. This paper explores the significance of platooning in smart highways, employing a coupled partial differential equation (PDE) and ordinary differential equation (ODE) model to elucidate the complex interaction between bulk traffic flow and CAV platoons. Our study focuses on developing a Dyna-style planning and learning framework tailored for platoon control, with a specific goal of reducing fuel consumption. By harnessing the coupled PDE-ODE model, we improve data efficiency in Dyna-style learning through virtual experiences. Simulation results validate the effectiveness of our macroscopic model in modeling platoons within mixed-autonomy settings, demonstrating a notable 10.11% reduction in vehicular fuel consumption compared to conventional approaches.

1 Introduction

Traffic congestion presents a formidable obstacle, arising when traffic demand surpasses road capacity, leading to prolonged wait times and heightened fuel consumption due to frequent acceleration and deceleration maneuvers [1]. Highways, with their inherent complexities like on-ramps and lane reductions, often experience exacerbated congestion scenarios. In response, platooning—coordinated movement of connected and autonomous vehicles (CAVs)—emerges as a promising strategy for optimizing traffic flow and mitigating congestion.

This study focuses on leveraging vehicle platooning to alleviate traffic congestion at bottlenecks in mixed autonomy settings, as illustrated in Fig. 1. Platooning management encompasses two main paradigms: model-based and model-free approaches. Model Predictive Control (MPC) stands out as a prevalent model-based method, offering flexibility and adaptability in platoon management. Studies by Wang et al. [19], Feng et al. [4], and Hu et al. [6] underscore MPC's efficacy in optimizing platoon behavior while ensuring fuel efficiency and minimal traffic disruption across varying environmental conditions. Another model-based approach involves the use of optimal control techniques like Linear Quadratic Regulator (LQR) and Model Predictive Path Integral Control (MPPI), which have shown promising results in platoon management. Research by Stankovic et al. [16] and Uhlemann [18] demonstrates the effectiveness of these methods in optimizing platoon behavior and reducing fuel consumption. Additionally, reinforcement learning (RL) offers a model-free approach, where agents interact with their environment to learn optimal action policies. Notably, Li et al. [9] and Liu et al. [11] showcase RL's effectiveness in platoon control, emphasizing its ability to learn policies that maximize cumulative rewards, thereby optimizing traffic flow and reducing fuel consumption in complex traffic scenarios.

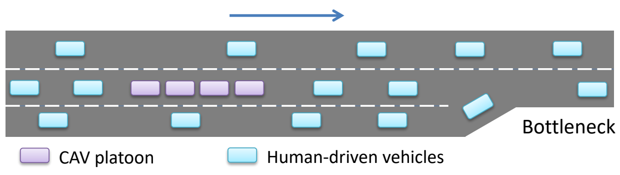


Figure 1: The mixed-autonomy traffic through a bottleneck on highway segment.

In this work, we present a novel contribution to the field by introducing a Dyna-style learning framework rooted in a first-order traffic flow model, aimed at effectively managing platoons and reducing vehicular fuel consumption during lane reductions. At the core of our approach lies the traditional macroscopic Lighthill-Whitham-Richards (LWR) model [10, 15], which consists of partial differential equations (PDEs) capturing traffic flow dynamics and ordinary differential equations (ODEs) tracking platoon trajectories. Through parameterization and Kalman filter updates [7, 20], our traffic model dynamically adapts to the environment, enabling accurate representation of forward dynamics. Unlike conventional neural network-based models prevalent in model-based RL, our parameterized traffic model offers superior interpretability and requires fewer parameters, owing to its foundation in physical fluid flow principles.

Furthermore, we evaluate our proposed framework within the Simulation of Urban MObility (SUMO) platform [2]. Results demonstrate that our framework enhances data efficiency and accelerates training compared to model-free learning algorithms. Under our framework's policy control, fuel consumption significantly decreases compared to the Krauss model [8], the default car-following model in SUMO.

The rest of this paper is organized as follows. Section 2 delves into the PDE-ODE traffic model, detailing its update and prediction mechanisms. Section 3 introduces our Dyna-style planning and learning framework. In Section 4, we meticulously evaluate our model's accuracy and policy performance. Finally, Section 5 summarizes our key findings and outlines future research directions.

2 Forward dynamics with a macroscopic model

2.1 The coupled PDE-ODE model

In this section, we model traffic flow on a multi-lane highway using a hyperbolic partial differential equation (PDE) based on the LWR model. Trajectories of platoons are tracked with an ordinary differential equation (ODE), assuming a fixed platoon length. Here, $\rho(x,t)$ represents vehicle density, $F(t,x,\rho)$ is the flow function, and $v = v(\rho)$ is the average speed. The platoon's downstream endpoint is denoted by $z_d(t)$, with initial conditions $\rho_0(x)$ and z_d^0 [12]. The model is summarized as:

$$\partial_t \rho + \partial_x F(t,x,\rho) = 0, \tag{1a}$$
$$\rho(0,x) = \rho_0(x), \tag{1b}$$
$$\dot{z}_d(t) = v_d(t, \rho(t, z_d(t)+)), \tag{1c}$$
$$z_d(0) = z_d^0. \tag{1d}$$

We adopt a linear density-speed relation expressed as:

$$v(\rho) = V\left(1 - \frac{\rho}{R}\right), \tag{2}$$

where V represents the maximum speed of vehicles, and R denotes the maximal reachable density on the road. The flow function $F(t,x,\rho)$ is defined as $f(\rho) = V\rho\left(1 - \frac{\rho}{R}\right)$. To determine the speed of the downstream point, we establish:

$$v_d(t,\rho) = \min\{V_d(t), v(\rho)\},$$

where $V_d(t) \in [0, V]$ represents the maximum speed of $z_d(t)$. The platoon is considered as a rigid body which has a fixed length. The speed of overall platoon is defined as v_p, and we have $v_p = v_d$.

To solve the PDE in Eq. (1a), we discretize both time and space, sampling with a time interval Δt and dividing the highway into cells with length Δx. The time interval and cell length should satisfy the Courant-Friedrichs-Lewy (CFL) condition, ensuring stability: $2V\Delta t \leq \Delta x$ [3]. The update of density in each cell i at time step k is given by:

$$\rho_i(k+1) = \rho_i(k) - \frac{\Delta t}{\Delta x}\left(F_{i+1/2}(k) - F_{i-1/2}(k)\right), \tag{3}$$

where $F_{i+1/2}$ represents the flow between cell i and cell $i+1$, computed using demand and supply functions. The demand and supply of a cell are computed as follows:

$$D_i(k) = \begin{cases} f(\rho_i(k)) & \text{if } \rho_i(k) < \rho_{cr}, \\ f_{\max} & \text{if } \rho_i(k) \geq \rho_{cr}, \end{cases} \qquad S_i(k) = \begin{cases} f_{\max} & \text{if } \rho_i(k) < \rho_{cr}, \\ f(\rho_i(k)) & \text{if } \rho_i(k) \geq \rho_{cr}, \end{cases}$$

where $\rho_{cr} = 0.5R$ denotes the critical density and $f_{\max}$ is the maximal flow between cells. The flow between cells is determined as the minimum between the demand of the upstream cell and the supply of the downstream cell:

$$F_i(k) = \min\{D_{i-1}(k), S_i(k)\}.$$

2.2 One-step prediction with Kalman filter

Kalman filtering is a powerful technique for estimating the true state of a system in the presence of noisy measurements [7]. The Kalman filter framework is structured as follows:

$$\hat{\phi}_k^{\ominus} = F_{k-1}\hat{\phi}_{k-1}^{\oplus} + G_{k-1}u_{k-1}, \tag{5a}$$
$$P_k^{\ominus} = F_{k-1}P_{k-1}^{\oplus}F_{k-1}^T + Q_{k-1}, \tag{5b}$$
$$K_k = P_k^{\ominus}h_k\left(r_k + h_k^{\top}P_k^{\ominus}h_k\right)^{-1}, \tag{5c}$$
$$P_k^{\oplus} = \left(I - K_kh_k^{\top}\right)P_k^{\ominus}, \tag{5d}$$
$$\hat{\phi}_k^{\oplus} = \hat{\phi}_k^{\ominus} + K_k\left(y_k - h_k^{\top}\hat{\phi}_k^{\ominus}\right), \tag{5e}$$

where $\hat{\phi}_k^{\ominus}$ represents the predicted state at step k, incorporating the dynamics of the system F_k and any control inputs $G_{k-1}u_{k-1}$ from the previous updated state $\hat{\phi}_{k-1}^{\oplus}$. The covariance matrix $P_k^{\ominus}$ captures the uncertainty associated with the predicted state, accounting for both the uncertainty carried over from the previous step $P_{k-1}^{\oplus}$ and the process noise Q_{k-1}. The Kalman gain K_k adjusts the predicted state based on the discrepancy between the observed state y_k and the predicted state, weighted by the measurement noise r_k and the transformation matrix h_k. Updating the covariance matrix via $P_k^{\oplus}$ refines the uncertainty estimate, adjusting it based on the Kalman gain and the measurement transformation. Finally, the updated state $\hat{\phi}_k^{\oplus}$ is computed by adjusting the predicted state with the Kalman gain, aligning it with the observed measurements.

In our project, we aim to develop a comprehensive predictive model for the simulation environment, enabling the forecasting of both traffic density and average speed in the subsequent time step based on current information. This model consists of two primary components: firstly, predicting density in the next time step by utilizing a traffic flow model, where we calculate the flow rate of each cell using demand and supply functions, and then applying a recursion formula derived from the PDE represented by Eq. (3) to update the density iteratively. Secondly, we focus on predicting speed in the next time step by leveraging the projected density from the previous step and coupling it with appropriate speed models.

For the second part, building on the PDE-ODE model's insight into the linear relationship between density and speed, we approach the task by treating V (maximal speed) and R (maximal density) as parameters that require adaptation to fit the environment. To achieve this, we employ a Kalman filter to iteratively update these parameters. In this adaptation, the state ϕ in the Kalman filter represents a two-dimensional vector encapsulating V and R. Given that maximal speed and maximal density are inherent attributes of road sections, the status transition matrix in Eq. (5a) simplifies to the identity matrix ($F_k = I$), and the control matrix becomes a zero matrix ($G_k = 0$), with no process noise ($Q_k = 0$) introduced. For simplicity, we define the weight r_k as 1. Consequently, the Kalman filter reduces to the following form:

$$K_k = P_{k-1}h_k\left(1 + h_k^{\top}P_{k-1}h_k\right)^{-1}, \tag{6a}$$
$$P_k = \left(I - K_kh_k^{\top}\right)P_{k-1}, \tag{6b}$$
$$\phi_k = \phi_{k-1} + K_k\left(y_k - h_k^{\top}\phi_{k-1}\right). \tag{6c}$$

Consider a specific road cell where we observe density ρ_k and average speed v_k at step k in the simulation process. Notably, the observed speed coincides with the actual cell speed. In Eq. (6c), we define $y_k = v_k$. R_{k-1} and V_{k-1} denote road attributes obtained from the previous step. By defining the linear transformation $h_k = [1, -\rho_k]$ and initializing the status $\phi_{k-1} = [V_{k-1}, \frac{V_{k-1}}{R_{k-1}}]$, we can predict the average speed of the cell by the inner product of these two vectors:

$$\overline{v}_k = h_k^{\top}\phi_{k-1} = V_{k-1}(1 - \frac{\rho_k}{R_{k-1}}).$$

Following the update in Eq. (6c), we acquire an updated parameter set $\phi_k = [V_k, \frac{V_k}{R_k}]$. By iterating through the steps of the simplified Kalman filter, these model parameters undergo continuous refinement and updating, ensuring the model's adaptability and accuracy in predicting the dynamics of the simulation environment.

To enhance the accuracy and flexibility of our model, we can adapt it to accommodate varying road conditions, particularly in regions with bottlenecks. This involves dividing the entire road into several sections of equal length and employing separate parameters for each section, thereby tailoring the model to the specific characteristics of each segment. With the status of CAV platoons denoted as $[x_p(k), v_p(k)]$, alongside cell density $\rho_i(k)$ and cell speed $\overline{v}_i(k)$ at step k, the prediction process for step $k+1$ unfolds as follows:

1. Platoon status prediction:

$$v_p(k+1) = v_p(k) + a_p(k) * \Delta t,$$
$$x_p(k+1) = x_p(k) + v_p(k) * \Delta t + \frac{1}{2} a_p(k) * \Delta t.$$

2. Cell density prediction:

$$F_i(k) = \min \left\{ D_{i-1}(k), S_i(k) \right\},$$
$$\rho_i(k+1) = \rho_i(k) - \frac{\Delta t}{\Delta x} \left(F_{i+1/2}(k) - F_{i-1/2}(k) \right),$$

3. Cell speed prediction:

$$\overline{v_i}(k+1) = V_n \left(1 - \frac{\rho_i(k+1)}{R_n}\right),$$

where i represents the cell index and n denotes the segment index. This segmentation approach enables our model to adapt more effectively to diverse road conditions.

3 Dyna-Style Planning and Learning

In this section, we introduce a Markov Decision Process (MDP) framework for managing vehicle platoons within a mixed autonomy setting. Leveraging a Dyna-style approach, our framework seamlessly integrates model-free learning with model-based planning to govern platoon behavior. The model-based planning component relies on insights gleaned from the coupled PDE-ODE model outlined in the preceding section, allowing for a comprehensive understanding of platoon dynamics and informed decision-making.

3.1 Markov decision process

The CAV platoon aims to learn an optimal action policy that minimizes fuel consumption within the simulation environment. To formalize this objective, we define a Markov Decision Process (MDP) model denoted as $M = \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma$. Here, $\mathcal{S}$ represents the set of states, $\mathcal{A}$ denotes the set of actions, and $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to [0, 1]$ characterizes the dynamics determined by the simulation environment. At a given state $s \in \mathcal{S}$, the platoon selects an action $a \in \mathcal{A}$ based on its policy and transitions to a new state $s' \in \mathcal{S}$ while receiving a reward $r \in \mathcal{R}$.

State The state is represented by a vector capturing various aspects of the environment. Assuming the entire road section is divided into N cells, each with a length of L meters, the state vector includes dimensions for platoon position and speed, followed by densities and average speeds of all cells. This yields a state vector with $2 + 2N$ dimensions:

$$s_t = [x_p, v_p, \rho_1, ..., \rho_N, \bar{v}_1, ..., \bar{v}_N].$$

Action The platoon adjusts its acceleration as its action. The action space is discrete: $a_t \in \{a | a_{\min} \leq a \leq a_{\max},\ a \in \mathbb{N}\}$. If an action results in the platoon's speed becoming negative or exceeding the maximum speed, the platoon's speed is constrained to remain within the allowed range.

Reward The platoon's primary objective is to traverse the entire road section while minimizing total vehicle fuel consumption. The average fuel consumption rate (L/s) is computed using a sixth-order polynomial based on the average vehicle speed, as described in [14]. The polynomial equation is:

$$k_t(v) = 5.7 \cdot 10^{-12} \cdot v^6 - 3.6 \cdot 10^{-9} \cdot v^5 + 7.6 \cdot 10^{-7} \cdot v^4 - 6.1 \cdot 10^{-5} \cdot v^3 + 1.9 \cdot 10^{-3} \cdot v^2$$
$$+ 1.6 \cdot 10^{-2} \cdot v + 0.99,$$

From this equation, the total fuel consumption can be derived. In each time step, the platoon incurs a negative total fuel consumption penalty r_{fc}. Additionally, the platoon receives a fixed reward r_{bonus} each time it enters the next road cell or reaches the end of the entire road section. To ensure the platoon completes the entire road section within a limited time steps, it incurs a timeout penalty r_{ot} based on the platoon's position. An additional penalty r_{acc} is applied to the platoon based on its acceleration. Consequently, the total reward is computed as $r_t = r_{fc} + r_{bonus} + r_{ot} + r_{acc}$.

3.2 Dyna-Q learning with the macroscopic model

Dyna-Q stands as a classical model-based reinforcement learning algorithm, utilizing Q-planning to generate experiences from a model and leveraging these experiences to update the policy [17]. While Dyna-Q represents a foundational approach in model-based reinforcement learning, our algorithm extends this framework by incorporating a novel parameterized PDE-ODE traffic model, a departure from the typical neural network-based models prevalent in contemporary research. This approach offers enhanced interpretability and requires fewer parameters compared to neural networks.

In our Dyna framework, actions are selected using an ϵ-greedy policy with the Q network Q_θ. At each step, the agent transitions to the next state based on the selected action, with state transitions stored in replay memory $\mathcal{D}$. Simultaneously, the Q network Q_θ is updated with the TD error, while the parameters of the traffic model W_ϕ are updated through a Kalman filter in in Eq. (6). These parameters $\phi = [R_1, ..., R_n, V_1, ..., V_n]$ encapsulate the maximum density and cell speed of all segments. Following the model update, W_ϕ predicts the next state and reward using sampled state-action pairs, and Q_θ is updated accordingly using these virtual experiences. The framework of our algorithm is presented in Algorithm 1.

Algorithm 1 Dyna-Q with the PDE-ODE model

```
1:  Initialize networks and model: $Q_\theta, Q_{\theta'}, W_\phi$
2:  Initialize replay memory: $\mathcal{D}$
3:  for episode = 1, M do
4:      Initialize state $\mathbf{s}_0$.
5:      for $t = 1, T$ do
6:          Choose action $\mathbf{a}_t$ using $\epsilon$-greedy($\mathbf{s}_t$, $Q_\theta$)
7:          $\mathbf{r}_t,\ \mathbf{s}_{t+1} \leftarrow$ take action $\mathbf{a}_t$ at state $\mathbf{s}_t$
8:          $\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{s}_{t+1})\}$
9:          Sample state transitions: $(\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{s}') \sim \mathcal{D}$
10:         Update $Q_\theta$ with TD error: $L(\theta) = (Q_\theta(\mathbf{s}, \mathbf{a}) - (\mathbf{r} + \max_{a'} Q_{\theta'}(\mathbf{s}', \mathbf{a}')))^2$
11:         Update $W_\phi$ using the Kalman filter in Eq. (6)
12:         for $c = 1, N$ do
13:             Sample state-action pairs: $(\mathbf{s}, \mathbf{a}) \sim \mathcal{D}$
14:             $\mathbf{r}_m,\ \mathbf{s}'_m \leftarrow W_\phi(\mathbf{s}, \mathbf{a})$
15:             Update $Q_\theta$ with TD error: $L(\theta) = (Q_\theta(\mathbf{s}, \mathbf{a}) - (\mathbf{r}_m + \max_{a'} Q_{\theta'}(\mathbf{s}'_m, \mathbf{a}')))^2$
16:         Update target network every $C$ steps: $\theta' \leftarrow \theta$
```

4 Numerical results

In this section, we provide an overview of our simulation environment setup. Subsequently, we evaluate the performance of the macroscopic model and Dyna-style policy within this environment. We delve into each aspect in the subsequent sections, systematically assessing the effectiveness and efficiency of our approach.

4.1 Simulation settings

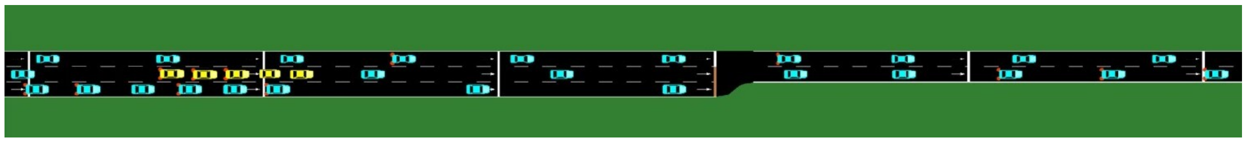


Figure 2: The road section is divided into three sections: a preheating section, a three-lane section, and a two-lane section. The bottleneck occurs at the junction between the three-lane and two-lane sections.

In the SUMO simulation environment, we construct a road section spanning a total length of 1000 meters, comprising three distinct segments: a 200-meter preheating section, a 500-meter three-lane section, and a 300-meter two-lane section. As depicted in Fig. 2, the bottleneck emerges at the junction between the three-lane and two-lane sections, where congestion is anticipated to occur.

In this environment, we distinguish between two types of vehicles: CAV platoon and human-driven vehicles. The platoon is considered a distinct entity with a fixed length, longer than standard vehicles. Specifically, the lengths of the platoon and human-driven vehicles are set to 30 meters and 5 meters, respectively. Both platoon and human-driven vehicles operate with a maximum speed of 15 meters per second, while the platoon's acceleration ranges from -5 to 3 meters per second squared. The flow rate of human-driven vehicles is maintained at 3600 per hour. The platoon enters the preheating section at the 100 seconds, operating at maximum speed. Upon entering the three-lane section, our approach assumes control of the platoon's operations.

4.2 Dyna-Q settings

Based on the Courant-Friedrichs-Lewy (CFL) condition $2V\Delta t \leq \Delta x$ [5], we establish the time interval and length of the road cell as $\Delta t = 1$ second and $\Delta x = 50$ meters, respectively, with the maximum vehicle speed set to $V = 15$ m/s. Consequently, the total number of road cells is 16, resulting in a state vector comprising 34 dimensions.

The entire road section is divided into 8 segments, each containing two cells and spanning 100 meters in length. For each segment, two parameters are defined: the maximum density R and the maximum speed V. Initially, the maximum speed for all segments is uniformly set to 15 m/s, consistent with the state space. The initial maximum density for segments with three lanes is set to 20 per cell, while two-thirds of this value is assigned to segments with two lanes. The covariance matrix P is initialized as an identity matrix with corresponding dimensions.

For Dyna-Q learning, we utilize an Adam optimizer with a learning rate of 0.0001 to update the Q network. The size of the replay buffer is set to 300,000, where new experiences replace old ones when the buffer is full. During each training iteration, batches of experiences with a size of 64 are sampled from the replay buffer. Actions are chosen according to an ϵ-greedy policy. Initially, within the first 20,000 training steps, ϵ gradually increases to its maximum value of 0.95. The discount factor is set to 0.99.

4.3 Evaluation of the PDE-ODE model

We assess the the PDE-ODE model's accuracy in two key aspects: density prediction and speed prediction. We enable the trained platoon to interact with the SUMO environment and compare the model's predictions with the true states, recording the error at each time step. This evaluation allows us to gauge the model's performance and identify areas for improvement.

Utilizing our approach, the agent spends over 200 time steps interacting with the SUMO environment. Table 1 presents the average prediction errors for all cells throughout the simulation process. Notably, the density prediction proves remarkably accurate, with the average absolute error between predicted and true density remaining under 0.12 veh/cell. Interestingly, the density error slightly decreases towards the end of the episode compared to the beginning. Furthermore, we examine the speed prediction error for both untrained parameters and those updated with the Kalman filter, depicted in Fig. 3. Notably, the error peaks around 100 and 200 time steps for both parameter sets, albeit significantly higher for untrained parameters. Throughout the episode, the absolute speed error remains below 1.5 m/s for parameters updated with the Kalman filter, whereas untrained parameters consistently yield errors exceeding 2 m/s, with some instances even surpassing 4 m/s. Clearly, post-training, the speed prediction error undergoes substantial reduction, signifying a marked improvement in model accuracy.

	Absolute error	Relative error
Density	0.12 veh/cell	0.6%
Speed	1.5 m/s	10%

Table 1: Average absolute and relative prediction errors.

The speed prediction, although not as accurate as density prediction, remains a valuable tool in our approach. This discrepancy arises from the limitations of using a first-order linear model to capture the relationship between density and speed, rendering it unable to fully capture complex road conditions such as traffic jams. However, we demonstrate that despite this limitation, an approximately accurate model can still facilitate the training of the agent towards learning a more effective control policy.

4.4 Performance comparison

By integrating the PDE-ODE model updated through the Kalman filter with DQN, we develop a model-based algorithm akin to Dyna-Q. To enable a comprehensive comparison, we utilize the original DQN as a baseline method. Both approaches are employed to train agents for 200 thousand steps across 3 random seeds. The training curves of both methods are depicted in Fig. 4. Rewards lower than -3000 indicate that the agent is penalized for timeouts. From this figure, we observe that the agent trained with Dyna-Q starts to mitigate these penalties at around 20,000 training

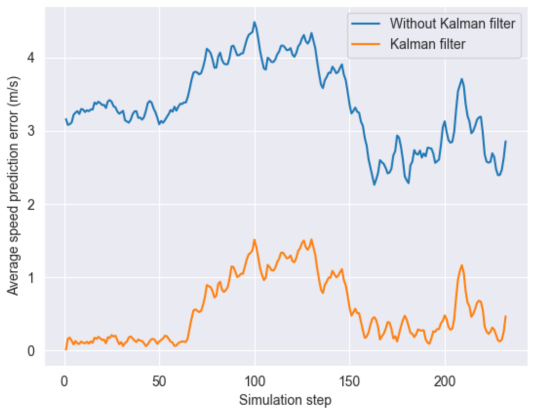


Figure 3: Average speed prediction errors for all cells using Kalman filter-trained parameters and untrained parameters.

steps. Subsequently, the Dyna-Q agent gradually converges to a policy characterized by higher rewards and lower fuel consumption. Conversely, the DQN agent fails to converge to a stable policy within 100,000 training steps. This illustrates that our algorithm, incorporating the density-speed model, enhances training efficiency and facilitates faster convergence towards an optimal control policy.

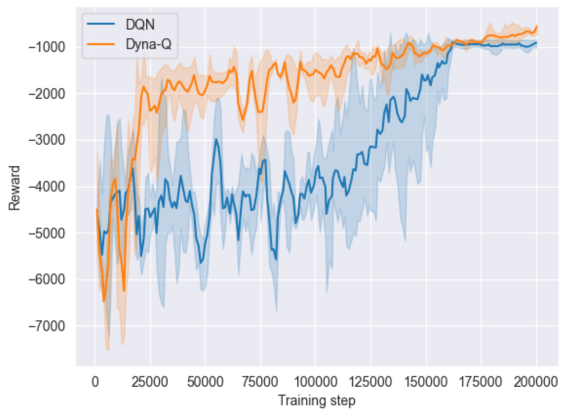


Figure 4: A comparison between the reward curves of the DQN and Dyna-Q methods, where Dyna-Q incorporates the macroscopic model.

The Krauss model serves as a typical car-following model in SUMO settings [13]. In our setup, we employ the Krauss model to control the vehicle platoon and consider it as a benchmark. The total fuel consumption rates of the two control methods are depicted in Fig 5. Throughout the entire driving process, the fuel consumption rate of the Dyna-Q policy control consistently remains lower than that of the Krauss model control. Specifically, the rewards for the Dyna-Q agent and Krauss model are respectively -496.07 and -1052.04. Excluding the reward for entering the next cell and acceleration penalty, the total fuel costs through conversion for Dyna-Q policy control and Krauss model control are respectively 395.60 L and 440.12 L, resulting in a reduction in fuel consumption by approximately 10.11%.

Traffic density profiles under both methods are illustrated in Fig 6. In Fig.6(a), the Krauss model control leads to a significant congestion scenario. Notably, cells numbered from 4 to 7, corresponding to the road section between 200 m and 350 m, exhibit densities surpassing 10 vehicles per cell within the initial 65% of the total time, with a peak density reaching about 17 vehicles per cell. Conversely, Fig.6(b) showcases a notable reduction in congestion length and duration under the Dyna-Q policy control. Here, congestion only persists between cell 6 and cell 7 within the initial 30% of the total time, with a peak density of approximately 14 vehicles per cell—lower than observed under the Krauss

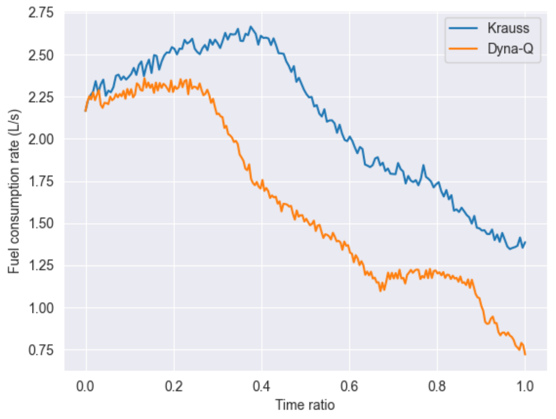
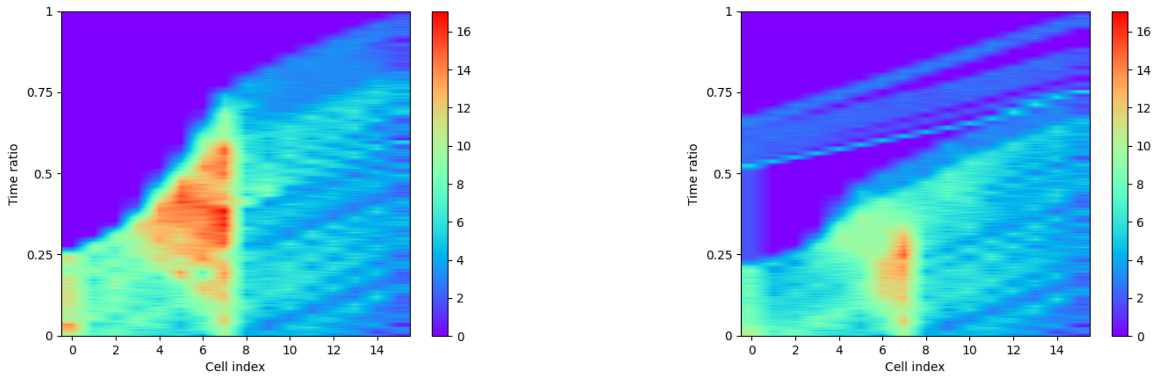


Figure 5: Fuel consumption rates throughout the simulation under the Krauss model and Dyna-Q.

model. These results underscore the capacity of the Dyna-Q policy not only to curtail total fuel consumption but also to alleviate congestion upstream of the bottleneck.



(a) Control with the Krauss model (b) Control with our Dyna-Q framework

Figure 6: Traffic density profiles under two control methods.

5 Concluding remarks

This paper introduces a novel approach to reduce fuel consumption of vehicles at highway bottlenecks in mixed autonomy traffic through the control of vehicle platoons. We propose a Dyna-style learning framework integrating a coupled PDE-ODE model. Additionally, we enhance the PDE-ODE model's predictive capabilities with a Kalman filter to estimate density and average speed for each road cell. Using the SUMO simulation, we evaluate the efficacy of our approach. Furthermore, we assess the performance of the Dyna-style planning and learning against traditional car-following methods. Our results demonstrate a reduction in total fuel consumption by approximately 10%. Moreover, our method exhibits faster convergence to an optimal policy compared to model-free learning, owing to the incorporation of the macroscopic model.

This work can be extended in several directions. Firstly, considering the platoon as a rigid body with a fixed length suggests the potential for adopting a more realistic platoon model in the simulation. Additionally, the first-order traffic model employed here may be too simplistic for complex scenarios, indicating the need for more precise traffic models. Finally, future research could explore the utilization of advanced deep reinforcement learning algorithms beyond simple Dyna-Q.

References

[1] Richard Arnott and Kenneth Small. The economics of traffic congestion. *American scientist*, 82(5):446–455, 1994.

[2] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility. In *The Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain*, volume 42, 2011.

[3] Richard Courant, Kurt Friedrichs, and Hans Lewy. On the partial difference equations of mathematical physics. *IBM journal of Research and Development*, 11(2):215–234, 1967.

[4] Shuo Feng, Ziyou Song, Zhaojian Li, Yi Zhang, and Li Li. Robust platoon control in mixed traffic flow based on tube model predictive control. *IEEE Transactions on Intelligent Vehicles*, 6(4):711–722, 2021.

[5] Paola Goatin. Macroscopic traffic flow modelling: from kinematic waves to autonomous vehicles. *Communications in Applied and Industrial Mathematics*, 14(1):1–16, 2023.

[6] Manjiang Hu, Chongkang Li, Yougang Bian, Hui Zhang, Zhaobo Qin, and Biao Xu. Fuel economy-oriented vehicle platoon control using economic model predictive control. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):20836–20849, 2022.

[7] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[8] Stefan Krauß. Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics. 1998.

[9] Meng Li, Zehong Cao, and Zhibin Li. A reinforcement learning-based vehicle platoon control strategy for reducing energy consumption in traffic oscillations. *IEEE Transactions on Neural Networks and Learning Systems*, 32(12):5309–5322, 2021.

[10] Michael James Lighthill and Gerald Beresford Whitham. On kinematic waves ii. a theory of traffic flow on long crowded roads. *Proceedings of the royal society of london. series a. mathematical and physical sciences*, 229(1178):317–345, 1955.

[11] Tong Liu, Lei Lei, Kan Zheng, and Kuan Zhang. Autonomous platoon control with integrated deep reinforcement learning and dynamic programming. *IEEE Internet of Things Journal*, 10(6):5476–5489, 2022.

[12] Giulia Piacentini, Paola Goatin, and Antonella Ferrara. A macroscopic model for platooning in highway traffic. *SIAM Journal on Applied Mathematics*, 80(1):639–656, 2020.

[13] Mitra Pourabdollah, Eric Bjärkvik, Florian Fürer, Björn Lindenberg, and Klaas Burgdorf. Calibration and evaluation of car following models using real-world driving data. In *2017 IEEE 20th International conference on intelligent transportation systems (ITSC)*, pages 1–6. IEEE, 2017.

[14] Rabie A Ramadan and Benjamin Seibold. Traffic flow control and fuel consumption reduction via moving bottlenecks. *arXiv preprint arXiv:1702.07995*, 2017.

[15] Paul I Richards. Shock waves on the highway. *Operations research*, 4(1):42–51, 1956.

[16] Srdjan S Stankovic, Milorad J Stanojevic, and Dragoslav D Siljak. Decentralized overlapping control of a platoon of vehicles. *IEEE Transactions on Control Systems Technology*, 8(5):816–832, 2000.

[17] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[18] Elisabeth Uhlemann. Platooning: connected vehicles for safety and efficiency [connected vehicles]. *IEEE Vehicular Technology Magazine*, 11(3):13–18, 2016.

[19] Pangwei Wang, Hui Deng, Juan Zhang, Li Wang, Mingfang Zhang, and Yongfu Li. Model predictive control for connected vehicle platoon under switching communication topology. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):7817–7830, 2021.

[20] Xi Xiong, Kaan Ozbay, Li Jin, and Chen Feng. Dynamic origin–destination matrix prediction with line graph neural networks and kalman filter. *Transportation Research Record*, 2674(8):491–503, 2020.