[40] Meixin Zhu, Xuesong Wang, and Yinhai Wang. Human-like autonomous car-following model with deep reinforcement learning. *Transportation research part C: emerging technologies*, 97:348–368, 2018.

[41] Tianyu Shi, Yifei Ai, Omar ElSamadisy, and Baher Abdulhai. Bilateral deep reinforcement learning approach for better-than-human car following model. *arXiv preprint arXiv:2203.04749*, 2022.

[42] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[43] Meead Saberi and Hani S Mahmassani. Hysteresis and capacity drop phenomena in freeway networks: Empirical characterization and interpretation. *Transportation research record*, 2391(1):44–55, 2013.

[44] Katja Vogel. A comparison of headway and time to collision as safety indicators. *Accident analysis & prevention*, 35(3):427–433, 2003.

[45] Dale F Cooper and N Ferguson. Traffic studies at t-junctions. 2. a conflict simulation record. *Traffic Engineering & Control*, 17(Analytic), 1976.

[46] Peter De Haan and Mario Keller. Modelling fuel consumption and pollutant emissions based on real-world driving patterns: the hbefa approach. *International journal of environment and pollution*, 22(3):240–258, 2004.

[47] TJ Ayres, L Li, David Schleuning, and D Young. Preferred time-headway of highway drivers. In *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585)*, pages 826–829. IEEE, 2001.

[48] Yang Zheng, Jiawei Wang, and Keqiang Li. Smoothing traffic flow via control of autonomous vehicles. *IEEE Internet of Things Journal*, 7(5):3882–3896, 2020.

[49] Jiawei Wang, Yang Zheng, Qing Xu, Jianqiang Wang, and Keqiang Li. Controllability analysis and optimal control of mixed traffic flow with human-driven and autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 22(12):7445–7459, 2020.

APPENDIX

I. Car Following Filter

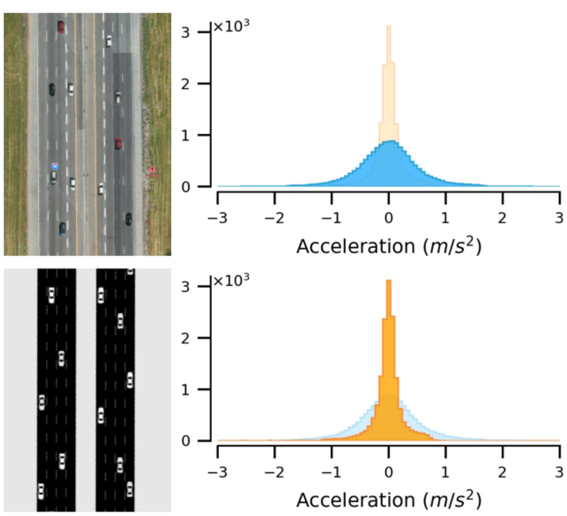

Fig. 3: Instantaneous accelerations observed during car-following behaviors at densities $[70, 150]$ veh/km. TOP: Real-world data from the I-24 MOTION dataset reveals a distribution having long tails extending to $[-3, 3]$ m/s^2. BOTTOM: IDM (in simulation) produces accelerations mostly within $[-0.5, 0.5]$ m/s^2, indicating much 'timid' driving behaviors than the real world.

We analyze the I-24 MOTION dataset [21] with study length $= 6.75$ km and study time $= 4$ h. The dataset contains various vehicle types such as semi-trailers, mid-sized trucks, motorbikes, and cars under different traffic conditions such as approaching standing traffic, lane changing, and free flow. To extract car-following trajectories, we select data points that meet the following criteria:

- Ego car is following another car, i.e., has a leader.
- Leader and ego cars are in the same lane ≥ 5 s.
- Ego car's speed is greater than 10% of the speed limit, i.e., not approaching stationary traffic.
- Ego car's space headway is less than 124 m, applying 4 s rule at the speed limit to avoid free flow conditions.

II. Model-based Robot Vehicles

Bilateral Control Module (BCM): BCM [9] uses information about both follower and leader vehicles to obtain a linear model whose acceleration is given by:

$$a = k_d \cdot \Delta_d + k_v \cdot (\Delta v_l - \Delta v_f) + k_c \cdot (v_{des} - v), \quad (1)$$

where Δ_d, Δv_l, Δv_f, v_{des}, and v, represent the difference in distance to the leader compared to the distance to the follower, the difference in velocity to the leader, the difference in velocity to the follower, the set desired velocity, and the current velocity of the vehicle, respectively. $k_d = 1$, $k_v = 1$, and $k_c = 1$ are gain parameters.

Linear Adaptive Cruise Control (LACC): LACC is an improvement on existing cruise control systems that allows vehicles to maintain a safe distance or speed without communication. The constant time-headway model by Rajamani [10] employs a first-order differential equation for approximation. The control acceleration at time t is given by:

$$a_t = (1 - \frac{\Delta t}{\tau}) \cdot a_{(t-1)} + \frac{\Delta t}{\tau} a_{cmd,(t-1)}, \quad (2)$$

$$a_{cmd} = k_1 \cdot e_x + k_2 \cdot \Delta v_l, \quad (3)$$

$$e_x = s - h \cdot v, \quad (4)$$

where $k_1 = 0.3$ and $k_2 = 0.4$ are design parameters, e_x is the gap error, s is the space headway, Δv_l is the velocity difference to the leader, $h = 1$ is the desired time gap, Δt is the control time-step, and $\tau = 0.1$ is the time lag of the system.

III. Heuristic-based Robot Vehicles

FollowerStopper (FS): FS [8] is an RV that travels at a fixed command velocity (target) under safe conditions but when required, slightly lowers the target velocity, opening up a gap to the vehicle ahead. This allows it to dampen oscillations and brake smoothly when needed. The command velocity is given by:

$$v_{cmd} = \begin{cases} 0, & \text{if } \Delta x \leq \Delta x_1 \\ v \frac{\Delta x - \Delta x_1}{\Delta x_2 - \Delta x_1}, & \text{if } \Delta x_1 < \Delta x \leq \Delta x_2 \\ v + (U - v) \frac{\Delta x - \Delta x_2}{\Delta x_3 - \Delta x_2}, & \text{if } \Delta x_2 < \Delta x \leq \Delta x_3 \\ U, & \text{if } \Delta x_3 < \Delta x \end{cases} \quad (5)$$

where $v = \min\left(\max\left(v_{\text{lead}}, 0\right), U\right)$ is the speed of the leader vehicle, Δx is the headway of the RV, and U is the desired velocity. The thresholds $(\Delta x_1,\ \Delta x_2,\ \Delta x_3)$ are defined as

$$\Delta x_k = \Delta x_k^0 + \frac{1}{2d_k}(\Delta v_-)^2, \quad k = 1, 2, 3. \quad (6)$$

The model parameters Δx_k^0, Δv_-, and d_k determine the spacing between vehicles and the RV's responsiveness to changes in velocity.

Proportional-integral with saturation (PIwS): PIwS [8] estimates the desired average velocity (U) of the vehicles in the network using its historical average velocity. The PIwS RV calculates the target velocity as

$$v_{target} = U + v_{catch} \times \min\left(\max\left(\frac{\Delta x - g_l}{g_u - g_l}, 0\right), 1\right), \quad (7)$$

which is used to calculate the command velocity at $t+1$ as

$$v_{cmd}^{t+1} = \beta_t(\alpha_t v_{target}^t + (1-\alpha_t)v_{lead}^t) + (1-\beta_t)v_{cmd}^t, \quad (8)$$

where v_{catch} is the catch-up velocity—a velocity higher than the average velocity allows the RV to catch up with its leader, Δx is the difference in position between the RV and its leader, g_l and g_u represent the lower and upper threshold distance, respectively; α_t and β_t represent the weight factors for target velocity v_{target} and command velocity v_{cmd}, respectively. Finally, v_{lead} represents the velocity of the leader vehicle.

IV. Reinforcement Learning (RL) Benchmarks

To benchmark with other RL techniques, we reproduce their original policies by following the provided experiment parameters and closely matching the performance. Specifically, to obtain RL policy with only local observations (RL + L), we follow Wu et al. [11]; to obtain RL with global observations (RL + G), we follow Vinitsky et al. [14]. Our reproduced RL + L achieves the performance within 1% error (measured with stabilization time and average velocity during stabilization) of the original work. Whereas for RL + G, our reproduction achieves the performance within 3% error (measured with outflow) of the original work. The precise implementations of the other RL methods validate our benchmarking experiments.

V. Congestion Stage Classifier (CSC)

One CSC each is trained in Ring and Bottleneck with independent datasets collected in the two environments. For each RV, the position and velocity of all vehicles in its local zone (set to 50 m) are collected. Fig. 4 shows the K-means clustering of collected data over all six classes ('Forming', 'Leaving', 'Congested', 'Free flow', 'Undefined', and 'No Vehicle') in both environments. As the data collected is sequential in nature and CSC predictions are made a number of time-steps into the future, a time offset of 10 time-steps is chosen to balance usefulness and accuracy (to illustrate, a prediction of congestion stage 100 time-steps in the future would be very useful, however, not very accurate; whereas a prediction of congestion state 1 time-step into the future can be very accurate but not very useful).

After windowing, the dataset includes instances where the congestion stage changes from t to $t + 10$, as well as instances where the congestion stage remains the same over the time window. To train CSC, we sample data to ensure a balanced representation of transition/non-transition instances as well as instances containing all six classes. Worth noting, the 'No vehicle' class presents a unique challenge. The collected data may contain instances changing from 'No vehicle' to another class after the 10 time-steps. However, based on the input corresponding to 'No vehicle' at t, we cannot predict the congestion stage at $t + 10$. Consequently, we discard data points where the 'No Vehicle' class transitions to another class after 10 time-steps. We replace this discarded data with synthetic examples that simulate various scenarios for the RV's position and velocity without leader vehicles.

The accuracy of the trained CSC is 95.5% in Ring and 85.2% in Bottleneck. The confusion matrix is shown in Fig. 5. CSC only observes downstream HVs in the same lane. Thus, when facing zipper lanes of Bottleneck (where traffic merges from adjacent lanes), the CSC cannot anticipate the merging traffic, resulting in lower accuracy in Bottleneck.

Category	Parameter	Value
Ring Simulation	Time Step (Δt)	0.1
	Simulation Horizon (T)	4500
	Warmup Time-steps	2500
	Speed Limit (m/s)	30
	Initial Speed (m/s)	0
Bottleneck Simulation	Time Step (Δt)	0.5
	Simulation Horizon (T)	1300
	Warmup Time-steps	100
	Speed Limit (m/s)	17
	Initial Speed (m/s)	6
	Inflow Rate (veh/hr)	3600
PPO Algorithm	Learning Rate (α)	0.00005
	Discount Factor (γ)	0.999
	GAE Estimation (λ)	0.97
	KL Divergence Target	0.02
	Entropy Coefficient Initial	0.1
	Entropy Coefficient Final	0.01
	Value Function Clip Param	20
	SGD Iterations	2
Ring CSC	Neural Network	$32, 16, 16$
	Batch Size	32
	Learning Rate	0.01
	Epochs	50
Bottleneck CSC	Neural Network	$32, 16, 16$
	Batch Size	256
	Learning Rate	0.001
	Epochs	100
Policy Networks	Our leader RV in Ring	$64, 32, 16$
	Our follower RV in Ring	$64, 32, 16$
	Our RV in Bottleneck	$32, 16, 8$
	RL + L	$32, 32, 32$
	RL + G	$256, 256$

TABLE III: Detailed experiment parameters. We show the simulation parameters of Ring and Bottleneck, as well as the parameters of Proximal Policy Optimization (PPO) and Congestion Stage Classifier (CSC). The hidden layer dimensions of various policy networks are also shown.

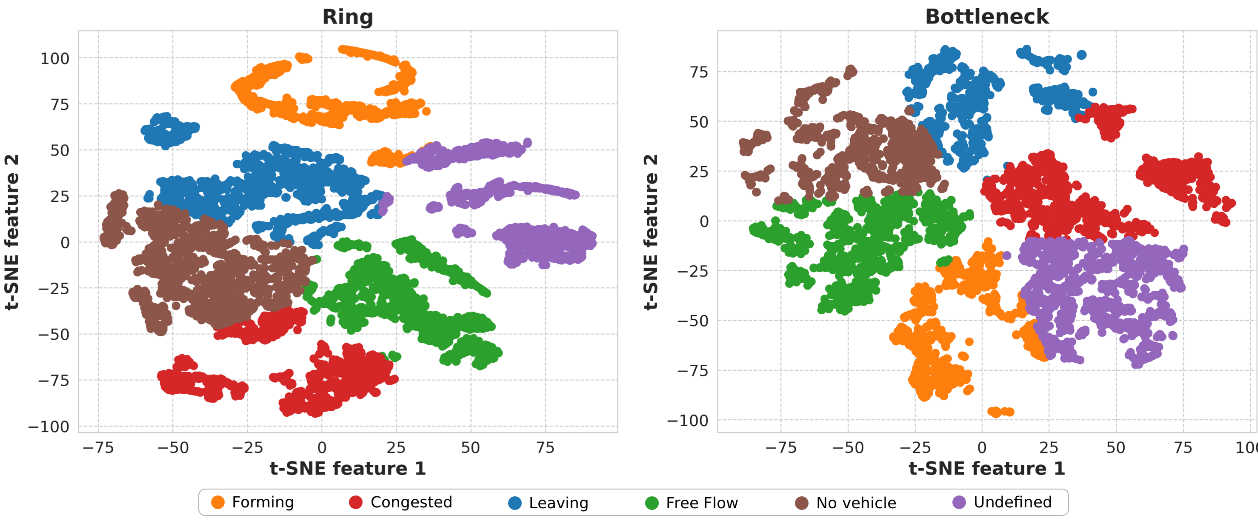

Fig. 4: The results of applying K-means clustering with t-SNE on a subset of CSC training data. LEFT: In Ring, the clusters are spread out, suggesting that the data is easily classifiable. RIGHT: In Bottleneck, overlapping clusters indicate that more complex interactions exist among the congestion stages, possibly due to the presence of zipper lanes causing vehicles abruptly merge.

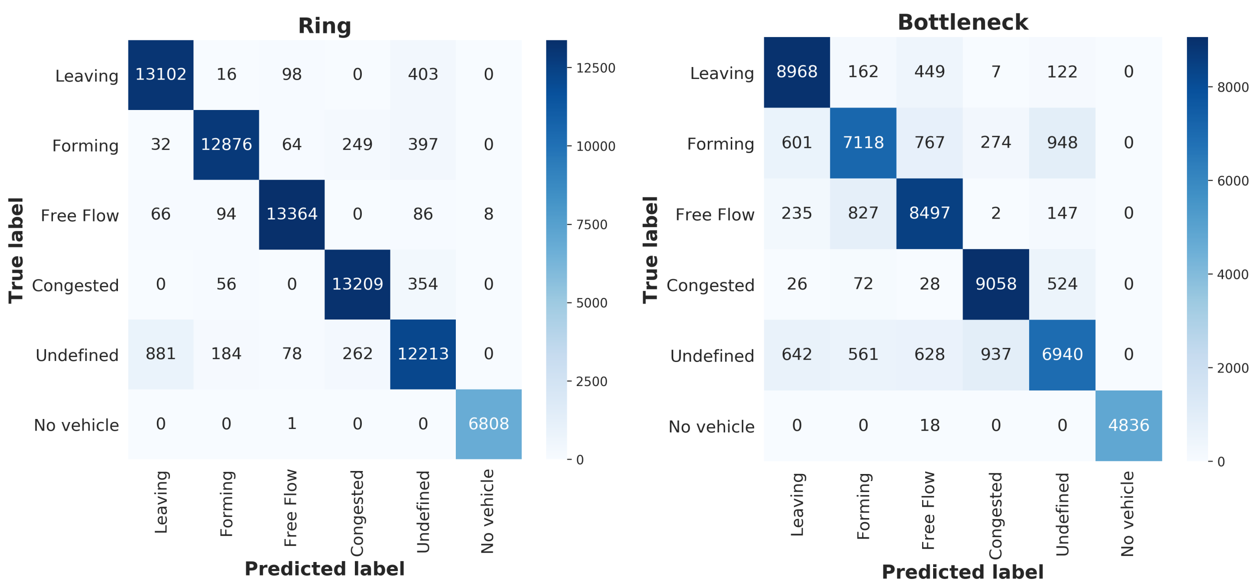

Fig. 5: Confusion matrix of a trained CSC in Ring (LEFT) and Bottleneck (RIGHT) on the validation set.

The CSC training parameters are provided in Table III.