MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

# Large-scale traffic control using autonomous vehicles and decentralized deep reinforcement learning

Maske, H.; Chu, T.; Kalabic, U.

TR2019-126 October 31, 2019

**Abstract**

In this work, we introduce a scalable, decentralized deep reinforcement learning (RL) scheme for optimizing vehicle traffic consisting of both autonomous and human-driven vehicles. The control inputs to the system are the following distance and lane placement of the autonomous vehicles and the human-vehicles are uncontrolled. One point of novelty of the scheme is that it is trained on images of traffic, where pixels are colored based on how much they are occupied by humandriven or autonomous vehicles. Another point of novelty is in how the scheme achieves scalable decentralization; it does so by training multiple RL agents, each responsible for its own region of control, but able to at least partially observe neighboring agents' regions of control. In this way, the scheme is infinitely scalable because an RL agent does not need to communicate with its neighbors, and can be applied in systems in which neighboring controllers are not RL-based. We perform a case study of two simulations on a two-lane oval highway in the Simulation of Urban MObility (SUMO) environment. In the first simulation, a single RL agent is applied so that its region of control coincides with an area of traffic congestion. In the second simulation, we apply multiple RL agents over the entire network. The results of the first simulation show that the single RL agent is able to improve traffic congestion. The results of the second simulation show that the decentralized RL scheme is able to achieve even better results.

*IEEE Intelligent Transportation Systems Conference (ITSC)*

# Large-scale traffic control using autonomous vehicles and decentralized deep reinforcement learning

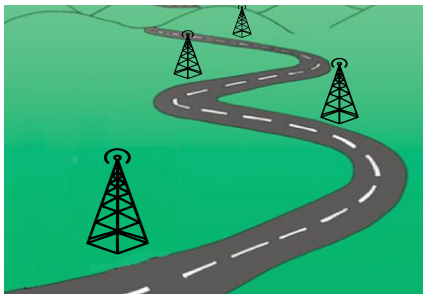
Harshal Maske Tianshu Chu Uroš Kalabić

***Abstract*— In this work, we introduce a scalable, decentralized deep reinforcement learning (RL) scheme for optimizing vehicle traffic consisting of both autonomous and human-driven vehicles. The control inputs to the system are the following distance and lane placement of the autonomous vehicles and the human-vehicles are uncontrolled. One point of novelty of the scheme is that it is trained on images of traffic, where pixels are colored based on how much they are occupied by human-driven or autonomous vehicles. Another point of novelty is in how the scheme achieves scalable decentralization; it does so by training multiple RL agents, each responsible for its own region of control, but able to at least partially observe neighboring agents' regions of control. In this way, the scheme is infinitely scalable because an RL agent does not need to communicate with its neighbors, and can be applied in systems in which neighboring controllers are not RL-based.**

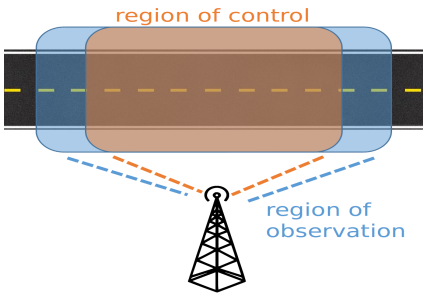
**We perform a case study of two simulations on a two-lane oval highway in the Simulation of Urban MObility (SUMO) environment. In the first simulation, a single RL agent is applied so that its region of control coincides with an area of traffic congestion. In the second simulation, we apply multiple RL agents over the entire network. The results of the first simulation show that the single RL agent is able to improve traffic congestion. The results of the second simulation show that the decentralized RL scheme is able to achieve even better results.**

## I. Introduction

With the introduction of connected vehicles and autonomous vehicles (AVs) on roadways, it will become possible to impact traffic by controlling individual vehicles based on knowledge of the traffic state and, in this work, we propose a control scheme that controls traffic by sending commands to AVs. The scheme we develop is based on model-free reinforcement learning (RL), which is a method used to find approximate solutions to dynamic programming problems with complex system dynamics. The motivation for pursuing a model-free approach is that, due to interactions between human agents and machines, traffic system dynamics are very complex and, although many models exist [1], some of which even consider multiple classes of vehicles [2], none capture all the complexities of traffic behavior. For this reason, our aim being the development of a generalized controller for traffic, we desire that our controller be model-free. We therefore pursue a deep RL (DRL) approach, which uses machine learning to learn an approximate model of system behavior and approximate the control policy. Specifically, we use the actor/critic deep-deterministic policy gradient (DDPG) algorithm [3] to train the neural networks (NNs) used in approximating the dynamics and optimal policy.



(a) Road with regional RL controllers




(b) Controlled and observed regions

Fig. 1: Decentralized RL strategy based on controlling road network segments

Apart from considering complex system behavior, the approach needs to also be appropriate for application to large-scale traffic systems. We propose a novel, scalable, decentralized RL approach. Specifically, we design our scheme so that each RL agent is responsible for the control of a certain segment of traffic, as shown in Fig. 1. To ensure scalability, we allow each agent to observe an area that is larger than what it controls, *i.e.*, the observed region contains the controlled region, as shown in Fig. 1b. Each agent, therefore, is trained on an image of traffic and, specifically, this image is pixelated so that the intensity of the "color" of each pixel represents vehicle density. Training NNs on images is a very successful approach in the field of machine learning and RL in general. A popular application thereof has been the development of RL for the playing of video games [4], [5] and, in a sense, our RL approach could also be seen as playing a game, but with vehicle traffic.

Instead of controlling AVs directly, our RL policy informs individual agents of control commands that are treated as set-points by the vehicles. In this way, we preserve the on-board

This work was supported by Mitsubishi Electric Research Laboratories.

H. Maske and U. Kalabić are with Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA. E-mail: {maske,kalabic}@merl.com

T. Chu is with Uhana Inc., Palo Alto, CA 94306, USA. E-mail: tchu@uhana.io

safety features and lower-level control system properties of each AV. The specific commands that we send are the desired headway distance and desired lane choice for each AV. The choices of command inputs are informed by the way in which actual vehicles drive. Namely, if we assume that AVs of the future will be designed to emulate the behaviors of human-driven, heteronomous vehicles (HVs), then it is reasonable to assume that AVs will modify driving behavior based on changes in headway and lane choice commands. In fact, this is a fair assumption given that the automotive industry measures a vehicle's level of autonomy by the number of driving functions that it relieves from the driver [6]. For this reason, we use the optimal velocity model (OVM) [7], [8], which is a model that relates following distance to vehicle speed, to design the longitudinal controller of our vehicle. For lane-change requests, we use a simple thresholding logic that requests a lane change whenever the RL action corresponding to a lane-change request passes a threshold.

Decentralized RL approaches include [9], [10] and some of them are specific to DDPG, *e.g.*, [11]–[13], with more references reviewed in [14]. However, in most of these applications it is necessary for neighboring RL agents to communicate with each other. Our approach is different in that, because we consider an agent as responsible for its own region, it is not necessary to know its neighbors' control commands when it is possible to instead observe their behavior. As such, it is possible to implement the approach in a traffic system where neighboring territories are controlled by non-RL schemes. Note that other methods that use AVs as actuators in a traffic system are not decentralized, *e.g.*, [15]–[17] for routing and [18]–[21] for control.

To test the performance of our RL approach, we perform a case study of two numerical simulations. The simulations are done in the Simulation of Urban MObility (SUMO) environment and consider vehicles driving on a circular two-lane track, of which $25\%$ are AVs. In both simulations, a vehicle stops in one of the tracks and thereby slows down vehicle traffic. The first simulation considers an RL agent controlling the quarter-segment of the track in which the vehicle stops and its results show that traffic is improved by our algorithm. In the second simulation, an RL agent is applied to each quarter-segment of the track; its results show further improvement in system behavior.

The rest of the paper is structured as follows. Section II describes the decentralized RL scheme. Section III presents the case study. Section IV is the conclusion.

## II. Decentralized RL for Traffic Control

In this section, we present a decentralized DRL approach used for the control of traffic using AVs. As shown in Fig. 1a, the system consists of roads, each of whose segments are controlled by different RL agents, making the approach fully scalable. Instead of implementing an explicit communication protocol between agents, we allow each agent to observe an area larger than it controls. In this way, agents' regions of control do not overlap, but their regions of observation overlap with neighboring regions of control. In the following, we describe the models used in vehicle control, the determination of states and actions, the choice of reward function, and the DRL algorithm.

### A. Longitudinal Vehicle Control Model

We model the longitudinal control of AVs and HVs using the OVM. Acceleration of a vehicle $i$ at time $t$ is given by,

$$u_{i,t} = \alpha_i \left(v^o(h_{i,t}) - v_{i,t}\right) + \beta_i (v_{i-1,t} - v_{i,t}), \qquad (1)$$

where $h_{i,t}$ is its headway or bumper-to-bumper distance between vehicle $i$ and its leading vehicle $i-1$, $v_{i,t}$ is its velocity, $v_{i-1,t}$ is the velocity of its leading vehicle, $\alpha_i$ and $\beta_i$ are its headway and relative velocity gains, respectively. The function $v^o(h)$ is a headway-based velocity policy,

$$v^o(h) = \begin{cases} 0 & \text{if } h \le h^s, \\ \frac{1}{2} v_{\max} \left(1 - \cos\left(\pi \frac{h-h^s}{h^g - h^s}\right)\right) & \text{if } h^s < h < h^g, \\ v_{\max} & \text{if } h \ge h^g, \end{cases}$$

where $h^s$ is the stopping headway, $h^g$ is the full-speed headway, and $v_{\max}$ is the maximum speed.

Given that we assume that AVs emulate HV behavior, this model is the same for both types of vehicles. To control AVs, we send a desired full-speed headway command $h_i^g$ to an AV $i$. For HVs, their corresponding $h^g$ is maintained at a constant value. The stopping headway $h^s$ is fixed in both AVs and HVs. The parameters $\alpha_i$ and $\beta_i$ differ among HVs, and are set to be constant and equal for all AVs. For all vehicles $i$, the acceleration $u_{i,t}$ is saturated to lie in the interval $[u_{min}, u_{max}]$, where $u_{min}$ and $u_{max}$ are the minimum and the maximum values of acceleration, respectively.

### B. Lane Change Control Models

The lane change request logics are different for HVs and AVs. For HVs, we implemented a simple request model, the use of which avoids an excessive number of lane change requests: At each time-step, we record the average speed $v_{\text{avg}}$ of vehicles moving in the available lane and then randomly determine whether to request a lane change with some probability $p$. An HV has a probability of requesting a lane change on two conditions: firstly, the HV must have non-positive acceleration, *i.e.*, it must either be stopped, moving at constant velocity, or decelerating; secondly, the average speed in the adjacent lane $v_{\text{avg}}$ must exceed the HV's current speed $v_i$ by no less than a certain amount $\Delta v_\ell$,

$$v_{\text{avg}} \le v_i + \Delta v_\ell. \qquad (2)$$

For AVs, the lane change request is an action of the RL agent. The lane change request command is denoted by $\ell_i \in [-1, 1]$ for the $i$-th AV. The lane change request may be towards the left or right lane, provided the corresponding lane is available. The logic for an AV lane-change request is,

$$\begin{cases} \text{left} & \text{if } \ell_i \ge 0.5 \text{ and left lane available,} \\ \text{right} & \text{if } \ell_i \le -0.5 \text{ and right lane available,} \end{cases}$$

and nothing otherwise.

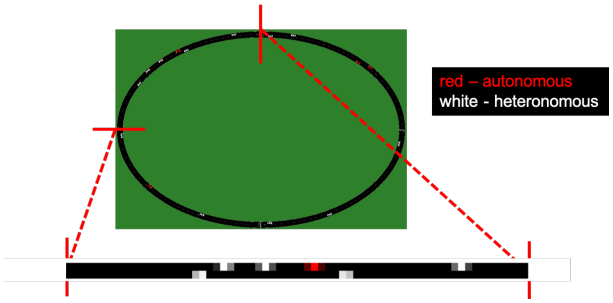



Fig. 2: Example of traffic image state passed to RL

For both HVs and AVs, we used the SUMO model [22] to execute a lane change. Note that, although SUMO provides models for requesting a lane change, in order to override the SUMO model for AVs, we also had to implement a model for HVs, choosing to implement the SUMO model.

### C. States

Given an observed region of the road network at a time-instant $t$, we take a snapshot of the observed part of the road network. An example image is shown in Fig. 2, where we can see that the size of pixels is on the order of the size of a vehicle; intuitively, this makes sense, as a denser grid will not capture more information about system behavior. In general, we choose the grid so that its elements coincide with lane demarcations and their along-track width is between half the length and the full length of a car, *i.e.*, between 2.5m and 5m. The grid elements are colored so that red represents an AV and white represents an HV, so the state of each grid element, or pixel, is given by $(c_r, c_w)$, where $c_r, c_w \in [0, 1]$ are the proportions of the grid occupied by AVs and HVs, respectively. We represent the image by a vector $X_t$, which contains all values $c_r$ and $c_w$ for all pixels in the observed segment.

The state is then set to be the three most recent snapshots of traffic,

$$s_t = \begin{bmatrix} X_{t-2} \\ X_{t-1} \\ X_t \end{bmatrix}. \tag{3}$$

In this way, we capture the position and approximate velocity and acceleration data corresponding to the traffic segment. Conceivably, it is possible to consider snapshots at additional time-instants; we have settled on 3 after some experimentation.

Note that the image-based approach does not rely on directly estimating vehicles states, and can be used with cameras observing traffic flow or with observers providing vehicle state estimates. We believe it is also more robust for RL-based application than, for example, setting the state to be a vector of vehicle states as in [18]; using the image-based approach, it is straightforward to apply RL to an open network since the number of states does not change. As an example of grid-based application, [23] applies DRL to traffic light control and utilizes a non-pixelated $\{0, 1\}$ grid to feed vehicle position, but still resorts to using numerical values of speed for input. In all, learning from images has played a pivotal role in the successes of DRL, and we expect that its adoption to the problem of traffic control will likely improve controller performance.

### D. Actions

The control actions available to each RL agent are governed by the positions of AVs in its region of control. For every AV, the algorithm chooses 2 commands, headway and lane choice, whose details were described previously. We set $N_A$ to be the maximum number of AVs that an agent can control, so that the action vector $a_t$ is an $2N_A$-dimensional vector,

$$a_t = \begin{bmatrix} h^g_1(t) & \ell_1(t) & \cdots & h^g_{N_A}(t) & \ell_{N_A}(t) \end{bmatrix}^{\mathrm{T}}. \tag{4}$$

Each action is applied to the number of AVs that are present in the control region, with the $i$-th couple of commands being applied to the $i$-th AV until there are no AVs left or $i = N_A$. In this way, we only control the first $N_A$ AVs in a region, the ordering of which is pre-set and fixed.

### E. Reward Function

Our goal is to minimize traffic delay in the observed region. We define the delay $d_i$ for some vehicle $i$ to be the ratio of the difference between the free-flow velocity $v_f$ and the vehicle current velocity $v_i$ and the free flow velocity, with the ratio floored at 0,

$$d_i = \max\{0, 1 - v_i/v_f\}. \tag{5}$$

The reward function that we maximize is given by,

$$r_j = -\frac{1}{|\boldsymbol{\mathcal{V}}|} \sum_{i \in \boldsymbol{\mathcal{V}}} C^{d_i} - 1, \tag{6}$$

where $\boldsymbol{\mathcal{V}} = \cup_i \mathcal{V}_i$ and $\mathcal{V}_i$ is the set of vehicle indexes containing $i$, the index of the $i$-th AV, the indexes of all vehicles $j$ whose distance to the AV is less than some maximum distance $x_{\max}$, and $C > 0$ is some weighting constant.

The choice of reward function therefore only considers the average penalty $C^{d_i} - 1$ for vehicles which can be impacted by an AV. Therefore $x_{\max}$ is chosen to be no greater than the maximum headway $h_{\max}$. The penalty itself is related to the delay so that it is increasing when $d_i = 0$ and monotonically increasing for all possible values of $d_i \in [0, 1]$; this way, we penalize higher increases in individual delays more than an equivalent decrease in delay for the whole system, penalizing the situation where an AV overly deteriorates its own delay in order to improve the performance of the system.

Note that, during exploration an agent might cause the headway between two vehicles to go below stopping distance $h^s$, indicating a failure or, worse, a collision. To sufficiently penalize this sort of behavior, when this occurs we set $r_t = -G$, where $G$ is a very large number, and terminate the training episode.

### F. Algorithm Implementation Details

To learn the policy for each decentralized agent, we use DDPG, a policy gradient algorithm that has been widely

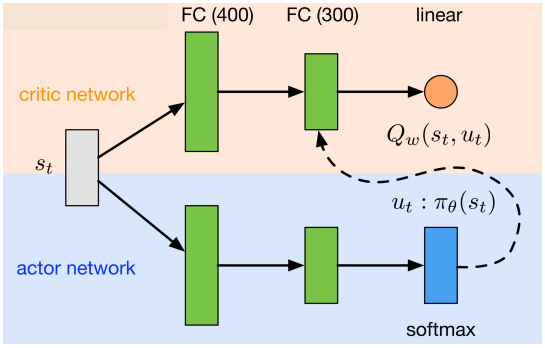



Fig. 3: DDPG actor and critic DNN structures, with different layer types in different colors

applied for continuous control or robotics and whose effectiveness has been verified in various applications. The gradient update procedure we use can be found in [3] and we avoid repeating here due to space considerations.

*1) Inputs and outputs:* The input to the critic and actor networks, shown in Fig. 3 is $s_t$ given in (3). The output of the actor network $\pi_\theta(s_t)$ is a $2N_A$-dimensional with each element lying in the interval $[-1, 1]$. To obtain an action according to equation (4), elements corresponding to the headway are scaled to the interval $[40\text{m}, 80\text{m}]$.

*2) NN settings:* We adopt the deep NN (DNN) structure proposed in [3] and illustrated in Fig. 3. The DNN contains two hidden layers of rectified non-linearity with 400 and 300 units respectively. The final layer of the critic network is linear with a scalar output of the Q-value, and the final layer of the actor network is `tanh` with a $2N_A$ dimensional vector output of combined headway and lane change recommendations. The actions are not included until the second hidden layer of the critic network. Default hyper-parameters are used for training DNN weights: the learning rates are $10^{-4}$ and $10^{-3}$ for actor and critic networks, respectively, and the critic network has a $10^{-2}$-weighted $L^2$-norm weight regularization. The global gradient norm is clipped at 40 for stabilizing the update.

*3) Training settings:* Default hyper-parameters are used for DDPG training The reward parameter $C$ is set to 2. The reward penalty corresponding to failure or collision is $G = 1000$. The DDPG model is trained over 3 million steps, and each episode simulates the vehicle dynamics for 60 seconds, *i.e.,* $T = 600$ steps. Thus, the training takes place for 500 episodes. We use the following constraint parameters adopted from [24], [25]: $u_{\min} = -6\text{m/s}^2$, $u_{\max} = 4\text{m/s}^2$, $v_{\max} = 30\text{m/s}$, $h_s = 5\text{m}$, $h_g = 60\text{m}$.

## III. Case Study in SUMO

We performed two numerical simulations in SUMO: A simulation of a single RL agent controlling a segment of the road and a simulation of multiple, decentralized RL agents whose regions of control cover the entire traffic network. The control logic has been implemented in Python and the interface between Python and SUMO is done by the TRAffic Control Interface (TRACI) [26].

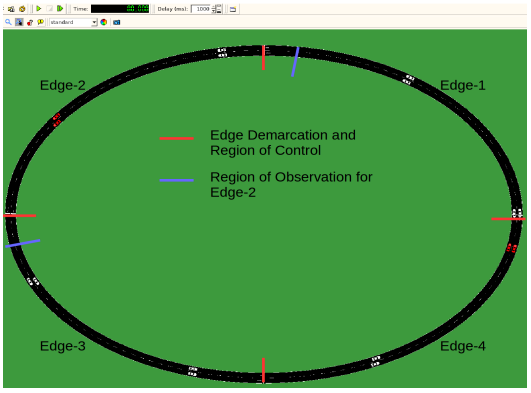



Fig. 4: Screenshot of traffic network in SUMO

### A. Simulation set-up

In this section, we describe the simulation set-up and simulations that we performed. We describe the traffic network, the simulation scenario, the two simulations we performed and results.

*1) Traffic Network:* The traffic network, whose screenshot is presented in Fig. 4, consists of a two-lane oval-shaped highway with total length of 793m. The network is split into four quadrants corresponding to regions of control. Observed regions are 20% larger than and are centered around the controlled regions. The traffic is composed of 4 AVs and 12 HVs, equivalent to a 25% penetration of AVs. Each training episode lasts 60s and both the control and simulation are updated using a time step of 0.1s. In order to be able to control all AVs, the parameter $N_A$ is set to 4.

At the beginning of each episode, vehicles are placed in groups of two and are equidistant from each other, as shown in Fig. 4; to increase variability in the initial condition, a Gaussian random signal is generated and used to shift the vehicles' position slightly.

*2) Scenario:* In each episode, the same HV performs a braking maneuver in the right lane at $t = 5$s, decelerating by $1\text{m/s}^2$ until stopping, and remaining stopped until $t = 55.5$s. The HV's maneuver causes vehicles in its lane to brake, being unable to change lane due to the fast-moving traffic. Without additional control, this causes congestion behind the stopped HV. This extreme example of slow-down in a single lane will be used to test the performance of our scheme in alleviating congestion.

*3) Simulations:* In the first simulation, we implement a single, regional RL agent, choosing the region to be the one in which the slowdown occurs. For the section simulation, we implement multiple, regional RL agents with control regions that cover the entire network, but do not overlap, and with observation regions that overlap with neighboring agents' regions of control.

### B. Results

*1) Single regional RL agent:* The results corresponding to the simulation with a single RL agent are presented in Figs. 5-8. The results show that a single RL agent is able to learn a policy that improves traffic congestion.

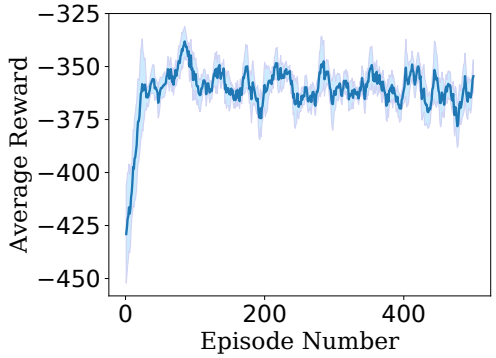



Fig. 5: Average reward during training for the single agent

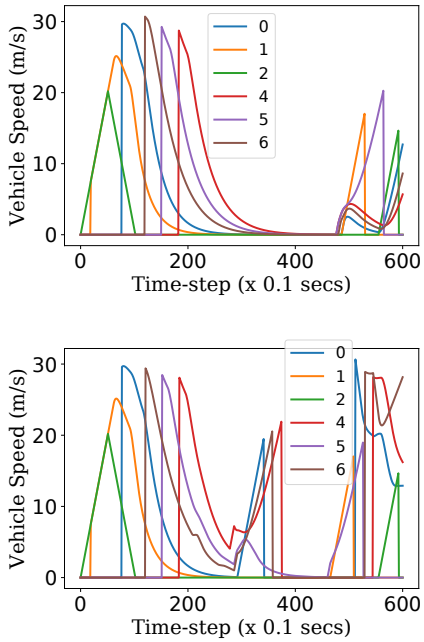



Fig. 6: HVs with (bottom) and without (top) control corresponding to single-agent simulation

Fig. 5 presents the average and the standard deviation of reward during learning over 500 episodes. The convergence to steady state is uniform, indicating sample efficiency of our decentralized approach.

In Fig. 6, we show the speed profiles of the HVs that have been most affected by congestion. HV 2 is the HV that is commanded to brake to a full stop. In the results, we show that HVs 0, 4 and 6 are relieved from congestion at about $t = 32$s, and HVs 1 and 5 at about $t = 45$s. The speed profiles of AVs is shown in Fig. 7. From the results, we can see that the speeds of AVs 3 and 11 are most impactful on traffic congestion.

In Fig. 8, we show the delay and speed averaged over all vehicles in the controlled region. The results show that the RL agent achieved better delay and speed performance over the course of the entire simulation. Better performance is achieved by the full decentralized DRL scheme, whose simulation results are discussed in the following.

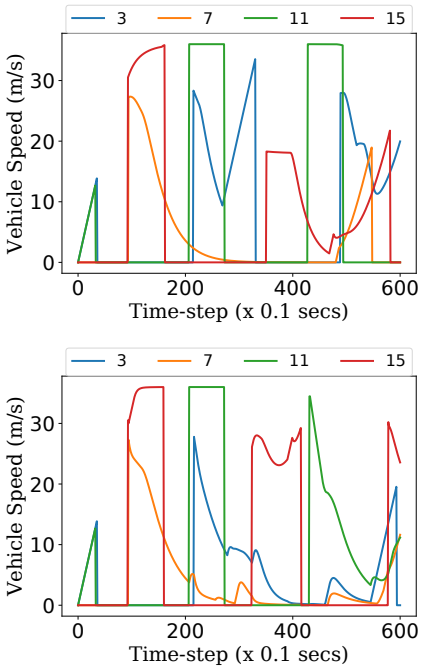



Fig. 7: AVs with (bottom) and without (top) control corresponding to single-agent simulation

*2) Decentralized RL:* The results corresponding to the simulation with multiple, decentralized RL agents are presented in Figs. 8 and 9. In Fig. 8, we see that we achieve further improvement in congestion alleviation over the single RL case. This is because the RL agent controlling the congestion region is now aided by the RL agent controlling the region preceding the region with congestion. In Fig. 9, we show the behavior of AVs in the preceding region. We observe that, in this region, the scheme increases the speed of AVs 3 and 15, while slowing down AV 11 when compared to the baseline. We therefore conclude that the RL agent controlling the inflow is able to improve congestion without knowledge of downstream traffic. Note that the RL agents controlling the other two regions do not contribute to congestion alleviation as their reward feedback stays constant due to the fact that they are too far removed from the occurrence of congestion.

## IV. Conclusion

In this paper, we presented a scalable, decentralized RL scheme for controlling vehicle traffic using AVs. One point of novelty is the use of an image of traffic for input to the DRL algorithm used in determining the policy. Another point of novelty is the achievement of scalable decentralization by applying RL agents to regions of control, but allowing them to partially observe neighboring agents' control regions.

We presented a case study consisting of two simulations. The first simulation implemented a single RL agent responsible for a region of traffic congestion; the second simulation implemented RL agents over the entire network. From the results, it is clear that a single agent is able to improve traffic and that multiple agents are able to perform even better.

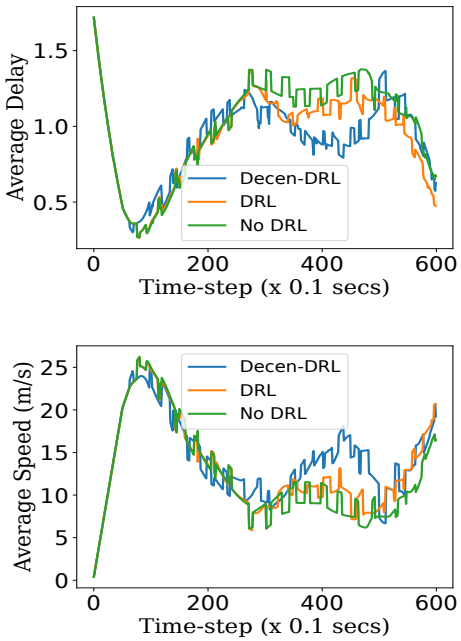



Fig. 8: Average delay (top) and speed (bottom) in a single episode corresponding to single-agent simulation

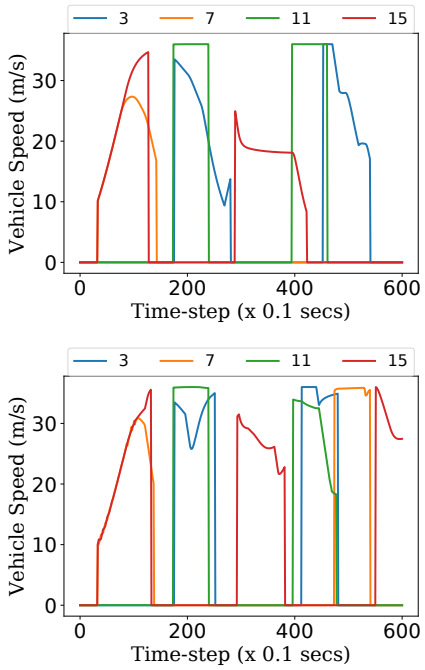



Fig. 9: AVs with (bottom) and without (top) control corresponding to full-network simulation

## REFERENCES


[1] M. Garavello, K. Han, and B. Piccoli, *Models for Vehicular Traffic on Networks*. Springfield, MO: AMS, 2016.

[2] S. Benzoni-Gavage and R. M. Colombo, "An $n$-populations model of traffic flow," *Eur. J. Appl. Math.*, vol. 14, no. 5, pp. 587–612, 2003.

[3] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," arXiv:1509.02971, 2015.

[4] V. Mnih *et al.*, "Human-level control through deep reinforcement learning."

[5] O. Vinyals *et al.*, "Starcraft II: A new challenge for reinforcement learning."

[6] SAE On-Road Automated Vehicle Standards Committee, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," Society of Automotive Engineers, Standard J3016_201806, Jun 2018.

[7] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama, "Dynamical model of traffic congestion and numerical simulation," *Phys. Rev. E*, vol. 51, no. 2, pp. 1035–1042, 1995.

[8] M. Batista and E. Twrdy, "Optimal velocity functions for car-following models," *J. Zhejiang Univ.-Sc. A*, vol. 11, no. 7, pp. 520–529, 2018.

[9] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep decentralized multi-task multi-agent reinforcement learning under partial observability," in *Proc. Int. Conf. Machine Learning*, Sydney, 2017, pp. 2681–2690.

[10] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. Int. Conf. Robotics and Automation*, Singapore, 2017, pp. 285–292.

[11] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Proc. Int. Conf. Autonomous Agents and Multiagent Systems*, Sao Paulo, 2017, pp. 66–83.

[12] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Advances Neural Info. Process. Syst.*, Long Beach, CA, 2017, pp. 6379–6390.

[13] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully decentralized multi-agent reinforcement learning with networked agents," arXiv:1802.08757, 2018.

[14] Y. Li, "Deep reinforcement learning: An overview," arXiv:1701.07274, 2017.

[15] D. A. Lazar, S. Coogan, and R. Pedarsani, "Capacity modeling and routing for traffic networks with mixed autonomy," in *Proc. Conf. Decision and Control*, Melbourne, Australia, 2017, pp. 5678–5683.

[16] N. Mehr and R. Horowitz, "Can the presence of autonomous vehicles worsen the equilibrium state of traffic networks?" in *Proc. Conf. Decision and Control*, Miami Beach, FL, 2018, pp. 1788–1793.

[17] J. Q. James and A. Y. S. Lam, "Autonomous vehicle logistic system: Joint routing and charging strategy," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 7, pp. 2175–2187, 2018.

[18] C. Wu, A. Kreidieh, E. Vinitsky, and A. M. Bayen, "Emergent behaviors in mixed-autonomy traffic," in *Proc. Conf. Robot Learning*, Mountain View, CA, 2017, pp. 398–407.

[19] N. Mehr, R. Li, and R. Horowitz, "A game theoretic macroscopic model of bypassing at traffic diverges with applications to mixed autonomy networks," arXiv:1809.02762, 2018.

[20] E. Vinitsky, A. Kreidieh, L. Le Flem, N. Kheterpal, K. Jang, F. Wu, R. Liaw, E. Liang, and A. M. Bayen, "Benchmarks for reinforcement learning in mixed-autonomy traffic," in *Proc. Conf. Robot Learning*, Zurich, 2018, pp. 399–409.

[21] R. E. Stern *et al.*, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments."

[22] J. Erdmann, "Lane-changing model in SUMO," in *Proc. SUMO Conf.*, Berlin, 2015, pp. 105–123.

[23] X. Liang, X. Du, G. Wang, and Z. Han, "Deep reinforcement learning for traffic light control in vehicular networks," *arXiv preprint arXiv:1803.11115*, 2018.

[24] W. Gao, Z.-P. Jiang, and K. Ozbay, "Data-driven adaptive optimal control of connected vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1122–1133, 2017.

[25] C. Massera Filho, M. H. Terra, and D. F. Wolf, "Safe optimization of highway traffic with robust model predictive control-based cooperative adaptive cruise control," *IEEE Trans. Intell. Trasnp. Syst.*, vol. 18, no. 11, pp. 3193–3203, 2017.

[26] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen, "Flow: Architecture and benchmarking for reinforcement learning in traffic control," arXiv:1710.05465, 2017.