Lagrangian Control through Deep-RL: Applications to Bottleneck Decongestion

Eugene Vinitsky* , Kanaad Parvate†, Aboudy Kreidieh‡, Cathy Wu†, Alexandre Bayen†‡§
*UC Berkeley, Department of Mechanical Engineering
†UC Berkeley, Electrical Engineering and Computer Science
‡UC Berkeley, Department of Civil and Environmental Engineering
§UC Berkeley, Institute for Transportation Studies

***Abstract*— Using deep reinforcement learning, we derive novel control policies for autonomous vehicles to improve the throughput of a bottleneck modeled after the San Francisco-Oakland Bay Bridge. Using *Flow*, a new library for applying deep reinforcement learning to traffic micro-simulators, we consider the problem of improving the throughput of a traffic benchmark: a two-stage bottleneck where four lanes reduce to two and then reduce to one. We first characterize the inflow-outflow curve of this bottleneck without any control. We introduce an inflow of autonomous vehicles with the intent of improving the congestion through Lagrangian control. To handle the varying number of autonomous vehicles in the system we derive a per-lane variable speed limits parametrization of the controller. We demonstrate that a 10% penetration rate of controlled autonomous vehicles can improve the throughput of the bottleneck by 200 vehicles per hour: a 25% improvement at high inflows. Finally, we compare the performance of our control policies to feedback ramp metering and show that the AV controller provides comparable performance to ramp metering without the need to build new ramp metering infrastructure. Illustrative videos of the results can be found at https://sites.google.com/view/itsc-lagrangian-avs/home and code and tutorials can be found at https://github.com/flow-project/flow.**

I. Introduction

Over the past few years, *deep reinforcement learning* (RL) has emerged as a novel control technique for highly non-linear, stochastic, data-rich problems. RL has been applied to problems as diverse as control of 3D humanoid locomotion [1], [2], control of Atari games directly from pixels [3], and control of multi-legged mini-robots [4]. These successes have prompted the application of RL techniques to intelligent transportation. RL has been applied to Variable Speed Limit control [5], [6], control of traffic lights [7], [8], and ramp metering [9], [10], among many others. In this article, we consider the prospect of using RL to learn longitudinal controllers for autonomous vehicles in a congested setting with only partial autonomy.

Advances in automation of transportation networks offer an unparalleled opportunity to implement novel traffic control paradigms. The past few years have seen a steady stream of highlights ranging from California approving autonomous cars without a driver [11] to Waymo ordering 20,000 vehicles for conversion to automation [12]. In areas such as Phoenix, Arizona and the California Bay Area, it is likely that the next 5-10 years will see the emergence of autonomy-on-demand services in which users can call for an automated vehicle to transport them to their next location.

There exists a vast amount of connected, autonomous vehicle (CAV) literature that attempts to quantify the potential impact of automation on traffic congestion. The central idea unifying CAV work is that automated vehicles can alleviate or replace human driving inefficiency, whether through incorporation of upstream/downstream traffic information or improved reaction time. In this work, modified vehicle acceleration profiles are used for tasks as varied as dissipating traffic shock-waves [13], improving highway capacity via decreased following gap [14], and inducing cooperative on-ramp merges [15]. Additionally, work has been done on CAV driving strategies for decongestion of a bottleneck [16].

In prior work, we focused on characterizing the traffic-smoothing capabilities of AVs in a variety of small, representative scenarios [17]. To do so, we developed *Flow* [18], a library for applying reinforcement learning to autonomous vehicles in traffic micro-simulators. In earlier research, inspired by work with AVs in [13], we demonstrated the ability of RL to learn a controller for a single autonomous vehicle that could smooth the spontaneous stop-and-go waves that emerge in a ring of vehicles [19]. This “toy” example demonstrated the potential for RL to learn controllers for AVs but left the problem of control of more complex scenarios to future work.

In the present work we use *Flow* to introduce a novel traffic benchmark: demonstrating the potential impact of CAVs on de-congestion of a traffic bottleneck. Inspired by the bottleneck dynamics of the San Francisco-Oakland Bay Bridge, we focus on a situation in which a multi-lane highway has its number of lanes cut in half by a zipper merge, and then cut in half again by another merge. In simulation, we demonstrate that this bottleneck exhibits the phenomenon of *capacity drop* [20] [21] in which the outflow of the bottleneck increases with inflow but suddenly drops down once the inflow exceeds a critical value.

One of the major successes of intelligent transportation infrastructure is the implementation of feedback control for ramp metering [22], in which the inflow to a bottleneck is reduced to keep the inflow below its critical value. This article attempts to extend the concept of metering (traditionally

Corresponding author: Eugene Vinitsky (evinitsky@berkeley.edu)
Email addresses:{aboudy, kanaad, evinitsky, bayen}@berkeley.edu, cathywu@mit.edu

operated by fixed [Eulerian] traffic light infrastructure) to AVs. For this, it relies on Lagrangian control of the flow in which AVs are used as mobile actuators to achieve effects similar to metering (i.e. flow control). Unlike ramp metering, where control can be applied to each vehicle but cannot be applied past the meter, the control afforded by AVs can be applied at any point but only affects the platoons that form behind them in their lanes. However, the AVs can completely control the flow speed of their platoons and can accelerate and decelerate to control the relative spacing of vehicles in their platoon. Control of platoon spacing makes it possible for two AVs in adjacent platoons to coordinate to encourage easier merging between their platoons. We refer to this as Lagrangian control, as is commonly done in fluid mechanics, in reference to the trajectory-based actuation as opposed to the Eulerian control volume-based actuation.

The dynamics of a bottleneck are both non-linear and difficult to model with microscopic models due to the complexity of lateral and longitudinal dynamics in multi-lane settings. To sidestep this issue, we use model-free reinforcement learning, in which we train the AVs with the goal of maximizing the outflow of the bottleneck. We show that despite the stochasticity in the platoon lengths and distributions of AVs, AVs can learn to effectively act like a ramp meter. AVs can react to the formation or warning-signs of congestion and regulate the traffic inflow to enable congestion dissipation. We demonstrate that a single centralized controller acting on the speed limits of the autonomous vehicles in the bottleneck can effectively stabilize the outflow of the bottleneck at 1000 vehicles per hour: 200 vehicles per hour above the uncontrolled equilibrium.

The main contributions of this work are:

1) The development of a deep-RL model-free framework for Lagrangian control of freeways by AVs
2) The learning of Lagrangian control policies for bottleneck congestion.
3) The demonstration of an improvement of 25% in bottleneck outflow at inflows past the critical inflow.
4) A code release of a novel traffic control benchmark at `https://github.com/flow-project/flow`.

The rest of the article is organized as follows. Section II provides an introduction to deep RL policy gradient methods, car following models, and *Flow*, the deep reinforcement learning to micro-simulator library that we use for our experiments. Section III formulates the capacity drop diagrams of our bottleneck as well as the results from the autonomous vehicle control. Section IV provides a discussion of the results. Finally Section. V summarizes our work and provides a discussion of possible future research directions.

II. Background

A. Reinforcement Learning

In this section, we discuss the notation and describe in brief the key ideas used in reinforcement learning. Reinforcement learning focuses on maximization of the discounted reward of a finite-horizon *Markov decision process* (MDP) [23]. The system described in this article solves tasks which conform to the standard structure of a finite-horizon discounted MDP, defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \rho_0, \gamma, T)$, where $\mathcal{S}$ is a (possibly infinite) set of states, $\mathcal{A}$ is a set of actions, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the transition probability distribution for moving from one state s to another state s' given action a, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\rho_0 : \mathcal{S} \to \mathbb{R}_{\geq 0}$ is the initial state distribution, $\gamma \in (0, 1]$ is the discount factor, and T is the horizon. For partially observable tasks, which conform to the structure of a *partially observable Markov decision process* (POMDP), two more components are required, namely Ω, a set of observations of the hidden states, and $\mathcal{O} : \mathcal{S} \times \Omega \to \mathbb{R}_{\geq 0}$, the observation probability distribution.

RL studies the problem of how an agent can learn to take actions in its environment to maximize its cumulative discounted reward: specifically it tries to optimize $R = \sum_{t=0}^{T} \gamma^t r_t$ where r_t is the reward at time t. The goal is to use the observed data from the MDP to optimize a policy $\Pi : \mathcal{S} \to \mathcal{A}$, mapping states to actions, that maximizes R. It is increasingly popular to parametrize the policy via a neural net. We will denote the parameters of this policy, also known as neural network weights, by θ and the policy by π_θ. A neural net consists of a stacked set of affine linear transforms and non-linearities that the input is alternately passed through. The presence of multiple stacked layers is the origin of the term "deep" reinforcement learning.

In this work we exclusively use a Gated Recurrent Unit Neural Net (GRU) [24], a neural net with a hidden state that gives the policy memory. Readout and editing of the hidden states is done by a series of "gates" whose parameters are evolved as the learning progresses. As will be discussed in Sec. III-B, the usage of memory is important for the partially observed tasks we tackle in this work. In our partially observed Markov decision process (POMDP), hidden states like the positions and velocities of the automated vehicles can only be fully observed on occasion and thus must be stored.

B. Policy Gradient Methods

Policy gradient methods take the set of state-action-reward pairs generated from the experiments and use them to estimate $\nabla_\theta R$, the gradient of the reward with respect to the parameters of the policy which can be used to update the policy. To optimize the parameters of the neural net we use *Trust Region Policy Optimization* (TRPO) [25], a policy gradient method. TRPO constrains the KL divergence, a measure of the distance of two probability distributions, between the original policy and the policy update to be within a fixed bound. This prevents the noisy gradient update from drastically shifting the policy in a bad direction.

C. Car Following Models

For our model of the driving dynamics, we used the *Intelligent Driver Model* [26] (IDM) that is built into SUMO [27]. IDM is a microscopic car-following model commonly used to model realistic driver behavior. Using this model, the acceleration for vehicle α is determined by its bumper-to-bumper headway s_α (distance to preceding vehicle), ego velocity v_α, and relative velocity Δv_α, via the following equation:

$$a_{\text{IDM}} = \frac{dv_\alpha}{dt} = a\left[1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right] \quad (1)$$

where s^* is the desired headway of the vehicle, denoted by:

$$s^*(v_\alpha, \Delta v_\alpha) = s_0 + \max\left(0, v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}}\right) \quad (2)$$

where $s_0, v_0, T, \delta, a, b$ are given parameters. Typical values for these parameters can be found in [26]. To better model the natural variability in driving behavior, we induce stochasticity in the desired driving speed v_0. On any edge, the value of v_0 for a given vehicle is sampled from a Gaussian whose mean is the speed limit of the lane and whose standard deviation is 20% of the speed limit.

These car following models are not inherently collision-free, we supplement them with a safe following rule: in the event that a vehicle is about to crash it immediately comes to a full stop. This is unrealistic, but empirically this behavior occurs rarely.

D. *Flow*

We run our experiments in *Flow* [18], a library we built that provides an interface between a traffic microsimulator, SUMO [27], and popular reinforcement learning libraries, rllab [28] and RLlib [29], reinforcement learning and distributed reinforcement learning libraries respectively. *Flow* enables users to create new traffic networks via a python interface, introduce autonomous controllers into the networks, and then train the controllers on high-CPU machines on the cloud via AWS EC2. To make it easier to reproduce our experiments or try to improve on our benchmarks, the code for *Flow*, scripts for running our experiments, and tutorials can be found at `https://github.com/cathywu/flow`

Fig. 1 describes the process of training the policy in *Flow*. The controller, here represented by policy Π, has output sampled from multi-dimensional Gaussian $\mathcal{N}(\mu, \sigma I)$ where μ and σI are vectors of means and standard deviations and σI is the covariance. These are taken in by the traffic micro-simulator, which outputs the next state and a reward. After accumulating enough samples, the states, actions, and rewards are passed to the training procedure, which combines them with the baselines to produce advantages i.e. estimates of which actions performed well. These are passed to the optimizer to compute a new policy.

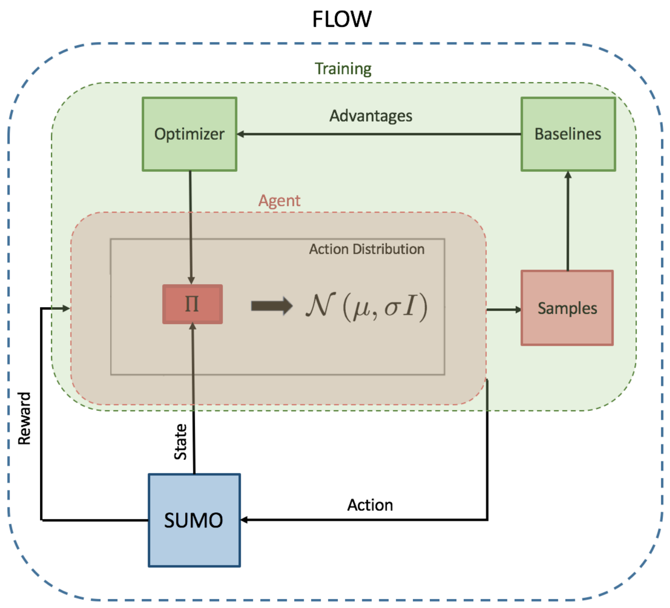

Fig. 1: Diagram of the iterative process in *Flow*. Portions in red correspond to the controller and rollout process, green to the training process, and blue to traffic simulation.

III. Experiments

A. *Experiment setup*

We attempt to decongest the bottleneck depicted in Fig. 3 in which a long straight segment is followed by two zipper merges taking four lanes to two, and then another zipper merge sending two lanes to one. This is a simplified model of the post-ramp meter bottleneck on the Oakland-San Francisco Bay Bridge. At inflows above 1500 vehicles per hour, congestion becomes the equilibrium state of the bottleneck model. Once congestion forms, as in Fig. 4, the congestion is unable to dissipate and begins to extend upstream. Of the indicated segments, segments 2, 3, 4 are controllable; these segments can be arbitrarily divided into further pieces on which control can be applied.

An important point to note is that for the purposes of this experiment, lane changing is disabled for all the vehicles in this system. This is partially justified by the lane changing structure seen in Fig. 2, where only lane changing between pairs of lanes is allowed. As discussed in Sec. V, the addition of lane-changing makes the problem more difficult and is postponed for later work.

B. *Reinforcement Learning Structure*

1) Action space: We parametrize the controller as a neural net mapping the observations to a mean and diagonal covariance matrix of a Gaussian. The actions are sampled from the Gaussian; this is a standard controller parametrization [30]. We pick a parametrization of the control action that is invariant to the number of AVs in the system; namely, the speed limits of the autonomous vehicles. Segments two and three are divided into two equally sized pieces, segment four is divided into three. Segments one and five are uncontrolled. For each lane in each piece, at every time-step the controller is allowed to shift the maximum speed of the autonomous vehicles in the segment. The dynamics model of the autonomous vehicles are otherwise given by the Intelligent Driver Model described in sec. II-C i.e.

$$v_j^{\text{AV}}(t + \Delta t) = \min\left(v_{AV}(t) + a_{IDM}\Delta t, v_j^{\max}(t)\right) \quad (3)$$

where $v_j^{AV}(t)$ is the velocity of autonomous vehicle j at time t, a_{IDM} is the acceleration given by an IDM controller, Δt

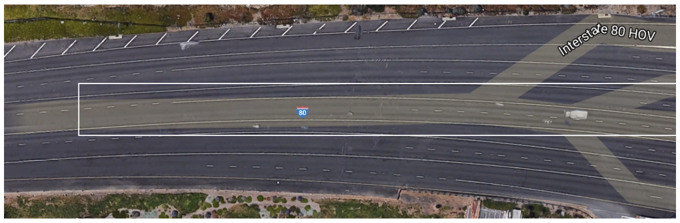

Fig. 2: Bay bridge merge. Relevant portion selected in white. Traffic travels from right to left.

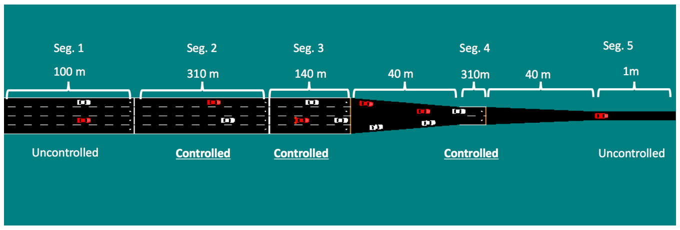

Fig. 3: Long entering segment followed by two zipper merges, a long segment, and then another zipper merge. Red cars are automated, human drivers are in white. Controlled segments and segment names are indicated. Scale is severely distorted to make visible relevant merge sections.

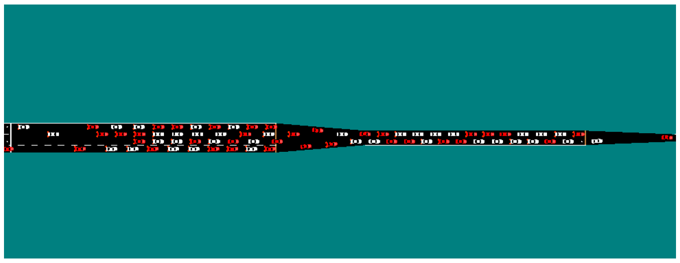

Fig. 4: Congestion forming in the bottleneck.

is the time-step, and $v_j^{\max}(t)$ is the maximum speed set by the RL agent for autonomous vehicle j. At each step for each segment the maximum speed of autonomous vehicle j is updated via

$$v_j^{\max}(t+1) = v_j^{\max}(t) + a_{\text{agent}} \tag{4}$$

where $a_{\text{agent}} \in [-1.5, 1.0]$. This range is picked to be the minimum and maximum acceleration values for a single time-step, so that unphysical accelerations are not commanded. Decelerations of 1.5 $\frac{\text{m}}{\text{s}^2}$ and accelerations of 1.0 $\frac{\text{m}}{\text{s}^2}$ are within range of most vehicles. We use these relatively low accelerations to make the scheme implementable in real traffic. Additionally, we note that when the autonomous vehicles enter the merge areas, their maximum speed is set back to the system's overall max speed of 23 meters per second.

2) Observation space: For the purposes of keeping in mind physical sensing constraints, the state space of the controller is:

- The density and average speed of human drivers in each lane for each observed piece

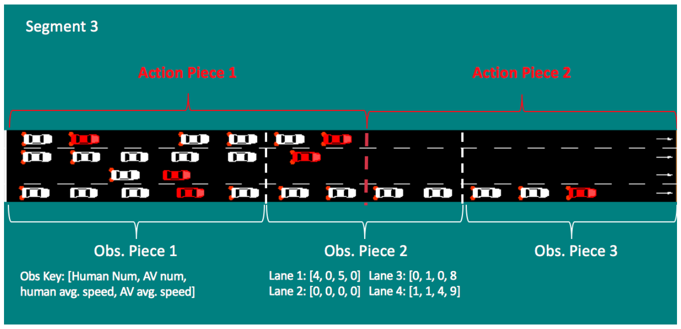

Fig. 5: Illustration of the observation and action division of segment 3 into three observation pieces and two action pieces. The provided key indicates the structure of the observation space for each lane.

- The density and average speed of AVs in each lane for each observed piece
- The outflow at the final bottleneck

The pieces are:

- One piece for each of segments 1 and five
- Three equally spaced pieces for each of segments 2, 3 and 4

This state space could conceivably be implemented on existing roadways equipped with loop detectors, sensing tubes, and vehicle-to-vehicle communication between the AVs. An illustration of this sensing structure is given in Fig. 5, with an extended description of what the state space values might look like for piece 2.

This parametrization of the state space enables us to have a fixed size state space even as the number of AVs vary. Furthermore, as long as the number of AVs is low enough that there are only one or two AVs per lane-segment, it is possible to track the positions of all the AVs by observing the changes in density and velocity as the AVs pass from segment to segment. Thus, at low penetration rates, our parametrization of the observation space does not entail a loss of observability.

C. Reward function

For our reward function we simply used the outflow over the past 20 seconds:

$$r_t = 3600 \sum_{i=t-T}^{t} \frac{n_{\text{exit}}}{T} \tag{5}$$

where n_{exit} is the number of vehicles that exited the system at time-step i. The factor of 3600 converts from vehicles per-second to vehicles per hour.

D. Capacity diagrams

Fig. 6 presents the inflow-outflow relationship of the uncontrolled bottleneck model. To compute this, we swept over inflows from 400 to 2500 in steps of 100, ran 10 runs for each inflow value, and stored the average outflow over the last 500 seconds. Fig. 6 presents the average value, 1 std-deviation from the average, and the min and max value

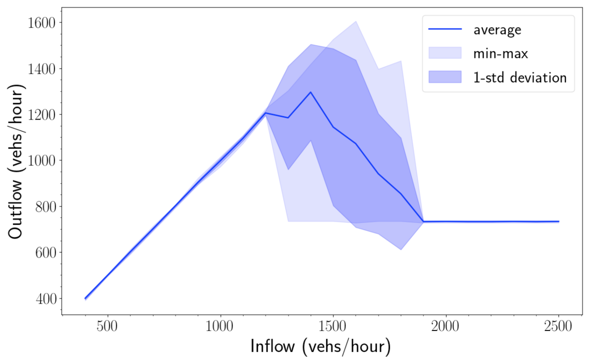

Fig. 6: Inflow vs. outflow for the uncontrolled bottleneck. The solid line represents the average over 10 runs at each inflow value, the darker transparent section is one standard deviation from the mean, and the lighter transparent section is bounded by the min and max over the two runs.

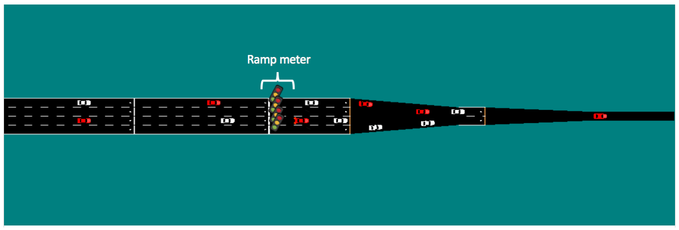

Fig. 7: Position of the ramp meter on the bottleneck is represented by the traffic lights.

of each inflow. Below an inflow of 1300 congestion does not occur, and above 1900 congestion will form with high certainty. A key point is that once the congestion forms at these high inflows, at values upwards of 1400, it does not dissolve. The maximum and minimum outflows for each inflow are indicated in Fig. 6. Since the congestion does not dissipate, the maximum outflow observed represents the highest possible achievable outflow, while the minimum outflow represents the eventual stable state of the system, i.e. *at inflows above 1400 the outflow will eventually drop to the equilibrium value of approximately 800 vehicles per hour.*

E. Feedback ramp metering

As a baseline to compare the efficacy of our model-free Lagrangian control, we use a ramp meter whose cycle time, the ratio of red light to green light time, is output by a feedback controller [31]. The position of the ramp meter is depicted in Fig. 7; it sits 140 meters before the bottleneck.

The desired outflow is determined by the feedback controller

$$q(k+1) = q(k) + K_f(n_{\text{crit}} - \hat{n}) \tag{6}$$

where $q(k)$ is the inflow in vehicles per hour, K_F is a feedback coefficient, n_{crit} is the critical number of vehicles in segment 4 above which congestion is likely to occur, and $\hat{n}$ is the current number of vehicles in segment 4. The cycle time, c consists of a fixed 6 second green phase and a variable length red phase. During the green phase, on average 2 vehicles in each lane are allowed to pass per cycle. Thus, we can convert between cycle-time c and inflow q via

$$c = \frac{2M}{q}3600 \tag{7}$$

where M is the number of lanes in the system. This conversion is used to compute the cycle time from the feedback controllers q value. We update the cycle every T seconds; this value was determined empirically. We used $T = 30,$, $K_F = 20$, $n_{\text{crit}} = 8$. These values were tuned empirically and we stress that there may be better values.

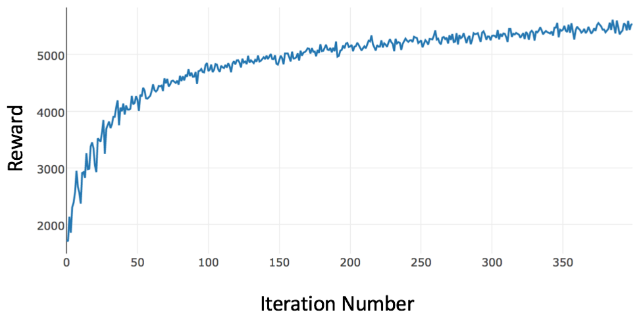

Fig. 8: Convergence of the reinforcement learning reward curve over 400 iterations.

F. Experiment details

We ran the reinforcement learning experiments with a discount factor of .995, a trust-region size of .01, a batch size of 80000, a horizon of 1000, and trained over 400 iterations. The controller is a GRU with hidden size of 64 and a tanh non-linearity. The baseline used to minimize the variance of the gradient is a polynomial baseline that is fitted after each iteration. 10% of the vehicles are autonomous. At the beginning of each training rollout we randomly sample an inflow value between 1000 and 2000 vehs/hour and keep it fixed over the course of the rollout. At each time-step, a random number of vehicles are emitted from the start edge. Thus, the number of vehicles in each platoon behind the AVs will be of variable length and it is possible that at any time-step any given lane may have zero autonomous vehicles in it. To populate the simulation fully with vehicles, we allow the experiment to run uncontrolled for 40 seconds before each run. Finally, taking note that the standard benchmark for ATARI games repeats each action four times [32], agent actions are actually sampled once for every two time-steps and the same action is applied for both time-steps.

G. Results

Fig. 8 depicts the reward curve over the 400 steps of the training cycle. The flattening near the end of the curve indicates that the training has almost completely converged. Thus, we have at least found a local minimum for the total discounted outflow.

As can be seen in Fig. 9, the partially autonomous system stabilizes the outflow around an average value of 1000

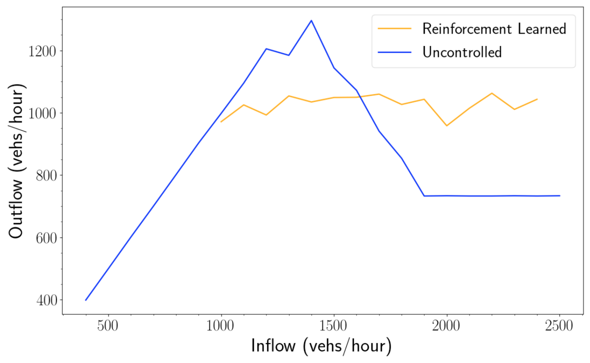

Fig. 9: Inflow vs. outflow for the bottleneck for the automated vehicle case (orange) and the uncontrolled case (blue). The solid line represents the average over 10 runs at each inflow value.

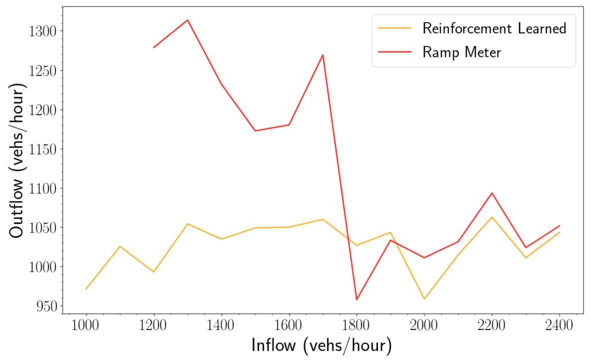

Fig. 10: Inflow vs. outflow for the bottleneck for the automated vehicle case (orange) and the feedback controlled ramp meter (red). The solid line represents the average over 10 runs at each inflow value for RL and 20 for the ramp meter.

vehicles. For values below an inflow of 1600, it under-performs the uncontrolled case, but consistently outperforms it from that point on. Furthermore, it has learned to control the system outside of the distribution it was trained on, with the control successfully extending up to an inflow of 2500 vehicles per hour despite only being trained up to an inflow of 2000 vehicles per hour.

Fig. 10 depicts the results of running 20 iterations of the feedback ramp meter over the range (1200, 2400) in steps of 100 and computing the average over each set of runs.

Videos of the results can be found at https://sites.google.com/view/itsc-lagrangian-avs/home.

IV. Discussion

As demonstrated in Fig. 9, RL has managed to learn a control strategy for the autonomous vehicles that can effectively stabilize the bottleneck outflow at the unstable equilibrium of 1000 vehicles per hour and performs competitively with ramp metering at high inflows. Although we under-perform the uncontrolled average velocity below the inflow of 1600 vehicles per hour, as discussed in sec. III-D, this is an artifact of not running the experiments long enough for them to achieve their equilibrium state; were we to run the uncontrolled bottleneck experiments for long enough they would always reach the minimum values depicted in Fig. 6. Furthermore, even if control were under-performing at low inflows, we could imagine that at low inflow values the AVs just imitate the human vehicles and control would only be turned on at high inflows.

Additionally, Fig. 10 demonstrates a comparison of the average outflow between ramp metering and RL. As in the uncontrolled case, RL under-performs at values below the critical inflow but matches the performance of feedback ramp metering above these values.

V. Conclusions and Future work

In this work we demonstrated that low levels of autonomous penetration, in this case 10%, are sufficient to learn an effective flow regulation strategy for a severe bottleneck. We demonstrate that even at low autonomous vehicle penetration rates, the controller is seemingly competitive with a ramp metering strategy.

The existence of the Mujoco benchmarks [33] has been instrumental in helping to compare different RL algorithms. In a similar vein, it is our hope that bottleneck control can serve as a benchmark for future work examining the impact of autonomous vehicles on transportation infrastructure. In this spirit, we outline a few open problems that remain.

As can be seen in the videos, the control strategy involves deciding when a given platoon is allowed to begin to exit the system. This strategy should be feasible even at much lower autonomous vehicle penetrations, so it remains to quantify the ability to control the bottleneck at different penetration rates. Furthermore, this type of control strategy should be possible to reformulate as an optimization problem; an analysis from this perspective might yield an improved strategy or one that can provide formal guarantees.

Another direction we intend to explore is the possibility of using the autonomous vehicles to achieve maximum possible outflow. As can be seen in the maximum and minimum values of Fig. 6, although a congested outflow of 800 vehicles is the equilibrium state of an inflow of 1600, there are occasional runs at which the maximum possible outflow of 1600 vehicles per hour is achieved. This suggests that in some circumstances, the uncontrolled case can stochastically arrive at a spacing of cars such that no significant series of merge conflicts occur at the bottleneck. Thus, it is possible that the autonomous vehicles can optimally space the vehicles in their platoons such that this maximum value is consistently achieved. Trying to achieve a consistent outflow of 1600 vehicles per hour at all high inflows is a possible future research direction.

Another open question is to develop strategies that are effective in the presence of lane-changing. In preliminary experiments, we found that lane-changing made it hard for the AVs to control their platoons, as the human drivers would dodge around the slower moving platoons. While it is technically true that lane-changing could be forbidden near bottlenecks, as is partially done on the San Francisco-Oakland Bay Bridge, it is possible that by effectively coordinating the platoons it could be possible to create situations where the incentive to lane-change is suppressed.

Finally, our control strategy is centralized; another approach would be to attempt to solve this problem with a decentralized strategy in which each AV is its own actor. While such a problem might be harder due to the difficulty of training policies in multi-agent reinforcement learning [34], each agent would have a significantly smaller set of possible actions which could simplify the problem.

VI. Acknowledgements*

The authors would like to thank Nishant Kheterpal and Kathy Jang for insight and help with edits. This work is supported by an AWS Machine Learning Research Award. Eugene Vinitsky is supported by an NSF Graduate Research Fellowship.

References

[1] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML*, pp. 1889–1897, 2015.

[2] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa, "Learning continuous control policies by stochastic value gradients," in *Advances in Neural Information Processing Systems*, pp. 2944–2952, 2015.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[4] A. Nagabandi, G. Yang, T. Asmar, G. Kahn, S. Levine, and R. S. Fearing, "Neural network dynamics models for control of under-actuated legged millirobots," *arXiv preprint arXiv:1711.05253*, 2017.

[5] Z. Li, P. Liu, C. Xu, H. Duan, and W. Wang, "Reinforcement learning-based variable speed limit control strategy to reduce traffic congestion at freeway recurrent bottlenecks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3204–3217, 2017.

[6] F. Zhu and S. V. Ukkusuri, "Accounting for dynamic speed limit control in a stochastic traffic environment: A reinforcement learning approach," *Transportation research part C: emerging technologies*, vol. 41, pp. 30–47, 2014.

[7] K.-L. A. Yau, J. Qadir, H. L. Khoo, M. H. Ling, and P. Komisarczuk, "A survey on reinforcement learning models and algorithms for traffic signal control," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, p. 34, 2017.

[8] B. Bakker, S. Whiteson, L. Kester, and F. C. Groen, "Traffic light control by multiagent reinforcement learning systems," in *Interactive Collaborative Information Systems*, pp. 475–510, Springer, 2010.

[9] F. Belletti, D. Haziza, G. Gomes, and A. M. Bayen, "Expert level control of ramp metering based on multi-task deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, 2017.

[10] A. Fares and W. Gomaa, "Multi-agent reinforcement learning control for ramp metering," in *Progress in Systems Engineering*, pp. 167–173, Springer, 2015.

[11]

[12] D. Etherington, "Waymo orders thousands of pacificas for 2018 self-driving fleet rollout," Feb 2018.

[13] R. E. Stern, S. Cui, M. L. D. Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, H. Pohlmann, F. Wu, B. Piccoli, *et al.*, "Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments," *arXiv preprint arXiv:1705.01693*, 2017.

[14] H. Liu, X. D. Kan, S. E. Shladover, X.-Y. Lu, and R. A. Ferlis, "Impact of cooperative adaptive cruise control (cacc) on multilane freeway merge capacity," tech. rep., 2018.

[15] R. Pueboobpaphan, F. Liu, and B. van Arem, "The impacts of a communication based merging assistant on traffic flows of manual and equipped vehicles at an on-ramp using traffic flow simulation," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pp. 1468–1473, IEEE, 2010.

[16] A. Kesting, M. Treiber, M. Schönhof, and D. Helbing, "Adaptive cruise control design for active congestion avoidance," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 6, pp. 668–683, 2008.

[17] C. Wu, A. Kreidieh, E. Vinitsky, and A. M. Bayen, "Emergent behaviors in mixed-autonomy traffic," in *Conference on Robot Learning*, pp. 398–407, 2017.

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

[19] Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S.-i. Tadaki, and S. Yukawa, "Traffic jams without bottlenecksexperimental evidence for the physical mechanism of the formation of a jam," *New journal of physics*, vol. 10, no. 3, p. 033001, 2008.

[20] F. L. Hall and K. Agyemang-Duah, "Freeway capacity drop and the definition of capacity," *Transportation research record*, no. 1320, 1991.

[21] K. Chung, J. Rudjanakanoknad, and M. J. Cassidy, "Relation between traffic density and capacity drop at three freeway bottlenecks," *Transportation Research Part B: Methodological*, vol. 41, no. 1, pp. 82–95, 2007.

[22] M. Papageorgiou, H. Hadj-Salem, and J.-M. Blosseville, "Alinea: A local feedback control law for on-ramp metering," *Transportation Research Record*, vol. 1320, no. 1, pp. 58–67, 1991.

[23] R. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, pp. 679–684, 1957.

[24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[25] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, pp. 1889–1897, 2015.

[26] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[27] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, pp. 128–138, December 2012.

[28] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning*, pp. 1329–1338, 2016.

[29] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, J. Gonzalez, K. Goldberg, and I. Stoica, "Ray rllib: A composable and scalable reinforcement learning library," *arXiv preprint arXiv:1712.09381*, 2017.

[30] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, pp. 1071–1079, 2014.

[31] A. D. Spiliopoulou, I. Papamichail, and M. Papageorgiou, "Toll plaza merging traffic control for throughput maximization," *Journal of Transportation Engineering*, vol. 136, no. 1, pp. 67–76, 2009.

[32] M. G. Bellemare, G. Ostrovski, A. Guez, P. S. Thomas, and R. Munos, "Increasing the action gap: New operators for reinforcement learning.," in *AAAI*, pp. 1476–1483, 2016.

[33] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033, IEEE, 2012.

[34] L. Mescheder, S. Nowozin, and A. Geiger, "The numerics of gans," in *Advances in Neural Information Processing Systems*, pp. 1823–1833, 2017.