

Flow: Architecture and Benchmarking for Reinforcement Learning in Traffic Control

Cathy Wu*, Aboudy Kreidieh†, Kanaad Parvate*, Eugene Vinitsky‡, Alexandre M Bayen*†§

*UC Berkeley, Electrical Engineering and Computer Science

†UC Berkeley, Department of Civil and Environmental Engineering

‡UC Berkeley, Department of Mechanical Engineering

§UC Berkeley, Institute for Transportation Studies

arXiv:1710.05465v1 [cs.AI] 16 Oct 2017

Abstract—Flow is a new computational framework, built to support a key need triggered by the rapid growth of autonomy in ground traffic: controllers for autonomous vehicles in the presence of complex nonlinear dynamics in traffic. Leveraging recent advances in deep Reinforcement Learning (RL), Flow enables the use of RL methods such as policy gradient for traffic control and enables benchmarking the performance of classical (including hand-designed) controllers with learned policies (control laws). Flow integrates traffic microsimulator SUMO with deep reinforcement learning library rllab and enables the easy design of traffic tasks, including different networks configurations and vehicle dynamics. We use Flow to develop reliable controllers for complex problems, such as controlling mixed-autonomy traffic (involving both autonomous and human-driven vehicles) in a ring road. For this, we first show that state-of-the-art hand-designed controllers excel when in-distribution, but fail to generalize; then, we show that even simple neural network policies can solve the stabilization task across density settings and generalize to out-of-distribution settings.

Index Terms—Learning and Adaptive Systems; Deep Reinforcement Learning; Traffic Microsimulation

I. INTRODUCTION

Transportation accounts for 28% of energy consumption in the US. Workers spent on aggregate over three million driver-years commuting to their jobs [1], with significant impact on nation-wide congestion. Based on 2012 estimates, U.S. commuters experienced an average of 52 hours of delay per year, causing \$121 billion of delay and fuel costs annually [2]. Depending on its use in traffic, automation has the potential to achieve many benefits or to exacerbate problems at the system level, with potential amelioration or worsening of various system metrics including *greenhouse gas* (GHG) emissions, *vehicle miles traveled* (VMT), *total travel time* (TTT). Estimates project that 2% of fuel consumption today is wasted due to congestion, a figure that rises to 4.2% in 2050 [3]. As such, the potential efficiency improvement provided by autonomous vehicles is two to four percent of total fuel consumption due to the alleviation of congestion alone.

In recent breakthrough experiments, Stern et al. [4] demonstrated a reduction in fuel consumption over 40% by the insertion of an autonomous vehicle in ring traffic to dampen the famous ring instabilities displayed by Sugiyama et al. in his seminal 2008 experiment [5]. This very disruptive field

operational test is one of the motivations for the present work: it demonstrates the power of automation and its potential impact on complex traffic phenomena such as *stop-and-go* waves [6].

The breakthrough results [4], [7] are part of a broader core set of robotics challenges concerning the deployment of multi-agent automation systems, such as fleets of self-driving cars [8], [9], coordinated traffic lights [10], [11], or other coordinated infrastructure. Robotics has already demonstrated tremendous potential in improving transportation systems through autonomous vehicles research; highly related problems include localization [12], [13], [14], path planning [15], [16], collision avoidance [17], and perception [18] problems. Considerable progress has also been made in recent decades in vehicle automation, including anti-lock braking systems (ABS), adaptive cruise control, lane keeping, automated parking, etc. [19], [20], [21], [22], which also have great potential to improve energy efficiency and safety in traffic. Down the road, the emergence of *automated districts*, i.e. districts where all vehicles are automated and operate efficiently with collaborative path-planning, might push this paradigm to next generation mobility [23]. Fleets of autonomous vehicles have recently been explored in the context of shared-mobility systems, such as autonomous mobility-on-demand systems, which abstracts out the low-level vehicle dynamics and considers a queuing theoretic model. Low-level vehicle dynamics, however, are of crucial importance, as exhibited by [24] and because many traffic phenomena, which affect energy consumption, safety, and travel time are exhibited at the level of low-level dynamics [5], [25], [26], [27]. In some settings, model-based controllers enable analytical solutions, or tractable algorithmic solutions. However, often, due to the nonlinearity of the models, numerous guarantees are lost in the process of developing controllers (i.e. optimality, run-time, complexity, approximation ratio, etc.). For example, while the ring setting enables elegant controllers to work in practice, the extension of these results (both theoretical and experimental) to arbitrary settings (network topologies, number of lanes, heterogeneity of the fleet, etc.) is challenging.

Deep reinforcement learning (RL), which is the main enabler in our framework, is a powerful tool for control and has already had demonstrated success in complex but data-rich problem settings such as Atari games [28], 3D locomotion and manipulation [29], [30], [31], chess [32], among others.

Corresponding author: Cathy Wu (cathywu@eecs.berkeley.edu)
Email addresses: {aboudy, kanaad, evinitsky, bayen}@berkeley.edu

RL testbeds exist for different problem domains, such as the *Arcade Learning Environment* (ALE) for Atari games [33], DeepMind Lab for a first-person 3D game [34], OpenAI gym for a variety of control problems [35], FAIR TorchCraft for Starcraft: Brood War [36], MuJoCo for multi-joint dynamics with Contact [37], TORCS for a car racing game [38], among others. DeepMind and Blizzard will collaborate to release the Starcraft II AI research environment [39]. Each of these RL testbeds enables the study of control through RL of a specific problem domain by leveraging of the data-rich setting of simulation. One of the primary goals of this article is to present a similarly suitable RL testbed for traffic dynamics by making use of an existing traffic simulator.

These recent advances in deep RL provide a promising alternative to model-based controller design, which the present article explores. One key step in the development of such paradigms is the ability to provide high fidelity microsimulations of traffic that can encompass accurate vehicle dynamics to simulate the action of these new RL-generated control policies, a pre-requisite to field experimental tests. This is precisely one of the aims of the present article. RL promises an approach to design controllers using black box machine learning systems. It still requires physical vehicle response to be incorporated in the simulation to learn controllers that match physical vehicle dynamics. This problem extends beyond the vehicle for which the controller is to be designed. For example, although vehicle velocity is intuitive as a control variable, it is important to keep in mind that other variables, such as actuator torques, are those actually controlled; another example is that the input may consist of data from cameras, LIDAR, or radar. The conversion between the variables might or might not be direct and may require the design of additional controllers, the performance of which would also have to be considered.

In the present article, we propose the first (to our knowledge) computational framework and architecture to systematically integrate deep RL and traffic microsimulation, thereby enabling the systematic study of autonomous vehicles in complex traffic settings, including mixed-autonomy and fully-autonomous settings. Our framework permits both RL and classical control techniques to be applied to microsimulations. As classical control is a primary approach for studying traffic dynamics, supporting benchmarking with such methods is crucial for measuring progress of learned controllers. As an illustration, this article provides a benchmark of the relative performance of learned and explicit controllers [4] for the mixed-autonomy ring road setting. The computational framework encompasses model-free reinforcement learning approaches, which complement model-based methods such as model-based reinforcement learning, dynamic programming, optimal control, and hand-designed controllers; these methods dramatically range in complexity, sometimes exhibiting prohibitive computational costs. Our initial case study investigates microscopic longitudinal dynamics (forwards-backwards) [40] and lateral dynamics (left-right) [41] of vehicles. We study a variety of network configurations, and our proposed framework largely extends to other reinforcement learning methods and other dynamics and settings, such as coordinated behaviors

[42], other sophisticated behavior models, and more complex network configurations.

The contribution of this article includes three components, (1) a computational framework and architecture, which provides a rich design space for traffic control problems and exposes model-free RL methods, (2) the implementation of several instantiations of RL algorithms that can solve complex control tasks, and (3) a set of use cases that illustrates the power of the building block and benchmark scenarios. Specifically, our contributions are:

- Flow, a computational framework for deep RL and control experiments for traffic microsimulation. Flow integrates the traffic microsimulator SUMO [43] with a standard deep reinforcement learning library rllab [44], thereby permitting the training of large-scale reinforcement learning experiments at scale on *Amazon Web Services (AWS) Elastic Compute Cloud* (EC2) for traffic control tasks. Our computational framework is open-source and available at <https://github.com/cathywu/flow>.
- An interface, provided by Flow for the design of traffic control tasks, including customized configurations of different road networks, vehicle types and vehicle dynamics, noise models, as well as other attributes provided by a standard *Markov Decision Process* (MDP) interface.
- Extensions of SUMO to support high frequency simulation and greater flexibility in controllers.
- Exposing model-free reinforcement algorithms for discounted finite (partially observed) MDPs such as policy gradient, with specific examples including *Trust Region Policy Optimization* (TRPO) [30] and *Generalized Advantage Estimation* (GAE) [29], to the domain of traffic control problems.
- Benchmarking of relative performance of learned and explicit controllers in rich traffic control settings. We present a benchmark on the mixed-autonomy single-lane ring road network and find that a reinforcement learning agent is capable of learning policies exceeding the performance of state-of-the-art controllers. The particular case of Sugiyama instabilities [5] is used to demonstrate the power of our tool.
- Case studies for building block networks. We demonstrate deep RL results on traffic at the level of multi-agent vehicle control. We demonstrate all-human, mixed-autonomy, and fully-autonomous experiments on more complex traffic control tasks, such as a multi-lane ring road and a figure 8 network. We provide additional networks, including a merge network and an intersection networks.

The rest of the article is organized as follows. Section II provides background on the RL framework used in the rest of the article. Section III describes the architecture of Flow and the processes it can handle in the three computational environments they are run (incl. SUMO and rllab). Section IV presents the various building blocks used by SUMO for building general networks (underlying maps). Section V presents the various settings for the optimization, incl. action / observation space, reward functions and policies. This is

followed by two experimental sections: in Section VI, in which we benchmark the performance of the RL-based algorithm to the seminal *FollowerStopper* controller [4], and Section VII which presents a series of various other experiments on the building block network modules of Flow. Finally, Section VIII presents related work to place this in the broader context of traffic flow modeling, deep RL and microsimulations.

II. PRELIMINARIES

In this section, we define the notation used in subsequent sections.

The system described in this article solves tasks which conform to the standard interface of a finite-horizon discounted *Markov decision process* (MDP) [45], [46], 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, $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ is the transition probability distribution, $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ is the reward function, $\rho_0 : \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ 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 interface of a *partially observable Markov decision process* (POMDP), two more components are required, namely Ω , a set of observations, and $\mathcal{O} : \mathcal{S} \times \Omega \rightarrow \mathbb{R}_{\geq 0}$, the observation probability distribution.

RL studies the problem of how agents can learn to take actions in its environment to maximize its cumulative reward. The Flow framework uses policy gradient methods [47], a class of reinforcement learning algorithms which optimize a stochastic policy $\pi_\theta : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}_{\geq 0}$. These algorithms iteratively update the parameters of the policy through optimizing the expected cumulative reward using sampled data from SUMO. The policy usually consists of neural networks, and may be of several forms. Two policies used in this article are the *Multilayer Perceptron* (MLP) and *Gated Recurrent Unit* (GRU). MLP is a classical artificial neural network with multiple hidden layers and utilizes backpropagation to optimize its parameters [48]. GRUs are recurrent neural network capable of storing memory on the previous states of the system through the use of parametrized update and reset gates, which are also optimized by the policy gradient method [49]. This enables GRUs to make decisions based on both current input and past inputs.

The autonomous vehicles in our system execute controllers which are parameterized policies, trained using policy gradient methods. For all experiments in this article, we use the *Trust Region Policy Optimization* (TRPO) [30] policy gradient method for learning the policy, linear feature baselines as described in [44], discount factor $\gamma = 0.999$, and step size 0.01. For most experiments, a diagonal Gaussian MLP policy is used with hidden layers (100, 50, 25) and \tanh non-linearity. The experiment stabilizing the ring, described later, uses a hidden layer of shape (3,3). For experiments requiring memory, a GRU policy with hidden layers (5,) and \tanh non-linearity is used.

III. OVERVIEW OF FLOW

Flow is created to fill the gap between modern machine learning and complex control problems in traffic. Flow is

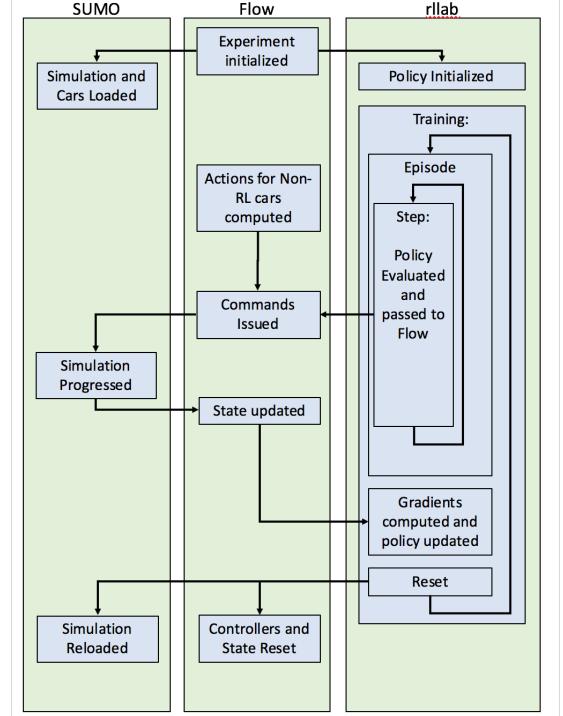


Fig. 1: Flow Process Diagram. Flow interfaces SUMO via TraCI with rllab to permit the construction and simulation of traffic MDPs and for the training and evaluation of policies (control laws). After initializing the simulation in some initial configuration, rllab collects samples by advancing and observing the simulation. In each step, vehicles are provided actions through a pre-specified controller or through a policy (via rllab). These actions are then applied via TraCI and the simulation progresses. At the end of an episode, rllab issues a reset command to the environment, which returns vehicles to their initial (possibly random) position.

a computational framework for traffic microsimulation with RL methods. Although the architecture is agnostic to specific machine learning and traffic software packages, we chose to integrate widely used open-source tools to promote access and extension.

The first of those open-source tools is SUMO (**S**imulation of **U**rban **M**obility) [43]. SUMO is a continuous-time and continuous-space microscopic traffic simulator. It is capable of handling large road networks and of modeling the dynamics of each vehicle in the simulation. SUMO was chosen particularly for its extensibility, as it includes an API called TraCI (**T**raffic **C**ontrol **I**nterface). TraCI allows users to extend existing SUMO functionality through querying and modifying the state of the simulation, at the single time-step resolution. This allows the user to easily provide intricate, custom commands that modify the simulation directly.

Secondly, we use rllab, an open source framework that enables running and evaluating RL algorithms on a variety of different scenarios, from classic tasks such as cartpole balancing to more complicated tasks such as 3D humanoid locomotion [44]. Flow uses rllab to facilitate the training, optimization, and application of control policies that manipulate the simulation. By modeling traffic scenarios as reinforcement learning problems, we use rllab to issue longitudinal and lateral controls to vehicles. Rllab further interfaces with OpenAI Gym, another framework for the development and

evaluation of reinforcement learning algorithms. The SUMO environments built in Flow are also compatible with OpenAI Gym.

Flow encapsulates SUMO via TraCI to permit the definition and simulation of traffic MDPs for rllab to train and evaluate policies. After initializing the simulation in some initial configuration, rllab collects samples by advancing and observing the simulation. In each step, vehicles are provided actions through a pre-specified controller or through a policy. These actions are then applied via TraCI and the simulation progresses. After a specified number of timesteps (i.e. the end of a rollout) or after the simulation has terminated early (i.e. a vehicle has crashed), rllab issues a reset command to the environment, which returns vehicles to their initial (possibly random) position. The interactions between Flow, SUMO/TraCI, and rllab are illustrated in Figure 1.

In addition to learned policies, Flow supports classical control (including hand-designed controllers and calibrated models of human dynamics) for longitudinal and lateral control. Flow also supports the car following models and lane-changing models that are provided in SUMO. These models work analogously to the policies generated by rllab, providing longitudinal and lateral controls to vehicles through ordinary differential equations. Together, these controllers comprise the overall dynamics of mixed-autonomy, fully-human, or full-autonomy settings.

Additionally, Flow provides various failsafes presented in Appendix C, including the ones that are built into SUMO, to prevent the vehicles from crashing and the simulation from terminating early.

Flow can be used to perform both pure model-based control experiments by using only pre-specified controllers for issuing actions, as well as experiments with a mixture of pre-specified and learned controllers. Together, this permits the study of heterogeneous or mixed-autonomy settings.

A. Architecture of Flow

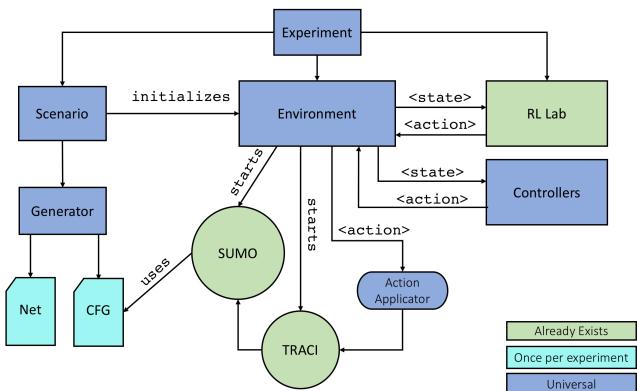


Fig. 2: Flow Architecture. A Flow experiment involves a scenario and environment, interfaced with rllab and controllers. The experiment scenario runs a generator to create road networks for use in SUMO, which is started by the environment. Controllers and rllab take experiment states and return actions, which are applied through SUMO’s TraCI API. (See Section III-A).

An experiment using Flow requires defining two components: a scenario and an environment. These and several

supporting components as well as their interactions are summarized in Figure 2.

The **scenario** for an experiment specifies network configuration in the form of network shape and attributes, for example two-lane loop road with circumference 200m, or by importing OpenStreetMap data (see Figure 3). Based on the specifications provided, the net and configuration files needed by SUMO are generated. The user also specifies the number and types of vehicles (car following model and a lane-change controller), which will be placed in the scenario.

The **generator** is a predefined class, which allows for rapid generation of scenarios with user-defined sizes, shapes, and configurations. The experiments presented in this article include large loop roads generated by specifying the number of lanes and ring circumference, figure eight networks with a crossing intersection, closed loop “merged” networks, and standard intersections.

The **environment** encodes the MDP, including functions to step through the simulation, retrieve the state, sample and apply actions, compute the reward, and reset the simulation. The environment is updated at each timestep of the simulation and, importantly, stores each vehicle’s state (e.g. position and velocity). Information from the environment is provided to a controller or passed to rllab to determine an action for a vehicle to apply, e.g. an acceleration. Note that the amount of information provided to either RL or to a controller can be restricted as desired, thus allowing fully observable or partially observable MDPs. This article studies both fully and partially observed settings.

When provided with actions to apply, Flow calls the **action applicator** which uses TraCI to enact the action on the vehicles. Actions specified as accelerations are converted into velocities, using numerical integration and based on the timestep and current state of the experiment. These velocities are then applied to vehicles using TraCI.

IV. NETWORKS

Flow currently supports learning policies on a variety of networks with a fixed number of vehicles. These include closed networks such as single and multi-lane ring roads, figure eight networks, and loops with merge as well as open networks, such as intersections. See Figure 3 for various example networks supported by Flow. In each of these networks, Flow can be used to study the design or learning of controllers which optimize the system-level velocity or fuel consumption, in the presence of different types of vehicles, model noise, etc.

Single-lane Ring Roads: The ring road network consists of a circular lane with a specified length, inspired by the 230m track studied by Sugiyama et al. [5]. This network has been extensively studied and serves as an experimental and numerical baseline for benchmarking.

Multi-lane Ring Roads: Multi-lane ring roads are a natural extension to problems involving a single lane ring. The inclusion of lane-changing behavior in this setting makes studying such problems exceedingly difficult from an analytical perspective, thereby constraining most classical control techniques to the single-lane case. Many multi-lane models

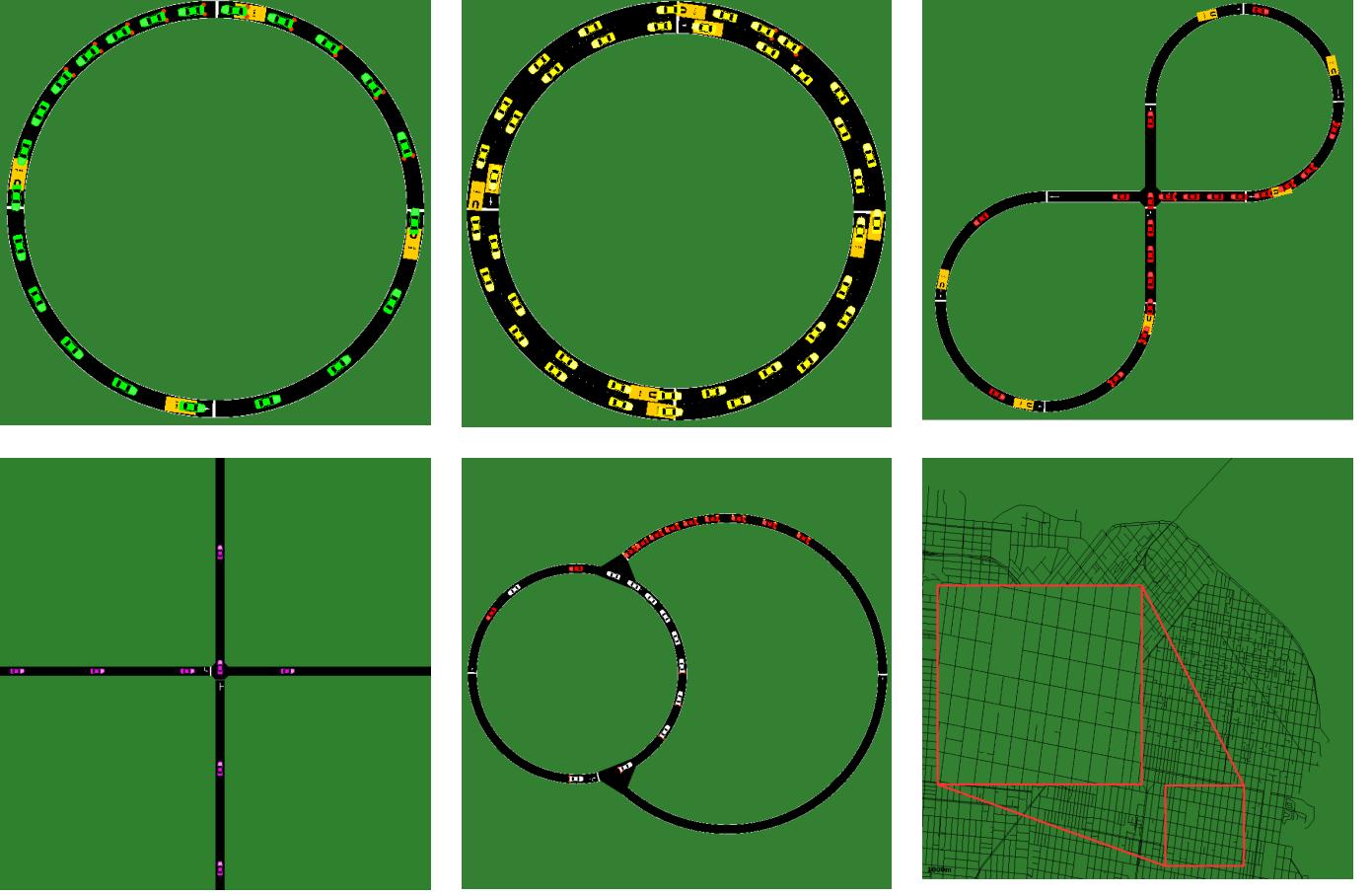


Fig. 3: Various network building blocks supported by the Flow framework. **Top left:** Single-lane Ring Road Network. **Top middle:** Multi-Lane Ring Road Network. **Top right:** Figure-Eight Road Network. **Bottom left:** Intersection Network. **Bottom Middle:** Closed Loop Merge Network. **Bottom right:** Imported San Francisco Network (currently operational for forward simulation). Maps in Flow can be generated from OSM data and visualized using SUMO.

forgo longitudinal dynamics in order to encourage tractable analysis [50], [51], [52], [53]. Recent strides have been made in developing simple stochastic models that retain longitudinal dynamics while capturing lane-changing dynamics in a single lane setting [54]. Modern machine learning methods, however, do not require a simplification of the dynamics for the problem to become tractable, as explored in Section VII.

Figure Eight: The figure eight network is a simple closed network with an intersection. Two ring roads, placed at opposite ends of the network, are connected by two perpendicular intersections. Vehicles that try to cross this intersection from opposite ends are constrained by a right-of-way model provided by SUMO to prevent crashes.

Loops with Merges: This network permits the study of merging behavior in closed loop networks. This network consists of two ring roads which are connected together. Vehicles in the smaller ring stay within this ring, while vehicles in the larger ring try to merge into the smaller ring and then back out to the larger ring. This typically results in congestion at the merge points.

Intersections: This network permits the study of intersection management in an open network. Vehicles arrive in the control zone of the intersection according to a Poisson distribution. At

the control zone, the system speeds or slows down vehicles to either maximize average velocity or minimize experienced delay. The building block can be used to build a general schema for arbitrary maps such as one the one shown in Figure 3 (bottom right).

V. TASK SPACE

Flow provides an interface for fine-grained traffic control task design. This section describes the options in the task design space, beyond the selection of a network configuration, as described in Section IV.

Action Space: When following a pre-defined route, a vehicle performs longitudinal (acceleration) and lateral (lane-changing) actions. Accordingly, for tasks with k autonomous vehicles, the action space is a set of accelerations $c \in \mathbb{R}^k$ and lane-changing decisions $d \in [-1, 1]^k$. The lane-changing values are rounded to the nearest integer (-1, 0, 1) denoting lane-change left, do not lane-change, and lane-change right, respectively; this keeps the action space representation continuous. In cases where the network only has one lane, the action space may be reduced to solely a set of accelerations.

Observation Space: The observation space may be any set of state information the user wishes to provide to the agent.

This information may fully or partially describe the state of the environment. For instance, the autonomous vehicles may observe only the preceding vehicle, only nearby vehicles, or all vehicles and their corresponding position, relative position, velocity, lane information, etc.

Custom Reward Functions: The reward function can be any function of vehicle speed, position, fuel consumption, acceleration, distance elapsed, etc. Note that existing OpenAI Gym environments (atari and mujoco) come with a pre-specified reward function [35]. However, depending on the context, a researcher, control engineer, or planner may desire a different reward function or may even want to study a range of reward functions.

For all experiments presented in this article, we evaluate the reward on the average velocity of vehicles in the network. At times, this reward is also augmented with an added penalty to discourage accelerations or excessive lane-changes by the autonomous vehicles.

Heterogeneous Settings: Flow supports traffic settings with heterogeneous vehicle types, such as those with different controllers or parameters. Additionally, simulations can contain both learning agents (autonomous vehicles) and vehicles with pre-specified controllers or dynamics. This permits the use of Flow for mixed autonomy experiments.

Noise and Perturbations: Arbitrary vehicle-level perturbations can be specified in an experiment, choosing and randomly perturbing a vehicle by overriding its control inputs and commanding a deceleration for some duration. Gaussian noise may also be introduced to the accelerations of all human car-following models in Flow.

Vehicle Placement: Flow supports several vehicle placement methods that may be used to generate randomized starting positions. Vehicles may be distributed uniformly across the length of the network or perturbed from uniformity by some noise. In addition, vehicles may be bunched together to reduce the space they take up on the network initially, and spread out across one or multiple lanes (if the network permits it); these create configurations resembling traffic jams. Finally, the sequence in which vehicles are placed in the system may also be randomly shuffled, and thus their ordering in the state space may be randomized.

VI. CASE STUDY: MIXED-AUTONOMY RING

This section uses Flow to benchmark the relative performance of an explicit controller and the reinforcement learning approach to a given set of scenarios. The next section will show similar outcomes of our RL approaches, including examples for which there are no known explicit controllers.

The goal of this section is to demonstrate the performance of the reinforcement learning approach on the problem introduced by the seminal work of Stern et al. [4] on the mixed-autonomy single-lane ring, following the canonical single-lane ring setup of Sugiyama et al. [5], consisting of 22 human-driven vehicles on a 230m ring track. The seminal work of Sugiyama et al. [5] shows that such a dynamical system produces backward propagating waves, causing part

of the traffic to come to a complete stop, even in the absence of typical traffic perturbations, such as lane changes and intersections. The breakthrough study of Stern et al. [4] studies the case of 21 human-driven vehicles and one vehicle following one of two proposed controllers, which we detail in Sections VI-1 and VI-2. This setting invokes a cascade of nonlinear dynamics from n (homogeneous) agents. In this and following sections, we study the potential for machine learning techniques (RL in particular) to produce well-performing controllers, even in the presence of highly nonlinear and complex settings.

We begin by defining the experimental setup and the state-of-the-art controllers that had been designed for the mixed-autonomy ring setting. We then benchmark the performance of the controller learned by Flow under the same experimental setup against the hand-designed controllers under a partially observed setting.

Experimental Scenario: In our numerical experiments, we similarly study 22 vehicles, one of which is autonomous, with ring lengths ranging between 180m and 380m, resulting in varying traffic densities. The vehicles are each 5m long and follow *Intelligent Driver Model* (IDM) dynamics with parameters specified by [55]. The IDM dynamics are additionally perturbed by Gaussian acceleration noise of $\mathcal{N}(0, 0.2)$, calibrated to match measures of stochasticity to the IDM model presented by [56]. We focus on the partially observed setting of observing only the velocity of the autonomous vehicle, the velocity of its preceding vehicle, and its *relative* position to the preceding vehicle. Each experiment runs for a finite time horizon, ranging from 150 to 300 seconds.

Definitions: We briefly present the important terms used in this case study. *Uniform flow* is an equilibrium state of the dynamical system (and a corresponding solution to the dynamics) where vehicles are traveling at a constant velocity. In this article, because the dynamical system has multiple equilibria, we use uniform flow to describe the unstable equilibrium in which the velocity is constant. We call this velocity the *equilibrium velocity* of the system. Uniform flow differentiates it from the stable equilibrium in which stop-and-go waves are formed, which does not exhibit a constant velocity.

Settings: We compare the following controllers and observation settings for the single autonomous vehicle:

- Learned agent with GRU policy with partial observation.
- Learned agent with MLP policy with partial observation.
- Proportional Integral (PI) controller with saturation with partial observation. This controller is given in [4] and is provided in Section VI-2.
- *FollowerStopper* with partial observation and desired velocity fixed at 4.15 m/s. The *FollowerStopper* controller is introduced in [4] and is provided in Section VI-1. *FollowerStopper* requires an external desired velocity, so we selected the largest fixed velocity which successfully stabilizes the ring at 260m; this is further discussed in the results.
- Fully-human setting using *Intelligent Driver Model* (IDM) controller, which is presented in more details in

Appendix A. This setting reduces to Sugiyamas result of traffic jams [5] and serves as a baseline comparison.

Explicit Controllers: In this section, we describe the two state-of-the-art controllers for the mixed-autonomy ring, against which we benchmark our learned policies generated using Flow.

1) *FollowerStopper*: Recent work by [4] presented two control models that may be used by autonomous vehicles to attenuate the emergence of stop-and-go waves in a traffic network. The first of these models is the *FollowerStopper*. This model commands the autonomous vehicles to maintain a desired velocity U , while ensuring that the vehicle does not crash into the vehicle behind it. Following this model, the command velocity v^{cmd} of the autonomous vehicle is:

$$v^{\text{cmd}} = \begin{cases} 0 & \text{if } \Delta x \leq \Delta x_1 \\ v \frac{\Delta x - \Delta x_1}{\Delta x_2 - \Delta x_1} & \text{if } \Delta x_1 < \Delta x \leq \Delta x_2 \\ v + (U - v) \frac{\Delta x - \Delta x_2}{\Delta x_3 - \Delta x_2} & \text{if } \Delta x_2 < \Delta x \leq \Delta x_3 \\ U & \text{if } \Delta x_3 < \Delta x \end{cases} \quad (1)$$

where $v = \min(\max(v^{\text{lead}}, 0), U)$, v^{lead} is the speed of the leading vehicles, Δx is the headway of the autonomous vehicle, subject to boundaries defined as:

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

The parameters of this model can be found in [4].

2) *PI with Saturation*: In addition to the *FollowerStopper* model, [4] presents a model titled the “PI with Saturation Controller” that attempts to estimate the average equilibrium velocity U for vehicles on the network, and then drives at that speed. This average is computed as a temporal average from its own history: $U = \frac{1}{m} \sum_{j=1}^m v_j^{\text{AV}}$. The target velocity at any given time is then defined as:

$$v^{\text{target}} = U + v^{\text{catch}} \times \min \left(\max \left(\frac{\Delta x - g_l}{g_u - g_l}, 0 \right), 1 \right) \quad (3)$$

Finally, the command velocity for the vehicle at time $j+1$, which also ensures that the vehicle does not crash, is:

$$v_{j+1}^{\text{cmd}} = \beta_j (\alpha_j v_j^{\text{target}} + (1 - \alpha_j) v_j^{\text{lead}}) + (1 - \beta_j) v_j^{\text{cmd}} \quad (4)$$

The values for all parameters in the model can be found in [4].

Results: Through this detailed case study of the mixed-autonomy single-lane ring, we demonstrate that Flow enables the fine-grained benchmarking of classical and learned controllers. Videos and additional results are available at <https://sites.google.com/view/ieee-tro-flow>.

1) *Performance*: Figure 6 shows several key findings. This traffic density vs. velocity plot shows the performance of the different learned and hand-designed controllers. First, we observe that GRU and MLP controllers (in partially observed settings) are capable of matching the uniform flow speed very closely for all trained densities, thereby effectively stabilizing traffic in all densities in the training range. The PI with Saturation controller, on the other hand, is only capable of properly performing at densities less than or equal to the density at which it was calibrated (less congested settings).

Figure 4 shows the velocity profiles for the different learned and hand-designed controllers for the 260m ring and additionally includes the *FollowerStopper* controller. We observe that although all controllers are able to stabilize the system, the GRU controller allows the system to reach the uniform flow equilibrium velocity most quickly. The GRU and MLP policies stabilize the system with less oscillatory behavior than the *FollowerStopper* and PI with Saturation controllers, as observed in the velocity profiles. In addition, the *FollowerStopper* controller is the least performant; the controller can only stabilize a 260m ring road to a speed of 4.15 m/s, well below the 4.82 m/s uniform flow velocity.

Finally, Figure 5 shows the space-time curves for all vehicles in the system, using a variety of controllers. We observe that the PI with Saturation and *FollowerStopper* controllers leave much smaller openings in the network (smaller headways) than the MLP and GRU policies. The MLP policy exhibits the largest openings, as can be seen by the large white portion of the MLP plot within Figure 5. If this were instead applied in a multi-lane ring study, then the smaller openings would have the benefit of preventing opportunistic lane changes, so this observation can lead to better reward design for more complex mixed-autonomy traffic studies.

2) *Robustness*: One of the strengths of our GRU and MLP policies is that it does not rely on external calibration of parameters that is specific to a particular traffic setting, such as density.

Although the PI with Saturation controller can conceptually adjust to different densities, with its moving average filter, we experimentally found that its performance is sensitive to its parameters. Using parameters calibrated for 260m ring roads (as described in [4]), the PI with Saturation controller indeed performs the best at 260m among the density range considered in this study. However, this controller’s performance quickly degrades at higher density (more congested settings), dipping close to the stop-and-go equilibrium velocity.

Similarly, the *FollowerStopper* Controller suffers from the same calibration deficiencies as the PI with Saturation Controller. Additionally, the desired velocity must be provided beforehand. Interestingly and moreover, we found that this controller often fails to stabilize the system if provided too high of a desired velocity, even if it is well below the equilibrium velocity. Instead, if a lower desired velocity is first provided as an intermediate control target, then the desired velocity may then subsequently be achieved. This suggests that a simple control law such as the *FollowerStopper* cannot optimally stabilize a mixed-autonomy ring, and additionally, that there is additional tuning and augmentation necessary to use the *FollowerStopper* controller.

3) *Generalization of the Learned Control Policy*: Training on different vehicle densities encourages the learning of a more robust policy. We found the policy to generalize even to densities outside of the training regime. Figure 6 shows the average velocity vehicles in the network achieve for the final 100s of simulation time; the gray regions indicate the test-time densities. Surprisingly, we found that even when training on different densities but in the *absence* of acceleration error in the human driver models, the learned policies successfully

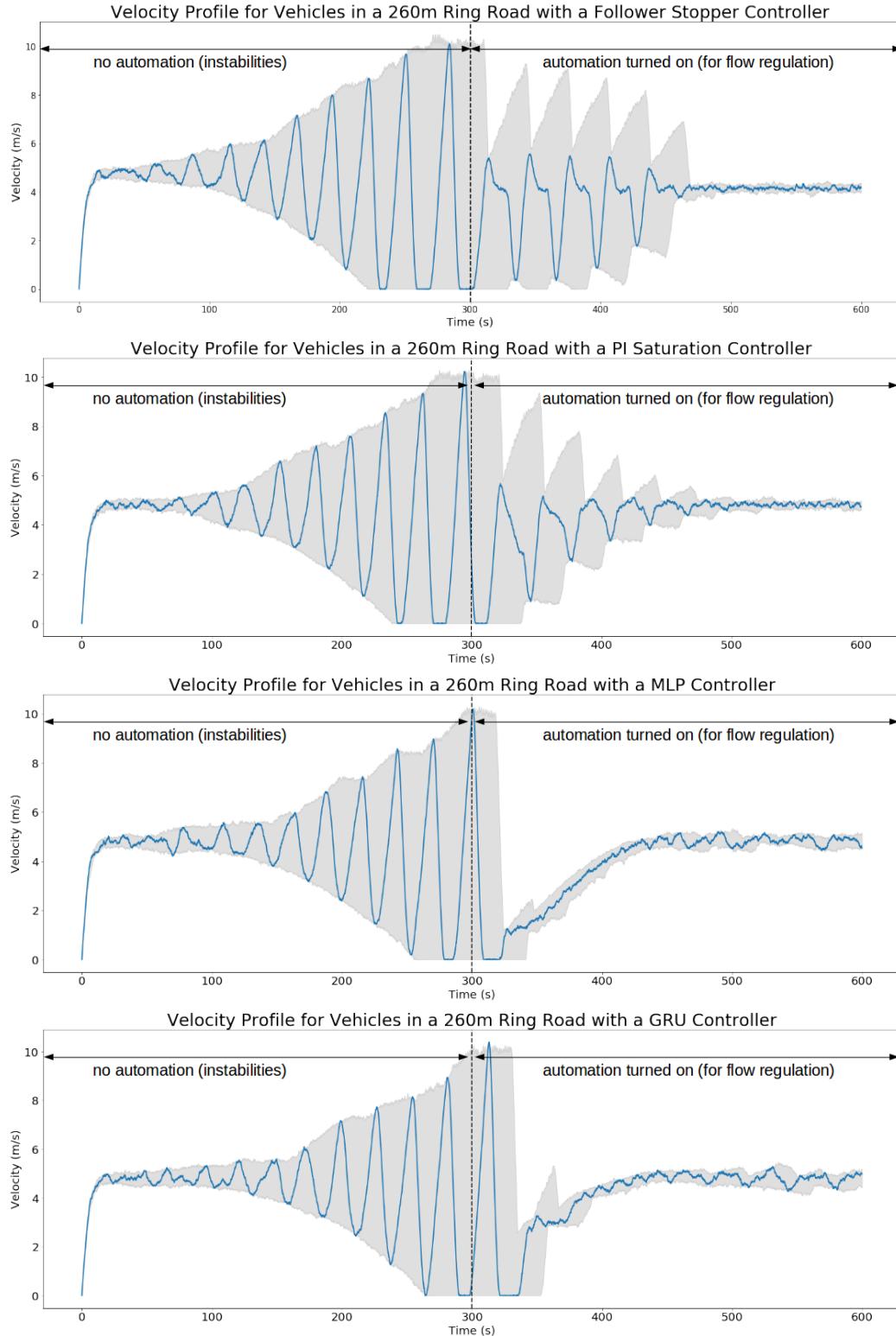


Fig. 4: All experiments are run for 300 seconds with the autonomous vehicle acting as a human driver before it is “activated”. As we can see, an autonomous vehicle trained in a partially observable ring road setting with variable traffic densities is capable of stabilizing the ring in a similar fashion to the *FollowerStopper* and PI with Saturation Controller. Among the four controller, the GRU controller allows the system to reach the uniform flow equilibrium velocity most quickly. In addition, the *FollowerStopper* controller is the most brittle and can only stabilize a 260m ring road to a speed of 4.15 m/s, well below the 4.82 m/s uniform flow velocity.

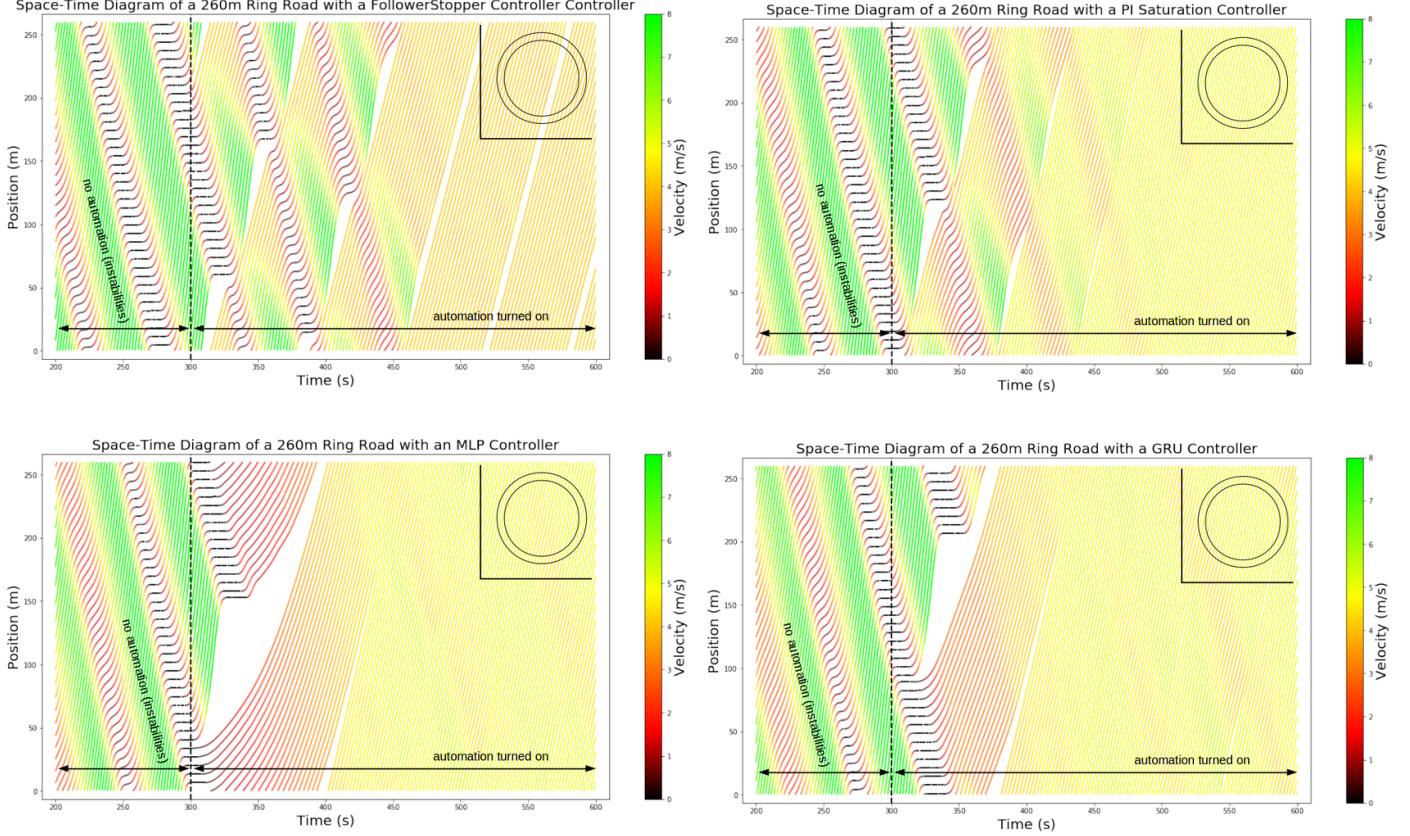


Fig. 5: Prior to the activation of the autonomous vehicles, all settings exhibit backward propagating waves resulting from stop-and-go behavior. The autonomous vehicles then perform various controlled policies aimed at attenuating these waves. **Top left:** Space-time diagram for the ring road with a *FollowerStopper* Controller. **Top Right:** Space-time diagram for the ring road with a PI with Saturation Controller. **Bottom left:** Space-time diagram for the ring road with an MLP Controller. **Bottom right:** Space-time diagram for the ring road with a GRU Controller.

stabilized settings *with* human model noise during test time.

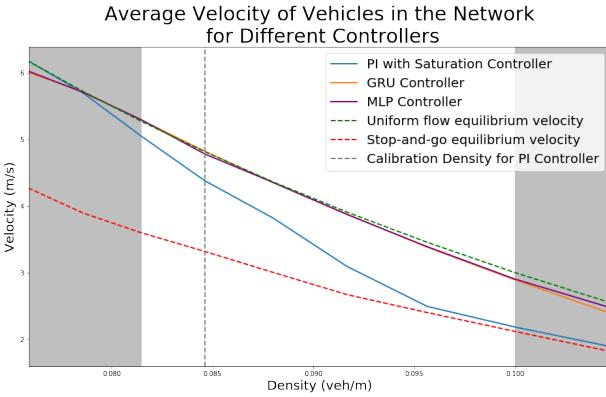


Fig. 6: The performance of the MLP, GRU, and PI Saturation controllers for various densities are averaged over ten runs for each of the tested densities. The GRU and MLP controllers are capable of matching the uniform flow speed very closely for all trained densities. The PI with Saturation controller, on the other hand, is only capable of properly performing at densities less than or equal to the density at which it was calibrated. Remarkably, the GRU and MLP controllers are also reliable enough to stabilize the system at velocities close to the uniform flow equilibrium even for densities outside the training set.

Discussion: This benchmark study demonstrates that deep RL, policy gradient methods in particular, using the same state

information provided to the hand-designed controllers and with access to samples from the overall traffic system (via a black box simulator), can learn a controller which performs better than state-of-the-art hand-designed controllers for the given setting, in terms of average system-level velocity.

This study focuses on the partially observed setting, since it is the more realistic setting for near-term deployments. Furthermore, there are hand-designed controllers in the literature for this setting, with which we can benchmark. We would expect that the fully observed setting (with an MLP policy) would perform as well if not better than our learned policies in the partially observed setting. Since our policies already closely track the equilibrium velocity curve, we do not explore the fully observed setting.

VII. FLOW EXPERIMENTS

This section uses Flow to extend the settings typically studied for controller design, presented in Section VI, to include examples more representative of real-world traffic. As traffic is a complex phenomena, it is crucial to build tools to explore more complex and realistic settings. In the following, we demonstrate these for three examples: a setting with multiple autonomous vehicles, a multi-lane ring road setting, and a figure eight road setting. A multi-lane ring

road is a simple closed network which extends the well-studied single-lane ring road but adds in lateral dynamics (lane changes).

Three natural settings for benchmarking are the fully autonomous setting, the mixed-autonomy setting, and the fully-human setting. Designed controllers for these settings may be implemented in Flow for additional benchmarking as well.

In the following experiments, a fully observable setting is assumed. The experiments are run on *Amazon Web Services* (AWS) *Elastic Compute Cloud* (EC2) instances of model *c4.2xlarge*, which have eight CPUs and 15 GB of memory.

Single-lane ring road with multiple autonomous vehicles: In this experiment, a total of 22 vehicles are placed in a ring road with a circumference of 230m. Strings of autonomous vehicles are placed consecutively, with between three and eleven autonomous vehicles. A string of consecutive autonomous vehicles learn to drive with a smaller headway than the human models, resulting in greater roadway utilization, thereby permitting a higher velocity for the overall system, as can be seen in Figure 7.

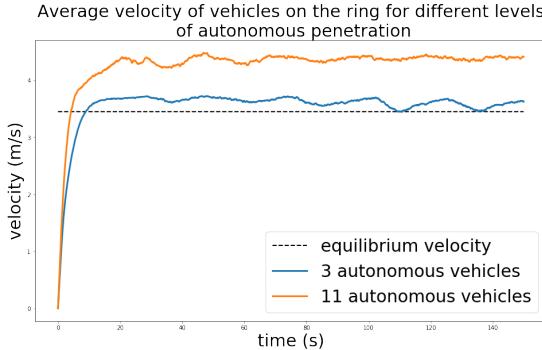


Fig. 7: Velocity profile for single-lane ring road with multiple autonomous vehicles. The addition of autonomous vehicles permits in exceeding the uniform flow equilibrium velocity. This velocity increases as the level of autonomous penetration increases. At three autonomous vehicles, the average velocity settles at 3.70 m/s; at 11 autonomous vehicles, the average velocity settles at 4.44 m/s.

Multi-lane ring road with multiple autonomous vehicles: The single-lane multiple vehicle experiment is extended to the multi-lane setting, with 44 vehicles placed in a two-lane ring road of circumference 230m. In this setting, a string of six autonomous vehicles are initialized side-by-side (all in one lane). The human-driven vehicles follow IDM longitudinal control and SUMO’s lane changing model for lateral control. The resulting average velocity is 3.66 m/s, an improvement over the 3.45 m/s uniform flow equilibrium velocity. This experiment demonstrates the reinforcement learning policy’s ability to generalize to settings with discontinuous model dynamics, such as lane changes.

Figure Eight: For this experiment, 14 vehicles are placed in a figure-eight with a ring radius of 30m and total length of 402m. The intersection in this environment is not controlled by a traffic light; instead vehicles cross the intersection following a right-of-way model provided by SUMO to prevent crashes.

In the absence of autonomous vehicles, human drivers begin queuing at the intersection, leading to a significant reduction in

the average speed of vehicles in the network. Figure 8 shows the average velocity of vehicles in the network for different levels of autonomy. With the inclusion of a single autonomous vehicle. With one autonomous vehicle, the vehicles begin moving around 1.5 times as fast as they had when they were forced to queue, while the fully autonomous setting exhibits an improvement of almost three times to the average velocities of vehicles.

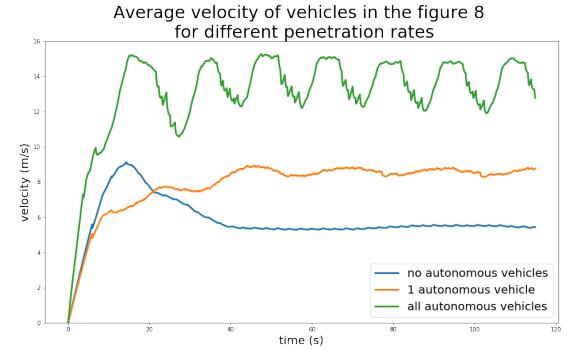


Fig. 8: Velocity profile in the figure eight for different levels of autonomous penetration. Similar to the platooning setting, the performance of the network improves as the number of autonomous vehicles improves. In particular, the inclusion of full autonomy almost triples the average velocity in the network from around 5 m/s in the absence of autonomy for around 14 m/s.

For each of these benchmarks, more investigation is required to understand the learned behaviors and policies and thereby take steps towards a real-world deployment.

VIII. RELATED WORK

Traffic Dynamics: Modeling and analysis of traffic dynamics is notoriously complex and yet is historically considered a prerequisite for traffic control [55], [57]. Researchers classically trade away the complexity of the model (and thus the realism of the model) in favor of the tractability of analysis, using high level abstraction with the goal of designing optimal controllers or other controllers with desirable properties, such as safety or comfort [58], [59], [60], [61]. Consequently, results in traffic control can largely be classified as small-scale simulation-based numerical analysis (for example, [62], [63], [64], [65]) or theoretical analysis on simple settings such as assuming non-oscillatory responses (e.g. [66]) or focusing on a single-lane ring road (e.g. [67], [68], [69], [70], [71], [72]).

In particular, with the advent of autonomous vehicles, new frameworks and techniques are urgently needed to establish a foundation for studying the control and the effects of autonomous vehicles, thereby preparing the world for their adoption. Modern reinforcement learning techniques indicate promise towards the goal of obtaining controllers with desirable (though perhaps not optimal) properties while simultaneously studying complex settings.

Deep RL and Traffic: Several recent studies incorporated ideas from deep learning in traffic optimization. Deep RL has been used for traffic prediction [73], [74] and control [75]. A deep RL architecture was used in [74] to predict traffic flows, demonstrating success even during special events with nonlinear features; to learn features to represent states

involving both space and time, [73] additionally used hierarchical autoencoding in the traffic flow prediction problem. A multi-agent deep RL algorithm was introduced in [75] to learn a policy for ramp metering. For additional uses of deep learning in traffic, we refer the reader to [76], which presents an overview comparing non-neural statistical methods versus neural networks in transportation research. These recent results demonstrate that deep learning and deep RL are a promising approach to traffic problems. This article aims to bridge the gap between deep RL and traffic control problems by providing a computational framework for learning well-performing controllers; a preliminary prototype of our architecture is published in [77].

Traffic Simulators: Traffic microsimulators include Quadstone Paramics [78], VISSIM [79], [80], AIMSUN [81], MATSIM [82], POLARIS [83], and SUMO [43]. The first three are closed-source commercial software, whereas the latter two are open source software. Each of these tools are capable of large-scale traffic microsimulation and can handle a variety of policies and control strategies. Each tool offers an *Application Programming Interface* (API) which permits overriding or extending the default models such as car following, lane changing, route choice, etc. Each of these simulators are widely used in the research community. These tools differ in their precise offerings and features, such as visualization tools, supported models, and simulation speed. Because most studies focus their study on a single simulator, a comprehensive comparison of these tools is unfortunately lacking.

In the present work, we choose to integrate SUMO, an open-source, extensible, microscopic simulator that can simulate large road networks. SUMO discretizes time and progresses the simulation for a user-specified timestep; furthermore, because SUMO is microscopic, individual vehicles are controlled by car following models—functions of the vehicle’s headway, velocity and the velocity of the preceding vehicle. The acceleration provided by the car following model is applied as a change of velocity over the course of the next timestep. SUMO’s car following models include IDM, IDMM, and Wiedermann.

SUMO has several current issues which limit its suitability for RL. First, all SUMO built-in car following models are configured with a minimal time headway, τ , that is used to ensure safety [84], and do not support time delays. Second, SUMO’s car following models are calibrated for a simulation timestep of 1.0 seconds, and their behavior for smaller timesteps is known to produce unnatural behaviors [85] whereas we would like to simulate at 10-100ms timesteps. Finally, there does not yet exist an interface between SUMO and RL libraries. Because the results of an RL experiment rely on the realism of the model/simulator, we need the traffic models to capture more realistic fine-grained dynamics, including operating at a higher granularity (smaller simulation step), with a different model of time delays, with acceleration-based control, etc.

Our work aims to address each of these limitations. Flow extends SUMO to permit rich custom controllers which may operate at smaller simulation steps and with time delays. These

richer control actions allow Flow to support a larger class of controllers, thus permitting a more realistic and suitable testbed for reinforcement learning in traffic dynamics. SUMO also includes a Python API called *TRAffic Control Interface* (TraCI), from which the user can retrieve information about the vehicles’ current states and issue precise commands to set the vehicles’ velocities, positions, and lanes. Using this interface, we can interface SUMO with RL libraries, read out state information, issue actions, define our own car following models, etc.

IX. CONCLUSION

Flow is a computational framework built on open source tools; it enables learning policies for autonomous vehicles in complex traffic settings involving nonlinear vehicle dynamics and arbitrary network configurations. This article demonstrates its capabilities and provides several concrete examples and a case study which effectively benchmarks learned policies against established control results. The expansion and combination of benchmark networks to additional network types, including arbitrary grid networks, more complex intersections, and importing arbitrary map networks, is the subject of ongoing work, and will be operational soon (it is already functional for simulation). More advanced RL algorithms will be developed alongside larger networks because current algorithms suffer poor sample complexity in the presence of combinatorial structures such as graphs (road networks) [86] and multiple agents [87]. Interesting and promising future directions include extending Flow to support additional features, such as evaluating safety (in addition to efficiency), using Flow as a tool to design specific controllers (which can be interpreted or for which properties such as optimality can be proven), and using it to inform public policy in preparation for the increased adoption of autonomous vehicles. Finally, as seen in many traffic management project led by State agencies, microsimulation tools are the last step before field implementation, which we hope to see for this work as well.

ACKNOWLEDGMENTS

The authors would like to thank Leah Dickstein and Nathan Mandi for early prototyping, Nishant Kheterpal, Kathy Jang, Saleh Albeik and Ananth Kuchibhotla for helping to build out the features, Rocky Duan and Alex Lee for rllab support, Jakob Erdmann for SUMO support, and Professor Alexander Skabardonis for several insightful discussions about vehicle dynamics and fail-safes. The team is extremely grateful to Professor Dan Work for technical conversations about the ring experiment and work, and to the inspirational work of the Piccoli-Seibold-Sprinkle-Work team.

APPENDIX A CLASSICAL CONTROLLERS

This section presents several classical controllers available in Flow that have been and may be used to model human or non-human driving behavior during experimentation.

A. Longitudinal controllers

Longitudinal Controllers: Longitudinal dynamics are usually defined by car following models [67]. Standard *car following models* (CFMs) are of the form:

$$a_i = \dot{v}_i = f(h_i, \dot{h}_i, v_i), \quad (5)$$

where the acceleration a_i of vehicle i is some typically nonlinear function of h_i, \dot{h}_i, v_i , which are respectively the headway, relative velocity, and velocity for vehicle i . A general model may include time delays from the input signals h_i, \dot{h}_i, v_i to the resulting output acceleration a_i . Example CFMs include the *Intelligent Driver Model* (IDM) [88] and the *Optimal Velocity Model* (OVM) [89], [90]. Our presented system implements several known CFMs and provides an easy way to implement custom CFMs.

Custom longitudinal controllers can be implemented in Flow using methods similar to the general car following model equation (5) shown above, in which a vehicle's acceleration is some function of its speed, headway, and relative velocity. Car following models are not limited to those inputs, however; full access to the state of the environment at each timestep is provided to controllers.

Out of the box, Flow supports a variety of car following models, including SUMO default models and custom models not provided by SUMO. Each model specifies the acceleration for a vehicle at a given time, which is commanded to that vehicle for the next time-step using Traci.

Controllers with arbitrary time delays between perception and action are supported in Flow. Delays are implemented by storing control actions in a queue. For delayed controllers, a new action is computed using the state at each timestep and enqueued, and an action corresponding to some previous state is dequeued and commanded. Descriptions of supported car-following models follow below.

1) *Second-order linear model*: The first, and simplest, car following model implemented is the forward-looking car following model specified in [67]. The model specifies the acceleration of vehicle i as a function of a vehicle's current position and velocity, as well as the position and velocity of the vehicle ahead. Thus: $\dot{v}_i = k_d(d_i - d_{\text{des}}) + k_v(v_{i-1} - v_i) + k_c(v_i - v_{\text{des}})$ where v_i, x_i are the velocity and position of the i -th vehicle, $d_i := x_{i-1} - x_i$ is the headway for the i -th vehicle, k_d, k_c, k_v are controller gains for the difference between the distance to the leading car and the desired distance, relative velocity, and the difference between current velocity and desired velocity, respectively. In addition, $d_{\text{des}}, v_{\text{des}}$ are the desired headways and velocities respectively.

2) *Intelligent Driver Model*: The *Intelligent Driver Model* (IDM) is a microscopic car-following model commonly used to model realistic driver behavior [88]. Using this model, the acceleration for vehicle α is defined by its bumper-to-bumper headway s_α (distance to preceding vehicle), velocity v_α , and relative velocity Δv_α , via the following equation:

$$a_{\text{IDM}} = \frac{dv_\alpha}{dt} = a \left[1 - \left(\frac{v_\alpha}{v_0} \right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha} \right)^2 \right] \quad (6)$$

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

$$s^*(v_\alpha, \Delta v_\alpha) = s_0 + \max \left(0, v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}} \right) \quad (7)$$

where $s_0, v_0, T, \delta, a, b$ are given parameters. Typical values for these parameters can be found in [88].

3) *Optimal Velocity Model (OVM)*: Another car following model implemented in Flow is the optimal velocity model from [69]. A variety of optimal velocity functions exist for use in specifying car following models [91], [55]; [69] uses a cosine-based function to define optimal velocity $V(h)$ as a function of headway:

$$V(h) = \begin{cases} 0 & h \leq h_{\text{st}} \\ \frac{v_{\text{max}}}{2} (1 - \cos(\pi \frac{h-h_{\text{st}}}{h_{\text{go}}} - h_{\text{st}})) & h_{\text{st}} < h < h_{\text{go}} \\ v_{\text{max}} & h \geq h_{\text{go}} \end{cases}$$

The values $h_{\text{st}}, h_{\text{go}}$ correspond to headway thresholds for choosing an optimal velocity, so that for headways below h_{st} , the optimal velocity is 0, and for headways above h_{go} , the optimal velocity is some maximum velocity v_{max} . The optimal velocity transitions using a cosine function for headways between h_{st} and h_{go} . $V(h)$ is used in the control law for the acceleration of the i -th vehicle, where $\dot{v}_i = \alpha[V(h_i) - v_i] + \beta[v_{i-1} - v_i]$ at each timestep. This controller can also be implemented with delay to simulate perception and reaction times for human drivers, in which case $\dot{v}_i(t)$ would be a function of states $h_i(t-\tau), v_i(t-\tau), v_{i-1}(t-\tau)$.

4) *Bilateral control model (BCM)*: The bilateral controller presented by [70], [71] considers not only the relation of a subject vehicle to the vehicle ahead but also to the vehicle behind it. In their controller, the subject vehicle's acceleration depends on the distance and velocity difference to both the vehicle ahead and behind, with

$$\dot{v}_i = k_d h_i + k_v((v_{i-1} - v_i) - (v_i - v_{i+1})) + k_c(v_i - v_{\text{des}})$$

where $h_i := (x_{i-1} - x_i) - (x_i - x_{i+1})$. In [70], [71], Horn and Wang argue that bilateral controllers can stabilize traffic.

Lateral Controllers: SUMO has lateral dynamics models dictating when and how to lane change [92]; however, to extend lateral control to the RL framework, Flow permits the easy design of new and higher fidelity lane changing models. The current implementation of Flow includes a proof of concept lane-changing model in which vehicles change lanes stochastically based on speed advantage when adjacent lanes satisfy a set of constraints. Vehicles in Flow do not check to change lanes at each timestep, as that might lead to an excessive number of lane changes. Instead, at some time interval, the vehicle determines if it should lane change. SUMO's existing lane-changing models can also be used in a Flow experiment in place of custom models.

As with longitudinal controllers, custom lateral controllers can also be built in Flow. These lane-changing models have access to the full state of the environment at each time step to use as potential inputs. This allows, for example, a vehicle to identify all nearby vehicles in adjacent lanes and their speeds, and then send a lane-change command if a lane is clear and offers potentially higher speed. Due to the rich development

interface available, Flow supports the integration of complex lateral controllers.

APPENDIX B ADDITIONAL EXPERIMENTS

This section uses Flow to benchmark the controllers described in the previous section. The experiments in this section contain no learning components.

Single-lane Ring (all human driver models):

1) *OVM from uniform initial state*: The first experiment runs the Sugiyama setup from an initial state in which all 22 vehicles were spaced evenly around the ring road and start with the same velocity. Each of the vehicles was using a *Optimal Vehicle Model* (OVM) controller, as described in the section on controllers above. The experiment begins from a stopped state, gets up to speed, and proceeds free of traffic shockwaves for its duration.

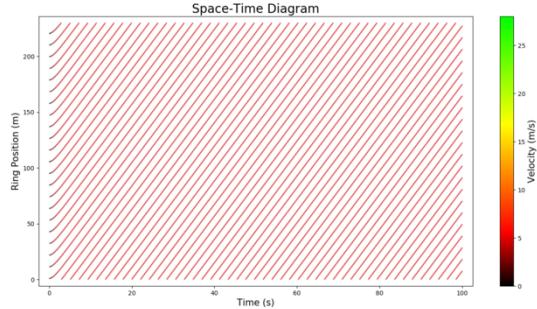


Fig. 9: 22 OVM vehicles from uniform state on a ring road, showing an average speed of 4.87 m/s across all vehicles.

2) *OVM with a perturbation (Figure 10)*: In this experiment, 22 OVM vehicles are run from a uniform, evenly-spaced starting state. No traffic shockwaves form until the system is perturbed 9 seconds into the experiment, once the vehicles have roughly reached their equilibrium velocities from the unperturbed setting. One vehicle is randomly chosen and an acceleration of -5 m/s^2 is applied for 1.5 seconds. The braking of that vehicle forces the vehicles behind it to slow down as well, and the system degrades into stop-and-go traffic.

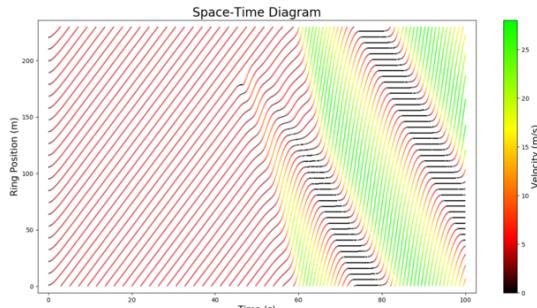


Fig. 10: 22 OVM vehicles on a ring road with a perturbation, breaking down from uniform motion into stop-and-go traffic with an average speed of 7.5 m/s.

A. OVM from a nonuniform motion state (Figure 11)

This experiment simulates the Sugiyama setup but from a non-uniform initial configuration. Starting with the first vehicle, the subsequent position of each vehicle is drawn from a Gaussian distribution with mean equal to the length of track divided by number of vehicles and a standard deviation given by one fifth the mean. The unstable starting state also incorporates a bunching factor, in which no vehicles are placed on some segment of the track, with the length of that segment being a user-defined variable. All 22 vehicles use the OVM controller. Instability is apparent from the beginning, with traffic rapidly degrading into traffic shockwaves and failing to recover.

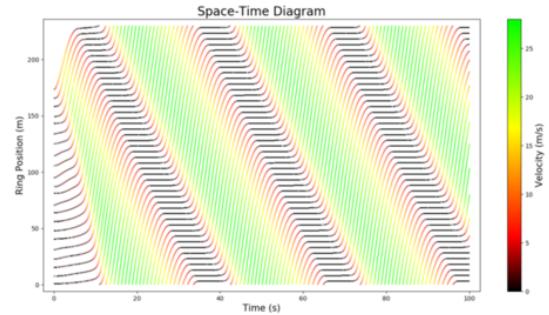


Fig. 11: 22 OVM vehicles from a nonuniform ring road initial state, showing stop-and-go traffic with an average speed of 7.8 m/s.

1) *BCM with a perturbation (Figure 12)*: 22 vehicles implementing the bilateral car following model (BCM), described in the controllers section, are implemented in this simulation. The simulation begins from a uniform, evenly-spaced starting state. As with the experiment above, a random vehicle is perturbed at an acceleration of -5 m/s^2 , 9 seconds into the simulation for 1.5 seconds. Some braking results, but unlike the OVM case described above, the BCM vehicles recover from this perturbation and traffic returns to uniform motion shortly after.

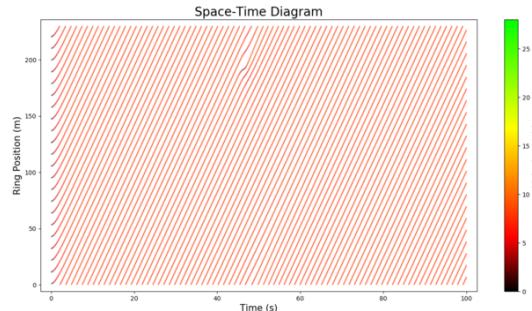


Fig. 12: 22 BCM vehicles on a ring road with a perturbation, showing an average speed of 7.9 m/s.

2) *BCM from an nonuniform state (Figure 13)*: Again, 22 BCM vehicles are run in this simulation, but from the same nonuniform starting state as in the nonuniform motion OVM case, in which vehicles are spaced randomly subject to a bunching factor. There is some initial instability and small traffic shockwaves, but again the BCM vehicles recover from this non-stable state and return to uniform motion.

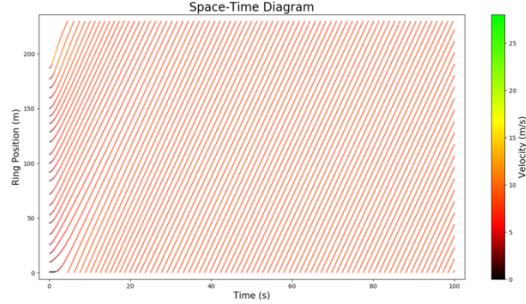


Fig. 13: 22 BCM vehicles from a nonuniform ring road initial state, showing an average speed of 7.9 m/s.

3) *Mixed BCM/OVM from a nonuniform initial state (Figure 14)*: Here, 11 BCM vehicles and 11 OVM vehicles begin from a randomly spaced, and bunched starting state as described above. The proportion of bilateral control vehicles proves sufficient to prevent the stop-and-go waves seen in the unstable OVM setting. Some velocity variation persists, however, unlike the full-BCM unstable setting which returns to a completely uniform motion state.

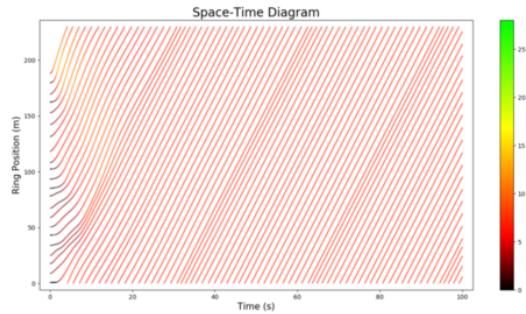


Fig. 14: 11 BCM and 11 OVM vehicles, from a nonuniform ring road initial state, showing an average speed of 7.1 m/s.

APPENDIX C FAIL-SAFES

Flow supplements its car following models with safe driving rules that prevent the inherently unstable car following models from crashing. As SUMO experiments terminate when a collision occurs, Flow provides a fail-safe mechanism, called the *final position rule*, which runs constantly alongside other controllers. Fail-safes are passed in the action commanded by the vehicle controller, regardless of whether it is an action specified by RL or a control model. Fail-safes are a standard feature in any traffic simulator that is required to handle large perturbations and string unstable traffic. The conservativeness of the fail-safe affects the braking behavior of the traffic. In general, fail-safes operate according to the principle of maintaining a minimum safe distance from the leading vehicle where the maximum acceleration and deceleration of the leading vehicle is stochastically generated [93], [94].

Final Position Rule: This fail-safe aims to keep a velocity such that if the preceding vehicle suddenly starts braking with max deceleration a , then even if the following vehicle has a delay τ it can still slow down such that it comes to

rest at the final position of the rear bumper of the preceding vehicle. If the preceding vehicle is initially at position $x_{i-1}(0)$, and decelerates maximally, it will come to rest at position $x_{i-1}(0) + \frac{v_{i-1}^2(0)}{2a}$. Because the fail-safe issues the maximum velocity, if the ego vehicle has delay τ , it will first travel a distance of $v_{\text{safe}}\tau$ and then begins to brake with maximum deceleration, which brings it to rest at position $x_i(0) + v_{\text{safe}} \cdot (\tau + \frac{v_{\text{safe}}}{2a})$.

SUMO-Imposed Safety Behavior: In addition to incorporating its own safe velocity models, Flow leverages various safety features from SUMO, which may also be used to prevent longitudinal and lateral collisions. These fail-safes serve as bounds on the accelerations and lane-changes human and autonomous vehicles may perform, and may be relaxed on any set of vehicles in the network to allow for the prospect of more aggressive actions to take place.

REFERENCES

- [1] U. DOT, “National transportation statistics,” *Bureau of Transportation Statistics, Washington, DC*, 2016.
- [2] D. Schrank, B. Eisele, and T. Lomax, “Tis 2012 urban mobility report,” *Texas A&M Transportation Inst.. The Texas A&M Univ. System*, 2012.
- [3] Z. Wadud, D. MacKenzie, and P. Leiby, “Help or hindrance? the travel, energy and carbon impacts of highly automated vehicles,” *Transportation Research Part A: Policy and Practice*, vol. 86, pp. 1–18, 2016.
- [4] R. E. Stern, S. Cui, M. L. D. Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, R. Haulcy, H. Pohlmann, F. Wu, B. Piccoli, B. Seibold, J. Sprinkle, and D. B. Work, “Dissipation of stop-and-go waves via control of autonomous vehicles: Field experiments,” *CoRR*, vol. abs/1705.01693, 2017. [Online]. Available: <http://arxiv.org/abs/1705.01693>
- [5] Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S.-i. Tadaki, and S. Yukawa, “Traffic jams without bottlenecks—experimental evidence for the physical mechanism of the formation of a jam,” *New Journal of Physics*, vol. 10, no. 3, p. 033001, 2008.
- [6] M. Garavello and B. Piccoli, *Traffic flow on networks: Conservation Laws Models*. Springfield, MO: American Institute of Mathematical Sciences, 2006.
- [7] F. Wu, R. Stern, S. Cui, M. L. D. Monache, R. Bhadani, M. Bunting, M. Churchill, N. Hamilton, R. Haulcy, B. Piccoli, B. Seibold, J. Sprinkle, and D. Work, “Tracking vehicle trajectories and fuel rates in oscillatory traffic,” *Transportation Research Part C: Emerging Technologies*, 2017.
- [8] M. Pavone, S. L. Smith, E. Frazzoli, and D. Rus, “Robotic load balancing for mobility-on-demand systems,” *The International Journal of Robotics Research*, vol. 31, no. 7, pp. 839–854, 2012.
- [9] e. a. Cathy Wu, “Emergent behaviors in mixed-autonomy traffic,” *Conference on Robot Learning*, 2017.
- [10] F. Belletti, D. Haziza, G. Gomes, and A. M. Bayen, “Expert level control of ramp metering based on multi-task deep reinforcement learning,” *CoRR*, vol. abs/1701.08832, 2017. [Online]. Available: <http://arxiv.org/abs/1701.08832>
- [11] X.-F. Xie, S. F. Smith, L. Lu, and G. J. Barlow, “Schedule-driven intersection control,” *Transportation Research Part C: Emerging Technologies*, vol. 24, pp. 168–189, 2012.
- [12] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte, “A high integrity imu/gps navigation loop for autonomous land vehicle applications,” *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 572–578, Jun 1999.
- [13] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, “A solution to the simultaneous localization and map building (slam) problem,” *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, Jun 2001.
- [14] Y. Cui and S. S. Ge, “Autonomous vehicle positioning with gps in urban canyon environments,” *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 15–25, Feb 2003.
- [15] Z. Shiller and Y. R. Gwo, “Dynamic motion planning of autonomous vehicles,” *IEEE Transactions on Robotics and Automation*, vol. 7, no. 2, pp. 241–249, Apr 1991.

- [16] S. D. Bopardikar, B. Englot, and A. Speranzon, "Multiobjective path planning: Localization constraints and collision probability," *IEEE Transactions on Robotics*, vol. 31, no. 3, pp. 562–577, June 2015.
- [17] J. Minguez and L. Montano, "Extending collision avoidance methods to consider the vehicle shape, kinematics, and dynamics of a mobile robot," *IEEE Transactions on Robotics*, vol. 25, no. 2, pp. 367–381, April 2009.
- [18] K. Kanatani and K. Watanabe, "Reconstruction of 3-d road geometry from images for autonomous land vehicles," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 1, pp. 127–132, Feb 1990.
- [19] B. Van Arem, C. J. Van Driel, and R. Visser, "The impact of cooperative adaptive cruise control on traffic-flow characteristics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 429–436, 2006.
- [20] J. Lee, J. Choi, K. Yi, M. Shin, and B. Ko, "Lane-keeping assistance control algorithm using differential braking to prevent unintended lane departures," *Control Engineering Practice*, vol. 23, pp. 1–13, 2014.
- [21] Y. S. Son, W. Kim, S.-H. Lee, and C. C. Chung, "Robust multirate control scheme with predictive virtual lanes for lane-keeping system of autonomous highway driving," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 8, pp. 3378–3391, 2015.
- [22] S. Lefevre, Y. Gao, D. Vasquez, H. E. Tseng, R. Bajcsy, and F. Borrelli, "Lane keeping assistance with learning-based driver model and model predictive control," in *12th International Symposium on Advanced Vehicle Control*, 2014.
- [23] G. Meyer and S. Shaheen, Eds., *Disrupting Mobility: Impacts of Sharing Economy and Innovative Transportation on Cities*. Springer, 2017.
- [24] D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for autonomous cars that leverage effects on human actions," in *Robotics: Science and Systems*, 2016.
- [25] J. Lee, M. Park, and H. Yeo, "A probability model for discretionary lane changes in highways," *KSCE Journal of Civil Engineering*, vol. 20, no. 7, pp. 2938–2946, 2016.
- [26] J. Rios-Torres and A. A. Malikopoulos, "A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1066–1077, 2017.
- [27] ——, "Automated and cooperative vehicle merging at highway on-ramps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 780–789, 2017.
- [28] 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.
- [29] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.
- [30] J. Schulman, S. Levine, P. Abbeel, M. I. Jordan, and P. Moritz, "Trust region policy optimization," in *ICML*, 2015, pp. 1889–1897.
- [31] 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*, 2015, pp. 2944–2952.
- [32] M. Lai, "Giraffe: Using deep reinforcement learning to play chess," *arXiv preprint arXiv:1509.01549*, 2015.
- [33] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res. (JAIR)*, vol. 47, pp. 253–279, 2013.
- [34] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik *et al.*, "Deepmind lab," *arXiv preprint arXiv:1612.03801*, 2016.
- [35] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.
- [36] G. Synnaeve, N. Nardelli, A. Auvolat, S. Chintala, T. Lacroix, Z. Lin, F. Richoux, and N. Usunier, "Torchcraft: a library for machine learning research on real-time strategy games," *arXiv preprint arXiv:1611.00625*, 2016.
- [37] 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*. IEEE, 2012, pp. 5026–5033.
- [38] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "Torcs, the open racing car simulator," *Software available at http://torcs.sourceforge.net*, 2000.
- [39] O. Vinyals, "Deepmind and blizzard to release starcraft ii as an ai research environment," <https://deepmind.com/blog/deepmind-and-blizzard-release-starcraft-ii-ai-research-environment/>, 2016.
- [40] M. Brackstone and M. McDonald, "Car-following: a historical review," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 2, no. 4, pp. 181–196, 1999.
- [41] Z. Zheng, "Recent developments and research needs in modeling lane changing," *Transportation research part B: methodological*, vol. 60, pp. 16–32, 2014.
- [42] A. Kotsialos, M. Papageorgiou, and A. Messmer, "Optimal coordinated and integrated motorway network traffic control," in *14th International Symposium on Transportation and Traffic Theory*, 1999.
- [43] 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, no. 3&4, 2012.
- [44] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," *CoRR*, vol. abs/1604.06778, 2016. [Online]. Available: <http://arxiv.org/abs/1604.06778>
- [45] R. Bellman, "A markovian decision process," DTIC Document, Tech. Rep., 1957.
- [46] R. A. Howard, "Dynamic programming and markov processes," 1960.
- [47] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour *et al.*, "Policy gradient methods for reinforcement learning with function approximation," in *NIPS*, vol. 99, 1999, pp. 1057–1063.
- [48] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.
- [49] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *International Conference on Machine Learning*, 2015, pp. 2067–2075.
- [50] P. G. Michalopoulos, D. E. Beskos, and Y. Yamauchi, "Multilane traffic flow dynamics: some macroscopic considerations," *Transportation Research Part B: Methodological*, vol. 18, no. 4, pp. 377–395, 1984.
- [51] A. Klar and R. Wegener, "A hierarchy of models for multilane vehicular traffic i: Modeling," *SIAM Journal on Applied Mathematics*, vol. 59, no. 3, pp. 983–1001, 1998.
- [52] A. Sasoh and T. Ohara, "Shock wave relation containing lane change source term for two-lane traffic flow," *Journal of the Physical Society of Japan*, vol. 71, no. 9, pp. 2339–2347, 2002.
- [53] C. F. Daganzo, "A behavioral theory of multi-lane traffic flow. part i: Long homogeneous freeway sections," *Transportation Research Part B: Methodological*, vol. 36, no. 2, pp. 131–158, 2002.
- [54] C. Wu, A. Kreidieh, E. Vinitsky, and A. Bayen, "Multi-lane reduction: A stochastic single-lane model for lane changing," in *Submission*, 2017.
- [55] M. Treiber and A. Kesting, "Traffic flow dynamics," *Traffic Flow Dynamics: Data, Models and Simulation*, Springer-Verlag Berlin Heidelberg, 2013.
- [56] ——, "The intelligent driver model with stochasticity-new insights into traffic flow oscillations," *Transportation Research Procedia*, vol. 23, pp. 174–187, 2017.
- [57] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, "Review of road traffic control strategies," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2043–2067, 2003.
- [58] P. A. Ioannou and C.-C. Chien, "Autonomous intelligent cruise control," *IEEE Trans. on Vehicular technology*, vol. 42, no. 4, pp. 657–672, 1993.
- [59] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE transactions on intelligent transportation systems*, vol. 4, no. 3, pp. 143–153, 2003.
- [60] Technical Committee ISO/TC 204, Intelligent transport systems, *Intelligent transport systems – Adaptive Cruise Control systems – Performance requirements and test procedures*, ISO ISO 15 622:2010, 2010.
- [61] E. W. Martin, K. Boriboonsomsin, N. D. Chan, N. Williams, S. A. Shaheen, and M. Barth, "Dynamic ecodriving in northern California: A study of survey and vehicle operations data from an ecodriving feedback device," in *92nd Annual Meeting of the Transportation Research Board, Washington, DC, January*, 2013.
- [62] C.-Y. Liang and P. Huei, "String stability analysis of adaptive cruise controlled vehicles," *JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing*, vol. 43, no. 3, pp. 671–677, 2000.
- [63] A. Bose and P. A. Ioannou, "Analysis of traffic flow with mixed manual and semiautomated vehicles," *IEEE Trans. on Intelligent Transportation Systems*, vol. 4, no. 4, pp. 173–188, 2003.
- [64] P. A. Ioannou and M. Stefanovic, "Evaluation of acc vehicles in mixed traffic: Lane change effects and sensitivity analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 1, pp. 79–89, 2005.
- [65] M. A. S. Kamal, J.-i. Imura, T. Hayakawa, A. Ohata, and K. Aihara, "Smart driving of a vehicle using model predictive control for improving

- traffic flow,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 878–888, 2014.
- [66] D. Swaroop, “String stability of interconnected systems: An application to platooning in automated highway systems,” *California Partners for Advanced Transit and Highways (PATH)*, 1997.
- [67] G. Orosz, R. E. Wilson, and G. Stépán, “Traffic jams: dynamics and control,” *Philosophical Trans. of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4455–4479, 2010.
- [68] G. Orosz, J. Moehlis, and F. Bullo, “Delayed car-following dynamics for human and robotic drivers,” in *ASME 2011 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*. American Society of Mechanical Engineers, 2011, pp. 529–538.
- [69] I. G. Jin and G. Orosz, “Dynamics of connected vehicle systems with delayed acceleration feedback,” *Transportation Research Part C: Emerging Technologies*, vol. 46, pp. 46–64, 2014.
- [70] B. K. Horn, “Suppressing traffic flow instabilities,” in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*. IEEE, 2013, pp. 13–20.
- [71] L. Wang, B. K. Horn, and G. Strang, “Eigenvalue and eigenvector analysis of stability for a line of traffic,” *Studies in Applied Mathematics*, 2016.
- [72] C. Wu, A. Bayen, and A. Mehta, “Stabilizing traffic with autonomous vehicles,” in *Submission*, 2017.
- [73] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang, “Traffic flow prediction with big data: a deep learning approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 865–873, 2015.
- [74] N. G. Polson and V. O. Sokolov, “Deep learning for short-term traffic flow prediction,” *Transportation Research Part C: Emerging Technologies*, vol. 79, pp. 1–17, 2017.
- [75] F. Belletti, D. Haziza, G. Gomes, and A. M. Bayen, “Expert level control of ramp metering based on multi-task deep reinforcement learning,” *CoRR*, vol. abs/1701.08832, 2017. [Online]. Available: <http://arxiv.org/abs/1701.08832>
- [76] M. G. Karlaftis and E. I. Vlahogianni, “Statistical methods versus neural networks in transportation research: Differences, similarities and some insights,” *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 3, pp. 387–399, 2011.
- [77] C. Wu, K. Parvate, N. Kheterpal, L. Dickstein, A. Mehta, E. Vinitsky, and A. Bayen, “Framework for control and deep reinforcement learning in traffic,” in *Submission*, 2017.
- [78] G. D. Cameron and G. I. Duncan, “Paramicsparallel microscopic simulation of road traffic,” *The Journal of Supercomputing*, vol. 10, no. 1, pp. 25–53, 1996.
- [79] M. Fellendorf, “Vissim: A microscopic simulation tool to evaluate actuated signal control including bus priority,” in *64th Institute of Transportation Engineers Annual Meeting*. Springer, 1994, pp. 1–9.
- [80] M. Fellendorf and P. Vortisch, “Microscopic traffic flow simulator vissim,” in *Fundamentals of traffic simulation*. Springer, 2010, pp. 63–93.
- [81] J. Casas, J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday, “Traffic simulation with aimsun,” in *Fundamentals of traffic simulation*. Springer, 2010, pp. 173–232.
- [82] K. N. Horni, A. and K. A. (eds.), *The Multi-Agent Transport Simulation MATSim*. Ubiquity, London, 2016.
- [83] J. Auld, M. Hope, H. Ley, V. Sokolov, B. Xu, and K. Zhang, “Polaris: Agent-based modeling framework development and implementation for integrated travel demand and network and operations simulations,” *Transportation Research Part C: Emerging Technologies*, vol. 64, pp. 101–116, 2016.
- [84] J. Erdmann. (2016) Simulation of urban mobility - wiki: Car-following-models. [Online]. Available: <http://sumo.dlr.de/wiki/Car-Following-Models#tau>
- [85] (2016) Simulation/basic definition. [Online]. Available: http://sumo.dlr.de/wiki/Simulation/Basic_Definition#Defining_the_Time_Step_Length
- [86] H. Dai, E. B. Khalil, Y. Zhang, B. Dilkina, and L. Song, “Learning combinatorial optimization algorithms over graphs,” *arXiv preprint arXiv:1704.01665*, 2017.
- [87] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” *CoRR*, vol. abs/1706.02275, 2017. [Online]. Available: <http://arxiv.org/abs/1706.02275>
- [88] 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.
- [89] M. Bando, K. Hasebe, A. Nakayama, A. Shibata, and Y. Sugiyama, “Structure stability of congestion in traffic dynamics,” *Japan Journal of Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 203–223, 1994.
- [90] ———, “Dynamical model of traffic congestion and numerical simulation,” *Physical review E*, vol. 51, no. 2, p. 1035, 1995.
- [91] M. Batista and E. Twrdy, “Optimal velocity functions for car-following models,” *Journal of Zhejiang University-SCIENCE A*, vol. 11, no. 7, pp. 520–529, 2010.
- [92] J. Erdmann, “SUMO’s lane-changing model,” in *Modeling Mobility with Open Data*. Springer, 2015, pp. 105–123.
- [93] R. Dowling, A. Skabardonis, and V. Alexiadis, “Traffic analysis toolbox volume iii: guidelines for applying traffic microsimulation modeling software,” Tech. Rep., 2004.
- [94] H. Yeo, A. Skabardonis, J. Halkias, J. Colyar, and V. Alexiadis, “Over-saturated freeway flow algorithm for use in next generation simulation,” *Transportation Research Record: Journal of the Transportation Research Board*, no. 2088, pp. 68–79, 2008.