Extracting Traffic Smoothing Controllers Directly From Driving Data using Offline RL

Thibaud Ardoin
Department of Computer Science
ENS Paris-Saclay
91190 Gif-sur-Yvette, France
thibaud.ardoin@ens-paris-saclay.fr

Eugene Vinitsky
UC Berkeley
Berkeley, CA 94704
vinitsky.eugene@gmail.com

Alexandre Bayen
UC Berkeley
Berkeley, CA 94704
bayen@berkeley.edu

Abstract

Recent work has demonstrated that autonomous vehicles (AVs) can be used to smooth emergent traffic shock-waves and improve the resultant energy efficiency of traffic. Building the controllers for these vehicles usually involves building a simulator as an intermediate step; this involves careful estimation of human driving dynamics and network calibration which can be challenging at scale. We investigate whether it is possible to learn optimal traffic controllers by using the abundance of available human driving data. We use Offline Reinforcement Learning to extract controllers from "human" driving data collected from simulator runs. In contrast to expectations that data from an expert or random policy is needed, we demonstrate that assuming slightly noisy models of human driving is sufficient to generate data that covers the state space enough to extract effective controllers. We compare with a hand-designed linear controller and show that our controller significantly outperforms it. This suggests the possibility of directly extracting traffic smoothing policies from abundant driver data collected from deployed AVs.

1 Introduction

Due to fundamental limitations in reaction time and sensing capabilities, human driving is inefficient and below theoretical upper limits on throughput and energy efficiency of highway traffic. In particular, [13] experimentally demonstrated that even in the absence of perturbations, instabilities in human driving lead to the formation of shock-waves that both slow overall traffic and consume a significant amount of energy. In a seminal work, [12] showed that dissipating these waves did not require a significant number of vehicles (they used one AV amongst 21 human drivers) and showed in a field demonstration that a simple PID controller could be used to totally eliminate the waves. The low penetration rate suggested that significant benefits from autonomy could be achieved at $\approx 5\%$ penetration rates of level-2 vehicles (autonomous distance keeping and lane changing). This penetration rate is low enough that it is likely to be achieved in the next few years if it is not already achieved.

However, the results in [12] are in relatively simple settings: all the vehicles are driving around a circular track. Scaling energy efficient, traffic-smoothing control to the full complexity of highway traffic, with its multi-agent, hybrid dynamics, is an open challenge. Generally, two key approaches to

Machine Learning for Autonomous Driving Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

tackling the problem of scale are either (1) designing controllers in simplified settings with theoretical guarantees and then testing in the simulation if the properties hold in complex networks as in [17] or (2) using optimization based approaches such as Reinforcement Learning to directly design controllers in the full complexity scenario. Both of these scenarios require accurate human driving models as well as a calibrated simulation to test in. However, the process of building these simulations and calibrating them to data can take months of effort and at large scale these simulations can often be too slow to efficiently sample from for optimization.

As an alternative, we can use Offline Reinforcement Learning to extract a traffic smoothing controller directly from data without needing to build or run a simulation as an intermediary step. There is now a wide variety of camera data of human driving available, both overhead camera data [1, 8] and dash-cam/radar/LIDAR data [16, 2, 14]. This abundance of data has sufficient information to relabel with energy consumption as a reward and use to perform Offline RL. Since the data comes directly from human drivers rather than potentially miscalibrated simulators, it is possible that the resultant controllers would be significantly likelier to transfer to real highways.

However, Offline RL often performs best when the data is collected from trajectories provided by a controller that guarantees sufficient coverage of the state and action space. In prior work when examining Offline RL for traffic smoothing control, [3] acquired data by having one of the vehicles take either random actions or actions drawn from an expert policy trained via RL. Without sufficient coverage provided by either a random or expert policy, the output controller can be overoptimistic and has ill-defined behavior outside of the data distribution. While human data might contain examples of humans generating waves, it is unlikely that it contains examples of humans actively smoothing the waves. While there may be small sequences of steps where a human driver partially smooths a wave by coincidence, this data is likely infrequent and hence it is unclear if traffic smoothing behavior can be extracted from human data alone. As a first step at determining whether traffic smoothing controllers can be extracted from human data given these limitations, we examine whether they can be extracted from simulated human data. By controlling the level of noise in the simulated driving data, we can establish lower bounds on how much variability would be needed in a human driving data-set.

Our contributions are as follows:

- We demonstrate that at extremely low levels of noise we can extract wave smoothing controllers.
- We investigate how the level of noise affects the training of the resultant controller.
- We demonstrate that the extracted controller outperforms baseline hand-designed controllers.

Sec.2 Provides background on the offline RL method used, our simulator, and control of traffic smoothing waves. Sec.3 describes our MDP and the results of our experiments. Sec.4 offers some conclusions and ideas for how offline RL can be used to scale traffic smoothing control.

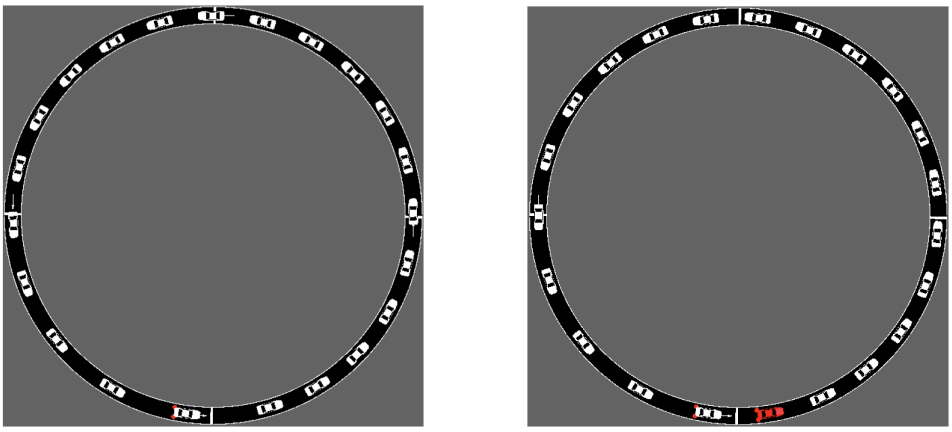
2 Background

2.1 Wave Formation and Control

In this work we use Flow [15], an interface that connects RL libraries to traffic micro-simulators (in this case SUMO [10]), to study the scenario shown in Fig. 1 where 22 vehicles are driving around a circular track. This is a simulated recreation of both the experiment that demonstrated the spontaneous formation of waves [13] and the experiment demonstrating that AVs could be used to smooth traffic [12]. 22 vehicles are placed on the ring and drive according to the Intelligent Driver Model [7], a popular model of human driving behavior in highway settings. In the controlled setting, we will replace one of these vehicles with an autonomous vehicle and see if we can remove the spontaneously forming waves that occur in the absence of control. For a given vehicle, the model is a function of the vehicle's speed, the distance to its leader vehicle, and the speed of the leader vehicle. To introduce stochasticity we also add a small amount of Gaussian noise so that the dynamics evolve as

$$x_{t+1} = x_t + v\Delta t \tag{1}$$

$$v_{t+1} = v_t + \left(f_{\text{IDM}}(v_t, v_t^{\text{lead}}, x_{t+1}^{\text{lead}} - x_{t+1}) + \sqrt{\Delta t}\, \mathcal{N}(0, \sigma) \right) \Delta t \tag{2}$$



(a) Formation of traffic wave with 22 human drivers.

(b) A single AV (red) inserted into the ring to smooth the wave.

Figure 1

where f_{IDM} is the driver model, x and v are position variables, and Δt is the simulator time-step. We are using first-order Euler integration to evolve the dynamics.

In the absence of noise the system is stable, but in the presence of small amounts of noise waves gradually begin to form as can be seen in Fig. 2 where the system starts in a steady state but gradually begins to evolve waves (the slanted red curves). Prior work has focused on removing these waves

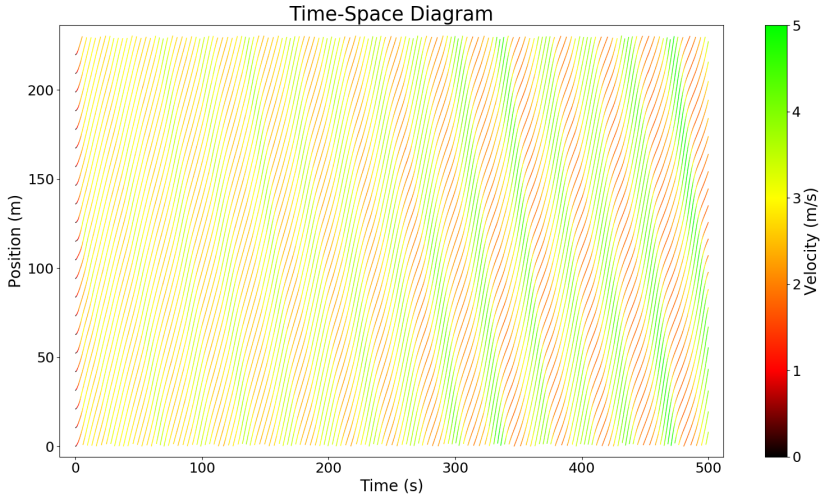


Figure 2: Formation of waves as 22 human drivers drive on the ring. Each line is the trajectory of a human driver and the red portions correspond to portions where vehicles have come to a near full stop. The alternating patterns of red and green indicate the formation of waves.

by using linear controllers [12, 17] or RL [15]. As our baseline, we will use the Follower-Stopper controller from [12], which uses feedback to regulate the speed of the controlled vehicle around a desired velocity that is pre-specified. As a basic intuition for how it works, by traveling at a slightly lower speed the controller opens up a gap to the vehicle in front of it which gives it time to brake smoothly as a wave approaches. The follower-stopper computes its acceleration using the following

equation:

$$v_{\text{cmd}} = \begin{cases} 0 & \text{if } \Delta x \leq \Delta x_1 \\ v \frac{\Delta x - \Delta x_1}{\Delta x_2 - \Delta x_1} & \text{if } \Delta x_1 < \Delta x \leq \Delta x_2 \\ v + (v_{\text{des}} - v) \frac{\Delta x - \Delta x_3}{\Delta x_3 - \Delta x_2} & \text{if } \Delta x_2 < \Delta x \leq \Delta x_3 \\ v_{\text{des}} & \text{if } x_3 < \Delta x \end{cases} \tag{3}$$

$$u = \min\left(\max\left(u_{\text{min}}, \frac{v_{\text{cmd}} - v}{\Delta t}\right), u_{\text{max}}\right) \tag{4}$$

where Δx is the distance to the lead vehicle, Δx_1, Δx_2, Δx_3 are specified values dividing the distance into an unsafe region, a region of linear control, a safe region where the maximum speed is achievable, and u is the commanded acceleration. For more details see [12] . We note that since this controller is designed to ensure stability rather than optimize a given reward function, it is not the optimal controller for the reward function we use in Sec. 3.1, however, we will perform a hyperparameter sweep of v_{des} to tune it as much as possible.

2.2 Offline Reinforcement Learning

Offline Reinforcement learning is essentially off-policy RL but with the constraint of a static data-set $\mathcal{D}$. This data is a set of transition (s_t, a_t, s_{t+1}, r_t) gathered using a certain *behavior* policy π_b that can technically be from any source but usually is a random or expert policy. The particular variant of RL we will use here, BEAR [9], is an actor-critic algorithm built upon TD3 [4] / SAC [6] that seek to learn a critic $Q(s, a)$ and a policy π_θ that maps the state into an action that maximizes the critic values. The key insight of BEAR is that actions that maximize the estimated Q function may be well outside the support of the training distribution. In turn, this causes overconfident estimates of the value of actions that are not even contained in the dataset.

BEAR resolves this challenge by constraining the policy to match the set of policies that could plausibly have generated the dataset. It does so using a sampled version of Maximum Mean Discrepancy [5] with a Gaussian kernel k (Eq. 2.2)

$$\text{MMD}^2\left(\{x_1, \ldots, x_n\}, \{y_1, \ldots, y_m\}\right) = \frac{1}{n^2} \sum_{i,i'} k\left(x_i, x_i'\right) - \frac{2}{nm} \sum_{i,j} k\left(x_i, y_j\right) + \frac{1}{m^2} \sum_{j,j'} k\left(y_j, y_j'\right)$$

It uses the estimated value of this discrepancy between the policy and the data as a constraint and updates the policy by maximizing a set of K Q functions; these K Q functions are used to induce conservativeness via ensembling. The resultant optimization problem is

$$\pi_\theta = \mathbb{E}_{s \sim \mathcal{D}} \mathbb{E}_{a \sim \pi(\cdot|s)} \left[\min_{j=1,\ldots,K} Q_j(s, a)\right] \text{ s.t. } \mathbb{E}_{s \sim \mathcal{D}} \left[\text{MMD}^2\left(\mathcal{D}(s), \pi(\cdot|s)\right)\right] \leq \epsilon \tag{5}$$

This optimization problem is then tackling using dual descent and the Q function updated using the standard approximation to Bellman's equations. For more details please see [9].

3 Experiments and Results

Here we briefly define the Partially Observed Markov Decision Process (POMDP) for our problem and the reward function of use. We then provide our results for the performance of the trained controllers and discuss the interaction between the controller optimality and the level of noise in the data collection process.

3.1 Defining the Markov Decision Process

The key ingredient in our definition of the MDP is a reward function that aligns with notions of wave smoothing. Since the speed of the system is maximized when the waves are gone, we simply use the average speed of the ring as the reward function base. Denoting s_t as the state at time t, u_t as the action at time t, r_t as the reward at time t, and v_t^i as the speed of vehicle i at time t, our reward

function is:

$$r(s_t, u_t) = 4 * \left(\frac{1}{N} \sum_{i=1}^{22} v_t^i - |u_t| \right) \tag{6}$$

where we have added an absolute penalty on the actions as a regularizer; this reward is taken from [15]. Note that we are using the average speed of all vehicles in the network as the reward function; this mean value would likely not be accessible in camera data. However, since the waves are periodic, the mean value simply leads to a smoothed speed trajectory due to the symmetry of the ring and could be replaced with the speed of only the controlled vehicles without any likely impact on the results. While this is an assumption, we leave a sparser reward function to future work. The remaining components of the MDP are as follows:

- The state space is $s_t = \left[v_t, v_t^{\text{lead}}, x_t^{\text{lead}} - x_t\right]$ where v_t is the vehicle speed, v_t^{lead} is the leader speed, and the third term is the distance to the lead vehicle.
- Transition dynamics: all the human vehicles are governed by Eq. 2 whereas the AV evolves as a double integrator.
- Action space: the actions are accelerations bounded between $[-1, 1]$.
- Discount factor: $\gamma = 0.99$.

3.2 Experimental protocol

We generate datasets of one million input transitions each of which is sampled from our simulated human drivers with different values of σ i.e. Gaussian noise with varying std. deviation. Here we are assuming that the different noise values approximately represent different amounts of variation in human driving patterns. We run each simulation and collect 1000 steps per simulation run with a simulation step $\Delta t = 0.5$. As an additional source of variability, at each rollout we uniformly sample a ring length from $[220, 270]$ meters, this varies the density of the vehicles on the ring which in turn changes the optimal speed for the ring. This is viewed as drawing samples from different levels of congestion.

For each of these datasets, we run BEAR using the hyperparameters in Table. 5a. During training we periodically evaluate the controller on a ring of size 230 meters to compute the expected reward of our controllers; noise is kept on at the same level as it is during training. Note that this evaluation procedure would technically not be available given a real world data source without constructing a simulation to evaluate on. We leave the question of how to select a good controller without an evaluation environment for future work.

3.3 Experimental results

In figure 3 we plot the mean evaluation reward over 5 training runs with noise values in $\sigma \in \{0.05, 0.1, 0.2, 0.5, 1\}$. On the plot we also present (black dotted line) the maximum reward of the follower-stopper described in Sec. 2.1 with $v_{\text{des}} = 3$. While the equilibrium speed on the ring without noise is $\approx 5 \frac{\text{meters}}{\text{second}}$, we empirically determined via grid search that $v_{\text{des}} = 3$ gave the optimal reward for the follower-stopper controller for our reward function. Note that this value is computed in the absence of noise and is essentially an upper-bound on follower-stopper performance. The follower-stopper is not designed to optimize the reward function so it is not expected to achieve great performance, but it does serve as a lower bound that we can compare against.

The reward evolution shows that data generated by IDM with $\sigma = 0.05$ is not capable of learning; this is consistent with the results of [3] which found that data extracted purely from human driving models without any noise did not manage to learn. As the noise increases to $\sigma = 0.1$ we start to get an increasing reward and our reward is maximized at values of $\sigma = 0.5$. Note that if the noise is too high or too low ($\sigma = 0.1$ and $\sigma = 1.0$ respectively) the reward curve starts to go unstable above some epoch. We see that $\sigma = 0.1$ is approximately the level at which getting a reasonable controller becomes possible. This gives us an estimate of how noisy a driving data set would need to be to extract traffic smoothing controllers.

We visualize the system dynamics under our respective controllers in Fig. 4. On the left, we see the formation of waves without control, and on the right the smoothing of waves when control is applied. Excitingly, although this behavior is never observed in the training data when control is turned on the

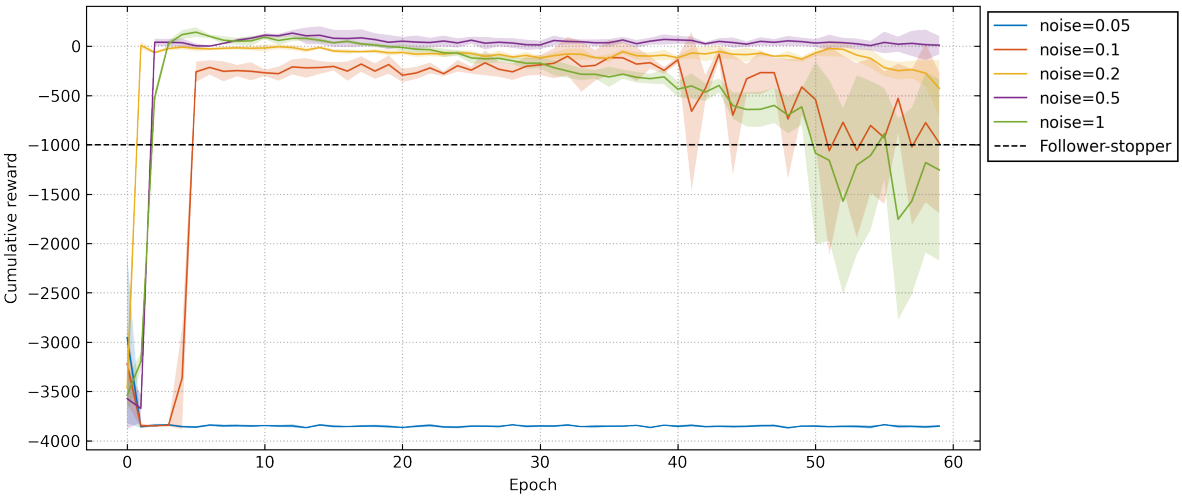
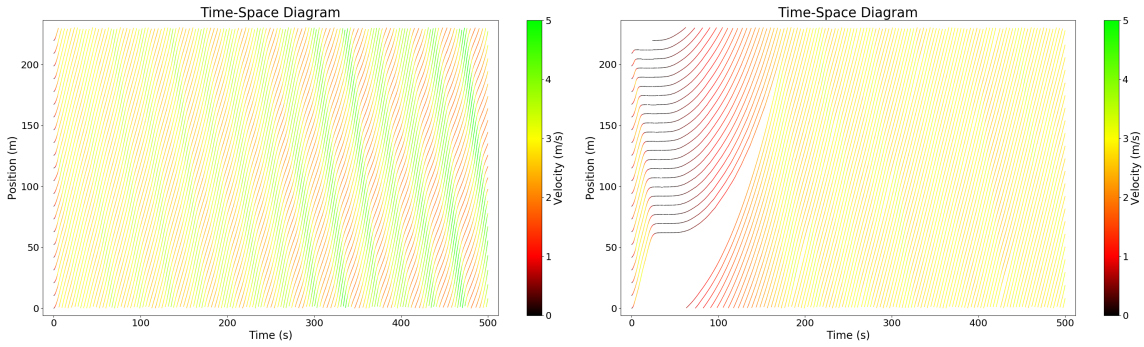


Figure 3: Evolution of the mean reward across 4 runs for each noise value. The dotted black line represents the reward achieved by the Follower Stopper with $v_{\text{des}} = 3.0$.

vehicle comes to a full stop and gradually accelerates to bring all the vehicles up to a stable speed. Waves are sharply reduced with control on but cannot be completely eliminated due to the significant amount of noise in the system.



(a) $\sigma = 0.3$, Human Driven

(b) $\sigma = 0.3$, Autonomous

Figure 4: Time-space diagram of the system dynamics with a single controlled AV. On the left, the system is unstable without control while the waves are significantly smoothed with control on the right. Waves cannot be completely reduced due to the inherent noise in the system.

4 Discussion and Future Work

In this work, we provide an affirmative answer to the question of whether traffic smoothing controllers can be extracted from human data without a controlled vehicle to generate the data. While we perform our tests entirely in simulation, the low level of noise above which we can extract policies is a promising sign for our ability to extract similar controllers from human driving data. Attempting to do this using publicly available datasets is a direction that would be interesting to pursue.

A key challenge that we were not able to resolve in this work is how to evaluate our controllers without using a simulation. Since the goal is to extract a controller without building a simulation in between, it is necessary to evaluate the controller using only the dataset. In future work, we would like to investigate using train-validation-test splits as is often done in supervised learning and see if this is sufficient for picking good controllers.

Furthermore, to scale up to complex networks with many controlled agents, it is necessary to extend these offline RL techniques to the multi-agent setting. Perhaps at low penetration rates of autonomous

vehicles, each driving agent is sufficiently isolated from the other agents that it can be treated effectively as a single agent problem. However, this is unlikely to work for all driving all problems and it will likely be necessary to treat these as multi-agent problems. In this vein, it would be interesting to examine whether offline RL works in the "train centralized, act decentralized" paradigm used to construct multiagent algorithms such as MADDPG [11].

Acknowledgments and Disclosure of Funding

The authors would like to thank Justin Fu for his invaluable discussions around offline RL algorithms as well as the release of the D4RL library. Eugene Vinitsky is a recipient of an NSF Graduate Research Fellowship and funded by the National Science Foundation under Grant Number CNS-1837244. Computational resources for this work were provided by an AWS Machine Learning Research grant. This material is also based upon work supported by the U.S. Department of Energy's Office of Energy Efficiency and Renewable Energy (EERE) award number CID DE-EE0008872. The views expressed herein do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

References

[1] V. Alexiadis, J. Colyar, J. Halkias, R. Hranac, and G. McHale. The next generation simulation program. *Institute of Transportation Engineers. ITE Journal*, 74(8):22, 2004.

[2] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11621–11631, 2020.

[3] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *ArXiv*, abs/2004.07219, 2020.

[4] S. Fujimoto, H. Van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[5] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[6] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[7] A. Kesting, M. Treiber, and D. Helbing. Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4585–4605, 2010.

[8] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein. The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2118–2125. IEEE, 2018.

[9] A. Kumar, J. Fu, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. *CoRR*, abs/1906.00949, 2019.

[10] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[11] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pages 6379–6390, 2017.

[12] R. E. Stern, S. Cui, M. L. Delle 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. *Transportation Research Part C: Emerging Technologies*, 89:205–221, 2018.

[13] 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*, 10(3):033001, 2008.

[14] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

[15] 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*, page 10, 2017.

[16] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019.

[17] Y. Zheng, J. Wang, and K. Li. Smoothing traffic flow via control of autonomous vehicles. *IEEE Internet of Things Journal*, 7(5):3882–3896, 2020.

A Hyperparameters

num eval steps per epoch	1000
num trains per train loop	1000
num expl steps per train loop	1000
min num steps before training	1000
max path length	1000
batch size	256
layer size	256
replay buffer size	2e6
discount	0.99
kernel choice	"gaussian"
mmd sigma	50
policy lr	1e-4
qf lr	3e-4
policy update style	"0"
reward scale	1
soft target tau	0.005
target mmd thresh	0.05
target update period	1

(a) Hyper-parameters used for the BEAR algorithm training in Fig. 3