Flow: Open Source Reinforcement Learning for Traffic Control

Nishant Kheterpal*
EECS Department
University of California, Berkeley

Eugene Vinitsky
ME Department
University of California, Berkeley

Cathy Wu
EECS Department
University of California, Berkeley

Aboudy Kreidieh
CEE Department
University of California, Berkeley

Kathy Jang
CS Department
University of California, Berkeley

Kanaad Parvate
EECS Department
University of California, Berkeley

Alexandre Bayen
ITS and CEE Department
University of California, Berkeley

Abstract

This work presents Flow, an open-source Python library enabling the application of distributed reinforcement learning (RL) to *mixed-autonomy* traffic control tasks, in which autonomous vehicles, human-driven vehicles, and infrastructure interact. Flow integrates SUMO, a traffic microsimulator, with RLlib, a distributed reinforcement learning library. Using Flow, researchers can programatically design new traffic networks, specify experiment configurations, and apply control to autonomous vehicles and intelligent infrastructure. We have used Flow to train autonomous vehicles to improve traffic flow in a wide variety of representative traffic scenarios; the results and scripts to generate the networks and perform training have been integrated into Flow as benchmarks. Community use of Flow is central to its design; extensibility and clarity were considered throughout development. Flow is available for open-source use at `flow-project.github.io` and `github.com/flow-project/flow`.

1 Introduction

We describe the design and motivation behind Flow, an open-source library linking deep reinforcement libraries rllab and RLlib with SUMO, a microscopic (vehicle-level) traffic simulator. This work will describe the design of Flow, the way it links existing open-source libraries and tools, and our focus on creating a valuable tool for the community. Our previous publications have discussed different aspects of Flow in greater detail [1] [2].

Ease of use for the general community guided the development of Flow, particularly our recent work. Much of our work in recent months has centered around improving the usability and scalability of Flow. Flow includes publicly-available Dockerfiles for running Flow experiments on Amazon

*email: `nskh@berkeley.edu`

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

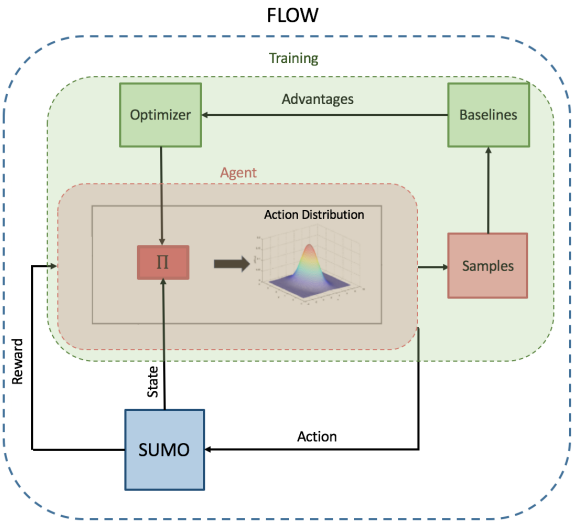

Figure 1: Process diagram describing the reinforcement learning training process and interactions between SUMO, Flow, and the libraries used to train.

EC2 and for containerizing an instance of Flow for users to try locally. All of Flow's development is public through our GitHub repository at `github.com/flow-project/flow`; we use GitHub's included project management tools to log issues, review pull requests from contributors, and prioritize development tasks. Any users are free to create issues and contribute to Flow. The Flow Google Group is another resource for users to ask questions of the community. Users may familiarize themselves with Flow using a series of tutorials in Jupyter Notebook format as well as our documentation on ReadTheDocs.

2 Preview of Demonstration

Our demonstration will highlight the variety of tasks that Flow facilitates. We will show scenario and road network generation in Flow, various methods of vehicle-level control, and representative results on key traffic control benchmarks.

2.1 Scenario Generation

Pythonic scenario generation is a core feature of Flow. Road networks such as multi-lane ring roads, figure-eight intersection networks, grid networks with traffic lights, and merge scenarios are programatically generated by specifying the shape and properties of their corresponding directed graph, enabling easy modification of existing networks. Some networks in Flow are displayed in Figure 2. Map data import from OpenStreetMap, a public collaborative mapping project, is supported out of the box as well. Scenarios can be specified using XML in Python generator files if users wish to create a new type of programatically generated road network.

Vehicle properties, traffic light behavior, network inflows, and more can be specified in code, using Python files specifying an experiment. Experiments in Flow include the number and type of vehicles, road scenario, network, RL algorithm used, and any relevant parameters.

2.2 Control

Flow supports pre-specified control approaches as well as control using reinforcement learning. Popular longitudinal vehicle controllers regulating speed and acceleration like IDM are included in Flow, among others [3]. Such controllers can be used to model human driving for evaluation in performance baselines or simulating human behavior in mixed-autonomy settings. Lateral behavior for vehicles can be specified as well; users can use SUMO's built-in lane-changing models or disable lane-changing entirely. Custom controllers can be implemented in a straightforward fashion as well,

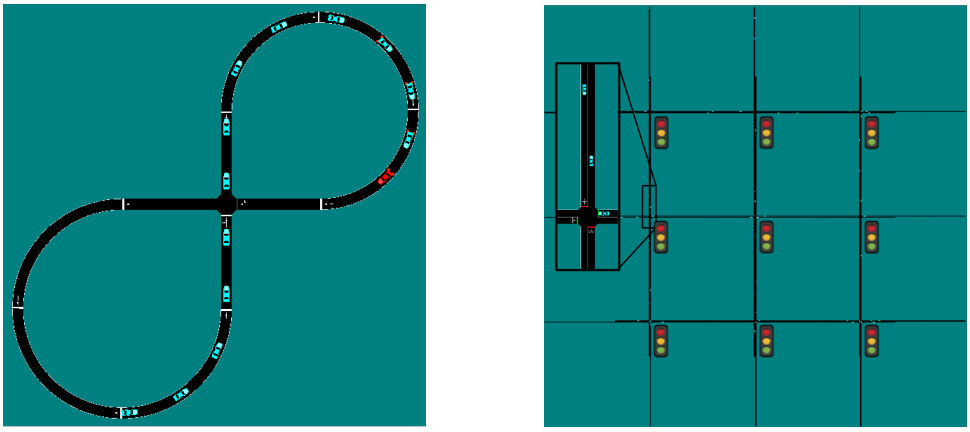

Figure 2: One scenario in Flow is the figure-eight (at left), which models intersection dynamics in a closed network. The length of the figure-eight network and number of vehicles can be modified algorithmically to study the effect of density on average velocity, for example. The grid network (at right) is a variable-size open network with traffic lights at each intersection that can be controlled by RL or pre-specified approaches.

for users of Flow who seek to test or develop traffic control methods independent of reinforcement learning.

Training agents to act in environments defined by traffic scenarios in Flow using RL is a core focus of the library. Flow includes an `RLController` class used by vehicles that act according to the outputs of the policy being trained. The demonstration will show the process of specifying and then training an experiment consisting of both RL-controlled vehicles and vehicles controlled by human driver models. Policies trained using reinforcement learning can be replayed and benchmarked to analyze the effectiveness of RL control compared to baseline settings. A video of one such trained policy can be viewed on YouTube, in which one autonomous vehicle stabilizes 21 other vehicles on a single-lane ring road, as studied by Sugiyama et al. [4] and Stern et al. [5].

2.3 Benchmarks

Our recent work released benchmarks for the deep reinforcement learning community to create controllers for mixed-autonomy settings [2]. Benchmarks are important in the RL community because they allow researchers to measure algorithm performance and make empirical comparisons between RL algorithms. MuJoCo [6] and the Arcade Learning Environment [7] are existing benchmarks for RL tools frequently used to evaluate the performance of reinforcement learning methods [8] [9] [10] [11]. Flow includes four benchmarks representing distinct traffic control tasks to encourage progress in the community of traffic control using reinforcement learning [2]. Users of Flow can test new RL approaches on these benchmarks and compare their performance in key traffic-related metrics to the highest-performing solutions thus far. This performance scoreboard will be publicly available.

3 Conclusion

Flow is a Python library linking deep reinforcement learning libraries with traffic microsimulation methods. Users of Flow could use the entire library as the core of a workflow that includes the design of traffic networks, specification of Markov decision processes, and an iterative process to train traffic-mitigating agents. They might also use components of Flow to aid in answering other questions, such as a rapid benchmarking tool for RL algorithms, a testbed for classical control techniques, or others. Future considerations around the development of Flow center around how to make use of the library as straightforward as possible for outside users. We look forward to discussions at the MLOSS workshop regarding community building, best practices for development, and guiding principles.

References

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

[2] E. Vinitsky, A. Kreidieh, L. Le Flem, N. Kheterpal, K. Jang, F. Wu, R. Liaw, E. Liang, and A. M. Bayen, "Benchmarks for reinforcement learning in mixed-autonomy traffic," in *Conference on Robot Learning*, 2018.

[3] M. Treiber and A. Kesting, "The intelligent driver model with stochasticity-new insights into traffic flow oscillations," *Transportation Research Procedia*, vol. 23, pp. 174–187, 2017.

[4] 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.

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

[6] 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.

[7] 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.

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

[9] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[10] H. Mania, A. Guy, and B. Recht, "Simple random search provides a competitive approach to reinforcement learning," *arXiv preprint arXiv:1803.07055*, 2018.

[11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.