Exhaust-ive Learning: Deep Reinforcement Learning for Energy Reduction on Highways

Leah Dickstein
Electrical Engineering and Computer Science
University of California, Berkeley
Email: leahdickstein@berkeley.edu

***Abstract*—The energy currently consumed by highways is enough to power 21 million homes in the United States. Within the next decade autonomous vehicles will be on our highways, transporting people and goods and navigating among human-powered vehicles. We investigated whether a set of autonomous vehicles with a central controller could beneficially guide traffic, reducing the systems overall energy consumption. After solving a toy control problem, we experimented with various reinforcement learning problem formulations before finding one that represented our problem and provided meaningful results. With our current formulation the autonomous cars drive as fast and steadily as possible, but dont have opportunities to guide the human-powered vehicles. Moving forward, we will implement safety constraints in the action space and try congested traffic environments to explore the optimal policy in different scenarios.**

I. Introduction

Transportation comprises about 28% of US annual energy consumption, with highways representing 21% of annual energy consumption. The energy currently consumed by highways is enough to power 21 million homes in the United States. As energy is a major problem in our global future, our goal is to reduce energy consumption/emissions in a way that can be realized on our roads soon. In the next few decades, experts have projected that we will have a *partial* penetration of autonomous vehicles on the road. Our project explores the opportunity of using these autonomous vehicles as control inputs for the traffic system to reduce system-level energy consumption.

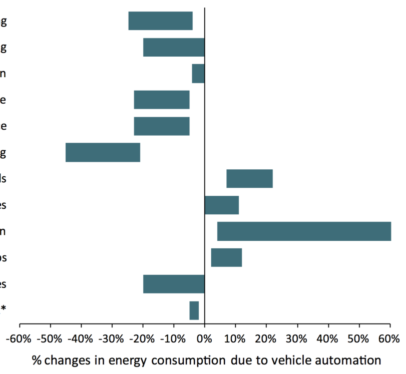


Fig 1. Wadud et al. [1] projected what would happen to energy consumption with full automation of highways. There has been no work done, however, on partial penetration.

In particular, congestion mitigation could save up to 5% of overall fuel consumption, and improved crash avoidance could save up to 20% of fuel consumption. We wish to validate these numbers and explore if we can achieve even greater fuel reduction using optimized policies (instead of fuel reduction being a side benefit of highway automation).

Related work: Francois Belletti, another graduate student in Professor Bayens lab, used reinforcement learning with a convolutional net to learn a policy for traffic metering [2]. The state space was discretized locations on a highway, and the policy learned represented number of cars to allow onto the highway via on-ramps (and leaving via off-ramps). They used a novel weight-sharing algorithm called Mutual Weight Regularization, which we think will be useful for our multi-agent control scheme (where potentially our multiple agents can share information). Stevens and Yeh [3] used reinforcement learning on traffic lights to increase traffic flow through intersections. Polson and Sokolov [4] used a deep learning architecture with dropout and hierarchical sparse vector auto-regressive techniques to predict traffic flows, demonstrating success even during highly nonlinear special events. Lv et al. [5] used hierarchical autoencoding as well, so that they can learn features to represent states involving both space and time. Since our inputs will be spatio-temporal, we will investigate these feature encoding techniques. Karlaftis and Vlahogianni [6] present an overview comparing non-neural statistical methods versus neural networks in transportation research.

II. Problem Statement

We wish to learn control policies for $|K|$ autonomous vehicles (where $|K|$ is one of many hyperparameters) and a central controller with the objective of minimizing vehicular energy consumption. For example, these vehicles may mitigate traffic jams and congestion by promoting a constant flowing of cars on highways and minimizing travel time and acceleration/deceleration. We initially study the setting where autonomous vehicles are empty (for example, they are employed by the government and their sole purpose is to follow the policy to decrease energy consumption).

How can a small team of autonomous vehicles in a complex mixed-autonomy traffic network setting minimize vehicular energy consumption of the entire system?

We pose the problem within the reinforcement learning framework:

- State: $(v,x,y,f,d)_{t,k}$ sequence/trajectory of position, velocity, acceleration all vehicles.
- Policy: a function $f : (v,x,y,f,d)_{t,k} \rightarrow (a_i,z_i)_{i\in K}$, which maps the state to control actions for each autonomous vehicle (acceleration, lateral change).
- Note that k represents all vehicles while K represents the set of autonomous vehicles
- Reward function: $-\sum_k 0.9 \cdot distance\_to\_target + 0.1 \cdot fuel\_consumption = -\sum_k 0.9 \cdot (3.36km - d_k) + 0.1 f_k$
- Example controls: Steering (e.g. lane changes), deceleration/braking, and acceleration/throttling.
- Success measure: Average return, which corresponds to total fuel consumption over the rollout or a metric that captures mpg for each car

We believe this is the first investigation of partial penetration of autonomous vehicles, and as such there are many interesting questions to explore. Some example questions we are interested in are: Can a few autonomous vehicles reduce overall energy consumption of the traffic network? Can they stabilize traffic? How many autonomous vehicles are needed, and what variables affect the number of autonomous vehicles needed? Is it possible to reduce energy while decreasing travel time, or is it only possible if everyone slows down? How much information do the autonomous vehicles need? How much communication between cars or with the infrastructure is needed? Is it possible to guide traffic with a decentralized algorithm? Do we learn new behaviors or do the reinforcement learning methods learn engineering-designed behaviors (such as platooning or Adaptive Cruise Control)? How will humans react to the self-driving cars and can we model the behaviors using generative adversarial networks (GANs)? How well do learned policies transfer between different highway environments?

III. Approach

A. Data

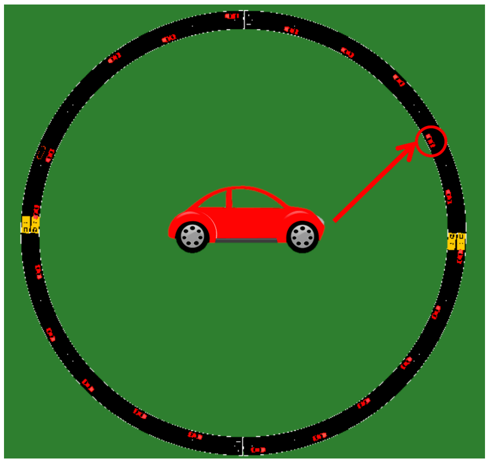


Fig 2. An example circular highway with 22 vehicles.

This project used the open-source microscopic traffic simulator SUMO [7]. SUMO outputs precise information about each vehicles position over time, emissions, lane changing events, etc. Since the solution is only as good/realistic as the simulator is realistic, we will (separate from this project) modify SUMO to make sure it matches with known real-world highway behaviors and output accurate energy emission values.

Other team members are working on an interface so that we can experiment with more fine-grained traffic dynamics and compare the deep reinforcement learning method with the control theoretic baseline. This interface that we are developing will use smaller time-steps to update the controls (for more precise results) and use a more realistic lane-changing model.

I wrote Python scripts to generate XML files that specify the circular highway environment we are using for the SUMO simulation. I first had 22 cars on a smaller 0.5km length highway, and I eventually settled on 12 cars on a 0.84km highway as the best initial test environment. I also wrote a Python script so that the autonomous vehicles are uniformly distributed among the human-powered vehicles on the highway.

B. Baseline Model

For our baseline model we solved a toy control problem: bring all cars velocity from 20m/s to 25m/s. The reward function was the mean-squared error from the goal velocity, summed over all the autonomous vehicles. This posing of the Markov Decision Process (MDP) meant there was no difference between partial and full observability. For the neural net architecture I used a single-layer 16-neuron neural net because I noticed the more standard 2-layer 32-neuron neural net was overfitting. I used the TRPO algorithm [8], which was developed at Berkeley, and 5 seeds because TRPO has randomness. The state was just the velocity of the autonomous vehicles and the action space was the acceleration we could give each car.

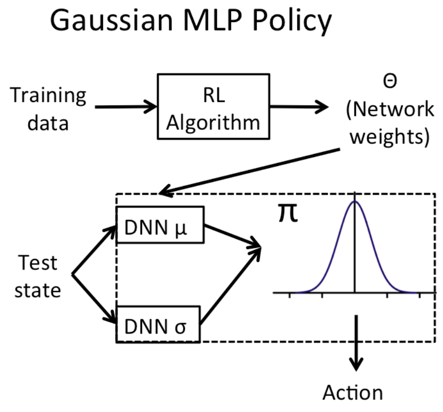


Fig 3. Graphic representation of a Gaussian MLP Policy

I chose to learn a Gaussian MLP Policy which works as follows: The training data of each iteration is composed of several rollouts, or sequences of states and actions. At the end of each iteration, I use a reinforcement learning algorithm (such as TRPO) to update θ, or the network weights. θ corresponds to two DNNs, one that represents the mean and one that represents standard deviation. Then when I input a test state it is run through the two separate DNNs to generate a mean and standard deviation, which is used to create a multivariate Gaussian over the action space. I sample from the Gaussian to get the action that will be applied next.

C. Final Model

One of the challenges of reinforcement learning is coercing the MDP to be both solvable (an optimum exists) and meaningful. We originally planned on having a state $(x, y, v, a)_{t,k}$, which would represent 2D position, velocity, and acceleration. This didnt provide enough useful information and we had both high variance and convergence to a suboptimal result. After trying different states, I eventually settled on $(v, x, y, f, d)_{t,k}$ as my state. This removes acceleration, which was redundant given velocity, and adds fuel consumption in the last timestep and distance travelled so far. Since the reward function is based on fuel consumption and distance travelled, that information needs to be encoded in the state. If fuel and distance arent included, the problem is non-injective and the results will have extremely high variance. With the new, larger state each (v, x, y, f, d) is a unique state with the correct deterministic reward, the problem becomes injective and the algorithm can effectively learn the connection between the state and reward function.

State	No. of Vehicles	Dimensionality
(v)	2	2
(v)	12	12
(x, y, v, a)	2	8
(x, y, v, a)	12	48
(v, x, y, f, d)	2	10
(v, x, y, f, d)	12	60
(v, x, y, f, d)	20	100

This table shows some of our experiments and the corresponding changes in problem dimensionality. For example, if the state is (v, x, y, f, d) and there are 12 autonomous vehicles on the highway, the observation will be the flattened vector of all 12 vehicles' states (dimension 60). Relative to the toy problem, the state grew from 2 dimensions to 60 dimensions. Since the problem was now more complicated, we returned the neural net architecture to the more standard 2-layer 32-neuron neural net. So far weve been working with a miniature problem of 12 total vehicles on the highway, but in the future well want to test with the full-size setup of 400 total vehicles and 20+ autonomous vehicles.

I initially began with a naive reward function $-\sum_k f_k$. This reward function causes all the cars to stop in the road (or as many cars as possible), indicating we need a component that incentivizes reaching a destination. Thus our current reward function is a weighted average between fuel consumption and distance from a destination (set to 3.36km or 4 times the length of the track) for all the cars $-\sum_k 0.9 \cdot (3.36km - d_k) + 0.1 f_k$. Future work would involve tweaking the weights in the reward function and investigating how the policy changes.

IV. Results

In reinforcement learning, the algorithm samples data as it trains and tries to explore intelligently. Thus, traditionally there is no concept of a train/validate/partitioning. Instead, there is a training period of some rollouts: in our case we used a batch size of 2,000 samples per iteration and typically trained for 1,000 iterations. At some point the return will converge, implying the policy has converged to an optimum. Success is determined by total return averaged over all the rollouts in the final iteration. One can generate rollouts using the learned policy, representing test time.

A. Baseline Model

After I could run experiments on AWS, I was able to supply enough computational power to solve the problem. The following figure represents several test-time rollouts using the learned policy for the 2-vehicle case. In this experiment I would run each rollout for 500 timesteps.

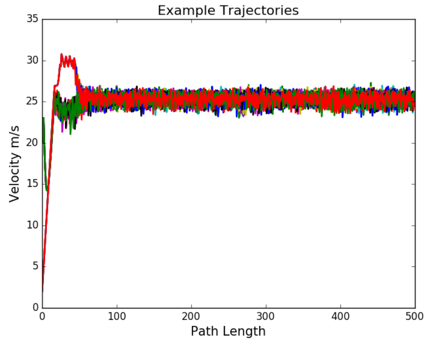


Fig 4. Example test-time rollouts from the solved baseline problem.

I can make the problem more difficult by increasing the number of vehicles, thus linearly growing the state and action dimensionality. The results are as follows:

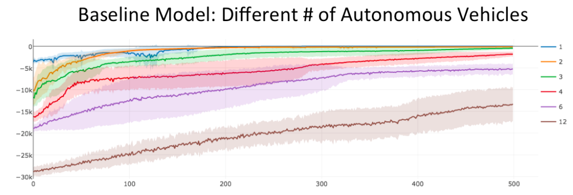


Fig 5. The results for the baseline model when we varied the number of autonomous vehicles. Quantitative results provided in the table below.

No. of cars	Avg Return / Total Cars	Error
1	-20	± 0.22
2	-70	± 0.42
3	-154	± 0.62
4	-464	± 1.08
6	-873	± 1.48
12	-1,109	± 1.66

As the problem becomes more difficult, the returns (as expected) are lower and take longer to converge. Error represents difference in velocity from the goal velocity of 25m/s. The increase in error in this problem is an optimistic signal that multi-agent control might not be as difficult as we initially feared.

B. Final Model

The main difference I noticed between the partial observability and full observability case is higher variance. The following figure compares an experiment where the state is only velocity and the reward function is the negative sum of fuel consumption. One can qualitatively notice that the full observability converged faster and then stayed near the converged value, whereas in the partial observability case the policy was extremely unstable and the return could grow worse. In some cases the policy in the partial observability case caused traffic jams and the program would crash, whereas this never happened in the full observability case.

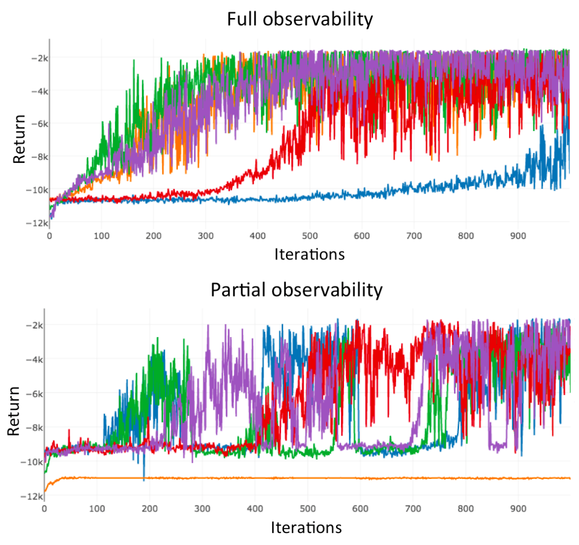
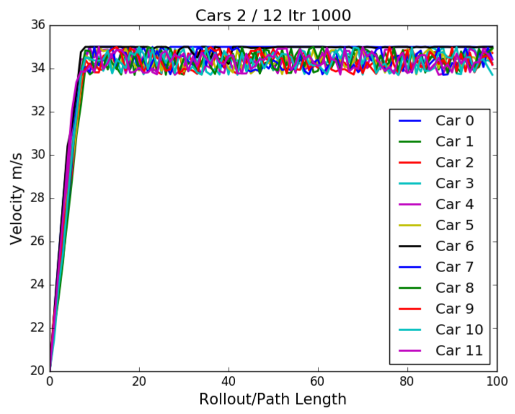
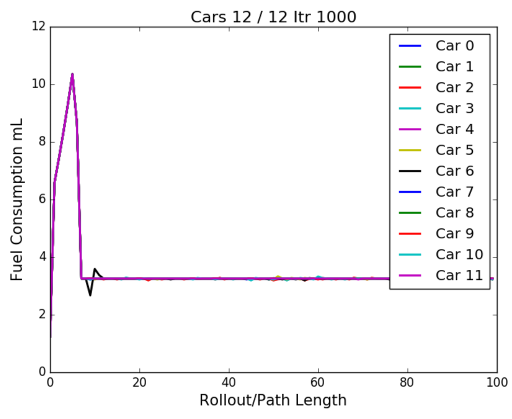


Fig 6. The return for each seed in the partial observablity case vs the full observability case. Notice that the partial observability case has greater variance and is more unstable.

With our reward function, the distance to destination component is heavily weighted so that the optimal behavior is for all the cars to drive as fast as possible. The following figures represent an experiment with 2 autonomous vehicles and 12 vehicles on an 0.84km highway, with a final destination achieved after driving 3.36km.



Notice that the autonomous vehicles Car 0 and Car 6 drive steadily at 35 m/s while the other human-powered vehicles oscillate around 34 m/s.



In addition, fuel consumption for the autonomous vehicles was a steady 2.31mL, whereas fuel consumption for the human-powered vehicles tended to jaggedly jump between 0 and 4mL. This could be an artifact of how SUMO ”drives” the human-powered vehicles, and needs to be investigated further.

No. of cars	Total Return
Baseline: 0	-1.84M
2	-1.80M
12	-1.88M

A surprisingly result is that the return significantly improved between the baseline (no autonomous vehicles) and 2 autonomous vehicles, then worsened with 12 autonomous vehicles. This remains to be investigated, although a hypothesis is that the larger action space made the problem more difficult.

V. Tools

For software I used rllab [9], which is an open-source reinforcement learning library developed at Berkeley. Experiments are run on AWS EC2 spot instances, which allows us to run many multi-hour experiments in parallel. I used the Docker platform to build a virtual machine on Maven 3 (a variant of Ubuntu) with rllab, SUMO, and all dependencies set up. The image is available publicly on DockerHub here.

VI. Conclusion

Although the goal of this research is to reduce energy consumption on highways, it must be posed as a secondary goal of the MDP. The primary goal is for cars to reach their destinations, which needs a much higher weighting in the reward function. In addition, I learned how to run experiments on AWS and interface rllab with SUMO. Many of our results need to be investigated further: the partial vs full observability comparison, how multi-agent control will affect our results and what happens when we tweak the weights in the reward function.

Moving forward, were interested in implementing constraints in the action space to ensure the learned policy is collision-free. The interesting case is when traffic is congested and there is risk of jams, but we can only construct these experiments when were confident the cars wont crash into each other. Weve chosen not to implement a negative penalty for crashes because it would impact the gradient too heavily and the cars would choose to stop in the highway instead of reaching their destinations. The alternative is that the cars keep driving but the penalty for crashes is too low and the learned policy encourages unsafe, fast driving.

Our interface to the simulator right now is prohibitively slow: it takes roughly 4 hours to get 2 million samples, and in the future well want 6+ million samples. I am working on improving the interface to the simulator: 1) I open an I/O connection to the simulator subprocess every rollout and I plan on reducing that to once per iteration 2) I will integrate with OpenAIs parallel sampler, which does rollouts in parallel. Hopefully we can do rollouts in parallel on separate AWS spot instances; if not, I will look into implementing that myself.

Finally, there are a couple other directions we can go. Weve been primarily working on the single lane case to get it to work, so we want to expand to multiple lanes because the results will be more interesting. Having many autonomous vehicles will quickly blow up the action space, leading to high variance. One of our ideas for variance reduction is finding a baseline thats dependent on both state and action while remaining either bias-free or relatively low bias.

Team Contributions

Leah Dickstein: 100%

Acknowledgments

This work was done in collaboration with Professor Alex Bayen's lab, as a project under Berkeley DeepDrive. I would like to acknowledge Cathy Wu for organizing the group and providing invaluable feedback and advice. In addition, Yan (Rocky) Duan provided help and support working with rllab. Eugene Vinitsky, Kanaad Parvate, and Nathan Mandi had interesting questions, insights, and advice.

References

[1] Zia Wadud, Don MacKenzie, Paul Leiby. ”Help or hindrance? The travel, energy and carbon impacts of highly automated vehicles”. (2016). Transportation Research Part A: Policy and Practice Vol. 86 Pages 1-18

[2] Francois Belletti, Daniel Haziza, Gabriel Gomes, Joseph Gonzalez, and Alexandre Bayen (2016) Expert level control of Ramp Metering based on Multi-task Deep Reinforcement Learning. Neural Information Processing Systems Deep Reinforcement Learning Workshop. https://drive.google.com/file/d/0B1PUpk7kwWu-TXVNTUlQcTVueDhCQnYzOUdSd0lNTlowNEJR/view

[3] Stevens, M. and Yeh, C. (2016). Reinforcement Learning for Traffic Optimization. http://cs229.stanford.edu/proj2016spr/report/047.pdf

[4] Polson, N. and Sokolov, V. (2016). Deep Learning Predictors for Traffic Flows. arXiv preprint arXiv: 1604.04527

[5] Lv, Y., Duan, Y., Kang, W., Li, Z., and Wang, F.-Y. (2015). Traffic flow prediction with big data: A deep learning approach. Intelligent Transportation Systems, IEEE Transactions on, 16(2):865-873.

[6] Karlaftis, M. and Vlahogianni, E. (2011). Statistical methods versus neural networks in transportation research: differences, similarities, and some insights. Transportation Research Part C: Emerging Technologies, 19(3):387-399.

[7] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, Laura Bieker. Recent Development and Applications of SUMO – Simulation of Urban MObility. International Journal on Advances in Systems and Measurements, Vol. 5, 2012.

[8] John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, Pieter Abbeel. Trust Region Policy Optimization. arXiv: 1502.05477v4, Jun 2016.

[9] Yan Duan, Xi Chen, Rein Houthoos, John Schulman, Pieter Abbeel. Benchmarking Deep Reinforcement Learning for Continuous Control. Proceedings of the 33rd International Conference on Machine Learning (ICML), 2016.