DeepTraffic: Driving Fast through Dense Traffic with Deep Reinforcement Learning

Lex Fridman, Benedikt Jenik, and Jack Terwilliger

Massachusetts Institute of Technology (MIT)

Abstract—We present a micro-traffic simulation (named "DeepTraffic") where the perception, control, and planning systems for one of the cars are all handled by a single neural network as part of a model-free, off-policy reinforcement learning process. The primary goal of DeepTraffic is to make the hands-on study of deep reinforcement learning accessible to thousands of students, educators, and researchers in order to inspire and fuel the exploration and evaluation of DQN variants and hyperparameter configurations through large-scale, open competition. This paper investigates the crowd-sourced hyperparameter tuning of the policy network that resulted from the first iteration of the DeepTraffic competition where thousands of participants actively searched through the hyperparameter space with the objective of their neural network submission to make it onto the top-10 leaderboard.

I. Introduction

The use of neural networks to estimate the state-action value function in a reinforcement learning (RL) framework has recently been demonstrated to be a remarkably powerful way to learn how to succeed (i.e., achieve high reward) in a very large state space with no prior knowledge [1], [2], [3], [4]. These accomplishments appear to the optimistic eyes of the authors to be promising steps down the road toward general intelligence. This promise is a source of excitement that our work seeks to share with the world. To this end, we have created a simulation of traffic on https://selfdrivingcars.mit.edu/deeptraffic where one of the vehicles is a reinforcement learning agent operating according to the Q-function estimated by a neural network. The simulation, training, and inference all can be performed with JavaScript [5], inside a web browser. Practically, this means that a person can tune the hyperparameters of and train their very first neural network in seconds. In other words, the design of DeepTraffic is motivated by a singular goal: make the exploration of deep reinforcement learning (Deep RL or DRL) accessible to everyone.

DeepTraffic is more than a simulation, it is a competition that served and continues to serve as a central project for a course on Deep Learning for Self-Driving Cars in which to-date over 7,000 students participated in-person and online on https://selfdrivingcars.mit.edu. The competition's first iteration has received over 13,000 submissions. Our paper explores these submissions by looking at them as a crowd-sourced tuning of neural network hyperparameters. Such an exploration raises practical questions, in a context of a specific simulated world, about what works and what doesn't for using deep reinforcement learning to optimize the actions of an agent in

that world. Moreover, we take a broader look about the impact of that single intelligent agent on the macro-patterns of traffic flow, and show a deep RL agent may in fact alleviate traffic jams not create them despite operating under a purely greedy policy.

The latest statistics on the number of submissions and the extent of crowdsourced network training and simulation are as follows:

- Number of submissions: 13,417
- Students participating in competition: 7,120
- Total network parameters optimized: 168.5 million
- Total duration of RL simulations: 96.6 years

Deep reinforcement learning has shown promise to learn to successfully operate in simulated physics environments like MuJoCo [6], in gaming environments [7], [1], and driving environments [8], [9]. Yet, the question of how so much can be learned from such sparse supervision is not yet well explored. We hope to take steps toward such understanding by drawing insights from the the exploration of the crowdsourced hyperparameter tuning (as discussed in §III) for a well-defined traffic simulation environment. This includes analysis of network size, temporal dynamics, discounting of reward, and impact of greedy behavior on the stability and performance of the macro-traffic system as a whole.

II. Simulation and Competition

DeepTraffic primarily is a JavaScript-based highway traffic simulation environment that lives in a web browser. It looks at a strip of highway with seven lanes and 20 cars driving at the same time (see first column of Fig. 1). No car is allowed to travel above the speed limit of 80mph.

DeepTraffic uses a discrete occupancy grid (see second column of Fig. 1) as a simplified representation of the world. Each cell contains the speed of the car occupying it, or an even higher maximum speed value if the cell is not occupied. This grid, while providing an efficient method of detecting collisions, also serves as a representation of the achievable driving speed in different areas of the map and therefore provides information where to go to be fast.

Since the focus of this simulation is to learn efficient movement patterns in heavy traffic, the problem of avoiding collisions is abstracted away by using a "safety system" (see third column of Fig. 1). This safety system looks at the occupancy grid and prevents actions that would lead to a crash, for example accelerating when there already is a car in front or changing lanes with a car on the next lane.

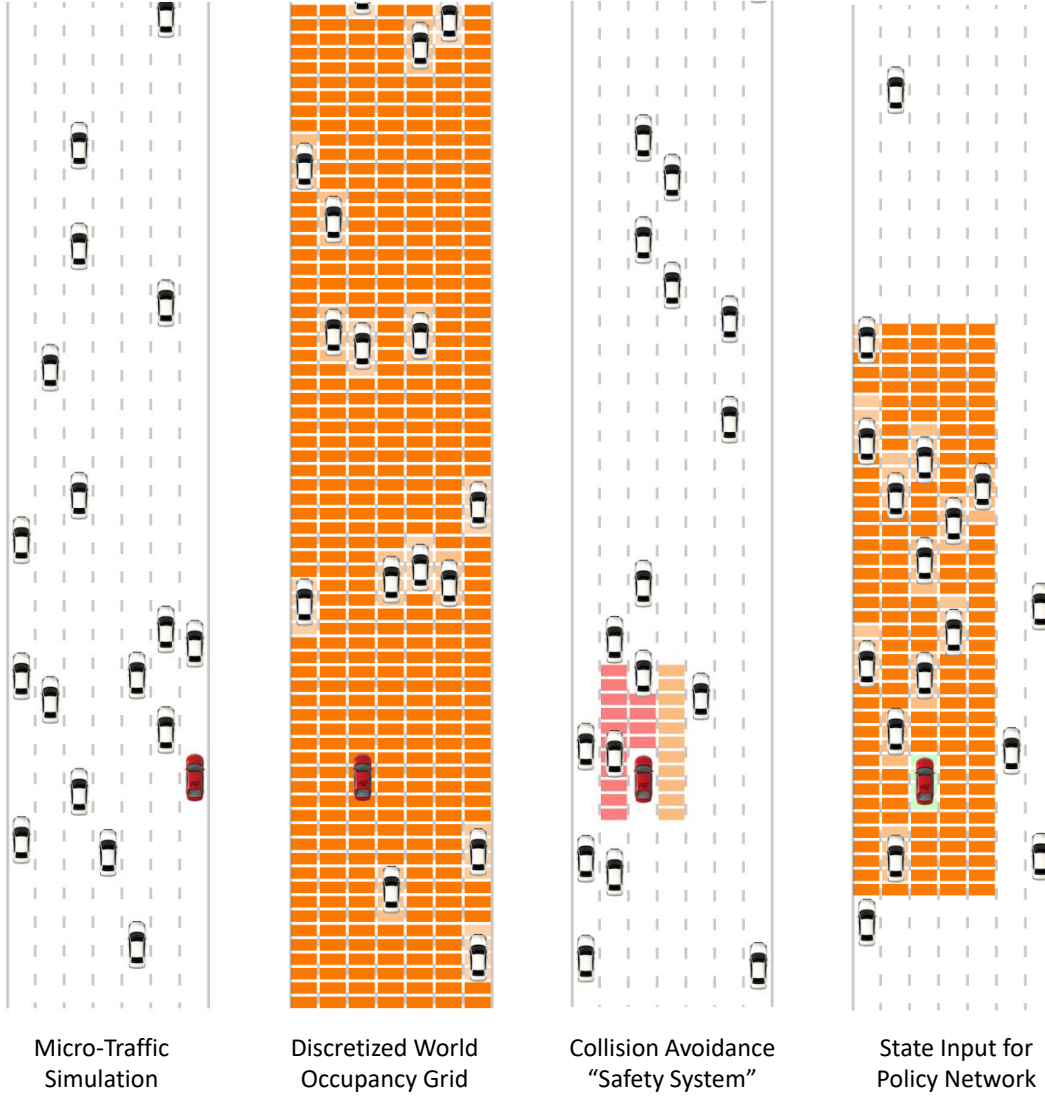|			
Micro-Traffic Simulation	Discretized World Occupancy Grid	Collision Avoidance "Safety System"	State Input for Policy Network

Fig. 1: Four perspectives on the DeepTraffic environment: the simulation, the occupancy grid, the collision avoidance system, and the slice of the occupancy grid that represents the reinforcement learning "state" based on which the policy network learns to estimate the expected reward received by taking each of the five available actions.

All cars have a choice of five actions, changing lanes to either side, accelerating or slowing down and simply doing nothing/keeping the same settings as before. The other cars choose the actions at random, e.g. changing lanes if the safety system permits. The random sampling of these actions is biased towards patterns that seem realistic, for example not changing the lanes too often in general, but with a higher probability if stuck behind another car.

Further, there is one car (displayed in red) that is not using these random actions. This is the car controlled by the deep reinforcement learning agent. It is able to choose an action every 30 frames (the time it takes to make a lane change) and gets a cutout of the state map as an input to compute its actions (see fourth column of Fig. 1).

Participants in the competition get a predefined neural network in an in-browser DQN implementation [5], where they get to configure different hyperparameters to achieve the best performance, meaning the highest average driving speed. For this network they are able to change the size of the input by defining the number of patches/cells ahead of the front of the car, behind it and next to the car, the network gets to look at. In addition, they get to specify the network layout, meaning adding and removing layers, and changing their sizes and activations. Beyond that, participants can also configure training options like learning rate and regularization methods, and also reinforcement learning specific parameters like exploration vs exploitation and future reward discounts.

When they are done, they can press a button to train the network in the browser, using a separate thread with drawing disabled and therefore much faster performance, while still being able to see improvements running live in the main visualization. When the participants are satisfied with the quality of their network they can submit the network for server-side evaluation to earn a spot in the leaderboard, the
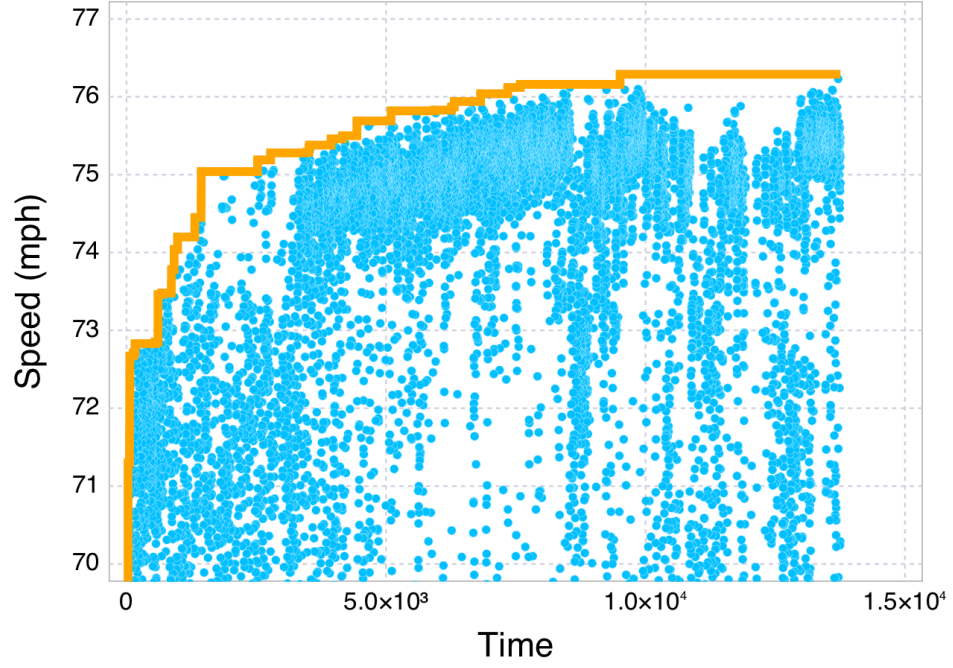
(a) The blue circles designate the evaluation speed (termed "score"). The orange squares designate the highest score achieved up to that time.
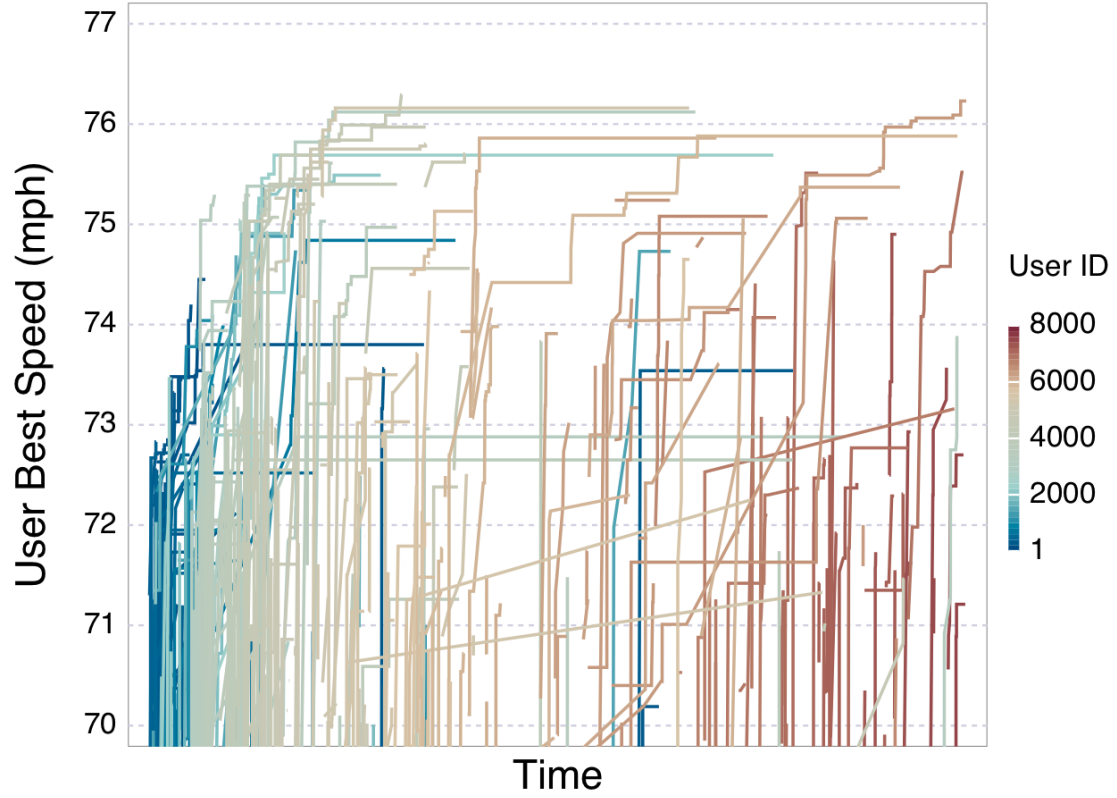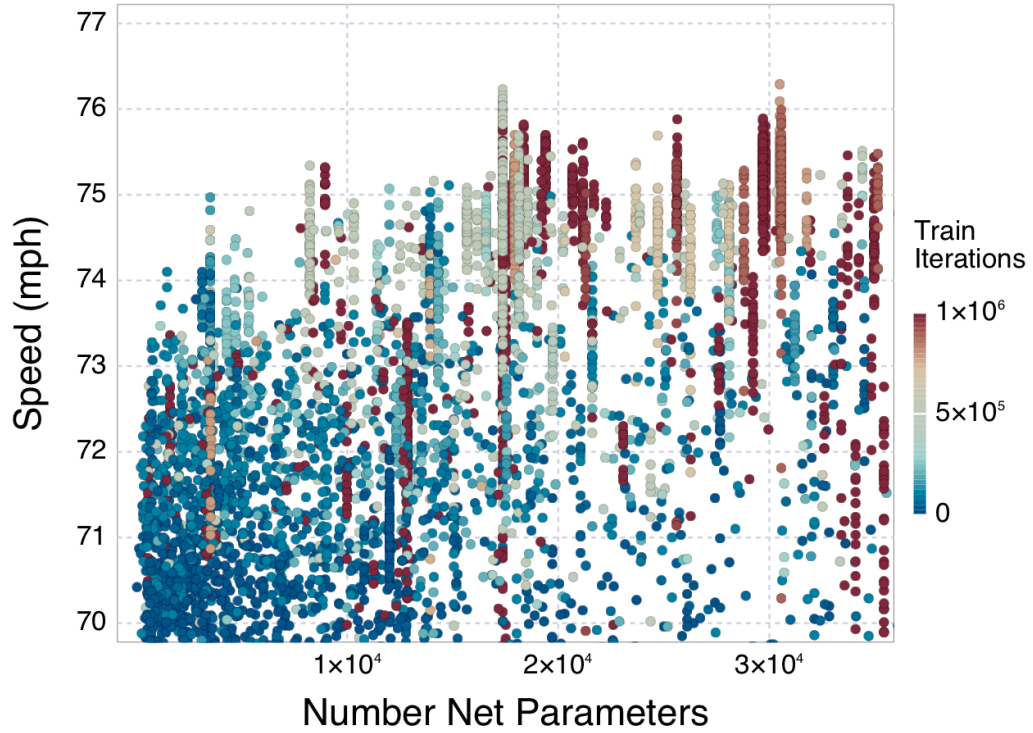

(b) Each continuous line designates the improvement achieved by individual users over multiple submissions.

Fig. 2: The x-axis is time since the start of the competition (from day 1 to day 90), and the y-axis is the score achieved by each of the 13,000 submitted deep RL agents.
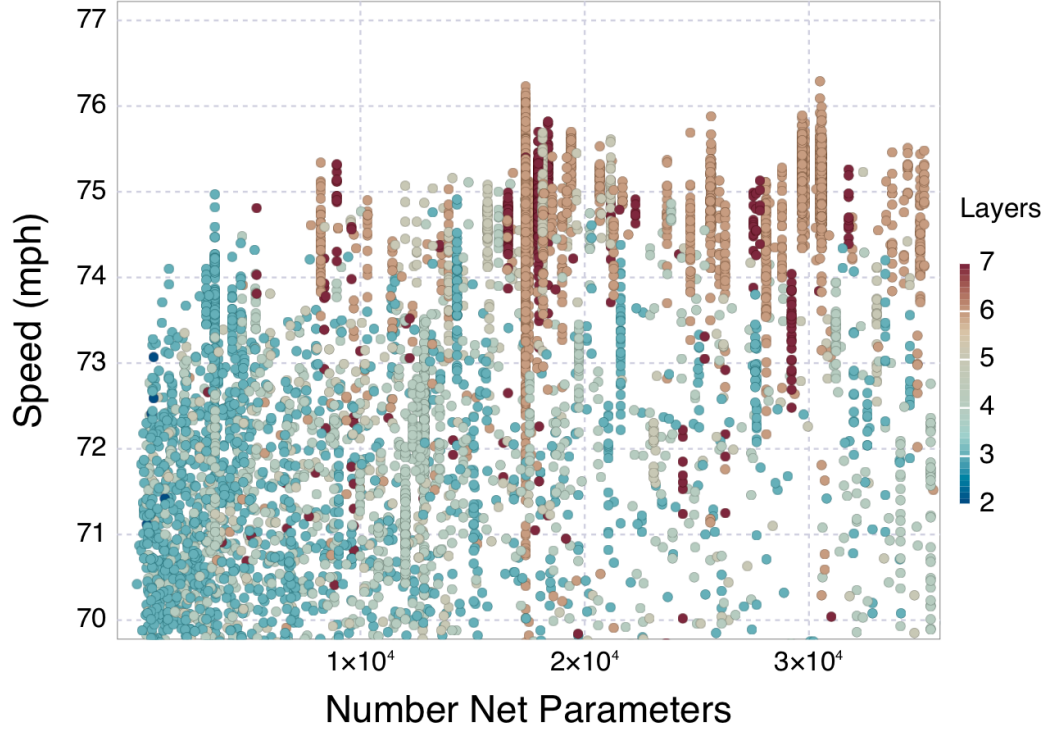
(a) Circle color indicates number of hidden layers from 1 layer (dark cyan) to 7 layers (dark red).

(b) Circle color indicates number of training iteration from 0 (dark cyan) to 15 layers (dark red).

Fig. 3: The evaluation speed achieved as the number of parameters is increased while varying either the number of layers or the number of training iterations. The y-axis doesn't show speeds below 70mph as those were achieved by networks that underwent very little training or had very few parameters to train.

most recent iteration of which is shown in Fig. 5b.

The evaluation process is fundamentally a challenging task (see §III-F) due to the need to maintain fair, stable evaluation of intelligent agent performance through a high-dimensional non-deterministic state space. The initial evaluation is carried out by running the agent 10 times through "runs" of 100,000 time steps each. The median of average speeds over those runs is taken as the resulting "score". If the score puts the agent into top 100 of all current submissions, its evaluation is rerun for 1000 iterations of 100,000 time steps each, adding up to a total of 100 million simulation time steps for a single top-100 agent simulation.

The DeepTraffic competition has an active community of participants, growing consistently since its release in January 2017. Fig. 2 gives an overview of submissions over time, the first subfigure (Fig. 2a) showing the aggregate improvement of agents over time and the second subfigure (Fig. 2b) showing the per-user improvement based on human-driven hyper-parameter exploration, with the top-10 leaderboard as it currently stands shown in Fig. 5b.

An interesting observation in the competitive chase toward greater performing Deep RL agents that is common to many Deep RL tasks is that neither the authors nor the competitors had an explicit model for an optimal policy. The high dimensionality and the stochastic nature of the state space made it intractable for model-based path planning methods [10]. In the early days of the competition, many competitors claimed that it's impossible for the car to achieve a speed above 73 mph. Once that barrier was broken, 74 mph became the new barrier, and so on, reaching the current socially-defined performance ceiling of 77 mph as shown in Fig. 2a. From the social perspective, it is interesting to observe that such self-imposed ceilings often led to performance plateaus, much like those in other competitive endeavors [11].

Network parameters were not shared publicly between competitors, but in several cases, small online communities formed to distribute the hyper-parameter tuning process across the members of that community. An interesting result of such distributed hyper-parameter tuning efforts was that occasional plateaus in performance were broken by one individual and then quickly matched and superseded by others.

III. NETWORK PERFORMANCE AND EXPLORATION

The large-scale crowdsourcing of hyperparameters through over 13,000 network submissions allows us to gather observations about what hyperparameters have a significant impact on evaluated agent performance and what come at a great cost without much performance gain. The key insights gathered from this exploration are highlights in the following subsections. In support of these insights, Fig. 2 shows the evolution of submission scores over time, Fig. 3 shows the impact of network size, depth, and training iteration, and Fig. 4 shows a rich set of impactful hyperparameters as they interact with final network performance in intricate and sometimes counter-intuitive ways.

A. Insight: Size Matters

The best networks generally have the most parameters and many layers. As Fig. 3a shows, the conventional wisdom holds up for DeepTraffic: the larger and deeper the network, the better the performance. However, there is some diminishing returns when the input size to the network is too large, as indicated by Fig. 4a. Accompanying the increased performance of networks with more parameters, is the requirement that the number of training iterations increases as well as shown in Fig. 3b. Therefore, the cost of using larger networks is a longer training time, which perhaps contributes to the observed diminishing returns of adding more parameters.

B. Insight: Live in the Moment

Knowledge of temporal dynamics does not significantly improve the agent's ability to achieve higher reward. This is a counter-intuitive observation for a simulation that is fundamentally about planning one's trajectory through space and time. As Fig. 4e shows, not looking back in time at all provides the best performance. Put another way, the temporal dynamics of the driving scene that led up to the current state does not have a significant impact on the evolution of the environment in the future, and thus optimization of actions through that environment does not need to consider the past. This is surprising since the high-dimensional state space in Deep RL approaches commonly encode the temporal dynamics of the scene in the definition of the state space [12], [2]. This is done in order to capture a sequence of partial observations that in-sum form a more complete observation of the state [13]. For DeepTraffic, the agent does not appear to need a larger temporal window (injecting "short-term memory" into the state representation) in order to more fully specify the current state.

C. Insight: Look Far Forward

Looking ahead spatially suffers much less from diminishing returns than does looking behind. As shown in Fig. 4c, the more of the state space in front of the agent that the network is able to consider, the more successful it is at avoiding situations that block it in. Fig. 4d, on the other hand, shows that performance gains level off quickly after more than 5 patches behind the agent are considered. Spatially, the future holds more promise than the past. Similarly, Fig. 4b shows that increases how far to the sides the agent looks suffers from diminishing returns as well, maxing out at 3 lanes to each side. The value of 3 in Fig. 4b represents sideways visibility that covers all lanes when the vehicle is positioned in the middle lane. In general, high-performance agents tend to prefer the middle lane that allows them the greatest flexibility in longer-term navigation through the vehicles ahead of it.
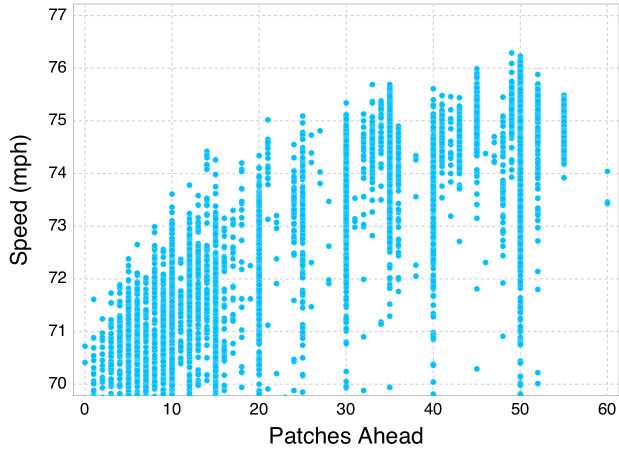
D. Insight: Plan for the Future

One of the defining challenges for reinforcement learning is the *temporal credit assignment problem* [14], that is, assigning value to an action in a specific state even though that action's consequences do not materialize until much later in time. As
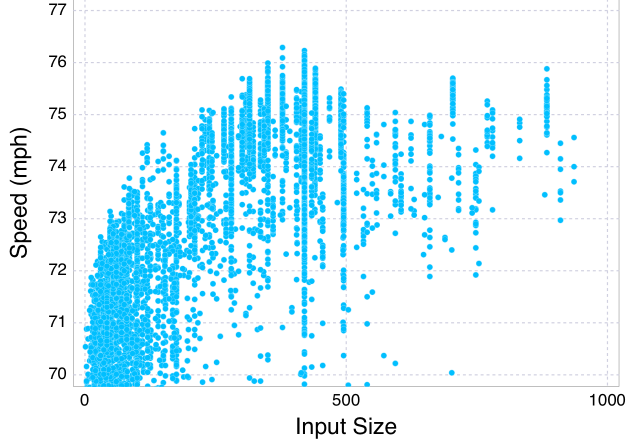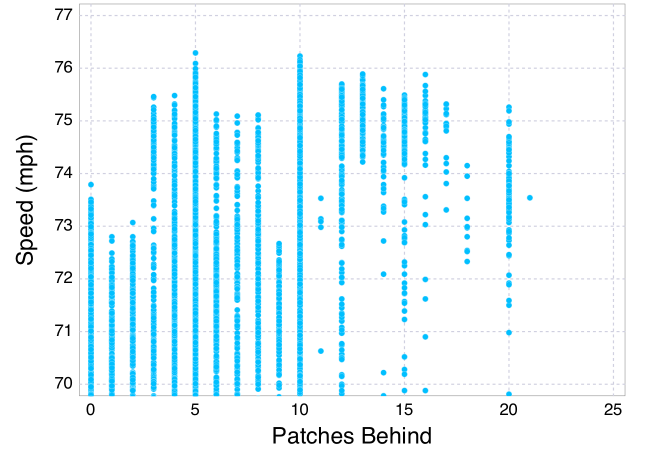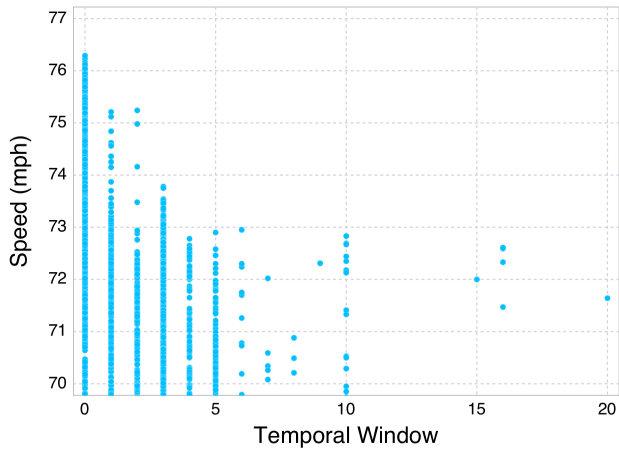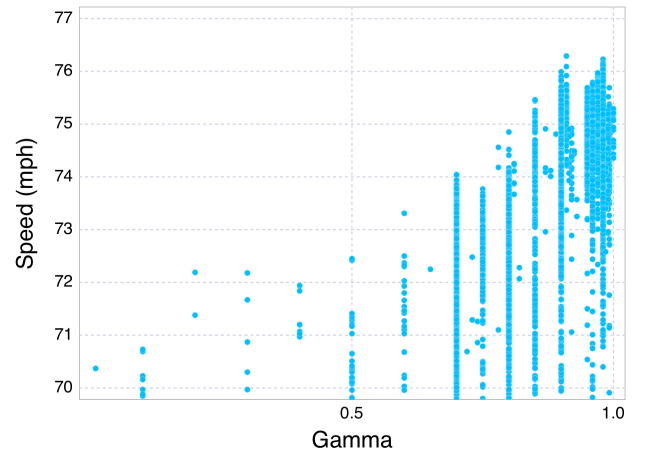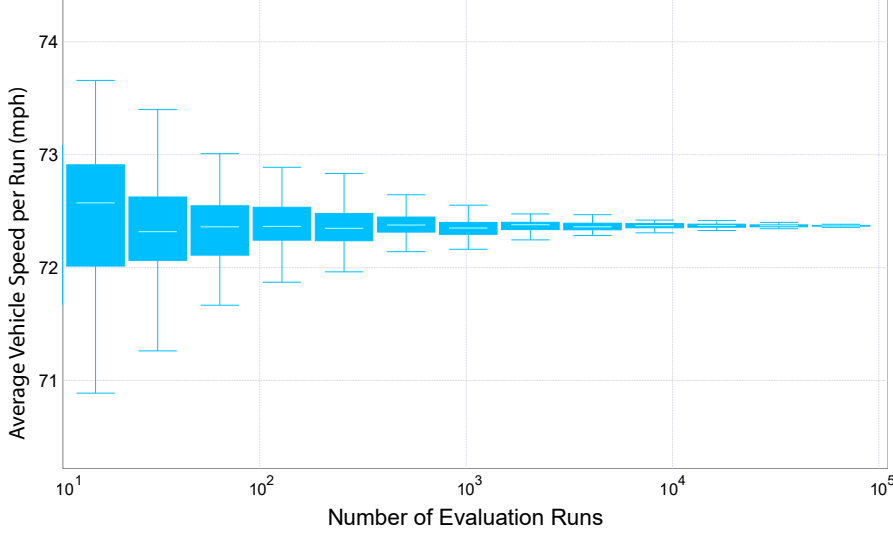
(a)
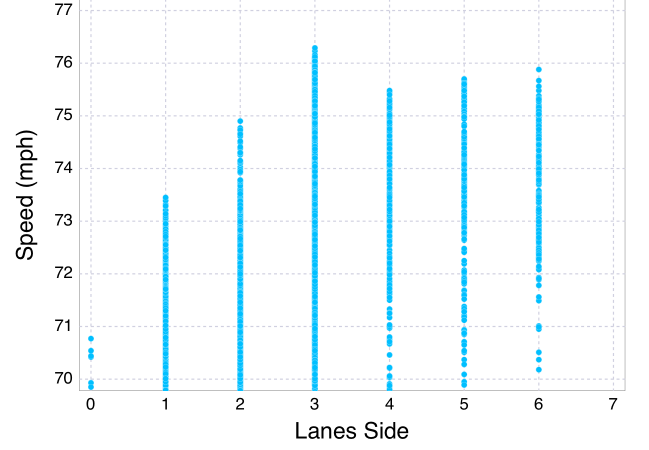
(b)

(c)

(d)

(e)

(f)

Fig. 4: Exploration of most impactful hyperparameters' effect on the overall performance of the reinforcement learning agent. Each point in the scatter plot represents one of the 13,000 agents investigated in this work. In each of the 6 plots, the y-axis is the achieved speed during the evaluation process (see §III-F).

Rank	User	MPH
1	Hoan Nguyen	76.29
2	Ali Reza Kohani	76.23
3	Jorcus96	76.16
4	leiming yu	75.97
5	Jordan Ott	75.91
6	Abhijeet Vaidya	75.88
7	Nándor Kedves	75.83
8	3upperm2n	75.80
9	Tamás Csaba Kádár	75.77
10	Mark S.	75.73

(a) A box plot showing the narrowing variance with increasing number of evaluation runs. Each run includes 10,000 simulation time steps.

(b) Leaderboard showing the top 10 agents of over 13,000.

Fig. 5: The evaluation process and the resulting score used to determine the competition winners.

shown in Fig. 4f, minimizing the discounting of future reward by increasing γ (referred to as "gamma" in the figure) nearly always improves performance. Much like the previous insight, the future is valuable for estimating the expected reward and planning ones actions accordingly. Achieving good average speed requires that the agent avoid clusters of other cars, and often the avoidance strategy requires planning many moves ahead.
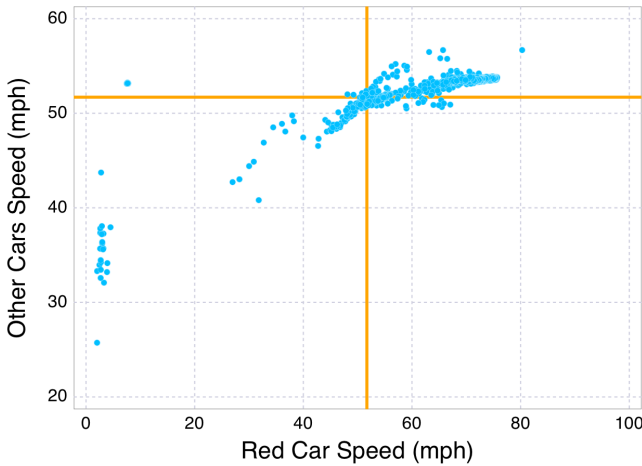

Fig. 6: The effect of the deep RL agent (red car) on the average speed of other cars. When the "red car" moves according to the same model as the other cars, traffic flows at 51.7 mph.

E. Insight: Greedy is Good for All

One of the most counter-intuitive and fascinating observations consistently made in DeepTraffic is that the higher the average speed achieved by the network, the higher the overall throughput of traffic measured by the average speed of other cars on the road. At least, when only one of the agents is intelligently planning its actions, it benefits everyone when that agent succeeds. Fig. 6 illustrates this point by showing the increase in overall traffic flow when the agent achieves speeds higher than the 51.7mph average achieved when movement is not intelligently planned. Future work will look at the effects multiple greedy agents have on individual agent speed and overall traffic throughput.

F. Insight: Evaluation is Expensive

Fig. 5a shows that it takes at least ten million simulation time steps (shown as 100 evaluation runs in the figure) to converge towards a stable estimation of deep reinforcement learning agent performance with a standard deviation of possible scores falling below 0.1. One of the open problems of running a deep reinforcement learning competition is to have an effective way of ranking the performance of the submitted policy networks. The very large size and non-deterministic nature of the state space make stable, consistent, and fair evaluation of an agent very challenging, given the amount of computational resources it takes to execute a forward pass 10+ million times through a network with 40,000+ parameters for each of the 13,000+ agents submitted to date.

IV. CONCLUSION

In this work we seek to make deep reinforcement learning accessible to tens of thousands of students, researchers, and educators. We look back at the crowdsourced hyperparameter space exploration and draw insights from what worked and what didn't. The two most surprising insights are as follows. First, the future, both in spatial and temporal domains, holds

much more useful information for the action determination of a deep reinforcement learning agent than does the past. Second, the faster the agent goes, the faster the rest of traffic flows. In other words, a single agent following a greedy strategy is good for everyone, at least when everyone is not also planning their movement intelligently.

REFERENCES

[1] 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.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[4] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning." in *AAAI*, 2016, pp. 2094–2100.

[5] A. Karpathy, "Convnetjs: Deep learning in your browser (2014)," *URL http://cs.stanford.edu/people/karpathy/convnetjs*, 2014.

[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] S. Shalev-Shwartz, N. Ben-Zrihem, A. Cohen, and A. Shashua, "Long-term planning by short-term prediction," *arXiv preprint arXiv:1602.01580*, 2016.

[9] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," *arXiv preprint arXiv:1610.03295*, 2016.

[10] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the rrt," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1478–1483.

[11] R. Bannister, *Four-Minute Mile*. Rowman & Littlefield, 2004.

[12] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, 2016, pp. 1928–1937.

[13] P. Zhu, X. Li, and P. Poupart, "On improving deep reinforcement learning for pomdps," *arXiv preprint arXiv:1704.07978*, 2017.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.