Navigating Holiday Traffic

Adam Fishman (afishman), Nathan Hatch (nhatch2), Yuxiang Yang (yuxiangy)

1 Introduction

Changing lanes in traffic requires complex coordination between many agents. It is a difficult task even for human drivers. Part of the challenge is that the environment contains moving obstacles. But another, perhaps more important, part of the challenge is that our car's actions *affect the behavior* of the other cars. In a traffic jam, merging is often impossible unless drivers in adjacent lanes react to the merge by yielding. These complications mean that we cannot hope to train a robotic car to change lanes using mere supervised learning. In order to account for how our actions affect the state of the world in the future, we need the full power of reinforcement learning (RL).

One challenge in applying RL to this scenario is the need for a *model* of the behavior of other vehicles in reaction to ours. Training on data collected offline suffers from the problem of *domain shift*, wherein our model will only be accurate as long as the behavior of the robotic car closely matches the policy used to collect the data. Hence, even if we use model-free RL algorithms, we still require some kind of simulator for training.

In this project, we aim to explore various simulated lane-change environments and investigate the performance of RL algorithms therein. The simulator must be sophisticated enough to generate complex reactive behaviors, and the traffic configuration must be difficult enough that trivial solutions (such as waiting for large traffic gaps to emerge spontaneously) do not work. We designed several variants of a highway simulator and trained RL policies to explore the complexity of this problem area. These RL problems turned out to be quite difficult, so our analysis focuses on understanding the limitations of the RL algorithms used. Our hope is to find improvements upon these algorithms and ultimately publish this work.

2 Related Works

Modeling traffic behaviors is an important first step to autonomous driving. Even in the absence of an ego-vehicle, traffic is a complex dynamical system to model. In the field of transportation engineering, traffic has long been described as a dynamical system [12] where individual cars are constantly varying their paths to achieve lower cost. Sriram et al. [10] are able to use computer vision and recurrent neural networks to predict future road states. Modeling the roadway is, in essence, a simplified version of pedestrian prediction, which is a well studied area in its own right [15, 7]. For our project, we looked into using simplified model to aid the learning of ego-vehicle, and compared its behavior to model-free methods.

To navigate traffic, the ego-agent must learn to make lane changes effectively, which is a complex process that requires interaction with other vehicles. According to Chandru et al. [2], one of the main challenges in dense traffic is the difficulty of finding a large enough gap to ensure worst-case collision avoidance. They work to address this problem by using a large set of collision avoidance behaviors to reduce the required gap size. Another approach is to use turn signaling to encourage other cars to yield enough space. This falls under the larger heading of inter-vehicle communication with humans, as surveyed by [3].

Looking to solve lane-changing with reinforcement learning, Sadigh et al. [11] formalize the problem of lane change as a dynamical system where the robot's action choice affects both agents. The authors use Inverse Reinforcement Learning to learn a policy for human drivers and then optimize a solution within an MPC loop. Learning this behavior is simplified by only allowing for one human and one

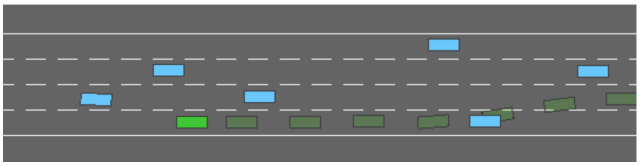


Figure 1: The original 4-lane highway driving environment.

autonomous car on the road. Model-free reinforcement learning has been shown to be an effective method for lane-change decisions as well. Hoel et al. [4] used a Deep-Q Network (DQN) [8] to model lane change decisions. Our project uses a similar formulation as a baseline method.

3 Preliminaries

3.1 Reinforcement Learning Formulation

At a high level, we model our road as a collection of agents where one agent is actively controlled by our system and all others follow their own, unknown desiderata. We call our controlled vehicle the *ego agent* and the other cars on the road the *other agents*. We define the state of the road s_t as the collective state of all agents together,

$$s_t = \left(s_t^{\text{ego}}, s_t^{\text{other}}\right) \tag{1}$$

Given an ego agent action a_t, the road state then evolves according to some dynamics function:

$$s_{t+1} \sim f(s_t, a_t) \tag{2}$$

This dynamics function encapsulates the way that the entire road will react to a single action. However, the distribution induced by the dynamics depends heavily on the individual policies of the other agents. If we have a perfect model, even a noisy one, of how other agents will move, the dynamics function can be modeled. However, on real roads, not knowing the other agents' policies makes the dynamics difficult to model. Works such as [10] attempt to learn a predictive model of the road, which could be used for model-based reinforcement learning or model-predictive control. In our project, we investigated both model-free and simple model-based approaches.

3.2 Simulated Highway Environment

As discussed in the introduction, learning lane-change policies using model-free RL requires a good simulator. As a starting point, we build our environment on top of a simulator called HighwayEnv [6], shown in Figure 1. To focus attention on high-level decisions, the action space of the ego-vehicle in this environment is a discrete set of 5 meta-actions $\{\text{LANE\_LEFT}, \text{LANE\_RIGHT}, \text{SLOWER}, \text{FASTER}, \text{IDLE}\}$. The environment relies on a low level controller to drive the ego-vehicle according to our high-level instructions. For example, it uses a proportional controller to choose the heading for lane-following.

At each timestep, the car observes its own position and velocity, as well as the relative position and velocity of 10 closest cars. To discourage rapid lane changes, we used the following reward function:

$$r(s_t, a_t) = \frac{v}{30} - 0.3 \times \text{lane\_change} - 1 \times \text{collision} \tag{3}$$

where v is the ego-vehicle's current velocity (in m/s), and {lane_change, collision} are indicator variables for corresponding events.

In the baseline environment, all other cars are controlled heuristically according to the intelligent driver model (IDM) [13] for vehicle spacing and the MOBIL model [5] for deciding when to change lanes. Section 4.3 discusses another way we considered modeling other vehicles.

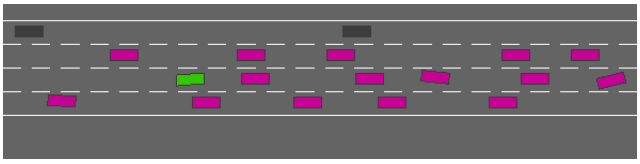


Figure 2: Illustration of the CarpoolEnv environment. We modeled the traffic regulation of High-Occupancy Vehicles by designating the top lane as the "carpool" lane, where only the black HOV vehicle and the green ego-vehicle may drive. Other vehicles (purple) must stay in non-carpool lanes. The ego-vehicle starts in the bottom lane at the beginning of each episode.

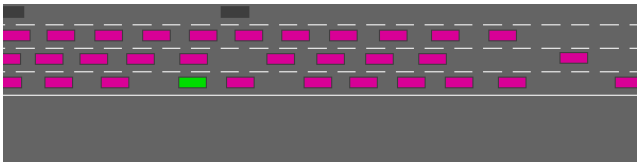


Figure 3: Illustration of the TrafficJamEnv environment. To model traffic jams, we (1) decreased the initial spacing between vehicles and (2) decreased the spacing parameters of the intelligent driver model (IDM). Specifically, we decreased DISTANCE_WANTED from 10m to 5m, and decreased TIME_WANTED from 1.5s to 0.15s.

4 Modeling Holiday Traffic

In the original HighwayEnv environment [6], the traffic is sparse enough that it is easy to find opportunities to change lanes. In this section, we describe our modifications to make the problem more difficult.

4.1 Carpool and HOV Lanes

As a first step, we added real-life traffic regulations of carpool lanes and HOV vehicles into the environment (Figure 2), which we now call CarpoolEnv. Adding the carpool lanes poses several challenges to the agent. Firstly, although it is beneficial for the ego-agent to stay in the carpool lane in the long run, merging to the carpool lane might require frequent braking and multiple lane changes. Therefore, the agent needs to consider the trade-off between short and long-term reward. Moreover, since HOV lanes move much faster than regular lanes, the final merge into the carpool lane requires a big gap to avoid collision, which requires the agent to be extremely cautious about nearby vehicles.

4.2 Traffic Jam and Turn Signals

We created a variant of CarpoolEnv called TrafficJamEnv in which the traffic is too dense for opportunistic lane changes (Figure 3). In this environment, it is impossible to change lanes safely unless the other cars yield. To enable yielding behavior, we implemented a form of *turn signaling* by edging the ego-vehicle slightly into the desired lane. This causes the IDM vehicles to yield as though the ego-vehicle were already in their lane. In this environment, the meta-actions LANE_LEFT and LANE_RIGHT instead activate the turn signal. Performing the same action a second time will complete the lane change.

4.3 Other Cars with More Intelligent Policies

The behavior of the other cars in CarpoolEnv and TrafficJamEnv is very passive. To address this, we created a variant of the HighwayEnv called IntelligentAgentEnv where some of the other cars on the road are optimizing an MDP based on their own perceived state of the road. When every car on the road uses this policy, we qualitatively observe a dramatic increase in crashes. Instead, we randomly choose several cars to use an MDP policy, while giving all other cars the IDM and MOBIL model. Due to the computational complexity of this environment, we did not have time to train a policy for it, but we intend to explore this as future work.

5 Solving Holiday Traffic

5.1 Direct Q-learning

As a baseline, we start by applying standard Q-learning techniques for the driving environment. We model the optimal action value function $Q^*(s, a)$ as a multi-layer perceptron (MLP), and fit the parameters using collected experience. Similar to [9], we keep a *replay buffer* to store all recent experiences of the agent. During training, we randomly sample a batch of N experiences $\{(s_n, a_n, s'_n, r_n, t_n)\}_{n=1...N}$ from the replay buffer, iterate through the next-step actions a'_n to compute the one-step optimal Q values of each sample, and optimize for the errors using gradient descent. We use ε-greedy exploration.

5.2 Finite MDP approximation

Due to the complex nature of the highway-driving environment, directly learning the policy without modeling the environment can be inefficient. As an alternative, the authors in [6] proposed a model-based approach to solve the highway driving environment called *finite-MDP*. Similar to model-predictive control, the agent chooses the action at each timestep by building a simplified model of the environment and performing value iteration.

Since value iteration is performed at each timestep, the environment model needs to be small enough for fast convergence. To this end, the authors simplified the problem into a finite MDP by discretizing the current state state s into a finite occupancy grid $s_{\text{fmdp}} \in \{0, 1\}^{|V|\times|L|\times|T|}$, where $s_{\text{fmdp}}[v, l, t] = 1$ means that the ego-vehicle would collide with another vehicle in t seconds, if it stays in lane l with speed v. The authors further assumed that all other vehicles move at constant speed and do not change lanes, which simplifies the finite MDP construction process.

Since the problem is now tabular, tabular value iteration is guaranteed to converge. The authors then chose the action a that maximizes $q_{\text{fmdp}}(s_{\text{fmdp}}, a)$ for the current state. To overcome the model errors, the authors construct and solve the finite MDP at each step with the latest observation.

5.3 Residual Q-learning

Although finite-MDP approximation provides a simple, tractable solution for the environment, the constant speed assumption it makes on other vehicles could still be frequently violated, especially in dense traffic. To overcome this, we propose to learn a *residual Q-function* that fills the gap between the optimal Q-function $q^*(s, a)$ and the Q-function approximated by the finite-MDP approximation $q_{\text{fmdp}}(s_{\text{fmdp}}, a)$, such that

$$q^*(s, a) = q_{\text{fmdp}}(s_{\text{fmdp}}, a) + q_{\text{NN}}(s, a) \tag{4}$$

where q_{NN} is a neural network.

Compared to learning the Q-function directly, learning the residual Q-function can make exploration much more efficient, especially in the early stages of training. Initially, when the output of the residual Q network q_{NN} is small, the Q-function is dominated by the finite-MDP Q-function q_{fmdp}, which is already aware of basic skills, such as collision avoidance. As the learning continues, the residual Q network can gradually learn from experience and override the values of q_{fmdp}. To train the residual Q network, we modify the replay buffer to also store in the finite-MDP approximation of each state, so that $q_{\text{fmdp}}(s_{\text{fmdp}}, a)$ can be retrieved during training.

6 Results and Analysis

6.1 CarpoolEnv

The constant-speed, constant-lane assumption of finite-MDP does lead to frequent collisions in CarpoolEnv, which we illustrate in Figure 4. When the traffic is dense, the change in speed or lane of nearby vehicles could significantly affect the performance of the ego-vehicle, and more advanced modeling of neighbouring vehicles is required to effectively navigate in such environments.

The performance of the learning-based approaches is shown in Figure 5. Learning to solve the entire environment from scratch, direct Q-learning learns very slowly and could hardly perform any better

Figure 4: Some typical failures of the FMDP agent. (Left) The ego-agent fails to understand the lane-changing behavior of the vehicle in front. (Middle) The ego-agent under-estimates the gap required for lane-changing. (Right) The ego-agent failed to notice the braking of car in front and kept driving at constant speed.

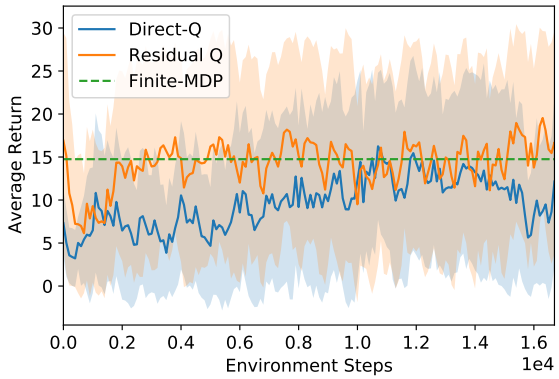


Figure 5: Learning curves for all our agents in CarpoolEnv. Finite-MDP does not require training and the average performance is shown in the dashed green line.

than the finite-MDP approximation. With the aid of the finite-MDP, residual Q-learning learns much faster, but it still does not out-perform the finite-MDP agent. In Figure 6, we show several examples in which residual Q-learning effectively corrects the finite-MDP agent. Further qualitative results can be seen on the website.[1]

Finally, we note that all our agents' performance in the carpool environment are very noisy. We hypothesize this as the result of our binary reward function, which adds penalty on collision, but does not add penalty for getting *close* to the vehicle in front. Therefore, it may be difficult for a model-free agent to learn a proper collision-avoidance behavior.

6.2 TrafficJamEnv

Although finite-MDP approximation works well in the regular carpool environment, building the finite-MDP in the TrafficJamEnv is significantly more challenging, as the yielding behavior of nearby vehicles is no longer modeled by the constant-speed assumption. Therefore, we only trained the direct Q-learning agent for the TrafficJamEnv. To make the training easier, we hand-collected 10 episodes of human demonstrations and added them to the replay buffer at the beginning of training.

As shown in Figure 7a, Q-learning in this environment requires a significantly larger amount of data compared to the previous carpool environment, which reveals the challenges of learning to drive in dense traffic. Distinct driving behaviors emerge from training (Figure 7b), where the agent learns to use turn signals to make spaces in nearby lanes before making full lane changes.

To explore the limits of what is possibile in this environment, we found that a manually-engineered open-loop policy for changing lanes performs very well. The five-action sequence LANE_LEFT (to activate the turn signal), SLOWER, IDLE, FASTER, LANE_LEFT almost always executes a safe lane change. For quantitative comparison, we executed this open-loop policy for the three lane changes, then switched to the finite-MDP policy. Over twenty trials, this crashed twice, with an average return of 26.2. (Compare to Figure 7a.) Qualitative results can be seen on the website (see footnote 1).

[1] https://sites.google.com/cs.washington.edu/holiday-traffic

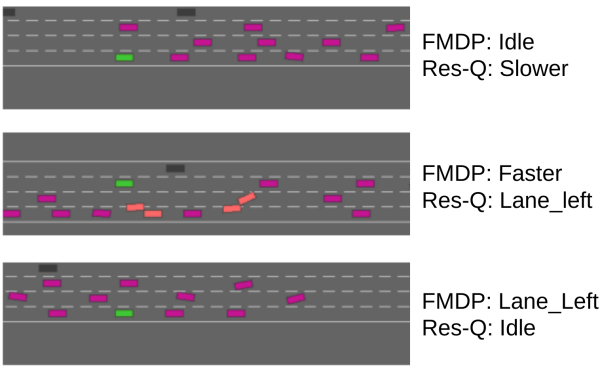
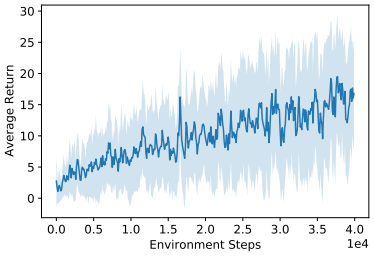
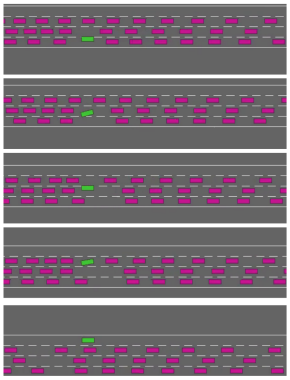


Figure 6: Examples of residual Q-learning correcting the FMDP-agent's behavior. (Top): Residual-Q decides to slow down the vehicle to avoid potential collision. (Mid): Residual-Q shifts the vehicle earlier to the carpool lane for long-term benefit. (Bottom): Residual-Q cancels a lane-change to avoid collision.



(a) Learning curve for the turn-signal environment.



(b) Snapshots of the agent using turn signals to make space and move to carpool lane.

Figure 7: Results on TrafficJamEnv.

7 Conclusion and Future Work

We presented three environments built on top of HighwayEnv which more realistically model traffic. We then trained agents on top of two of these environments and demonstrated that a combination of Q-learning with a simplified and approximated model-based method performs well.

There were several challenges we discovered while working on this project. Collisions in our environment are quite common, which leads to noisy model-free results. We suspect that training would be faster if we modeled the physics of the other cars explicitly in the model. We also realized that influence is an important part of lane-changing, but the environment in its current form is unable to model influence without the other agents learning simultaneously. Ultimately, this is part of the biggest challenge we encountered: we need more realistic modeling of the other agents on the road.

As mentioned in the introduction, we aim to continue this work with a goal of publication. As part of that, we would like to explore how to effectively influence other drivers. Xie et al.[14] demonstrated an effective method for learning to influence across multiple trials, but for a driving scenario, influence must function continuously while driving as well. We are also curious to see the emergent influence strategies developed by multiple agents on the road with turn signals. Other work on emergent behavior has seen unexpected strategies develop in multi-agent games [1]. We are also interested in exploring how well these policies generalize between different road conditions. Ideally, we'd be able to train in one type of traffic and see good performance in unseen conditions, a necessary trait for safe

autonomous driving. And finally, we are interested in exploring what is necessary for a more realistic and sophisticated simulator, such as a 3D simulator or a simulator that requires continuous control.

References

[1] Bowen Baker, I. Kanitscheider, T. Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *ArXiv*, abs/1909.07528, 2020.

[2] R. Chandru, Y. Selvaraj, M. Brännström, R. Kianfar, and N. Murgovski. Safe autonomous lane changes in dense traffic. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2017.

[3] Berthold Färber. *Communication and Communication Problems Between Autonomous Vehicles and Human Drivers*, pages 125–144. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.

[4] C. Hoel, K. Wolff, and L. Laine. Automated speed and lane change decision making using deep reinforcement learning. *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2148–2155, 2018.

[5] Arne Kesting, Martin Treiber, and Dirk Helbing. General lane-changing model mobil for car-following models. *Transportation Research Record*, 1999(1):86–94, 2007.

[6] Edouard Leurent. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

[7] W. Ma, De-An Huang, N. Lee, and Kris M. Kitani. Forecasting interactive dynamics of pedestrians with fictitious play. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4636–4644, 2017.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, Andrei A. Rusu, J. Veness, Marc G. Bellemare, A. Graves, Martin A. Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, S. Petersen, C. Beattie, A. Sadik, Ioannis Antonoglou, H. King, D. Kumaran, Daan Wierstra, S. Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[10] Sriram N N, Buyu Liu, F. Pittaluga, and M. Chandraker. Smart: Simultaneous multi-agent recurrent trajectory prediction. *ArXiv*, abs/2007.13078, 2020.

[11] D. Sadigh, S. Sastry, S. Seshia, and Anca D. Dragan. Planning for autonomous cars that leverage effects on human actions. In *Robotics: Science and Systems*, 2016.

[12] Michael J. Smith. The stability of a dynamic model of traffic assignment—an application of a method of lyapunov. *Transportation Science*, 18(3):245–252, 1984.

[13] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[14] Annie Xie, Dylan P. Losey, R. Tolsma, Chelsea Finn, and D. Sadigh. Learning latent representations to influence multi-agent interaction. *ArXiv*, abs/2011.06619, 2020.

[15] Brian D. Ziebart, Nathan D. Ratliff, G. Gallagher, C. Mertz, K. M. Peterson, J. Bagnell, M. Hebert, Anind K. Dey, and S. Srinivasa. Planning-based prediction for pedestrians. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3931–3936, 2009.