Motion Planning in Dense Traffic

Hung-Tien Huang Shije Wang Timothy Cale Workman

April 14, 2022

Abstract

A major challenge in autonomous driving is anticipating the actions of other drivers. In dense traffic scenarios, this presents many unknowns. To navigate safely, an autonomous vehicle must reason about the likely actions of other drivers as well as anticipate how those drivers may react to its own actions. In this work, an off-the-shelf Python package is used to model typical highway traffic scenarios. The deep Q reinforcement learning algorithm is applied to enable autonomous vehicle make decisions on when to change lane. It is shown that our method performs significantly better than random uniform policy.

1 Introduction

Driving in real world dense traffic scenarios remains a challenging task for autonomous vehicles as actions depend on nearby traffic participants [1]. For an autonomous vehicle to merge from the on-ramp to the main lane, the vehicle must first reason whether nearby vehicles are willing to yield or not, then interact with multiple road-users to induce them to yield, and finally change the lane when a safe margin is available [2].

Traditional motion planning methods can be classified into four groups according to their implementations: graph search, sampling, interpolating, and optimization [3]. These methods do not take nearby traffic into consideration and hence often fail in dense traffic [2]. Probability models are also used to take the dynamics between traffic participants into consideration [4, 5, 6]. However, these methods do not scale due to the curse of dimensionality [2]. Reinforcement learning (RL) has the capability to avoid the curse of dimensionality. The method introduced by Saxena et al. that uses RL does not provide safety guarantees. The one proposed by Bouton et al. that provides safety guarantees is behaving too conservatively.

To address the issue, the Interactive Model Predictive Controller (IntMPC) framework is proposed by Brito, Agarwal, and Alonso-Mora. A RL interaction-aware policy is learned and used to provide a velocity reference at the merging point to a low-level controller based on the location and velocity of itself and nearby vehicles. Given the velocity reference, the low-level controller generates a local optimal control command that follows a reference path while satisfying kino-dynamics and collision avoidance constraints. The reference path could be provided by a global path planner.[1] The behavior of other road participants are modeled by the Predictive Intelligent Driver Model (P-IDM). Simulation results demonstrate that IntMPC triggers interactive negotiating behavior to reason about the other drivers' cooperation and exploit their cooperativeness to induce them to yield while remaining safe.

In this work, an autonomous vehicle is set to travel down the highway indefinitely. The interaction-aware policy module in the IntMPC framework is modified to answer the question of when to change lane so that the distance travelled can be maximized. Instead of providing velocity reference, the RL agent only makes decisions on when to change the lane according to both the location and velocity of itself and nearby vehicles. Since the MDP only has discrete action space, the deep Q-learning algorithm is selected to solve the Markov Decision Process (MDP) [9]. Additionally, the concept of curriculum RL is used to train the RL agent [10]. A two-stage curriculum is set up to train the RL agent. At the first stage, the RL agent will learn to drive down the highway without collision. At the second stage, the RL agent will learn how to drive down the highway without collision while maximizing the distance travelled.

Preliminary results showed that the proposed deep Q-learning agent performed significantly better than the uniform random policy agent. However, the number of collision increases in the second stage as the agent get rewarded for travelling faster.

[1] The decision problem in automated vehicles can be roughly classified into three categories: global planning, behavioral planning, and local planning. Global planner answers the question of which roads one should take in order to go from point A to point B in a city. Behavioral planner make sure that the traffic rule is followed. Local planner provides the throttle, brake, and steering control input. Interested reader could refer to [3] for detailed explanations.

2 Approach

In this work, the open-source Python package highway-env is used to simulate the highway driving scenarios [11]. The Intelligent Driver Model (IDM) implemented in highway-env is used to simulate the behavior of all other vehicles except the ego-vehicle [12]. The ego-vehicle is instructed to drive down the highway indefinitely. The interactive planner of the ego-vehicle is responsible for deciding when and which lane the ego-vehicle should change to. The interactive planner is formulated as a Markov Decision Process (MDP) and the deep Q-learning along with the curriculum learning framework are used to solve the problem [9, 10].

2.1 Vehicle Representation

Consider a set $\mathcal{X}$ of n vehicles in a dense traffic scenario comprising an autonomous vehicle (ego-vehicle) and $n-1$ human drivers refer to as other vehicles. For each vehicle in $\mathcal{X}$ at a given time step k, the state of the i^{th} vehicle is represented as $\mathbf{s}_k^i = \langle x_k, y_k, \phi_k, v_k \rangle^\intercal, \forall i \in [0, n-1]^\intercal \cap \mathbb{Z}$, where x_k and y_k are the location of i^{th} vehicle in the Cartesian coordinate with respect to the global inertial frame, and ϕ_k and v_k are the heading and speed of the i^{th} vehicle. The ego-vehicle is set to be the 0^{th} vehicle in the $\mathcal{X}$.

2.2 Leader and Follower

The leader is defined as the nearest vehicle in front of the ego-vehicle and the follower as the nearest vehicle behind it. Both leader and follower may be in lanes adjacent to the ego-vehicle. The state of the leader and follower vehicles in lane $i, \forall i \in [0, 2] \cap \mathbb{Z}$, at time step k are denoted as $\mathbf{s}_{i,k}^l$ and $\mathbf{s}_{i,k}^f$, where the lane that the ego-vehicle is currently in is defined to be $i = 0$.

2.3 Markov Decision Process

The objective of the interactive planner is to allow the ego-vehicle to travel the maximum distance without collision before the episode timeout occurs.

State

Given the state of the leader $\mathbf{s}_{i,k}^l$ and follower $\mathbf{s}_{i,k}^f$ at line i, the neighbor vehicles state $\mathbf{N}_i$ with respect to the inertial frame of the ego-vehicle is denoted as:

$$\mathbf{N}_{i,k} = \langle \mathbf{s}_{i,k}^l - \mathbf{s}_{i,k}^0, \mathbf{s}_{i,k}^f - \mathbf{s}_{i,k}^0, \rangle, \forall i \in [0, 1, 2] \cap \mathbb{Z} \tag{1}$$

The observation $\mathbf{O}_k$ available to the RL agent at time step k is defined as:

$$\mathbf{O}_k = \begin{bmatrix} \mathbf{s}_k^0 & \mathbf{N}_{0,k} & \mathbf{N}_{1,k} & \mathbf{N}_{2,k} \end{bmatrix} \tag{2}$$

Note that the states of the MDP is defined to be the observation $\mathcal{O}$ that only includes the state of the ego-vehicle and 6 other neighbor vehicles instead of the entire state space $\mathcal{S}$ that contains the states of all vehicle.

Reward

The reward of the MDP is as same as the one defined by highway-env [11] and is engineered to encourage the ego-vehicle to make a decision that is fast and collision-free. For any time step k, the reward is defined as:

$$R_k(s, a) = \alpha \frac{v - v_{\min}}{v_{\max} - v_{\min}} - \beta \tag{3}$$

, where α controls the importance of travelling faster and β is the collision penalty.

Policy and Action

The action a is selected from the action space $\mathcal{A}$ that consists of one of the following action: merge left, merge right, speed up, slow down, and idle. The policy π is a stochastic discrete probability distribution of the action $\mathbf{A}$ given the observation $\mathbf{O}$, and such probability distribution is assumed to be a multinomial distribution.

$$\mathbf{A}_k \mid \mathbf{o}_k \sim \text{Multinomial}\,(1,\ \mathbf{p}) \tag{4a}$$

$$\mathbf{p} = \langle p_{\text{merge left}}, p_{\text{merge right}}, p_{\text{faster}}, p_{\text{slower}}, p_{\text{stay put}} \rangle^\intercal \tag{4b}$$

$$\pi(a_k \mid \mathbf{o}_k) = p(a_k \mid \mathbf{o}_k), \forall a_k \in \mathcal{A} \tag{4c}$$

2.4 Deep Q-Learning

Deep Q-Learning or deep Q-Network (DQN) is a variant of classical model-free Q-learning RL algorithm. For classical Q-learning, a table that maps a state and action pair to a value that represents the sum of future reward is constructed. However, such method requires both the state space and action space to be discrete. Furthermore, such method does not scale as the MDP becomes more complicated. To tackle the issue and enable continuous state space, function approximators are used to replace the table that maps state and action pairs to values. When a nonlinear function approximator such as neural network is used, the RL algorithms are known to be unstable or even diverge. Hence, the deep Q-network is proposed to address the issue by introducing a replay buffer so that the correlations between observation sequence and smoothing over changes in the data distribution [9]. For detailed explanation to DQN, reader should consult the work by Mnih et al. [9].

In this work, the MDP has a continuous state space and a discrete action space. Therefore, DQN is selected to solve the MDP.

2.5 Training Curriculum

A curriculum is a general concept that encompasses both schedules for organizing experiences and schedules for acquiring experience by training on tasks [10]. In RL, a curriculum serves to sort the experience an agent acquires over time in order to accelerate or improve learning [10]. In this work, a two-stage curriculum is set up to guide the agent to drive on highway without collision while maximize the distance within given time. This is achieved by manipulating the parameters for reward signal.

Stage 1

At this stage, the RL agent learns how to drive down the highway indefinitely without colliding with neighboring vehicles. This is achieved by setting α term in the reward signal to zero, so that there is no reward for travelling fast.

Stage 2

At this stage, the RL agent not only has to drive down the highway free of collision but also has to maximize the distance travelled within the given time. This is achieved by setting the α term in the reward signal to non-zero value. In the meantime, the collision penalty β is increased to effectively penalize the agent for going further by bumping into other vehicles.

2.6 Evaluation

The RL agents will be evaluated with the following evaluation metrics:

- Distance Traveled: One objective of the RL agent is to generate a sequence of decisions that gets the ego-vehicle to the destination in the shortest amount of time. Therefore, the faster the ego-vehicle is driving, the longer the distance the ego-vehicle could travel. Higher is better.
- Collisions: The percentage episodes terminated due to collision among all the episodes being tested. Lower is better.
- Steps Until Crash: For all the episodes that are terminated due to collision, the number of steps the ego-vehicle took until collision. Higher is better.

3 Experimental Setup

In this work, the agent is to run on a four-lane highway with 50 neighboring vehicles randomly initialized on the highway. In the first stage of the curriculum, the penalty β in the reward signal is set to 5. In the second stage, the α term for encouraging faster speed is set to 3 and the β term is increased to 10. The agent will only be rewarded when the speed is within the range of 20 and 30 miles per hour. The architecture of the deep Q-network is presented in Table 1.

Table 1: Deep Q-Network Architecture

	Input Shape	Output Shape	Activation
Layer 1	(n, 28)	(n, 100)	ReLu
Layer 2	(n, 100)	(n, 100)	ReLu
Layer 3	(n, 100)	(n, 100)	ReLu
Layer 4	(n, 100)	(n, 100)	ReLu
Layer 5	(n, 100)	(n, 4)	None

4 Result

To evaluate the proposed method, all the policies are evaluated for 1000 episodes. The performance of the random uniform policy and the DQN policies are reported. In addition, the learning curves in terms of different evaluation metrics are also reported.

4.1 Random Uniform Policy

The random policy is defined as choosing one of the action in the action space randomly with equal probability. Out of 1000 episodes, 910 episodes terminated due to collision with other vehicles. For the episodes that terminated with collision, the random policy crashes the vehicle after making 10 decisions on average with standard deviation of 6.91. The average total distance travelled is 308 meters with standard deviation of 196.73 meters.

4.2 DQN Policy

The DQN policy is trained from a 2-stage curriculum. The stage 1 policy is the one trained from the first stage of the curriculum. The stage 2 policy is the one trained from the second stage of the curriculum. The learning curves are shown in Figure 1.

Stage 1 Policy

Out of 1000 episodes, 127 episodes terminated due to collision with other vehicles. For the episodes that terminated with collision, the DQN stage 1 policy crashes the vehicle after making 15 decisions on average with standard deviation of 10.89. The average total distance travelled is 1233.93 meters with standard deviation of 10.89 meters.

Stage 2 Policy

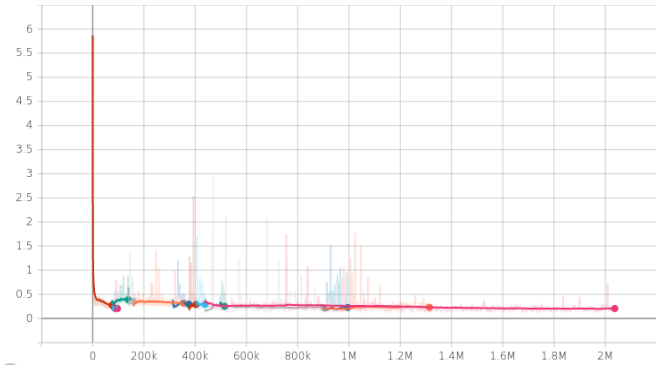
Out of 1000 episodes, 312 episodes terminated due to collision with other vehicles. For the episodes that terminated with collision, the DQN stage 2 policy crashes the vehicle after making 31.4 decisions on average with standard deviation of 18.55. The average total distance travelled is 1471.27 meters with standard deviation of 423.61 meters.

5 Discussion

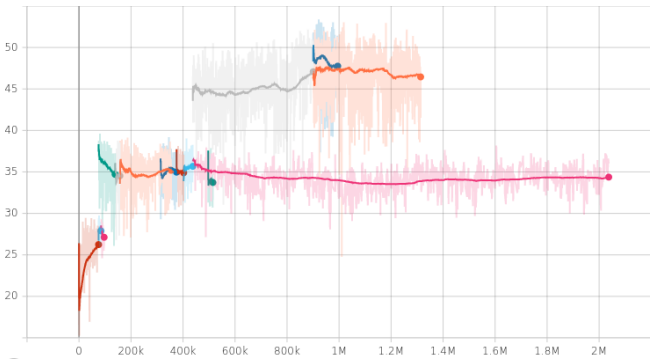
It is observed that the DQN policy performed significantly better than the uniform random policy, in terms of number of collisions and total distance travelled. It takes slightly longer for DQN policy to collide with other vehicles compared to uniform random policy.

The application of curriculum RL does not perform as expected. As shown in Figure 1d, the DQN agent indeed learns to travel faster when it is trained on stage 2 of the curriculum. However, the frequency of the DQN agent collide with other vehicles also increased by 3 times comparing to that of in stage 1 as shown in Figure 1c. The reward signal experienced a significant boost after 400,000 optimization steps since the training of stage 2 curriculum begins as shown in Figure 1b. In general, it is observed that the learning curves shown in Figure 1 are noisy except for training loss.

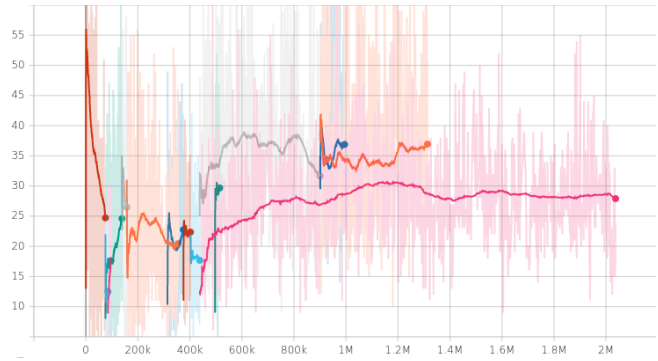
Based on the observations, the authors rule out the possibility of learning rate of numerical optimizer being too large since the training loss is converging. Instead, the authors the reward signal in the second stage of the curriculum needs to be revised and will be the subject of future investigation of this work.



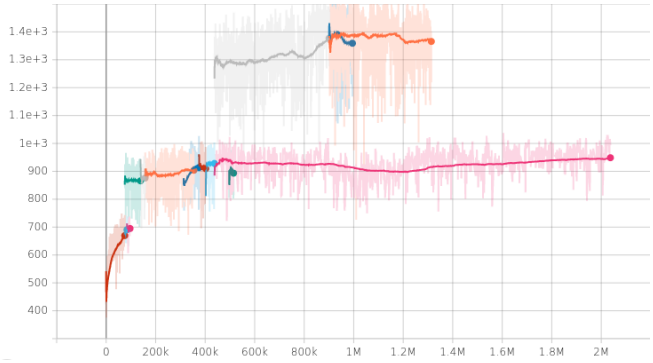
(a) Training Loss v.s. Optimization Steps



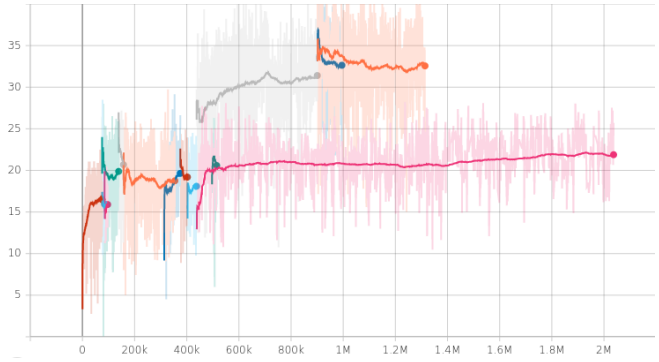
(b) Reward v.s. Optimization Steps



(c) Number of Crashes v.s. Optimization Steps



(d) Distance [m] v.s. Optimization Steps



(e) Steps to Collision v.s. Optimization Steps

Figure 1: The learning curves. The learning curves are non-continuous due to two reasons. First, technical issues with Python multiprocessing are encountered that cause the training program to crash for unknown reasons after long execution time. Second, the policy is trained with 2-stage curriculum, so boost in certain metrics are expected. The transition from stage 1 to stage 2 happens at around 400k steps.

6 Conclusion

In this work, a DQN RL agent is trained with a two-stage curriculum to travel down the highway as fast as possible without collisions. The RL agent takes action based on its own position and velocity as well as the location and velocity of its surrounding vehicles. Preliminary result shows that the DQN agent performs significantly better than that of uniform random policy. However, the second stage of the curriculum increased the distance travelled, but the number of collision also increased as a side effect, which is not desired. The author suspects that the reward signal for the second stage needs to be modified in order to tackle the issue.

References

[1] Simon Ulbrich et al. "Structuring Cooperative Behavior Planning Implementations for Automated Driving". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. 2015, pp. 2159–2165. DOI: 10.1109/ITSC.2015.349.

[2] Bruno Brito, Achin Agarwal, and Javier Alonso-Mora. "Learning Interaction-aware Guidance Policies for Motion Planning in Dense Traffic Scenarios". In: (July 2021). arXiv: 2107.04538 [cs.RO].

[3] David González et al. "A Review of Motion Planning Techniques for Automated Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 17.4 (2016), pp. 1135–1145. DOI: 10.1109/TITS.2015.2498841.

[4] Edward Schmerling et al. "Multimodal Probabilistic Model-Based Planning for Human-Robot Interaction". In: (Oct. 2017). arXiv: 1710.09483 [cs.RO].

[5] Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. "LUCIDGames: Online Unscented Inverse Dynamic Games for Adaptive Trajectory Prediction and Planning". In: (Nov. 2020). arXiv: 2011.08152 [cs.RO].

[6] Pete Trautman. "Sparse Interacting Gaussian Processes: Efficiency and Optimality Theorems of Autonomous Crowd Navigation". In: (May 2017). arXiv: 1705.03639 [cs.RO].

[7] Dhruv Mauria Saxena et al. "Driving in Dense Traffic with Model-Free Reinforcement Learning". In: (Sept. 2019). DOI: 10.1109/ICRA40945.2020.9197132. arXiv: 1909.06710 [cs.RO].

[8] Maxime Bouton et al. "Reinforcement Learning with Probabilistic Guarantees for Autonomous Driving". In: *Workshop on Safety Risk and Uncertainty in Reinforcement Learning, Conference on Uncertainty in Artificial Intelligence (UAI), 2018* (Apr. 2019). arXiv: 1904.07189 [cs.RO].

[9] Volodymyr Mnih et al. "Playing Atari with Deep Reinforcement Learning". In: (Dec. 2013). arXiv: 1312.5602 [cs.LG].

[10] Sanmit Narvekar et al. "Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey". In: *Journal of Machine Learning Research 21(181):1-50, 2020* (Mar. 2020). arXiv: 2003.04960 [cs.LG].

[11] Edouard Leurent. *An Environment for Autonomous Driving Decision-Making*. https://github.com/eleurent/highway-env. 2018.

[12] Edouard Leurent and Jean Mercat. "Social Attention for Autonomous Decision-Making in Dense Traffic". In: (Nov. 2019). arXiv: 1911.12250 [cs.LG].