Investigation of Near-accident Car-driving Scenario using Deep Imitation Learning and Reinforcement Learning

Team Members: Wentao Zhong Jiaqiao Zhang Project Advisor: Erdem Bıyık

May 2020

1 Abstract

According to [1], the road traffic can be divided into three categories in terms of human driver's responsibility: navigation, guidance and control. In this work, we focus on the guidance level in the high risk scenarios, which is responsible for the output of desired trajectory and/or speed. First, We take the approach of implementing deep imitation learning to obtain the driver agent model using data generated by predefined dominant control law. Then, reinforcement learning is applied to find the policy in high risk scenarios via switching control model considering both efficiency and safety.

2 Introduction

The autonomous driving technology grows rapidly recent years. However, the high risk scenario, where a potential accident is likely to happen, could not be tackled well since the action need to be changed correspondingly with other drivers. Hence, the agent may need to change actions significantly to stay safe.

According to the recent studies [2] [3], the reinforcement learning (**RL**) and imitation learning (**IL**) are two dominant approaches for autonomous driving in learning actions. Reinforcement learning is applied to learn the driving policies which maximize the reward function and imitation learning tries to learn the behavior from expert, i.e. behaviour cloning. One shortcome of RL is that it need to fully explore the environment while the IL requires large amount of expert demonstrations data. Both approaches are not suitable for rapid transition of action since the action learned is continuous. Our approach intend to combine RL and IL approaches to allow the driving agent switch actions accordingly to achieve safety requirement in near-accident scenario.

First, we obtain driver agent models using deep imitation learning. The input to our algorithm is the dataset generated from CARLO simulation containing the 4-D observation (location of the ego vehicle, velocity of the ego vehicle, location of other vehicle with noise and velocity of other vehicle with noise), control input (steering and throttle) and driving mode indicator. We then use Conditional Imitation Learning (CoIL) [4] to output a predicted control input given observation and driving mode indicator.

The basic environment setup used is CARLO [4] to perform 2D driving simulation. Two different high risk scenarios including cross traffic and wrong direction are tested. Different driving modes are evaluated based on completion time and collision rate.

3 Related work

Imitation learning (**IL**) is one of the popular methods. Muller implement behavior cloning IL to solve off-road obstacle avoidance [5]. The algorithm learns driving policy which consist of state-action pairs from the dataset. One drawback is they suffer from the generalization of unpredicated behavior subject to new test domain. Also it requires a huge amount of expert demonstrations and lead to low data efficiency. [6] Codevilla proposed a method called Conditional Imitation Learning (**CoIL**) which extends IL with high-level commands, as shown in Figure 1 [7]. It learns separate IL models for each high-level commands and some features are shared between different learned IL models. It improves data efficiency but it requires high level commands at test time. Our approach proproses to solve this problem using reinforcement learning to learn agent providing the high-level commands instead of using commands provided from drivers.

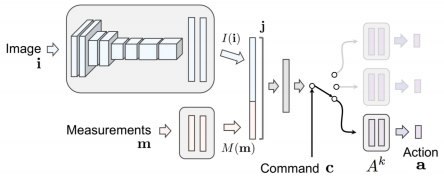


Figure 1: Network architecture of CoIL

Reinforcement learning (**RL**) is one main approach applied in autonomous driving [2]. It explores the environment first and then take actions in each state which maximize the pre-defined reward. One short come is that the state space in driving scenario is very large, which makes it hard to fully explore. According to [8], Hierarchical Reinforcement Learning is proposed to solve this problem. It consist of multiple layers. The higher layer acts like a manager to give goal for lower layers and lower layer acts like worker to achieve the goal. It improves the exploration efficiency. Finally, Nair extend this Hierarchical RL to use expert demonstrations to obtain the high level commands for the exploration of RL [9]. However, all the algorithms could not address the problem at near-accident scenario since it is difficult to design the low-level reward function for Hierarchical RL. Instead of using RL to learn the low-level policy, our approach first use IL to obtain the low-level policy. Then we use RL to obtain the high-level commands.

4 Dataset and Features

The dataset we used for the low-level IL training is obtained from the CARLO simulation for two different driving scenarios [4]. For the first cross traffic scenario as shown in Fig 2a, we collect the expert demonstrations from three different driving modes (timid, aggressive, normal) in simulation. The control policies for the ego car for different driving modes are pre-defined based on time to collision (TTC) and time to entering intersection of both ego and ado cars according to [10]. The dataset for scenario 1 contains observations (including location of the ego vehicle, velocity of the ego vehicle, location of other vehicle with noise and velocity of other vehicle with noise); action (throttle) from the control policies and one indicator which indicates the specific driving mode.

For the second wrong direction scenario as shown in Fig 2b, the expert demonstration data is generated similarly using two different driving modes (timid, aggressive) in simulation. The observation space is 5D with additional observation of ado vehicle location. The control policy for the two driving modes are defined similarly as in scenario 1. Except for TTC, we calculate a predicted collision point based on ado car's heading, location and velocity. The composition of this dataset is similar to the dataset of scenario 1. The steering control is set to be zero since the ego car is moving in a lane. The vehicle is assumed to be a point mass in the simulation. Roughly 85% data is used for training examples and 10% data is used for validation examples and 5% data is used for test examples for the CoIL training.

Implementation code can be found at `https://github.com/JiaqiaoZhang/CS229_Final_Project`

5 Methods

The main method is firstly trying to learn different driving mode using CoIL from the dataset described before. We defined three main driving modes according to [10], timid mode, normal mode and aggressive mode. These three modes are defined by evaluating the driving risk which is based on time for the following car to reach the preceding car and predicted time to collision. The timid driving mode have low driving risk and it tends to keep low velocity and large gap with other car. The aggressive driving mode has high driving risk and tends to keep high velocity and small gap with other cars. The normal driving mode behaves neutrally between these two modes.

Deep neural network is used to learn the driving model using Conditional Imitation Learning (CoIL). According to [7], the input to the neural network is the observation of the vehicle in the simulated scenario. The output is the action (steering and throttle). An additional command which is used to determine the driving mode is provided. CoIL allow us to train models which can be switched given the high-level commands.

Then the high-level agent which gives the commands to switch the different low-level agent obtained from CoIL before is trained using Proximal Policy Optimization (PPO) RL algorithm [11]. PPO alternate between sampling data through interaction with the environment, and optimizing a novel objective function according to [11]. The novel objective achieves a way to do a Trust Region update which is compatible with Stochastic Gradient Descent, and simplifies the algorithm by removing the KL penalty and need to make adaptive updates. The environment used to train RL is CARLO for two different scenarios.

6 Experiments/Results/Discussion

The preliminary experiments are done using the scenario proposed in [4]. The experiments are conducted in two different scenarios:

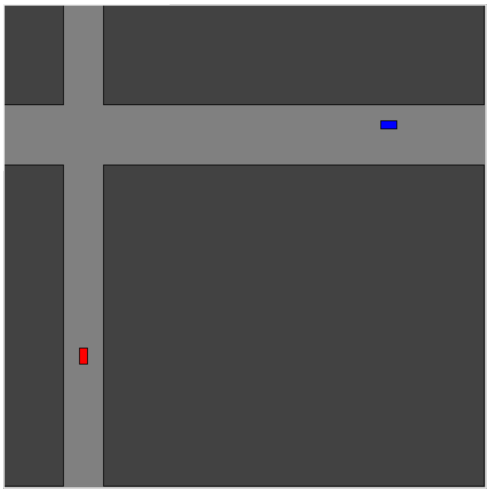
(1) the first scenario is characterized by a ego car which is approaching a crossroad while ado car (simulated by computer using pre-defined control law) is also approaching (figure 2a), which is called the intersection scenario. For this scenario, there are overall three control mode dominating ago car: aggressive, normal and easy. In the aggressive mode, the ado will perform in a way that will generally have higher speed and are more likely to collide with ego car, while in the easy mode the collision is prevented. The normal mode is designed in a way that the completion time and collision rate are both between aggressive mode and normal mode. All three mode are hard coded control laws.
(2) The second scenario is simulating the situation where the ado car is driving in the opposite direction towards the ego car, as shown in figure 2b. For this scenario, there are only two mode: aggressive and timid.

The simulator used in this experiment is CARLO, a customized 2D driving simulator that involves simple dynamic model and visualizations. CARLO can handle 2D simulation with fast implementation in perception and measurement data.

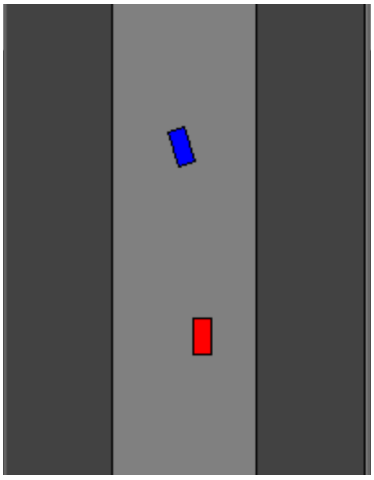
We assume a point-mass dynamic model in this scenario, and no other obstacle in the scenario. Considering safety and efficiency, a test is defined as success when ego car reaches the target under a certain amount of time without colliding into the ago car or other environment. The data we used are composed of two parts: first part is provided by Erdem Bıyık, which is associated with the paper [1]; second part is generated by hard coded control laws in the gym environment.

For both scenarios, the observations are: ego car's location and velocity, ado car's location and velocity. The only difference between scenario 1 and 2 is in scenario 1, the dimension of ego car's location is one (because ego car only drives in a straight line), while in scenario 2, the ego car's dimension is two. The observation along with higher level command (timid or aggressive) is fed into neural network to obtain a policy that minimizes the loss function defined by the difference between the ego car and expert behaviors.

Implementation code can be found at `https://github.com/JiaqiaoZhang/CS229_Final_Project`

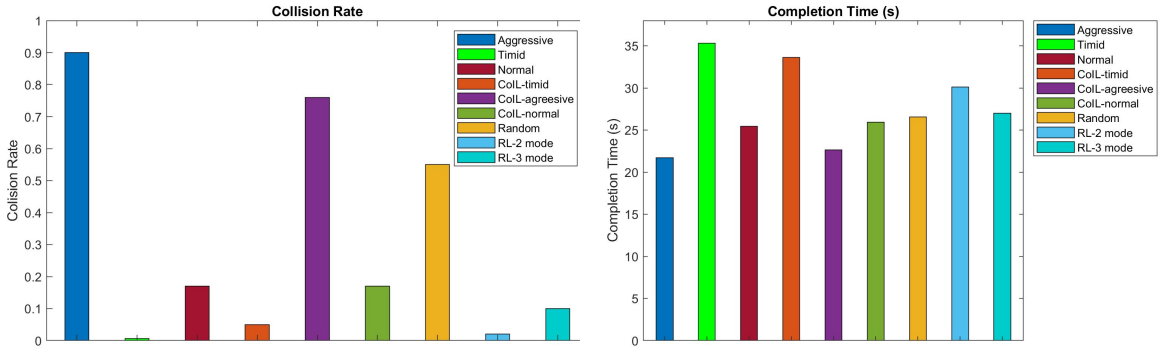


(a) Scenario1 (ego car: red, ado car: blue)



(b) Scenario2 (ego car: red, ado car: blue)

Figure 2: Scenario illustration



(a) Scenario 1 collision rate

(b) Scenario 1 completion rate

Figure 3: Scenario 1 experiments results

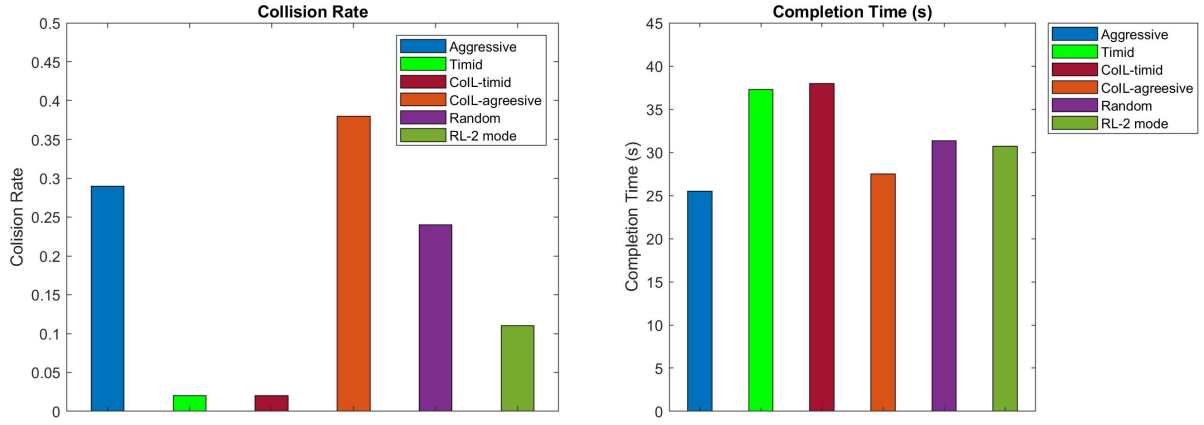
The results of scenario 1 is shown in Figure 2a. As for now, for conditional imitation learning, we have three results: "COIL-aggressive", "COIL-middle" and "COIL-timid". "Aggressive", "Timid" and "Normal" stand for the testing result of hard coded control policies before CoIL and RL. "Random" shows the result of policy combining three control modes randomly. Finally, "RL-2 mode" and "RL-3 mode" are the results of using reinforcement learning when there are two control modes for selection (aggressive and timid), and there are three control modes for selection (aggressive, middle and timid).

First, let's compare the results of imitation learning and hard coded policy. In terms of the collision rate, there is not much difference between normal and CoIL-normal or timid and CoIL-timid. The collision rate for aggressive is 0.9 while for CoIL-aggressive is 0.24. When it comes to the completion time, the trend is exactly on the opposite side. This is partly because if the ego car wants to behave aggressively, it will have a comparatively higher speed, which will result in less time and a higher collision rate. It is no surprising that the timid mode has the longest completion time, while the aggressive mode has the lowest completion time.

A random policy also is included for later comparison with results from reinforcement learning. It can be seen that the collision rate and completion time are in the middle of three hard code policy, which makes sense because random policy chooses three modes with equal probability, and the model is closely related to the ego car action, which is throttle in this case.

Implementation code can be found at `https://github.com/JiaqiaoZhang/CS229_Final_Project`

Finally, results of RL-2 mode and RL-3 mode show that the reinforcement learning actually achieves a good balance in choosing modes. Both RL results have a lower completion time and lower collision rate than the random one, which shows that RL policies can effectively choose the modes according to the observation. Although RL's collision rate is not as low as the timid one, the completion time has reduced by 5.21s and 8.32s respectively. It worth to mention that the result of RL is closely related to the penalty/reward when there is a collision. This finding also verifies the results provided by the paper [4]. What's more is the difference between choosing from 2 driving modes and 3 driving modes. From our study, we find that there is a obvious improvement of the completion time between 2 and 3 driving modes, while the collision rate of 3 driving mode is 0.1, which is higher than 2 driving mode: 0.02. This finding is interesting because it suggests that first training with more driving mode. However, this study only compares the results from 2 to 3 driving mode, whether more modes (greater than 3) will contributes to lower completion time is still remain to be discovered. Also, all those results are based on 2D simulation CARLO and point mass physical model, more work should be done using real car kinematic and dynamic model, such as in CARLA environment [12].



(a) Scenario 2 collision rate

(b) Scenario 2 completion rate

Figure 4: Scenario 2 experiments results

The results of scenario 2 is shown in figure 4. For this scenario, we only train the RL policy in 2 driving mode. The overall performance of RL policy is similar to what we have seen in scenario 1. The completion time of RL-2 mode is about the same as the random policy, but the collision rate of RL-2 mode is much lower than random one (from 0.11 to 0.24).

7 Conclusion/Future Work

In summary, we can draw a conclusion that the approach - first using conditional imitation learning to learn driving model from expert, then training a high level policy using reinforcement learning does a great job in two scenarios. Although two scenarios shown in this project is rather simple and there are some assumptions in the simulation, this does shed light on the application of CoIL-RL in more complicated scenario where the motion planning of the vehicle is challenging due to the environment. However, there are still some work remain to be done. First, more high risk scenarios including halting car, merge, unprotected Turn can be used to evaluate the performance of different driving model and switching techniques. Second the simulation can be done in CARLA [12] where the physical model is more realistic. Finally, the expert data can be obtained from real drivers instead of hard coded policy.

Implementation code can be found at `https://github.com/JiaqiaoZhang/CS229_Final_Project`

8 Contributions

1. Test and write hard coded policy
2. Generate data for scenario 1 and 2
3. Train policy for each driving mode using imitation learning
4. Train policy combining 2 or 3 driving modes using CoIL
5. Train reinforcement learning using DQN and PPO
6. Test and evaluate the results.

All team members contributes equally to this project.

Implementation code can be found at `https://github.com/JiaqiaoZhang/CS229_Final_Project`

References

[1] Edmund Donges. A conceptual framework for active safety in road traffic. *Vehicle System Dynamics*, 32(2-3):113–128, 1999.

[2] Markus Wulfmeier, Dushyant Rao, Dominic Zeng Wang, Peter Ondruska, and Ingmar Posner. Large-scale cost function learning for path planning using deep inverse reinforcement learning. *The International Journal of Robotics Research*, 36(10):1073–1087, 2017.

[3] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Proceedings of the 1st International Conference on Neural Information Processing Systems*, NIPS'88, page 305–313, Cambridge, MA, USA, 1988. MIT Press.

[4] Zhangjie Cao, Erdem Biyik, Woodrow Z. Wang, Allan Raventos, Adrien Gaidon, Guy Rosman, and Dorsa Sadigh. Reinforcement learning based control of imitative policies for near-accident driving. In *Science and Systems (RSS)*, July 2020.

[5] Urs Muller, Jan Ben, Eric Cosatto, Beat Flepp, and Yann L. Cun. Off-road obstacle avoidance through end-to-end learning. In Y. Weiss, B. Schölkopf, and J. C. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 739–746. MIT Press, 2006.

[6] Felipe Codevilla, Matthias Müller, Alexey Dosovitskiy, Antonio López, and Vladlen Koltun. End-to-end driving via conditional imitation learning. *CoRR*, abs/1710.02410, 2017.

[7] Felipe Codevilla, Matthias Muller, Antonio Lopez, Vladlen Koltun, and Alexey Dosovitskiy. End-to-end driving via conditional imitation learning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[8] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3675–3683. Curran Associates, Inc., 2016.

[9] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. *CoRR*, abs/1709.10089, 2017.

[10] Qingwen Xue, Ke Wang, Jian Lu, and Yujie Liu. Rapid driving style recognition in car-following using machine learning and vehicle trajectory data. *Journal of Advanced Transportation*, 2019:1–11, 01 2019.

[11] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[12] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

Implementation code can be found at `https://github.com/JiaqiaoZhang/CS229_Final_Project`