Offline Reinforcement Learning for Autonomous Driving

EECS 545 Term Project, Winter 2024

Shitanshu Bhushan
University of Michigan
sbhushan@umich.edu

Kuang-Yang Cheng
University of Michigan
ykycheng@umich.edu

Yuxing Liu
University of Michigan
dylanliu@umich.edu

Aakash Puntambekar
University of Michigan
apuntamb@umich.edu

Abstract

Learning human driving behaviors and patterns is crucial for real-world autonomous driving applications. The two main challenges are (1) predicting the future trajectory of the self-driving car and (2) simulating the driver's behavior. To learn these complex tasks, we need to have accurate and representative data. In this project, one key insight is that we utilize a dataset with highly interactive driving scenarios. This leads to our primary objective of learning from human behavior. We investigate the use of offline model-based reinforcement learning (RL) as a different method for making trajectory decisions. First, we introduced offline RL methods and construct an offline RL dataset for the roundabout trajectory decisions on the real world driving dataset INTERACTION. Then, we design a reward function satisfying demands of the roundabout scenario. Further, we deploy conservative Q-learning algorithm and analyze its performance under test and augmented data. Finally, the discussions and conclusions are provided to analyzed the future goal and challenge for offline RL in autonomous driving.

1 Introduction

Autonomous driving has recently received a high degree of public attention, not all positive. Recent incidents involving General Motors' Cruise division and Tesla's autopilot system have been highlighted in the media and have shaken public confidence in autonomous technologies [(Domonoske, 2024)(Krisher, 2023)]. However, we believe that market demand for autonomous driving could potentially be very high if the technology is able to win over public opinion. Although passenger cars receive the bulk of attention, other sectors could also benefit from the technology. For example, over-the-road trucks that are autonomous would be able to meet tighter delivery deadlines as there is no human driver needing periodic breaks. While non-road and non-land vehicles stand to benefit from this technology as well (indeed the concept of autopilot has existed in aviation for decades), on-road vehicles pose challenges in route planning and obstacle avoidance that are unique. In order to build public confidence in on-road autonomous technologies, many avenues must be explored. We propose exploring offline reinforcement learning (RL) from human feedback as one of these avenues.

2 Proposed method

2.1 High Level Approach

At a high level, our approach will be to train an agent using a RL algorithm using some fraction of this data set. As per the paradigm of RL, we will assign a reward based on the similarity of the actions taken by our autonomous agent to the actions a human would take given the same scene. The remainder of the data will then be used to test our agent. Based on the INTERACTION dataset [(Zhan et al., 2019)], which contains data recorded from drones and traffic cameras, we can utilize the pre-recorded complex driving behaviors, such as negotiations and aggressive decisions, to perform offline RL for cloning behavior or predicting motion. Initially, we will focus on training a solitary conservative Q-learning (CQL) agent. Subsequently, we intend to expand our scope by training multiple CQL agents. During the inference phase, we'll aggregate the q-values of these agents for a specific state-action pair by computing their mean. We can then compare the results of both to see if our ensemble performs better. We will also then upload the results of our agent on the INTERACTION-Dataset-based PREdicTion (INTERPRET) Challenge to see how well our agent performs against existing methods.

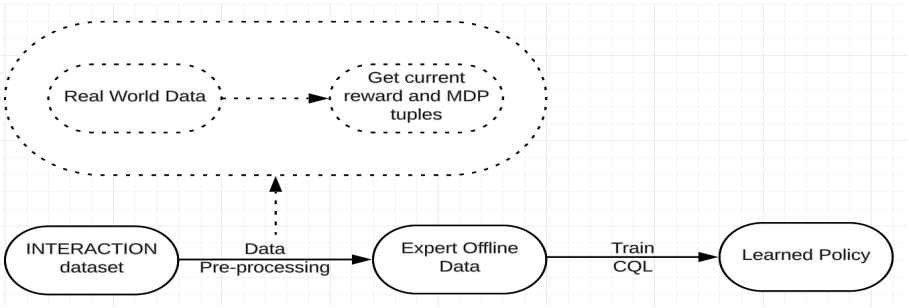

Figure 1: Pipeline of Method

2.2 Problem Formulation

RL is a Markov Decision Process (MDP), specified by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, r, \mu_0, \gamma)$. $\mathcal{S}$ defines the state space, $\mathcal{A}$ denotes the action space. $T(\mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t)$ is a conditional probability that describes the dynamics of the system and $r(\mathbf{s}, \mathbf{a})$ represents the reward function. $\mu_0(s)$ denotes the initial state distribution. $\gamma \in (0, 1)$ denotes the discount factor. We wish to learn a policy that maximizes return, denoted by

$$J(\pi) := \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

We must find this policy while only having access to an offline dataset of transitions collected using a behavior policy $\pi_\beta, \mathcal{D} = \left\{ \left(\mathbf{s}_t^i, \mathbf{a}_t^i, \mathbf{s}_{t+1}^i, r_t^i \right) \right\}$.

2.3 Data Preprocessing: Construct Offline RL Data

To learn real human driving behavior by applying offline RL, we have chosen to construct an offline RL dataset using the real-world driving dataset INTERACTION [(Zhan et al., 2019)] which has real driving trajectories with motion information, vehicle shape and time from complex urban scenarios. It is widely used for trajectory prediction research and applications. In this project, we considier the validation of decision and planning in the DR_USA_Roundabout_FT mapas show in Fig.2. In the following, we give the processing to construct the corresponding MDP tuple for offline dataset.

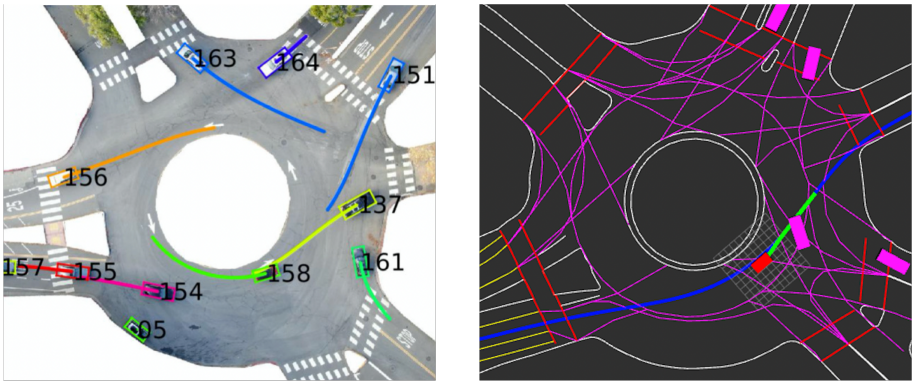

Figure 2: Example of the detection and tracking results in highly interactive roundabout driving scenario in the INTERACTION dataset. [(Zhan et al., 2019)]

State: We represent the current state of the ego vehicle using a 37-dimensional vector, denoted as $s = [ego\_id, interactions\_id, ego\_features, interactions\_features]$. The state vector s is segmented into four primary setctions, as shown in TABLE 1. ego_id indicates the presence of the ego vehicle and $interactions\_id$ denotes three nearest surrounding interactive agents, streamlining the computation process. $ego\_features$ encapsulates features of the ego vehicle, such as its dimensions, current and previous speed values, present coordinates, and projected coordinates for the next moment, assuming the ego vehicle continues on its current path and speed. The features of the three interactive agents represented by $interactions\_features$ also include the similar information as $ego\_features$. The subsequent state, s', is determined using the same methodology.

Action: We consider the ego vehicle's speed v_x and v_y in the next time step as its action a, because the trajectories form INTERACTION [(Zhan et al., 2019)] are human driving's data.

Table 1: State vector

Component	Description
ego_id	ego vehicle's id
$interactions\_id$	three interactions' id
$ego\_features$	ego vehicle's features
$interactions\_features$	three interactions' features

2.4 Reward Function for Trajectory Decisions

As described previously in this report, our chosen data set, INTERACTION [(Zhan et al., 2019)], is comprised of tabulated vehicle position data as a function of time. These data have been collected during times at which the vehicle is being driven by a human rather than by an autonomous agent. For our purposes, we seek to define a reward function based on the deviation of our RL agent's trajectory from a human's decisions given the same scenario. This reward function will be used in our Q-learning implementation and will contain several tunable hyperparameters.

We begin by describing the intuition behind our desired reward function. Consider a table of data that shows the trajectory of a vehicle as a function of time. Note that in the INTERACTION dataset [(Zhan et al., 2019)], the data is comprised of not just the vehicle we wish to train as an autonomous agent, but rather also the objects around the this subject vehicle, therefore we have available to us a rich set of data containing high-resolution (in time) recorded positions. Our reward function should be capable of rewarding the agent's capability to match both a human's position and velocity decisions, while penalizing heavily any collisions. Mathematically, we express this as follows:

$$R = R_p + R_v + R_C \tag{1}$$

where R is the total reward, R_p is the reward for the position match, R_v is the result for the velocity match and R_C is a penalty (that is, negative reward) for a collision with another object.

To expand each of the terms in Equation ((1)) further, we first need to define the notion of future trajectory length. Recall that the goal here is for an autonomous agent to plan a trajectory given a certain scenario, which is an instantaneous snapshot of the positions of the agent and its surrounding objects. Thus, a tunable hyperparameter is the number of timesteps S that will be planned.

We now define each of the terms individually as follows:

$$R_p = r_p X_{cl} F_\alpha \tag{2}$$

$$R_p = r_p\left(\frac{1}{\sigma\sqrt{(2\pi)}}\exp\left(-\frac{1}{2}\left(\frac{v - min(v_{max}, v_{cl})}{\sigma}\right)^2\right) - b_v\right) \tag{3}$$

$$R_C = r_C C \tag{4}$$

We now provide the intuition for each of the terms in equations (2) - (4). First, the tunable hyperparameters $r_p > 0$, $r_v > 0$ and $r_C < 0$ will affect the training of the RL agent.

R_p is the safety reward. We reward our agent for maintaining a safe distance from the objects around it. X_{cl} in (2) is the distance from our RL agent to the closest object in the interaction scene. This information comes directly from the INTERACTION data set. The angle factor F_α is a function of the angle between the path of our RL agent and the direction of the closest object. The F_α is set to 1 if the angle between these directions is $\pm$ 30° and zero otherwise. We calculate the angle itself using the law of cosines. Figure 3 shows the geometry of the scene and the distances we used to compute the angle α.

R_v is the velocity reward. This reward, as shown in Equation (3), takes the form of a Gaussian distribution centered around the maximum velocity. We define the distribution to be centered at the minimum of half the velocity it would take to reach the closest object in one time step (100 ms), denoted by v_{cl}, or the maximum velocity for the scene, v_{max}. The offset, b_v, enables us to make the reward negative far from the center, allowing for possible penalties if the velocity deviates from the desired center of the distribution. The offset is a tunable parameter,

R_C is the velocity reward, or rather, penalty. C contains information about whether the position of our RL agent coincides with any other object, indicating a collision. C is -1 if there is a collision and 0 otherwise. This can be amplified by the hyperparamerter r_C.

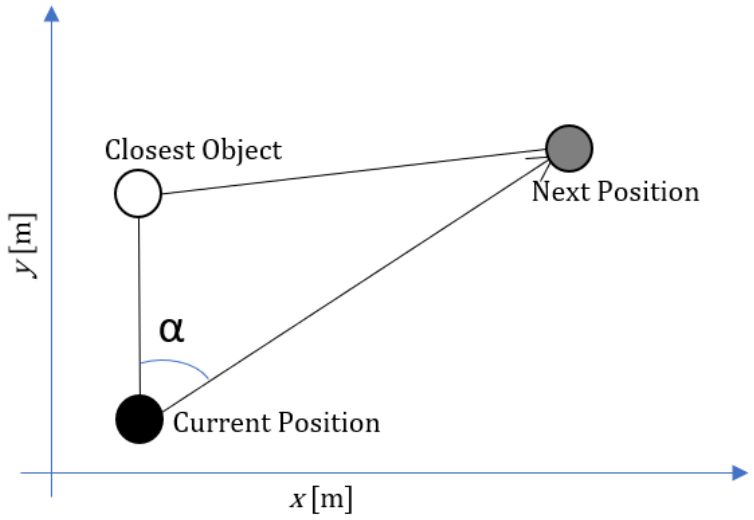

Figure 3: Visual intuition for computation of angle $\alpha as part of velocity reward component$

Substituting the details above into Equation ((1)), we obtain our as-implemented reward function:

$$R = r_p X_{cl} F_\alpha + r_p(\frac{1}{\sigma\sqrt{(2\pi)}} \exp{(-\frac{1}{2}(\frac{v-\bar{v}}{\sigma})^2)} + b_v) + r_C C \tag{5}$$

2.5 Replay Buffer

Replay buffer (also known as experience replay) is a replay memory technique where we store experience tuples, $\langle \mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}, r_t \rangle$ pooled over all timesteps and for all vehicles, in a dataset $\mathcal{D}$. During training, we sample experiences from this dataset which helps increase sample efficiency by allowing samples to be reused and also improves computational efficiency as we can do mini-batch updates[(Bruin et al., 2015)].

Replay buffer also helps improve learning by removing the issue of highly correlated samples for consecutive time-steps. By using a replay buffer, instead of training on consecutive car trajectories, we sample randomly from the buffer allowing the samples to be from many different and diverse trajectories.

2.6 Conservative Q-learning

We have Q function, which is the expected value of the total reward we will get if we start from state s_t and then take action a_t,

$$Q^\theta\left(\mathbf{s}_t, \mathbf{a}_t\right) = E\left[\sum_{t'=t}^{T} \gamma^{t'-t} r\left(\mathbf{s}_t, \mathbf{a}_t\right)\right] \tag{6}$$

Q functions follow Bellman equations,

$$Q^\theta\left(\mathbf{s}_t, \mathbf{a}_t\right) = r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma E\left[Q^\theta\left(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}\right)\right] \tag{7}$$

From the above, an optimal policy can be obtained by

$$\pi(a_t, s_t) = argmax_{a_t} Q^\theta(s_t, a_t) \tag{8}$$

This is the basis of Q-learning and the objective for training the Q function would be:

$$\min_{Q^\theta} E_{(\mathbf{s},\mathbf{a})\sim\pi_\beta(\mathbf{s},\mathbf{a})}\left[\left(Q^\theta(\mathbf{s}_t, \mathbf{a}_t) - \left(r\left(\mathbf{s}_t, \mathbf{a}_t\right) + \gamma E\left[Q^\theta\left(\mathbf{s}_{t+1}, \mathbf{a}_{t+1}\right)\right]\right)\right)^2\right] \tag{9}$$

In an offline learning setting, Q-learning faces an issue called distribution shift: while our function approximator is trained under one distribution, it will be evaluated on a different distribution[(Levine et al., 2020)]. In an online setting, this would be less of an issue as we could just learn from this out of distribution state but in an offline only setting, we cannot do the same and must entirely rely on static dataset.

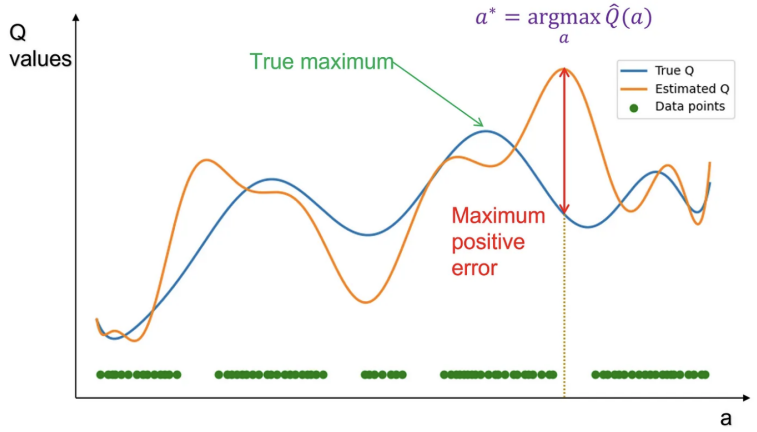

Figure 4: Distribution Shift in Q-learning[(Kapoutsis, 2022)]

From the figure we can see that, as in Q-learning we select π_{new} to maximise the expected value of Q, we end up with an erroneous over-estimation for these out of distribution data.

This gives us the motivation for CQL[(Kumar et al., 2020)], keep our standard Bellman error minimization term and add another term that explicitly seeks to find actions with high Q values and then minimize their value. We also add another term to make sure we don't end up being too conservative. This gives us the following objective:

$$\begin{aligned}\hat{Q}^{\pi} = \arg\min_{Q} \max_{\mu} \alpha E_{\mathbf{s}\sim D, \mathbf{a}\sim\mu(\mathbf{a}|\mathbf{s})}[Q(\mathbf{s},\mathbf{a})] - \alpha E_{(\mathbf{s},\mathbf{a})\sim D}[Q(\mathbf{s},\mathbf{a})] \\ + E_{(\mathbf{s},\mathbf{a},\mathbf{s}')\sim D}\left[\left(Q(\mathbf{s},\mathbf{a}) - (r(\mathbf{s},\mathbf{a}) + E_{\pi}\left[Q\left(\mathbf{s}',\mathbf{a}'\right)\right])\right)^2\right]\end{aligned} \tag{10}$$

The third term here is our standard Bellman error. The first term always pushes down Q values to prevent overestimation. The second term here is used to counterbalance by maximizing the Q-values of the actions in the dataset.

Given that our action space is continuous, we use a Soft-Actor critic (SAC) [(Haarnoja et al., 2018)] implementation of CQL. SAC is defined for RL tasks involving continuous actions, aiming to maximize both the entropy of the policy and the total expected lifetime rewards.

3 Related work

3.1 Offline Reinforcement Learning

There is an increasing interest in the development and enhancement of RL algorithms for offline scenarios. One set of techniques for Offline RL is dedicated to enhancing the stability of off-policy Q-learning by mitigating the tendency to overestimate the Q-function in areas beyond the range of the dataset [(Kostrikov et al., 2021),(Kumar et al., 2020)].

Another category of approaches is centered around doing offline RL in a model-based environment, employing a learning process akin to the one described in [(Janner et al., 2019)]. At a macroscopic level, these methods initially optimize a model $f\left(s_t, a_t\right)$ to accurately anticipate the transition dynamics of the environment. Next, they train a policy by conducting autoregressive rollouts of the dynamics model using actions sampled from the policy that is being optimized. However, like the Q-learning algorithms, the model frequently fails to accurately predict outcomes in areas of the state space that are not inside the data distribution. Hence, recent studies, as those mentioned in the range [(Kidambi et al., 2020),(Yu et al., 2020)], have developed dynamics models that take into account uncertainty. These models incorporate a penalty in the reward function or termination function based on the level of uncertainty in the state estimation during policy optimization.

Our approach makes use of the INTERACTION data set, for which there is limited prior art. Our works differs from the prior works on this data set by investigating offline RL in a roundabout scenario, while also making use of ensemble Q-learning.

3.2 Offline RL dataset for autonomous driving

In recent years, many corporations and research entities have started to publicly release their autonomous driving datasets [(Chang et al., 2019) (Ettinger et al., 2021)]. However, identifying the most valuable

datasets can be difficult as it typically involves extensive online research. Fields like motion prediction, imitation learning, representation learning, behavior modeling, and algorithm testing rely on the availability of high-quality motion datasets. Ideal datasets should cover interactive driving situations across diverse driving cultures and behaviors. Instead of using popular autonomous driving datasets like Argoverse [(Chang et al., 2019)] and Waymo Open Dataset [(Ettinger et al., 2021)], we use the real-world driving trajectory dataset INTERACTION [(Zhan et al., 2019)].

INTERACTION dataset [(Zhan et al., 2019)] is a dataset we chose to use in this project which features a diverse range of driving scenarios captured by drones, showcasing high levels of interaction. It provides different interactive scenarios including: roundaboud with 10479 vehicle data, unsignalized intersection with 14876 vehicle data, merging and lane change with 10933 vehicle data, and signalized intersection with 3775 vehicle data.

3.3 Similarities and Difference to Existing Work

In the landscape of autonomous vehicle (AV) research, our work situates itself amidst the growing exploration of offline reinforcement learning (RL) for motion planning. Drawing on the foundation of using a dynamics model trained on historical driving data, our research aligns with contemporary approaches in offline model-based RL. This methodology, leveraging the INTERACTION dataset, aims to refine motion planning policies by utilizing rich, real-world driving scenarios without necessitating further online data collection that poses safety risks.

Despite these commonalities with existing research, our project carves out its distinct niche through several key innovations and focus areas:

Utilization of the INTERACTION Dataset: Contrary to common practice where popular datasets like Argoverse or the Waymo Open Dataset are employed, our research utilizes the INTERACTION dataset. This choice is strategic, aiming to harness the dataset's comprehensive coverage of diverse, interactive driving situations across various driving cultures and behaviors, specifically focusing on complex scenarios such as roundabouts, which have seen limited exploration in existing CQL-based studies.

Unique Approach to Safety and Interaction: Aligning with the feedback from our team, our approach innovatively applies ensemble CQL, differing from existing practices by focusing on roundabout navigation. Furthermore, our custom reward function, which emphasizes maintaining a safe distance over direct collision avoidance, presents a nuanced method of enhancing safety, reflective of the proactive measures integral to real-world driving.

In summary, while our project is grounded in the principles of offline CQL RL, it diverges through an effort to tackle pivotal issues such as safety, interaction/proximity, and the complexity of real-world driving scenarios.

4 Experimental results

4.1 Offline training dataset

We process real world driving trajectories in the roundabout map from INTERACTION [(Zhan et al., 2019)] to create corresponding offline data. First, we get ego's state and action from its driving trajectory. We then filter out the nearest three interactive agents at the same timestamp. This information is input into our designed reward function to calculate the reward. Finally, we obtain 6134 ego vehicle's MDP tuples as the offline data. The data preprocessing framework for autonomous driving is illustrated in Fig. 5.

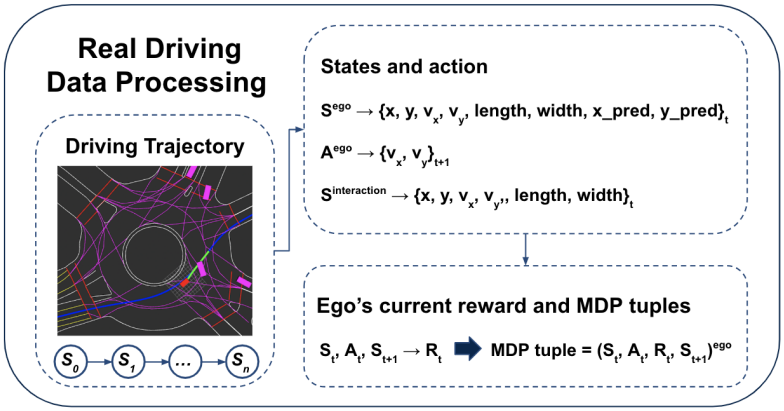

Figure 5: The offline training dataset preprocessing framework for autonomous driving based on INTERACTION dataset

4.2 Training Results

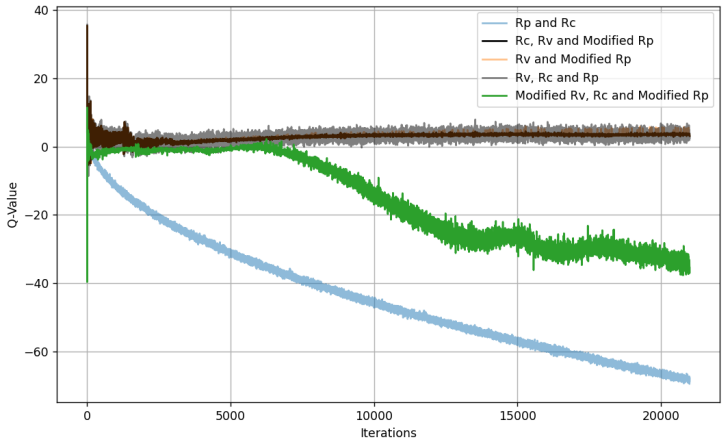

Figure 6: Q-value curves as a function of training iterations for variations of our reward function. Rc and Rv are the collision and velocity rewards, respectively. Rp is the safety reward as described in the Reward Function section above. Modified Rp and Modified Rv refer to scaling the safety and velocity component, respectively, of our reward function and bounding to [-1 1].

Figure 6 shows the results of ablation experiments we performed, systematically removing one component of the reward function. Recall that the full reward function is comprised of safety, velocity and collision components.

For the collision reward, velocity reward and modified safety reward, the q-values were initially high, indicating an explorative approach that captures higher rewards. As the training progresses, the q-values exhibited a decline followed by a stabilization (more stable than the base).The positive and stabilizing trend of the q-values underlines the contribution of the safety reward to enhance the agent's ability to navigate and position effectively within complex environments. The agent's behavior indicates an adaptation that balances exploration with safety.

For the safety and collision reward components, the q-values were persistently negative with considerable fluctuations, indicating a struggle in achieving a stable policy. The absence of positive trends or stabilization points to issues with reward alignment or possibly the agent's inability to resolve conflicting signals. The consistent negativity and high volatility suggest that the original settings of ignoring rv is a totally poor reward structuring. It has excessive penalization which also highlight the need for rv in the reward function.

For the velocity and Modified Rp, we had fewer spikes and less pronounced declines, this configuration indicates a fairly well-balanced interaction between velocity and modified position reward. It aligns well with the desired driving behaviors. So Rc here may don't have too much contributions to the final. It may due to our small penalization or the collision between egos happened not too frequently.

For the modified Rv, Rc and Modified Rp. We found that it actually marked by significantly negative q-values and a downward trend, this configuration also suggests that the modification of Rv seems overly penalizing or misaligned which leads to deterrence of potentially beneficial strategies. We need to do more fine-tuning for the Rv to place the penalty in a more reasonable interval.

The plot above demonstrates that each modification has a distinct impact on the learning dynamics and decision-making processes of the autonomous driving agent. The modifications of Rp significantly enhance the model's performance by promoting safer and more strategic behaviors. However, some configurations, particularly those with modified Rv, appear to hinder performance, suggesting over-penalization that does not produce the desired result. These findings underscore the necessity for careful tuning of reward components to ensure they constructively contribute to the agent's learning and operational effectiveness.

We evaluated our agent on metrics given on the INTERACTION INTERPRET challenge, min ADE represents the minimum value of the euclid distance averaged by time between the ground truth and our predicted position with the lowest value, min FDE represents the minimum value of the euclid distance at the last predicted timestamps between the ground truth and our predicted position with the lowest value and miss rate is calculated if at the last timestep our predicted position is outside a threshold of 2 meters from actual position then it is called a miss.

We can see that on min ADE and min FDE, our predictions perfrom on par or better than the leaders on the leaderboard but our miss rate is very high (85%), which would suggest that it frequently fails to predict within a specified threshold of the actual endpoint.

Table 2: Agent's performance on INTERACTION Metrics

min ADE	min FDE	Miss rate
0.42629	0.31765	0.85762

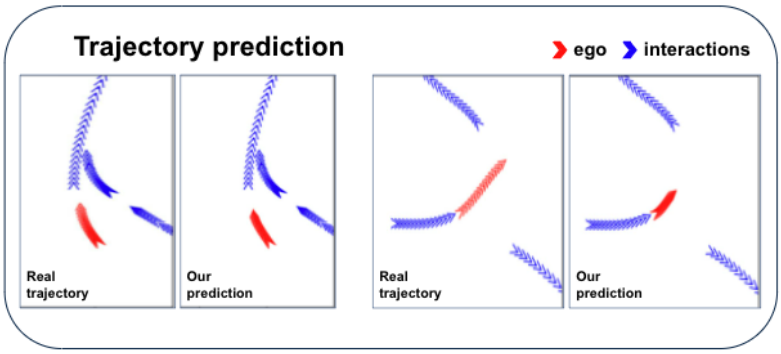

Figure 7: Illustration of our prediction results when applied to the vehicle's future action in the next three seconds.

Figure 7 shows a comparison of our trajectory predictions compared to those contained in the dataset over a 3 second time period. The pair of plots on the left and right are two separate scenes from the INTERACTION dataset. For each pair, the left plot shows the trajectory contained in the dataset and the right plot shows our prediction. From the plots, it is clear that our trajectory follows that in the dataset - however, it our agent travels slower and does not cover as much ground in the 3 second snapshot shown.

5 Conclusion

In this project, we propose an offline RL method for autonomous driving using real-world interactive driving data. We first create an offline RL dataset and build an MDP tuple from the real-world driving dataset known as INTERACTION. We develop a reward function to consider the velocity of our RL agent and its safety (i.e. being able to maintain a safe distance from objects around it). For the latter, we reward safe distances while heavily penalizing collisions. We then apply CQL to our designed reward function to model human driving behavior.

Our results show that we successfully predicted the ego's future action and trajectory, but there is a slight gap between our speed and the actual trajectory. The results reveal a nuanced understanding of how modifications to velocity and position rewards influence an agent's performance in autonomous driving scenarios. We anticipated that our modifications to Rv, which were intended, would promote more realistic driving speeds. However, the modifications caused instability in the training behavior of our model. This unexpected result underscores the complexity of modeling real-world driving behaviors.

As future work, we plan to optimize our reward function to improve the velocity prediction. We will use a more iterative approach to ensure that they contribute constructively to the agent's learning and decision-making processes. Our model also has several hyperparameters that we would like to tune for more stable learning.

6 Author Contribution

Aakash - Developed reward function and its mathematical formulation and did background research on autonomous vehicle industry current events.
Shitanshu - Did research on approach to take and problem formulation with focus on Q-learning and CQL. Implemented replay buffer and SAC-CQL and worked with Yuxing to set up evaluation metrics.
Kuang-Yang - Did Research on datasets, analyzed the data on INTERACTION dataset, designed state vector and processed the dataset for offline RL use.
Yuxing - Reviewed, summarized, and integrated related works by other people on offline reinforcement learning. Worked with Shitanshu to modify the rp rewards, implement evaluation metrics, and run training experiments.

References

Tim De Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. 2015. The importance of experience replay database composition in deep reinforcement learning.

Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. 2019. Argoverse: 3d tracking and forecasting with rich maps.

Camila Domonoske. 2024. "gm's driverless car company cruise is under investigation by several agencies" national public radio: Wbex chicago, january 25, 2024. https://www.npr.org/2024/01/25/1226953350/gm-driverless-cruise-investigation-federal-agencies.

Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles Qi, Yin Zhou, Zoey Yang, Aurelien Chouard, Pei Sun, Jiquan Ngiam, Vijay Vasudevan, Alexander McCauley, Jonathon Shlens, and Dragomir Anguelov. 2021. Large scale interactive motion forecasting for autonomous driving : The waymo open motion dataset.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. 2019. When to trust your model: Model-based policy optimization.

Athanasios Kapoutsis. 2022. The monster of distribution shift in offline rl and how to pacify it (https://medium.com/@athanasios.kapoutsis/the-monster-of-distribution-shift-in-offline-rl-and-how-to-pacify-it-4ea9a5db043).

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. 2020. Model-based offline reinforcement learning.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline reinforcement learning with implicit q-learning.

Tom Krisher. 2023. Tesla's recall of 2 million vehicles to fix its autopilot system uses technology that may not work.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative q-learning for offline reinforcement learning.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. 2020. Mopo: Model-based offline policy optimization.

Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kummerle, Hendrik Konigshof, Christoph Stiller, Arnaud de La Fortelle, and Masayoshi Tomizuka. 2019. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps.

7 Appendix

Our code can be download here: link