Project report: Safe and Efficient offline learning

Tianyu Shi
McGill University
Montreal, Canada
tianyu.shi3@mail.mcgill.ca

***Abstract*—Intelligent motion planning is one of the core components in robotics, which has received extensive interests. Due to the uncertainty and complexity of the environment, most classical rule-based methods cannot solve the problem of complicated decision tasks. Deep reinforcement learning has demonstrated impressive achievements in many fields such as playing games and robotics. However, many practical online reinforcement learning applications, such as autonomous driving, will involve interactions with environment, which can result in high deployment cost. In this project, we propose an offline reinforcement learning approach to learn from the collected dataset. Due to the conservativeness of most off-line reinforcement learning methods, we introduce an exploration strategy to make the agent to explore efficiently in the environment. Meanwhile, we also introduce the Lyapunov functions to provide the safety guarantee during the policy learning process. An efficient and safe off-line reinforcement learning method is designed to allow the agent can not only efficiently explore the action space but also guarantee safety. Numerous experiments have been conducted in autonomous driving tasks to evaluate the learned policy. These experimental results indicate that the proposed model outperforms several baseline methods.**

***Index Terms*—off-line reinforcement learning , Motion Planning, safe Learning, Autonomous driving**

I. Introduction

While reinforcement learning (RL) algorithms have achieved impressive results in games, for example on the Atari platform [1], they are rarely applied into real-world system (e.g. robotics). One of the main obstacles is the high deployment cost for most robotics tasks. The process of reinforcement learning involves interacting with the environment to collect the experience into reply buffer, and then sampling experience from replay buffer to optimize the policy. In many robotics settings, this kind of online interaction is impractical, either because data collection is expensive (e.g., in robotics, educational agents, or healthcare) and dangerous (e.g., in autonomous driving) [2]. Furthermore, such as in autonomous driving tasks, a large datasets will also improve the generalization ability of the agent [3].

Fortunately, in most robotics tasks, we can use human/expert policy to collect dataset to train our agent. In autonomous driving field, most companies have released their own datasets, such as Baidu, Waymo, etc [4]. These driving data can be considered as human demonstration to help the agent to learn the behavior. Imitation learning that learns policies via supervised training on human driving data has been applied to a variety of tasks, including modeling lane following [5] , navigational behavior [3], off-road driving [6], etc. In [7], they proposed a joint learning algorithm that learns a shared cost function employed by behavior and trajectory components. It provides interpretable costing function for each driving constraints and make the end-to-end learning trainable. In [8], they proposed a Learning from Demonstration (LfD) framework, supervised auxiliary task prediction was used to guide the main task of predicting the driving commands, and an end-to-end trainable network for imitating the expert demonstrator's driving commands was introduced. However, due to the inherent characteristic of end-to-end imitation learning, there was an inevitable cascading failure. The first reason is that the learned driving policy might not generalize to various scenarios since most of the data were collected in situations with limited variations. The second reason is that they didn't consider exploration strategy in their method which makes their methods lack of generalization abilities.

As pointed out by [9], distribution shift problem, which is introduced by the mis-match of state distribution between the collected data-set and the current environment is a major challenge in offline learning setting. The main idea to solve this problem is to constrain the learned policy within the range of state visitation space. In [9], they proposed to train the generative model to select the most similar action previously seen from the dataset. In [10], they introduced the value penalty onto the target Q-values to regularize the learned policy towards the expert policy, which is very similar to minimizing the KL-divergence between the learned policy and expert policy from the dateset [11]. These offline reinforcement learning approach can regularize the learned behavior within the range guaranteed by dataset. However, their learned behavior will be too conservative, which makes the agent cannot efficiently explore the state and action space.

To encourage the exploration, in [12], they proposed to use meta learning to encourage the efficient policy exploration. Instead of adding a random noise onto the learned action, they proposed to train flexible exploration behaviors that are independent of the actor policy within the inner loop optimization, encouraging a globally exploration. Another way is to create a bonus for exploration, e.g estimates the upper confidence bounds [13]. However, a too optimistic exploration strategy will also allow the agent to try unsafe behavior. As a result, a good balance between the exploration and safety is

*This is the final project for Fall 2020: COMP 766 / ECSE 683 Learning and Optimization for Robot Control, supervised by Prof. Hsiu-Chin Lin.

also an important issue.

In this project, we propose a learning based approach which overcame the aforementioned shortcomings. An exploration strategy is used to guarantee a globally exploration and the learned behavior is simultaneously optimized based on Lyapunov stability guarantee. We evaluated our method based on several autonomous driving tasks and demonstrated the effectiveness of our method on balancing both efficiency and safety for offline learning.

II. Background

A. Offline Reinforcement learning

A typical reinforcement learning is formulated as an agent interacting with the environment, following MDP process (S, A, P, r) with state space S, action space A, transition probability P and reward r. The goal of the agent is to maximize the sum of discounted total return $R_t = \sum_{i=t+1} \gamma^i r(s_i, a_i, s_{i+1})$. where γ is the discount factor. The policy is a mapping $\pi : S \to A$, we can use the value function to evaluate the quality of this policy, $Q^\pi(s, a) = E_\pi[R_t|s, a]$. For a given policy π, the value function can be computed through bellman equation:

$$Q(s, a) \leftarrow r + \gamma Q(s', a') \tag{1}$$

Specifically, (s', a') stands for the next state and action while (s, a) stands for the current state and action. In an offline setting, the (s, a, r, s') is not come from the environment but from the collected dataset. During a value update where the target policy selects an unfamiliar action a' at the next state s' in the backed-up value estimate, such that (s, a) is unlikely to be contained in the dataset. Then extrapolation error is introduced due to the wrong estimation of $Q(s', a')$ which is due to the lack of sufficient data near (s', a').

In an on-policy setting, extrapolation error may be a source of beneficial exploration through encouraging agent to be optimism in the face of uncertainty strategy [14]. In this case, if the value function overestimates an unknown state-action pair, the policy will collect data in the region of uncertainty, and the value estimate will be corrected. However, when learning in offline setting, extrapolation error will never be corrected due to the inability to collect new data.

B. Variational Auto-Encoder

The variational auto-encoder (VAE) is a generative model which aims to maximize the marginal log-likelihood $logp(X) = \sum_{i=1}^N logp(x_i)$ where $X = x_1, ..., x_N$ is the dataset. Equivalently, we can instead optimize the variational lower-bound:

$$logp(X) \geq E_{q(X|z)}[logp(X|z)] + D_{KL}(q(z|X)||p(z)) \tag{2}$$

where $p(x)$ is chosen a prior, generally the multivariate normal distribution $N(0, I)$. We can define the posterior $q(z|X) = N(z|\mu(X), \sigma^2(X)I)$ as the encoder and $p(X|z)$ as the decoder. Given a sample $x = (s, a)$ with state and action information from the dataset. The VAE is trained to generate the most similar action from the dataset, which is referred as the reconstruction loss term. To regularize the generation behavior, KL-divergence term according to the distribution of the latent vectors z is added. To make the variational lower bound differentiable, we can use the re-parametrization trick [15]:

$$\mu_x, \sigma_x = M(x), \sum(x) \tag{3}$$

$$z = \epsilon\sigma_x + \mu_x \tag{4}$$

where μ_x and σ_x represent the deterministic function with the samples x as input, and $\epsilon \sim N(0, 1)$.

C. Deep Deterministic Policy Gradient

In our experiment, we consider the agent have continuous action generation. As a result, it's impractically to directly apply Q learning because Q learning can only generate discrete action. Deep deterministic policy gradient (DDPG) [16] address the issue by training a parameteric policy network together with policy gradient. In training, the critic $Q_\theta(s, a)$ is updated using the Bellman equation as in Q-learning, the actor is updated to maximize the expected reward w.r.t $Q_\theta(s, a)$, $\mu(s, \theta^\pi)$ is a parametric function, such as a neural network, that maps the state to action,

$$\max_{\theta^\pi}\{J(\theta^\pi := E[Q_\theta(s, \mu(s, \theta^\pi))]\} \tag{5}$$

This is achieved by using the gradient update:

$$\theta^\pi \leftarrow \theta^\pi + \nabla_{\theta^\pi} J(\theta^\pi) \tag{6}$$

where, $\nabla_{\theta^\pi} J(\theta^\pi) = E[\nabla_a Q_\theta(s, \mu(s, \theta^\pi))\nabla_{\theta^\pi}\mu(s)]$. The performance of DDPG critically depends on a proper choice of exploration strategy π_e. In practice, an guassian noise is usually added onto the action generated by the DDPG policy to encourage the agent to explore a locally wider range. However, as pointed out by [12], adding noises to the on-going policy would only explore locally close to what the actor policy dictates, which limits the agent to explore potentially beneficial novel states to improve the policy.

D. Lyapunov stability guarantee

Given control input u and state input x, the controlled system can be defined as:

$$\frac{dx}{dt} = f(x, u) \equiv f_u(x) \tag{7}$$

Each $x(t) \in D$ is a state vector and D is the state space of the system. Meanwhile $u(t) \in R^m$ is the control input vector.

A system is stable at the origin for any $\epsilon \in R^+$, there exists $\delta(\epsilon) \in R^+$, such that if $||x(0)|| < \delta$ then $||x(t)|| < \epsilon$ for all $t \geq 0$. The system is asymptotically stable if it is stable and also $lim_{t\to\infty}||x(t)|| = 0$ for all $||x(0)|| < \delta$ [17].

According to the chain rule, the derivative of a continuously differentiable Lyapunov function $V(x)$ over the vector field of system dynamics $f_u(x)$ can be written as:

$$\nabla_{f_u} V(x) = \sum_{i=1}^n \frac{\partial V}{\partial x_i}\frac{dx_i}{dt} = \sum_{i=1}^n \frac{\partial V}{\partial x_i}[f_u]_i(x) \tag{8}$$

Consider a controlled system $f_u(x)$ with state x and input u if :

$$V(0) = 0, and, \forall x \in D, V(x) > 0, and, \nabla_{f_u} V(x) < 0 \quad (9)$$

Then the system is asymtotically stable at the origin. Furthermore, if given the goal point x^*, and if we consider $V(x^*) = 0$ within Eq.9, then the system will be stable at the goal point.

III. Methodology

A. Learning to explore

For offline learning, the goal is to select the most likely action that was seen from the dataset. For a given state, our model will generate feasible candidate actions with high similarity to the batch of data and then select the highest valued action through the Q-network. Give a batch of data B,the imitation problem can be formulated as maximizing the marginal likelihood of $P_B(a|s)$ by minimizing the extrapolation error. To generate the most likely action, we use the parametric generative model $G_w(s)$as suggested by [9]. The actions can be sampled from the approximation to $\mathrm{argmax}_a P_B^G(a|s)$. As shown in the experiment from Sec. V, the original method proposed by [9] is a conservative model because they use the variational auto-envoder (VAE) [15] model to constrain the action within the boundary of dataset. We expect to construct a better exploration strategies that are potentially better than the default Gaussian noise onto the action [16]. Furthermore, it's also possible that the actor policy gets stuck in a local range in the state space and thus cannot escape even with random Gaussian noises added.

Most existing offline learning strategies [9]–[11] are based on the implicit assumption that the exploration policy π_e should be similar to expert policy π learned from the dateset, or maybe add some stochastic noise within a personalized range. However, this assumption may not be true. Instead, it may be beneficial to make π_e significantly different from the actor π in order to explore the space that has not been explored previously. Even in the case of using Gaussian noise for exploration, the magnitude of the Gaussian noise is also a critical parameter that may influence the performance significantly and the clipped boundary in [14]. Therefore, it is important to develop a systematic approach to dynamically learn the exploration strategy, instead of using simple random noise.

In this project, to encourage a globally exploration, we consider to add a learnable noise onto the weights and biases of the network and induce stochasticity into the agent's policy that can be used to aid an efficient exploration. Unlike adding noise on action, this exploration strategy is not limited into offline reinforcement learning, but it still can be extended into other imitation learning methods as long as it's using neural network as part of the model [18].

To be specific, consider a linear layer of a neural network with p inputs and q outputs. represented by:

$$y = wx + b \quad (10)$$

where $x \in R^p$ is the layer input and $w \in R^{q \times p}$ is the weight matrix, and $b \in R^q$ is the bias. The corresponding network with perturbation noise is defined as:

$$y = (\mu^w + \sigma^w \cdot \epsilon^w)x + \mu^b + \sigma^b \cdot \epsilon^b \quad (11)$$

where $\mu^w + \sigma^w \cdot \epsilon^w$ and $\mu^b + \sigma^b \cdot \epsilon^b$ replace the w and b in Eq. 10. The parameters μ^w, μ^b, σ^b are learnable parameters of the neural network. ϵ^w and ϵ^b are noisy variables which can be learned through policy gradient update.The main difference is that in the noisy layer both the weights vector and the bias is perturbed by the learnable noise. Firstly, the objective of reinforcement learning is to learn the policy π so as to maximize the value function:

$$\pi(s) = \underset{a(\epsilon)}{\mathrm{argmax}}\, Q_\theta(s, a(\epsilon)) \quad (12)$$

The perturbation noise $\epsilon(\phi)$ can be trained to maximize the value function $Q_\theta(s, a(\epsilon))$ through policy gradient:

$$\phi \leftarrow \underset{\phi}{\mathrm{argmax}} \sum_{(s,a)\in B} Q_\theta(s, a(\epsilon)) \quad (13)$$

where ϕ is the parameter of the perturbation model.

B. Learning to provide safety guarantee

After we get the exploration strategy, one important question is how to guarantee the safety of the learned policy. In this project, we consider to minimize the Lyapunov risk so as to optimize the generated action from exploration policy in order to provide a safe guarantee of the system. According to E.q 9, conceptually, the overall lyapunov control design problem can be formulated in this following unified minimizing the minimax form of cost function:

$$\inf_{\theta,u} \sup_{x\in B} [\max(0, -V_\theta(x)) + \max(0, \nabla_{f_u} V_\theta(x)) + V_\theta^2(x^*))] \quad (14)$$

The objective is to make the system can converge around the target state x^* over all state $x \in B$. For the learning objective we want to minimize the Lyapunov risk function which is defined as:

$$\mathrm{L}(\theta, u) = E_{x\sim\rho(B)}[\max(0, -V_\theta(x)) + \max(0, \nabla_{f_u} V_\theta(x)) + V_\theta^2(x^*))] \quad (15)$$

where x is the state variable that can be sampled with a distribution ρ from the batch of data. For evaluation of the state function, the design of the lyapunov function is of great importance. As pointed out by [19], a specific design form (e.g. quadratic form) of lyapunov function is not suitable for nonlinear system dynamics. As a result, we consider to represent the lyapunov function as a neural network form for nonlinear dynamics, the lyapunov function should follow convexity, positive definite and continuously differentiable, therefore, we define the function as:

$$V(x) = \sigma_{k+1}(g(x) - g(0)) + \epsilon||x||^2 \quad (16)$$

where σ_k is a positive convex non-decreasing function with $\sigma_k(0) = 0$, g is the Input convex network as given in [20], and ϵ is a small constant. Especially, for σ_k we can use the smooth ReLU given by [21].

C. Jointly learning for safe and efficient exploration

Unlike most offline learning [9], [10] and safe learning approachs [17], [21], the intuition of the approach we propose in this project is straightforward: instead of learning a exploration strategy or attempting to separately generate the stable policy via a Lyapunov function, we propose to jointly learn a policy that can balance a safe and efficient exploration given on the training dataset. We argue that an over conservative policy or over optimistic policy are both not most beneficial to get high returns from the environment. We use a combination of exploration objective and stable guarantee as our loss function:

$$L(\phi,\psi) \equiv L(w) = \frac{\lambda_w}{2}||w||_2^2 + \lambda_s L_s(w) + \lambda_e L_e(w) \quad (17)$$

where the first term is a regularization term, second term is to minimize the Lyapunov risk for safety guarantee and the third term is to encourage efficient exploration. And λ_s and λ_e are two hyperparameters that scale the loss components.Note that w is the parameters of the total loss function and ϕ, ψ are parameters of exploration and risk losses. The main framework of our offline learning method is given in Algorithm 1.

Algorithm 1 Jointly learning safety and efficiency

Initialize: Batch of data B, horizon T, number of sampled action n, loss weights λ_s,λ_e, Q-networks $Q_{\theta 1}$ and $Q_{\theta 2}$. perturbation network ϵ_ϕ and VAE $G_w = (E_{w1}, D_{w2})$, Lyapunov function V_ψ.
for episode p=1:M **do**
  Sample mini-batch of data (s,a,r,s') from the dataset
  μ, $\sigma = E_{w1}(s,a)$, $\dot{a} = D_{w_2}(s,z)$, $z \sim N(\mu,\sigma)$
  w $\leftarrow argmin_w \sum(a-\dot{a})^2 + D_{KL}(N(\mu,\sigma)||N(0,1))$
  Sample n actions: $a_i \sim G_w(s')_{i=1}^n$
  (Explore efficiently) Generate Perturbed actions: $a_i \sim \epsilon_\phi(s')_{i=1}^n$
  Compute exploration loss: L_e according to Eq.13
  (Guarantee Safety) Compute Lyapunov risk L_s according to Eq.15
  Compute joint loss: $L(w') = \frac{\lambda'_w}{2}||w'||_2^2 + \lambda_s L_s(w') + \lambda_e L_e(w')$
  Update parameters: $\phi \leftarrow \phi\nabla_\phi L(\phi)$,$\psi \leftarrow \psi\nabla_\psi L(\psi)$
end for

IV. Experiments

A. Problem formulation

In this project, we control a four wheels autonomous vehicle with bicycle model:

$$\dot{x} = vcos(\phi+\beta) \quad (18)$$

$$\dot{y} = vsin(\phi+\beta) \quad (19)$$

$$\dot{v} = a \quad (20)$$

$$\dot{\phi} = \frac{v}{l}sin\beta \quad (21)$$

$$\beta = \tan^{-1}\frac{1}{2tan\delta} \quad (22)$$

where (x,y) is the vehicle position; v is the forward speed; ϕ is its heading; a is the acceleration command; β is the slip angle at the center of gravity; δ is the front wheel angle used as steering command. The autonomous vehicle's settings are given as following:

(1) **Simulator:** *highway-env*[1].
(2) **State space:** $q = [x, y, v_x, v_y, a]$.
(3) **Action space:** $A = [acceleration, steering]$.
(4) **Degree of freedom:** $N = 3$.
(5) **Task:** Tracking target points.

B. Scenario description

1) Highway scenario: A typical lane-change scenario in the highway is displayed in Fig. 1. The ego vehicle (EV) was intended to take a lane-change maneuver to its left side. During the movement, three surrounding obstacle vehicles should be considered by the motion planning algorithm: the leading vehicle in the ego vehicles' current lane (ego lane, EL), and two nearest vehicles in the target lane (TL). Specifically, to

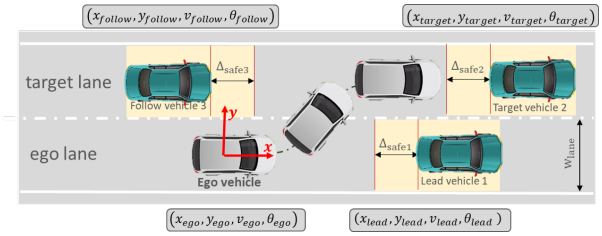


Fig. 1. lane-change scenario construction.

satisfy the obstacle avoidance constraints and vehicle dynamics constraints, the ego vehicle needs to determine a suitable lateral yaw rate and longitudinal acceleration in a given time horizon, steering into the TL. In addition to ensuring the success of lane changing, the motion planning module should also guarantee the efficiency and safety of the ego vehicle's movement (small acceleration). Also, the motion planning module should satisfy a safe distance between the EV and surrounding obstacle vehicles.The reward is defined as:

$$R = a\frac{v - v_{min}}{v_{max} - vmin} - bcollision \quad (23)$$

where v,v_{min},v_{max} are the current, minimum and maximum speed of the ego-vehicle repectively, and a, b are two coefficients.

2) Parking scenario: As shown in Fig. 2, the goal of this scenario is to generate a target point and make the robot get to the target position while keeping the robot within its physical limits. The vehicle must park in a given space with the appropriate heading and position.The reward is defined as:

$$R = -a||s - s_g||^2 - bviolation \quad (24)$$

where the $s = [x, y, v_x, v_y, cos(\phi), sin(\phi)]$ is the current and $s_g = [x_g, y_g, 0, 0, cos(\phi_g), sin(\phi_g)]$ is the goal and violation represent the penalty on constraint violation.

[1]https://highway-env.readthedocs.io/en/latest/

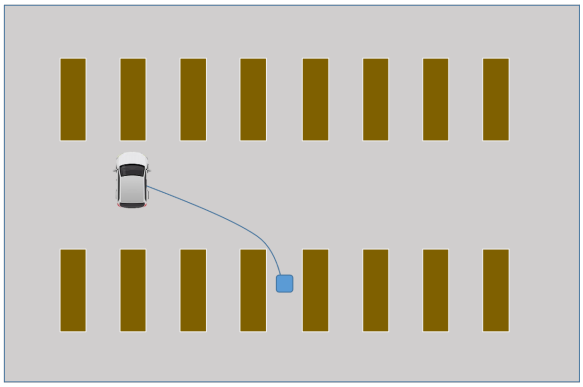


Fig. 2. parking scenario construction.

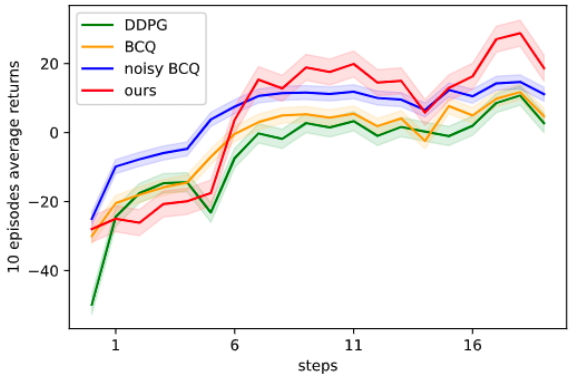


Fig. 3. Returns in highway.

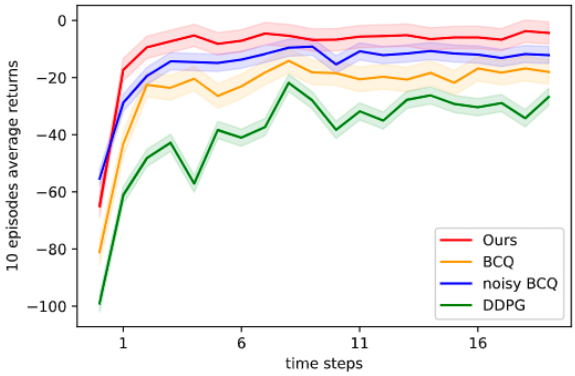


Fig. 4. Returns in parking.

3) Data Collection: Stable Baselines is a set of improved implementations of Reinforcement Learning (RL) algorithms based on OpenAI Baselines. For a fair comparison, we use the DDPG model from *open ai stable-baselines*[2] as the expert model to collect demonstration data.

V. Model Validation and Simulation

A. Methods comparison

In this section, to evaluate the effectiveness of our proposed approach, we compare our method with the state-of-the-art offline reinforcement learning baseline [9]. For all experiments, we run 200 episodes with collection of the average 10 episodes returns results of 5 random seeds. The explanations for selecting these baselines are given as follows.

- Deep Deterministic Policy Gradient (DDPG) [16]: DDPG is a deterministic version of model-free RL algorithm to deal with continuous action space. Autonomous vehicle agent can reliably learn the optimal policy with continuous actions. We construct the training framework based on Open-ai stable baseline to serve as expert behavior, which is similar to the concurrent training as given in [14].
- Batch Constraint Reinforcement Learing(BCQ) [9]: This is a batch reinforcement learning method for continuous control. BCQ aims to perform Q-learning while constraining the action space to eliminate actions which are unlikely to be selected by the behavioral policy π_b, and are therefore unlikely to be contained in the batch of data.
- Noisy BCQ: In this version, we consider only adding exploration strategy on the policy as given in Sec III. However, it doesn't have any safety guarantee.

We first conduct a comparison on the training efficiencies of the proposed method and other baseline methods in these two scenarios.

As shown in Fig. 3 and Fig. 4, both our method and BCQ method will be better than expert model which is in accordance to the results in [14] because we select a well trained expert policy to collect part of the dataset. Furthermore, our method can make an improvement on BCQ and achieve the overall best performance in these two scenarios. Most interestingly, we found that a small perturbation on the policy will also increase the exploration of the naive BCQ method which will achieve a better return than before. And it further validate our argument that BCQ is a conservative method which is not quite beneficial for exploration.

To further evaluate the performance of the learned policy, we analyze the distance to the closest vehicle and the velocity and acceleration performance within these two scenarios. We also evaluate the success rate within the parking scenario.

Firstly, for highway scenario, we evaluate the minimum distance to the surrounding vehicles during the lane change process. As shown in Fig. 5, the vehicle controlled with our method will have overall bigger minimum distance than noisy BCQ. Furthermore, our method's minimum distance are mainly a normal distribution with expectation as 9m. According to Fig. 6, the minimum distance will have some around 2m to 6m while some are around 14m which demonstrates a big variation than our method. The reason is that noisy BCQ only considers exploration but doesn't consider the system stability, which makes them unstable in the lane change scenario.

Secondly, for parking scenario, we evaluate the steering angle and acceleration performance of the baseline method and our method. As shown in Fig. 7, the vehicle controlled with our method will have overall smaller steering angle than noisy

[2]https://stable-baselines.readthedocs.io/en/master/

BCQ, which leads to a smaller acceleration of our method than noisy BCQ as shown in Fig. 8. In the simulation, we found that noisy BCQ tends to have a sharp steering angle which will make the vehicle move faster to the goal point but will result in overshooting to target point and it will further lead to oscillation around the target point.

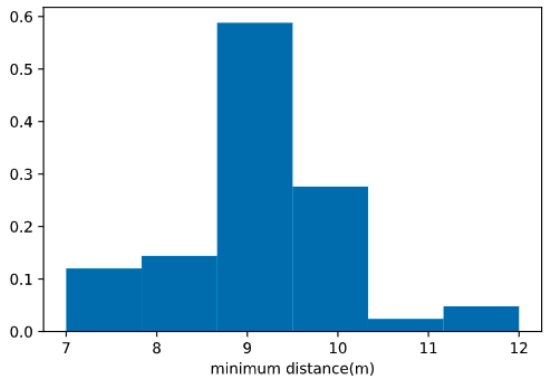


Fig. 5. Minimum distance with our method.

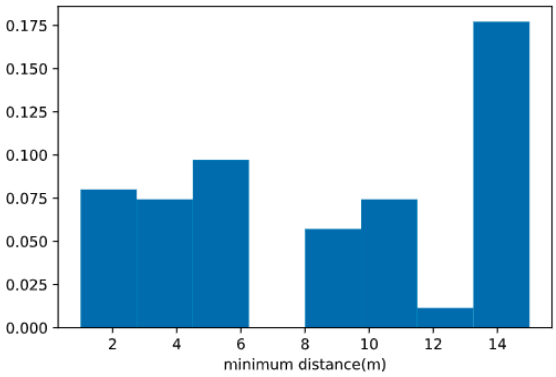


Fig. 6. Minimum distance with noisy BCQ.

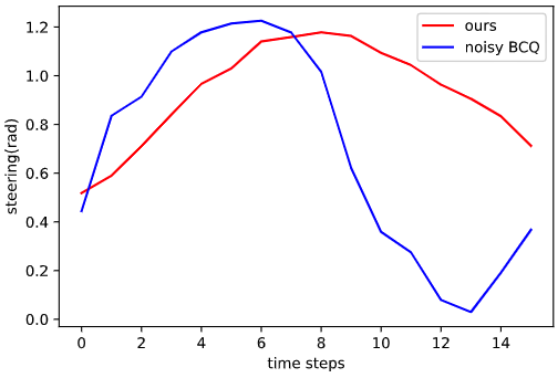


Fig. 7. Comparison of steering performance.

We can see from the state action density plot in Fig. 10 that the original BCQ method tends to explore very cautiously with very limited amount state and action visitation. On the other hand, the noisy BCQ demonstrates more diverse state

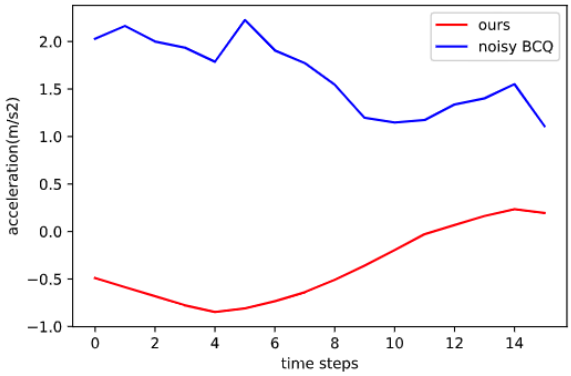


Fig. 8. Comparison of acceleration performance.

and action visitations, even it will explore some action that it is not common given the same state as BCQ. As a result, the noisy BCQ explore the state action space more efficiently than BCQ. Furthermore, our method could explore nearly the same state range as noisy BCQ but it will tend to explore the action space in a reasonable range, which is due to the safety concern within our method. Combined with the previous experiment results, we can conclude that our method can achieve the best balance between safety and efficiency among BCQ and noisy BCQ. From Fig. 9, most importantly, our method can achieve the best success rates in the parking scenario.

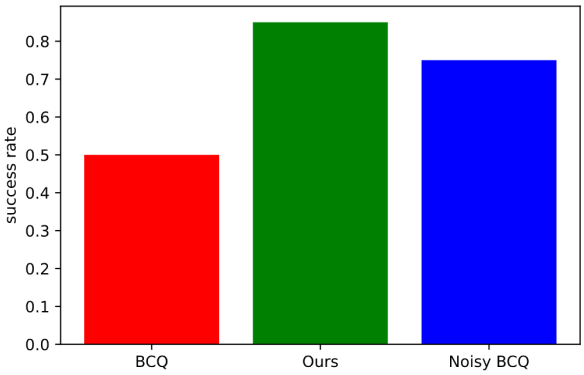
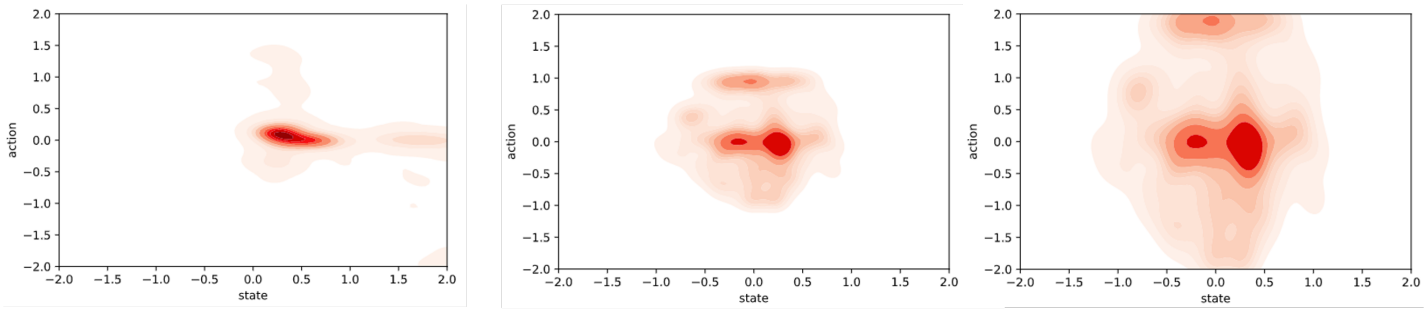


Fig. 9. Comparison of different success rates in parking scenario.

VI. Conclusion and Future Work

In this project, We design an algorithm to explore the state action space efficiently while guarantee the system safety. To be specific, we adaptively learn the noisy onto the network's parameters. Simultaneously, we will optimize the learned policy to provide safe guarantee based on Lyapunov stability. The combination of safe and exploration loss demonstrated better overall performance than other methods. More importantly, our method is not limited into offline reinforcement learning but still can be integrated into other imitation learning methods. As most imitation learning methods focus on generating the most likely behaviors from the dataset. To our understanding, existing imitation learning methods haven't explicitly discussed



(a) BCQ (b) Ours (c) Noisy BCQ

Fig. 10. State action density contours in parking scenario. We transform the multi-dimensional features of state and action into one dimensional vectors using PCA. Then we visualize the state action density. More darker represents agent visit these state and action more frequently.

the balance of safety and efficiency, this finding has a broader impact on not only improve efficient exploration strategy but also provide safe exploration.

Our work has limitations and future research is needed. Firstly, we will conduct extensive experiments on other tasks, such as Open-AI robotics tasks. Secondly, better balancing strategies can be explored, such as meta gradient method [8] to optimize the learning objectives. Another way is to try the method from computer vision field of dealing with different losses combination [22] to better optimize these weights of different loss terms.

REFERENCES

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[2] S. Levine, A. Kumar, G. Tucker, and J. Fu, "Offline reinforcement learning: Tutorial, review, and perspectives on open problems," *arXiv preprint arXiv:2005.01643*, 2020.

[3] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–9.

[4] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. Paixão, F. Mutz *et al.*, "Self-driving cars: A survey," *arXiv preprint arXiv:1901.04407*, 2019.

[5] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2174–2182.

[6] D. Silver, J. A. Bagnell, and A. Stentz, "Learning from demonstration for autonomous navigation in complex unstructured terrain," *The International Journal of Robotics Research*, vol. 29, no. 12, pp. 1565–1592, 2010.

[7] A. Sadat, M. Ren, A. Pokrovsky, Y.-C. Lin, E. Yumer, and R. Urtasun, "Jointly learnable behavior and trajectory planning for self-driving vehicles," *arXiv preprint arXiv:1910.04586*, 2019.

[8] A. Mehta, A. Subramanian, and A. Subramanian, "Learning end-to-end autonomous driving using guided auxiliary supervision," *arXiv preprint arXiv:1808.10393*, 2018.

[9] S. Fujimoto, E. Conti, M. Ghavamzadeh, and J. Pineau, "Benchmarking batch deep reinforcement learning algorithms," *arXiv preprint arXiv:1910.01708*, 2019.

[10] Y. Wu, G. Tucker, and O. Nachum, "Behavior regularized offline reinforcement learning," *arXiv preprint arXiv:1911.11361*, 2019.

[11] N. Jaques, A. Ghandeharioun, J. H. Shen, C. Ferguson, A. Lapedriza, N. Jones, S. Gu, and R. Picard, "Way off-policy batch deep reinforcement learning of implicit human preferences in dialog," *arXiv preprint arXiv:1907.00456*, 2019.

[12] T. Xu, Q. Liu, L. Zhao, and J. Peng, "Learning to explore via meta-policy gradient," in *International Conference on Machine Learning*, 2018, pp. 5463–5472.

[13] R. Kumaraswamy, M. Schlegel, A. White, and M. White, "Context-dependent upper-confidence bounds for directed exploration," *Advances in Neural Information Processing Systems*, vol. 31, pp. 4779–4789, 2018.

[14] S. Fujimoto, D. Meger, and D. Precup, "Off-policy deep reinforcement learning without exploration," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2052–2062.

[15] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

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

[17] Y.-C. Chang, N. Roohi, and S. Gao, "Neural lyapunov control," in *Advances in Neural Information Processing Systems*, 2019, pp. 3245–3254.

[18] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[19] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," *arXiv preprint arXiv:1808.00924*, 2018.

[20] B. Amos, L. Xu, and J. Z. Kolter, "Input convex neural networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 146–155.

[21] G. Manek and J. Z. Kolter, "Learning stable deep dynamics models," *arXiv preprint arXiv:2001.06116*, 2020.

[22] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482–7491.