PRACTICAL INVESTIGATION INTO SAFE REINFORCEMENT LEARNING WITH AUTONOMOUS DRIVE

Tao Sun
taosun.victor@gmail.com
SanDisk, a Western Digital Brand

ABSTRACT

Reinforcement learning is a popular and powerful tool to learn optimal policies concerning games and control systems. However, in order to find the optimal policy, most reinforcement learning algorithms would do a few trial-and-error explorations at the beginning of the learning stage. Concerning real world control systems, these trial-and-error explorations might violate the safety constraints and cause damages to the system. This problem could be more serious when it comes to safety-critical applications, such as autonomous drive. In this project, we try to implement a recently proposed practical approach to help the autonomous drive system learn its optimal policy safely. Specifically, our goal is to ensure the system's explorations to stay within the safety constraints during the overall learning stage. We present the implementation and corresponding results, together with some discussions.

1 INTRODUCTION

In the past few years, reinforcement learning (RL) has been explored in many environments, such as video games Mnih et al. (2015), Go games Silver et al. (2016), robotic design Kober et al. (2013) and so on. However, the RL algorithms normally provide an optimal policy only in the long term, and the immediate actions proposed might be unsafe and break the safety constraints. The unsafe actions might produce harmful results to the system. This is more serious when it comes to safety-critical areas. Thus safety for RL has been an open question in industrial applications.

Autonomous driving system falls into such kind of safety-critical area. Consider a self-driving car learning to drive with certain RL algorithms. At the very beginning of the learning stage, the RL algorithms might propose some unsafe actions based upon currently learned policy. Of course the unsatisfactory reward from such unsafe actions would help to improve the learned policy, however, the proposed action has to be applied in order for the system to take the negative feedback from the reward. Such unsafe actions might bring serious results, such as car accidents. Here is the dilemma, the RL algorithm has to explore the unsafe action space to improve its policy, but serious outcomes might come when such actions are carried out. Such situation is very common in real-world applications. Despite the powerful impact of RL algorithm, safety ensurance is an important problem to address.

In general, several works have been published to discuss the safety guarantees in RL applications. Berkenkamp et al. (2017) describes theoreical conditions under which safe explorations could be guaranteed for certain control systems. Some prior knowledge about the physical system and certain mathematical conditions should be satisfied. Basically a physical model describing the deterministic part of the dynamics should be known, above that a Lyapunov function should be identified for policy attraction region under certain Lipschitz continuity conditions. Despite the results revealed by mathematical descriptions, sometimes a physical model is hard to find for complicated systems. Another example is Achiam et al. (2017), in which a modified trust-region policy gradient method is used to solve the constrained policy optimization problem. In each iteration, the optimization algorithm projects the the policy to a safe feasibility set. The safety constraints are met in expectation, not every iteration, which would still cause some problems.

A more recent work Dalal et al. (2018) proposes a much simpler and practical approach. In this approach, the safety signal is defined as a measure concerning the safety constraints. Some prior

sampled data regarding state transition, actions taken and corresponding safety signals are dumped and used to train a model called "safety layer". The safety layer is used to predict the resultant safety signals based on the current state and proposed action, to decide whether the action should be revised before being carried out. The approach is pretty straightforward and easy to implement. In the paper, the authors claim that the approach solves step-wise safety problem. However, their examples presented are not really complicated, with low dimensional state space and simple environment.

This triggers the current project, whether the approach proposed by Dalal et al. (2018) could be applied to more complicated applications, specifically, autonomous drive, to ensure step-wise safety during reinforcement learning process. It is well known that autonomous drive is a complicated application with complex environment and high dimensional state space (image). We would like to adopt the proposed safety-layer approach to autonomous drive application, to see how it works.

2 Autonomous Drive with Reinforcement Learning

Autonomous drive has been a popular yet challeging topic today. Despite the fact that many efficient RL algorithms has been proposed, their applications to autonomous drive still suffer from compliated system setup and lack of powerful enough hardwares. Recently $Wayve.ai$ introduces a method to train a self-driving car both in simulations and in real world Kendall et al. (2018). They deploy the method with the input state as a single image taken by the front camera on the car, and the output action from the policy learned by RL algorithms. This is the first practical implementation regarding RL applied to drive a real car. The method could let the car learn to drive in one day. The major point concerning algorithms is that they extract the features from the input image to represent the state with a variational auto-encoder (VAE). The representation from the VAE has much lower dimension than the original input image, so that the corresponding neural networks have smaller sizes and thus, fewer parameters to tune. The light-weighted neural network is relatively easy to train compared with traditional approaches of heavy-weighted neural networks. The schematic view of the above statement is as Figure 1.

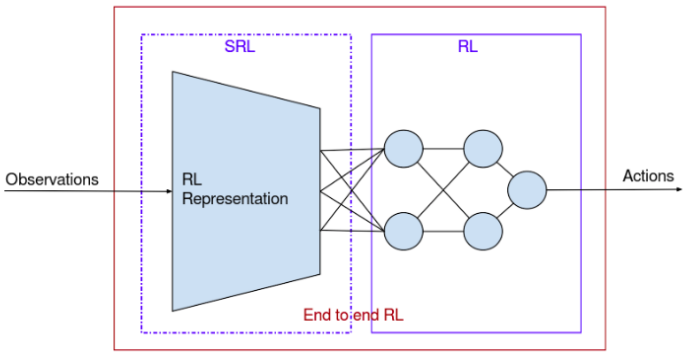


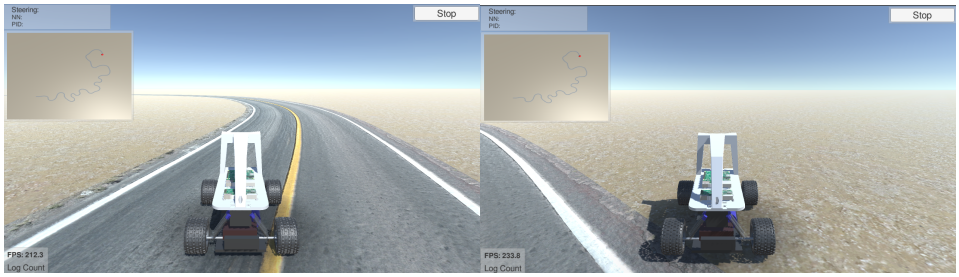
Figure 1: Extracting features from observations with state representation learning (SRL), then feeding the extracted features to reinforcement learning (RL) algorithms. The SRL used in this project is a VAE. Figure adapted from Raffin & Sokolkov (2019)

A practical implementation of Wayve.ai's approach Kendall et al. (2018) could be found as Raffin & Sokolkov (2019). In the implementation, the donkeyCar simulator Kramer (2018) is used as the environment for autonomous drive. The VAE is trained based on the extracted images from the front camera of the toy car. The implementation provides several RL algorithms to experiment with: PPO2 Schulman et al. (2017), DDPG Lillicrap et al. (2015) and SAC Haarnoja et al. (2018). The reward function is defined as in Equation 1, with w_1 and w_2 as predefined parameters. Basically the algorithm get small positive reward when traveling on the lane, and get large negative punishment when traveling off the lane. This reward would help the algorithm to improve its learned policy

based on positive and negative feedbacks. With the help of VAE and RL algorithms, the toy car learns to drive in a few minutes.

$$reward = \begin{cases} -10 - w_1 \times throttle & \text{when off the lane} \\ +1 + w_2 \times throttle & \text{when on the lane} \end{cases} \tag{1}$$

At the initial stage of training, several unsafe actions are observed, as shown in Figure 2. The RL algorithms needs to perform several unsatisfactory actions to get negative reward, before it could improve its policy to a proper state. These unsafe actions at the initial stage, however, are not desired. That's why we would like to explore ensuring safety during RL process.



(a) (b)

Figure 2: Unsafe actions taken during training process. (a) Left cross out of the lane. (b) Right cross out of the lane.

3 Safety Layer method

As stated in the Introduction section, Dalal et al. (2018) proposed the safety layer method. The method first defines the so-called "safety signal" as the physical quantity indicating safety critical measurement that must stay within some constraints. For example in autonomous drive application, the safety signal could be defined as the maximum deviation of the car traveling outside the lane.

The representations defined concerning the specific application in this project is as the following. The tuple to describe the problem is $(S, A, R, \gamma, C, \theta)$. S is the state space, here is the extracted features from the VAE. A is the action space as *(steering angle, throttle)*. R is the reward function, defined as Equation 1. $\gamma \in (0, 1)$ is the discount factor. $C = \{c : S \times A \to \mathbb{R}, c < C_{critical}\}$ is the safety constraint. Here c represents the safety signal, and is defined as **the car's maximum deviation off the lane**. $C_{critical}$ is the corresponding constrain criteria that must be satisfied. The last one θ is the parameters representing the policy. The problem could be stated as, the RL algorithm learns a policy θ that takes input from state space S and produces corresponding action as of A, then gets the reward as of R. At each step during learning, the safety constrain C must be satisfied.

$$\begin{aligned} \min_{\theta} \quad & \mathbb{E}(\sum_{t=0}^{\infty} \gamma^t R(s_t, \mu_\theta(s_t))) \\ \text{s.t.} \quad & c(s_t) \leq C_{critical} \end{aligned} \tag{2}$$

Directly solving Equation 2 is very difficult, since the physical model governing the mapping between the safety signal and state/action pair is generally unknown. The method proposes to use a linear approximation to represent the safety signal.

$$c(s') \approx c(s) + g(s, w)^\intercal a \tag{3}$$

where $g(s, w)$ is a neural network takes s as input and produces an output as a vector with the same dimention as a. The neural network $g(s, w)$ needs to be trained in advance, requiring sampling of

safety signal transitions and corresponding states/actions pairs. Imaging we have a set of samples $D = \{(s_j, a_j, s'_j)\}$, then the neural network could be trained as

$$arg\min_{w} \sum_{(s,a,s')\in D} (c_i(s') - (c_i(s) + g(s,w)^\intercal a))^2 \tag{4}$$

Then we could add a safety layer to the RL algorithm. Basically the actions produced by the RL algorithm are screened by the safety layer to satisfy the constraints. The process is shown as Figure 3.

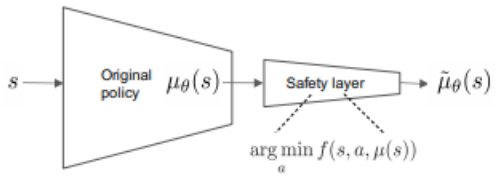


Figure 3: A safety layer added after the RL policy layer. The figure is adapted from Dalal et al. (2018).

Now Equation 3 becomes the following form. Obviously a pretrained safety layer has been added on top of the RL policy layer, so that each time the proposed action produced by the RL algorithm would be revised by the safety layer. As a result, corresponding constraints could be satisfied.

$$\begin{aligned} arg\min_{a} \quad & \frac{1}{2}|a - \mu_\theta(s)|^2 \\ \text{s.t.} \quad & c(s) + g(s,w)^\intercal a \leq C_{critical} \end{aligned} \tag{5}$$

A close form solution exists for the constrained optimization problem presented in Equation 5 (thanks for the materials learned from ECE 490!). The close form solution is presented as the following.

$$\lambda_p = \frac{g(s,w)^\intercal \mu_\theta(s) + c(s) - C_{critical}}{g(s,w)^\intercal g(s,w)}$$

$$\lambda^* = \begin{cases} \lambda_p & \text{when } \lambda_p > 0 \\ 0 & \text{when } \lambda_p \leq 0 \end{cases} \tag{6}$$

$$a^* = \mu_\theta(s) - \lambda^* g(s,w)$$

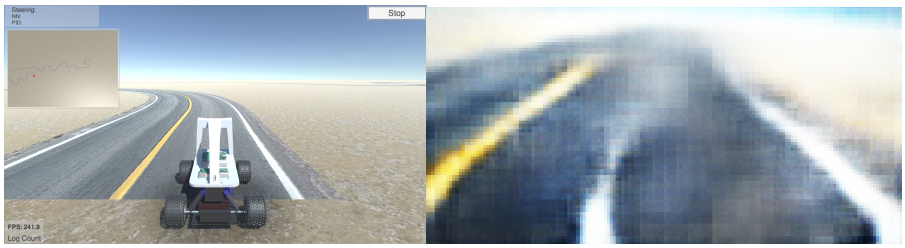
Equation 6 is basically a linear projection of the originally proposed action to the safe hyperplane. The method is pretty easy to be implemented. In the paper Dalal et al. (2018), the authors have tried several easy examples to show that the above proposed method works so that there are zero safety violations. However, those examples are easy ones, with low dimension state space and simple environment. When it comes to the autonomous drive, the cases become more complicated concerning state space and environment. We would implement this method to autonomous drive simulations to see how it works.

4 IMPLEMENTATION

In this section, we implement the method from Dalal et al. (2018) applied to autonomous drive system with a donkey car simulator. We have tried two examples, with the first example to be a simple environment and the second one to be a complicated environment. The method is tested on both with results presented.

4.1 SIMPLE ENVIRONMENT

The first example is a simple environment, as shown from Figure 4. The lane is straight at most parts, with only several smooth turns from time to time. The environment, front camera image and VAE image are shown in the figure. We could see the VAE image could reveal the major features from the camera image. Our procedure for the RL algorithm training is shown as Algorithm 1.



(a) Environment (b) VAE image

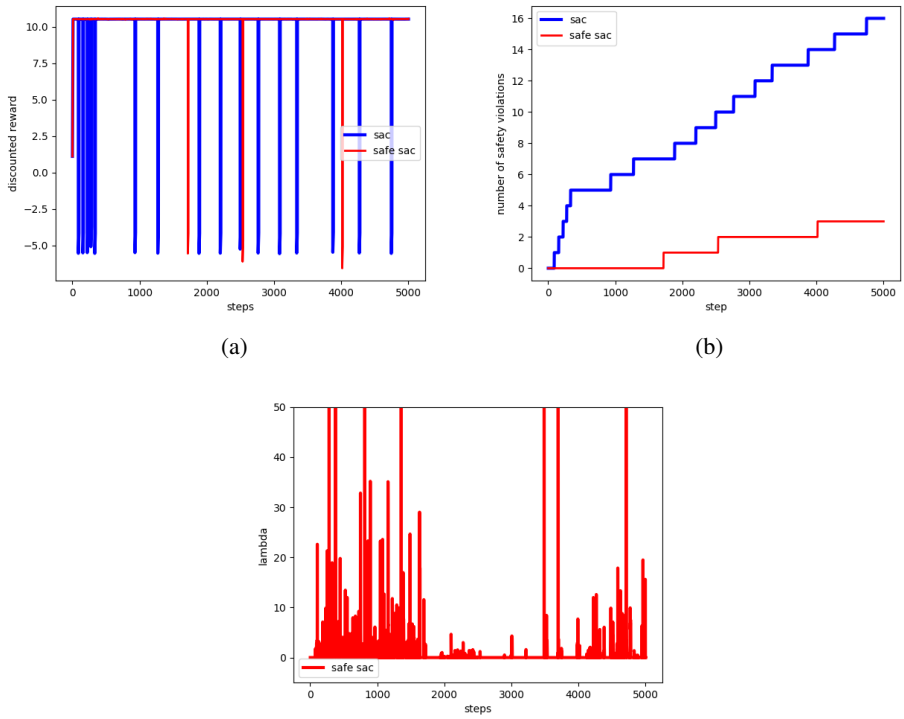
(c) Camera image

Figure 4: First example, a simple environment. (a) Schematic view of the environment. (b) Image input from VAE, a compressed feature. (c) Image input from the front camera.

Algorithm 1 The Safety Layer with Linear Approximation

Input: donkey car simulator
Step 1
Collect data tuples by random sampling:
$\{c$: safety signal,
s: state (VAE image),
a: action (steering angle / throttle),
c': new safety signal$\}$
Step 2
Construct a neural network as the safety layer $g(s, w)$
Train the safety layer according to Equation 4 with Gradient Descent
Step 3
for step $\in$ (1, learning steps) **do**
Propose the initial action μ with the input state s based on current RL policy
Correct μ according to Equation 6 to get action μ'
Carry out action μ' to obtain the next state and reward
Update the policy according to the RL algorithm
end for

Here we collect the data tuples with random sampling and several partially trained models, in order to cover the major properties of the environment. For the safety layer we constructed a neural network with 2 fully connected hidden layers, with the first layer having 64 neurons and the second layer having 16 neurons. For the RL algorithms, we choose SAC algorithm Haarnoja et al. (2018). We run the learning process (starting from a barely trained model) without and with the safety layer, and try to compare the outcomes of these two. The results are shown in Figure 5.



(a) (b)

(c)

Figure 5: Comparison of the RL training process with/without safety layer. (a) Discounted reward. (b) Number of violations. (c) the value of λ, applied to safety layer method.

From Figure 5 we could obviously see that without safety layer, there are a number of safety violations at the training process. Also Figure 5b reveals that for the case without the safety layer, the safety violations are pretty common at the early stage. With the learning process going on, the safety violations are not so common any more, and would appear after many steps. It is reasonable to predict that after many steps of training, the policy is improved and there will be no more safety violations. For the case with safety layer, we could see that for 5000 steps of training, the number of safety violations is much smaller than the case without the safety layer. Moreover, at the very beginning of the training stage, there are no safety violations. Until at the middle and later stage, there are some safety violations happen. This reveals the fact that the safety layer do help to prevent safety violations at the early learning stage, and in general it ensures the safety. The safety violations appearing at the later time might be caused by not enough sampling of the state, since the lane is very long and the environment varies to some extent. For the case with the safety layer, the value of λ evolving with training steps is showin in Figure 5c. We could see that at the initial learning stage, the value of λ is pretty large, showing the originally proposed action from the partially trained policy would severly violate the safety constraint, and the safety layer does spend effort in bringing the car back. In the middle of the training process, the value of λ is very small, showing the RL algorithm has updated its polity to a satisfactory state so that there are fewer safety violations. Then in the later part of the learning process, the value of λ goes up again, showing the environment is varying.

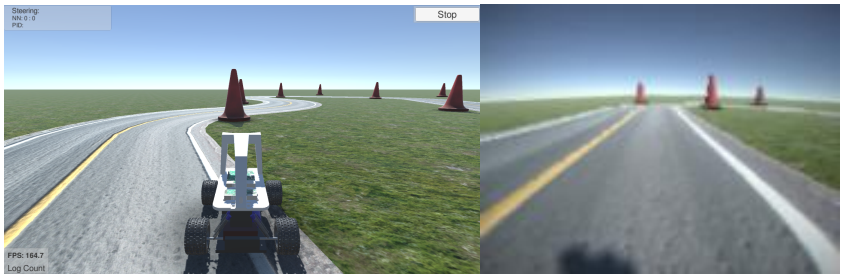
Two videos are available for the RL training process without/with the safety layer Sun (2019a), Sun (2019b). From the videos, it is very clear that for the training process with the safety layer, each time when the car is about to violate the safety constraint (gets off the lane), the safety layer will simply push it back to the track. As a contrast, for the training process without the safety layer, there is no such luck. It will gets off the track from time to time.

The paper proposing the method Dalal et al. (2018) shows that in the two example games, they achieve zero safety violations with the help of the safety layer. In our current case concerning

autonomous drive system, since the environment is a lot more complicated, the safety layer could only confine the violation to some extent. But in general it works to reduce the number of safety violations.

4.2 Complicated Environment

Next we try a more complicated environment. The environment is shown in Figure 6. There are a number of continuous sharp turns on the way, and barely any simple straightforward paths exist. Also there are lots of obstacles on the road, making the environment more challenging. For the training stage of the safety layer, simple random sampling might not be able to capture the complicated environment. Together with the random sampling, we also adopt a trained model to drive the car along the road for a very long time (even the trained model violates the safety constraints for several times). The collected data are combined together to train the safety layer.



(a) Environment (b) Camera image

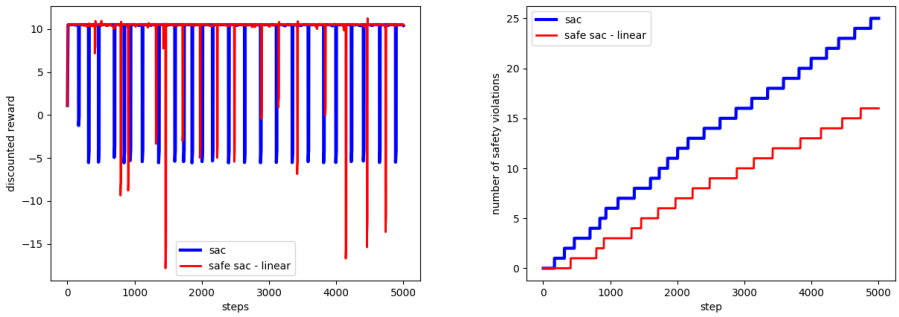
Figure 6: Second example, a complicated environment. (a) Schematic view of the environment. (b) Image input from the front camera.

The results for the RL training process without/with safety layer (start from a barely trained model) under linear approximation is shown in Figure 7. We could see that even with the safety layer enabled, there are still many safety violations during the RL training stage, although the violations are slightly fewer than the case without the safety layer. For this complicated environment, we see that the safety layer does not provide very good result concerning eliminating the safety violations. The reason could be that, in this complicated environment, there are lots of sharp turns (as in Figure 7c). When the car is about to violate the left boundary, the safety layer would push it back to the center of the track. However, since this is a sharp turn, the car is approaching the right boundary of the track very quickly. When it is about to violate the right boundary, although the safety layer still tries to push it backward, the effect is no longer enough to prevent the car from getting off the lane. This shows the drawback of the current safety layer method. The corresponding video showing this could be found at Sun (2019c).

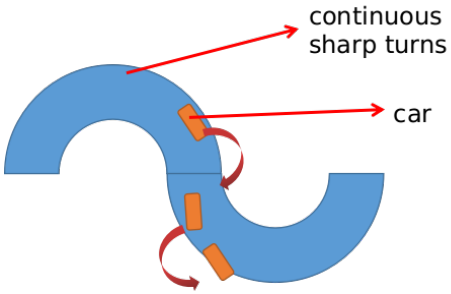
We tried to introduce more complicated model to describe the relation between the safety signal and the state/action pairs. Currently only linear approximation is used. Here we try to introduce the quadratic approximation.

$$c(s') \approx c(s) + g_1(s, w_1)^\intercal a + \frac{1}{2} a^\intercal g_2(s, w_2) a$$

$$arg \min_{w_1, w_2} \sum_{(s,a,s') \in D} (c_i(s') - (c(s) + g_1(s, w_1)^\intercal a + \frac{1}{2} a^\intercal g_2(s, w_2) a)^2 \tag{7}$$



(a) (b)



(c)

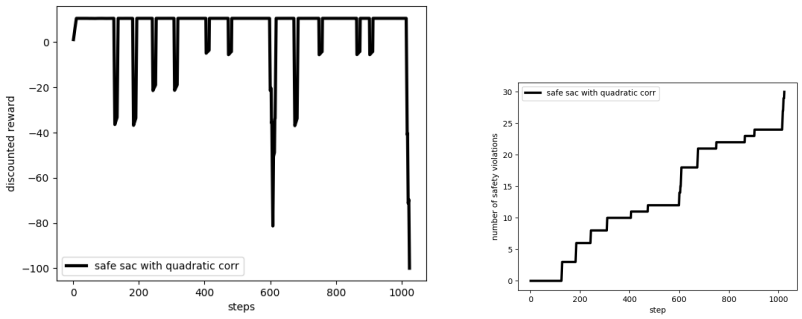
Figure 7: Comparison of the RL training process with/without safety layer for complicated environment. (a) Discounted reward. (b) Number of violations. (c) A schematic view of continuous sharp turns.

Where $g_1(s, w_1)$ is a vector as described before, and $g_2(s, w_2)$ is a symmetric matrix. Both g_1 and g_2 are produced by neural networks. Then the safety layer would revise the proposed action according to the following.

$$\begin{aligned} arg\min_{a} \quad & \frac{1}{2}|a - \mu_\theta(s)|^2 \\ \text{s.t.} \quad & c(s) + g_1(s, w_1)^\intercal a + \frac{1}{2} a^\intercal g_2(s, w_2) a \leq C_{critical} \end{aligned} \tag{8}$$

Equation 8 is a quadratic constrained quadratic programming (QCQP) problem. Unfortunately there are no close form solutions to the QCQP problem. Here we use some heuristics to solve this problem, such as Park & Boyd (2017). Basically each time the current policy proposes the action, we solve Equation 8 with QCQP solver Park & Boyd (2017) to get the revised action and then apply it to the car. The QCQP problem might not always be solvable. In case of failures from the solver, we take the originally proposed action. We have tried this quadratic approximation for safety layer for our complicated environment. However, it does not work. The results are shown in Figure 8.

Figure 8 shows that with the quadratic approximation of the safety layer, the performance becomes even worse than the non-safety-layer case. This shows that under current setup, more complicated approximation for the safety signals would not enhance the performance, since it brings new trouble (e.g. QCQP not solvable). More energies should be spent on the RL algorithm itself, for instance, redesign the reward function to reduce the impact of current speed, so that the learned policy would not drive the car too fast, in order to take in-time actions to avoid safety violations with sudden appearance of sharp turns.



(a) (b)

Figure 8: Comparison of the RL training process with/without safety layer (quadratic approximation) for complicated environment. (a) Discounted reward. (b) Number of violations.

5 CONCLUSION

In this project, we implement the linearly approximated safety layer method proposed by Dalal et al. (2018) to investigate its effect on autonomous drive applications. Concerning simple environment, the method does bring in much fewer safety violations (although still not zero violations). When it comes to complicated environment, the RL training process with safety layer still has many safety violations, although fewer than the case without the safety layer. We then bring in the quadratic approximation for the safety layer, only to see it makes the performance even worse.

Looks like the proposed simple safety layer method still has some issues when applied to RL problems with complicated environments. The safety guarantee at the inital stage of the RL process still remains an open questions. More considerations should be laid concerning design of the RL algorithm and the correction of the proposed actions to produce a general enough approach for safety guarantees.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 22–31. JMLR. org, 2017.

Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. Safe model-based reinforcement learning with stability guarantees. In *Advances in neural information processing systems*, pp. 908–918, 2017.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. *arXiv preprint arXiv:1807.00412*, 2018.

Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Tawn Kramer. Self driving car sandbox. `https://github.com/tawnkramer/sdsandbox`, 2018.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Jaehyun Park and Stephen Boyd. General heuristics for nonconvex quadratically constrained quadratic programming. *arXiv preprint arXiv:1703.07870*, 2017.

Antonin Raffin and Roma Sokolkov. Learning to drive smoothly in minutes. `https://github.com/araffin/learning-to-drive-in-5-minutes/`, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Tao Sun. Safety layer method applied to autonomous drive. `https://github.com/taosun-victor/drive_safeRL/blob/master/store-simple-linear-use/sac-03.gif`, 2019a.

Tao Sun. Safety layer method applied to autonomous drive. `https://github.com/taosun-victor/drive_safeRL/blob/master/store-simple-linear-use/safe-sac-03.gif`, 2019b.

Tao Sun. Safety layer method applied to autonomous drive. `https://github.com/taosun-victor/drive_safeRL/blob/master/store-complex-linear-use/complex-safe-sac.gif`, 2019c.