Safe Reinforcement Learning through Buffer and Barrier Functions for Autonomous Driving

Zipei Zhao
zzhao115@jh.edu

Yue Wan
ywan23@jh.edu

1 Introduction

Autonomous driving presents complex challenges requiring rapid decision-making in dynamic environments. Reinforcement learning (RL) has emerged as a promising approach, offering flexibility in solving control problems. However, traditional RL methods often ignore safety considerations during exploration, leading to unacceptable unsafe behaviors in real-world scenarios. To address this, Safe RL incorporates safety constraints to ensure exploration and performance optimization within safe boundaries.

This work explores a framework combining Trust Region Policy Optimization (TRPO), Control Barrier Functions (CBFs), and a dual-buffer mechanism. TRPO stabilizes policy updates, CBFs enforce real-time safety constraints, and the dual-buffer mechanism enhances sample efficiency by prioritizing critical transitions during training. Together, these methods provide a robust approach to safe RL for autonomous driving.

We propose a framework that:

- **Integrates TRPO, CBFs, and a buffer mechanism** to address safety and learning efficiency in autonomous driving.
- **Implements a car-following simulation** involving five vehicles to evaluate safety, reward, and learning stability.
- **Performs a comparative analysis** of TRPO, TRPO-CBF, and TRPO-CBF with buffers to highlight the strengths and limitations of each approach.

Our experiments demonstrate that TRPO-CBF achieves the best performance in terms of maintaining safety and reward stability. While the buffer mechanism avoids collisions, it introduces instability, likely due to suboptimal parameter tuning and a lack of computational sources.

2 Preliminaries

2.1 Related Work

Reinforcement learning (RL) has demonstrated success in solving complex control problems in domains like robotics and autonomous driving. However, traditional RL lacks safety guarantees during exploration, which is critical for safety-sensitive tasks such as autonomous driving. Safe reinforcement learning (Safe RL) addresses this gap by integrating safety constraints into the learning process. Surveys by García and Fernández (2015) [1] and Gu et al. (2024) [2] highlight methods such as constrained optimization, risk-sensitive exploration, and layered safety strategies, emphasizing their importance for ensuring safety without compromising performance.

Control Barrier Functions (CBFs) provide a mathematical framework to enforce safety by ensuring forward invariance of predefined safe sets. Ames et al. (2015)[3] introduced the foundational theory of CBFs, while Cheng et al. (2019)[4] integrated CBFs with RL to guarantee safety during real-time action execution. Complementary to this, Zhang et al. (2019)[5] proposed a dual replay buffer mechanism to balance learning efficiency and generalization, prioritizing transitions that improve safety. After this, Nageshrao et al. (2019)[6] demonstrated the dual-buffer mechanism's ability to enhance Safe RL by balancing safe learning and exploration. Together, these methods provide robust tools for combining real-time safety enforcement with efficient learning.

Our work combines Trust Region Policy Optimization (TRPO), CBFs, and a dual-buffer mechanism to address safety and efficiency in RL. TRPO, originally implemented in Cheng et al. (2019)[4], ensures stable

policy updates, while CBFs enforce safety constraints during execution. The dual-buffer mechanism, as refined by Nageshrao et al. (2019)[6], prioritizes critical transitions, enhancing sample efficiency. Compared to methods like Constrained Policy Optimization (Achiam et al., 2017)[7] and Lyapunov-based RL (Chow and Pavone, 2018)[8], our framework uniquely integrates real-time safety guarantees with efficient learning, making it tailored for safety-critical domains like autonomous driving.

2.2 Problem Formulation

The objective is to address an autonomous driving scenario where a reinforcement learning (RL) agent must learn to safely and efficiently control a vehicle in a highway environment under safety constraints. The problem is modeled as an Infinite Horizon Markov Decision Process (MDP), defined by the tuple $(S, A, P, r, \rho_0, \gamma)$, where:

- S is the state space, which includes the positions, velocities, and relative distances of the ego vehicle and surrounding vehicles,
- A is the action space, representing the continuous acceleration and steering controls of the ego vehicle,
- $P(s'|s, a)$ defines the transition dynamics, which describe how the system transitions from state s to state s' after taking action a,
- $r(s, a)$ is the reward function that evaluates the immediate performance of the ego vehicle based on safety, efficiency, and comfort,
- ρ_0 is the initial state distribution, representing the starting positions and velocities of the vehicles, and
- $\gamma \in (0, 1)$ is the discount factor, balancing the importance of immediate versus future rewards.

In the experimental setting, the environment consists of a highway scenario with five cars in a row. The ego vehicle must safely navigate while maintaining reasonable speeds, avoiding collisions, and adhering to traffic rules.

The goal of the RL agent is to learn a stochastic policy $\pi(a|s)$ that maximizes the expected discounted cumulative reward:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \tag{1}$$

where $\tau = (s_0, a_0, s_1, a_1, \ldots)$ denotes a trajectory generated by the policy π.

State Space

The state $s \in S$ includes:

- Ego vehicle's position, velocity, and lane index,
- Relative positions and velocities of the surrounding vehicles, and
- Lane-level information such as the distance to the nearest car ahead and behind in the same or adjacent lanes.

Action Space

The action $a \in A$ consists of continuous control inputs:

- Acceleration (or deceleration) to adjust speed, and
- Steering angle to maintain or change lanes.

Reward Function

The reward function $r(s, a)$ is designed to balance safety, efficiency, and driving comfort:

- A positive reward for maintaining a target speed within safe limits,
- A penalty for collisions or driving too close to other vehicles,
- A penalty for abrupt acceleration or sharp steering actions to encourage smooth driving, and
- A penalty for deviating from the desired lane or driving in an unsafe manner.

By structuring the problem in this way, the RL agent is tasked with achieving efficient and safe driving behavior while adapting to dynamic traffic conditions. Our experiments follow the convention of the RL Problem Formulation, while changing certain actions and define our own reward functions.

3 Methods

The proposed methodology integrates Trust Region Policy Optimization (TRPO), Control Barrier Functions (CBFs), Gaussian Processes (GPs), and a dual-buffer mechanism to achieve safe and efficient reinforcement learning. TRPO is used to optimize the policy, generating actions that aim to maximize rewards. The CBF ensures safety by filtering these actions and projecting unsafe ones into a predefined safe set. GPs support the CBF by modeling system dynamics and quantifying uncertainties to account for model inaccuracies. To improve learning efficiency, transitions are stored in two buffers—safe transitions and those corrected by the CBF—for prioritized replay during policy updates. Together, these components work in a pipeline to ensure safe and efficient policy learning. The following sections provide detailed descriptions of each component and their integration.

3.1 Trust Region Policy Optimization (TRPO)

Trust Region Policy Optimization (TRPO) is a policy gradient method designed to optimize policies in reinforcement learning while ensuring stability and monotonic improvement. In our framework, TRPO is used to iteratively improve the agent's policy $\pi_\theta(a|s)$, which determines the probability of taking action a in state s, by maximizing the expected cumulative reward. The optimization problem is defined as:

$$\max_\theta \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_\theta}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^{\pi_{\theta_{\text{old}}}}(s,a)\right], \tag{2}$$

where $A^{\pi_{\theta_{\text{old}}}}(s,a)$ is the advantage function under the previous policy $\pi_{\theta_{\text{old}}}$, and ρ^π is the state visitation distribution under policy π.

Stability with Trust Region

To ensure stable updates and prevent huge changes to the policy, TRPO constrains the Kullback-Leibler (KL) divergence between the new policy π_θ and the old policy $\pi_{\theta_{\text{old}}}$:

$$D_{\text{KL}}(\pi_{\theta_{\text{old}}}\|\pi_\theta) = \mathbb{E}_{s\sim\rho^\pi}\left[\sum_a \pi_\theta(a|s)\log\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}\right] \le \delta, \tag{3}$$

where δ is a predefined trust region threshold that controls the limit of each policy update.

Surrogate Objective

TRPO reformulates the optimization problem into a constrained surrogate objective:

$$\max_\theta \mathbb{E}_{s\sim\rho^\pi, a\sim\pi_{\theta_{\text{old}}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^{\pi_{\theta_{\text{old}}}}(s,a)\right], \quad \text{s.t.} \quad D_{\text{KL}}(\pi_{\theta_{\text{old}}}\|\pi_\theta) \le \delta. \tag{4}$$

This surrogate objective ensures that the policy improves in regions where the advantage function $A^{\pi_{\theta_{\text{old}}}}(s,a)$ is positive while staying close to the old policy.

Practical Implementation

The constrained optimization problem is solved approximately by using the conjugate gradient method. The update step for the policy parameters θ is given by:

$$\theta_{\text{new}} = \theta_{\text{old}} + \alpha \nabla L(\theta), \tag{5}$$

where $L(\theta)$ is the linearized surrogate objective to be defined later, and α is a step size determined by a backtracking line search to satisfy the KL constraint.

TRPO generates the proposed action u^{RL} for the agent based on the current policy in this framework.

3.2 Control Barrier Functions (CBFs)

Control Barrier Functions (CBFs) are mathematical tools used to enforce safety constraints by ensuring that the system's state remains within a predefined safe set during execution. The safe set $C \subseteq S$ is defined as the superlevel set of a continuously differentiable scalar function $h(s)$:

$$C = \{s \in S : h(s) \geq 0\}. \tag{6}$$

Here, $h(s)$ quantifies the degree to which a state s satisfies the safety constraint, where $h(s) \geq 0$ indicates a safe state, and $h(s) < 0$ denotes an unsafe state.

To maintain safety, CBFs enforce forward invariance of the safe set C. This is achieved by constraining the control input a such that the safety condition is preserved over time. Specifically, the following inequality must hold:

$$\sup_{a \in A} \left[\frac{\partial h(s)}{\partial s} (f(s) + g(s)a) + \alpha h(s) \right] \geq 0, \tag{7}$$

where:

- $f(s)$ represents the nominal dynamics of the system,
- $g(s)$ maps the control input a to its effect on the system dynamics,
- $\alpha > 0$ is a tunable parameter that dictates the rate at which the system should approach safety.

If the action u^{RL} proposed by the RL policy violates this condition, the CBF intervenes to compute a safe action u^{safe}. This is formulated as a quadratic programming (QP) problem:

$$u^{\text{safe}} = \arg\min_{a} \|a - u^{\text{RL}}\|^2, \quad \text{s.t.} \quad \frac{\partial h(s)}{\partial s} (f(s) + g(s)a) + \alpha h(s) \geq 0. \tag{8}$$

This optimization ensures that u^{safe} is as close as possible to u^{RL} while maintaining the safety constraints defined by $h(s)$.

In our framework, the CBF serves as a real-time safety filter, ensuring that every action executed by the system adheres to the safety requirements. Integrating this mechanism allows the agent to explore and learn without violating critical constraints, enabling safe RL for continuous control tasks.

3.3 Gaussian Processes (GPs)

Gaussian Processes (GPs) are probabilistic models used to approximate unknown functions and quantify uncertainty, making them ideal for modeling the unknown dynamics $d(s)$ in control tasks. The system's dynamics are represented as:

$$s_{t+1} = f(s_t) + g(s_t)a_t + d(s_t), \tag{9}$$

where:

- $f(s_t)$ and $g(s_t)$ are the known nominal dynamics,
- $d(s_t)$ is the unknown component that needs to be modeled.

GPs define a prior distribution over $d(s)$ with a mean function $\mu_d(s)$ and a covariance function $k(s, s')$. For any state s, the GP provides:

$$d(s) \sim \mathcal{GP}(\mu_d(s), k(s, s')). \tag{10}$$

GP Predictions

Given a set of observed data points $\mathcal{D} = \{(s_i, d_i)\}_{i=1}^n$, where d_i represents the unknown dynamics at state s_i, the GP predicts the mean and variance of $d(s)$ at a new input s:

$$\mu_d(s) = k^\top (K + \sigma_{\text{noise}}^2 I)^{-1} y, \tag{11}$$

$$\sigma_d^2(s) = k(s, s) - k^\top (K + \sigma_{\text{noise}}^2 I)^{-1} k, \tag{12}$$

where:

- K is the covariance matrix of the observed data based on the kernel function $k(s, s')$,
- k is the covariance vector between the new input s and the observed data,
- y is the vector of observed outputs d_i,
- σ_{noise}^2 is the variance of observation noise.

Integration with CBF

The GP predictions are used to refine the safety constraints enforced by the Control Barrier Function (CBF). Specifically, the mean prediction $\mu_d(s)$ and the uncertainty $\sigma_d(s)$ are incorporated into the CBF's constraint:

$$h(f(s) + g(s)a + \mu_d(s) - k\sigma_d(s)) + \alpha h(s) \geq 0, \tag{13}$$

where k is a confidence parameter that scales the uncertainty. This integration ensures that the CBF accounts for the expected value and uncertainty of the unknown dynamics when enforcing safety constraints.

By incorporating GPs into the CBF, the framework can handle partially modeled dynamics and account for uncertainties, ensuring robust safety guarantees in dynamic.

3.4 Buffer Mechanism

The dual-buffer mechanism is incorporated into the framework to enhance learning efficiency by prioritizing critical transitions during policy updates. The mechanism consists of two buffers:

- **Safe Buffer (Buf_S)**: Stores transitions where the proposed action from the RL policy was deemed safe and required no intervention by the Control Barrier Function (CBF).
- **Collision Buffer (Buf_C)**: Stores transitions where the proposed action was unsafe and corrected by the CBF to ensure safety.

Storing Transitions

During training, after executing an action u_t^{safe}, a transition tuple $(s_t, u_t^{\text{RL}}, r_t, s_{t+1})$ is generated, where:

- s_t and s_{t+1} are the states before and after the action,
- u_t^{RL} is the action proposed by the RL policy,
- u_t^{safe} is the action executed after CBF correction,
- r_t is the observed reward.

Transitions are stored in the buffers based on the relationship between u_t^{RL} and u_t^{safe}:

- If $u_t^{\text{RL}} = u_t^{\text{safe}}$, the transition is stored in the **safe buffer** (Buf_S).
- If $u_t^{\text{RL}} \neq u_t^{\text{safe}}$, indicating that the action was corrected by the CBF, the transition is stored in the **collision buffer** (Buf_C).

Sampling Transitions for Policy Updates

During policy updates, transitions are sampled from both buffers to train the RL policy:

$$\mathcal{B} = \mathcal{B}_S \cup \mathcal{B}_C, \tag{14}$$

where:

- $\mathcal{B}_S$ is a minibatch sampled from Buf_S,
- $\mathcal{B}_C$ is a minibatch sampled from Buf_C.

To ensure balanced learning, the proportion of samples from each buffer is dynamically adjusted. Transitions from Buf_C are often weighted more heavily during policy updates to penalize unsafe actions and encourage the RL policy to avoid proposing unsafe actions in the future.

Incorporation into Policy Updates

The RL policy is updated using the sampled transitions $\mathcal{B}$ with a modified loss function that includes penalties for transitions from Buf_C. The loss function in TRPO is modified as:

$$L(\theta) = \mathbb{E}_{(s_t,a_t,r_t)\in\mathcal{B}}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) A^\pi(s_t,a_t)\right] - \lambda \mathbb{E}_{(s_t,a_t,r_t)\in\mathcal{B}_C}\left[\|a_t - u_t^{\text{safe}}\|^2\right], \tag{15}$$

where λ is a penalty weight applied to transitions from Buf_C.

By integrating the dual-buffer mechanism, the framework ensures that the RL policy improves both safety and reward maximization, aligning with the overall goal of safe and efficient RL.

3.5 Combined Method

The proposed framework integrates Trust Region Policy Optimization (TRPO), Control Barrier Functions (CBFs), Gaussian Processes (GPs), and a dual-buffer mechanism into a unified methodology for safe and efficient reinforcement learning. Each component plays a distinct role in the framework, ensuring that the agent learns an optimal policy while maintaining safety throughout the training process. Figure 1 is the flowchart for the combined method:

Flow of the Framework

1. **Policy Action Generation (TRPO):** The TRPO algorithm generates a proposed action u^{RL} based on the current policy $\pi_\theta(a|s)$. This action is aimed at maximizing the expected cumulative reward while adhering to the trust region constraints for stable updates (see Equation 4).

2. **Safety Check (CBF + GP):** The proposed action u^{RL} is passed to the Control Barrier Function (CBF), which ensures the safety of the action. The CBF uses the system's nominal dynamics $f(s)$ and $g(s)$, as well as the GP predictions for the unknown dynamics $\mu_d(s)$ and $\sigma_d(s)$, to enforce the safety constraint defined in Equation 13. If u^{RL} violates this constraint, the CBF computes a corrected safe action u^{safe} by solving the quadratic programming (QP) problem described in Equation 8.

3. **Action Execution:** The safe action u^{safe} is executed in the environment, transitioning the system to the next state s_{t+1} and generating a reward r_t.

4. **Transition Storage (Buffers):** The transition tuple $(s_t, u_t^{\text{RL}}, r_t, s_{t+1})$ is stored in one of two buffers:

- **Safe Buffer (Buf_S):** If $u^{\text{RL}} = u^{\text{safe}}$, the transition is stored in the safe buffer.
- **Collision Buffer (Buf_C):** If $u^{\text{RL}} \neq u^{\text{safe}}$, the transition is stored in the collision buffer.

5. **Policy Update:** During policy updates, transitions are sampled from both buffers. The safe buffer transitions reinforce correct behaviors, while the collision buffer transitions penalize unsafe actions. A modified loss function incorporates penalties for unsafe actions to guide the policy toward safer exploration, as defined in Equation 15:

By integrating these components, the framework achieves a balance between reward maximization, safety guarantees, and learning efficiency.

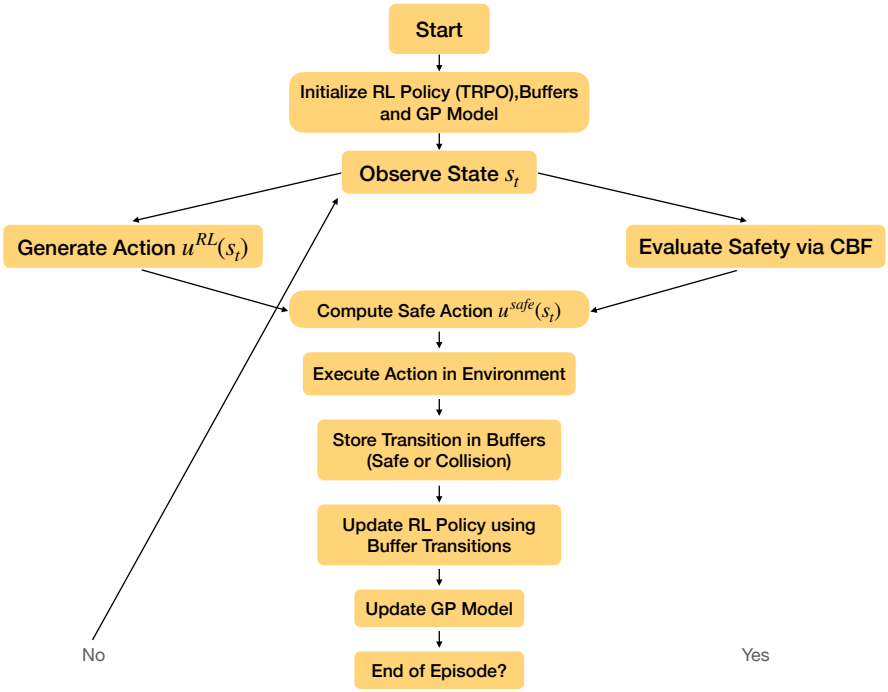

Figure 1: Flowchart of the proposed framework

Algorithm

Algorithm 1 RL-CBF Algorithm with Buffer Mechanism

Initialize: RL policy π_0^{RL}, GP model, safe buffer (Buf_S), collision buffer (Buf_C), and state $s_0 \sim \rho_0$.
for each episode **do**
  for each timestep t **do**
    Generate action $u_0^{RL}(s_t)$ from π_0^{RL}.
    Solve for $u_0^{CBF}(s_t)$ (Equation 8).
    Deploy $u_0(s_t) = u_0^{RL}(s_t) + u_0^{CBF}(s_t)$.
    Observe (s_t, u_0, r_t, s_{t+1}).
    if $u_0^{RL}(s_t) = u_0(s_t)$ **then**
      Store in Buf_S.
    else
      Store in Buf_C.
    end if
    Update GP model using Equation 11 and 12.
  end for
  Sample transitions from Buf_S and Buf_C for minibatch $\mathcal{B}$.
  Update π_k^{RL} using modified loss (Equation 15).
  Train approximation $u_{\phi_k}^{\text{bar}}$ for prior CBF controllers.
  for each timestep t **do**
    Generate action $u_k^{RL}(s_t) + u_{\phi_k}^{\text{bar}}(s_t)$.
    Solve for $u_k^{CBF}(s_t)$ (Equation 8).
    Deploy $u_k(s_t) = u_k^{RL}(s_t) + u_{\phi_k}^{\text{bar}}(s_t) + u_k^{CBF}(s_t)$.
    Observe and store transitions in Buf_S or Buf_C.
  end for
  Update GP model and increment k.
end for
Return: $\pi_k^{RL}, u_{\phi_k}^{\text{bar}}, u_k^{CBF}$.

4 Experiment

4.1 Experimental Setups

We implement our algorithm in a car-following simulation as Figure 2. We consider a chain of five cars following each other on a straight road and the fourth car is controlled to accelerate or decelerate. The goal is to train a policy that maximizes fuel efficiency when traffic congestion without meeting collisions.

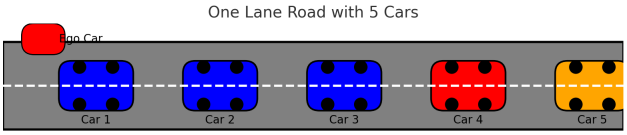

Figure 2: The 4th car in the row is the controlled vehicle, it uses a crude nominal model of the dynamics of other vehicles to initialize its learning; Cars 1, 2, and 3 maintain reasonable speeds and gaps using pre-defined dynamics, Car 5 reacts similarly to cars ahead, following safe braking and accelerating behavior.

Car Dynamics

The car dynamics follow the equation from [9]

$$\begin{bmatrix} \dot{s}^{(i)} \\ \dot{v}^{(i)} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 \\ 0 & -k_d \end{bmatrix} \begin{bmatrix} s^{(i)} \\ v^{(i)} \end{bmatrix}}_{f(s_t)} + \underbrace{\begin{bmatrix} 0 \\ 1 \end{bmatrix} a}_{g(s_t)a}, \tag{16}$$

$f(s_t)$ and $g(s_t)$ represent the state-dependent and control-dependent components of the dynamics, respectively.

The fourth car knows the position, velocity, and acceleration of other cars while it has a crude model that $k_d = 0$ and inaccurate models of other cars.

For the fourth car, $k_d = 0$, meaning the crude model assumes no natural damping in the velocity. Thus:

$$f(s_t) = \begin{bmatrix} v^{(i)} \\ 0 \end{bmatrix}.$$

Reward Function

The reward function follows the equation from [9]

$$r = -\sum_{t=1}^{T} [v_t^{(4)} \max((a_t^{(4)}), 0) + \sum_{i=3}^{4} G_i(\frac{500}{s_t^{(i)} - s_t^{(i+1)}})], \tag{17}$$

where

$$G_m(x) = \begin{cases} |x| \text{ if } s^{(m)} - s^{(m+1)} \leq 3 \\ 0 \text{ otherwise} \end{cases}. \tag{18}$$

The function optimizes fuel efficiency and encourages cars to keep a 3-meter distance from others.

With the buffer, the reward is updated that

$$y_j = \begin{cases} r_{j+1} \text{ if sample is from } Buf_c \\ r_{j+1} + \gamma r \text{ if sample is from } Buf_s \end{cases}. \tag{19}$$

4.2 Experimental Results

We compare the performance of the original TRPO algorithm, the TRPO algorithm with CBF and the modified TRPO algorithm with CBF and buffers. The performance is measured in terms of the proximity to collision and the average episode reward. Each algorithm is run for 200 episodes each time and run four times to achieve an average result.

Proximity to Collision

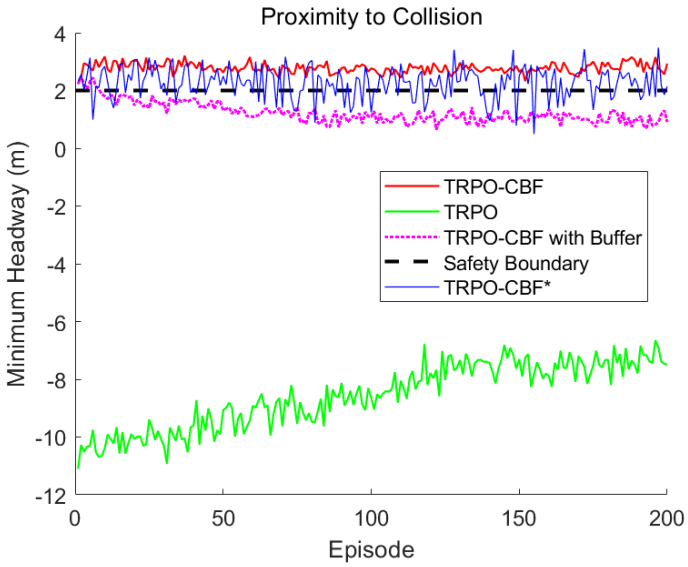

Figure 3: The proximity to collision of different algorithms in 200 episodes. The black line stands for safety boundary and values below it are exit from safe. Values below zero mean collisions.

The result is shown in Fig.3. The x-axis represents the learning episode and the y-axis represents the minimum headway between cars during each learning episode.

The figure shows that the simple TRPO algorithm without CBF always violates safety and meets collision. Other algorithms avoid collisions successfully. However, the TRPO-CBF with buffer algorithm fails to stay in the safety set with additional buffers.

TRPO-CBF* and TRPO-CBF are the same algorithms run on different devices. It can be found that TRPO-CBF* performs worse than TRPO-CBF. TRPO-CBF always maintains in the safe set, while TRPO-CBF* has bigger fluctuations and sometimes exits from the safe set.

Average Episode Reward

The result is shown in Fig.4. The x-axis represents the learning episode and the y-axis represents the average episode reward. Shaded error bar is used to display the stability of algorithms. TRPO algorithm without CBF is not considered in this comparison because it always violates safety constraints, which is not the desired situation.

We can find that TRPO-CBF method has better performance in terms of the reward values and stability. TRPO-CBF with buffer shows great instability during the beginning of the learning process and fails to reach a final stable state within 200 episodes.

TRPO-CBF* and TRPO-CBF are the same algorithm run on different devices, as in the previous part. They demonstrate distinct performance regarding reward values, stability, and learning speed. TRPO-CBF* performs worse than TRPO-CBF in that its average rewards are lower, and the shaded error is bigger, implying more fluctuations. Besides, it also fails to reach a final stable state within the 200 episodes.

5 Discussion

The results of previous experiments, which simulate a five-car-following situation, show that applying CBF to MDP truly avoids collisions. The TRPO-CBF method performs best considering the proximity to collisions and reward. Using additional buffers to distinguish between safe and unsafe cases seems unable to improve the existing TRPO-CBF method, though it successfully avoids collision cases. A potential reason for this limitation might lie in the tunable parameters of the loss function. The parameters used to balance safe and unsafe transitions in the buffer mechanism may not be optimally configured, leading to instability in the learning process.

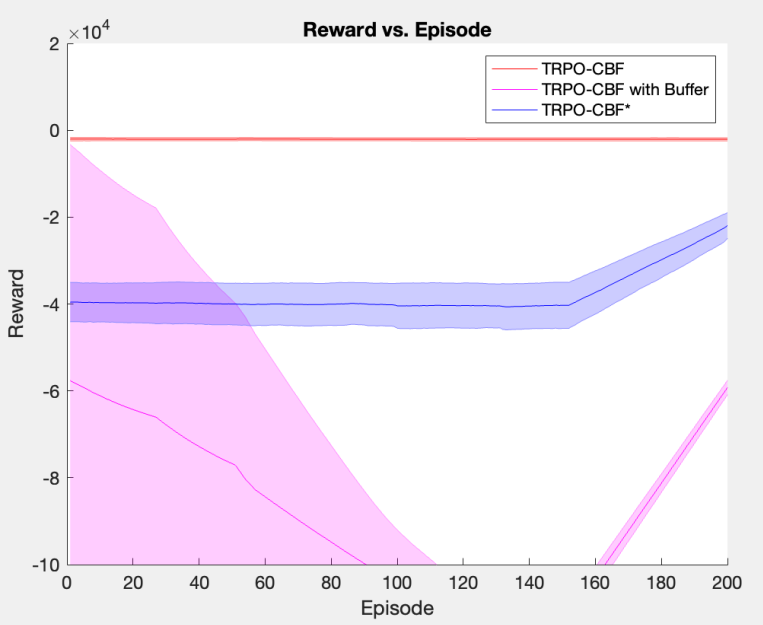

Figure 4: The average episode reward of different algorithms in 200 episodes. The rewards for the TRPO-CBF algorithm and TRPO-CBF with buffer algorithm are compared.

Our experiments also conclude that computing power has a negligible impact on overall performance. Two hundred episodes are still not enough for our computer to reach a stable state for both TRPO-CBF and TRPO-CBF with buffer methods. While we can draw conclusions from the current experimental data, more episodes need to be run for more rigorous comparisons.

In real-life scenarios, there are more vehicle actions beyond simple acceleration and deceleration. For example, vehicles can shift to other traffic lanes while driving. Although in the car-following situation, the TRPO-CBF method tends to be the optimal algorithm among those mentioned in the paper, it may not maintain its outstanding performance in more complicated cases with additional vehicle actions. We plan to extend the simulation to include multi-lane traffic scenarios and evaluate the performance of different methods under these more complex conditions.

6 Conclusion

This study evaluated the integration of Trust Region Policy Optimization (TRPO), Control Barrier Functions (CBFs), and a dual-buffer mechanism in a five-car-following scenario to improve safe and efficient reinforcement learning. The results confirmed that TRPO-CBF, adapted from previous work, effectively avoids collisions and achieves high rewards with stability. However, adding buffers introduced instability, potentially due to suboptimal tuning of the loss function parameters, and failed to improve upon the original TRPO-CBF method within the given training duration.

Our findings highlight the importance of carefully tuning loss function parameters and the need for extended training to achieve stable learning outcomes. Future work will focus on refining the buffer mechanism, optimizing its parameters, and scaling the framework to multi-lane traffic scenarios to address more complex real-world conditions. This progression aims to enhance the scalability and robustness of safe reinforcement learning for autonomous driving applications.

References

[1] J. García, Fern, and o Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, vol. 16, no. 42, pp. 1437–1480, 2015. [Online]. Available: http://jmlr.org/papers/v16/garcia15a.html.

[2] S. Gu, L. Yang, Y. Du, *et al.*, "A review of safe reinforcement learning: Methods, theories and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[3] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control**this work is partially supported by the national science foundation grants 1239055, 1239037 and 1239085.," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015, Analysis and Design of Hybrid Systems ADHS, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2015.11.152. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896315024106.

[4] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, "End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 3387–3395, Jul. 2019. DOI: 10.1609/aaai.v33i01.33013387. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4213.

[5] L. Zhang, Z. Zhang, Z. Pan, *et al.*, "A framework of dual replay buffer: Balancing forgetting and generalization in reinforcement learning."

[6] S. Nageshrao, H. E. Tseng, and D. Filev, "Autonomous highway driving using deep reinforcement learning," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019, pp. 2326–2331. DOI: 10.1109/SMC.2019.8914621.

[7] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Jun. 2017, pp. 22–31. [Online]. Available: https://proceedings.mlr.press/v70/achiam17a.html.

[8] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh, *Lyapunov-based safe policy optimization for continuous control*, 2019. arXiv: 1901.10031 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1901.10031.

[9] C. R. He, I. G. Jin, and G. Orosz, "Data-based fuel-economy optimization of connected automated trucks in traffic," in *2018 Annual American Control Conference (ACC)*, IEEE, 2018, pp. 5576–5581.