Deep RL for Self-Driving in CARLA

Thomas Kauffman
Kahlert School of Computing
University of Utah
u1374928@utah.edu

Kutay Eken
Kahlert School of Computing
University of Utah
kutay.eken@cs.utah.edu

Abstract

Autonomous driving in urban environments requires precise lane-following capabilities, even in simple scenarios such as driving on a straight path. Solving this problem is important and interesting because staying within lane boundaries safely is a key skill for self-driving vehicles, and using RL allows the agent to learn this behavior on its own, rather then relying on hand-coded rules. In this paper, we aim to train an RL agent in the CARLA simulator to drive safely from a specified start point (A) to an endpoint (B) on a straight road while staying in lanes. CARLA's reward function framework guides the learning process by providing feedback based on the agent's behavior. The agent's performance is evaluated by how consistently it says in-lane and reaches the destination safely.

1 Introduction

Ensuring the safety of autonomous vehicles is one of the most critical challenges in modern artificial intelligence and robotics. As self-driving continues to advance, with the existence of companies like *Tesla*, *Waymo*, and *Cruise*, it becomes more important to explore the decision-making process of the agents and their confidence levels. Safety is the major concern for both developers and the public, and reinforcement learning (RL) offers a promising avenue for designing intelligent agents that are capable of learning complex behaviors through interaction with the environment.
Especially, Deep Reinforcement Learning (DRL) has proven to be a powerful framework for handling complex decision-making tasks. Proximal Policy Optimization (PPO) is one of the most common DRL algorithms as it balances between performance and stability. While this makes PPO a very common baseline in autonomous driving tasks, most of the existing work in DRL for self-driving focuses on optimizing reward without explicitly accounting for uncertainty. This can potentially lead to unsafe decisions, especially in ambiguous scenarios.
In this project, we investigate whether incorporating uncertainty into the reward function can encourage safer driving behavior. Specifically, we compare a standard PPO agent to an "anxious PPO" agent, one that incorporates a risk factor into the reward function. Using the CARLA simulator as our testing environment, we evaluate both agents across a series of driving tasks to determine whether modeling uncertainty improves key metrics for safety.

2 Related Work

There is abundant literature on using DRL to train self-driving agents, namely lane-keeping among other settings. There is much work examining using DRL specifically in Carla, using the DQN and DDPG algorithms to train their agents[2, 4]. These works, however, do not use PPO to train their agents. Other work uses PPO for self driving in Carla, with no uncertainty consideration in the reward function [1, 7]. There is also work investigating uncertainty-aware learning approaches. Some of this work does not utilize a reward function [6], or doesn't apply the approach to self-driving [3]. Wu et al. on the other hand, trains different agents with PPO, DQN, and DDPG using a custom reward function

Class Project for Advanced AI, Spring 2025.

with risk-awareness incorporated as we intend to do, concluding that PPO is likely the best algorithm to leverage this risk awareness. They tested this agent in the gymnasium environment CarRacing [5]. We plan to further test and expand this idea by comparing vanilla PPO with PPO using a similar risk-aware reward function in a more realistic environment, being CARLA, with traffic incorporated into the environment.

3 Problem Statement

The goal of this work is to enable autonomous driving agents to safely and efficiently navigate from a starting location *A* to a destination *B* within a structured road environment, specifically a straight road segment. The agent must maintain lane discipline, minimize unsafe behaviors such as lane departures and collisions, and make consistent progress toward the destination.

We model this task as a Markov Decision Process (MDP), defined by the tuple (S, A, P, R, γ), where:

- S is the state space, representing the agent's observations—such as speed, lane offset, collision status, and distance to target points.
- A is the action space, representing continuous control inputs including steering, throttle, and braking.
- $P : S \times A \times S \to [0, 1]$ is the transition probability function, governed by the physics and dynamics simulated by the CARLA environment.
- $R : S \times A \to \mathbb{R}$ is the reward function, providing feedback based on lane-keeping performance, collision avoidance, and progress toward the final destination.
- $\gamma \in [0, 1]$ is the discount factor that balances immediate and future rewards.

The objective is to learn a policy $\pi : S \to A$ that maximizes the expected cumulative discounted reward:

$$\pi^* = \arg\max_{\pi} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \,\middle|\, a_t \sim \pi(s_t)\right],$$

where the agent aims to reach the destination while maintaining safe and consistent lane-following behavior.

In addition to standard reward optimization, we investigate whether incorporating an explicit risk penalty—such as penalizing deviations from the lane center—into the reward function can improve safety and robustness. This introduces a notion of uncertainty-awareness into the agent's decision-making process.

While there are many possible methods for solving this problem, we focus on Proximal Policy Optimization (PPO) due to its stability, sample efficiency, and strong performance in continuous control tasks. PPO's clipped objective helps ensure reliable policy improvement, making it a suitable baseline for comparing a standard reward structure against one that incorporates risk-awareness.

Overall, the problem formulation remains solution-agnostic and could be extended to other reinforcement learning algorithms or more complex driving environments beyond straight roads.

4 Method

We have developed the following framework:

1. **Environment**

   We use CARLA v 0.9.15. This is a stable version of Carla built with Unreal Engine with robust and accurate documentation, as well as much third-party guidance available throughout the internet.

   To better observe the agent's behavior and provide a structured reference for navigation, we created waypoints along the path to the target destination. These waypoints are visualized as small rectangles in Figure 4. Waypoints that have been successfully passed by the agent are marked in yellow, while unpassed waypoints remain blue. During testing, a red marker

is additionally placed at the target destination to clearly indicate the endpoint for evaluation purposes.

2. **Model**
   We will use PPO from Stable Baselines.

3. **Action Space**
   We use a continuous action space to dictate steering, throttle, and braking. Each action is a continous float $s, t, b \in [-1, 1]$ where $s =$ steer, $t =$ throttle, and $b =$ brake. We use the tanh activation function for these outputs. CARLA requires $t, b \in [0, 1]$, so t and b are translated accordingly with $x_{new} = (x_{old} + 1)/2$

4. **Observation Space**
   We define the observation space with 7 continuous values:
   (a) Speed $[0, 50]$ km/h
   (b) Lane offset $[-10, 10]$ meters from center
   (c) Collision indicator $\{0, 1\}$
   (d) Distance to target $[0, 200]$ meters
   (e) Distance to waypoint $[0, 200]$ meters
   (f) Direction to target $[-\pi, \pi]$ radians
   (g) Direction to waypoint $[-\pi, \pi]$ radians

5. **Vanilla Reward Function**
   The vanilla reward function simply rewards for each timestep progress towards the target as well as discovering waypoints, and penalizes lane departure and collisions.

$$\mathbf{R_1}(\mathbf{s_t}) = r^{\text{cross}} + r^{\text{collision}} + r^{\text{waypoint progress}} + r^{\text{target progress}}$$

$$r^{\text{cross}} = \begin{cases} -a & \text{if } |\text{lane offset}| > 2 \\ 0 & \text{otherwise} \end{cases}$$

$$r^{\text{collision}} = -\text{collision}$$

$$r^{\text{waypoint progress}} = \begin{cases} b & \text{if distance to waypoint} < 1 \\ 0 & \text{otherwise} \end{cases}$$

$$r^{\text{target progress}} = c\Delta(\text{distance})$$

   where a, b, and c, are hyperparameter constants. Waypoint discovery is only rewarded for new waypoints after the most recently discovered waypoint or starting location at the beginning of the episode.

6. **Anxious Reward Function**
   We then train another agent with PPO, but with a tailored reward function with a risk factor. The reward framework is explained nicely in [5], and is quite simple, though requires some customization due to the difference in environments. Along with the original reward, a risk factor for lane drifting will be added. This reward function can be written as

$$\mathbf{R_2}(\mathbf{s_t}) = r^{\text{risk}} + r^{\text{collision}} + r^{\text{waypoint progress}} + r^{\text{target progress}}$$

$$r^{\text{risk}} = \begin{cases} -c|\text{lane offset}| & \text{if } |\text{lane offset}| > 0.1 \\ 0 & \text{otherwise} \end{cases}$$

   and the CARLA API will be used to detect lane drifting with much granularity. c will be tuned for according to performance.

7. **Training Setup**
   Each agent was trained using the Proximal Policy Optimization (PPO) algorithm from Stable Baselines3. We used the following hyperparameters:

- Learning rate: 1×10^{-3}
- Number of environment steps between updates (`n_steps`): 2048
- Batch size: 128
- Number of epochs per update: 10
- Discount factor (γ): 0.99
- Generalized Advantage Estimation parameter (λ): 0.95
- Clipping range: 0.2
- Entropy coefficient: 0.01

We trained each agent for a total of **500,000 timesteps** using a CPU device. Training was performed in a single environment instance, and no early stopping was used.

8. **Evaluation**
   We will then compare through trials which agent can stay in the lane more effectively using the metrics explained in section 5.

5 Experimental Design

We hypothesize that implementing Proximal Policy Optimization (PPO) with a customized reward function that incorporates an uncertainty bonus in autonomous driving control systems will reduce lane drifting and improve overall travel safety.

To test this hypothesis, we use the CARLA simulator as the experimental domain, which provides a realistic environment for training and testing autonomous driving agents. The experiments involve training a PPO agent on straight road segments and testing it on the same straight road segment used during training. No domain shift or unseen environment testing was performed, as the agent demonstrated robustness during training and was not significantly affected by minor variations in the road environment. In this setup, the agents are primarily focused on reaching the destination and maintaining lane discipline. The target destination is positioned exactly 50 meters straight ahead from the spawn point along the same lane.

Performance is evaluated using several key metrics.

- **Average distance from the center of the lane** - to measure the degree of lane drifting.
- **Number of lane departures** - to quantify how often the agent fails to stay within its lane.
- **Collision rate** - measured as the number of collisions per unit distance traveled
- **Route completion rate** - the percentage of routes successfully completed.

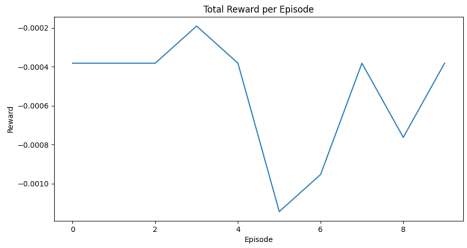
Finally, these metrics are compared against the baseline agent to determine the effectiveness of the proposed PPO-based approach with the customized reward function.

6 Results and Discussion

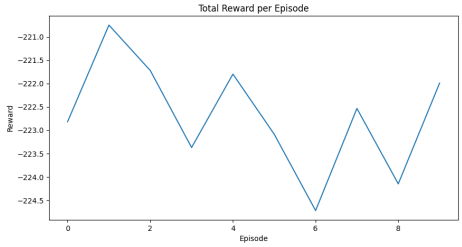
Our experiments aimed to investigate whether incorporating an uncertainty-aware reward bonus into the PPO training framework would improve lane-keeping performance and overall driving safety. Based on our findings, the results neither strongly confirmed nor refuted our original hypothesis.

Each agent was trained for 500,000 timesteps, which we chose based on common practice for similar continuous control tasks. Although we briefly experimented with extending the training time, we did not observe significant changes in agent behavior. As a result, we report our findings based on the 500,000 timestep training runs.

Quantitatively, we observed some differences between the baseline PPO agent and the Anxious PPO agent. As shown in Figure 1, the Anxious PPO agent consistently received negative total rewards during training but achieved higher reward values compared to the regular PPO agent.



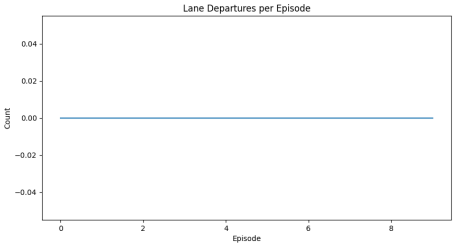
(a) Anxious PPO Rewards.



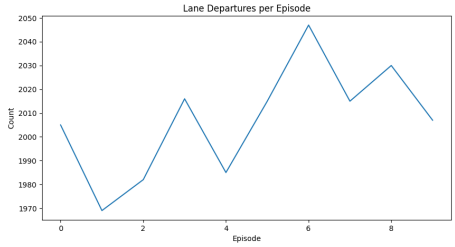
(b) Regular PPO Rewards.

Figure 1: Comparison of rewards for the Anxious PPO agent and the baseline PPO agent.

Further evaluation of safety metrics revealed a clear difference. As shown in Figure 2, the Anxious PPO agent recorded zero lane departures throughout the post-training evaluation, while the baseline PPO agent frequently departed from its lane boundaries.



(a) Lane Departure for Anxious PPO.



(b) Lane Departure for Regular PPO

Figure 2: Comparison of lane departures for the Anxious PPO agent and the baseline PPO agent.

Collision behavior was also examined. As shown in Figure 3, the baseline PPO agent had collision in every episode and Anxious PPO did not have any collision due to stopping when it detects danger.

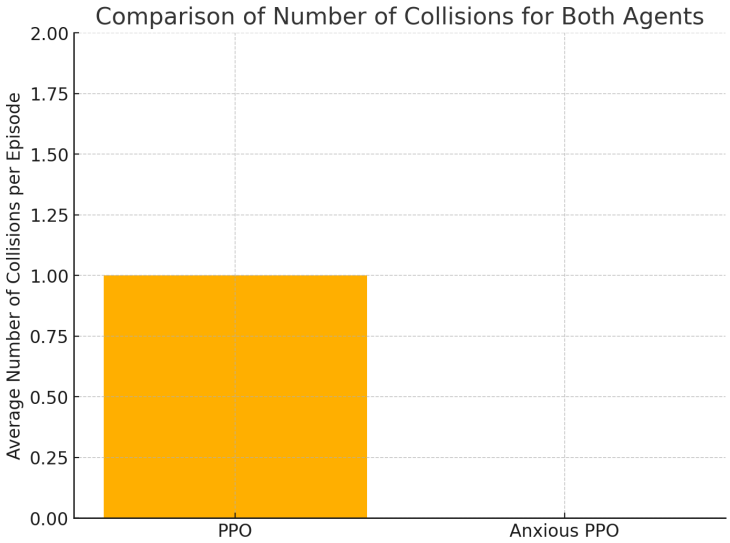


Figure 3: Number of collisions recorded during training for the Anxious PPO agent and the baseline PPO agent.

Regarding route completion, neither agent successfully completed the route. The Anxious PPO agent typically halted completely before reaching the destination, while the baseline PPO agent often collided or left the drivable area before completion. Thus, although the Anxious PPO agent prioritized safety, it did so at the cost of task completion.

The behaviors observed during testing were consistent with and even more extreme than those during training. The Anxious PPO agent largely remained stationary during the episode, choosing not to move forward rather than risk drifting out of the lane. In contrast, the baseline PPO agent continued to drive but drifted significantly away from the lane center without corrective action. These results further reinforce that the risk-aware reward structure led to an overly conservative strategy, severely prioritizing safety over any meaningful progress toward the goal.

(a) Baseline PPO During Testing.

(b) Anxious PPO During Testing

Figure 4: Comparison of Agents During Test Time

7 Conclusion and Future Work

In this project, we set out to explore whether adding an uncertainty-aware bonus to the reward function could help a PPO agent drive more safely in the CARLA simulator. Our results showed that while the Anxious PPO agent greatly reduced lane departures and collisions compared to a baseline agent, it often came at the cost of completing the task. Instead of trying to correct itself when drifting, the agent would simply stop driving altogether. This highlights how tricky it can be to balance safety and efficiency in autonomous driving.

One of the main things we learned is that PPO struggled to consistently learn good policies in this simple lane-following task, especially with a continuous action space. We iterated on the baseline vanilla reward function many times. At first, we tried an extremely simple reward function that penalizes lane-crossing heavily and collisions while giving a bonus for forward movement, with the reasoning that with enough exploration, the agent would learn the optimal route. This can be written as the following:

$$\mathbf{R_1}(\mathbf{s_t}) = r^{cross} + r^{speed} + r^{collision}$$
$$r^{cross} = -a$$
$$r^{speed} = b(speed)$$
$$r^{collision} = -collision$$

We had much trouble getting the agent to make any meaningful progress with this approach. Therefore we incorporated an acceleration bonus as seen below:

$$\mathbf{R_1}(\mathbf{s_t}) = r^{\text{cross}} + r^{\text{speed}} + r^{\text{collision}}$$

$$r^{\text{cross}} = \begin{cases} -a & \text{if } |\text{lane offset}| > 2 \\ 0 & \text{otherwise} \end{cases}$$

$$r^{\text{collision}} = -\text{collision}$$

$$r^{\text{speed}} = b(\text{speed}) + \text{movement bonus} + \text{acceleration bonus}$$

Finally, we realized that the agent would not be able to learn the optimal route to the target without salient information regarding the location of the target. We also found that the speed reward was not meaningfully contributing. Thus, we removed the speed reward and added waypoint and target progress rewards. Nonetheless, an optimal safe driving policy was not learned.

While the Anxious reward structure encouraged safer behavior, it also made the agent too conservative. Just making the agent "more anxious" about drifting wasn't enough to solve the problem — getting good driving behavior clearly needs more careful reward design or different algorithms altogether.

As part of a quick trial outside the main project, we briefly tried training an agent with Soft Actor-Critic (SAC) instead of PPO. SAC showed much stronger early learning results, but due to time limits and a desire to stay focused, we didn't fully explore it here.

If we had more time, there are several directions we would want to explore. One would be to discretize the action space to make it easier for PPO to learn stable policies. Another would be to integrate environment features, like waypoints and destination markers, more directly into the agent's inputs to help it make better decisions. Once we have a safer baseline agent, we could also test it in more challenging settings like intersections, turns, and traffic with other vehicles. Finally, combining simple vector observations with camera images to create a multimodal input could help the agent handle more complex, visually rich environments.

Overall, this project gave us valuable insights into the challenges of safe reinforcement learning for autonomous driving, and opened up several exciting directions for future research.

References

[1] Rodrigo Gutiérrez-Moreno, Rafael Barea, Elena López-Guillén, Javier Araluce, and Luis M Bergasa. Reinforcement learning-based autonomous driving at intersections in CARLA simulator. 22:8373.

[2] Jumman Hossain. Autonomous driving with deep reinforcement learning in CARLA simulation.

[3] Xingzhou Lou, Dong Yan, Wei Shen, Yuzi Yan, Jian Xie, and Junge Zhang. Uncertainty-aware reward model: Teaching reward models to know what is unknown.

[4] Óscar Pérez-Gil, Rafael Barea, Elena López-Guillén, Luis M Bergasa, Carlos Gómez-Huélamo, Rodrigo Gutiérrez, and Alejandro Díaz-Díaz. Deep reinforcement learning based control for autonomous vehicles in CARLA. 81:3553–3576.

[5] Lin-Chi Wu, Zengjie Zhang, Sofie Haesaert, Zhiqiang Ma, and Zhiyong Sun. Risk-aware reward shaping of reinforcement learning agents for autonomous driving.

[6] Junkai Zhang, Weitong Zhang, Dongruo Zhou, and Quanquan Gu. Uncertainty-aware reward-free exploration with general function approximation. closest thing I've found so far.

[7] Jianfeng Zhao, Yang Zhao, Weixia Li, and Chaoqun Zeng. End-to-end autonomous driving algorithm based on PPO and its implementation. In *2024 IEEE 13th Data Driven Control and Learning Systems Conference (DDCLS)*, pages 1852–1858. IEEE.