Project Report: Training a Self-Driving Agent with Deep Reinforcement Learning

Tenzing Jampa Bhutia

22B1808

Seasons of Code 2025

Week 6 - RL in Self-Driving Cars

July 14, 2025

Abstract

This report details the development and evaluation of an autonomous driving agent trained using Proximal Policy Optimization (PPO) in the `highway-env` simulator. The primary objective was to create a policy capable of navigating dense highway traffic safely and efficiently. The project highlights a successful training strategy involving callbacks for performance monitoring and early stopping, culminating in a robust agent that consistently achieves high rewards and demonstrates intelligent driving behavior.

Contents

1 Introduction and Objective

The goal of this project was to tackle the complex challenge of autonomous driving by training an agent to navigate a busy highway. Using the `highway-env` simulator, I set out to develop a robust driving policy that could handle dense traffic, make intelligent decisions about speed and lane changes, and, most importantly, avoid collisions.

For this task, I chose the **Proximal Policy Optimization (PPO)** algorithm, a state-of-the-art method known for its stability and performance in continuous control problems like this one. My implementation relied on the excellent `stable-baselines3` library, which provides a solid and optimized version of PPO, allowing me to focus more on the environment setup and training strategy.

2 Algorithm and Implementation Details

My approach was centered around creating a realistic and challenging environment and then applying a well-tuned PPO model to master it.

2.1 Environment Configuration

I used the `highway-v0` environment with a custom configuration designed to test the agent's abilities:

- **Action Space:** I configured a `ContinuousAction` space. This felt more natural for a driving task, as it allows the agent to make fine-grained adjustments to the throttle and steering, rather than being limited to a few discrete choices.
- **Observation Space:** The agent used `Kinematics` observations, giving it information about the position and velocity of the 10 nearest vehicles. This gives the agent enough information to "see" its immediate surroundings and react accordingly.
- **Traffic Density:** To create a suitably challenging scenario, I set the environment to have **3 lanes** populated with **20 other vehicles**.
- **Reward Structure:** The agent was rewarded for maintaining a safe and efficient speed (between 20 and 30 m/s) and was given a significant penalty of -2 for any collision, strongly discouraging unsafe maneuvers.

2.2 PPO Model and Training Strategy

I built the PPO model using a standard `MlpPolicy` (Multi-layer Perceptron) and selected hyperparameters that are generally known to provide a good balance between learning speed and stability.

The real core of my training strategy was the use of **callbacks**. Instead of just letting the model train for a fixed, arbitrary number of steps, I implemented:

1. An `EvalCallback` to periodically test the agent's performance every 5,000 steps and save the best-performing model. This ensures that even if performance dips later in training, I always have the most competent version of the agent saved.
2. A `StopTrainingOnRewardThreshold` callback to automatically end the training process once the agent's average reward surpassed a target of 160. This was a huge time-saver and prevented the model from training unnecessarily long.

3 Results and Performance Analysis

The training process was a definite success. The agent showed clear and consistent improvement from the very beginning. The training logs provide a great story of the agent's journey from a clumsy beginner to a skilled driver.

The agent started with a very low mean reward of around 6.94. As it explored the environment, its performance grew steadily. The most significant learning leaps occurred within the first 20,000 timesteps. Table 1 summarizes its progress from the evaluation logs.

Table 1: Agent Performance Growth During Training

Timesteps	**Mean Reward**
5,000	29.5
10,000	84.5
20,000	143.0
30,000	149.0
35,000	**170.02**

As you can see, the agent learned incredibly quickly. At the 35,000-timestep mark, its performance (`mean_reward` of 170.02) surpassed the threshold of 160 I had set. The training then automatically stopped, and the best model was saved. The final evaluation showed not only a high reward but also a very low standard deviation (± 2.33), indicating that the agent's high performance was consistent across multiple runs.

The final video, recorded using the best model, provides visual proof of this success. The agent can be seen confidently weaving through dense traffic, maintaining its speed, and executing smooth lane changes to avoid collisions.

4 Insights and Conclusion

This project was a fantastic hands-on experience in applying reinforcement learning to a practical, complex problem. My biggest takeaway was the power of an efficient training workflow. Using callbacks to monitor performance and stop training early was instrumental to the success of this project. It allowed me to get a high-performing agent in just over 20 minutes of training time.

I encountered a few `SettingWithCopyWarning` messages from the underlying `highway-env` library during training, but these didn't impact the agent's learning and seem to be a known issue within the environment's code.

In conclusion, the PPO algorithm proved to be more than capable of solving the decision-making challenge for autonomous driving. By carefully configuring the environment and implementing a smart training loop with monitoring and early stopping, I successfully trained an agent that drives safely and efficiently in a simulated highway environment.