Exploring the Dualistic Nature of the Reward Function for an Autonomous Agent in CARLA

Joel Andrew Miller
Electrical & Computer Engineering
University of Western Ontario
jmill266@uwo.ca

***Abstract*—This paper explores the dualistic nature to consider when defining the reward function for a reinforcement learning (RL) agent used for autonomous vehicle (AV) operation. Through a series of experiments within the CARLA simulation environment, this work investigates the impact of various reward structures on a simple RL agent's behavior. The objective of this work was to offer valuable insights and practical guidance for future AV researchers, on optimizing reward functions despite their paradoxical nature. The results showcase both the beneficial and detrimental behaviors that emerge under different reward settings, thereby underscoring the intricacy of its design in RL, especially for AVs.**

Keywords—Reinforcement Learning, CARLA, Autonomous Vehicle, RL in CARLA.

I. Introduction

In recent years, the global AV market has witnessed enormous growth. In 2022, estimates valued the industry at an unprecedented $121 billion[1] [1], while more optimistic projections suggested a potential market size of $1,900 billion [2]. Regardless of the exact valuation, both estimations predicted exponential growth over the next 10 years.

This economic surge only underscores the mounting interest in the technology which powers these machines, creating fertile ground for research, development, and innovation, as businesses promise financial support to obtain cutting-edge solutions. Yet, this colossal price-tag has also introduced competition, and an incentive to privatize valuable contributions behind closely guarded walls. Moreover, the development of RL systems requires extensive training and abundant resources to finance large amounts of computational power. This creates a stark divide between entities based on the size of their pockets, as a larger budget can accelerate machine learning training process, creating a comparative disadvantage for those with smaller resources [3].

A. Objective and Contribution

Given these dynamics, the goal of this work is to provide insight into the components that define an optimal reward function, in the context of a RL vehicle in a simulated driving environment. This work provides a cause-and-effect results section for readers to comprehensively understand the effects – both expected and unexpected - of several reward functions implemented. And since, training a RL agent to an effectively tuned reward function can significantly reduce the amount of unnecessary training iterations, through this work, the authors hope to democratize innovation by enabling a broader base of researchers to understand and ultimately contribute to the advancement of autonomous driving technologies.

B. Paper Structure

The remainder of the paper is as follows: Section II will introduce the reader to the preliminary elements of the project, including the simulation environment that was used, as well as an overview of reinforcement learning. Section III will provide an overview of similar existing research. Section IV will explain the problem formulation, detailing how RL was used in a CARLA environment. Section V provides the results. Finally, Section VI will discuss the limitations of this work, along with next steps and a discussion.

II. Related Works

A. Hossian (2023)

Hossain's work emphasizes the application of Deep Q-Learning for enabling an autonomous car to maintain its lane while avoiding other vehicles, tested within the CARLA simulation environment. Lot of the inspiration for this paper came from the preliminary work completed by Hossain. While his work focusing on the challenges of RL in complex, high-dimensional environments, it does not delve into the nuanced effects of varying reward functions on learning outcomes [4].

B. Toromanoff (2020)

Toromanoff et al. introduce "implicit affordances" as a novel technique for urban driving, including complex tasks like traffic light detection and pedestrian avoidance, leveraging RL for end-to-end control from vision. Their contribution provides an RL agent capable of handling intricate urban scenarios, particularly traffic light management [5]. However, their approach focuses on the integration of auxiliary tasks (affordances) for improved learning efficiency and interpretability, rather than exploring the reward function's dualistic nature.

C. Pérez-Gil et al. (2022)

Pérez-Gil et al. focus on implementing and comparing the performance of two DRL algorithms: Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG). Their findings demonstrate that both DQN and DDPG can achieve the set objectives, with DDPG showing superior performance [6]. No focus is given to the paradoxical nature of the reward function.

D. Shao et al. (2022)

Shao et al. (2022) introduce the Interpretable Sensor Fusion Transformer (InterFuser), aimed at enhancing safety

[1] All prices in USD

in autonomous driving through comprehensive scene understanding and adversarial event detection by fusing multi-modal sensor data. They address two main challenges: recognizing rare adverse events and ensuring the interpretability of decision-making processes in AI models [7]. The goal of Shao et al.'s paper is not to demonstrate the ironies inherent with the reward function.

III. Preliminaries

A. Simulation Software: CARLA

Fig. 1. Example simulation environment from the CARLA simulation software.

The driving environment for this project was simulated using a prebuilt town from CARLA (CAR Learning to Act). CARLA is an open-source software developed on Unreal Engine 4, designed to support the comprehensive needs of autonomous driving research. CARLA offers a realistic, controlled environment complete with open digital assets such as buildings, vehicles, and pedestrians, and its flexible framework allows for the specification of diverse sensor suites and environmental conditions, accommodating a wide range of experimental setups. By simulating dynamic scenarios, CARLA provides a robust platform for developing, training, and validating autonomous driving systems. For this project, CARLA version 0.9.15 was utilized [8], an example of the CARLA environment is provided in Fig. 1. Traffic was not used in this project to keep the environments complexity simple, in an attempt to understand the dynamics of the vehicle in a static environment.

B. Reinforcement Learning

Reinforcement Learning (RL) is a type of machine learning that focuses on training an agent in a dynamic environment. In RL, the agent's observations from its environment, are the data that the model uses to train, and the agent's ultimate objective is to learn a policy π that maximizes its cumulative reward over time.

1) The RL Process

Formally, the RL process can be conceptualized as a loop of discrete agent-environment interactions. The symbiotic cycle follows the sequence of steps outlined as follows. (1) At time t, the agent receives an observation o_t from its environment. (2) Based on its o_t, the agent responds with action a_t, (3) causing the state s_t of the environment to update. Afterwards, (4) the agent will receive the subsequent observation o_{t+1} and reward r_{t+1} based on the updated state s_{t+1} of the environment. Finally, (5) the agent updates its policy with the goal of maximizing future rewards. This symbiotic relationship, often referred to as the agent-environment interaction (illustrated in Fig. 2), is repeated, while the agent learns which observation-action pair yields the highest cumulative reward [9].

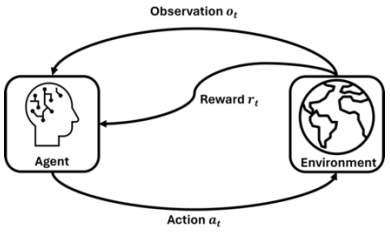


Fig. 2. The agent-environment interaction, which constitutes the refinforcement learning process.

The process of RL, used to update the policy in step (5) can be summarized with the following equation:

$$\pi^*(s) = \arg\max_{\pi} E\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \pi\right]$$

In this context, $\pi^*(s)$ represents the optimal policy for a given state s, while π refers to the policy under evaluation. E[$\sum$] denotes the expected sum from $t = 0$ to infinity of the reward r_t multiplied by a discount factor γ to determine the importance of future rewards.

2) The Reward Function

In RL, the reward can be considered a function of the environment, that takes the environment's current state s_t and the agent's action a_t as input and spits out a reward value r_{t+1} – either positive or negative – as output. This is represented formally in the equation shown below:

$$reward\ function(a_t, s_t) = r_{t+1}$$

Herein lies the heart of RL; this critical component is what guides the agent's learning process. Since the goal of an agent in RL is to maximize its cumulative reward, this function is extremely useful, as it provides a quantitative evaluation of an agent's actions in relation to a desired objective. However, crafting an effective reward function is both an art and a science. Careful consideration must be given to ensure that rewards drive desired behavior, without creating unintended consequences. Examples of this paradoxical behavior is explored further in the results (Section VI) below.

3) Deep Q-Learning

There are several ways to design the "brain" of the agent in RL. However, for the purposes of this project, a Deep Q-Learning (DQL) was utilized to determine the respective Q-value for each action based on the agent's current state. The Q-value at time step t for an action a_t taken in state s_t can be calculated using the following formula:

$$Q(s_t, a_t) = E[r_t + \gamma \max_a Q(s_{t+1}, a]) \mid s_t, a_t]$$

Where the term $\gamma \max_a Q(s_{t+1}, a)$ is the maximum Q-value for the subsequent states considering every possible action.

DQL uses a neural network to predict the Q-value for every possible action based on the agent's current state. In DQL, the loss function compares the Q-value prediction to a Q-value target and uses gradient decent to update the weights of the neural network. The Q-Loss is shown in the formula below:

$$Loss = r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$$

Where $r_{t+1} + \gamma \max_a Q(s_{t+1}, a)$ denotes the Q-value target and $Q(s_t, a_t)$ denotes the former Q-value prediction. Once the training is complete, the neural network will be able to receive the agent's observation as input and provide the Q-

value for each possible action as output, as illustrated in Fig. 3.

Fig. 3. An illustration of the Deep Q-Network used to predict the value for each action given the agents obervation of the environment.

IV. Methodology

The RL process, for the purposes of this project, was defined as illustrated in Fig. 4. The agent implemented in this work can be considered relatively simple, as it only features a single sensor, and uses an end-to-end architecture. The goal of this work was not to introduce a groundbreaking reward structure, or state-of-the-art sensor fusion, localization, or perception, but rather to provide insight, discussing the effects of a reward function on the vehicle's behavior. This resource can be used as a tool for researchers new to the domain of RL or AV, to become familiar with this technology. Interested readers should examine the source code (available here[2]) for a more hands-on approach, as each of the experiments run below may be recreated and improved upon.

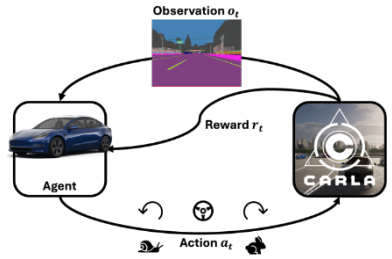


Fig. 4. An illustration of the RL process as implemented in this paper.

A. Agent

The RL agent implemented in this project utilized the DQL library implemented in stable baselines [10]. This python library implements DQL as first defined in the work done by Mnih et. Al. [11].

The actual neural network takes an image as input and features 3 hidden layers. The initial hidden layer processes the input image by applying 16 filters of size 8x8 with a stride of 4, followed by the application of a rectifier nonlinearity function. Following this, the second hidden layer utilizes 32 filters of size 4x4 with a stride of 2, and once again applies a rectifier nonlinearity. The final hidden layer is fully connected, containing 256 units that use a rectifier function. For the output, there's a fully connected linear layer that generates a single output for each possible action, which totals 36 for this specific project.

B. Environment

The environment was simulated in CARLA's prebuilt "Town 10". The map features a downtown urban environment next to an ocean promenade, with skyscrapers, and residential building.

C. Observation Space

The observation space for the agent is derived from a camera mounted to the hood of the agent vehicle. The image data is segmented based on the object type depicted in the scene. The resulting image matrix is size (240,320,3) with pixel values ranging from 0 to 255.

D. Action Space

The action space of the vehicle is defined as 36 discrete values. This value is then mapped to steering and acceleration values using the algorithm provided below.

Algorithm 1

Input: action
Output: steering, acceleration
Initialize steering_values = [-0.9, -0.25, -0.1, -0.05, 0.0, 0.05, 0.1, 0.25, 0.9]
Initialize: acceleration values = [0.0, 0.3, 0.7, 1.0]
action_mapping(action):
  steering = steering_values[action // 4]
  acceleration = acceleration_values[action mod(%) 4]

Steering: *0 represents neutral steering, negative values indicate left turns, positive values indicate right turns. Steering wheel control ranges from -1 to 1.*

Throttle: *Acceleration pedal control ranges from 0 to 1.*

E. Episode

In RL, an "episode" refers to a sequence of steps taken by an agent from the start to the end of an interaction with an environment. In the case of this project, the AV is spawned in a random location in the map (on a road) and the episode ends when the vehicle collides with something in its environment.

V. Results

The primary aim of this section is to demonstrate the dualistic nature of the reward function. Each experiment will be structured as follows:

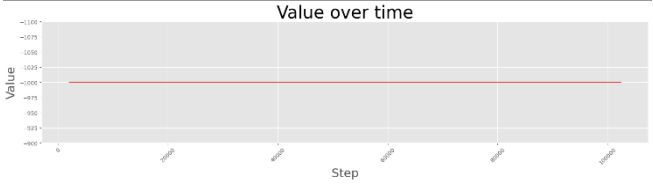
- The high-level objective of the AV, along with the practical implementation of the reward function is given in a table.
- A plot of the agent's cumulative reward over time (per episode), captured during the training process, is used to illustrate the agents ability to "learn".
- A brief discussion to provide insight to the ironies that are inherent in setting reward functions to achieve a desired objective.

A. Simple Rewards

1) Experiment 1

TABLE I. Experiment 1

Objective	Reward
Avoid Obstacles	-1000 on collision



[2] Code repository link: https://github.com/Joelmillr/ECE_9309_Final_Project

Fig. 5. Experiment 1 cumulative reward over each episode

This is an extremely poor way to define a reward. This reward function does not incentivize the agent to learn. No matter what path the agent takes, or how long it maneuvers through the map, the reward function will be the same for every experiment. The agent's behavior during this period eventually stops maneuvering through the environment all together, simply repeating the same action over and over until it crashes into something triggering a respawn.

2) Experiment 2

TABLE II. EXPERIMENT 2

Objective	Reward
Avoid Obstacles	-1000 on collision
	+1 for each timestep without collision

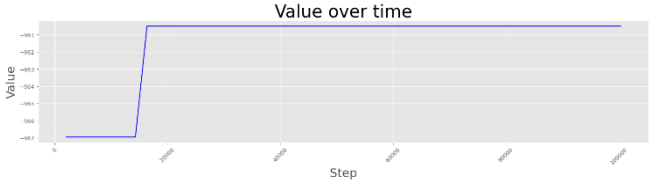


Fig. 6. Experiment 2 cumulative reward over each episode

Again, this is not a great way to define a reward function. The agent learned overtime, that the slower it moved through its environment, the more reward it would receive (i.e. the more timesteps it went without collision), and if the training process ran for long enough, the agent learned to sit completely still to get an infinite reward.

3) Experiment 3

TABLE III. EXPERIMENT 3

Objective	Reward
Avoid Obstacles	-1000 on collision
Move through the map	+ 1 when speed > 50 km/h

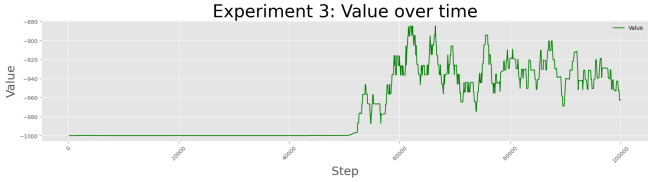


Fig. 7. Experiment 3 cumulative reward over each episode

The purpose of this experiment was to see how the agent responded to a reward which incentivized movement. This reward function showed promising initial results - where the agent moved through the map - but if the training ran for too long, the agent learned to game the reward function, and spin in circles to maintain a speed of 50 km/h, without actually moving throughout the map.

B. Reaccuring Problems

These last three experiments demonstrate some significant challenges that seemed to reappear throughout various experimentation and exploration of this RL system. The reward function was either too abstract for the agent to learn anything (as demonstrated in experiment 1), the reward function would encourage the agent to move too slowly, or not at all (as demonstrated in experiment 2), or the reward function was exploited by the AV due to its insufficient structure (as demonstrated in experiment 3). These experiments seem to demonstrate the need for a more granular reward function, that acts as a guardrail to lead the agent to achieve the objective. The next 2 experiments were proposed to encourage the agent to move throughout its environment and learn from its observation.

1) Experiment 4

TABLE IV. EXPERIMENT 4

Objective	Reward
Avoid Obstacles	-1000 on collision
Lane Keeping & Move through map	+ 10 for traveling over 25 km/h in lane.
	+ 20 for traveling over 50 km/h in lane.

Fig. 8. Experiment 4 cumulative reward over each episode

The purpose of this reward function was to incentivize the agent to learn to travel throughout the map within the lanes of the road. This reward function also served as an indirect guiderail for the agent to avoid collision, as most of the vehicle's accidents within this environment were the result of the vehicle traveling off road. The agent's behavior exemplified an ability to "lane keep", demonstrating its ability to take turns with a bending road. The drawback of this approach was the agent's inability to conceptualize its speed. The agent was unable to recognize the difference between its throttle and its speed, often flooring it around corners, causing the vehicle to spin out and lose control. However, if the agent spawned within a curved road, it was able to take the turn at a safe pace, maintaining control of the vehicle.

C. Experiment 5

TABLE V. EXPERIMENT 5

Objective	Reward
Avoid Obstacles	-1000 on collision
Lane Keeping & Move through map	+ 10 for traveling over 25 km/h in lane.
	+ 20 for traveling over 50 km/h in lane.
Reach Destination	+500 for destination reached
	+ 10 if agent gets 0.8 km over the last timestep
	+ 5 if agent 0.4 km over the last timestep

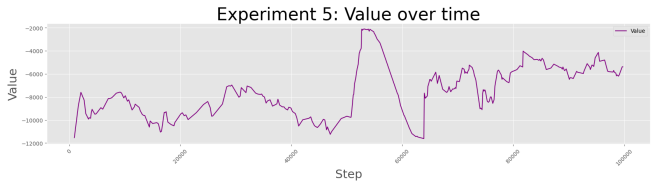


Fig. 9. Experiment 5 cumulative reward over each episode

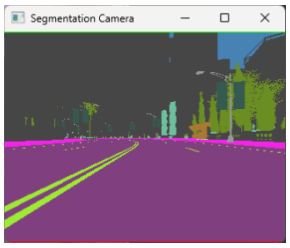


Fig. 10. Example of agents observation of the environment

The final experiment investigated feeding aspects of the agent's behaviour as input directly to the deep Q-network. As seen in Fig. 10, the top layer of pixel were replaced by colors which represented the agents speed (red for < 25km/h, blue for < 50km/h, and green for > 50km/h) and the bottom layer was replaced with pixels that illustrated if the agent was moving closer to a random destination (red for no improvement, blue for closer by 0.4 km, green for closer by 0.8 km). This allowed the agent to understand how its actions could affect the reward function. After the agent has trained for 50,000 timesteps it begins to demonstrate an ability to reach a desired destination on the map (albeit only if there is not a building in the way), and an understanding of lane keeping at a desired speed.

VI. Future Work

Future work aims to improve the agents' understanding of its environment. Distance detection would be a reasonable next step to allow the agent to understand how its actions can influence its space between an object. Additionally, AV techniques like localization and mapping could be beneficial to help the agent maneuver throughout its environment more efficiently. Additionally, tuning the exact values of each reward function may lead to valuable insight. Future work will aim to dissect how values of a reward function can influence an agent's behavior.

VII. Conclusion

This work explores the use of reward shaping in the context of an AV in a simulated CARLA environment. It demonstrates the intricacies of the vehicle optimizing for a reward function while demonstrating its paradoxical nature. This work does not intent to introduce a novel or ground-breaking approach to defining the reward function, but rather aims to provide insight to other researchers on the dualistic behavior that is inherent from defining a high-level objective, and the difficulty describing that objective quantitatively to a machine. Results demonstrate how one may shape a reward function to incentivize positive behavior that will lead to completing a high-level objective.

VIII. Reference

[1] "Autonomous vehicle market size to hit USD 2,353.93 BN by 2032.," Precedence Research, 1074, Mar. 2023. [Online]. Available: https://www.precedenceresearch.com/autonomous-vehicle-market

[2] "Autonomous Vehicle Market Size," Fortune Business Insights, FBI109045, Feb. 2024. [Online]. Available: https://www.fortunebusinessinsights.com/autonomous-vehicle-market-109045#

[3] N. Ahmed and M. Wahed, "The De-democratization of AI: Deep Learning and the Compute Divide in Artificial".

[4] J. Hossain, "Autonomous Driving with Deep Reinforcement Learning in CARLA Simulation." arXiv, Jun. 19, 2023. Accessed: Mar. 22, 2024. [Online]. Available: http://arxiv.org/abs/2306.11217

[5] M. Toromanoff, E. Wirbel, and F. Moutarde, "End-to-End Model-Free Reinforcement Learning for Urban Driving using Implicit Affordances." arXiv, Mar. 16, 2020. Accessed: Apr. 01, 2024. [Online]. Available: http://arxiv.org/abs/1911.10868

[6] Ó. Pérez-Gil *et al.*, "Deep reinforcement learning based control for Autonomous Vehicles in CARLA," *Multimed. Tools Appl.*, vol. 81, no. 3, pp. 3553–3576, Jan. 2022, doi: 10.1007/s11042-021-11437-3.

[7] H. Shao, L. Wang, R. Chen, H. Li, and Y. Liu, "Safety-Enhanced Autonomous Driving Using Interpretable Sensor Fusion Transformer." arXiv, Dec. 07, 2022. Accessed: Apr. 01, 2024. [Online]. Available: http://arxiv.org/abs/2207.14024

[8] A. Dosovitskiy, "CARLA: An Open Urban Driving Simulator".

[9] R. S. Sutton and A. G. Barto, *Reinforcement learning: an introduction*. in Adaptive computation and machine learning. Cambridge, Mass: MIT Press, 1998.

[10] Hill, Ashley and Raffin, Antonin and Ernestus, Maximilian and Gleave, Adam and Kanervisto, Anssi and Traore, Rene and Dhariwal, Prafulla and Hesse, Christopher and Klimov, Oleg and Nichol, Alex and Plappert, Matthias and Radford, Alec and Schulman, John and Sidor, Szymon and Wu, Yuhuai, "Stable Baselines." GitHub, 2018. [Online]. Available: https://github.com/hill-a/stable-baselines

[11] V. Mnih *et al.*, "Playing Atari with Deep Reinforcement Learning." arXiv, Dec. 19, 2013. Accessed: Apr. 02, 2024. [Online]. Available: http://arxiv.org/abs/1312.5602