An Assessment of the Potential Behaviours of Self-Driving Cars

Luke Guppy
3^{rd} Year 2023-2024
Supervisor: Andrew Hague

Contents

Abstract

This project explores the implementation of curriculum learning to train self-driving cars within a realistic driving simulation environment developed in Unity using the ML-Agents package. The objective is to train an autonomous agent with the ability to navigate an environment while adhering to traffic rules and avoiding collisions. The simulation incorporates various elements such as traffic lights, lane centering, corner slowing, and path following, all integral components of a comprehensive curriculum aimed at progressively teaching the agent to drive safely and efficiently.

Throughout the development process, multiple stages of reward systems were devised and iteratively refined to incentivise desired behaviours, ultimately culminating in the creation of a successful framework. The resulting agent demonstrates proficient path following capabilities, effectively negotiating complex scenarios while following traffic regulations.

This project contributes to the advancement of autonomous vehicle technology by providing insights into the effectiveness of curriculum learning methodologies for training self-driving car agents. The modular nature of the simulation allows for flexible experimentation and evaluation of various training strategies, paving the way for further research in this domain. It also gives a strong insight as to how a reward system can be developed to iteratively implement new behaviours.

Keywords

Self-driving cars, Reinforcement learning, Autonomous agents, Simulation environments, Curriculum learning, Traffic simulation, Agent-based modeling, Reward optimisation

1 Introduction to the Project

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

1.1 Refined Problem Statement and Addressed Gap

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

2 Background Research

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

2.1 Extending AI-Driven Vehicle Research:

This project represents a focused extension of existing research on AI-driven vehicles, rather than more generalised or specialised approaches. While prior studies have contributed valuable insights, they often lack a specific focus on scenarios with the highest collision rates, such as straight roads, cross-junctions, and T-junctions. By honing in on these critical driving situations, this research aims to improve our understanding of how AI-driven cars navigate and adapt, particularly in the presence of human behaviors.

2.2 Areas Of Research:

Key areas of research:

- Road sections with the highest collision rates [1]
- Human factors which cause the most incidents [1] [2] [3]
- Existing reward functions for driving simulations [4] [5] [6]
- Sensory inputs and visualisation of the environment [7]
- Inputs and outputs of neural networks for learning in automated vehicles [8] [9]
- Relationships between AI and human behaviours [10]
- Current progress and future plans for self-driving cars [11]

2.3 Summary Of Research

2.3.1 Foundation

The research started with a general overview of the current situation and progress of AI driving in the world. This meant understanding how sensors work, what features of the road the AIs use and other relevant features. It appears that there is a surprising number of sensors of various types: Cameras, Lidar, Radar, Ultrasonic, Infrared (for lanes) and GPS [7] which all have various strengths and weaknesses in the real world. In the simulation this can be heavily abstracted by the use of ray-casts and directly feeding information.

2.3.2 Collision Occurrence

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

2.3.3 Existing Implementations

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

2.3.4 High Level Overview

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

3 Technical Aspects

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

3.1 Unity

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

3.2 Unity Simulation Abstraction

ML-Agents abstracts complex machine learning processes, simplifying their application within Unity so that it appears to be a black-box which you use methods to feed inputs and return outputs. This abstraction makes it only necessary to define reward structures, observations, and actions for agents without worrying about the intricacies of machine learning algorithms. For our project, this means we can create a realistic driving simulation with AI-driven cars navigating complex scenarios without getting bogged down in the technical complexities of machine learning.

3.3 ML-Agents

ML-Agents serves as a powerful bridge between machine learning algorithms and Unity-based simulations or games. Developed by Unity itself, this open-source python based package is tailored to integrate machine learning, specifically reinforcement learning, into Unity environments. As this project involves creating a simulated driving environment using Unity, ML-Agents is a natural fit.

3.3.1 Reinforcement Learning Framework

At the core of ML-Agents is a reinforcement learning framework. This framework allows developers to train intelligent agents through a process of trial and error, guided by a self-implemented reward system. In this project, the agent is the car, which will have the goal of training to reach a certain goal with the highest reward over time. However, it is strongly limited by the reward function which drastically impacts the behaviours it learns making it the most important aspect.

3.3.2 Neural Network Policies

ML-Agents employs the use of a neural network to model the policies of the agent. The neural network adapts and refines its parameters during training, optimising the decision-making process of the agent. In our case, the neural network will be crucial for teaching the AI-driven cars to navigate roads, respond to obstacles, and follow appropriate behavior at junctions. This is governed by the inputs which will be a series of sensory inputs, distances to checkpoint and any other inputs which may be added later.

3.3.3 ML-Agents Algorithm

To give an understanding of what applications would be successful to give the expected result, it's important to understand how these algorithms work. In the context of ML-Agents, the machine learning algorithms are primarily centered around Proximal Policy Optimization (PPO), a deep reinforcement learning algorithm. PPO strikes a balance between exploration and exploitation, allowing intelligent agents to learn optimal policies through trial and error within the simulated driving environment.[12]

The technical complexities lie in the architecture of the neural networks used to represent the agent, specifically tailored for PPO. These neural networks undergo iterative updates during training, incorporating advanced optimisation techniques associated with PPO. The objective remains to maximise the cumulative reward obtained by the agents over successive interactions with the environment.

During training, PPO utilizes a trust region approach, ensuring that policy updates are performed within a constrained region to maintain stability and prevent abrupt changes. The trust region mechanism establishes constraints on the magnitude of policy updates during training. This ensures that the learning process occurs gradually, preventing drastic policy changes that might destabilise the training.

By constraining policy updates, the trust region approach enhances the stability of the learning process and contributes to the algorithm's ability to adapt to varying conditions in the simulated driving environment in a more natural manner. This careful balance is crucial for achieving effective learning of complex driving scenarios and not learning drastic unpredictable behaviours.

ML-Agents chose PPO based on research at Cornell University: "Our experiments test PPO on a collection of benchmark tasks, including simulated robotic locomotion and Atari game playing, and we show that PPO outperforms other online policy gradient methods, and overall strikes a favorable balance between sample complexity, simplicity, and wall-time." [13] This gives a reliable foundation of how PPO would effectively work in this environment.

The technical complexities also extend to hyper-parameter tuning, where factors such as learning rates, discount factors, and exploration parameters play a pivotal role in optimising the training process. However these are set to default values which have been optimised by trial and error across many projects and editing them will likely have limited or even negative impacts.

3.3.4 Non-Linearity In ML-Agents

ML-Agents predominantly employs the Swish activation function for most of its neural network architectures, particularly for the hidden layers. Swish is a popular activation function that introduces non-linearity by smoothly blending the linear and nonlinear activations. Specifically, it is defined as $x * sigmoid(x)$ and avoids previous issues of other activation functions such as the vanishing gradient problem for $sigmoid(x)$. This problem occurs as the gradient tends to 0 with extreme values causing difficulties in gradient descent for learning and progressing towards the local maximum.

Swish allows neurons to output the relation with a smooth curve, promoting stronger gradients and potentially faster convergence during training. While other activation functions like tanh or sigmoid may also be utilized in specific cases based on the neural network architecture or task requirements, Swish serves as the default choice due to its efficiency in capturing complex relationships between inputs, thereby enabling the discovery of nonlinear patterns and the emergence of complex behaviors within the deep learning structure.

It was chosen by ML-Agents due to its promising results over these other popular activation functions as detailed by Google Brain in 2017.

3.4 Community Support and Documentation

ML-Agents benefits from an active community and extensive documentation, making it easy to implement and troubleshoot problems encountered with the project. The community support is invaluable for addressing challenges, seeking solutions, and staying updated on best practices during the development of the project.

3.5 Git

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

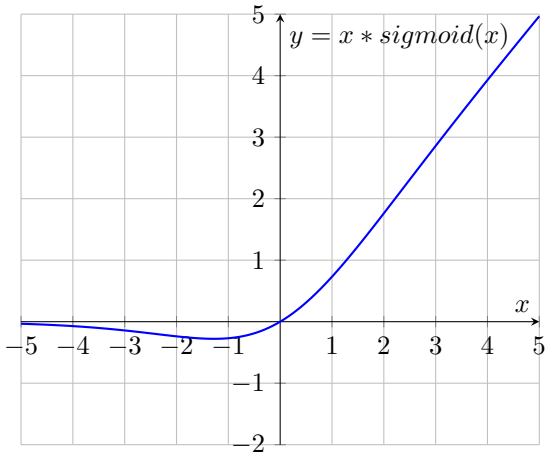

Figure 1: Swish Graph

4 The Simulation Environment

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

4.1 Roads

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

4.2 Pathing

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

4.3 The Car Control System

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

4.4 Environmental Cars

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

4.4.1 General Functionality

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

4.4.2 Logical Implementation

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

4.4.3 Traffic Light Interaction

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

4.5 Traffic Light System

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

4.5.1 Target Types

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

4.5.2 Logical Implementation

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

4.5.3 Smart Traffic

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

5 The Agent

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

5.1 Structure

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

5.2 Controls

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

5.3 ML-Agents Interaction

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

6 Reinforcement Learning

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

6.1 Reward Systems

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

6.2 Curriculum Learning

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

7 The Deep Neural Network

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

7.1 Proximal Policy Optimisation

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

7.2 Configuration

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

7.3 Hyper-parameters

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

7.4 Observations and Outputs

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

8 Deep Reinforcement Learning

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

9 Reward System

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

9.1 Curriculum Learning Implementation

- What it is
- How it is being applied
- Behaviours being implemented
- Specific implementation

9.2 Initial Reward System

- Weighted average
- Negative
- Issues with unpredicted behaviours
- Positive change

9.3 Imitation Learning

- Graph for difference
- Reasons why non-imitation is better
- Using non-imitation for rest

9.4 Final Reward System

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

10 Problems and Oversights

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

11 Conclusion

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

12 Future Outlook

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

13 Project Management

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi.

14 Objectives

14.1 Must-Have Objectives:

- **Develop a Realistic Simulation Environment:** Create a Unity-based simulation environment that abstracts real-world driving scenarios, focusing on essential features while omitting minor details.
- **Design and Implement Learning Agents:** Develop reinforcement learning agents for the cars, ensuring they can navigate the simulation environment effectively and adapt to changing conditions.
- **Conduct In-Depth Research:** Carry out comprehensive research on AI-driven cars, traffic dynamics, and human behaviour to inform the development of the simulation and learning agents.
- **Training and Performance Analysis:** Train the AI agents and conduct rigorous performance analysis on the relevant features such as safety, speed, efficiency, and other metrics.
- **Varied Situations:** Provide a series of different situations for learning (e.g., traffic lights, cross junctions, t-junctions, pedestrians) to challenge the AI cars to learn more general and applicable behaviours.
- **Adapt Agent Based On Feedback:** Tweak and adjust the learning agent rewards depending on its results to better improve its learning capability.

14.2 Additional Objectives (Exploration if Time Allows):

- **Implement More Advanced Road Features:** Introducing more advanced situations such as roundabouts, overtaking or multi-lane roads to challenge the AI and further demonstrate AI-human interactions.
- **Explore New Infrastructure:** Expand the simulation to explore potential new road infrastructure e.g., a lane specifically for AI, changing the speed limits for a simulation with only AI cars.
- **Reinforcement Learning Optimisation:** Investigate techniques for optimising the machine learning models used by AI agents to adapt more effectively to dynamic road conditions.
- **Traffic Optimisation Strategies:** Investigate strategies for optimising traffic flow and reducing congestion through the deployment of AI-driven cars, including adaptive traffic signal coordination and lane management.
- **Safety Assessment:** Compare the safety between the human and AI cars including results about crashes and other incidents.
- **Customisable Scenarios:** Develop a feature within the simulation that allows users to customise driving scenarios to conduct controlled experiments.

References

[1] L. Vogel and C. J. Bester, "A relationship between accident types and causes."

[2] D. O. Transport, "Reported road casualties great britain, annual report: 2022," 2022.

[3] D. Shinar, "Crash causes, countermeasures, and safety policy implications," *Accident Analysis and Prevention*, vol. 125, pp. 224–231, 2019.

[4] D. Hayashi, Y. Xu, T. Bando, and K. Takeda, "A predictive reward function for human-like driving based on a transition model of surrounding environment," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 7618–7624, 2019.

[5] T. Kohsuke, S. Yuta, O. Yuichi, and S. Taro, "Design of reward functions for rl-based high-speed autonomous driving," in *2022 IEEE 15th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pp. 32–37, 2022.

[6] S. Wang, D. Jia, and X. Weng, "Deep reinforcement learning for autonomous driving," 2019.

[7] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, "Autonomous vehicles: challenges, opportunities, and future implications for transportation policies," *Journal of Modern Transportation*, vol. 24, 2016.

[8] A. Karalakou, D. Troullinos, G. Chalkiadakis, and M. Papageorgiou, "Deep reinforcement learning reward function design for autonomous driving in lane-free traffic," *Systems*, vol. 11, no. 3, 2023.

[9] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2022.

[10] W. B. Knox, A. Allievi, H. Banzhaf, F. Schmitt, and P. Stone, "Reward (mis)design for autonomous driving," *Artificial Intelligence*, vol. 316, p. 103829, 2023.

[11] K. Muhammad, A. Ullah, J. Lloret, J. D. Ser, and V. H. C. de Albuquerque, "Deep learning for safe autonomous driving: Current challenges and future directions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4316–4336, 2021.

[12] V. Pierre, "Training with proximal policy optimization," 2018.

[13] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.