IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 4, May 2023

End to End Driving Assistant System using Proximal Policy Optimization

V Gokulakrishnan[1], P Nehru[2], S Sivasankar[3], A Velmurugan[4], R Ranjith[5], M Praveen Kumar[6]
Department of Computer Science and Engineering
Dhanalakhmi Srinivasan Engineering College, Perambalur, Tamil Nadu, India

Abstract: *Machine learning is critical in the advancement of autonomous driving technology and we will use state-of-the-art methods to train an agent to drive autonomously with an open-source simulator Carla we can conduct our experiment with a hyper-realistic urban simulation environment providing us with valuable data for training our models. autonomous driving systems struggle with road geometry and complexity many solutions rely on hand-crafted driving policies which are expensive and sub-optimal deep reinforcement learning DRL has shown promise in learning complex decision-making tasks this thesis explores using an on-policy DRL algorithm called proximal policy optimization (PPO) to navigate a predetermined route in a simulated driving environment the goal is to train an agent on a continuous state and action space our contribution is a ppo-based agent that can reliably drive in our Carla based Environment. the aim is to develop an end-to-end solution for autonomous driving that can avoid crashes and drive in the right direction this paper summarizes results and analyses and discusses further work to simplify the problem.*

Keywords: Machine Learning, Carla Simulator, Deep Reinforcement Learning, Proximal Policy Optimization

I. INTRODUCTION

Technology is evolving rapidly these days, one of which is transportation. driving automobiles and vehicles are equipped with a lot of technology, but there are still many car accidents in this world. According to this condition, many researchers have conducted research to solve the vehicle collision problem. One is self-driving, a car that can move around without a driver. Cars must avoid you as well as driving accidents in the environment. Vehicles are forced to know their surroundings and move based on some of the rules and situations. In fact, it is about collecting and analyzing information about the driver's behavior while driving. Essential as input to automated systems involved in the agent learning process. autonomous vehicles have the potential to improve transportation safety and efficiency studies have focused on autonomous driving reinforcement learning RL algorithms have shown potential for decision-making and control problems RL learns in a trial-and-error way without human supervision.

Autonomous driving experiments include multiple inputs such as steering angle, braking, acceleration, light detection and distance measurement (LIDAR), input and infrared (IR) sensor input. Therefore, many variations of the observed possibilities occur. Proposed a human-like driving system that enables self-driving cars to make human-like decisions using CNNs to perceive, perceive, and abstract information in input street scenes captured by onboards sensor. Instead of using a color image, it uses an unstable red-green-blue (RGB) image. they claimed to only use depth information in their system. recommended convolutional layers to extract create a feature map and compute bounding boxes and class probabilities as output layers. They argued that real-time inference for autonomous driving requires these constraints, thus reducing model size and increasing energy efficiency, and also proposed several end-to-end his DNN architectures. generate control actions for vehicle speed and steering wheel angle. An end-to-end control system was studied using multiple CNN architectures. They showed their results by reporting rounds for each network completion rate and pass smoothness.

In addition, the real-time lane detection of the autonomous driving simulation system made full use of CNN to detect left and right turn lanes. On the other hand, the use of DRL is just as important as the complexity of learning policies in high-dimensional environments. DRL cannot be separated from reinforcement learning (RL). RL is a promising

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 4, May 2023

solution, especially for policy promotion, predictive awareness, path and moving plan. Also, the integration of DRL into autonomous driving systems is strongly influenced by the benefits of RL discovery and exploitation capabilities. Using DRL, we experimented with capturing surrounding objects with a camera and determining the distance between an agent and an object with a LIDAR sensor. They produce an output related to the Q value. The reward and punishment values are recorded in the experience replay's Q-Value. Ultimately, experience replays serve to update weights in the network, influencing agents' ability to find their destination. A federated recurrent neural network (RNN) for integrating information into a proposed DRL framework for automated driving.

They used long-short-term memory (LSTM) networks to model long-term dependencies on previous states as a complement to current observations. We investigated whether DRL can train efficient self-driving cars by applying DRL in single- and multi-agent settings. this experiment tested and identified what kinds of neural networks work well and how single-agent models can improve multi-agent learning. proposed a hybrid approach for integrating his path planning pipeline into his vision-based DRL framework. We trained an agent to follow the path planner waypoints and defined a reward function that penalized the agent when it deviated from the path or collided. An open racing car simulator (TORCS) is used as the simulation environment. Additionally, implementations of Deep Q-Network (DQN) and DDPG in the DRL framework were compared. We used the open-source simulator Car Learning to Act (CARLA) as the simulation environment. In this paper, we proposed an autonomous driving system using PPO-based DRL in a simulation environment. The agent's mission in this system is to reach the destination as fast as possible without hitting a wall and to meet the challenge of crossing many roads to reach the destination.

II. POLICY BASED REINFORCEMENT LEARNING

Reinforcement learning can teach how an agent should behave by interacting with the environment in order to maximize the expected cumulative reward for a given task. There are two general categories of reinforcement learning algorithms: value-based methods and policy-based methods. Value-based methods can approximate the value function in a non-guideline-compliant way using neural networks, but the main advantage of guideline-based methods like the RL algorithm is that while directly optimizing the amount of profit, It is possible to maintain a stable state during that time. Function approximation. Therefore, our research focuses on policy-based RL methods. The most commonly used loss function for updating RL policies is of the form:

$$L^{PG}(\theta) - \hat{E}_t\left[log\pi_\theta(a_t|s_t)\widehat{A_t}\right] - (1)$$

where E_t is the expectation operator and $\pi_\theta$is the stochastic reinforcement learning politics is an estimator of the profit function of time step t while it may be tempting and easy to perform multiple optimization steps on this loss function, the widespread use of $L^{PG}(\theta)$can pose many challenges. sample Inefficiency, Exploration and Exploration Balance undesirably high variability of exploitation and learning content politics. Empirically, this often results in disruptive large-scale policy updates. distribution of observations and rewards at future time steps.

Compared to this basic loss function $L^{PG}(\theta)$ for updating the policy, some advanced policy-based algorithms such as TRPO and PPO offer advantages over traditional value-based and policy-based approaches. It adopts an actor critique structure that can combine The PPO algorithm is also much easier to implement, more general, and (empirically) has better sample complexity. In particular, PPO uses a neural network architecture that shares parameters between the policy and the value function, and its loss function also combines the policy surrogate and the value function error term defined as

$$L_t{}^{CLIP+VS+F}(\theta) - \hat{E}_t\left[L_t{}^{CLIP}(\theta) - c_1 L_t{}^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right] - (2)$$

where $L_t{}^{CLIP}$ is the truncated replacement target and c1, c2 are coefficient, $L_t{}^{VF}$, is the value function ($V_\theta(s_t)$-$V_t{}^{targ}$)^2 and S represents the entropy loss. Specifically, a clipped interchangeable lens $L_t{}^{CLIP}$ has the following format:

$$L_t{}^{CLIP}(\theta) - \hat{E}_t\left[min\left(r_t(\theta)\widehat{A_t}, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\widehat{A_t}\right)\right] - (3)$$

where ε is the hyperparameter and $r_t(\theta)$ is the likelihood ratio $r_t(\theta)$=$\pi_\theta(a_t|\, s_t)/\pi_{\theta_{old}}(a_t|s_t)$. Thus, the likelihood ratio r is bounded to 1 − ε or 1 + ε depending on whether the profit is positive or negative. This forms the restricted target after multiplication by the profit estimate $\widehat{A_t}$. The final value of $L_t{}^{CLIP}$ is the minimum of this pruned and unpruned target $r_t(\theta)\widehat{A_t}$. This effectively avoids large policy updates compared to the unpruned version. This is also known as the conservative policy iteration algorithm's loss function. The full PPO algorithm uses a fixed-length

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 4, May 2023

trajectory segment T for each iteration that runs the policy in T steps, with each of N parallel actors collecting data at every time step. A simplified version of the generalized profit estimate is taken and takes the form:

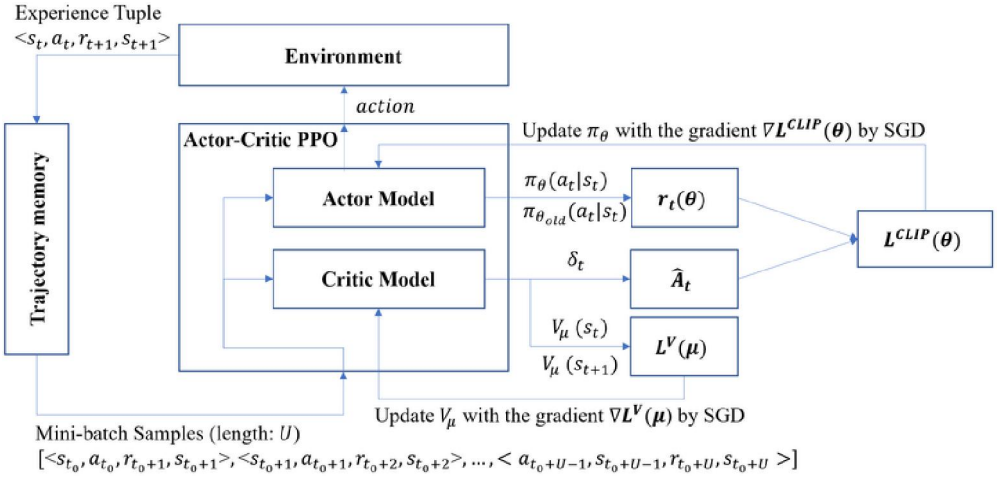

Figure2.1 Anactor-critic layout of PPO, the diagram.

$$\widehat{A_t} = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \quad -(4)$$

where $\delta_t = r_t + \gamma V(s_t + 1) - V(s_t)$ and γ is the discount factor. PPO then constitutes the losses in equation (1). (2) With these NT time steps of data, optimize them with mini-batch SGD for K epoch.

PPO requires an Actor $\pi(a_t|s_t\theta)$network and a Critic $V(s_t\theta_t)$network. We implement these as two different multilayer perceptron's (MLPs). This is a better representation of the Actor-Critic architecture shown inFigure 2.1. Since the input s_t is a vector, we use MLP here. The actor MLP consists of four fully connected layers of sizes 500, 300, 100, and a_{dim}. where $a_{dim}$is the size of the action space. Activation function is used for $tanh$ in every layer. The output of the last layer of this MLP represents the unscaled mean of the Gaussian distribution sampling the action, denoting the unscaled mean as$o_i$and the scaled mean as μ_i of action i^{th}. To get a scaled average, we must first point out that each agent's action is constrained to a given range of valid values. As a result, it makes sense to scale the output of the MLP to the range of each action. This is done with the following transformations:

$$\mu_i = \frac{\max(\min(o_i, 1) - 1) + 1}{2} - (5)$$

Passing the raw output of the last fully connected layer to the clipping function gives values in therange [−1, 1]. Adding 1 and dividing by 2 returns a value in the range [0, 1]. We also provide trainable parameters $\sigma_i$to define a multivariate Gaussian distribution that samples the action. Each $\sigma_i$is initialized to the chosen value. We call this σ_{init}. The μ_i layer weights are also initialized with a variance scale with a scale factor of 0.2. This is done to reduce the likelihood that the initial weights will significantly affect the policy. Since our environment uses a continuous action space to select actions, we sample a multivariate Gaussian distribution given by $b_y\sigma_i \sim N(\mu_i, \sigma_i)$, and choose $\sigma_i = \mu_i$in evaluation mode. increase. The critic is a simple MLP with four fully connected layers of sizes 500, 300, 100 and 1, and the output represents $V(s_t\theta_v) \approx R(s_t)$. Use tanh for the first three layers and no activation for the last layer. The last layer has no activations, so the critic can represent any value of $R(s_t)$.

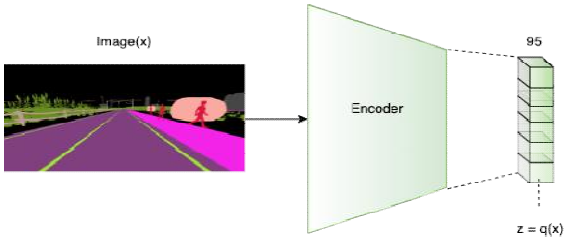

Figure2.2 This diagram depicts the PPO training pipe line

Note:allthevariablenamesaremissing the subscript t. Diagram inspired by the works

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 4, May 2023

To optimize the network, we use the action mean and standard deviation from the network to compute the log probability $log\pi_\theta(a|s)$for each action a understate π given state s.

III. METHODOLOGY

The reward function aims to integrate the main goal of this study, namely the development of auto lane-changing strategies with a focus on safety, efficiency and comfort. More specifically, these ideas are described as follows:

1) Comfort: Jerk rating (transverse and longitudinal direction);
2) Efficiency: Evaluation of driving time and relative distance to the target lane;
3) Security: Evaluate the risk of collisions and near misses.

The reward function for comfort can be expressed as:

$$R_{com}f(t) = -\alpha.a_x(t)^2 - \beta.\dot{a}y(t)^2 \quad -(6)$$

where ax and ay are the horizontal and vertical jerk. shock. This reward feature was introduced to avoid sudden vehicle acceleration or deceleration that may cause it discomfort of car occupants.in terms of efficiency, ego vehicles should be out there move as fast as possible without crossing the desired track speed limit so efficiency pays off defined as

$$\begin{cases} R_{time}(t) = -\delta_t \\ R_{lane}(t) = -|P_x - P_x*| \\ R_{speed}(t) = -|V_y - V_{desired}| \end{cases}$$

$$R_{eff}(t) = w_t.R_{time}(t) + w_l.R_{lane}(t) + w_s.R_{speed}(t) \text{- (7)}$$

where $R_{time}(t)$ is the time-related sub-reward, $R_{lane}(t)$ is the difference between the lateral position P_x of the vehicle relative to the target lateral position P_x*, Longitudinal velocity P_x. while $R_{speed}(t)$ represents the difference between the ego-vehicle's longitudinal speed P_x with respect to the targeted longitudinal speed $V_{desired}$.The overall efficiency reward $R_{eff}(t)$ can be computed by adjusting the sum of these three rewards by their respective weights w_t., w_l or w_s. To make auto lane changes safer, we are introducing a near-collision penalty feature instead of just returning a penalty when a collision actually occurs. In this case, the ego vehicle can learn to abort the lane change maneuver if the relative distance deviates from the lane change maneuver indicates an imminent collision.

$$R_{safty} = \begin{cases} R_{near-collision} & if\ D < 20\ m \\ -200 & if\ collision \end{cases} \text{- (8)}$$

where D is the relative distance from the ego vehicle to the neighbouring vehicle.

IV. SIMULATION EXPERIMENT

CARLA is an urban driving simulator that tests algorithms. in more demanding and realistic driving scenarios. CARLA is an open-source simulator for autonomous driving research developed with Unreal Engine 4. This simulator focuses on recreating realistic driving environments in common urban driving scenarios. 7 different cards are included, and 3 more are sold separately for a total of 10 cards. As a server-client system, the server renders simulations on command from the client and the physics engine.

Figure 4.1 image from the environment spectating camera.

A Python API-based client, on the other hand, provides steering, throttle, and braking. command the server and receive sensor information. For example, steering control has a range of [-1.0, 1.0], while throttle and brake commands have a

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 4, May 2023

range of [0.0, 1.0]. Additionally, we provide a generic API that allows us to create vehicles, cameras, and other sensors that we can use at our discretion.

We describe the vehicle and sensor setup used in our environment. this vehicle is equipped with a front camera mounted on the front of the vehicle and a surround camera for observation. The front camera outputs a 160x80 semantically segmented image**.** Figure 4.1 shows an example of viewer camera output. CARLA offers a wide range of sensors that can be attached to the driving agent and receive data at any time step.

Throughout our training, we have focused on some very important variables. factors/indicators to focus on in this section:

Average distance from road center during training cycle

Comparison of Town02 and Town07 (urban and semi-urban).

have a unique landscape

Episode length during training

$\sigma init$ (search sounds) changes in training

Asynchronous nature of environment setup

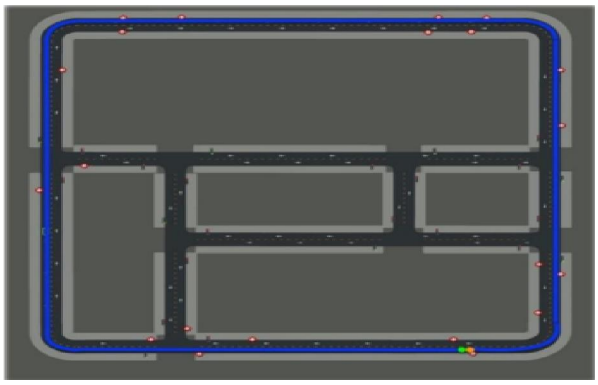

Figure 4.2 Town 2 top-view and path followed by our agent highlighted in blue.

Figure 4.3 Town 7 top-view and path followed by our agent highlighted in blue.

First, let's take a look at these two cities and our basic chart. default navigation route. In the maps in Figures 4.2 and 4.3 below, the agent is driving counter clockwise on the blue navigation route. Starts at the position of the orange dot and ends at the position of the green dot.

V. CONCLUSION AND FUTURE WORK

This section examines and summarizes our results and offerings. suggestions on how we can improve our work and areas of autonomous driving research that we should focus on in the future. this work may be extended in various ways in the future. Experimenting other state-of-the-art off-policy algorithms such as TD3, SAC, and DDPG used in it is a continuous domain and more tolerant to hyperparameter perturbations. Continuous action domains can also be discretised and formulated for Duelling Deep Q networks, Double Deep Q networks, etc. it is also important to resolve

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
International Open-Access, Double-Blind, Peer-Reviewed, Refereed, Multidisciplinary Online Journal
Impact Factor: 7.301
Volume 3, Issue 4, May 2023

some outstanding issues for the best possible results. reinforcement learning and formulation of optimization problems by imitation, etc. learning goals. This can be further improved by combining these two learning paradigms

REFERENCES

[1]. Imam, N.Y. and Rachmawati, E., "Autonomous driving system using proximal policy optimization in deep reinforcement learning", IAES International Journal of Artificial Intelligence, 12(1), p.422.2023.

[2]. Sharma, Arjit, and Sahil Sharma. "Wad: A deep reinforcement learning agent for urban autonomous driving." arXiv preprint arXiv:2108.12134,.2021

[3]. Quang Tran, D. and Bae, S.H., "Proximal policy optimization through a deep reinforcement learning framework for multiple autonomous vehicles at a non-signalized intersection", Applied Sciences, vol 10(16), p.5722. 2020.

[4]. Wen, Lu, Jingliang Duan, Shengbo Eben Li, Shaobing Xu, and Huei Peng. "Safe reinforcement learning for autonomous vehicles through parallel constrained policy optimization." IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pp. 1-7. IEEE, 2020.

[5]. Ye, F., Cheng, X., Wang, P., Chan, C. Y., & Zhang, J," Automated Lane change strategy using proximal policy optimization-based deep reinforcement learning", IEEE Intelligent Vehicles Symposium (IV) (pp. 1746-1752). IEEE.2020.

[6]. You, C., Lu, J., Filev, D., & Tsiotras, P. "Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning.", Robotics and Autonomous Systems, vol 114, pp.1-18.2019.

[7]. Wei, H., Liu, X., Mashayekhy, L., & Decker, K. "Mixed-autonomy traffic control with proximal policy optimization" IEEE Vehicular Networking Conference (VNC) (pp. 1-8). IEEE 2019

[8]. Schutera, Mark, Niklas Goby, Dirk Neumann, and Markus Reischl. "Transfer learning versus multi-agent learning regarding distributed decision-making in highway traffic." arXiv preprint arXiv:1810. 08515, 2018.

[9]. Pan, Yunpeng, Ching-An Cheng, Kamil Saigol, Keuntaek Lee, Xinyan Yan, EvangelosTheodorou, and Byron Boots. "Agile autonomous driving using end-to-end deep imitation learning." arXiv preprint arXiv:1709.07174,.2017

[10]. Gupta, J. K., Egorov, M., &Kochenderfer, M "Cooperative multi-agent control using deep reinforcement learning." In Autonomous Agents and Multiagent Systems: AAMAS Workshops, 16 (pp. 66-83). 2017