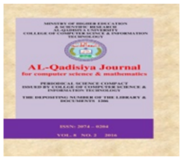
Available online at www.qu.edu.iq/journalcm
JOURNAL OF AL-QADISIYAH FOR COMPUTER SCIENCE AND MATHEMATICS
ISSN:2521-3504(online) ISSN:2074-0204(print)



Self- Driving With Deep Reinforcement (Survey)

Hayder Salah Abdalameer[1] and "Ali Abid Al-Shammary[2]

[1] *College of Computer Science and Information Technology, University of Al-Qadisiyah, Iraq* ***Email*** *: hayder.salah@qu.edu.iq*

[2] *College of Computer Science and Information Technology, University of Al-Qadisiyah, Iraq .****Email****: ali.obied@qu.edu.iq*

ARTICLE INFO

Article history:
Received: 03 /06/2022
Rrevised form: 08 /07/2022
Accepted : 17 /07/2022
Available online: 12 /08/2022

Keywords:

Autonomous driving, Deep Reinforcement Learning, Reinforcement learning , Time-to-Collision (TTC), conventional nural Network , Light Detection And Ranging (LiDAR) .

ABSTRACT

We will focus concentrating on the scientific and technological challenges that the development of autonomous vehicles brings to a number of different manufacturers and research organizations (AVs). It is anticipated that automobiles operated by humans will continue to be a common sight on the roads for the foreseeable future; nevertheless, it is possible that autonomous vehicles may coexist in the same traffic environments. In order to keep traffic moving smoothly throughout the various sorts of mixed-use zones, autonomous vehicles (AVs) need to have driving rules and negotiation abilities that are analogous to those used by humans. In order to develop driving abilities that are comparable to those of humans, model-free deep reinforcement learning is used to simulate the actions of a skilled human driver. In this simulation, the difficulty of avoiding static obstacles on a two-lane roadway is investigated .

MSC..

https://doi.org/10.29304/jqcm.2022.14.3.982

1. Introduction

Modern society, the economy, and the environment stand to benefit greatly from self-driving automobiles' revolutionary capabilities in the twenty-first century. Various legal, technological, and computational challenges must be overcome before completely autonomous cars may be used (AVs). Industrial and academic research, for example, has been focusing on the

∗Corresponding author

Email addresses:

Communicated by 'sub etitor'

incorporation of prior knowledge, a better understanding of the car's surrounding context (such as traffic signs) as well as the decision-making process that results in driving policies for different environments and scenarios. [1]. It is feasible that conventional cars and vehicles with (consistently) developing autonomous capabilities may share the road in the future for a length of time, based on present studies. In order to ensure safe and efficient traffic flow, AV driving laws will be restricted in certain mixed traffic scenarios. It is our opinion that autonomous cars should be used to drive in an almost human-like way. First and foremost, human drivers must anticipate and predict the behaviors of their fellow road users since any unexpected conduct might spark a conflict or pose a danger to the safety of the whole road community. Second, although AV driving may save fuel or time, human passengers may be bothered by driving characteristics like as unexpected stops, reckless driving, or extreme caution. It's because of this that machine-learning algorithms have to include human elements. Programming an AV system is a difficult undertaking. Sensor and map inputs, to mention a few uses, are employed in a full AV system, which is usually composed of numerous linked engineering stacks or layers [2]. A mix of data-driven and rule-based procedures has emerged in the last decade as machine learning approaches have been incorporated to these stacks [3].

Reinforcement learning in uncertain, large, and stochastic situations has been made possible via the use of deep reinforcement learning (DRL), a technique that blends deep learning with reinforcement learning (RL).

Deep learning architectures in autonomous driving (AD) systems have allowed several perception-level tasks to be performed with high accuracy. It's no longer sufficient to use typical supervised learning approaches when it comes to autonomous driving systems. An autonomous driving agent's behaviour is anticipated by the environment in which it operates, such as the job of calculating a city's ideal driving speed. Supervisory signals such as time to collision (TTC) and lateral error relative to the agent's ideal trajectory characterize the agent's dynamics and the uncertainty of its surroundings. Defining the stochastic cost function that must be maximized is the first step in solving these kinds of problems. As the agent travels through its surroundings, it must pick up on new configurations and anticipate the best decision at any given moment [4]. A large combinatorial space exists here since there are so many different configurations in which the agent and environment might be monitored. This is a sequential issue that we're aiming to solve in each of these examples.

It is a requirement of conventional Reinforcement Learning (RL) settings that the agent learn and represent its environment while also responding appropriately when each moment is supplied. It's all about the policy when it comes to deciding what the best course of action is.

Here, we'll look at how reinforcement learning may be used to improve driving behavior, perception, route planning, and low-level controller design, just to name a few areas where it's a viable option.

Last but not least, we want to motivate users by emphasizing the significant computing challenges and risks involved with current RL approaches, such as "imitation learning and deep Q learning. According to Figure 1, we can see that the use of reinforcement learning (RL) in the autonomous driving or self-driving sector is a relatively recent issue. RL/DRL algorithms" have been more popular in recent years, however this has led to several implementation and deployment challenges in the actual world .

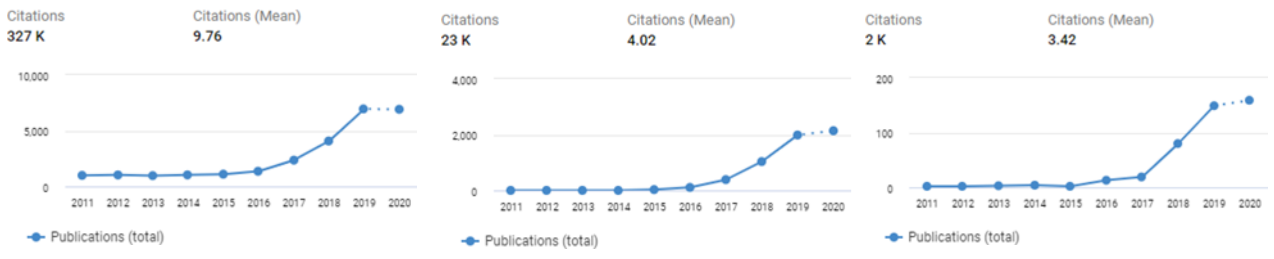


Fig. 1. "Trend of publications for keywords" 1. "reinforcement learning", "2.deep reinforcement, and 3.reinforcement learning AND" ("autonomous cars" OR "autonomous vehicles" OR "self driving") "for academic publication trends from this" [5].

The following is a summary of this work's main contributions:

• It tells the little-known narrative of RL's history in the automobile sector.

• RL's usage in different autonomous driving tasks has been well analyzed in the literature.

• Real-world autonomous driving challenges and opportunities are discussed.

2. The Concepts Of Agent

An agent-based system, which is part of Artificial Intelligence, is a growing area for controlling and performing basic or complex issues.

2.1. Definition Of An Agent

An agent is a computer program that acts on behalf of its user or owner (an agent is mainly a particular software component). An agent must possess the following characteristics: (Autonomous, proactive, responsive, communicative, and so forth.) Agents have a number of characteristics that help them do a task. A self-contained agent may adapt to and learn from its environment. It may also respond to changes in its environment and make decisions in order to achieve its goals. The ability to work without the intervention of people or another model is widely agreed upon as the essential attribute of the agent. Furthermore, the significance of particular features is determined by the agent's spatial location.

2.2. Characteristic Of Intelligent Software Agents

One of the reasons for the diverse definitions of the intelligent agent was the researchers' differing interpretations of some of the qualities (what constitutes autonomy). What we've chosen from the qualities is the most essential focus of studies in defining the intelligent agent, either directly or indirectly. However, they are not the sole features that all researchers have adopted[6].

2.2.1. Autonomy and Intelligence

Researchers believe that performing autonomous behaviors is the most essential factor. In addition to his initiative in accomplishing his goals, an autonomous agent has the ability to govern his actions and behavior. It acts without the intervention of others (users or agents). Some researchers conflate autonomy and intellect. In other words, they see the agent's ability to take initiative as a benefit of being an intelligent agent. Julia, according to Foner, is clever since it is autonomous and has initiative.

2.2.2. Learning

Intelligence is a noun with the property of being able to learn. Is it necessary for the agent to learn in order for it to be intelligent? While some studies believe that learning is a key characteristic of an agent, others never mention it. Because it is impossible to predict what an agent will confront in a dynamic environment, the agent stands out when it has the ability to adapt and learn from his surroundings.

2.2.3. Co-operation

Complex problems may necessitate the collaboration of multiple agents. The goals to be reached and fulfilled are agreed upon by the cooperating agents. The key differentiating feature of agents is their cooperation. Foner is concerned with issue resolution in which the agent and the user collaborate. Cooperation can take place on two levels: between the user and the agent, and between the agent and other agents. Tasks are broken down into smaller, distributed challenges for agents to solve through agent collaboration. This collaboration occurs when the helpers communicate with one another.

2.2.4. Lifelike

Agents are lifelike; it is the delusional style, therefore emotions and social interactions are within the agent’s capabilities. To generate convincing characters, a variety of techniques and procedures are necessary, including speech recognition and animation. It is vital to comprehend them in addition to psychological sociology and dialogue procedures. Agents having human traits, like as emotion, that can be displayed as a realistic human face, are intriguing. According to some academics, presenting realistic features through an agent is a must. It’s a lifelike agent since it’s a spontaneous, not a mechanical, agent that acts in unpredictable ways.

2.2.5. Mobility

It refers to the transfer of data across a network, such as the Internet. Is the agent capable of performing this action? Moving the agent across the network, according to researchers, gives a novel mechanism for recovering data and transactions. The agent can switch between servers to determine the optimum route for you and provide you with the results. The negative is that the agent may present your computer with numerous options that waste time and force you to stay connected to the Internet.

3. component for AD system

With the pipeline from sensor stream to control actuation shown in Figure 2, AD systems are depicted in their most basic form A contemporary autonomous driving system's sensor architecture includes a variety of cameras, radars, and laser range finders (LLDARs), as well as a GPS-GNSS system for precise location and inertial measurement units (IMUs) for capturing 3D poses of the vehicle in space [7]. Using this perception module, a decision-making system may create a mid-level picture of how the environment is now functioning (such as a bird's-eye view map of all obstacles and agents). There are perceptional discrepancies all the way up the informational food chain. The utilization of several sources increases detection confidence, which is critical for public safety. This is achieved by a combination of several perception tasks like semantic segmentation [7], [8], "motion estimation" [9], "depth estimation" [10], "soiling detection" [11], "etc which can be efficiently unified into a multi-task model" [12], [13].

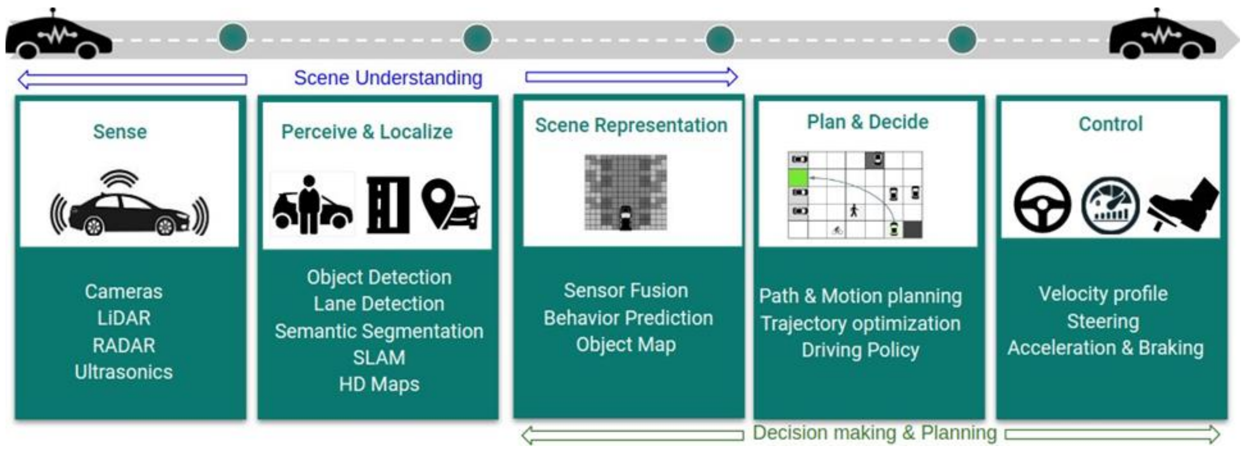


Fig. 2. “Standard components in a modern autonomous driving systems pipeline listing the various tasks. The key problems addressed by these modules are Scene Understanding, Decision and Planning”[7][8].

3.1. Scene Understanding

A link between the perception module's mid-level abstract representation of the perception state and the high-level action or decision-making module is established by this essential module. In terms of idea, there are three actions included in this module: Decision-making and planning are shown in Figure 2. Detection and localization algorithms are included into the first module, which aims to offer a deeper understanding of the scene.

3.2. Localization and Mapping

Mapping is a key component of automated driving[14]. One way to find out exactly where your car is right now is to map out a section of the region.

3.3. Planning and Driving policy

Trajectory planning is a key component in the development of self-driving vehicles. It is necessary to use this module in order to construct motion-level instructions that lead the agent given a route-level plan derived using HD or GPS-based maps. To move an agent from one position to another, the classical motion planning method uses just the two axes of translation and rotation[15].

3.4. Control

Based on pre-determined maps, such as Google Maps, or professional driving recordings of the same values at each waypoint, a controller decides the right speed, steering angle and braking actions at each waypoint. Trajectory tracking, on the other hand, employs a time-based model of the vehicle's dynamics to follow the waypoints sequentially.

Modern vehicle control systems are based on classical optimum control theory, which asserts that the goal of optimal control is to minimize a cost function x = f (x(t),u(t)) across a set of states x(t) and actions u(t)) (t). Predetermined control input is common in most circumstances.

x Xfree [16] is constrained to a small time horizon in a viable state space x Closed loop control methods like PID (proportional integral-derivative) controllers and MPC are used to regulate the velocity (Model predictive control). There are three terms in a PID's goal: present error (proportional term), the consequence of previous mistakes (integral term), and any future errors (proportional term) (derivative term). This class of methods is designed to stabilize the vehicle's behavior while it follows a predetermined route [17].

4. REINFORCEMENT LEARNING

Software that uses machine learning (ML) to improve its performance on a particular task is known as "learning by doing" or "learning by doing well." supervised learning, unsupervised learning, and reinforcement learning are all examples of machine learning algorithms (RL)[18] . While in supervised learning methods, the model is often trained using labeled data in order to perform classification or regression, techniques like density estimation or clustering are widely used in unsupervised learning algorithms. Autonomous agents use the RL paradigm to improve their performance on specific tasks, on the other hand. Whatever may be considered as sensing and acting on its surroundings through sensors, according to Russell and Norvig [19]. The performance of RL agents isn't evaluated by an expert, but rather by a reward function R. Every time it encounters a situation, the agent may choose an action to do, and if that action is seen to be useful by its surroundings, it will be rewarded. The agent's goal is to maximize the overall benefits it receives throughout the course of its life. The agent may progressively increase its long-term reward by using information obtained about the projected utility (i.e. discounted sum of expected future benefits) of different stateaction combinations. One of the most challenging elements of reinforcement learning is managing the trade-off between exploration and exploitation. To get the most out of its efforts, an agent must make use of the information it has gathered to choose actions that are known to provide high returns [19]

A MDP is a tuple consisting of a "set S of states, a set A of actions, a transition function T, and a reward function R" [20], "i.e. a tuple < S, A,T,R >. When in any state s ∈ S, selecting an action a ∈ A will result in the environment entering a new state s 0 ∈ S with a transition probability T(s, a, s 0) ∈ (0,1), and give a reward R(s, a). This process is illustrated in Fig. 3. The stochastic policy π : S → D is a mapping from the state space to a probability over the set of actions, and π (a|s) represents the probability of choosing action a at state s. The goal is to find the optimal policy π ∗ , which results in the highest expected sum of discounted rewards" [21]:

$$\pi * = \underset{\pi}{\operatorname{argmax}}\ E\pi \left\{ \sum_{k=0}^{H-1} \gamma^k r_{k+1} \mid s_0 = s \right\}, \qquad (1)$$

$$:= V\pi(s)$$

"for all states s ∈ S, where rk = R(sk,ak) is the reward at time k and V π (s), the ‘value function ’ at state s following a policy π , is the expected ‘return’ (or ‘utility’) when starting at s and following the policy π thereafter" [22]. "An important, related concept is the action-value function, a.k.a.‘Q-function’ defined as:

$$Q\pi(s,a) = E\pi \left\{ \sum_{k=0}^{H-1} \gamma^k r_{k+1} \mid s_0 = s, a_0 = a \right\}. \qquad (2)"$$

Future incentives are discounted based on the discount factor γ ∈ [0,1]. Myopic behavior is encouraged by low values γ , which lead agents to focus only on short-term gains. Conversely, agents with high values are more long-term oriented and seek to maximize their long-term gains. When talking about the MDP's horizon, we're referring to the amount of time steps. While H has a limited value in episodic domains, H is limitless in infinite-horizon situations H = ∞. After a certain amount of time steps, or when an agent achieves a predetermined target state, episodic domains may be terminated. [22] An episodic domain's "terminal state" refers to the last state that may be reached. Smaller discount factors may be utilized in infinite-horizon or goal-oriented domains in order to create a balance between short- and long-term advantages. Discount factors of (close to) 1 can motivate agents to concentrate on accomplishing the goal. Any beginning state may be utilized to determine the largest predicted discounted sum of rewards if the optimal policy for an MDP is known.

4.1. Value-based methods

Q-learning is a well-known RL algorithm that's been around for a while. Rather than using any models, this TD approach teaches itself how to assess the utility of specific state-action pairs as a whole (Q-functions defined in Eqn. 2). It has been shown that Q-learning can converge to the optimal state-action values for an MDP with a probability of one when all actions in all states are sampled infinitely often and the state-action values are represented discretely[21]. Quantitative learning (Q-learning) may learn (almost) optimum values for state-action combinations provided enough samples are collected for each combination. If a Q-learning agent converges to the optimal Q values for an MDP and then chooses actions greedily, it will get the same anticipated sum of discounted rewards as the value function with π∗ . Q-learning requires agents to update their Q values in accordance with the following rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a| \in A} Q(s|, a|) - Q(s,a)], \qquad (3)”$$

“where Q(s,a) is an estimate of the utility of selecting action a in state s, α ∈ [0,1] is the learning rate which controls the degree to which Q values are updated at each time step, and γ ∈ [0,1] is the same discount factor used in Eqn. 1. The theoretical guarantees of Q-learning hold with any arbitrary initial Q values” [21]; As a result, it is possible to learn the optimum Q values for an MDP beginning with any initial estimate of the action value function. Each Q(s,a) delivers a maximum potential reward, or it may be pessimistic (minimum), or it can use knowledge of the issue to guarantee quicker convergence. If you're looking for a non-linear approximation of the Q function across high-dimensional state spaces[23] , you may utilize deep neural networks (DNNs) as an approximation of the Q function. pixels in a frame of an Atari game, for instance. All behaviors are predicted by a neural network without the use of any explicit domain-specific data or features. In practice For example, DQN employs replays of previous experiences to break the association between upcoming samples and to increase sampling speed. The parameters of the target network for DQN are fixed for a period of iterations while the parameters of the online network are modified [24] to increase the stability of the system.

4.2. Policy-based methods

"The distinction between value-based and policy-based methods boils down to who bears the responsibility of optimality. While both methods must propose actions and evaluate the resulting behavior, value-based methods focus on evaluating the optimal cumulative reward and ensuring that a policy follows the recommendations, policy-based methods aim to directly estimate the optimal policy, with the value being a secondary consideration if at all calculated. A neural network is commonly used to parameterize a policy πθ. Gradient descent is used in policy gradient methods to estimate the policy parameters that maximize the expected reward. The result can be either a stochastic or deterministic policy, with actions chosen by sampling"[25]. Continuous action spaces are common in real-world applications. Reinforcement learning in domains with continuous actions is enabled by deterministic policy gradient (DPG) techniques [25] [22]. The Reinforce [26] algorithm is a policy-based technique that is simple to use. The whole reward is discounted. . $gt = P\ H1\ k = 0\ \gamma\ k\ rk{+}t{+}1$ No estimator is necessary for policy evaluation because by playing the, one time step is calculated the complete program. The parameters are updated in the performance gradient's direction:

$$\theta \leftarrow \theta + \alpha \gamma t\ g \nabla \log \pi\theta(a|s), \qquad (4)$$

A steady incremental update has a learning rate of α. We favor state-action combinations that provide the greatest outcomes since that's what makes sense to us. trust region policy optimization [27] prohibits trust region policy The risk of an inaccurate update is reduced since new regulations aren't allowed to deviate too much from old ones. TRPO's surrogate objective function is based on this principle. The Kullback-Leibler (KL) divergence between existing and new proposed policies should be reduced for each policy gradient change. Using this method, policy performance steadily improves..

5. REINFORCEMENT LEARNING FOR AUTONOMOUS DRIVING TASKS

Path planning, trajectory optimization, motion planning, and dynamic path planning are some of the autonomous driving tasks, as are scenario-based policy learning for highways, intersections, merging/dividing, and intent prediction from expert data for traffic actors like pedestrians and vehicles. Other autonomous driving tasks include controller optimization, reward learning, and inverse reinforcement learning from expert data[28]. Let's take a look at the state space, action space, and incentive schemes in the context of "autonomous driving" before diving into DRL's applicability to AD tasks.

5.1. State Spaces, Action Spaces and Rewards

DRL may be used to autonomous driving tasks only if proper state spaces, action spaces, and function relevance are designed and rewarded. When it comes to autonomous driving, there are a variety of status and action representations that have been studied. It is usual for autonomous vehicles and any obstacles within their sensor view area to have the following state space properties: location, direction, and velocity. State space variance may be eliminated by constructing a Cartesian or Polar occupancy grid around the ego vehicle Lane numbers, route curvature, the ego-previous vehicle's and future trajectory, longitudinal data such as Time-to-Collision (TTC), and scene data such as traffic laws and signal placements are all provided in the lane data section.

Condensed abstracted data reduces the complexity of the state space by removing finer contextual information from raw sensor data, such as camera images, LiDAR Light Detection And Ranging, and radar. As long as a 2D bird's eye view (BEV) is used, it is independent of the sensor and close to the scene's physical layout. Top-down view of scene with grid of occupancy, historical and predicted trajectory and semantic data such as traffic signal placements is shown in Figure 3. This intermediate format retains the spatial organization of roadways when graph-based representations fail. Some simulations, such as Carla and Flow, provide this point of view. Vehicle policies must be capable of controlling a wide range of devices. Continuous-valued actuators for vehicle control include the steering angle, throttle, and brake.. The alternative to discretization for actuators' continuous values is to use DRL algorithms that develop a policy on the fly (e.g. DDPG). It is also possible for agents to leverage the Temporal abstractions choices framework [29] in order to speed up the process of picking actions. These sub-policies may be used to extend a basic action across a lengthy period of time. Designing incentive mechanisms for DRL agents for self-driving vehicles is still a work in progress. Distance traveled to a destination [30], ego vehicle speed [30]–[31], ego vehicle stop [32], collisions with other road users or scene objects [30], [33], infractions on sidewalks [30], keeping in lane, and maintaining comfort and stability while avoiding extreme acceleration, braking, or steering [31], [32], and following traffic rules [33] are some examples of criteria for AD tasks.

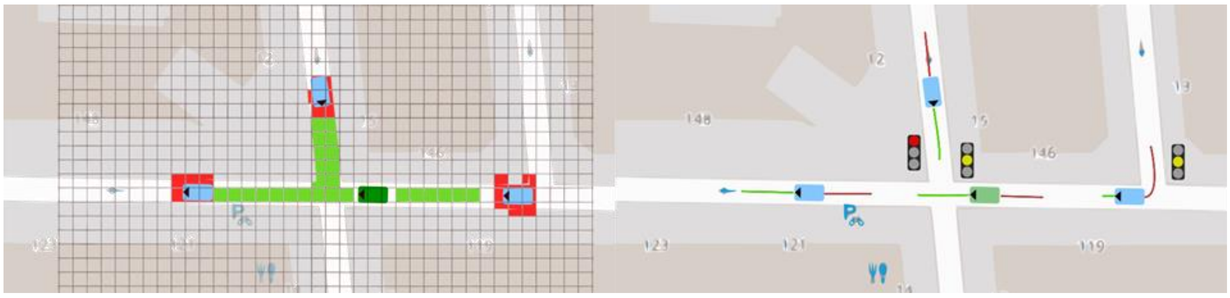


Fig. 3. "Bird Eye View (BEV) 2D representation of a driving scene. Left demonstrates an occupancy grid. Right shows the combination of semantic information (traffic lights) with past (red) and projected (green) trajectories. The ego car is represented by a green rectangle in both images"[33].

5.2. Simulator & Scenario generation tools

supervised learning configuration with training sets containing picture, label pairings for diverse modalities is addressed in autonomous driving datasets. Reinforcement learning necessitates a setting in which state-action pairings can be retrieved while modeling the dynamics of the vehicle's state, environment, and stochasticity in the environment's and agent's movements and actions, respectively. Reinforcement learning algorithms are trained and validated using a variety of simulators. Various high-fidelity perception simulators capable of imitating cameras, LiDARs, and radar . Some simulators can also provide information on the vehicle's status and dynamics. [34] provides readers with a comprehensive overview of the sensors and simulations used in the autonomous driving sector. Before moving on to costly evaluations in the real world, learned driving policies are stress tested in virtual scenarios. [35] proposes a multi-fidelity reinforcement learning (MFRL) system in which various simulators are available. In MFRL, a cascade of simulators with increasing realism is utilized to depict state dynamics (and hence computational cost) in order to train and validate RL algorithms while obtaining near-optimal policies for the real world with fewer expensive real-world samples using a remote-controlled car. CARLA Challenge [36], a Carla simulator-based AD competition with pre-crash situations described in a National Highway Traffic Safety Administration report [37], is a pre-crash AD competition based on the Carla simulator. The ego-vehicle loses control, the ego-vehicle reacts to an unseen obstruction, the ego-vehicle shifts lanes to avoid a sluggish leading vehicle, and other scenarios are used to test the systems. The total distance traveled in various circuits is used to calculate agents' scores, with total points deducted for infractions.

6. REAL WORLD CHALLENGES AND FUTURE PERSPECTIVES

This section presents and discusses the issues of conducting reinforcement learning for real-world autonomous driving, as well as the relevant research methodologies for tackling them.

6.1. Validating RL systems

Using a supervised learning method, an autonomous driving system was suggested in a paper by I. Lee [38]. Autonomous driving with minimum hardware dependencies was promoted using low-cost components to carry out artificial neural network learning. As opposed to just following the pre-programmed routes, our self-driving car was able to detect and avoid roadblocks based on pictures captured by the front-facing cameras. Researchers employed a server computer with a tiny embedded system and four GPUs in the experiment, along with an actual compact automobile simulator. An image-based neural network with cameras performed far better in real-world contexts than a single sensor at classifying images, even if a combination of many sensors beat a single sensor. This data was utilized to perform reinforcement learning exercises using images captured by the vehicles' front-facing cameras.

Using a convolutional neural network, S. Park et al. [39] altered vehicle steering parameters. In this experiment, CNN inputs from a black box were used to regulate the steering settings of an autonomous driving automobile. Three layers, including convolutional and pooling layers, were included in the CNN framework, and an algorithm with two entirely connected layers was shown. Based on CNN's photos, precise vanishing point coordinates were used to calculate steering angles. Although it is difficult to find stable vanishing points due to several circumstances, such as the driving environment and lane disappearances, the CNN acquired steering angle prediction errors within a permitted scope and offered significant results.

Autonomous driving in the actual world was made possible by X. Pan et al.[40] using virtual worlds and reinforcement learning. A simulated driving environment identical to actual roads was used for reinforcement learning training, and the results were compared to those obtained on real roads. Reinforcement learning techniques used in the research included the standard A3C [41], the reinforcement learning method presented therein, and the random domains approach. Reinforcement learning was utilized to teach autonomous driving in a simulated environment using three different tactics, and the results were compared for accuracy. According to the research, the study's technique had a precision rate of 43.4%, which was greater than the precision rate of 28.33% obtained using virtual reinforcement training with no actual data.

Prior research has employed an end-to-end technique, but Li and his colleagues [42] suggest a steering system that uses a "recognition module" and a "control module" to review driving pictures. In order to use a multiple classification neural network to forecast driving direction, the "recognition module" was given inputs of images taken from the driver's point of view. In order to make choices, the "control module" used reinforcement learning. The development of this self-driving car model included the use of a prototype agent, reinforcement learning, and deep learning. Based on the findings of the performance assessment in the virtual environment, a recognition module was used to predict lane characteristics in a steady and accurate way. A unique AI algorithm was employed for reinforcement learning training on this control module, which worked effectively in a variety of driving scenarios.

For the study of autonomous driving, H. Yi et al. [43] used a multi-agent method, with 16 agents learning at the same time, employing DDPG instead of DQN. Performance was assessed by averaging the compensation values achieved throughout the learning period. The negative readings at the start of the experiment meant that little progress could be made. However, as the number of training sessions grew, compensation also increased, showing that autonomous driving was becoming more and more common. Despite the fact that wages rose at the same rate, compensation did not follow up. There were also periods of negative compensation values, despite strong learning, which was connected to inaccurate compensation value assessments for brake values. This issue is being addressed by a collision prevention model in the present research.

7. CONCLUSION

Reinforcement learning is still a problem that is now being debated in the field of autonomous driving in the real world. Despite a few successful commercial implementations, there is very little literature or large-scale public datasets available. So, we came up with the idea of organizing RL applications for autonomous vehicles. Situations in which a vehicle is driven autonomously need the negotiating and dynamic decision-making skills that RL is known for. The development of mature solutions is, however, hampered by a number of challenges, which we explore in full. There is a theoretical foundation for reinforcement learning and a complete literature review on employing reinforcement learning in autonomous driving tasks in this study.

Because of the sensitivity of the results to hyper-parameter choices, reinforcement learning discoveries are famously difficult to reproduce. Researchers and practitioners alike need a strong starting point from which well-known reinforcement learning algorithms may be implemented, documented, and properly evaluated for their own benefit.

Future challenges in autonomous driving include the development of multi-agent reinforcement learning systems, which have received little attention so far.

Furthermore, the implementation of RL algorithms is a challenge for both academics and practitioners. We provide examples of widely known and actively used open-source RLI systems that enable for the assessment of RL approaches as well as the development and expansion of new RLI systems. Finally, it is our goal that this overview article will serve as a springboard for further investigation and development.

REFERENCES

[1] S. Shalev-Shwartz, S. Shammah, and A. Shashua, “On a formal model of safe and scalable self-driving cars,” arXiv preprint arXiv:1708.06374, 2017.

[2] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion,Ahmad A. Al Sallab2, Senthil Yogamani and Patrick Pérez, “Deep Reinforcement Learning for Autonomous Driving” in 2021 arXiv:2002.00444v2 .

[3] L. Fridman, D. E. Brown, M. Glazer, W. Angell, S. Dodd, B. Jenik, J. Terwilliger, A. Patsekin, J. Kindelsberger, L. Ding, et al., “Mit advanced vehicle technology study: Large-scale naturalistic driving study of driver behavior and interaction with automation,” IEEE Access, vol. 7, pp. 102021–102038, 2019.

[4] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction (Second Edition). MIT Press, 2018.

[5] T. D. Team. Dimensions publication trends. [Online]. Available: https://app.dimensions.ai/discover/publication

[6] A.M. Kareem, A.Obied, Testbed for intelligent agent : A survey , Journal of Al-Qadisiyah for computer science and mathematics 13 (2021) Page-23.

[7] M. Siam, S. Elkerdawy, M. Jagersand, and S. Yogamani, “Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges,” in 2017 IEEE 20th international conference on intelligent transportation systems (ITSC). IEEE, 2017, pp. 1–8.

[8] K. El Madawi, H. Rashed, A. El Sallab, O. Nasr, H. Kamel, and S. Yogamani, “Rgb and lidar fusion based 3d semantic segmentation for autonomous driving,” in 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE, 2019, pp. 7–12.

[9] M. Siam, H. Mahgoub, M. Zahran, S. Yogamani, M. Jagersand, and A. El-Sallab, “Modnet: Motion and appearance based moving object detection network for autonomous driving,” in 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2018, pp. 2859–2864.

[10] V. R. Kumar, S. Milz, C. Witt, M. Simon, K. Amende, J. Petzold, S. Yogamani, and T. Pech, “Monocular fisheye camera depth estimation using sparse lidar supervision,” in 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2018, pp. 2853–2858.

[11] M. Uˇricáˇ ˇr, P. Kˇrížek, G. Sistu, and S. Yogamani, “Soilingnet: Soiling detection on automotive surround-view cameras,” in 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE, 2019, pp. 67–72.

[12] G. Sistu, I. Leang, S. Chennupati, S. Yogamani, C. Hughes, S. Milz, and S. Rawashdeh, “Neurall: Towards a unified visual perception model for automated driving,” in 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE, 2019, pp. 796–803.

[13] S. Yogamani, C. Hughes, J. Horgan, G. Sistu, P. Varley, D. O’Dea, M. Uricár, S. Milz, M. Simon, K. Amende et al., “Woodscape: A multi-task, multi-camera fisheye dataset for autonomous driving,” in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 9308–9318.

[14] S. Milz, G. Arbeiter, C. Witt, B. Abdallah, and S. Yogamani, “Visual slam for automated driving: Exploring the applications of deep learning,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 247–257.

[15] S. M. LaValle, Planning Algorithms. New York, NY, USA: Cambridge University Press, 2006.

[16] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How, “Real-time motion planning with applications to autonomous urban driving,” IEEE Transactions on Control Systems Technology, vol. 17, no. 5, pp. 1105–1118, 2009.

[17] B. Paden, M. Cáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of ˇ motion planning and control techniques for self-driving urban vehicles,” 15 IEEE Transactions on intelligent vehicles, vol. 1, no. 1, pp. 33–55, 2016.

[18] T. M. Mitchell, Machine learning, ser. McGraw-Hill series in computer science. Boston (Mass.), Burr Ridge (Ill.), Dubuque (Iowa): McGrawHill, 1997.

[19] S. J. Russell and P. Norvig, Artificial intelligence: a modern approach (3rd edition). Prentice Hall, 2009.

[20] M. L. Puterman, Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[21] C. J. Watkins and P. Dayan, “Technical note: Q-learning,” Machine Learning, vol. 8, no. 3-4, 1992.

[22] S. Levine, V. Koltun, Continuous inverse optimal control with locally optimal examples, arXiv preprint arXiv:1206.4617 (2012).

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, 2015.

[24] C. J. C. H. Watkins, “Learning from delayed rewards,” Ph.D. dissertation, King’s College, Cambridge, 1989.

[25] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in ICML, 2014.

[26] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," Machine Learning, vol. 8, pp. 229–256, 1992.

[27] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in International Conference on Machine Learning, 2015, pp. 1889–1897.

[28] E. Leurent, Y. Blanco, D. Efimov, and O.-A. Maillard, "A survey of state-action representations for autonomous driving," HAL archives, 2018.

[29] R. S. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," Artificial intelligence, vol. 112, no. 1-2, pp. 181–211, 1999.

[30] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in Proceedings of the 1st Annual Conference on Robot Learning, 2017, pp. 1–16.

[31] S. Kardell and M. Kuosku, "Autonomous vehicle control via deep reinforcement learning," Master's thesis, Chalmers University of Technology, 2017.

[32] J. Chen, B. Yuan, and M. Tomizuka, "Model-free deep reinforcement learning for urban autonomous driving," in 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE, 2019, pp. 2765– 2771.

[33] C.Li, K. Czarnecki, Urban driving with multi-objective deep reinforcement learning. arXiv preprint arXiv:1811.08586(2018).

[34] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," Sensors, vol. 19, no. 3, p. 648, 2019.

[35] M. Cutler, T. J. Walsh, and J. P. How, "Reinforcement learning with multi-fidelity simulators," in 2014 IEEE International Conference on 17 Robotics and Automation (ICRA). IEEE, 2014, pp. 3888–3895.

[36] F. C. German Ros, Vladlen Koltun and A. M. Lopez, "Carla autonomous driving challenge," https://carlachallenge.org/, 2019, [Online; accessed 14-April-2019].

[37] W. G. Najm, J. D. Smith, M. Yanagisawa et al., "Pre-crash scenario typology for crash avoidance research," United States. National Highway Traffic Safety Administration, Tech. Rep., 2007.

[38] Kang, I. Obstacle Avoidance and Autonomous Driving by Embedded Deep Neural Networks. Master's Thesis, Hanyang University, Seoul, Korea, 2020.

[39] Park, S.; Hwang, K.; Park, H.; Choi, Y.; Park, J. Application of CNN for steering control of autonomous vehicle. In Proceeding of the Spring Conference of the Korea Institute of information and Communication Sciences, Yeo-su, Korea, 20–22 May 2018; pp. 468–469.

[40] Pan, X.; You, Y.; Wang, Z.; Lu, G. Virtual to Real Reinforcement Learning for Autonomous Driving. arXiv 2017, arXiv:1704.03952.

[41] Mirchevska, B.; Blum, M.; Louis, L.; Boedecker, J.; Werling, M. Reinforcement learning for autonomous maneuvering in highway scenarios. Workshop Driv. Assist. Syst. Auton. Driv. 2017, 32–41.

[42] Li, D.; Zhao, D.; Zhang, Q.; Chen, Y. Reinforcement learning and deep learning based lateral control for autonomous driving. IEEE Comput. Intell. Mag. 2019, 14, 83–98. [CrossRef]

[43] Yi, H.; Park, E.; Kim, S. Multi-agent Deep Reinforcement Learning for Autonomous Driving. J.-Form. Sci. Comput. 2018, 24, 670–674.