Literature Review of Deep Reinforcement Learning for Autonomous Vehicle Driving

Nicolas Guarini
Department of Computer and Systems Sciences (DSV)
Stockholm University, Stockholm, Sweden
nigu2420@student.su.se – web@nicolasguarini.it

***Abstract*—This review explores Deep Reinforcement Learning (DRL) in autonomous driving, focusing on its role in addressing challenges in decision-making, perception, and control. Key approaches and implementations by companies like Tesla and Google are also analyzed, highlighting DRL's impact and future potential in this field.**

***Index Terms*—reinforcement learning, deep reinforcement learning, self-driving cars, autonomous driving**

I. Introduction

A. Background

Reinforcement Learning (RL) is a machine learning approach which closely mirrors the learning processes of humans and animals: trying and learning from errors, receiving feedback in the form of rewards or penalties [1]. Unlike other forms of machine learning, RL involves an *agent* that observes and *acts* within an *environment*, receiving *rewards* for good decisions, and penalties for bad ones. Unlike supervised learning, which relies on sets of labeled examples, and unsupervised learning, which is typically about finding structure hidden in sets of unlabeled data, reinforcement learning is about maximizing a reward signal. We can therefore consider it to be a third machine learning paradigm, alongside the two just mentioned [1], particulary suited for decision-making tasks. The main elements of a RL system are: a *policy*, which defines the agent's way of behaving, a *reward signal*, which defines the goal of a problem, a *value function*, which specifies what is good in the long run, and a *model* of the environment, which mimics the behavior of the environment [1]. Methods that uses models and planning are called *model-based* methods, while simpler explicitly trial-and-error learners are called *model-free* methods. Although Reinforcement Learning had a lot of success in the past, in particular since the late 1980s [1], those approaches were limited to low-dimensional problems, therefore lacking on scalability, because of complexity issues such as memory, computational, and sample complexity [2]. There are some ther challenges faced in RL, including the need for trial-and-error interactions to infer optimal policies using only reward signals, managing temporally correlated observations, and addressing the credit assignment problem, where the impact of actions is often delayed over time [4]. In more recent years, specifically around 2015, the rise of deep neural networks and deep learning had a huge impact on many areas of machine learning, and specifically in reinforcement learning, providing new tools to overcome some of the problems mentioned earlier, thanks especially to their ability to find compact low-dimensional representations of high-dimensional data [3], enabling RL to scale to problems that were previously intractable, defining the field of Deep Reinforcement Learning. Over the years, DRL has found applications across a wide range of fields, including robotics, where it enables learning control policies from raw sensory data, video games, where it achieves superhuman performance in complex scenarios, driving simulators, which serve as testing grounds for advanced decision-making algorithms [4], and autonomous driving, where it plays a critical role in perception, planning, and control for real-world implementation.

B. Motivation

The relevance of Deep Reinforcement Learning for autonomous driving has never been higher, as autonomous vehicles have increasingly transitioned from theoretical possibilities to widespread, tangible reality in recent years. These vehicles operate in dynamic, uncertain, and unpredictable environments, requiring advanced decision-making, adaptability, and optimization, areas where DRL excels. It is therefore valuable to conduct a detailed review of the most used approaches and technologies in this context.

C. Objectives

The objective of this review is to explore the primary approaches to DRL applied to autonomous driving and examine how leading companies, such as Tesla and Google, have implemented autonomous driving methods. By analyzing real-world implementations and advancements, this review aims to highlight the role of DRL analyzing the main challenges in perception, decision-making, and control, as well as its integration into the development pipelines of industry giants. Additionally, it seeks to identify emerging trends and potential future directions for leveraging DRL in this rapidly evolving field.

II. Methodology

The papers used as references were sourced primarly through the IEEExplore and ResearchGate platforms using the keywords "reinforcement learning", "autonomous driving",

and "self driving", filtering them from the year 2015 onwards and prioritizing papers with more references.

Filtering papers from 2015 onward is justified due to significant advancements and shifts in these fields during that period, including a new wave of research and applications, such as Mnih et al.'s work on the Deep-Q-Network, which marked a turning point in DRL. Moreover, around 2015, hardware capabilities improved significantly and autonomous driving concepts transitioned from experimental projects to real commercial systems, with substantial contributions from major companies like Tesla and Google (Waymo).

From this search, it was possible to identify the algorithms that are analyzed in the subsequent sections.

For the introductions of the specific algorithms, the original papers were used, while the general introduction to Reinforcement Learning is referenced from Reinforcement Learning: An Introduction by Sutton & Barto, which is widely recognized as a foundational text in RL. Therefore, although it was not identified through a systematic literature search, its status as a seminal work in the field justifies its use as a reliable and authoritative reference.

We can therefore divide the references used into three categories:

- *Foundational*: works that cover the theoretical aspects and algorithms of deep reinforcement learning ([1], [2], [3], [4], [8], [9], [10]);
- *Autonomous Driving Applications*: studies on applying RL in autonomous driving systems ([5], [6], [7], [11]);
- *Industry Leaders' Implementations*: studies and reports discussing the practical implementation and safety of large scale autonomous systems ([12], [13]);

III. Reinforcement Learning in Autonomous Vehicles

RL is the most natural way to approach these types of problem, since supervised learning methods do not learn the dynamics of the environment or that of the agent [1], while RL is built for handling sequential decision processes. However, autonomous driving remains a challenging domain for machine learning, since the concept of "correct" driving behavior is only loosely defined, since very often different responses to similar situations are equally acceptable [5].

A. Markov Decision Process

The Autonomous Driving problem is modeled as a Markov Decision Process (MDP), which is a tuple [1]

$$\text{MDP} = (S, A, p, r)$$

where

- S is the *state* space, that is, a set of states;
- A is the *action* space, that is, a set of actions;
- $p : A \times S \times S \to [0, 1]$ is the *state-transition probabilities* function, that is, $p(a, s, s')$ is the probabilty that action a in state s at time t will lead to state s' at time $t + 1$;
- $r : S \times A \times S \to \mathbb{R}$ is the *expected immediate reward* function, that is, $r(s, a, s')$ is the reward received after transitioning from state s to state s' with action a.

Using this model, we can view a policy π as a mapping from the state space S to the action space A [1], and the goal of Reinforcement Learning will be to maximize the expected reward from each state $s \in S$.

MDP is therefore a very flexible and abstract framework that can be applied to many different problems [1].

B. Software Architecture for Autonomous Driving

The most widely adopted modern approach for Autonomous Driving is to separate the various tasks in 5 main high-level modules, with a sort of *divide-et-impera* approach [5]:

- **Sensing**: tasks of collecting raw data from sensor infrastructure such as cameras, radars, LiDARs, ultrasonic sensors;
- **Perception**: includes tasks like lane detection, object (other cars, pedestrians, traffic lights, etc...) detection, which includes the processing of the raw data obtained by the *Sensing* module;
- **Abstracting**: includes localised high definition maps to provide a precise virtual reconstruction of the environment and therefore high-level scene understanding;
- **Planning**: includes tasks for path planning and predicting the future agent's manouvers and trajectories;
- **Control**: orchestrates the high level orders such as steering, acceleration, braking;

The algorithms we will discuss in the following sections operates at the planning / control level.

C. DRL Algorithms for Autonomous Driving

Some of the most widely used and safest algorithms are Deep Q-Networks (DQN), Proximal Policy Optimization (PPO), Trust Region Policy Optimization (TRPO) [6], and Hierarchical Reinforcement Learning (H-RL) [11].

A recent research by Dhinakaran et al. in 2024 [6] showed that DQN obtained an average success rate of 75.2% and a safety score of 8.7, PPO obtained an average success rate of 82.6% and a safety score of 9.5, while TRPO obtained an average success rate of 78.1% and a safety score of 9.2 [6].

D. Deep Q-Networks (DQN)

To understand what DQNs are, we first need to explore Convolutional Neural Networks (CNN), which are neural networks specialized in image processing. At a high level, CNNs can be seen as a way to perform machine learning using images as input. Notable examples include CNN models like AlexNet and GoogLeNet, which became well known for their exceptional performance in image recognition [7]. A DQN is a variation of a CNN, which outputs not classifications, but Q-values (rewards) for actions based on input states. It enables agents to learn policies directly from high-dimensional sensory inputs through end-to-end training, without the need of a huge amount of human driving data, but instead learning from its own behavior [7].

DQN are based on Q-learning, a RL method where agents transitions between states by making actions, receiving rewards or penalties, and updating its knowledge using an update formula:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t - Q(s_t, a_t) + \gamma \underbrace{\max}_{a'} Q(s_{t+1}, a'))$$

where [7]:

- $Q(s_t, a_t)$ is the estimated value of taking the action a_t in the state s_t;
- α is the learning rate, determining how much new information overrides old information;
- r_t is the reward received after performing action a_t in state s_t, representing the immediate feedback on the taken action;
- γ is the discount factor, which balances the importance of immediate and future rewards.

During a training session, the DQN computes the Q-value for a state of the environment using the formula just mentioned, and this state-action cycle is repeated until the agent reaches the goal. If, during the cycle, the agents fail (for example, it hits an obstacle or violates traffic laws), the *episode* is discontinued and the agent starts a new one [7].

In the context of autonomous driving, a typical approach could involve the DQN interpreting the road scene from camera input (potentially combined with various types of sensors) and outputting a specific action. Often, the action space is simplified to include only a finite number of actions (that is, *discrete action space*) (e.g., steer X degrees to the left, steer Y degrees to the right, keep the steering wheel straight, etc.) [7]. The episodes are then executed, providing a reward each time the agent successfully overcomes an obstacle, for example. A good metric to evaluate the learning progress could therefore be the distance successfully covered by the agent at various speeds. Naturally, the faster the car goes, the more effort the agent will exert to cover long distances [7].

E. Trust Region Policy Optimization (TRPO)

TRPO is an algorithm that optimizes control policies (strategies for making decisions in an environment) with guaranteed monotonic improvements [8]. The main idea behind this algorithm is to minimize a *surrogate loss function* with a penalty on the KL divergence[1]. This penalty limits the size of the policy update at each iteration, preventing large performance drops [8]. This combination of surrogate function minimization and KL divergence constraint ensures that each policy update leads to a reduction (or at least maintenance) of the expected cost. In this way, the algorithm progresses monotonically toward an optimal policy [8].

There are two variants of this algorithm [8]:

- *single-path*: it samples individual trajectories, and can be applied in model-free settings.
- *vine*: it builds a set of rollouts, and perform multiple actions from each state. Therefore, it requires the system to be restored to particular states, which is typically only possible in simulations.

The TRPO algorithm repeatedly perform these steps [8]:

- Collect state-action pairs (using *single-path* or *vine* procedures) and estimate their Q-values[2] using Monte Carlo methods;
- Build the estimated objective and constraints by averaging the collected samples;
- Update the policy parameters by solving the constrained optimization problem, using the conjugate gradient method and a line search.

Unlike DQN, which derives the policy implicitly by selecting the actions with the highest Q-values in each state [7], TRPO directly optimizes the policy by searching for the parameters that maximize the expected reward. This makes DQN more suitable for problems with discrete action spaces, while TRPO should perform better in problems with continuous action spaces [8].

The autonomous driving problem can be modeled as either a continuous or discrete action space problem, depending on the chosen level of abstraction. However, the continuous approach is more realistic (but more resource intensive) as it reflects the vehicle's physics, so TRPO is expected to deliver better performance.

F. Proximal Policy Optimization (PPO)

PPO is a newer family of policy gradient methods, which have part of the benefits of TRPO, but they are more general, simpler to implement, and with better complexity [9]. At their core, both PPO and TRPO are policy gradient methods, which means that they learn a policy by iteratively improving it, based on the gradient of a performance measure.

PPO aims to achieve the reliable performance of TRPO while simplifying implementation and improving sample complexity. Instead of using a hard constraint, PPO uses a "clipped" surrogate objective function [9]:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\big[\min(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta),\ 1-\epsilon,\ 1+\epsilon)\hat{A}_t)\big]$$

where:

- $\hat{\mathbb{E}}_t$ represents the empirical average over a batch of sampled trajectories;
- $\hat{A}_t$ is the advantage estimate at time step t, indicating how much better an action a_t is compared to the expected value for state s_t;
- $r_t(\theta)$ is the probability ratio;
- ϵ is a hyperparameter (e.g. $\epsilon = 0.2$);
- $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)$ ensures that the probability ratio r_t stays within the range $[1-\epsilon, 1+\epsilon]$, preventing too large policy updates;

L^{CLIP} penalizes large policy updates by clipping the probability ratio, which quantifies how much the action probabilities

[1] Kullback–Leibler divergence, is a type of statistical distance which measures how much a model probability distribution is different from a true probability distribution.

[2] Q-values represent the expected reward an agent can obtain by taking a specific action in a given state and then following a specific policy.

change between the old and new policies. This dynamic prevents the policy from taking too drastic steps that could destabilize the learning process, forming a lower bound on the original surrogate objective, ensuring that updates are cautious [9].

As an alternative to clipping, PPO can also implement an adaptive KL penalty, which dynamically adjusts the penalty coefficient to keep the KL divergence close to the target value [9].

Some of the main benefits of PPO are [9]:

- PPO relies on first-order optimization methods like stochastic gradient descent, which are simpler to implement than for example second-order methods like conjugate gradient, used in TRPO;
- PPO can be used in with architectures that include noise or parameter sharing between the policy and value function, unlike TRPO which is incompatible with these features;
- Experiments have shown that PPO generally outperforms other policy gradient methods like A2C[3] and TRPO, on both continuous and discrete tasks.

G. Hierarchical RL

Hierarchical Reinforcement Learning is a different approach from the previous we have discussed, which aims to address the complexity of decision making by decomposing a task into a hierarchy of subtasks [10]. This approach introduces the idea that an agent can learn not only low-level actions but also higher-level policies that guide the behavior over longer time horizons. The agent will therefore choose between different *options* (which are themselves learned policies), and whithin each one of them, it performs a set of actions that leads to the completion of the sub-task [10].

This dynamic allows the agent to build *reusable knowledge*, improving learning speed and generalization and enabling, in general, more efficient learning by breaking down complex tasks into more manageable components [10].

The general structure of this approach is a hierarchy of two levels [10]:

- *Scheduler* π_θ (higher-level policy): trained to maximize the environment reward;
- *Worker* π_ϕ (lower-level policy): trained to efficiently discover optiions that will be used by the *scheduler* π_θ to accomplish the tasks.

A common H-RL framework for self-driving is the following:

- *High-Level Maneuver Selection* (*Scheduler*): a master policy chooses the maneuver to be executed in the current state;
- *Low-Level Motion Control* (*Worker*): the corresponding maneuver policy would be activated and output the specific commands (for example, whell angle, accelerator, brakes, ...) to the actuators of the car.

[3]Advantage Actor Critic method

Combining multiple maneuvers can constitute various driving tasks, allowing the trained maneuver policy to be applied to different tasks. Therefore, this hierarchical approach demonstrate superior transferability compared to more classical (end-to-end) RL approaches in driving decision-making [11].

IV. State-of-the-Art Technologies in Industry Leaders

This section focuses on the case study of Waymo. While it is not the only company operating in the self-driving car industry, Waymo stands out among major players for publishing detailed papers on the technologies used and the rationale behind its implementation choices.

A. Waymo

Waymo is an autonomous driving technology company under Alphabet Inc., specializing in developing self-driving cars and ride-hailing services using advanced AI and sensor systems. One of the main solutions that Waymo found to improve the safety and reliability of autonomous driving policies was to combine Imitation Learning (IL) with Reinforcement Learning [12]. Specifically, a policy was trained on over 160,000 km of urban driving data, and performance efficiency was measured in various test scenarios grouped into different levels of collision probability. It was found that, while Imitation Learning alone performed quite well in low-difficulty scenarios, combining it with Reinforcement Learning led to a 38% reduction in failures [12].

The combination of these two techniques is very interesting and is the first large-scale implementation. One initial challenge is managing the numerous edge cases that occur in everyday driving, and for this, Imitation Learning is great and scales well with real driving data. However, there are still many situations that occur rarely in the data, causing the imitation policy to respond unpredictably. This is where Reinforcement Learning comes into play, as its operation based on reward functions explicitly determines what constitutes a safe or unsafe response. RL, operating in a closed-loop system that establishes clear cause-effect relationships between observation, actions, and outcomes, results in policies that are less vulnerable to covariate shifts and spurious correlations typical of IL, and more aware of safety considerations explicitly encoded in reward functions [12].

On the other hand, relying solely on RL is also problematic because it is highly dependent on reward design, which remains an open challenge in autonomous driving [12].

IL and RL, in this context, are thus extremely complementary: IL enhances realism and adherence to driving conventions, while RL improves safety and robustness, especially in very rare cases that are difficult to manage due to the lack of adequate amount of data [12].

This method proposed by Waymo, is called BC-SAC, due to the two combined RL and IL algorithms:

- *Behavior Cloning (BC)* (Imitation Learning), where the agent learns to mimic the behavior of an expert by directly mapping observations to actions using supervised

learning. The agent is trained on a dataset of expert demonstrations to replicate their decisions in similar situations [12];

- *Soft Actor-Critic (SAC)* (Reinforcement Learning): evolution of DQN, where instead of using a deterministic policy based on Q-values, a stochastic policy is employed and entropy is optimized to encourage exploration and improve stability [12].

Waymo's autonomous vehicles utilize a sensor suite combining LiDAR, cameras, and radar for 360-degree environmental perception, capable of detecting objects up to 300 meters away. The perception system integrates sensor data to identify and track objects, while the planning module leverages machine learning to execute safe maneuvers [13]. Redundant systems in computing, braking, and steering ensure safety in case of component failure [13].

V. Conclusion

This review has examined the role of Deep Reinforcement Learning (DRL) in addressing the challenges of autonomous driving, focusing on key areas such as decision-making, perception, and control. By exploring state-of-the-art algorithms like DQN, PPO, and TRPO, and analyzing industrial implementations from companies like Waymo, the study highlighted both the capabilities and limitations of DRL.

Furthermore, the integration of DRL with complementary approaches, such as Imitation Learning, has demonstrated potential for improving system robustness and handling edge cases.

As autonomous driving continues to evolve, DRL stands out as a critical technology for enabling adaptive, safe, and efficient driving systems, paving the way for further advancements in this dynamic field

References

[1] Richard S. Sutton and Andrew G. Barto. "Reinforcement Learning: An Introduction". Second Edition. Cambridge, MA: The MIT Press, 2018.

[2] A.L. Strehl, L. Li, E. Wiewiora, J. Langford and M.L. Littman, "PAC model-free reinforcement learning", Proc. Int. Conf. Machine Learning, pp. 881-888, 2006.

[3] Y. Bengio, A. Courville and P. Vincent, "Representation learning: a review and new perspectives", IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798-1828, 2013.

[4] K. Arulkumaran, M. P. Deisenroth, M. Brundage and A. A. Bharath, "Deep Reinforcement Learning: A Brief Survey," in IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26-38, Nov. 2017.

[5] Talpaert, Victor & Sobh, Ibrahim & Kiran, Bangalore & Mannion, Patrick & Yogamani, Senthil & Sallab, Ahmad & Perez, Patrick. (2019). Exploring Applications of Deep Reinforcement Learning for Real-world Autonomous Driving Systems. 564-572.

[6] M. Dhinakaran, R. T. Rajasekaran, V. Balaji, V. Aarthi and S. Ambika, "Advanced Deep Reinforcement Learning Strategies for Enhanced Autonomous Vehicle Navigation Systems," 2024 2nd International Conference on Computer, Communication and Control (IC4), Indore, India, 2024.

[7] T. Okuyama, T. Gonsalves and J. Upadhay, "Autonomous Driving System based on Deep Q Learnig," 2018 International Conference on Intelligent Autonomous Systems (ICoIAS), Singapore, 2018.

[8] Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust Region Policy Optimization. In Proceedings of the 32nd International Conference on Machine Learning (pp. 1889–1897). Lille, France: PMLR.

[9] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv:1707.06347 [cs.LG].

[10] Zhang, Yu, Xu. Hierarchical Reinforcement Learning By Discovering Intrinsic Options. (2021). University of Southern California.

[11] Duan, Jingliang, et al. "Hierarchical Reinforcement Learning for Self-driving Decision-making without Reliance on Labelled Driving Data." IET Intelligent Transport Systems, vol. 14, no. 5, Feb. 2020, pp. 297–305. Portico, Crossref.

[12] Lu, Yiren, et al. "Imitation Is Not Enough: Robustifying Imitation with Reinforcement Learning for Challenging Driving Scenarios." 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2023, pp. 7553–60.

[13] Waymo. "Waymo Safety Report", February 2021.