Application of Reinforcement Learning for Autonomous Driving

State of the Art Approaches — Safety Guarantees — Future Research Directions

Igor Sadalski

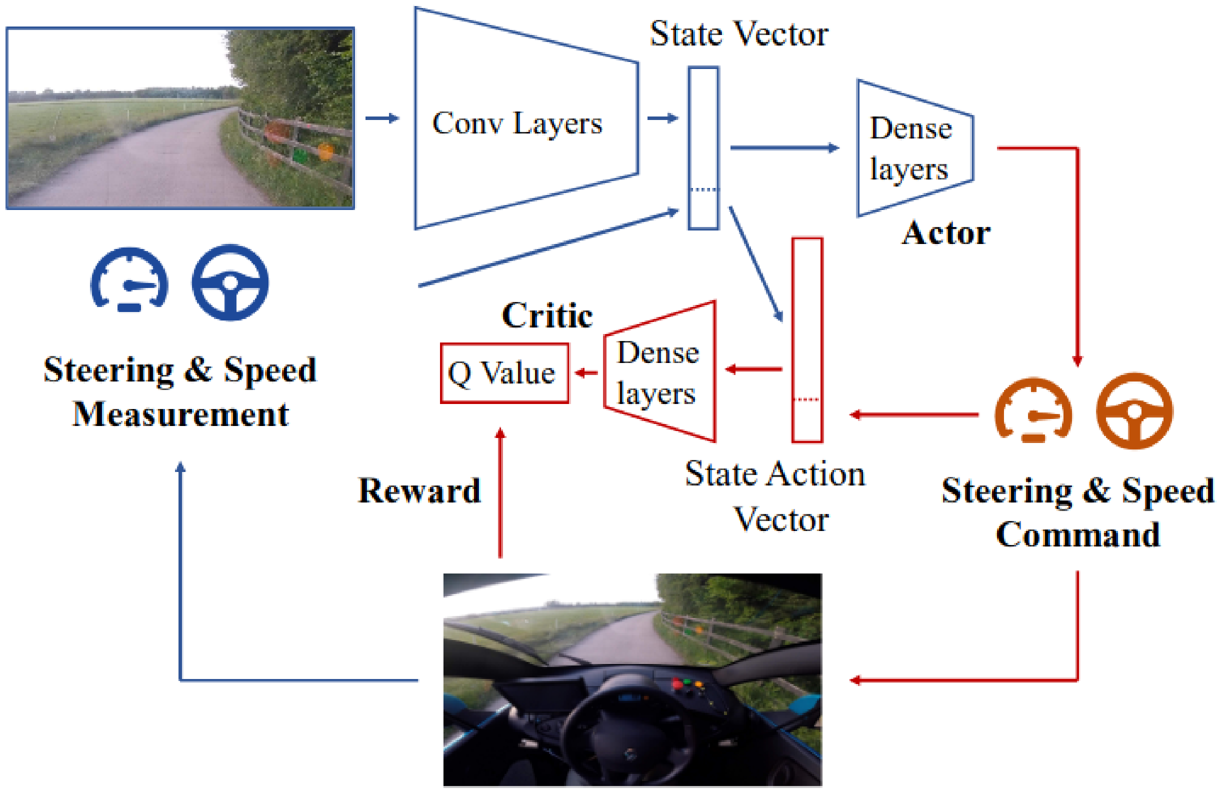


Figure 1: Example of an reinforcement learning architecture that utilizes deep deterministic policy gradient and actor-critic methods. It learns how to drive on a real car in under 20 minutes without any prior data or experience.[1].

Abstract

Autonomous driving is an important and in large part still unsolved problem. Current solution developed by the biggest industrial companies focus on modular approaches, where various algorithms are responsible for very specific tasks and must be combined together for the system to work. Data on accidents shows that such solution have several shortcoming, most notably high effort in designing such techniques and their inefficiencies. One way to overcome it is to use reinforcement learning (RL), which has shown great potential in recent years with successful demonstrations in various domain. Using RL for autonomous driving (AV) application is still relatively new but we can distinguish three main approaches: value-based, policy-based and actor-critic methods. The last based on our review performs best in the real-world, with Deep Deterministic Policy Gradient showing the most advantages. While learning-based systems generalize well when previous dynamics of the model are imperfect their major limitation is the lack of the safety guarantees that can be hardcoded into the system. There are many different solution in use today that aim to address this problem. Most successful ones revolve around encouraging safety in RL system by conducting a safe exploration during the machine learning phase. Nonetheless, the field still lacks valid AVs examples of how such systems would behave in real world. Apart from this, there are various efforts in improving the reinforcement learning in in autonomous driving such as human-computer interactions or sample efficiency that can drastically improve its performance. All in all, however, guided by current experimental results of the field, exclusive use of reinforcement learning is not mature or safe enough for any form of large scale deployment on real hardware due to possible risks it posses for its users. Lastly, future researchers should prioritize safety and simulation to reality transfer in their studies to address this.

List of Objectives:

- Analyse the data of the current approaches for control of autonomous cars and comment on their major limitations.
- Explain what is reinforcement learning and provide broad overview of different strategies of how it can be applied for control of AVs.
- Perform a detialed review of actual real life implementations of reinforcement learning based solutions for autonomous cars commenting on their limitations.
- Group and analyse specific techniques used to embed safety guarantees into the learning controllers.
- Identify and comment on the emerging research trends within the reinforcement learning applied for autonomous driving.

Contents

1 Introduction

Driving is a ubiquitous part of our life from logistic that power our economies to the mobility it offers for everyone. However, human driving causes loss of lives and time every year. Only last year approximately, 1.34 million people world wide lost their life in car accidents [2]. From another standpoint, each year humans spend countless hours behind the wheel, even up to 4 years of life in UK. Whether, these are workers communing to work, families traveling for vacations or lorry drivers delivering crucial supplies to local shops, such amount of time is staggering just considering the opportunity cost of using it for more creative and useful means. Lastly, inefficient human driving in cities causes excess carbon footprint and dangerous pollution resulting in respiratory diseases troubling many.

Loss of human lives and time due to manual driving have prompted a valid questions if developing autonomous cars that would overcome theser problems possible. Promise of autonomous systems that could carry human and various good around in a completely automated manner has captivated imagination of various investors and governments. In US alone, last year it is estimated that billions of dollars has been spend on autonomous vehicles development. However, despite claims by various automakers such as Tesla that this problem can by solve by the end of this decade, recent data shows that it is exaggerated [3]. We note that as of now AVs constitute only minuscule part of our driving ecosystem and this trends, given various hard challenges encountered in the field, seems nowhere to be reversed in the future.

Such finding are both appalling and disheartening given how significant benefits automation of this filed could bring for our societies. Finding ways how we could overcome the current stall in progress is a central focus on this review, however, instead of focusing on means for improving current techniques we investigate an emerging and completely revolutionary paradigm of solving these problems - reinforcement learning or RL for short. Upon reading this review, we aim at leaving the readers with a broad, yet grounded, understanding of: what reinforcement learning is it, how it compares to present techniques, how it is applied for AVs and what are the future research in this field. To reflect our goals the work is divided into three main section roughly revolving around the themes of past, present and future in AVs research (with the section on safety being somewhere in-between current and future efforts). In developing a safer and better future where autonomous cars are a everyday commodity serving humanity we hope to contribute this work as a one step towards it.

2 Modular autonomous driving approaches

The field of autonomous driving really took off in 2004, when US Defense agency DARPA held "The DARPA Grand Challenge" competition. First-of-its-kind contest to autonomously navigate a 142-mile course across a desert. Three years later in 2007 it held a similar event but this time conducted in urban settings and being the first time autonomous vehicles interacted with other urban participants. Positive results in both of the competition have spared an interest into the field. Yet, to begin with we must establish what does autonomous driving even mean. For this consider a concept of autonomy levels. According to SAE, international organization responsible for engineering standards, its J3016 automated-driving graphic lists main level of automation. In it an automated driving starts at level 3 denoted as conditional automation, where "human driver will response appropriately to a request to intervene". The levels go up to 5 which is considered a full automation. Obviously, claim by car-makers should be taken with pinch of salt. Nonetheless, currently some of the cars such as Mercedes-Benz EQS and S-Class claim to have attained a level

3 automated with Alphabet Waymo car going as far as stating it has achieved a level 4 automation (these have however a speed limit of 30 mph and can operate only in specific areas mainly in San Francisco).

2.1 Elements of modular autonomous car

For the most part nowadays the private sector is driving the change in the autonomous vehicles space. Report filed by companies such us Waymo or Argo AI, one of the best funded and best staffed AV companies, offer insight into how these car operate.

From a control perspective cars are a low degree of freedom, nonlinear system with high mass and inertia and at high speeds slowed response. Furthermore, they are safety-critical applications, operating in unknown and unstructured environments with other unpredictable human agents. Lastly, they can get information about their partially-observable environments only through sensor fusion often with uncertainty that is hard to estimate. All this makes them a unique and challenging problem from controls perspective.

Typically the environment around the car is recreated using a sensor fusion of different detectors such as: LIDAR, IMU, cameras, radars or microphones that are fed into the onboard computing module. This raw data is used to recreate the environment, happening sequentially in four main steps[4]. The first step is to build the detailed three-dimensional map with road features detecting. Then comes the detection of main dynamic objects around the car (such as pedestrians, cyclist or vehicles). For each of these objects, car needs to predict how all they are going to move (step 3) based on which it finally asses what control actions it should take (step 4). Each of these steps have in turn many subtasks that need to work and communicate seamlessly between each other. To prove the point, up to date many different algorithms, such as Simultaneous Localisation and Mapping (SLAM) or computer vision with ability to detect and correctly classify different objects based on camera and LIDAR, have been developed. These will not be the covered in this review as we are more concerned with the actual control of the vehicle, however, it is important to that all these steps add to the complexity of a modern autonomous car and offer possible points of failures in such complicated systems.

2.2 Rate of success of current techniques

Autonomous driving is still a relatively young technology and as so there is very scare data on how it actually performs in the real world. California and in particular San Francisco area in US is one of the only places worldwide where such data has been systematically gathered over the last years. With leading autonomous car makers, favourable legislation it acts as a trend setter and a valuable insight in the current status of this space. Here we extensively review three reports [3], [5], [6] based on the California Department of Motor Vehicles, main body overseeing the traffic accidents, in order to asses how mature are current techniques.

Out of 7 main companies operating AVs in California over the span of 2014 to 2017 analysed AVs crashes where on the rise in a linear fashion and in 2017 only slightly below 30 where recorded [3]. By far the most accidents happened on the intersection (71 %), most likely due to a high number of human agents and complexity in their maneuvers. Fortunately in only 3% of these crashes autonomous cars were going above 20 mph, thus, significantly decreasing any potential damage. Furthermore, rate of accidents when a pedestrian or vehicle was present lower by 10% for AV compared to normal traffic collisions and below a regular human driver. These finding where supported by other work [6], where a similar data set of Level 3 AVs in California over the years 2015 to 2020 was used. However, there, authors observed that in terms of collision rate per every mile driven, autonomous car drastically under performed compared with humans. Based on these finding researchers claimed that "self-driving AV prototype systems significantly overstates their technological capability to operate on roads as conventional vehicles" [6].

Such harsh opinion seems aligned with another works commenting on the same data set, where they stated that "challenges of fully autonomous vehicles [...] are significant and underestimated" [5]. In that work, authors further note that from 144 tested AV that have driven culminate 1,116,605 miles there where 42 accidents. In such light, again, these cars are 15 to 4000 times worse than human drivers which have significantly lower collision rate per miles driver. They list a number of possible flaws in the modular control structure of modern autonomous vehicles; some stemming from hardware malfunction, such as network failures, sensor malfunction, data corruption or incorrect actuators behaviour. Others come from the control architecture itself. While the first group lies beyond the scope of this review, across car manufactures examined, it has a smaller impact compared to on average 64% rate of errors from planner/controller or perception/recognition software [5]. This highlight some shortcoming in the current methods.

2.3 Major limitation and challenges?

To analyse failures in modern AV systems authors in [7] proposed a detailed set of possible errors in each part of a state of the art autonomous software stack (such as perception detection and localization, planning, control). They note that error that cause crash during their experiments on a full-scale autonomous racing car, where due to the wrong speed profile calculation, infeasible trajectory output and to big lateral error in path matching. Once again, all these partially come from the complexity and number of interconnected systems in the AV technological stack.

To sum up, though data is incomplete the overall picture is that the current AVs already contribute to traffic accidents, by some metrics being more error prone compared to a regular human driver. What is more, nearly two out of three most of these crashes, when analysed in detail, happen due to the errors induced by the current techniques. All this prompts a question, whether it is possible to come up with a different control and navigation algorithm that would account for the before mentioned shortcomings.

3 Application of Reinforcement Learning for control of autonomus cars

One proposed technique to address the above limitations is use of an algorithm called reinforcement learning, also called RL for short. In a nutshell, it iterative learns a set of best actions for any given situation based on the interactions with its environments and historical data. Since the human-constructed constrains imposed on this learning algorithms are often minimal, given enough time and proper structure, *in theory it could have a highly optimal response for any situation and any system.*

Above statement has been supported with various experimental work. Researchers have shown that RL is capable of learning by itself how to play and beat human-experts at game of Go. Game that proved incredibly challenging for different techniques owing to the enormous search space and a difficult to evaluate board position and moves [8]. In another landmark study robotic researchers developed an end-to-end full locomotion policy deployed on a physical robot [9] . What was remarkable was that apart from robustness, ability to withstanding external disturbance without compromising stability, the raw sensor data was directly fed into controller generating torque commands without any intermediate steps. Additionally, the whole system was trained it under

20 minutes on a typical GPU, remarkably short time for such achievement. And in yet another work researchers successfully learned control policies from sensory input from camera playing seven Atari 2600 games (such as Pong, Space Invaders, etc.), in some outperforming human expert [10].

All these where achieved without any human-expert hard coding some game strategies or fine-tuning the parameters. Crucially, the system was able to learn purely by its interaction with the virtual environment how to choose end-to-end optimal policy without human-expert in the loop, while still outperforming one. From a broader prespective, replacing error-prone, rule-based modular approaches with single bloack in a similar fashion to [11] could pose a significant breakthrough for AVs. Naturally, question arises about the detailed workings, feasibility and limitations or possible improvement of current systems and these questions will be the main focus of this review.

3.1 Introduction to reinforcement learning.

Over the year various approaches, from optimal control and behavioral psychology, aiming at te have converged to a single field called reinforcement learning. In the community one definition of reinforcement learning is that it is a machine learning algorithm where the agents learns by receiving rewards and penalties for taking specific actions. Thus, it is an active, sequential, goal-oriented approach where we try to optimize some reward. In words of Hado van Hasselt, a renowned researcher in the space, it is a ”science and framework of learning to make decision from interactions”.

The simplest and most general structure of a reinforcement learning algorithm has been schematically shown at figure 2. It consist of agent receiving some state and reward from the environment and a policy based on which it takes new action. The action of the agent are then fed back to the environment which generates a new state and a new reward based on them. Imagine that you want to cast the problem of driving to work as an reinforcement learning. In this example, you would be the agent, making decisions such as when to turn a steering wheel, push the throttle or brakes. These actions would be made in response to you see on the road and what you read on the car controls, the states and most likely they would be guided by some unconscious policy (e.g. I am going to quickly, maybe I should slow down). Your environment would be the actual real-world and after each decision you would probably reflect on well you performed a given maneuver. Ideally you would like to perform actions that would maximize some of your goal, say most energy efficient driving, not braking the traffic rules or alternatively going as fast as possible. In such case you would rewarded with some innate satisfaction if you fulfilled your goals.

The following has been loosely inspired by RL lectures conducted at University College London by Deep Mind Researchers and comprehensive introduction to reinforcement learnig by Sutton et al. [12].

State, reward and action: Our previous simple example can be further formalized. First, we define vector or high dimensional scalars of actions A_t, states, S_t and rewards, R_t at teach time step t. In most simple case, our environment will be a high dimensional function that takes as input action and maps them to a state and reward at a next time step, figure 2. Similarly, agent is another function that can map state and reward to an action. In some form, our agent will have a memory of all previous state-action pars and corresponding rewards. Importantly, reinforcement learning (most often) is defined as a solution method to a general Markov Decision Process (MDP), a type of mathematical framework used for modeling transition between discrete events used for control and decision science [13]. With it we can assume that ”any goal can be formalized as the outcomes of maximizing a cumulative reward” (van Hasselt) and that any transition between the

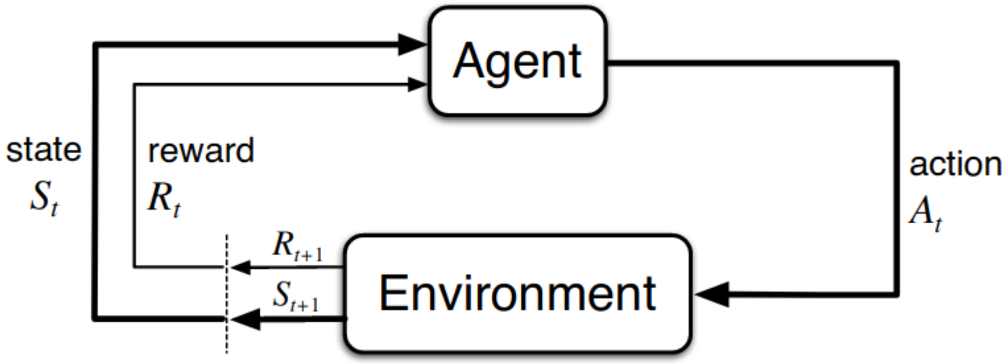


Figure 2: Schematics of how RL works from [12].

states are either probabilistic or stochastic.

Agent, policy and value function:

Our states are then feed in into the agent and policy. The difference between the agent and its policy is that, while both of these are mapping function from state and rewards into the actions, the policy can be though of as a set of instruction on how to make this transitions. Then agent is everything around it, i.e. updating the policy, storing past data, pre-processing the state, etc. Agent is in fact an advanced architecture where policy, a decision making component, lives. In a similar fashion the previously introduced value function is also a sub-component of the agent giving it an insight into rewards obtainable from different states. In specific agent can choose to select its actions based on policy (e.g. policy-based methods), values or both (actor-critic methods). Furthermore, storing past data it can use it to try to devise some model of the objects it is representing for quicker computations in the future (model-based), e.g. deriving the dynamical equation describing a complex robot that are purely approximated based on the interactions, or just use some advanced patter recognition to obtain the optimal states from given actions (model-free). All these are useful insights explored in developing the agent architecture itself, as an example see figure 3. Nonetheless, these all efforts aim at finding a set of commands that will maximize the sum of the expected future rewards given some model dynamics and in the process we try to develop a reinforcement learning agent architecture so as to maximize it.

3.2 Supplementary concepts

In this subsection we focus on more specialized topic around RL, of the likes of deep learning and deep reinforcement learning or convolutional neural networks. These are ubiquitously used in the field, either as agent or policy function approximations or input state, and understating them soundly is important.

3.2.1 What is Deep Learning (DL)?

To begin with one of the most useful concepts in reinforcement learning is deep learning, a techniques used to extract patters and underlying features directly from data. The following description has been adopted from MIT 6.S191 ”Introduction to Deep Learning” lecture series by Alexander Amini.

To begin with the fundamental building blocks of any deep neural network is a neuron. Such neuron consists of various inputs with corresponding weights, a sum, a non-linear activation function (e.g. sigmoid function, hyperbolic tangent, rectified linear unit or ReLU, thanks to these we can model non-linearities in our networks) and bias terms that shifts the input to activation function and output prediction. Equation 1 presents the output of a neuron in a summation and matrix form. It is based on bias w_0, inputs x_i or X in a vector form, weights w_i or W, non-linear function $g(z)$ with an output y. Stacking many neurons together and using their outputs as inputs to next ones we can start to create a layered network. Connecting every input to every output we would create a dense layer neural network, however, this is not always necessary.

$$y = g(w_o + \sum_{i=1}^{m} x_i w_i) = g(w_o \mathbf{X}^T \mathbf{W}) \tag{1}$$

Another crucial concepts for deep learning is the process of training the network for which we first need to define: a loss function defined as $L(f(x^{(i)}; \mathbf{W}, y^(i)))$ which can be represented in different forms depending on the tasks (such as: mean squared error or cross entropy loss). This function compares predicted value generated by the network with an actual value it should output and tells the network whether it has made a right or wrong prediction. For instance, when training our network we want to minimize our loss function, as it will mean that our network well captures the underlying behaviour. We can do this by adjusting our weights. For each combination of weights we select we can determine the corresponding output of the loss function. As we want to minimize the loss we can aim to employ techniques searching for such optimal weights. There are different optimization techniques, of the likes of a stochastic gradient descent, that we can use in this process. In general, the process of computing the gradients to improve the network is called backpropagation and for more layered networks we use these gradients and a chain rule to propagate back to our input neuron. Schematically find the optimal weight, $\mathbf{W}^*$, was presented as an optimization problem in equation 2.

$$\mathbf{W}^* = \text{argmin}\frac{1}{n}\sum_{i=1}^{n} L(f(x^{(i)}; \mathbf{W}, y^{(i)}) \tag{2}$$

3.2.2 What is deep reinforcement learning (DRL)?

Concept of deep neural networks acting as function approximations can be used for representation of reinforcement learning components such as: policy, value function, model or agent. This is especially useful if we are working with high-dimensional states (e.g. camera frames) or we need to propagate our system far into the future gathering a lot of data. In such instances, without deep learning the memory and computing requirement might be simply to great for any implementation. Another fundamental concept in deep reinforcement learning or DRL, is that of Q-function 3, which quantifies the expected total future reward at a specific state and action.

$$Q(s_t, a_t) = \mathbb{E}[R_t | s_t, a_t] \tag{3}$$

Knowing Q-function if very useful as we can evaluate different action, however, finding it is not simple. The general approaches to structuring such DNN is to feed the state as its input and output Q-value for each possible action. Generally, to train such Q-function we define a Q-loss function, equation 4 based on which our value function is refined.

$$L = \mathbb{E}[||(r + \gamma max Q(s', a')) - Q(s, a)||^2] \tag{4}$$

Such method describe steps used to devise the DQN one of the most popular RL methods currently used variation of which we cover more in depth later.

3.2.3 What are Convolutional Neural Networks (CNNs)?

For most of the computer vision analysy algorithms computer can see any picture as a big matrix, three layers deep representing RGB scale of pixels. On a high level, to classify a certain object various algorithms can perform a feature identification. Manual features extraction with domain knowledge expert like human is very impractical due to great manual effort required. Such approaches also introduce various defects like: viewpoint variation, illuminations condition, scale variation, deformation, occlusion or background clutter that can a render. To overcome this we use a can variation of deep learning that both preserves the spatial structure of pixels in our output and extract various useful features. In specific, instead of using dense layers, we can use overlapping connections that link the next hidden layer to only portions of the input image. With only few convolutional layers we can fist extract, or filter, local features and then, using another layer, combine them into even bigger distinct pattern. The convolution operation can be incorporated into the neural network which is called CNN.

For the structure itself of the CNN most often we have input followed by repeating blocks of convolution layers, non-linearity activation's and pooling operation reducing the dimensionality). This process is called feature learning, and its main aim is to separate noise in picture from important features. This is then followed by a regular dense fully connected classifier with several layers whose outputs are what we actually use. The feature learning can be followed by many different architectures like object detection, segmentation or even probabilistic control by using deep reinforcement learning after the initial CNN. (The following has been loosely also adopted from MIT 6.S191 ”Introduction to Deep Learning” lecture series).

3.3 Different strategies for solving reinforcement learning

Generally speaking, reinforcement learning is an umbrella terms for different algorithms aimed at solving problems formulated as Markov Decision Processes. Initially defined mainly for systems with discrete action space [14] and with more simple function approximations, last years have seen an increase in more sophisticated methods [9], [15]. In particular, the major advancements, especially with application for autonomous driving, have been in the formulation of strategies used for solving RL problems. Various such algorithms, alongside examples of scientific works trying to implement them, have been analysed in literature reivews [16]–[18]. These span tens of examples and methods [16]. Here only the three most relevant categories (i.e. value-based, policy-based and Actor-Critic methods) will be analysed each with few examples of specialized state-of-the-art approaches[11], [19]–[23]. Notably, all selected works are focused on autonomous driving applications.

As for the selection for the main categories, these broadly cover the whole space of current solutions to creating an reinforcement learning agent. Starting from the first one, value-based methods are of of the most popular RL solvers [14], [18]. They aim at finding the value-function and then compose a policy which simply chooses the action that maximizes the future expected return [15], [19]. On the other hand, as opposed to value-learning based, it is possible to cut straight to policy-optimization, without intermediate step of evaluating the value-function. This is possible by performing various actions according to some fixed policy and then observing what rewards they bring. Next we can simply fine-tun the parameters in policy to account for this [22], [23].

Lastly, a recent drive to combining the advantages of the two previous approaches resulted in new category of actor-critic methods. These hybrid methods follow a logic that one of the best methods to improve is to get a critic, or a better agent, to review you actions and comment on them. In the case of autonomous driving consider a case where a critic points out to the actor that from a specific state the system was irrecoverable. In such situation all the action and experience gained from this point onward would be irrelevant and actor could focus on action that happened before this event, as they led to an incorrect state. This can be implemented with policy function being the actor and value-function acting as a critic, which feedback guides the learning process [11], [20], [21], [24], [25].

For the three above categories we selected few examples of papers implementing them, summarized in table 1. The key metrics for that table emphasise the advantages of these different solving strategies. As we are still far from a general RL-solving strategy, with presented techniques heavily tailored to the tasks, we try to analyse them in a context of the goals authors are trying to achieve. Thus, each algorithm is compared against its advantages, unique elements employed during implementation, motivation of the authors, carried tasks and well experiments and with their results. We also provide acronyms for easier navigation in the passages. Building from this, in the following sections we review how these strategies work in detail, what are their limitations and how they could be possibly improved. While algorithms are presented in table 1 we also present their varying architectures in figure 3.

	DQN		**Actor-Critic**			**Policy-optimization**	
author	Zhang et al. (2018) [15]	Yurtsever et al. (2020) [19]	Jaritz et al. (2020) [11]	Chen et al. (2019) [20]	Zhu et al. (2019) [21]	Ye et al. (2020) [22]	Chen et al. (2020) [23]
solving strategy	**Double deep Q-Learning**	**Hybrid DQN**	**Asynchronous Actor Critic**	**Soft Actor Critic and DDQN**	**Deterministic Policy Gradient**	**safe PPO based DRL**	**Monte Carlo Tree Search**
advantages	reduce DQNs unstabilities; more stable and realiable learning;	model based path planners introduce higher safety in the system	convergence is faster; parallel learning	better sample complexity and convergence; less parameter tunning	good performace for continuous control,	combines the policy with a safety intervention module (safer DRL)	predicting maneuvers to improve the stability
unique elements	discrete action space; hierarchical reward	adding reward for generating close results to the path planner	discrete control; reward computed at each frame not at the end	latent encoding that can decrese the sample complexity	experience buffer used to avoid learning from correlated data	considering safety efficiency and comfort in reward reward function;	iterative searching process; tree policy structure
motivation	direct perception approach	integrating DRL with path planners	experience decoleration	comparing three DRL techniques	safe, comfortable, efficient system	safe decision making system	predicting driving states
task	control vehicle speed in AV	path following to a goal in urban setting with camera	end-to-end vehicle control from camera	end-to-end navigation of an urban intersection	velocity control for the car following	motion planing for lane-changing on a highway	navigating a racetrack with obstacles
experiments	naturalistic driving data for enviroment	CARLA simulation with A* algorithm for path generation	simulation with realistic physics and lighting	simulated roundabout with other cars	comparison to real highway data	simulation with real world highway segment	3 different tracks in USS simulator; random obstacles
results	271.73% quicker than deep Q-learning	3 times quicker convergence and 80% higher reward than DQN	full control in simulation and generalisation to real world	0% DDQN agents reach the goal point and only 58% SAC doesones	time to collision >5s was 8% for DDPG, 27% less than for human	95%-99% success rate in traffic, on average 0.5% collision rate	57% and 12% more stable control c.f. to DQN and DDPG
acronym	**DDQN**	**HDQN**	**A3C**	**SAC**	**DDPG**	**PPO**	**MCTS**

Table 1: Based on broad review of examples where RL was used for control of AV [17], [18], [26], [27].

3.3.1 Double Deep Q-Learning (DDQN)

The first detailed technique we are going to review is Double Deep Q-Learning. It is an improved DQN technique which uses two deep neural networks. Such structure helps account for a problematic bias in overestimation of learned values by having two separate Q-value estimators [28]. The first network is used to update the other one, which results in more unbiased Q-values. Accounting for instabilities of DQN, double deep Q-learning shows a better performance for tasks like autonomous driving [15].

3.3.2 Hybrid DQN (HDQN)

Different approach, also building from DQN, is a hybrid DQN method focusing on integrating a model-based solver, like Model Predicitive Control (MPC, an advanced model-based control technique based on optimization) with reinforcement learning. Most notably, such approaches enable safety constraints to be hard-coded in the system they also achieve more robust driving experience and faster learning process [19]. Given a reliable path planer, one method include defining reward for a RL controller close to results generated by MPC [19]. DQN are still used as approximation and in specific we have two fully connected layers that output a set of discrete driving actions based on maximizing our future expected reward sum - Q.

3.3.3 Asynchronous Actor Critic (A3C)

Another methods that uses discrete action is Asynchronous Actor Critic or A3C for short. It works by introducing and ”advantage” which is an algorithm reflection on whether a set of selected action was better or worse than when it was evaluated back in the past. Most notably such approach allows for correction for non-optimal strategies [11]. A3C also outputs continuous probabilities for different commands actions so it can be sampled more smoothly. The method has shown good performance for experience decorrelation, i.e. better learning from different episodes like turn or sharp curves[11], which is crucial for sparse or not systematic data.

3.3.4 Soft Actor Critic (SAC)

Soft actor critic is a different actor-critic approach that focuses the agent to maximize both the rewards and property called entropy and so it optimizes the modified reward function. On high level, algorithm solves a similar problem as in classical RL definition, defining a soft value function and then choosing the value that maximizes Q-function. It also outputs a stochastic neural policy (meaning that parameters of the policy dictate the sampling probability of actions) from which later some actions are selected. Compared to Proximal Policy Optimization or previous asynchronous actor critic this method does not require new samples to be collected at each gradient step and it has both a better convergence and a lower sample complexity (hence being less sensitive to hyper-parameters tuning) [20], [24].

3.3.5 Deep Deterministic Policy Gradient (DDPG)

The last analysed actor-critic method, deep deterministic policy gradient, is a form of an off-policy actor-critic. In it both actor and critic are represented as a neural network (e.g. with input one 30 unit hidden layer and an output [21]). Actor, based on a state sends an action estimate to the critic, based on which he evaluates a gradient of Q-function across all different actions. This is then fed into the policy gradient which estimates the update of the actor. It has been shown that such approach has a good experience when it comes to a continuous control, which is a big advantage over SAC or A3C outputting to discrete actions, thus, making the control more smooth and accurate.

3.3.6 Safe Proximal Policy Optimization (PPO)

Safe proximal policy optimization as presented in [22], combines the policy with the safe intervention module. PPO works by adding a soft constraints that can be optimized by a first order approximations which is combined with line search and trust regions that guide the optimizer to search for more favourable solutions[25]. Also, safe PPO is considered to be more efficient in sampling and learning policies than other policy-based techniques [22], [25].

3.3.7 Monte Carlo Tree Seach (MCTS)

Biggest benefit using the MCTS it its ability to predict driving maneuvers, therefore, it has an enhanced performance and stability while driving. This is achieved with DNN obtaining the probabilities of specific action-states outcomes. MCTS the uses this information to reconstruct possible trajectories and choose the optimal one based on a current road conditions [23]. In more detail, the whole process occurs in four main stages: selection, expansion (based on the maximized q-values), simulation and backpropagation (to reinforce neurons yielding best results for the network). Notably, in the simulation phase random action are taken (as a part of the exploration) and the whole process last till the termination condition are reached. Once that happens backpropagation occurs which reinforces the most favourable neurons in the system. Advantages of this technique include: possibility to stop the algorithm whenever and return the best estimate, however, its main disadvantage is very long time needed for searches to converge to a good result (especially a downside given the dynamic nature of autonomous driving).

3.3.8 Overview of the methods

Across the works presented, for the tasks, authors mainly focused on either vehicle control [15], [19], [21] or motion planing and navigation [22], [23]. The first group focuses on following a predefined path or regulating some physical properties to meet a target (say a vehicle speed to stop when needed [15]), while the second took some observations to generate a high-level objectives that where later implemented with some other control technique (e.g. turning at an intersection in a most efficient way). Lastly, some works tried to merge the two where RL-controller both devised a path and then followed it [11], [20]. Example of such end-to-end approaches could be a race car navigating a track with obstacles. Reasonably, given the density of modern traffic, most of the works has been carried out for the urban setting, either at an intersections [20] or highways [21], [22]. Only one work conducted on footage captured at an actual highway [22] an even this one, as all others, has been first conducted in a simulator. Transfer from simulation to reality is still imperfect [29] so question how applicable these techniques really are in the real world in a paramount questions for a future research (we also review it more in detail in the next section based on three examples where such sim-to-real transfer was successful). While such transfer can be improved with use of a realistic physic engines with varying light conditions or introduced noise [11], [19], [23], still, even the advanced rendering techniques cannot capture the complexity of real world [29].

In conclusion, the results give promising improvements over current value based methods, such as nearly 272% increase with DDQN, 60% more stability with MCTS or 3 times quicker convergence with HDQN compared to regular DQN algorithms. In simulation, and for specific tasks such as controlling distance between two cars, some of the methods have even better performance than humans [21]. Notwithstanding, despite some authors suggesting that their strategies could be generalized to real world [11], across most tasks such claims are dubious at best. It is important to remember that a success on par of 99% means that 1 out of every 100 maneuvers fails. Such

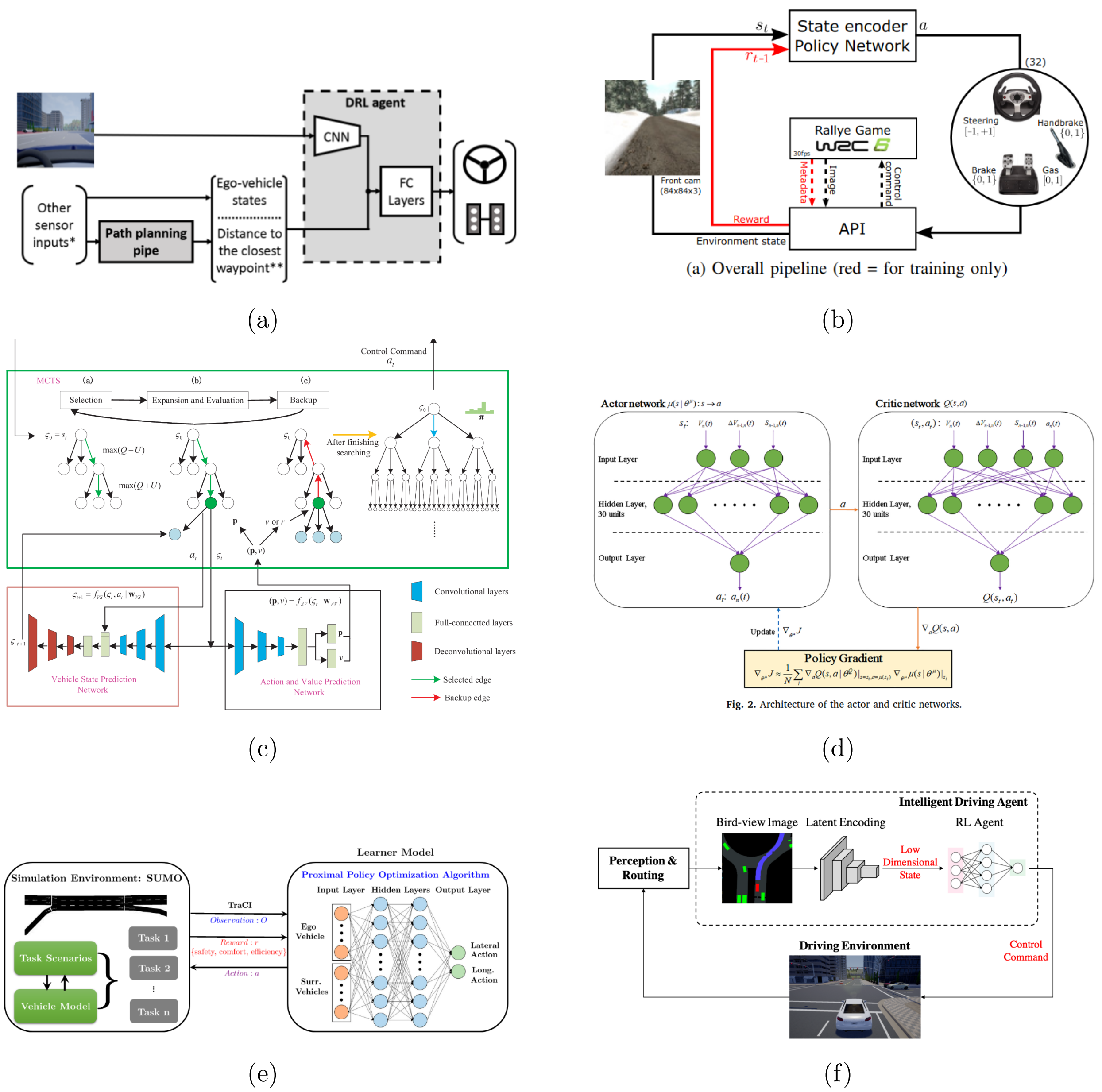


Figure 3: Overview of different RL architectures (name, taken from): (a) HDQN [19] (b) A3C [11], (c) SAC [20], (d) DDPG [21], (e) PPO [22], (f) MCTS [23].

results would paralyse a regular urban driver doing hundreds of such maneuvers each day. In a same light, 0.5% collision rate [22] is far from a car we would consider safe to drive for humans.

3.4 Experimental works and results

Various authors have showcased how reinforcement learning could be applied to autonomous driving with examples such as collision-free navigation at urban intersections [30], breaking system that autonomously decides when to stop the car (based on the readings from sensor) [31] or lake keeping assist [32]. Most of these methods, are end-to-end learned systems. Instead of various connected and independent modules (e.g. computer-vision-enhanced scene understanding, mid-level model-based optimization planers or pre-planed rule-based decision [27]), RL for autonomous

	A. Kendall. et. all. [1]	A. Folkers, et. all. [34]	A. Amini, et. all. [33]
reward and termination	forward speed; breaking traffic rules or human input	distance to certain target; end of episode, collision, exceding speed, max time	distance driven without intervention; car moving out of the lane, exceeding max possible rotation, crash
state	monocular camera feed, steering and speed measurement composing state vector	angle acceleration, position, ”perception map” ternary obstacle 2D grid with -1,0,1 entries	RGB camera
convolutional layers	variational encouder	CNN with two dense layers ReLu activation and double max pooling	CNN but no recurrence
action	steering and speed continuous from 0 to 1 for speed and -1 to 1 for angle	acceleartion and steering wheel angle continuous action space	steering angle and speed both continuous
real world test enviroment	countryside road without any traffic, signs or intersections	typical parking lot with other stationary cars, cardboard boxes acting as obstacles	contryside road, no traffic or intersections
agent	actor-critic framework both actor and the critic are represented by dense layers	deep actor critic, NNs representing functions approximation for policy and value	model free policy optimization, NN for agent outputs prediction of continous distribution
policy	Deep Determinsitic Policy Gradient	Proximal Policy Optimization (PPO)	policy gradient
results	the best policy has 0 disengagements over a 250m road required only 11 episodes to learn	non-quantitative but the authors comments that AV can adjust its paths for obstacles	technique is inferior compared to human driver having both higher mean deviation from road center and recovery rate

Table 2: Review of three papers that implemented real world RL for autonomous driving

driving offers to provide singular solution such complicated technology stack. This is achieved by RL agent training by itself which could in theory drastically reduce costs and complexity of deploying AV to real world. However, for some claim to become a reality we first need to have experimental evidence that these systems can safely co-exist with humans. Previous works [11], [15], [19]–[23] where analysed in simulation. In the next section we will focus on more relevent real life implementation [1], [33], [34].

3.4.1 Real life implementation of RL for control of AV?

Unfortunately, there are only spares examples of applying reinforcement learning for control of real world AV. Current surveys and literature reviews [17], [18], [26], [27] found only three examples when the algorithm was successfully deployed in the real life on a full scale platform [1], [33], [34]. Since these examples are scares we will focus in detail on each one of them as well as their differences and similarities, these are also presented, alogsite some key metrics constituting a typical reinforcement learning problem (as defined in figure 2), in table 2. We also provide comments one authors results.

Reward and termination: Starting from the first category, the termination condition across all works presented is fairly similar revolving around breaking some traffic rules or endangering human life. On the other hand, authors vary in choosing the reward for their system. While [1] focuses on the speed, encouraging more aggressive maneuvers, [33] defines reward as distance without intervention, thus focusing and promoting safe behaviour. Both authors use similar techniques of policy gradient with neural networks representation for actor (and critic in case of [1]) but by choosing a different reward, and so a fundamentally different task for RL agents to learn, they obtain completely different results. Such observations emphasises that reward engineering and tuning is still a big obstacle on road to creating autonomous clever agents that can learn by themselves. One way how it can be overcome is by devising the reward as a linear combination of different factors like safety (quantified as time to the collision) or comfort (rate of change of acceleration) each with different weight. However, this still requires humans to define these rewards and fine tune the weights. Another example of overcoming the problem of to specific reward function is to make them as general as possible. Using distance to a target as the reward does not promotes either safe or aggressive driving but rather leaves this for the RL agent to figure out during the learning period [33]. These reward might be harder to teach, as they required more examples, however, using model-based learning and parallel computation they should generalize better without that much cost.

State and action: Input states are similar across the works, consisting of sensor fusion of camera feed, car acceleration and the steering angle [1], [34], with the only exception of [33] where only the RGB camera is used. In each example the camera reading is filtered using convlutional neural networks. CNN are used for feature extraction from the feed, consequently allowing the AV to associate certain patterns in its vision with control action it should take. Remarkably, apart from [1] which uses a variational encoder (more sophisticated vision detection system), all authors have rather shallow and simple networks, e.g. only two dense layers with linear activation (ReLu) [34] and no memory of the past [33]. This emphasises that RL agents are general enough to navigate real world without the need for a detailed object detection with only broad patters at hand.

Agent and policy: All actions the agents can execute are the change of steering angle and speed (or acceleration which is identical in principle). Interestingly, contrary to initial definition of RL problem [12] each of these actions is continuous outputting a normalized result from zero or minus one to one. Both the continuous definition of action space and normalization of the outputs is aimed at making generalize agents that are independent of the hardware they need to run on. Without the discrete actions available all agents must incorporate some form of policy gradient methods (like Deep Determininst Policy Gradient - [1], Proximal Policy Optimization - [34] and Policy gradient - [33]). Further, the real-world complexity and high dimensional state-space forces the use of neural networks as a function approximation for the representation of policy and value functions (value functions are still present in some policy gradient techniques since [1] uses the actor-critic framework requiring it). Example architecture used for a real world RL implementation [1] has been presented on figure 1. Fundamentally, it is a feedback loop where steering and speed measurements, alongside camera input (compressed through convolutional layers), are condensed into a state vector, which is fed into the agent. Agent itself, is represented by actor approximated by a dense layers and it action output is feed both to the autonomous car, as well as critic (another dense layer function approximation). It then uses the reward to update the Q-value and improve, or critique, the next prediction of the actor.

Environment and experiments: The environment for the studies was either a countryside road or a typical small-town parking. All these had in common a lack of any traffic, signs, intersections or pedestrians. In other words, more sophisticated elements of the urban setting. The tests where very simple, such as lane keeping, obstacle avoidance, recovering from crash-facing state. They were performed at low speeds, 3 m/s was a limit speed in [34]. The algorithm can fullfill the basics tasks, as adjusting path based on obstacles [34] or lane keeping over short, 250m, distances [1], nonetheless, authors note that the techniques are inferior compared to a human driver [33]. In addition, compared to state-of-the art AVs which can navigate complex urban setting with other participants in changing light and weather conditions and considering different traffic rules[4], [35], reinforcement learning based control does not even come close to recreating such outcomes.

Results: In principle this technique could be very powerful, however, current results emphasise that as of now it is still in its infancy. With a first successful deployment of RL on real car reported only around 2018 [1] and the open source tools, computing power and mathematical frameworks becoming mainstream only around 2016 (coinciding with the breakthrough of the DeepMind work [8]). While the idea to map raw camera or laser input directly to steering is not new, with a famous Carnegie Mellon autonomous car ALVINN originating as back as 1989 [36], current algorithms pose to learn by themselves, rather than from hand crafted data sets. Besides, the full end-to-end architecture of these is incredibly well generalizes for different settings, as compared to a rigid multi-module systems used in the present cars (see 2.1). RL-based controller can also learn by themselves how to collect data and then use it for improvement in the future. Nevertheless, the results of control of AV in current literature are still far from adoption in real life.

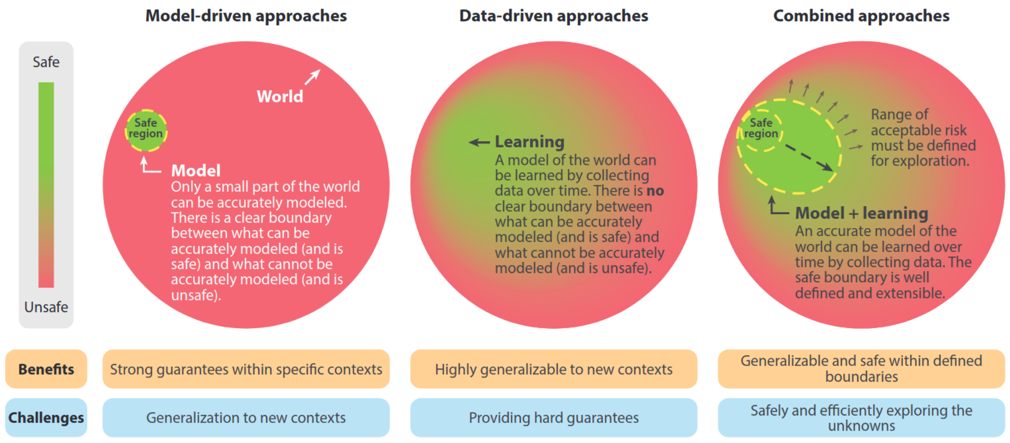
3.5 Limitation of reinforcement learning approaches

One of the biggest limitations of current RL is its applicability for a real world systems. Physical world is notoriously hard to model in a simulation, due to the various unpredictable events and uncertainties. In a comprehensive overview of challenges in transition of RL agents from physical simulator to the real world [37] authors summarise nine different challenges holding back real world RL. These are summarised here for completeness: learning on the real system from limited samples, system delays, high-dimensional continuous state and action spaces, non-violatable system constraints, partially observable and non-stationary systems, multi-objective reward functions, real-time learning, off-line training from the fixed data and providing explainable policies.

As we formulated earlier, autonomous driving touches upon many of these problems as we need to interface with hard safety constraints and learn in real time on limited amount of samples. Not only this but also we would wish to be able to have explainable policies, especially in the case of fatal crashes so that we can prevent them in future. However, casting this aside safety is one the most significant obstacle for large-scale adoption of autonomous cars as it acts as a safeguard against some humans getting injured.

4 Safety constraints and guarantees.

Safety is often to most critical area of research for many autonomous systems that are deployed among humans. It is also a relatively old problem. As of now there are various industries that have developed techniques for control of safety-critical applications such as adaptive control that is used as an autopilot for passenger airplanes [38]. The standard control theory approach involves developing a detailed model of the differential equations governing the system and using them to devise an optimal control commands. Hence, given perfect knowledge of underlying dynamics we can provide high level of safety as we can expect how our action can influence our results. Unfortunately, such approaches have several drawbacks. Firstly, dynamical systems can change in time, imagine an airplane which at every cruise has a different mass distribution of its cargo



Only a small part of the world can be accurately modeled. There is a clear boundary between what can be accurately modeled (and is safe) and what cannot be accurately modeled (and is unsafe).

A model of the world can be learned by collecting data over time. There is **no** clear boundary between what can be accurately modeled (and is safe) and what cannot be accurately modeled (and is unsafe).

Range of acceptable risk must be defined for exploration.

An accurate model of the world can be learned over time by collecting data. The safe boundary is well defined and extensible.

	Model-driven approaches	Data-driven approaches	Combined approaches
Benefits	Strong guarantees within specific contexts	Highly generalizable to new contexts	Generalizable and safe within defined boundaries
Challenges	Generalization to new contexts	Providing hard guarantees	Safely and efficiently exploring the unknowns

Figure 4: Comparison of model, data and combined driven approaches with their corresponding benefits and challenges taken from [39]

.

or fuel and just such change can render our controller ineffective. Theoretically, this can be fixed by performing a system identification, i.e. experimentally or theoretically obtaining the set of governing differential equations with their corresponding constants, every time, however, such process can be lengthy, costly and difficult in many cases.

On the other hand, while standard reinforcement learning approaches can deal with imperfect prior knowledge of the model, to a point where they can control even systems with unknown previous dynamics, they cannot provide any safety guarantees. Thus making RL infeasible for a safety-critical deployment. Furthermore, state of the art RL algorithms require high volumes of data to train doing so often in real setting. Such testing can be dangerous, as during initial learning period algorithm does semi-random actions, and very costly, as thousands of trial-and-error runs are needed for any satisfactory learning. On the other hand, simulated data does not necessarily transfer well into real world with current simulators and has problems of it own [29].

This dilemma between the standard control theory and RL approaches has been presented in figure 4, which succinctly represents on high level the pros and cons of both techniques. In particular merging these two to develop techniques that can provide formal safety guarantees, while being able to account for the imperfect prior knowledge of dynamics and having a small data-foot print, has been a large focus in the recent academic literature. In a broad review of current techniques for achieving safe learning in robotics [39] authors analyse different approaches from both control theory perspective as well as reinforcement learning communities. They reviewed 26 different techniques and classified them into three main groups, namely, learning uncertainty dynamic to safely improve performance, learning-based control under dynamics uncertainty and encouraging safety and robustness in RL.

Building from their work, in our review we quickly analyse the first two categories, as they are less applicable for the RL deployed on autonomoys cars (as they are more focused on classical model-based controllers), and instead focus on three main algorithms used for encouraging safety and robustness in RL. Choosing the ones that are most suitable for autonomous driving, namely, safe exploration, uncertainty-aware RL and learning a safety critic. These three are in detailed analysed in a table 3 with rows breaking down what a given technique aims to solve, how it is achieved, what are the advantages and limitations as well as quick comment on the experimental results.

	Pham et al. (2018) [40]	Kahn et al. (2017) [41]	Srinivasan et al. (2020) [42]
technique	**safe exploration**	**uncretinaty aware RL**	**learning a safety critic**
motivation	coupling of stochastic policies with constrained optimization	safe navigation in unknown enviorments	learning safety transfer between tasks
approach	learning architecture that takesas input possibly unsafe actions and outputs actions satisfying given constraints.	performing collisions at low speeds until the confidence is gained;	train a safety-critic estimating the probability of failure and constrain the actions of policy
detailed execution	DNN agent and a constrained optimization layer; compare an action and constraint prediction only execute if the constrains are not violated	train uncertainty-aware collision prediction model (DNN with bootstrapping; speed-depended collision cost;trajectories produced with MPC controller	in parallel learn safety critic and optimal pre-trained policy; next fine-tune to a new target task; gathering data all policies are constrained by critic and safety constrain cost
advantages	adaptable system that can learn online in real life with moving target and obstacles	possible integration for a model-predictive control pipeline	pre-trained framework that can be adjusted to an new task with a safe learning process
possible future work	applying the technique to the bipedal locomotion or multi-agent system	generalizing the work for other dynamical systems	optimizing the pre-training; handling out-of-distribution data for critic; real world tests
limitations and challenges	learning policy from scratch is to time consuming (better action space exploration); formalizing safety constraints is challenging;	succeses of method i.e. if we have a "learning" or fatal crash is sensitive to uncertainty estimates; evaluating these is hard	high computational cost of the pre-training and the fine-tuning of the system
experiments	3D reaching on a 6-DoF industrial arm with collision avoidance;	mobile robots (drone, RC car) navigating in an unknown enviroment with carboard obstacles	Set of 3D (robotic locomotion, one arm cube rotation) and 2D (navigating grid) tasks with different actions to perform
results	successfull real life trajectory execution with collision avoidance in 6h of learning, no collision during training over 7000 episodes	both car and drone have ∼30% less crashes and at 5-10% lower speeds this increases with iterations; successful completion is on par with regular system	overall more stable learning in 2/3 tasks ∼20% faster learning; ∼60% smaller failure rate during fine tuning than average of 4 different methods; 2nd best maximum total reward

Table 3: Overview of different techniques use to encourage safety and robustnes in RL based on

4.1 Fundamental concepts and definitions.

As a side note, to unify the assessment of effectiveness for achieving safety authors of the same review [39] define three different main safety levels: soft constraints (when possible minimal violations of robot path and constrain set are allowed, level I), probabilistic constraints (where no violations will occur with high probability, level II) and hard constraints (when there are no violations, level III). Also it is important to distinguish between different prior models of dynamics for a system. Here we simply mean whether we have derived theoretically differential equations of motion and if so are they linearised.

4.2 Safely learning uncertain dynamics and safety certification.

Stemming from the current methods used in the industry, where the governing differential equation of the system are manually derived and then applied in the controller block, it is possible to embed these as well into the model-based RL agent [43]. In contrast to controllers with static gains, the specific parameters or weights of the equation for RL are learned online. As a result this overcomes any possible disturbances or model inaccuracies (as long as the physical geometry of the system is preserved). However, still the major obstacle for such technique is how not to risk a failure while learning parameter values. More in detail, it is useful to teach algorithm how to safely recover from exploratory actions so that starting from an initially safe policy the estimates can collect data inside the safe region. Additionally, such actions could increase the region of attraction of the system, i.e. subset of the space that is forward invariant (meaning that each trajectory originating in it states in the set and converges to some stable equilibrium) [43]. Although, this approach mostly provides probabilistic constraint satisfaction and require prior linear dynamics to be known, on a toy inverted pendulum problem, both the safe set was estimated correctly and after only 50 iterations safely optimized policy converged to a stable pendulum position nearly 33% quicker than its non-optimized counter-part.

In contrast to [43] a different group of solutions, safety certification, focuses on constraining the output of control policy with a backup controller that is provably safe. Remarkably, the main policy do not even have to inherently consider safety. Apart from working with more imperfect model-knowledge (structured nonlinear vs linear dynamics for safe learning) on average these also provide hard constraints satisfaction [39] rather than probabilistic ones. Nonetheless, this comes at a cost of such techniques being more complicated and less relevant for our review. [44] as they focus less on RL formulation itself.

4.2.1 Enouraging safety for RL

The selected approaches for encouraging safety in RL [40]–[42] has been presented in table 3. These were selected as they are most relevant to the autonomous driving problem and could be relatively simply integrated into the existing DRL-based AV pipelines. They are also general enough to outline the space of how approaches for this problem are developed. All presented works have been first conducted in the simulation

4.2.2 Safe exploration.

The first technique, safe exploration, focuses on defining a region that is fully excluded from the learning process. This is achieved by an additional constraint layer. It is superimposed on the outputs of the policy that can predict how the system will evolve [40]. It further, safeguards against the policy ever generating an action that would lead to entering the unsafe region. Reference [40] uses a reward strategy guaranteeing that during the training, the safety constraints are never violated. In specific, during the interaction with environment phase the we first develop a constrained optimization layer and then use a policy update strategies. The first one, is achieved by solving a constrained least-squares problems and finding the safe control actions with a differentiable quadratic program solver, type of a convex optimization. While the other simply clips the policy outputs if it can violate some constraints. This technique has experimentally shown good results and no violations, however, it is very computationally costly.

4.2.3 Uncertainty-aware RL.

Never entering an unsafe region, say if it could injure a human or damage the robot, sometimes might be a necessity and thus a prefered solution. However, if that is not a case, such overly conservative approach could lead to an extended length of a learning process, as we need to discard some actions if they are unsafe. Furthermore, it increased computational complexity and impares the exploration phase [40]. Also, it relies heavily on the manual specification of a region where we do not want to end up. Although, authors have shown that their model can cope with both moving target and the excluded region, defining the latter is laborious, does not generalize well and could be very problematic in an unknown environments. One possible solution, that assumes the unknown and possibly hazardous region will not results in a failure of the system, is to enter it while behaving very cautiously. Then, once the dynamics of this new environment are mastered, ideally the robot would start to behave more aggressively. As an example, consider a human driver on his way from home to work, who can cut corners, confidently speed and perfectly stop when needed but in a completely new setting follows all the rules and acts conservative. What is more, at a slower speeds collisions in the exploration phase of RL have a lesser chance to result in damage, yet, their occurrence still provides a useful information for the policy about how to prevent such outcomes in the future.

One possible method to implement this was presented in [41]. This method proved to be very effective in an experimental setup, still it has some limitations. Implemented on a simple drone an a small RC car, with random cardboard obstacles, this method produced less crashes and the ones occurring where at the lower speeds. All this was achieved without significant loss in the successful completion of the task [41]. Nonetheless, the major limitation of this approach is that the speed is fully dictated by our uncertainty prediction. Notably, making an error in it can result in a fatal crush at high speeds. Unfortunately, predicting unknowns for such system is not simple. To improve this it is possible to used bootstrapping (technique aimed at generate multiple populations from data and then training models with each of it; next it revolves around comparing the similarities between them [41], [45]) and dropouts (method to randomly drop units from neural networks during the training phase [41], [46]). Nonetheless, these techniques lack a more general understanding of the surrounding and can often yield incorrect results.

4.2.4 Learning a safety critic

Both safe exploration, [40], and uncertainty aware RL, [41], are efficient solutions that can encourage exclusion of some regions during the learning phase or safer collisions if we enter them. Fundamentally, they act as an external layered filters that augment outputs of the RL agent, instead of being the part of the agents itself. Additionally, both techniques cannot transfer the pre-trained policies or behaviour to new settings. All these challenges could be addressed by introducing a safety critic [39]. In such framework, critic is a pre-trained action-value Q-function that is able to evaluate if a proposed action will result in an unsafe region. Such criterion allows then the solver to consider different and safer alternatives. Reference [42] learn such critic from abstract spare labels and then tries to fine tune them to a policy tasked with a specific job. Hence, promoting safety-related transfer of knowledge in a new but similar task.

The simulated experiments for a broad range of different tasks (such as locomotion, one arm cube rotation or navigation in a grid) have yielded both a faster learning convergence as well as nearly 60% lower failure rate. Still, the success of the method heavily depends on the environment or task-specific hyper parameters [39], [42].

4.3 Overview of the safety methods.

To summarize, all of the methods presented are still fairly new and their authors note [40]–[42] the simplicity of the experiments and lack of demonstrations in systems with higher degrees of freedom (complex bipedal robots, multi-agent system). Also any illustrations of how it could generalize to different dynamical system are not present. This is crucial, since it is possible that for more complicated tasks and robots, such approaches would simply never converge. Compared to safely learning technique [43] or [44], those approaches generalize better to unknown prior dynamic of the system, working even with systems of unknown dynamics. On the other hand, this comes at a cost, as [40]–[42] can only satisfy soft level I constraints. Compared to the standard control theory (that works only with known dynamics) or standard RL (unable to provide any guarantees) all these techniques are nonetheless a positive improvement. Still, future work will probably focus on finding how to provide level II or III constraints for a systems with only generic nonlinear or unknown dynamics. In the light of applicability for AV, only once we will be there it is possible that any real-world implementation will be undertaken and no human will be hurt.

5 Future research directions.

To account for various challenges facing reinforcement learning new research paradigms have emerged in recent years. Here we present a curated narration of three main selected efforts: verifiable artificial intelligence, human-computer interactions and sample efficiency, which where selected as an answer to the most pressing challenges for RL in the future as they would allow for an efficient and trustworthy systems that could safely cooperat with others.

5.1 Verifiable Artificial Intelligence

To begin with, learning based systems such as reinforcement learning are very adaptable and generalize well. This is thanks to, as discussed before, RL policy being often encoded into deep neural network (e.g. DQN). Such representation has shown the most success in the literature [9]–[11], [14], however, one of its major limitations is that vast number of parameters in the network, approximated between few thousands to milions by [47], lacks any human-understandable meaning. This causes various learning based system to behave like a sort of black-box algorithm. Their structure makes manual fine tunning RL systems impossible and our inability to explain these systems often limits its applications in safety-critical systems such as autonomous driving. That is the case, since, as we can encounter edge cases it is possible that our network may results in a critical failure for no apparent reason. On top of this, prominent lawmakers call for the sector to tightly regulate ethical dilemmas, like questions whether car should protect the driver at all costs or not, and embed them into the AV architecture.

Comprehensive review of the five biggest challenges and corresponding principles aiming at developing a formal verification methods in AI has has been presented in [47]. These could act as a guidance in developing these systems in the future. Developing structure mentioned in [47] allows us to verify various assumptions or actions we want our autonomous to take. On the other hand, it is necessary to cast our changing ethical considerations and legal requirements into the decision making layer structure of such systems. Authors in [48] have proposed a Value Sensitive Design (VSD) framework, which is a systematic way for engineers to incorporate existing technical solutions with legal or ethical requirements that decrease the opaque decision-making structure of AVs.

Building from work of [47] which proposes a set of general solutions to follow and reference [48] aimed at incorporating local laws or requirements one recent work [49] aimed at more quantitative incorporation of these observations. Authors propose to resolve the enormous set of conflicting objectives by using "rulebooks", set of priority ordered groups. Sparing the details regarding the derivation of the formal rules covered in the paper, authors analyse real AV examples for three cases: unavoidable collision, lane keeping and intersection lane change. Depending on what rule set was used AV: stayed, left the current lane or behaved more or less aggressively. Interestingly authors state that this technique runs on their proprietary AV with only 15 rules and estimate that using their technique for a complete coverage of Massachusetts, including various rare corner cases, they would roughly need 200 rules in a dozen ordered priority groups. They highlight how their technique can easily incorporate different sets of rules, e.g. local, national, international by simple addition to the diagram and stress need for a public discussion on participating in such algorithms creation. While this techniques limits the generality of a possible RL controller we recognize that it may be necessary due to compliance requirements. Still the question lingers if such complicated set of rules can execute at a run time in a split second, time often in possible critical situations an AV may encounter.

5.2 Human-Computer Interactions

From a different perspective, with increasing shift from AV being a demonstration to them regularly participating in urban traffic, one big challenge is to devise techniques for safe and efficient interaction between humans and machines. While driving a car we often use, even if not consciously, various cues about other drivers. Experienced drivers can quickly judge whether someone behaves aggressively or responds dangerously slowly, based on such assessment they can then correct their actions. Inability of artificially trained autonomous cars to understanding such behaviours limits their applicability. There are various current solutions proposed to account for this. In work of [50] authors analyse conflictive and competitive scenarios of interaction between human and autonomous vehicles while merging into highway. Interestingly, instead of trying to model the behaviour of human they devise so called altruistic agents, which is formalised only from experimental learning.

They test their results in 15,00 randomly initialized simulation scenarios and the gathered data is showcasing that their sympathetic cooperative autonomous agent has a nearly 56% smaller chance of causing a crash, at 2.6% in the study, compared to an egoistic one. This is an improvement from their earlier study [51] where a similar technique in a same highway-merging scenario achieved a 30% improvement, standing at 5.3%, for a socially cooperative case. While these results consider only one interaction scenario and cover 2D toy problems, assuming point mass cars and ideal state estimation, drops of around 50% are a significant improvement. The biggest limitation of the study, as noted by the authors themselves, is the human-expert fine tuning of reward for each distinct driving scenario.

As a relevant side note, one field to address this is the inverse reinforcement learning where algorithm seeks to automate reward definition by learning from expert demonstrations. Works such as reference[52] use only 20 human demonstrations of various simple tasks such as placing dishes, pouring water into a cup (each time slightly alternating the starting position) to develop an RL algorithm that has around 80% success rate in repeating these tasks while implemented on a real robotic manipulator. These results are applicable only to a very specific tasks and required a human-expert to manually perform the task several times. Moreover, they introduced a bias where human performed trajectory was assumed to be an optimal one, which does not necessarily need to be the case especially for a non-intuitive, complex systems. To overcome these limitations one group published a work [53] where the overall human-expert input is even smaller. They designed and tested in real world an algorithm where robot learned the reward function just by analysing a small number of examples of successful outcomes of the task just assessed by binary labels of the user-operator. Together works to incorporate human-machine interaction [50], [51] as well as making the systems smarter [52], [53] could yield better systems in the future.

5.3 Sample efficiency

Lastly, as mentioned before in the review, one of the major limitations for development of autonomous driving in real-world setting is its high sample inefficiency, i.e. large number or tests or interactions with environment needed during the learning process. This is often costly and can be dangerous. Watching how animals or humans can learn a task from just few demonstrations researchers have started to wonder if there is something unique about they way in which they leverage already learned behaviours. In a literature review [18] authors investigate techniques to increase the sample efficiency of RL algorithms and they identify four main research directions aiming at overcoming them: reward shaping (where we try to teach the agent intermediate goals with a more frequent reward function), meta-learning (algorithm to learn new skills from small amounts of experience), actor critic methods (policy gradient algorithm described later) and transfer learning (reusing architecture of a previously trained policy and adjusting it for a new task). Here we will investigate more in depth meta-learning as it have shown the greatest success for for implementation in AV.

To understand meta-learning imagine that for you where able to perform multiple tests with your car at various surfaces, e.g. concrete, asphalt, dirt, sand, etc. For each of these you employed a reinforcement learning algorithm which taught itself an optimal policy for each of these settings. Riding in real world now you expect to encounter each of these surface conditions. Furthermore, some of new roads probably could be approximated as a mix of say concrete and asphalt data you have gathered. Thus, instead of learning a completely new policy it might be desirable to find a meta-learning policy that chooses from the already pre-trained policies selecting the optimal ones or tweaking them to enhance performance [54]. Compose a new policy from previously learned ones in order to address a novel task is one way how to decrease the burden of learning tasks from scratch each time and share experience accorss the platforms. Authors in [54] test their algorithm on a simple model of a car that needs to drive to a specific point. They teach the car in a simulation where they have different dynamics one modeling slippage and the other alignment issues and these two vary in space so that car needs to pass through different regimes. In their case meta-policy needs to choose between two basis policies, each trained separately in completely different dynamic setting. The meta-policy does not know the spatial distribution of different dynamics regions but by interacting with the environment it learns how to choose between the two pre-trained policies. All in all, these approaches improve the transfer learning and increase the sample efficiency of learning methods.

6 Discussion

The main focus on this work was on giving an informative account of how advances in recent reinforcement learning methods can be applied for the field of autonomous driving and if so how effective would they be. In contrast to different reviews out there [17], [27], [39] the main contribution of this one is the strong emphasis for the actual real-world implementation and safety investigation. For the first one, not only did we comment on all such demonstration up to date, but our critical-analysis in many different sections revolved around answering a question of how well a given technique could transfer in the real world. Secondly, we also extensively reviewed how safety could be embedded in such systems with examples of three different approaches described in detail.

From a different perspective, another important finding of our study is that there are two similarities across the most successfully works. Firstly, nearly all the works used deep learning as function approximation for agent and policy. This was motivated by an state vector being high dimensional (camera feed input feed to our RL algorithm). For these tabular system of data management are very inefficient (as the data amount scales exponentially with more examples), however, deep learning based solution (where more data just causes more backpropagation in network which occurs at a fixed memory cost) is less of a problem. Secondly, also from our study, the actor-critic with policy gradient methods is the most successful technique for real life implementation of RL, featuring in nearly all examples we investigated. The primary reason for this might be that actor-critics are a versatile method combining the best of value-based on policy-based solver. Furthermore, already there are works that show how to equip such architectures with layers improving both the sample efficiency and introducing soft safety constraints. Lastly, and in part as a result of a previous point, these allow us to hard code more information for the network making it more explainable for the user.

The last interesting result of our work is that since nearly all of the papers use some form of camera feed convolutional neural networks, acting as a pattern recognition filters for RL algorithms, play a pivotal role in creating the state vector for RL solver. Despite this, to our attention have not come any papers that evaluate how these crucial elements of modern RL pipelines could be systematically developed. This results in experimental tinkering with the architectures of such networks (e.g. how many layers, what type of activation function, etc.) without any formal framework for this. In turn it makes generalization of some of the methods dubious.

7 Conclusions

The conclusion are pretty straight forward, namely that the state-of-the-art reinforcement learning algorithms applied for AVs are barely capable of undertaking even the simple control and navigation tasks, being nowhere near human-level performance or even the modular industry approach.

This harsh account comes from two lines of reasoning. First, the inherent nature of learning systems is poorly verifiable (thus we cannot easily predict how the system will behave) but what is more, embedding safety guarantees into it is an open problem without any solutions that can be easily generalized. Without ensuring safety, even on a level of probabilistic constraints, we have no way of guaranteeing a safe performance of the system once it starts interaction with humans. Secondly, from purely experimental perspective, the algorithms have been tested only in very simple setups (like rural roads without any signs, intersection or other road participants) and even in such setting the results has been mixed as robots where either inferior to humans or they could only behave well over short distances. It follows that in more complex environments the results could be catastrophic. What is more, even in simulations some more advanced techniques fail simple tasks such as navigating a roundabout. Despite some other challenges, we firmly believe that future researchers should first prioritize the safety and real-world implementations in their work. These are major unanswered questions that are absolutely instrumental for any meaningful and real progress in the field.

References

[1] A. Kendall, J. Hawke, D. Janz, *et al.*, "Learning to drive in a day," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8248–8254.

[2] W. H. Organization, *Global status report on road safety 2018*. World Health Organization, 2018.

[3] C. Xu, Z. Ding, C. Wang, and Z. Li, "Statistical analysis of the patterns and characteristics of connected and autonomous vehicle involved crashes," *Journal of safety research*, vol. 71, pp. 41–47, 2019.

[4] Waymo. "Waymo safety report, on the road to fully self-driving." (2021), [Online]. Available: `https://downloads.ctfassets.net/sv23gofxcuiz/4gZ7ZUxd4SRj1D1W6z3rpR/2ea16814cdb42f9e8eb34cae4f30b35d/2021-03-waymo-safety-report.pdf` (visited on 10/30/2022).

[5] S. S. Banerjee, S. Jha, J. Cyriac, Z. T. Kalbarczyk, and R. K. Iyer, "Hands off the wheel in autonomous vehicles?: A systems perspective on over a million miles of field data," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, IEEE, 2018, pp. 586–597.

[6] R. L. McCarthy, "Autonomous vehicle accident data analysis: California ol 316 reports: 2015–2020," *ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg*, vol. 8, no. 3, 2022.

[7] J. Betz, A. Heilmeier, A. Wischnewski, T. Stahl, and M. Lienkamp, "Autonomous driving—a crash explained in detail," *Applied Sciences*, vol. 9, no. 23, p. 5126, 2019.

[8] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[9] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to walk in minutes using massively parallel deep reinforcement learning," in *Conference on Robot Learning*, PMLR, 2022, pp. 91–100.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[11] M. Jaritz, R. De Charette, M. Toromanoff, E. Perot, and F. Nashashibi, "End-to-end race driving with deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 2070–2075.

[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[13] Wikipedia, *Markov decision process — Wikipedia, the free encyclopedia*, `http://en.wikipedia.org/w/index.php?title=Markov%20decision%20process&oldid=1124829194`, [Online; accessed 13-December-2022], 2022.

[14] L. Kaiser, M. Babaeizadeh, P. Milos, *et al.*, "Model-based reinforcement learning for atari," *arXiv preprint arXiv:1903.00374*, 2019.

[15] Y. Zhang, P. Sun, Y. Yin, L. Lin, and X. Wang, "Human-like autonomous vehicle speed control by deep reinforcement learning with double q-learning," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1251–1256.

[16] B. B. Elallid, N. Benamar, A. S. Hafid, T. Rachidi, and N. Mrani, "A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving," *Journal of King Saud University-Computer and Information Sciences*, 2022.

[17] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[18] B. R. Kiran, I. Sobh, V. Talpaert, *et al.*, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[19] E. Yurtsever, L. Capito, K. Redmill, and U. Ozgune, "Integrating deep reinforcement learning with model-based path planners for automated driving," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2020, pp. 1311–1316.

[20] 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.

[21] M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, and R. Ke, "Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving," *Transportation Research Part C: Emerging Technologies*, vol. 117, p. 102 662, 2020.

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

[23] J. Chen, C. Zhang, J. Luo, J. Xie, and Y. Wan, "Driving maneuvers prediction based autonomous driving control by deep monte carlo tree search," *IEEE transactions on vehicular technology*, vol. 69, no. 7, pp. 7146–7158, 2020.

[24] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, PMLR, 2018, pp. 1861–1870.

[25] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, PMLR, 2015, pp. 1889–1897.

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

[27] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, pp. 187–210, 2018.

[28] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016.

[29] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, *et al.*, "Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[30] Y. Guan, Y. Ren, H. Ma, *et al.*, "Learn collision-free self-driving skills at urban intersections with model-based reinforcement learning," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2021, pp. 3462–3469.

[31] H. Chae, C. M. Kang, B. Kim, J. Kim, C. C. Chung, and J. W. Choi, "Autonomous braking system via deep reinforcement learning," in *2017 IEEE 20th International conference on intelligent transportation systems (ITSC)*, IEEE, 2017, pp. 1–6.

[32] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," *arXiv preprint arXiv:1612.04340*, 2016.

[33] A. Amini, I. Gilitschenski, J. Phillips, *et al.*, "Learning robust control policies for end-to-end autonomous driving from data-driven simulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1143–1150, 2020.

[34] A. Folkers, M. Rick, and C. Büskens, "Controlling an autonomous vehicle with deep reinforcement learning," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019, pp. 2025–2031.

[35] A. Argo. "Developing a self-driving system you can trust." (2021), [Online]. Available: `https://www.argo.ai/wp-content/uploads/2021/04/ArgoSafetyReport.pdf` (visited on 10/30/2022).

[36] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," *Advances in neural information processing systems*, vol. 1, 1988.

[37] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, *et al.*, "Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[38] K. A. Wise, E. Lavretsky, and N. Hovakimyan, "Adaptive control of flight: Theory, applications, and open problems," in *2006 American Control Conference*, IEEE, 2006, 6–pp.

[39] L. Brunke, M. Greeff, A. W. Hall, *et al.*, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, 2022.

[40] T.-H. Pham, G. De Magistris, and R. Tachibana, "Optlayer-practical constrained optimization for deep reinforcement learning in the real world," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 6236–6243.

[41] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv preprint arXiv:1702.01182*, 2017.

[42] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn, "Learning to be safe: Deep rl with a safety critic," *arXiv preprint arXiv:2010.14603*, 2020.

[43] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, "Safe model-based reinforcement learning with stability guarantees," *Advances in neural information processing systems*, vol. 30, 2017.

[44] A. Taylor, A. Singletary, Y. Yue, and A. Ames, "Learning for safety-critical control with control barrier functions," in *Learning for Dynamics and Control*, PMLR, 2020, pp. 708–717.

[45] S. Abney, "Bootstrapping," in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 360–367.

[46] H. Wu and X. Gu, "Towards dropout training for convolutional neural networks," *Neural Networks*, vol. 71, pp. 1–10, 2015.

[47] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Toward verified artificial intelligence," *Communications of the ACM*, vol. 65, no. 7, pp. 46–55, 2022.

[48] S. Umbrello and R. V. Yampolskiy, "Designing ai for explainability and verifiability: A value sensitive design approach to avoid artificial stupidity in autonomous vehicles," *International Journal of Social Robotics*, vol. 14, no. 2, pp. 313–322, 2022.

[49] A. Censi, K. Slutsky, T. Wongpiromsarn, *et al.*, "Liability, ethics, and culture-aware behavior specification using rulebooks," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8536–8542.

[50] B. Toghi, R. Valiente, D. Sadigh, R. Pedarsani, and Y. P. Fallah, "Cooperative autonomous vehicles that sympathize with human drivers," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2021, pp. 4517–4524.

[51] B. Toghi, R. Valiente, D. Sadigh, R. Pedarsani, and Y. P. Fallah, "Social coordination and altruism in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[52] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International conference on machine learning*, PMLR, 2016, pp. 49–58.

[53] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine, "End-to-end robotic reinforcement learning without reward engineering," *arXiv preprint arXiv:1904.07854*, 2019.

[54] R. Liaw, S. Krishnan, A. Garg, D. Crankshaw, J. E. Gonzalez, and K. Goldberg, "Composing meta-policies for autonomous driving using hierarchical deep reinforcement learning," *arXiv preprint arXiv:1711.01503*, 2017.