A Survey of Deep Learning Applications to Autonomous Vehicle Control

Sampo Kuutti, Richard Bowden, *Senior Member, IEEE*, Yaochu Jin, *Fellow, IEEE*, Phil Barber, and Saber Fallah

***Abstract*— Designing a controller for autonomous vehicles capable of providing adequate performance in all driving scenarios is challenging due to the highly complex environment and inability to test the system in the wide variety of scenarios which it may encounter after deployment. However, deep learning methods have shown great promise in not only providing excellent performance for complex and non-linear control problems, but also in generalising previously learned rules to new scenarios. For these reasons, the use of deep learning for vehicle control is becoming increasingly popular. Although important advancements have been achieved in this field, these works have not been fully summarised. This paper surveys a wide range of research works reported in the literature which aim to control a vehicle through deep learning methods. Although there exists overlap between control and perception, the focus of this paper is on vehicle control, rather than the wider perception problem which includes tasks such as semantic segmentation and object detection. The paper identifies the strengths and limitations of available deep learning methods through comparative analysis and discusses the research challenges in terms of computation, architecture selection, goal specification, generalisation, verification and validation, as well as safety. Overall, this survey brings timely and topical information to a rapidly evolving field relevant to intelligent transportation systems.**

***Index Terms*— Machine learning, neural networks, intelligent control, computer vision, advanced driver assistance, autonomous vehicles.**

I. Introduction

IN 2016, traffic accidents resulted in 37,000 fatalities in the United States [1] and 25,500 fatalities in the European Union [2]. With the steady increase in the number of vehicles on the road, issues such as traffic congestion, pollution, and road safety are becoming critical issues [3]. Autonomous vehicles have gained significant interest as solutions to these challenges [4]–[7]. For instance, 90% of all car accidents are estimated to be caused by human errors, while only 2% are caused by vehicle failures [8]. Further benefits from autonomous vehicles in terms of better fuel economy [9], [10], reduced pollution, car sharing [11], increased productivity, and improved traffic flow [12] have also been reported.

Some of the earliest autonomous vehicle projects were presented in 1980s by Carnegie Mellon University for driving in structured environments [13] and the University of Bundeswehr Munich for highway driving [14]. Since then, projects such as DARPA Grand Challenges [15], [16] have continued to drive forward research in autonomous vehicles. Outside of academia, car manufacturers and tech companies have also carried out research to develop their own autonomous vehicles. This has led to multiple Advanced Driver Assistance Systems such as Adaptive Cruise Control (ACC), Lane Keeping Assistance, and Lane Departure Warning technologies, which provide modern vehicles with partial autonomy. These technologies not only increase the safety of modern vehicles and make driving easier but also pave the way for fully autonomous vehicles which do not require any human intervention.

Early autonomous vehicle systems were heavily reliant on accurate sensory data, utilising multi-sensor setups and expensive sensors such as LIDAR to provide accurate environment perception. Control of these autonomous vehicles was handled via rule-based controllers, where the parameters are set by the developers and hand-tuned after simulation and field testing [17]–[19]. The downside of this approach is the time intensive hand-tuning of parameters [20] and the difficulty of such rule-based controllers to generalise to new scenarios [21]. Also, the highly non-linear nature of driving means that control methods based on linearisation of the vehicle model or other algebraic analytical solutions are often infeasible or do not scale well [22], [23]. Recently, deep learning has gained attention due to the numerous state-of-the-art results it has achieved in fields such as image classification and speech recognition [24]–[26]. This has led to increasing use of deep learning in autonomous vehicle applications, including planning and decision making [27]–[31], perception [32]–[36], as well as mapping and localisation [37]–[39]. The performance of Convolutional Neural Networks (CNNs) with raw camera inputs has the potential to reduce the number of sensors used by autonomous vehicles. This has led to some organisations investigating autonomous vehicles without expensive sensors

Manuscript received November 14, 2018; revised April 26, 2019 and October 10, 2019; accepted December 16, 2019. Date of publication January 7, 2020; date of current version February 2, 2021. This work was supported in part by the U.K.-Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/R512217/1 and in part by the Jaguar Land Rover. The Associate Editor for this article was K. Wang. *(Corresponding author: Sampo Kuutti.)*

Sampo Kuutti and Saber Fallah are with the Centre for Automotive Engineering, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: s.j.kuutti@surrey.ac.uk; s.fallah@surrey.ac.uk).

Richard Bowden is with the Centre for Vision Speech and Signal Processing, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: r.bowden@surrey.ac.uk).

Yaochu Jin is with the Department of Computer Science, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: yaochu.jin@surrey.ac.uk).

Phil Barber was with Jaguar Land Rover Ltd., Coventry CV4 7JJ, U.K. He is now with Phil Barber Industries, Yorkshire DL7 9UN, U.K. (e-mail: pbarber2@jaguarlandrover.com).

Digital Object Identifier 10.1109/TITS.2019.2962338

such as LIDAR, instead employing extensive use of deep learning for scene understanding, object recognition, semantic segmentation, and motion estimation. The strong results of deep learning in these perception problems have also sparked interest in using Deep Neural Networks (DNNs) to produce control actions in autonomous vehicles. Indeed, autonomous vehicle control often has a strong link to perception, as many techniques use CNNs to predict control actions based on images of the scene, without any separate perception module, thereby removing the separation between the perception and control layer.

Deep learning offers several benefits for vehicle control. The ability to self-optimise its behaviour from data and adapt to new scenarios makes deep learning well suited to control problems in complex and dynamic environments [40]–[42]. Rather than having to tune each parameter iteratively while trying to maintain performance in all foreseeable scenarios, deep learning enables developers to describe the desired behaviour and teach the system to perform well and generalise to new environments through learning [43]–[47]. For these reasons, there has been significant interest in deep learning for autonomous vehicle control in recent years. There are a variety of different sensor configurations; whilst some researchers aim to control the vehicle with camera vision only, others utilise lower dimensional data from ranging sensors, and some use multi-sensor set ups. There are also some differences in terms of the control objective, some formulate the system as a high-level controller which provides, for example, desired acceleration, which is then realised through a low-level controller, often using classical control techniques. Others aim to learn driving end-to-end, mapping observations directly to low-level vehicle control interface commands. Although there has been a large variety of different approaches used to tackle autonomous vehicle control via deep learning, currently there is a lack of analysis and comparison between these different techniques. This manuscript aims to fill this gap in the literature, by reviewing the deep learning approaches to vehicle control and analysing their performance. Furthermore, the manuscript will evaluate the current state of the field, identify the main research challenges, and make recommendations for the direction of future research.

The remainder of this manuscript is structured as follows. Section II provides a brief introduction to deep learning methods and approaches relevant to autonomous vehicles. Section III discusses recent approaches to autonomous vehicle control using deep learning, which is broken into three categories: (A) lateral, (B) longitudinal, and (C) simultaneous lateral and longitudinal control. Section IV presents the main research challenges from the previous section's discussion. Finally, Section V concludes the current state of the field and provides recommendations for the direction of future research.

II. Review of Deep Learning

In this section, we briefly introduce the deep learning techniques and approaches related to the works discussed in later sections. A brief summary on learning strategies, datasets, and tools for deep learning in autonomous vehicles is given. Since a full description on all deep learning algorithms used in autonomous vehicles would be out of the scope of this manuscript, we refer the interested reader to the insightful texts on this topic in [46]–[52].

A. Supervised Learning

In deep learning, the objective is to update the weights of a deep neural network during training, such that the model learns to represent a useful function for its task. There are numerous learning algorithms available, but most algorithms described in this manuscript can be classified as supervised or reinforcement learning. Supervised learning utilises labelled data, where an expert demonstrates performing the chosen task at hand. Each data point in the set includes an observation-action pair, which the neural network then learns to model. During training, the network approximates its own action for each observation, and compares the error to the labelled action by the expert. The advantage of supervised learning is speed of training convergence and no need to specify how the task should be performed. While the simplicity of the supervised approach is appealing, the approach has some disadvantages. Firstly, during training the network makes predictions on the control action in an offline framework, where the network's predictions do not affect the states seen during training. However, once deployed, the network's actions will affect future states, breaching the i.i.d. assumption made by most learning algorithms [53]–[55]. This leads to a distribution shift between training and operation, which can lead to the network making mistakes due to the unfamiliar state distributions seen during operation. Secondly, learning a behaviour from demonstration leaves the network susceptible to biases in the data set. For complex tasks, such as autonomous driving, the diversity of the data set should be ensured if the aim is to train a generalisable model which can drive in all different environments [56], [57].

B. Reinforcement Learning

Reinforcement learning enables the model to learn to perform the task through trial and error. Reinforcement learning can be modelled as a Markov decision process, formally described as a tuple (S, A, P, R), where S denotes the state space, A represents the action space of possible actions, P denotes the state transition probability model, and R represents the reward function. At each time-step the agent observes a set of states s_t, takes an action a_t from possible actions A, and then the environment transitions according to P. The agent then observes a new set of states s_{t+1} and receives a reward r_t. The aim of the agent is to learn a policy $\pi(s_t, a_t)$ mapping observations to actions such that the accumulated rewards are maximised. Therefore, the agent can learn from its own actions through interactions with the environment and receives an estimate of its performance through the reward function. The advantage of this approach is that no labelled data sets are required and a behaviour which generalises well to new scenarios can be learned through reinforcement learning. The downside of reinforcement learning is its low sample efficiency [58], which means converging to an optimal policy can be slow, thereby requiring time-intensive simulations or costly real-world training [50].

Reinforcement learning algorithms can be divided into three classes: value-based, policy gradient, and actor-critic algorithms [59]. Value-based algorithms (e.g. Q-learning [60]) estimate the value function $V(s)$, which represents the value (expected reward) of being in a given state. If the state transition dynamics P are known, the policy can choose actions which bring it to states such that the expected rewards are maximised. However, in most reinforcement learning settings the environment model is not known. Therefore, the state-action value or quality function $Q(s, a)$, which estimates the value of a given action in a given state, is used instead. The optimal policy is then found by greedily maximising the state-action value function $Q(s, a)$. The disadvantage of this approach is that there is no guarantee on the optimality of the learned policy [61], [62]. Policy gradient algorithms (e.g REINFORCE [63]) do not estimate a value function, but instead parametrise the policy and then update the parameters to maximise the expected rewards. This is done by constructing a loss function and estimating a gradient of the loss function with respect to the network parameters. During training, the network parameters are then updated in the direction of the policy gradient. The main disadvantage of this approach is the high variance in the estimated policy gradients [64]–[66]. The third class, actor-critic algorithms (e.g. A3C [67]), are hybrid methods which combine the use of a value function with a parametrised policy function. This creates a trade-off between the disadvantages of the high variance of policy gradients and the bias of value-based methods [51], [68], [69]. Another separating factor between different reinforcement learning algorithms is the type of reward function used. The reward function used can be either sparse or dense. In a sparse reward function, the agent only receives a reward following specific events, such as success or failure in its task. The benefit of this approach is that the success (e.g. reaching a goal location) or failure (e.g. colliding with another object) is easy to define for most tasks. However, this can further exacerbate the sample complexity issue in reinforcement learning, since the agent would only receive a reward relatively rarely, resulting in slow convergence. On the other hand, in a dense reward function the agent is given a reward at every time-step based on the state it is in. This means that the agent receives a continuous learning signal, estimating how useful the chosen actions were in their respective states.

C. Datasets and Tools for Deep Learning

The rapid progress in the implementation of deep learning systems on autonomous vehicles has led to the availability of diverse deep learning data sets for autonomous driving and perception. Perhaps the most well known data set for autonomous driving is the KITTI benchmark suite [70], [71], which includes multiple data sets for evaluation of stereo vision, optical flow, scene flow, simultaneous localisation and mapping, object detection and tracking, road detection and semantic segmentation. Other useful data sets include the Waymo Open [72], Oxford Robotcar [73], ApolloScape [74], Udacity [75], ETH Pedestrian [76], and Caltech Pedestrian [77] data sets. For a more complete overview of available autonomous driving data sets, see the survey by Yin & Berger [78]. Besides public data sets, there are also a number of other tools available for the development of deep learning in autonomous vehicles. The current leading Artificial Intelligence (AI) platform for autonomous driving is the NVIDIA Drive PX2 [79], which provides two Tegra system-on-chips (SoC) and two Pascal graphics processors with dedicated memory and specialised support for DNN calculations. For more diverse tasks, the MobilEye EyeQ5 [80] provides four fully programmable accelerators, each optimised for a different family of machine learning algorithms. This diversity can be useful in systems where different families of deep learning algorithms have been used. On the other hand, Altera's Cyclone V [81] SoC provides a driving solution optimised for sensor fusion. For a more in-depth review of autonomous driving hardware platforms, see the discussion by Liu et al. [82].

III. Deep Learning Applications to Vehicle Control

The motion control of a vehicle can be broadly divided into two tasks; lateral motion of the vehicle is controlled by the steering of the vehicle, whilst longitudinal motion is controlled through manipulating the gas and brake pedals of the vehicle. Lateral control systems aim to control the vehicle's position on the lane, as well as carry out other lateral actions such as lane changes or collision avoidance manoeuvres. In the deep learning domain, this is typically achieved by capturing the environment using the images from on-board cameras as the input to the neural network. Longitudinal control manages the acceleration of the vehicle such that it maintains the desirable velocity on the road, keeps a safe distance from the preceding vehicle, and avoids rear-end collisions. While lateral control is typically achieved through vision, the longitudinal control relies on measurements of relative velocity and distance to the preceding/following vehicles. This means that ranging sensors such as RADAR or LIDAR are more commonly used in longitudinal control systems. The majority of the current research projects have chosen to focus on only one of these actions, thereby simplifying the control problem. Moreover, both types of control systems have different challenges and differ in terms of implementation (e.g. sensor setups, test/use cases). For these reasons this section is split into three subsections, with the first two subsections discussing lateral and longitudinal control systems, independently, and the third subsection focusing on techniques which have attempted to combine both longitudinal and lateral control.

A. Lateral Control Systems

One of the earliest applications of artificial neural networks to the vehicle control problem was the Autonomous Land Vehicle in a Neural Network (ALVINN) system by Pomerleau in 1989 which was first described in [83] and further extended in [84]. ALVINN utilised a feedforward neural network, with a 30×32-neuron input layer, one hidden layer with four neurons, and a 30-neuron output layer in which each neuron represents a possible discrete steering action. The system used the input from a camera together with the steering commands of the

human driver as training data. To increase the amount of data and variety of scenarios available, the author employed data augmentation methods to increase the available training data without recording any additional footage; each image was shifted and rotated, so as to make the vehicle appear to be situated at a different part of the road laterally. Additionally, to avoid bias towards recent inputs (e.g. if a training session ends in a long right hand turn, the system could be biased to turn right more often) a buffering solution was used where previously encountered training patterns were retained in the buffer. The buffer contained 4 patterns of previous data at any time, which were periodically replaced such that the patterns in the buffer had no right or left bias on average. Both the image shifting as well as buffering solutions were shown to significantly improve the system performance. The system was trained on a 150m stretch of road, after which it was tested on a separate stretch of road at speeds ranging from 5 to 55mph allowing steering without intervention for distances of up to 22 miles. The system was shown to be able to remain, on average, 1.6cm distance from the centre of the road compared to that of 4.0cm under human control. This demonstrated that neural networks can learn to steer a vehicle from recorded data.

The first to suggest reinforcement learning for vehicle steering was the work carried out by Yu [85]. Yu proposed a road following system based on Pomerleau's work utilising reinforcement learning to design a controller. The advantage of which was the ability to learn from previous experiences to drive in new environments and continuously learn and improve its road following ability through online learning. Combining supervised learning and reinforcement learning, Moriarty et al. [86] developed a lane-selection strategy for a highway environment. The results showed that the vehicles with learned controllers managed to maintain speeds close to the desired speed and resulted in less lane-changes. Moreover, the learned control strategy resulted in better traffic flow than manually constructed controllers.

The neural networks utilised in the aforementioned early works are significantly smaller when compared to what is feasible with today's technology [87]. Indeed, while neural networks are hardly new, the research interest and adoption to various applications has exploded in recent years due to increased computing power, especially through parallel graphics processing units (GPUs) which can significantly reduce training time and improve performance. Moreover, the availability of large public data sets and hardware solutions optimised for deep learning have made training and validation of neural network systems easier. Overall, these recent advancements have enabled better performance through more complex systems with vastly increased amounts of training data and episodes.

Utilising deeper models with CNNs, Muller et al. [88] trained a sub-scale radio controlled car to navigate off-road in the DARPA Autonomous VEhicle (DAVE) project. The model was trained with training data collected from two forward-facing cameras while a human was controlling the vehicle. Using a 6-layer CNN, the model learned to navigate around obstacles when driving at speeds of 2m/s. Building

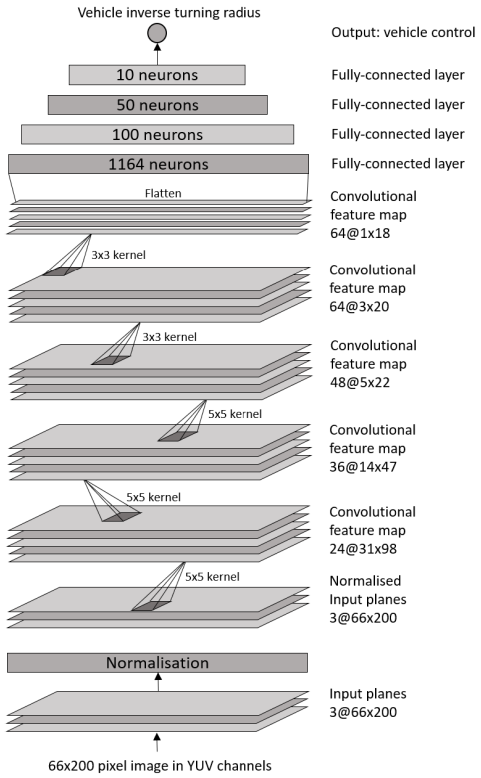

Fig. 1. Convolutional Neural Network utilised in the NVIDIA end-to-end steering system. (Figure recreated based on [87]).

on the approach of DAVE, NVIDIA utilised a CNN to create an end-to-end control system for steering of a vehicle through supervised learning [87]. The system is capable of self-optimising the system performance and detecting useful environmental features (e.g. detection of roads and lanes). The CNN used (see Fig. 1) can learn the steering policy without explicit manual decomposition of the environmental features, path planning, or control actions using a small amount of training data. The training data set consisted of recorded camera footage and steering signals from a human driven vehicle. The CNN consisted of 9 layers, including a normalisation layer, 5 convolutional layers and 3 fully connected layers, with a total of 27 million connections and 250,000 parameters. This method achieved a 98% autonomy in initial testing and 100% autonomy during a 10-mile highway test, measured based on the number of interventions required over a given test time. However, it should be noted that this measure does not include lane changes or turns, and therefore only evaluates the system's ability to stay in its current lane.

A further example of supervised learning for steering of an autonomous vehicle is the work by Rausch et al. [42], where supervised learning was employed to create an end-to-end lateral vehicle controller. Rausch et al. utilised a CNN with four hidden layers, three convolutional layers and one fully connected layer. The training data was the steering angle and front-facing camera footage which was provided by a human steering a vehicle in a CarSim [89] simulation, with imaging captured at 12 frames per second (FPS) at a resolution of 1912×1036. The data collection was collected from a 15-minute simulation run resulting in a total of 10,800 frames.

TABLE I
A COMPARISON OF LATERAL CONTROL TECHNIQUES

Ref.	Learning Strategy	Network	Inputs	Outputs	Pros	Cons	Experiments
[83], [84]	Supervised Learning	Feedforward network with 1 hidden layer	Camera image	Discretised steering angles	First promising results for neural network-based vehicle controllers	Simple network and discretised steering angle outputs degrade performance	Real & Simulation
[85]	Reinforcement Learning	Feedforward network with 1 hidden layer	Camera image	Discretised steering angles	Supports online learning	Simple network and discretised steering angle outputs degrade performance	Simulation
[88]	Supervised Learning	6-layer CNN	Camera images	Steering angle	Robust to environmental diversity	Large errors, trained and tested on a sub-scale vehicle model	Real world (sub-scale vehicle)
[87]	Supervised Learning	9-layer CNN	Camera image	Steering angle values	High level of autonomy during field tests	Only considers lane following, requires interventions by the driver	Real world & Simulated
[42]	Supervised Learning	8-layer CNN	Camera image	Steering angle values	Learns from minimal training data	Noisy behaviour of the steering signal	Simulation
[93]	Supervised Learning	C-LSTM	Camera image	Steering angle values	Considers temporal dependencies	RNNs can be difficult to train, lack of live testing	No live testing, tested on data set image examples only
[96]	Reinforcement Learning	3 feedforward networks	Host vehicle states and road geometry	Vehicle yaw acceleration	Executes lane changes successfully	Limited testing or results, lack of comparison to other lane change algorithms	Simulation

Inappropriate frames caused by bad driving behaviour or graphic errors (e.g. due to a fault in the simulator) were removed from the training data manually. Then, the neural network was trained with three different optimisation algorithms to update the network weights, namely Stochastic Gradient Descent (SGD) [90], Adam [91], and Nesterov's Accelerated Gradient (NAG) [92]. During training Adam resulted in the best loss convergence, while during the evaluation, the NAG trained network performed the best in terms of keeping the vehicle in the centre of the lane. Therefore, convergence of the loss function is not necessarily representative of a well-trained neural network. The neural networks were shown to learn good estimations of the human driver's steering policy, however by comparing the steering angles, it could be seen that the steering signal of the neural networks included noisy behaviour. A potential reason is that the system estimates the required steering angle at each frame, with no context regarding previous states or actions. This results in the steering signals between subsequent time steps varying significantly from each other, causing noisy output. This could be resolved by utilising a RNN to provide memory of previous inputs and outputs for the system, giving it temporal context.

Introducing temporal context to a deep learning steering model, Eraqi et al. [93] utilised a Convolutional Long Short-Term Memory Recurrent Neural Network (C-LSTM) to learn to steer a vehicle based on visual and dynamic temporal dependencies. The network was trained to predict steering angles based on image inputs, and then compared it to a simple CNN architecture used in [94]. Experimental results showed improved accuracy and smoother steering variations when using the C-LSTM network. However, the model was only evaluated offline by comparing the predicted control action against ground truth, which does not necessarily give an accurate evaluation of driving quality [95]. Live testing, where the model can control the vehicle to test the learned driving behaviour, should be used instead.

There has also been lateral control techniques for lane change manoeuvrers presented. Wang et al. [96] used reinforcement learning to train an agent to execute lane change manoeuvrers using a Deep Q-Network (DQN). The network uses host vehicle speed, longitudinal acceleration, position, yaw angle, target lane, lane width and road curvature to provide a continuous value for the desired yaw acceleration. To ensure Q-learning could be used to output continuous action values, a modified Q-learning approach was used to support continuous action values, where the Q-function was a quadratic function approximated by three single hidden layer feedforward neural networks. The proposed approach was tested in a simulated highway environment, with preliminary results showing effective lane change manoeuvrers learned by the agent.

A summary of the research works covered in this section can be seen in Table I. Due to the advancements mentioned previously, the recent trend has been to move to deeper models with increased amounts of training data. Recent works have also investigated introducing temporal cues into the learning model, but this suffers from instability in training. Moreover, many of the models developed so far have been trained and evaluated in relatively simple environments. For instance, most researchers have decided to focus on lateral control for a single task. For example in models trained for lane keeping no decision-making for e.g. lane changes or turns to different

roads have been incorporated in these systems. This opens possible avenues for future research where multiple actions could be carried out by the same DNN. It should also be noted that the majority of these works were trained and evaluated in simulated environments, which further simplifies the task and would require further tests to validate their real world performance. Nevertheless, there have been important developments in this field and these results show great promise for the use of deep learning for autonomous vehicle control.

B. Longitudinal Control Systems

Machine learning methods have also shown promise in applications to vehicle longitudinal control, such as ACC design. The ACC can be described as an optimal tracking control problem for a complex nonlinear system [97], [98] and therefore is poorly suited to control systems based on linear vehicle models or other algebraic analytical solutions [99]. Such traditional control systems provide poor adaptability in complex environments and do not conform to the driver's habits [100]. The strong nonlinear nature of the system makes it difficult to build a vehicle model without significant uncertainty, limiting the effectiveness of model-based solutions. However, neural networks have shown great potential for optimising nonlinear, high-dimensional control systems [40], [41], [101]–[106]. For instance, reinforcement learning can learn an optimal control policy through interaction with the environment, without knowledge of the system model [50]. Furthermore, the strong adaptive capacity and model-free capability of reinforcement learning makes it an attractive solution for ACC design. In early works, Dai et al. [107] proposed a fuzzy reinforcement learning method for longitudinal control of an autonomous vehicle. The method combines a Q estimator network (QEN) with a Takagi-Sugeno-type Fuzzy Inference System (FIS). The QEN is used to estimate the optimal action value function whilst the FIS gets the control output based on the estimated action value function. The described approach was evaluated in a simulation of a car-following scenario where the lead vehicle varies its velocity over time with a maximum episode duration of 80s. The controller was shown to be able to successfully drive the vehicle without failing after 68 trials. However, the reward function of the proposed approach by Dai et al. is only based on the spacing between the lead and the following vehicle. The reward function is the key to a successful reinforcement learning approach as it is the means by which the developer indicates the desirability of being in any given state. Therefore, the reward function needs to accurately capture the task to be performed and the manner in which it should be completed. For longitudinal control, the reward function should motivate the agent to adopt a safe and efficient driving strategy. For these reasons, a reward function with only one parameter such as inter-vehicle spacing may not be sufficient in real-time applications.

There are several works in which the use of multi-objective reward functions have been explored. For example, Desjardins & Chaib-Draa [23] used a multi-objective reward function based on time headway (distance in time from the lead vehicle) and time headway derivative. The agent was encouraged through the reward function to keep a 2s time headway to the lead vehicle, and the time headway derivative provided information regarding whether the vehicle is moving closer to or farther from the lead vehicle, and allowed it to adjust its driving strategy accordingly. Taking the time headway derivative into consideration in the reward function encourages the agent to choose actions which help it progress toward the desired state (ideal time headway). The authors used this reward function in a policy-gradient method for a Cooperative Adaptive Cruise Control (CACC) system. The neural network architecture chosen had two inputs, a single hidden layer of 20 neurons, and an output layer with 3 discrete actions (brake, accelerate, do nothing). In the learning process, an average of over 2.2 million iterations were obtained over ten learning simulations. The chosen method was shown to be efficient in CACC, providing average time headway errors of 0.039s in an emergency braking scenario. While the magnitude of the time headway errors remain small, it should be noted that the velocity profile of the subject vehicle showed oscillatory behaviour. This would make the system uncomfortable for the passengers as well as pose a potential safety risk. Potential solutions for this could include utilising continuous action values, the use of RNNs, or negative rewards for changes in acceleration to help smooth the velocity profile of the vehicle. Similarly, Sun [99] proposed a CACC system based on rewards from time headway and time headway derivative in a Q-learning algorithm. This approach was shown to reduce the learning time of the neural network. Over one hundred learning simulations, the best performing policy (the policy which obtained the highest reward) was chosen for evaluation. The algorithm was evaluated in a simulation of a stop-and-go environment in which the lead vehicle accelerated and decelerated periodically. The agent was shown to provide adequate performance in a platoon scenario. However, whilst such multi-objective reward functions are an improvement over single objective reward functions such as the one proposed by Dai et al. [107], this reward function does not consider passenger comfort which could lead to harsh accelerations or decelerations.

Huang et al. [108] presented a Parameterised Batch Actor-Critic (PBAC) reinforcement learning algorithm for longitudinal control of autonomous vehicles based on actor-critic algorithms. A multi-objective reward function was designed to reward the algorithm for tracking precision and drive smoothness. The method was validated by field experiments on various driving environments (e.g. flat, slippery, sloping, etc.) and the results suggested the method can track time-varying speeds more precisely than traditional Proportion-Integration (PI) or Kernel-based Least Square Policy Iteration (KLSPI) controllers trained with reinforcement learning [109], [110]. This was due to lower sensitivity to noise of speeds and accelerations. Moreover, smooth driving was achieved using the proposed method. The addition of driving smoothness in the reward function makes these systems more comfortable for passengers. However, the method was evaluated in an environment without adjacent vehicles or other obstacles. This allowed the authors to not consider safety parameters in the

reward function, which leaves the algorithm susceptible to crashes in environments with other vehicles present. Therefore, additional terms for safety would be required in the reward function to ensure safe behaviour of the autonomous vehicle.

One such reward function was proposed by Chae et al. [111], who proposed an autonomous braking system for collision avoidance based on a DQN approach. The reward function balances two conflicting objectives: avoiding collision and getting out of high risk situations. To speed up convergence, a replay memory was used to store a number of episodes of which some are chosen randomly to help train the network. Additionally, a 'trauma memory' of rare critical events (e.g. collision) was used to improve stability and make the agent more reliable. The system was evaluated in situations where the vehicle had to avoid collision with a pedestrian, using various Time-to-Collision (TTC) values with 10,000 tests for each TTC value. It was shown that for TTC values above 1.5s, collisions were avoided every time, whereas at 0.9s (lowest TTC value used) the collision rate was as high as 61.29%. Additionally, the system was evaluated in a test procedure specified by the Euro NCAP test protocol (CVFA and CVNA tests [112]) and the system passed these tests without collision. Therefore, the system was considered to exhibit desirable and consistent brake control behaviour. In addition, Chen et al. [100] presented a personalised ACC which can learn from human demonstration. The proposed algorithm is based on Q-learning with a reward function based on distance to the front vehicle, vehicle speed, and acceleration. The authors used a Q-learning algorithm based on a feedforward artificial neural network to estimate the Q-function and calculated the desired velocity, which is then converted to low-level control commands by a Proportional Integral Derivative (PID) controller. The neural network used to estimate the Q-function consists of an input layer with 5 nodes, a hidden layer with 3 nodes, and an output layer with 1 node which predicts the desired velocity. The performance of the system was evaluated based on comfort and driving smoothness in simulation with different velocities and desired inter-vehicle clearances. The system was shown to provide better performance when compared to traditional ACC approaches. Similarly, Zhao et al. [113] proposed a personalised ACC approach which considers safety, comfort, as well as personalised driving styles. The reward function considers the driver habits, passenger comfort, and safety in an effort to find a good tradeoff between safety and comfort. The proposed approach uses a Model-free Optimal Control (MFOC) algorithm based on an actor-critic neural network structure. By optimising the algorithm to drive in a more human-like fashion, the human driver is more likely to trust the system and continue using it. For this purpose, the network would also be capable of learning from the human driver when the cruise control feature was switched off to better tune its parameters and to adopt a driving strategy based on the owner's driving habits. The proposed algorithm was tested in a simulation under various environments and was shown to perform better than PID and Linear Quadratic Regulator (LQR) based controllers. For instance, in an emergency braking test scenario shown in Fig. 2, the MFOC maintained a safer clearance compared to PID, while the LQR failed the test by causing a rear end collision. However, while conforming to individual driving habits can be useful to ensure the user feels safe and comfortable in the car, strategies for mitigating the negative effects of learning bad driving habits should also be considered to ensure the long term reliability and safety of the system.

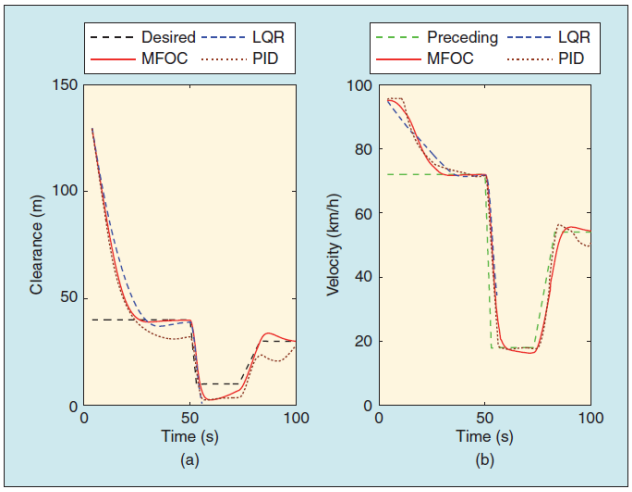

Fig. 2. MFOC Controller compared to PID and LQR controllers in an emergency braking scenario. (a) Clearance between the lead and follower vehicle. (b) Velocity profiles of the lead vehicle and the three controllers [113].

Reinforcement learning has been shown to be an effective approach for vehicle longitudinal control systems as shown by the discussion above. However, the main drawback for reinforcement learning is the time-intensive training [50], [114]. In contrast, supervised learning methods simplify the learning process with the use of prior knowledge of the supervisor, but lack the level of adaption that makes reinforcement learning attractive to complex decision-making systems such as autonomous driving. For these reasons, there are multiple examples in the literature that combine reinforcement and supervised learning to exploit the advantages of both approaches; reinforcement learning allows for self-adaptation in new and complex environments whilst the prior knowledge of supervised learning speeds up the learning process. For example, Zhao et al. [22], [115], [116] introduced a supervised reinforcement learning algorithm for an ACC system. By utilising actor-critic methods, the authors propose a novel supervised actor-critic (SAC) learning scheme, which is then implemented with feed-forward neural networks into a hierarchical acceleration controller. The proposed approach was evaluated in a simulation for an emergency braking scenario. The network was trained for emergency braking in dry conditions, whilst it was evaluated in both dry and wet road conditions and results were compared to the performance of a PID controller. The simulation results demonstrated that the SAC algorithm has superior performance compared to that of the PID controller as well as a supervised learning based controller (without reinforcement learning), and can adapt to changing road conditions. This shows the benefits of combining supervised learning with reinforcement learning to leverage the combined advantages of both methods.

TABLE II
A COMPARISON OF LONGITUDINAL CONTROL TECHNIQUES

Ref.	Learning Strategy	Network	Inputs	Outputs	Pros	Cons	Experiments
[107]	Fuzzy Reinforcement Learning	Feedforward network with 1 hidden layer	Relative distance, relative speed, previous control input	Throttle angle, brake torque	model-free, continuous action values	Single term reward function	Simulation
[23]	Reinforcement Learning	Feedforward network with 1 hidden layer	Time headway, headway derivative	Accelerate, brake, or no-op	Maintains a safe distance	Oscillatory acceleration behaviour, no term for comfort in reward function	Simulation
[108]	Reinforcement Learning	Actor-Critic Network with feedforward networks	Velocity, velocity tracking error, acceleration error, expected acceleration	Gas and brake commands	Learns from minimal training data	Noisy behaviour of the acceleration signal	Real world
[111]	Reinforcement Learning	Feedforward network with 5 hidden layers	Vehicle velocity, relative position of the pedestrian for past 5 time steps	Discretised deceleration actions	Reliably avoids collisions	Only considers collision avoidance with pedestrians, high rate of collision at low TTC	Simulation
[100]	Reinforcement Learning	Feedforward network with 1 hidden layer	Relative distance, relative velocity, relative acceleration (normalised)	Desired acceleration	Provides smooth driving styles, learns personal driving styles	No methods for preventing learning of bad habits from human drivers	Simulation
[113]	Reinforcement Learning	Actor Critic Network with feedforward networks	Relative distance, host velocity, relative velocity, host acceleration	Desired acceleration	Performs well in a variety of scenarios, safety and comfort considered, learns personal driving styles	Adapting unsafe driver habits could degrade safety	Simulation
[22]	Supervised Reinforcement Learning	Actor-Critic Network with feedforward networks	Relative distance, relative velocity	Desired acceleration	Pre-training by supervised learning accelerates learning process and helps guarantee convergence, performs well in critical scenarios	Requires supervision to converge, driving comfort not considered	Simulation

Pre-training the network via supervised learning helps reduce the training time of reinforcement learning and improves the convergence of the algorithm, both of which are common problems in reinforcement learning algorithms. Meanwhile, by exploring different actions through trial and error, reinforcement learning improves the performance beyond what supervised learning can provide. Also, the authors stated that using an actor-critic network architecture was beneficial as the evaluation of actions by the critic boosts the system's performance in critical scenarios such as emergency braking.

A summary of the longitudinal control methods can be seen in Table II. In contrast to lateral control systems, vision-based inputs are not generally used for longitudinal control. Instead sensor inputs from ranging sensors (e.g. RADAR, LIDAR) and host vehicle states are more commonly used. These lower dimensional inputs (e.g. time headway or relative distance) can then easily be used to define a reward function for reinforcement learning. The second major difference between lateral and longitudinal control algorithms is the choice of learning strategies. While lateral control techniques favour supervised learning techniques trained on labelled datasets, longitudinal control techniques favour reinforcement learning methods which learn through interaction with the environment. However, as seen in this section, the reward function in reinforcement learning needs to be carefully designed. Safety, performance, and comfort all need to be considered. Poorly designed reward functions result in poor performance or the model not converging. Another challenge with reinforcement learning algorithms is the trade-off between exploration and exploitation. During training, the agent must take random actions to explore the environment. However, to perform well in its task the agent should exploit its knowledge to find the optimal action. Example solutions for this are the ϵ-greedy exploration policies and the Upper Confidence Bound (UCB) algorithm. ϵ-greedy strategies choose a random action with a probability ϵ, which decreases overtime as the agent learns its environment. On the other hand, UCB encourages exploration in states with high uncertainty, whilst exploitation is encouraged in regions with high confidence. Therefore, intrinsic motivation is implemented in the system, encouraging the agent to learn about its environment, whilst exploitation can be taken advantage of in states which have already been explored adequately [51], [117]–[119]. Other approaches have sought to use supervised learning as a pre-training step to get the advantages of both reinforcement and supervised learning.

C. Simultaneous Lateral & Longitudinal Control Systems

The previous sections demonstrated that DNNs can be trained for either longitudinal or lateral control of a vehicle. However, for autonomous driving, the vehicle must be able to control both steering and acceleration simultaneously. In early works towards full vehicle control through deep learning, Xia et al. [120] introduced an autonomous

driving system based on Q-learning combined with learning from the experience of a professional driver. The reward value of the professional driver's strategy and the Q-value learned through the Q-learning method were combined in the pre-training phase to improve the speed of convergence during training. A filtered experience replay stores a limited number of episodes and allows elimination of poor experimental rounds from memory, improving convergence on a control strategy. The proposed Deep Q-learning with filtered experiences (DQFE) approach was compared to a naive neural fitted Q-iteration (NFQ) [121] algorithm without pre-training by an experienced driver. During training, it was shown that the DQFE approach reduced the training time by 71.2% for the 300 training episodes. Moreover, during 50 tests on a competition track, the proposed approach completed the track 49 times, compared to only 33 with NFQ. Additionally, DQFE performed better in terms of mean distance from centre of the track. Therefore, the addition of filtered experience replay improved the speed of convergence as well as performance of the algorithm. Comparing two neural networks for lane keeping systems, Sallab et al. [122] investigated the effects of discretised and continuous actions. Two approaches, DQN and a Deep Deterministic Actor Critic (DDAC) algorithm, were evaluated in a TORCS simulator [123]. In the two networks developed by the authors, the DQN could only output discretised values (steer, gear, brake, and acceleration), while the DDAC supports continuous action values. The DDAC consisted of two networks; an Actor Network which is a neural network responsible for taking actions based on perceived states and the Critic Network which criticises the value of the action taken. The experimental results showed that the DQN algorithm suffered in performance due to the fact that it cannot support continuous actions or state spaces. The DQN algorithm is suitable for continuous (input) states, however it still requires discrete actions since it finds the action that maximises the action-value function. This would require an iterative process at every time step for continuous action spaces [124]. As shown in Fig. 3, the ability to support continuous action values allowed the DDAC algorithm to follow curved tracks more smoothly and stay closer to the centre of the lane when compared to the DQN algorithm, thereby producing better performance for lane keeping.

Vision based vehicle control using CNNs has also been researched. For instance, Zhang et al. [125] proposed a supervised learning method, SafeDAgger, for training a CNN to drive in a TORCS simulation. The proposed method is based on the Dataset Aggregation (DAgger) imitation learning algorithm [54]. In DAgger, the agent first learns a primary policy through traditional supervised learning, with the training set generated by a reference policy. Then, the algorithm iteratively generates new training examples through the learned policies, which are then labelled by the reference policy. The new expanded dataset can then be used to update the learned policy through supervised learning. This has the advantage that states which were not reached in the initial training set can be covered in the new extended training set. The primary policy is then iteratively fine-tuned using the new training set. Zhang et al. proposed an extension to this method,

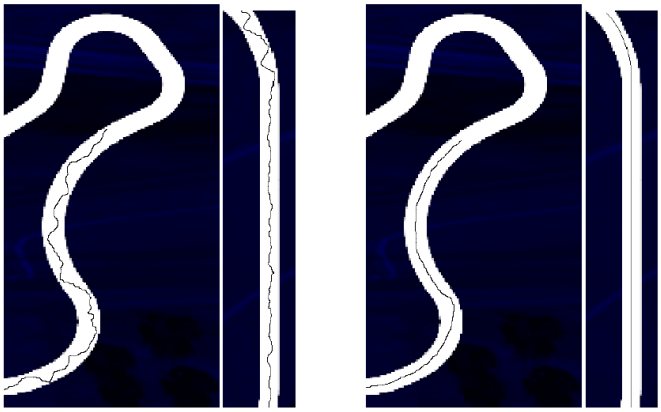

Fig. 3. The lane keeping performance of (a) the DQN with discretised outputs and (b) DDAC with continuous output values [122].

called SafeDAgger, where the system estimates (in any given state) whether the primary policy is likely to deviate from the reference policy. If the primary policy is likely to deviate by more than a specified threshold, the reference policy is used to drive the vehicle instead. The safety policy is estimated by a fully connected network where the input is the last convolutional layer's activation. The authors used this method to train a CNN to predict a continuous steering wheel angle and a binary decision for braking (brake or do not brake). The authors then evaluated supervised learning, DAgger, and SafeDAgger by driving them on three test tracks, with up to three laps on each track. Out of the three algorithms evaluated, SafeDAgger was found to perform best in terms of the number of completed laps, number of collisions, and mean squared error of steering angles. In another work, Pan et al. [126] used DAgger-like imitation learning to learn to drive at high speeds autonomously, with continuous actions for both steering and acceleration. The reference policy for the dataset was obtained from a model predictive controller operated using expensive high resolution sensors, which the CNN then learned to imitate using only low cost camera sensors for observations. The technique was first tested in Robot Operation System (ROS) Gazebo [127] simulations, followed by a real-world 30m long dirt track with a 1/5-scale vehicle. The sub-scale vehicle successfully learned to drive at speeds up to 7.5m/s around the track. Instead of using direct vision for control, Wang et al. [128] demonstrated that DAgger can be used to train an object-centric policy, which uses salient objects in the image (e.g. vehicles, pedestrians) to output a control action. The trained control policy was tested in Grand Theft Auto V simulation, with a discrete control action (left, straight, right, fast, slow stop) which was then translated to a continuous control with a PID controller. The test results demonstrated improved performance with the object-centric policy compared to models without attention or those based on heuristic object selection. Vision based techniques have also been used to mitigate collisions by Porav & Newman [129], who built on the previous work by Chae et al. [111] by using a deep reinforcement learning algorithm for collision mitigation which can provide continuous control actions for

both velocity and steering. The system uses a Variational AutoEncoder (VAE) coupled with an RNN to predict the movement of obstacles and learns a control policy with Deep Deterministic Policy Gradient (DDPG) to mitigate collisions in low TTC scenarios. The network used a semantically segmented image to predict continuous steering and deceleration actions. The proposed technique shows improvement over braking-only policies for TTC values between 0.5 and 1.5s, and up to 60% reduction in collision rates.

Inverse Reinforcement Learning (IRL) approaches have also been investigated in the context of control systems as a way to overcome the difficulty of defining an optimal reward function. IRL is a subset of reinforcement learning, in which the reward function is not specified, but the agent attempts to learn it from an expert's demonstrations. In IRL, the agent assumes that the expert is completing the task by following an unknown reward function. It then estimates a reward function in which the demonstrators' trajectory is the most likely one. This has the advantage that instead of requiring the developer to explicitly specify a reward function, they simply have to demonstrate the intended behaviour. This can be advantageous since in large and complex tasks, defining an adequate reward function to provide optimal agent behaviour can be both difficult and time consuming [130]. IRL approaches have been shown to not only reduce the amount of time required for design and optimisation, but also improve the system performance by creating more robust reward functions. Abbeel & Ng [131] showed that when IRL was applied to a problem where the agent learned by observing an expert, the agent performed as well as the expert when evaluated with respect to the reward function used by the expert, even if the reward function derived from observations was not the expert's true reward function. Moreover, it was shown that in a simplistic highway driving scenario with 5 different actions for lane selection available to the agent and multiple driving styles demonstrated, the IRL algorithm successfully learned to mimic the demonstrated driving behaviours. Further, Silver et al. [21] used an IRL algorithm based on Maximum Margin Planning [132] which was shown to be effective in a demonstration of an autonomous vehicle in unstructured terrain. The vehicle was shown to perform better than an agent based on traditional reinforcement learning with a hand-tuned reward function. Additionally, the IRL approach was shown to require significantly less time to design and optimise compared to the reinforcement learning agent. Kuderer et al. [20] proposed a vehicle controller that can learn individual driving styles from demonstration using IRL. The algorithm assumes that the demonstrator is driving in a way to maximise an unknown reward function. From this, the learning model estimates the weights in a linear reward function based on 9 features for driving. Initially, the weights were equally set and were then updated based on demonstrations of 8 minutes per driver. After finding the driving policy, the chosen trajectories were compared to those observed from human drivers. The system was shown to learn drivers' personal driving styles from minimal training data and performed adequately in simulated testing.

Building on the IRL approaches, Wulfmeier et al. [133] proposed an IRL approach for deep learning. The proposed algorithm is based on the Maximum Entropy [134] model for a trajectory planner, and uses CNNs to infer the reward functions from expert demonstration. The approach was trained on a dataset collected over the course of one year with a total of 120km of driving a modified golfcart on walkways and cycle lanes. The input to the network was the LIDAR point cloud map, which was represented on a discretised grid map. The output of the network was a discrete set of actions. The proposed approach was demonstrated to work better than a manually constructed cost function. Moreover, the learned algorithm was shown to be more robust to sensor noise. This shows that the use of DNNs in an IRL algorithm for trajectory planning was beneficial overall. Therefore IRL techniques could be considered as a potential way to overcome the difficulties of designing an optimal reward function for driving.

However, there are some challenges for IRL approaches in practical applications. Firstly, there is no guarantee of optimality of the demonstrations. For example, in a driving demonstration, no human driver can carry out the driving tasks optimally every time. Therefore, the training data will include suboptimal demonstrations which will affect the final reward function constructed. There are some solutions to minimising the effect of suboptimal demonstrations; using multiple trajectories and averaging over multiple sets to find a reward function or removing the assumption of global optimality [135]. Secondly, reward ambiguity can lead to further problems in IRL approaches. Given expert demonstrations of driving strategies, there can be multiple reward functions that explain the expert's behaviour. Therefore, an effective IRL algorithm must find a reward function that considers the expert's trajectory optimal and rejects other possible trajectories. Thirdly, the reward function derived through IRL methods may not be safe, as noted by Abbeel et al. [136], who used IRL to operate an autonomous helicopter and had to manually tune the reward function for safety. Therefore, hand tuning of the derived reward function may be required to ensure safe behaviour. Lastly, the computational burden of IRL methods can be heavy since they often require iteratively solving reinforcement learning problems with each new reward function derived [130]. Nevertheless, in tasks where an adequately accurate reward function cannot be easily defined, IRL approaches can provide an effective solution.

While the previously mentioned works in this section demonstrate that a DNN can be trained to drive a vehicle, training a vehicle to simply follow a road or keep in its lane without any outside context is not sufficient for deploying fully autonomous vehicles. Humans drive vehicles with the goal of arriving at our target destination, and learning to drive from camera images to imitate human driving behaviour is not enough to understand the full context behind the human driver's action. For instance, it has been reported [83], that upon reaching a fork in the road end-to-end driving techniques tend to oscillate between the two possible driving directions. Not only is this impractical if our goal is to continue in the left direction, but can result in unsafe behaviour where the DNN oscillates between left and right but never picking either direction. Aiming to provide autonomous vehicles with contextual awareness, Hecker et al. [137] collected a data

set with a 360-degree view from 8 cameras and a driver following a route plan. This data set was then used to train a DNN to predict steering wheel angle and velocities from example images and route plans in the data set. Qualitative testing was done to evaluate learning on instances from the data set, suggesting the model was learning to imitate the human driver, but no live testing was completed to validate performance. With a similar aim, Codevilla et al. [138] trained a supervised learning algorithm, which uses both images and a high-level navigational command for its driving policy. The network was trained through end-to-end supervised learning, conditioned by a high-level command which could be follow road, go straight, turn left, or turn right. The authors tested two network architectures which could take the navigational command into account; one where the command was an additional input to the network, and one where the network branched at the end into multiple sub-modules (feedforward layers), one for each possible command. The authors noted that the latter architecture performed better. The resulting network was initially tested in CARLA [139] simulation, followed by real-world testing on a 1/5-scale car. The resulting policy successfully learned to turn the correct way at intersections as commanded. The authors noted that data augmentation and noise injection during training was key to learning a robust control policy. This method was further extended in [140], by using an extra module for velocity prediction, which helps the network in some situations, such as when the vehicle is stopped at a traffic light, to predict the expected vehicle velocity from visual cues and prevent it from getting stuck when the vehicle comes to a full stop. Further improvements to the model were a deeper network architecture and a larger training set, which reduced the variance in training. A slightly different approach was explored using reinforcement learning by Paxton et al. [141] where the high-level command is provided by another DNN responsible for decision making. The system consisted a DDPG network for low-level control and a DQN for a stochastic high level policy subject to linear temporal logic constraints. The aim of the vehicle was to navigate a busy intersection, where some lanes had stopped vehicles so that the host vehicle had to successfully change lanes as well. The system was tested in 100 simulated intersections with and without stopped cars ahead, for a total of 200 tests. Without stopped cars the agent succeeded every time, whereas with stopped cars ahead, 3 collisions occurred.

Moving away from end-to-end approaches, researchers at Waymo recently presented ChauffeurNet [142]. ChauffeurNet uses mid-to-mid learning to learn a driving policy, where the input is a pre-processed top-down view of the surrounding environment which represents useful features such as roadmap, traffic lights, a route plan to follow, dynamic objects, and past agent poses. The agent then processes these inputs through an RNN to provide a heading, speed, and waypoint, which are then achieved through a low-level controller. This had the advantage that pre-processed inputs could be obtained either from simulation or real-world data, which makes transferring driving policies from simulation to the real world easier [143], [144]. Furthermore, synthesising perturbations to model recoveries from incorrect lane positions or even scenarios such as collisions or driving off-road provides the model with robustness to errors and allows the model to learn to avoid such scenarios.

An overview of full vehicle control approaches can be seen in Table III. Unlike previous sections, a variety of learning strategies have been utilised here, however supervised learning is still the preferred approach. An important note on the works where full vehicle control via neural networks is researched, is that robust and high performing models still seem out of reach. For instance, techniques which implement full vehicle control tend to have poorer performance on steering than techniques which only consider steering. This is explained by the significant increase in the complexity of the task which the neural network is trained to perform. For this reason, several of the works summarised in this section have been trained and evaluated in simplified simulated environments. While full vehicle control should be the end goal of autonomous vehicle control techniques, current approaches have yet to achieve adequate performance in complex and dynamic environments. Therefore future research is required to further improve the control performance of neural network-driven autonomous vehicles.

IV. Challenges

The previous section discussed various examples of deep learning applied to vehicle controller design. While this shows that there is a significant amount of interest in the research of such systems, they are still far from ready for commercial application. There remains a number of challenges that must be overcome before learned autonomous vehicle technology is ready for widespread commercial use. This section is dedicated to discussing the technological challenges for deep learning based control of autonomous vehicles. It is worth remembering that besides these technological challenges, issues such as user acceptance, cost efficiency, machine ethics for artificial intelligence technologies, and lack of legislation/regulation for autonomous vehicles must also be addressed. However, the aim of this manuscript is to focus on deep learning based autonomous vehicle control methods and their technical challenges, therefore general and non-technological challenges for autonomous vehicles are out of the scope of this manuscript, for further reading on these topics, see [145]–[150].

A. Computation

The major drawback for deep learning methods is the large amount of data and time required for adequate training, especially for reinforcement learning methods. This can lead to long training periods which can cause delays and additional cost in the design of an autonomous vehicle. The common solution to reduce training data requirements or the time required for training is to combine reinforcement learning with supervised learning, which helps reduce the training time whilst still providing good adaptability. Nevertheless, for a fully autonomous vehicle, the amount of training data required to build a reliable and robust system can be vast. It is challenging to train a vehicle to drive in all possible scenarios that it could encounter in the real world due to

TABLE III
A COMPARISON OF FULL VEHICLE CONTROL TECHNIQUES

Ref.	Learning Strategy	Network	Inputs	Outputs	Pros	Cons	Experiments
[120]	Supervised Reinforcement Learning	Feedforward network with 2 hidden layers	Not mentioned	Steering, acceleration, braking	Fast training	Unstable (Can steer off the road)	Simulation
[122]	Reinforcement Learning	Fully connected / Actor-Critic Network with feedforward networks	Position in lane, velocity	Steering, gear, brake, and acceleration values (discretised for DQN)	Continuous policy provides smooth steering	Simple simulation environment	Simulation
[125]	Supervised Learning	CNN / Feedforward	Simulated camera image	Steering angle, binary braking decision	Estimates safety of the policy in any given state, DAgger provides robustness to compounding errors	Simple simulation environment, simplified longitudinal output	Simulation
[126]	Supervised Learning	CNN	Camera image	Steering and throttle	High speed driving, Learns to drive on low cost cameras, Robustness of DAgger to compounding errors	Trained only for elliptical race tracks with no other vehicles, Requires iteratively building the dataset with the reference policy	Real world (sub-scale vehicle) & Simulation
[128]	Supervised Learning	CNN	Image	9 discrete actions for motion	Object-centric policy provides attention to important objects	Highly simplified action space	Simulation
[129]	Reinforcement Learning	VAE-RNN	Semantically segmented image	Steering, acceleration	Improves collision rates over braking only policies	Only considers imminent collision scenarios	Simulation
[133]	Inverse Reinforcement Learning	CNN	LIDAR point clouds on a grid map	Discrete motions	Robust to noise, avoids handcrafting of cost function	Increased computation burden of IRL, no guarantee of cost function optimality	No live testing
[137]	Supervised Learning	CNN	360-degree view camera image, route plan	Steering angle, velocity	Takes route plan into account	Lack of live testing	No live testing, tested on data set image examples only
[138]	Supervised Learning	CNN	Camera image, navigational command	Steering angle, acceleration	Takes navigational commands into account, generalises to new environments	Occasionally fails to take correct turn on first attempt	Real (sub-scale vehicle) & Simulation
[141]	Reinforcement Learning	Feedforward network with 1 hidden layer	Host vehicle states, set of features for each nearby vehicle, vehicle position and priority in intersection	steering angle rate, acceleration	Considers decision making provided by another DNN	Large number inputs which would be difficult to extract in reality, Not collision free	Simulation
[142]	Supervised Learning	CNN-RNN	Pre-processed top-down image of surroundings	Heading, velocity, waypoint	Ease of transfer from simulation to real world, robust to deviations from trajectory	Can output waypoints which make turns infeasible, can be over aggressive with other vehicles in new scenarios	Real world & Simulation

the huge quantity of data that needs to be collected. There are several companies researching autonomous driving using machine learning and collaborating and sharing data would be the fastest route to move from experimental systems to commercial ones. However, this is unlikely as companies researching autonomous vehicles are not willing to share their resources due to fear of diluting their competitive advantage [151]. However, while increasing the amount of available data is useful to learn more complex behaviours, using larger data sets brings its own challenges, such as ensuring diversity of the data. If the amount of data used for training the model is increased, without ensuring variety in the data set, the risk of overfitting to the data set increases. For instance, Codevilla et al. [140] compared 4 driving models trained with 2, 10, 50, and 100 hours of data, and it was shown that the model trained with 10 hours of driving data performed best in most scenarios. This is due to many of the instances in the training set being very similar, captured in typical driving conditions. As the data set size increases, rare driving scenarios (where the model is more likely to fail) are encountered increasingly rarely during training. Therefore, when generating large data sets, diversity in the data set must be ensured.

Further computational complexity is caused by the continuous states and actions in which the agent has to operate. As stated in the previous section, continuous action values are necessary for a deployable vehicle control system to have adequate performance. However, as the number of dimensions grows, the computational complexity grows exponentially [152], this is known as the Curse of Dimensionality [153]. In the high-dimensional problems of vehicle

control, this has a significant effect on the computational complexity of any solution. Although discretisation of the system can reduce the complexity, as seen in previous examples, this can lead to degradation in system performance. Other solutions include using multiple learners to reduce learning time [154], [155], evolution strategies which are highly parallelisable [156], or removing unnecessary data from the training and system input data [157].

Overall, the high computational burden of DNNs is a challenge to not only the development and training of the networks but also the deployment of such systems in vehicles. The high computational overhead of the deep learning algorithms will require high computing capabilities on-board, driving up the system cost and power requirements, which must be kept in mind during the system design.

B. Architectures

Another challenge with deep learning is selecting the architecture of the neural networks. There are no clear guidelines for 'good' neural network architecture for a given task. For instance, in terms of size and number of layers, it has been shown that too few neurons will lead to a system with poor performance. However, too many neurons may overfit to the training data and therefore not generalise well. Also, given that additional neurons will lead to increased computational complexity, finding an optimal number of neurons would be of great benefit to deep learning methods [158], [159]. Other parameters can also have an effect on the performance, training, and convergence of the system. The fundamental architecture, training method, learning rate, loss function, batch size etc., all need to be decided upon and defined, which affect the performance of the agent. However, there are few methods for choosing these parameters, and often trial-and-error and heuristics are the only viable options for optimising each parameter due to the complexity of DNNs [49]. This is generally achieved by choosing a range of values for the hyperparameters in the neural network, and finding the best performing values. However, using such trial-and-error methods for exploring the hyperparameter space can be slow, given the amount of computation required for each training run.

Solutions to this challenge currently being researched include computerised ways of finding optimal values for these parameters, either by trialling across a range or using model-based methods to converge on the best values. There are several methods for changing the parameters over the chosen range, such as Coordinate Descent [160], Grid Search [160], [161], and Random Search [162]. Coordinate Descent keeps all hyperparameters except one fixed, and finds the best value for one parameter at a time. Grid Search optimises every parameter simultaneously, including the cross-product of all intervals. However, this vastly increases the computational expense by requiring a large number of neural network models to be trained and therefore is only suitable when the models can be trained quickly. Random Search often finds a good set of parameters faster than a Grid Search by sampling the chosen interval randomly [162], [163]. However, this has the disadvantage that the parameter space is often not covered completely, and some sample points can be very close to each other. These disadvantages can be solved by using quasi-random sequences [164]. Alternatively, one can use model-based hyperparameter optimisation methods, such as Bayesian optimisation or tree-structured Parzen estimators, which tend to yield better results but are more time intensive [164]–[167]. Other proposed approaches focus on automated hyperparameter tuning by eliminating undesirable regions of the hyperparameter search space in order to converge to optimal values [168], [169]. Recent research has also explored neural architecture search methods which take hardware efficiency into account by incorporating the hardware feedback into the learning signal [170]–[173]. This has resulted in neural network architectures which are specialised for specific hardware platforms, and demonstrate a hardware efficiency benefit over non-specialised architectures. Such methods could also be extended to find efficient network architectures for vehicle on-board hardware platforms. It should be noted that automated neural architecture search is an active area of research, for further discussion on this topic we refer the reader to the survey by Elsken et al. [174].

While architecture selection is a general problem for many deep learning applications, a complex task such as autonomous driving also brings its own challenges. Currently, most end-to-end driving systems have been limited to smaller networks. This is due to the relatively small datasets used, which would cause deeper networks to overfit to the training data. However, as noted in [140], when large amounts of data are available, deeper architectures can reduce both bias and variance in training, resulting in more robust control policies. Further thought should be given to architectures specifically designed for autonomous driving, such as the conditional imitation learning model [138], where the network included a different final network layer for each high-level command used for driving. These challenges translate to mid-to-mid approaches as well, as the selection of high-level features represented in the input to the network must be chosen carefully. Future works investigating specialised network architectures for autonomous driving can therefore be expected.

C. Goal Specification

Adequate goal specification is a challenge specific to reinforcement learning methods. One of the advantages of reinforcement learning is that the behaviour of the agent does not need to be specified implicitly as it would be in rule-based systems. Only the reward function, which can often be easier to define than the value function, and the control action (e.g. steering, acceleration, braking) need to be defined. However, the goal of reinforcement learning is to maximise the long term accumulated reward as defined by the reward function. Therefore, the desired behaviour of the agent must be accurately captured by the reward function, otherwise unexpected and undesired behaviour might occur. For instance, instead of using binary rewards for successful or unsuccessful completion of tasks, intermediate rewards can be used to guide the agent towards desired behaviour, this process in known as reward shaping [175], [176]. For example,

Desjardins & Chaib-Draa [23] used the time headway derivative to reward the agent for actions that helped it move towards the ideal time headway state. Furthermore, for a complex task such as driving, a multi-objective reward function needs to consider different objectives which may conflict with each other. For example, for driving, these objectives may include maintaining a safe distance from other vehicles, staying close to the centre of the lane, avoiding pedestrians, not changing lanes too often, maintaining desired velocity, and avoiding harsh accelerations/braking. Hence, the reward function should not only consider all factors that affect the agent's behaviour, but also the weight of these factors.

A further challenge for agents which control both lateral & longitudinal actions is the difficulty of defining a reward function when the agent must be able to perform multiple actions (steering, braking, and acceleration). In reinforcement learning, the agent uses the feedback from the reward function to improve its own performance. However, when the agent is carrying out multiple actions, it may not be clear which of the actions resulted in the given reward. For example, if the vehicle steers away from the road, the acceleration may not be at fault but a negative reward signal is sent to the agent. One solution to this is a Hybrid Reward Architecture [177], where the system uses a decomposed reward function and learns a separate value function for each component reward function. Alternatively, Shalev-Shwartz et al. [178] proposed a solution in which the reward function is decomposed into a high level decision making system, through which the agent learns to drive safely and make strategic decisions (e.g. which cars to overtake or give way to), and a low level reward function which helps the agent learn an optimal policy for different actions (e.g. overtaking, merging, decelerating etc.).

The developer should also take care that the agent does not exploit the reward function in unexpected ways, resulting in unintended behaviour. This effect is also known as reward hacking. Reward hacking occurs when the agent finds an unanticipated way of exploiting the reward function to gain large rewards in a way which goes against the developers' defined objective(s) for the agent. For example, a robot used in ball paddling with a reward function based on the distance between the ball and the desired highest point, may attempt to move the racket up and keep the ball resting on it [152]. Potential solutions to avoid reward hacking were proposed by Amodei, et al. [179] in the form of adversarial reward functions, model look-ahead, reward capping, multiple reward functions, and trip wires. Adversarial reward functions utilise a reward function which is its own agent, similar to generative adversarial networks. The reward function agent can then explore the environment, making it more robust to reward hacking. It could, for example, try to find instances where the system claims a high reward from its actions, while a human would label it as a low reward. On the other hand, model look-ahead gives a reward based on anticipated future states, instead of the present one. Reward capping is a simple solution to reward hacking, where a maximum value is imposed on the reward function, thereby preventing unexpected high reward scenarios. Multiple reward functions can also increase robustness to reward hacking, since multiple rewards can be more difficult to hack than a single one. Finally, trip wires are deliberately placed vulnerabilities in the system, where reward hacking is most likely to occur. These vulnerabilities are then monitored to alert the system if the agent is attempting to exploit its reward function. Another approach to solving these challenges in goal specification is using inverse reinforcement learning to extract a reward function from expert demonstrations of the task [180]–[183].

D. Adaptability & Generalisation

Another challenge for learned control systems is dealing with different environments with a scalable approach. For example, a driving strategy that is successful in an urban environment may not be optimal on a highway, since they are very different environments with different traffic flow patterns and safety issues. Similar issues arise with changing weather conditions, seasons, climates etc. A neural network's ability to use what it has learned from previous experiences to operate in a completely new environment is referred to as generalisation. However, the problem with generalisation is that even if the system demonstrates good generalisation in one new environment, there is no guarantee it will generalise to other possible environments. Moreover, considering the complex operating environment of a vehicle, it is not possible to test the system in all scenarios. Therefore, building a deep learning system capable of generalising to such a vast variety of situations, as well as validating its generalisation capability poses major challenges. This is a challenge that must be overcome for deep learning driven autonomous vehicles to be deployable in the real world, as the vehicles must be able to cope with the various different environments it will be used in.

Generally, to avoid poor generalisation in DNNs the training must be stopped before the DNN starts to overfit to the training data. Overfitting refers to creating a model that fits the training data too well, losing its ability to generalise to new data. Overfitting occurs when the network is trained with either insufficient amounts of training data or too many training episodes on the same training data. This results in the neural network memorising the training data, thereby losing generalisation. Unfortunately, there are no known methods of choosing the optimal stopping point in order to avoid overfitting [184]. However, it is possible to get some indication of the network's generalisation capability by having three different data sets: training, validation, and test sets. The training and validation sets are used during training, but only the training set results are used to update the network weights [185]. The purpose of the validation set is to minimise overfitting, by monitoring the error in the validation data set. In this way, it will be ensured that changes which reduce the error on the training set also reduce the error on the validation set, thereby avoiding overfitting. If the accuracy of the validation set starts to decrease over the training iterations, then the network is starting to overfit and training should be stopped. In addition to stopping overfitting, a validation set can also be used to compare different network architectures (e.g. comparing two different networks with different numbers

of hidden layers) to provide a measure of generalisation. Nevertheless, utilising the validation set simultaneously in the selection of the network and to terminate training can result in overfitting to the validation set. Therefore an additional independent set, known as testing set, is required for the evaluation of the network performance [186]. The testing set is only used to test the final network to confirm its performance and generalisation capabilities. The testing set must provide an unbiased evaluation of the network's generalisation [185]. Therefore, it is crucial that the test set is not used to choose between different networks or network architectures.

There are also techniques available for DNNs which aim to reduce the test error, although often at the cost of increased training error, known as regularisation techniques [48]. The basis of regularisation techniques is to introduce some constraints on the deep learning model, which either introduce prior knowledge into the model or promote simpler models in order to achieve better generalisation capability. There are a variety of regularisation techniques available to choose from. For instance, L1 and L2 regularisation techniques introduce a constraint on the model by including an additional term in the cost function of the learning model, which makes the network prefer smaller weights. The smaller weights in the network reduce the effect of individual inputs on its behaviour, which means that the effect of local noise is reduced and the network is more likely to learn trends across the whole data set [49], [187]. Similarly, imposing constraints on the network weights through weight clipping has also been shown to improve robustness [188], [189]. Another popular regularisation technique is dropout, which drops out some randomly selected neurons from training and only updates the remaining weights for the given training example. At each weight update, a different set of neurons is omitted, thereby preventing complex co-adaptions between neurons. This helps each neuron learn features which are important for the given task and therefore helps reduce overfitting [190], [191].

E. Verification & Validation

The testing of the system needs to be rigorous to validate the performance and safety of the system. However, the problem is that real-world testing can be expensive in terms of time, labour, and finances. Indeed, full-scale vehicle studies with multiple vehicles have typically been achieved through collaboration of government research projects with automotive manufacturers, such as Demo '97 [192]–[194] or Demo 2000 [195]. Alternatively, simulation studies can reduce the amount field testing required, and can be used as a first step for performance and safety evaluation. Simulation studies are significantly cheaper, faster, more flexible, and can be used to set up situations not easily achieved in real life (e.g. crashes). Indeed, with the increasing accuracy and speed of simulation tools, simulation has become an increasingly dominant method of study in this field [196].

While simulation has multiple advantages, the model errors must be kept in consideration throughout the verification and validation process. This is especially critical for training, as training an agent in an imprecise model will result in a system that will not transfer to the real-world without significant modifications [152], [197]. Complex mechanical interactions, such as contacts and friction, are often difficult to model accurately. These small variations between the simulation model and the real-world can have drastic consequences on the system behaviour in the real world. In other words, the problem is the agent overfitting its policies to the simulation environment, and not transferring well to a real-world environment. For a system that can be evaluated and used in the real-world, training, as well as testing, in both simulation and field tests would be required [198]. The large number of trials required for reinforcement learning algorithms to converge, makes them susceptible to this issue where simulation is used for training. However, recent studies in robot manipulation have shown effective transfer of learned policies from simulation to the real world [199]–[202].

Validation of the model and simulation environment alone is not enough for autonomous vehicles, as the influence of the training data can be equal to that of the algorithm itself [203]. Therefore, there should be emphasis on validating the quality of the training set as well. Ensuring that the data set represents the desired operational environment adequately, and covers the potential states is important. For instance, data sets that are biased towards a certain action (e.g. turn left) or scenario (e.g. driving in daytime) can introduce harmful biases into the learning model. Therefore, data sets should be validated to understand if they contain potentially harmful biases or patterns that could lead to undesirable behaviour of the learned control policy [56].

F. Safety

In a safety-critical system, such as vehicle operation, a serious malfunction or failure could result in death or serious harm to people or property. Therefore, the safety of road users must be ensured before such systems are deployed commercially. However, ensuring functional safety in deep learning systems can be challenging. As the neural networks become more complex, the solutions they provide and how they come to those solutions becomes increasingly difficult to interpret [204]. This is known as the black box problem. The opacity of these solutions is an obstacle to their implementation in safety-critical applications; while it is possible to show that these systems provide good performance in our validation environment, it is impossible to test these systems in all the possible environments they would encounter in the real world. Therefore, if we do not understand the way in which the system makes its decisions, ensuring it does not make unsafe decisions in new environments becomes increasingly difficult. It becomes even more challenging in online learning methods, since they change their policies during operation and therefore could potentially shift from safe policies to unsafe policies over time [205]–[209].

Any autonomous vehicle system not only needs to drive safely, but it also needs to be capable of reacting in a safe manner to other vehicles or pedestrians acting unpredictably. It can be difficult to guarantee the safety for any vehicle controller if, for example, another driver is acting recklessly or

TABLE IV
A SUMMARY OF RESEARCH CHALLENGES

Challenge	Sub-challenges	Potential Solutions
Computation	• Computation requirements for deep learning • Large data sets for supervised learning • Curse of dimensionality in high dimensional problems • Simulation requirements for sample inefficient techniques	• Scalable model architectures • Sharing data sets • Improving sample efficiency in reinforcement learning • Parallelisable architectures for training
Architectures	• Lack of clear rules for network architectures • Reliance on heuristics and trial-and-error	• Automated neural architecture search methods • Specialised architectures for autonomous driving
Goal Specification	• Well designed reward functions for complex tasks • Multi-objective reward functions • Reward hacking	• Reward shaping • Inverse reinforcement learning • Hybrid reward architectures
Adaptability & Generalisation	• Wide variety of the operational environment • Overfitting to training data/environment	• Representative data sets and/or training environments • Effective use of regularisation techniques
Verification & Validation	• Inability to test in all possible scenarios • High cost of field testing • Inaccuracies in simulation • Biases and gaps in data sets	• High fidelity simulations • Effective simulation to real world transfer • Validation of data set coverage
Safety	• Complexity and opaqueness of DNNs • Safe training in the real world • Adversarial attacks	• Research into interpretability of DNNs • Fail safes and virtual safety cages • Human oversight • Improving model robustness to perturbations

a previously unseen pedestrian runs onto the road. Therefore, it would be useful to include unsafe and aggressive driving behaviours of other vehicles into the training data of the vehicle controller to enable it to learn how to deal with such situations. One option to improve reliability and safety in such situations is utilising a trauma memory [111] where rare negative events (e.g. collisions) are stored. These are then used in training to persistently remind the agent of these events and ensure it maintains safe behaviour.

Also, safety must be maintained during any training or testing in the real world. For instance, during early training of a reinforcement learning agent, the agent is more likely to use exploration than exploitation of past experiences, which means the agent will effectively be learning through trial and error. Therefore, care must be taken to ensure the exploration happens in a safe manner. This is especially true in any environment including other road users or pedestrians, since inappropriate actions chosen due to exploration could have disastrous results. Exploration poses safety challenges as the agent is encouraged to take random actions, which can lead to catastrophic events if not considered beforehand [210]–[213]. Potential solutions include the use of demonstrations such as in IRL to provide examples of safe behaviour which could be used as a baseline policy, simulated exploration where exploration happens in a simulated environment, bounded exploration which limits exploration in state spaces which are considered unsafe, and human oversight although this is limited in scalability and not feasible in some real-time systems. The same holds true for any testing and evaluation of the system; until the system has been deemed to perform adequately and in a safe manner, all necessary precautions must be taken to ensure safety [179].

An approach for ensuring functional safety for deep learning based autonomous vehicles is suggested by Shalev-Shwartz et al. [178]. In the proposed system architecture, the policy function is decomposed into a learnable part and a non-learnable part. The learnable part is responsible for the comfort of driving and for making strategic decisions (e.g. which cars to overtake or give way to). This policy is learned from experience by maximising an expected reward from the reward function. On the other hand, the non-learnable policy is responsible for the safety by minimising a cost function with hard constraints (e.g. the vehicle is not allowed within a specified distance of other vehicles' trajectories) to ensure functional safety. Alternatively, Xiong et al. [214] suggested a control structure which combines reinforcement learning based control with safety based control and path tracking. The aim is to combine a traditional control method with a reinforcement method to take advantage of the superior performance of deep learning systems whilst ensuring safety through traditional control theory. The path tracking element is included to ensure the vehicle stays on (or as close as is safe to) the centre of the lane. The reinforcement learning approach is based on the DDPG algorithm. Also, the safety based controller uses an Artificial Potential Field method [215] which models any obstacles with a repulsive force to steer the vehicle away from them. The final steering policy is then found by the weighted summation of the three models. The system was shown to keep a safe distance in a simulated environment where the vehicle had to drive along a curve with other vehicles nearby.

Furthermore, malicious inputs to deep learning systems have to be considered. It has been shown that visual classification DNN systems are vulnerable to adversarial examples, which are perturbed images that cause the DNNs to misclassify them with high confidence [216]–[219], including misclassification of traffic signs [220]. DNNs have been shown to be vulnerable to printed adversarial examples in the real world [221] and even to 3D-printed physical adversarial examples [222], which suggests they are a threat to DNN applications in the real world. Moreover, the image modification of the adversarial examples have been shown to be subtle enough that a human

eye does not notice the modification, making prevention of such malicious attacks difficult [221]. These types of weaknesses in DNNs could be exploited and pose a security concern for any technology using DNNs. Although defences against these attacks have been proposed [223], state-of-the-art attacks can by-pass defences and detection mechanisms.

V. Concluding Remarks

In this manuscript, a survey of autonomous vehicle control approaches utilising deep learning was presented. The approaches were separated into three categories: lateral (steering), longitudinal (acceleration and braking), and simultaneous lateral and longitudinal control methods. The focus of this manuscript has been on the vehicle control techniques rather than perception, however there is some obvious overlap between them. It was shown that research interest in this field has grown significantly in recent years and is expected to continue to do so. The applications discussed in this paper show great promise for the application of deep learning to autonomous vehicle control. However, current deep learning based controller performance has significant room for improvement. Moreover, much of the current research is only limited to simulation. While testing in simulation is useful for feasibility studies and initial performance evaluations, extensive testing and training in the field will be required before these systems are ready for deployment.

The main research challenges to deep learning based vehicle control were also discussed and can be seen summarised in Table IV. Computation was identified as a challenge due to the large amount of data required to train deep learning models. Architectures were also identified as a challenge due to the difficulty of choosing the optimal network architecture for a given task. Goal specification is a challenge for reinforcement learning techniques due to the importance of designing a reward function which promotes the desired behaviour. Adaptability and generalisation is a challenge in the autonomous vehicle domain due to the highly complex nature of the operational environment. Verification and validation is a further challenge due to the high cost and time requirements of field tests and training. While simulation is an obvious solution to reduce the amount of physical field testing required, the use of simulation in training and testing has its own drawbacks. Safety was identified as a crucial challenge due to the safety critical nature of the autonomous vehicle domain. This is made more challenging due to the opaque nature of deep learning methods, making safety validation of these systems problematic.

Therefore, further research into interpretability of neural networks and functional safety validation methods for neural network-driven vehicles will be required. Before deep learning can be deployed on the road, some safety validation techniques will need to be found to address their opaqueness. Ensuring the safety of these deep neural networks is a major barrier preventing them from being used commercially. Furthermore, as noted by Salay et al. [224] in their analysis of the ISO26262 [225] standard, more than 40% of the required software techniques in the current version of the standard are incompatible with machine learning techniques, whilst the rest are either directly applicable or applicable if modified slightly. This reveals further need for these standards to be revised to address machine learning systems for autonomous vehicles [226]. Other safety aspects which warrant further research include defences against adversarial attacks, as they currently present a significant safety problem for the use of DNNs in autonomous vehicles. Also, robustness to erroneous inputs from sensory data or communication failures must be investigated. There is currently a significant gap in the literature for investigation of fault tolerant systems. Further research into how deep learning control systems deal with issues such as communication failures, erroneous sensory inputs, input noise, or sensor failure would move the industry towards robust and safe solutions. Furthermore, while research into deep neural networks with both lateral and longitudinal vehicle control is still relatively sparse, there is significant on-going research in this area. Full vehicle control with deep neural networks is typically achieved in simple simulation scenarios and/or with discretised outputs. Much work can be done to improve the performance of the full vehicle control techniques as well. Techniques in Sections III-A and B show promising results for lateral and longitudinal control systems, and future work will be required to bridge these techniques into an autonomous vehicle system with strong performance in the more general case of combined lateral and longitudinal control. This will also include further experiments in the real world to validate the performance of the learned control policies. Other avenues for future research include learning driving manoeuvrers which are still typically achieved through classical control techniques, such as overtaking [227], [228] or merging [229], [230]. Further work will also be needed to design autonomous vehicles which can understand the rules of the road and the behaviour of other road users. Some on-going research was discussed where the deep neural network can take into account the intended route or target destination, but more research is needed to ensure these techniques can stop at stop signs and red lights, respect speed limits, or negotiate intersections and roundabouts with other vehicles.

References

[1] National Highway Traffic Safety Administration (NHTSA). (2017). *2016 Fatal Motor Vehicle Crashes: Overview*. [Online]. Available: https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812456

[2] European Commission. (2017). *2016 Road Safety Statistics: What is Behind the Figures*. [Online]. Available: http://europa.eu/rapid/press-release_MEMO-17-675_en.htm

[3] World Health Organization. (2018). *Global Status Report on Road Safety 2018*. [Online]. Available: https://www.who.int/violence_injury_prevention/road_safety_status/2018/en/

[4] A. Eskandarian, *Handbook of intelligent vehicles*, vol. 2. London, U.K.: Springer, 2012.

[5] S. Thrun, "Toward robotic cars," *Commun. ACM*, vol. 53, no. 4, p. 99, Apr. 2010.

[6] C. Urmson and W. Whittaker, "Self—Driving cars and the urban challenge," *IEEE Intell. Syst.*, vol. 23, no. 2, pp. 66–68, Mar. 2008.

[7] U. Montanaro *et al.*, "Towards connected autonomous driving: Review of use-cases," *Vehicle Syst. Dyn.*, vol. 57, no. 6, pp. 779–814, Jun. 2019.

[8] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," Nature Highway Traffic Safety Admin., U.S. Nat. Center Statist. Anal., Tech. Rep. DOT HS 812 115, Feb. 2015, pp. 1–2.

[9] T. Luettel, M. Himmelsbach, and H.-J. Wuensche, "Autonomous ground vehicles—Concepts and a path to the future," *Proc. IEEE*, vol. 100, no. Special Centennial Issue, pp. 1831–1839, May 2012.

[10] W. Payre, J. Cestac, and P. Delhomme, "Intention to use a fully automated car: Attitudes and *a priori* acceptability," *Transp. Res. F, Traffic Psychol. Behaviour*, vol. 27, pp. 252–263, Nov. 2014.

[11] P. Ross, "Robot, you can drive my car," *IEEE Spectr.*, vol. 51, no. 6, pp. 60–90, Jun. 2014.

[12] Department for Transport. (2017). *Research on the Impacts of Connected and Autonomous Vehicles (CAVs) on Traffic Flow: Summary Report*. [Online]. Available: https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/530091/impacts-of-connected-and-autonomous-vehicles-on-traffic-flow-summary-report.pdf

[13] C. Thorpe, M. Herbert, T. Kanade, and S. Shafter, "Toward autonomous driving: The CMU Navlab. II. Architecture and systems," *IEEE Expert*, vol. 6, no. 4, pp. 44–52, Aug. 1991.

[14] E. Dickmanns and A. Zapp, "Autonomous high speed road vehicle guidance by computer vision 1," *IFAC Proc. Volumes*, vol. 20, no. 5, pp. 221–226, Jul. 1987.

[15] S. Thrun *et al.*, "Stanley: The robot that won the darpa grand challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, 2006.

[16] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous vehicles City Traffic*, vol. 56. Berlin, Germany: Springer, 2009.

[17] T. Le-Anh and M. De Koster, "A review of design and control of automated guided vehicle systems," *Eur. J. Oper. Res.*, vol. 171, no. 1, pp. 1–23, May 2006.

[18] B. Paden, M. Cap, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self–driving urban vehicles," *IEEE Trans. Intell. Veh.*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[19] M. Pasquier, C. Quek, and M. Toh, "Fuzzylot: A novel self-organising fuzzy-neural rule-based pilot system for automated vehicles," *Neural Netw.*, vol. 14, no. 8, pp. 1099–1112, Oct. 2001.

[20] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 2641–2646.

[21] D. Silver, J. A. Bagnell, and A. Stentz, "Learning autonomous driving styles and maneuvers from expert demonstration," in *Experimental Robotics*. Berlin, Germany: Springer, 2013, pp. 371–386.

[22] D. Zhao, B. Wang, and D. Liu, "A supervised Actor—Critic approach for adaptive cruise control," *Soft Comput.*, vol. 17, no. 11, pp. 2089–2099, Nov. 2013.

[23] C. Desjardins and B. Chaib-draa, "Cooperative adaptive cruise control: A reinforcement learning approach," *IEEE Trans. Intell. Transport. Syst.*, vol. 12, no. 4, pp. 1248–1260, Dec. 2011.

[24] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017.

[25] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[26] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[27] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision–Making for autonomous vehicles," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 1, no. 1, pp. 187–210, May 2018.

[28] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robot. Auto. Syst.*, vol. 86, pp. 13–28, Dec. 2016.

[29] S. M. Veres, L. Molnar, N. K. Lincoln, and C. P. Morice, "Autonomous vehicle control systems—A review of decision making," *Proc. Inst. Mech. Eng., I, J. Syst. Control Eng.*, vol. 225, no. 2, pp. 155–195, 2011.

[30] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde, "LIDAR-based driving path generation using fully convolutional neural networks," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–6.

[31] S. Dixit *et al.*, "Trajectory planning and tracking for autonomous overtaking: State-of-the-art and future prospects," *Annu. Rev. Control*, vol. 45, pp. 76–86, Jan. 2018.

[32] H. Zhu, K.-V. Yuen, L. Mihaylova, and H. Leung, "Overview of environment perception for intelligent vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 10, pp. 2584–2601, Oct. 2017.

[33] J. Van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transp. Res. C, Emerg. Technol.*, vol. 89, pp. 384–406, Apr. 2018.

[34] J. Janai, F. Güney, A. Behl, and A. Geiger, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," 2017, *arXiv:1704.05519*. [Online]. Available: https://arxiv.org/abs/1704.05519

[35] R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Ten years of pedestrian detection, what have we learned," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2014, pp. 613–627.

[36] S. Zhang, R. Benenson, M. Omran, J. Hosang, and B. Schiele, "How far are we from solving pedestrian detection?" in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1259–1267.

[37] S. Lowry *et al.*, "Visual place recognition: A survey," *IEEE Trans. Robot.*, vol. 32, no. 1, pp. 1–19, Feb. 2016.

[38] K. Konda and R. Memisevic, "Learning visual odometry with a convolutional network," in *Proc. 10th Int. Conf. Comput. Vis. Theory Appl.*, 2015, pp. 486–490.

[39] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 829–846, Apr. 2018.

[40] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2015.

[41] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with large-scale data collection," in *Proc. Int. Symp. Experim. Robot.* Cham, Switzerland: Springer, 2016, pp. 173–184.

[42] V. Rausch, A. Hansen, E. Solowjow, C. Liu, E. Kreuzer, and J. K. Hedrick, "Learning a deep neural net policy for end-to-end control of autonomous vehicles," in *Proc. Amer. Control Conf. (ACC)*, May 2017, pp. 4914–4919.

[43] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015.

[44] I. Arel, D. C. Rose, and T. P. Karnowski, "Deep machine learning—A new frontier in artificial intelligence research [research frontier]," *IEEE Comput. Intell. Mag.*, vol. 5, no. 4, pp. 13–18, Nov. 2010.

[45] J. Tani, M. Ito, and Y. Sugita, "Self-organization of distributedly represented multiple behavior schemata in a mirror system: Reviews of robot experiments using RNNPB," *Neural Netw.*, vol. 17, nos. 8–9, pp. 1273–1289, Oct. 2004.

[46] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[47] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[48] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org/

[49] M. Nielsen, *Neural Networks and Deep Learning*. San Francisco, CA, USA: Determination Press, 2015.

[50] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 9. Cambridge, MA, USA: MIT Press, 1998.

[51] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

[52] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*. [Online]. Available: https://arxiv.org/abs/1701.07274

[53] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 161–168.

[54] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 627–635.

[55] P. de Haan, D. Jayaraman, and S. Levine, "Causal confusion in imitation learning," 2019, *arXiv:1905.11979*. [Online]. Available: https://arxiv.org/abs/1905.11979

[56] A. Torralba and A. A. Efros, "Unbiased look at dataset bias," in *Proc. CVPR*, Jun. 2011, pp. 1521–1528.

[57] A. Gupta, A. Murali, D. P. Gandhi, and L. Pinto, "Robot learning in homes: Improving generalization and reducing dataset bias," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9094–9104.

[58] Z. Wang *et al.*, "Sample efficient actor-critic with experience replay," 2016, *arXiv:1611.01224*. [Online]. Available: https://arxiv.org/abs/1611.01224

[59] V. R. Konda and J. N. Tsitsiklis, "On actor-critic algorithms," *SIAM J. Control Optim.*, vol. 42, no. 4, pp. 1143–1166, 2003.

[60] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.

[61] G. J. Gordon, "Stable function approximation in dynamic programming," in *Proc. 12th Int. Conf. Mach. Learn.*, Jul. 1995, pp. 261–268.

[62] J. N. Tsitsiklis and B. Van Roy, "Feature-based methods for large scale dynamic programming," *Mach. Learn.*, vol. 22, nos. 1–3, pp. 59–94, 1996.

[63] R. J. Williams, *Reinforcement-Learning Connectionist Systems*. Boston, MA, USA: College of Computer Science, Northeastern Univ., 1987.

[64] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.

[65] M. Riedmiller, J. Peters, and S. Schaal, "Evaluation of policy gradient methods and variants on the cart–pole benchmark," in *Proc. IEEE Int. Symp. Approx. Dyn. Program. Reinforcement Learn.*, Apr. 2007, pp. 254–261.

[66] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. 31st Int. Conf. Mach. Learn.*, Beijing, China, vol. 32, 2014.

[67] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[68] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A survey of actor–critic reinforcement learning: Standard and natural policy gradients," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012.

[69] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*. [Online]. Available: https://arxiv.org/abs/1506.02438

[70] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013.

[71] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 3354–3361.

[72] (2019). *Waymo Open Dataset: An Autonomous Driving Dataset*. [Online]. Available: https://www.waymo.com/open

[73] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The Oxford RobotCar dataset," *Int. J. Robot. Res.*, vol. 36, no. 1, pp. 3–15, Jan. 2017, doi: 10.1177/0278364916679498.

[74] X. Huang *et al.*, "The ApolloScape open dataset for autonomous driving and its application," 2018, *arXiv:1803.06184*. [Online]. Available: https://arxiv.org/abs/1803.06184

[75] Udacity Inc. (2018). *Udacity Self-driving Car Dataset*. [Online]. Available: https://github.com/udacity/self-driving-car

[76] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "A mobile vision system for robust multi-person tracking," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[77] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 304–311.

[78] H. Yin and C. Berger, "When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–8.

[79] NVIDIA Corporation. (2018). *Autonomous car development platform from NVIDIA DRIVE PX2*. [Online]. Available: https://www.nvidia.com/en-us/self-driving-cars/drive-platform/

[80] MobilEye. (2018). *The Evolution of EyeQ*. [Online]. Available: https://www.mobileye.com/our-technology/evolution-eyeq-chip/

[81] Intel Corporation. (2018). *Cyclone V—Overview*. [Online]. Available: https://www.altera.com/products/fpga/cyclone-series/cyclone-v/overview.html

[82] S. Liu, J. Tang, Z. Zhang, and J.-L. Gaudiot, "CAAD: Computer architecture for autonomous driving," 2017, *arXiv:1702.01894*. [Online]. Available: https://arxiv.org/abs/1702.01894

[83] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 1989, pp. 305–313.

[84] D. Pomerleau, "Neural network vision for robot driving," in *Intelligent Unmanned Ground Vehicles*. Cambridge, MA, USA: MIT Press, 1997, pp. 53–72.

[85] G. Yu and I. Sethi, "Road-following with continuous learning," in *Proc. Intell. Vehicles Symp.*, Detroit, MI, USA, Nov. 2002, pp. 412–417.

[86] D. E. Moriarty, S. Handley, and P. Langley, "Learning distributed strategies for traffic control," *Proc. 5th Int. Conf. Soc. Adapt. Behav.*, May 1998, pp. 437–446.

[87] M. Bojarski *et al.*, "End to end learning for self-driving cars," May 2016, *arXiv:1604.07316*. [Online]. Available: https://arxiv.org/abs/1604.07316

[88] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 739–746.

[89] Mechanical Simulation Corporation. *CarSim*. [Online]. Available: https://www.carsim.com

[90] L. Bottou, "Large–scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*. Physica-Verlag HD, 2010, pp. 177–186.

[91] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980

[92] W. Su, S. Boyd, and E. Candes, "A differential equation for modeling nesterov's accelerated gradient method: Theory and insights," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2510–2518.

[93] H. M. Eraqi, M. N. Moustafa, and J. Honer, "End-to-end deep learning for steering autonomous vehicles considering temporal dependencies," 2017, *arXiv:1710.03804*. [Online]. Available: https://arxiv.org/abs/1710.03804

[94] R. Rothe, R. Timofte, and L. V. Gool, "DEX: Deep expectation of apparent age from a single image," in *Proc. IEEE Int. Conf. Comput. Vis. Workshop (ICCVW)*, Dec. 2015, pp. 10–15.

[95] F. Codevilla, A. M. López, V. Koltun, and A. Dosovitskiy, "On offline evaluation of vision-based driving models," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 236–251.

[96] P. Wang, C.-Y. Chan, and A. D. L. Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2018, pp. 1379–1384.

[97] A. Vahidi and A. Eskandarian, "Research advances in intelligent collision avoidance and adaptive cruise control," *IEEE Trans. Intell. Transport. Syst.*, vol. 4, no. 3, pp. 143–153, Sep. 2003.

[98] S. Moon, I. Moon, and K. Yi, "Design, tuning, and evaluation of a full-range adaptive cruise control system with collision avoidance," *Control Eng. Pract.*, vol. 17, no. 4, pp. 442–455, Apr. 2009.

[99] Q. Sun, "Cooperative adaptive cruise control performance analysis," Ph.D. dissertation, Centre Recherche Informatique, Signal Automatique de Lille (CRIStAL), École Centrale de Lille, Villeneuve-d'Ascq, France, 2016.

[100] X. Chen, Y. Zhai, C. Lu, J. Gong, and G. Wang, "A learning model for personalized adaptive cruise control," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 379–384.

[101] D. Wang and J. Huang, "Neural network–based adaptive dynamic surface control for a class of uncertain nonlinear systems in strict–feedback form," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 195–202, Jan. 2005.

[102] M. Polycarpou, "Stable adaptive neural control scheme for nonlinear systems," *IEEE Trans. Automat. Contr.*, vol. 41, no. 3, pp. 447–451, Mar. 1996.

[103] R. Sanner and M. Mears, "Stable adaptive tracking of uncertainty systems using nonlinearly parameterized on-line approximators," *IEEE Trans. Neural Netw.*, vol. 3, no. 6, pp. 837–863, 1992.

[104] D. Wang and J. Huang, "Adaptive neural network control for a class of uncertain nonlinear systems in pure-feedback form," *Automatica*, vol. 38, no. 8, pp. 1365–1372, Aug. 2002.

[105] B. Ren, S. Ge, C.-Y. Su, and T. Heng Lee, "Adaptive neural control for a class of uncertain nonlinear systems in pure–feedback form with hysteresis input," *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, vol. 39, no. 2, pp. 431–443, Apr. 2009.

[106] T. Zhang, "Adaptive neural network control for strict-feedback nonlinear systems using backstepping design," *Automatica*, vol. 36, no. 12, pp. 1835–1846, Dec. 2000.

[107] X. Dai, C. Li, and A. Rad, "An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 3, pp. 285–293, Sep. 2005.

[108] Z. Huang, X. Xu, H. He, J. Tan, and Z. Sun, "Parameterized batch reinforcement learning for longitudinal control of autonomous land vehicles," *IEEE Trans. Syst., Man, Cybern, Syst.*, vol. 49, no. 4, pp. 730–741, Apr. 2019.

[109] X. Xu, D. Hu, and X. Lu, "Kernel–based least squares policy iteration for reinforcement learning," *IEEE Trans. Neural Netw.*, vol. 18, no. 4, pp. 973–992, Jul. 2007.

[110] J. Wang, X. Xu, D. Liu, Z. Sun, and Q. Chen, "Self–learning cruise control using kernel–based least squares policy iteration," *IEEE Trans. Contr. Syst. Technol.*, vol. 22, no. 3, pp. 1078–1087, May 2014.

[111] H. Chae, C. M. Kang, B. Kim, J. Kim, C. C. Chung, and J. W. Choi, "Autonomous braking system via deep reinforcement learning," in *Proc. IEEE 20th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2017, pp. 1–6.

[112] "European new car assessment programme: Test Protocol—AEB VRU systems," Euro NCAP, Brussels, Belgium, Tech. Rep., 2015.
[113] D. Zhao, Z. Xia, and Q. Zhang, "Model–free optimal control based intelligent cruise control with hardware-in-the-loop demonstration [research frontier]," *IEEE Comput. Intell. Mag.*, vol. 12, no. 2, pp. 56–69, May 2017.
[114] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.
[115] D. Zhao, Z. Hu, Z. Xia, C. Alippi, Y. Zhu, and D. Wang, "Full-range adaptive cruise control based on supervised adaptive dynamic programming," *Neurocomputing*, vol. 125, pp. 57–67, Feb. 2014.
[116] B. Wang, D. Zhao, C. Li, and Y. Dai, "Design and implementation of an adaptive cruise control system based on supervised actor-critic learning," in *Proc. 5th Int. Conf. Inf. Sci. Technol. (ICIST)*, Apr. 2015, pp. 243–248.
[117] T. Lai and H. Robbins, "Asymptotically efficient adaptive allocation rules," *Adv. Appl. Math.*, vol. 6, no. 1, pp. 4–22, Mar. 1985.
[118] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1471–1479.
[119] J. Schmidhuber, "A possibility for implementing curiosity and boredom in model-building neural controllers," in *Proc. Int. Conf. Simulation Adapt. Behav., Animals Animats*, 1991, pp. 222–227.
[120] W. Xia, H. Li, and B. Li, "A control strategy of autonomous vehicles based on deep reinforcement learning," in *Proc. 9th Int. Symp. Comput. Intell. Design (ISCID)*, Dec. 2016, pp. 198–201.
[121] M. Riedmiller, "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method," in *Proc. Eur. Conf. Mach. Learn.* Berlin, Germany: Springer, 2005, pp. 317–328.
[122] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-End deep reinforcement learning for lane keeping assist," 2016, *arXiv:1612.04340*. [Online]. Available: https://arxiv.org/abs/1612.04340
[123] *The Open Racing Car Simulator*. Accessed: Dec. 28, 2019. [Online]. Available: http://torcs.sourceforge.net/
[124] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: https://arxiv.org/abs/1509.02971
[125] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end autonomous driving," 2016, *arXiv:1605.06450*. [Online]. Available: https://arxiv.org/abs/1605.06450
[126] Y. Pan *et al.*, "Agile autonomous driving using end-to-end deep imitation learning," in *Robotics: Science and Systems*. Pittsburgh, PA, USA: Robotics Science and Systems (RSS) Foundation, 2018.
[127] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Apr. 2005, pp. 2149–2154.
[128] D. Wang, C. Devin, Q.-Z. Cai, F. Yu, and T. Darrell, "Deep object centric policies for autonomous driving," 2018, *arXiv:1811.05432*. [Online]. Available: https://arxiv.org/abs/1811.05432
[129] H. Porav and P. Newman, "Imminent collision mitigation with reinforcement learning and vision," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 958–964.
[130] S. Zhifei and E. Meng Joo, "A review of inverse reinforcement learning theory and recent advances," in *Proc. IEEE Congr. Evol. Comput.*, Jun. 2012, pp. 1–8.
[131] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, 2004, p. 1.
[132] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 729–736.
[133] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *Int. J. Robot. Res.*, vol. 36, no. 10, pp. 1073–1087, Sep. 2017.
[134] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum Entropy Inverse Reinforcement Learning.," in *Proc. AAAI Conf. Artif. Intell.*, 2008, pp. 1433–1438.
[135] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2012, pp. 41–48.
[136] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, "An application of reinforcement learning to aerobatic helicopter flight," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 19, 2007, p. 1.
[137] S. Hecker, D. Dai, and L. Van Gool, "End-to-end learning of driving models with surround-view cameras and route planners," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 435–453.
[138] F. Codevilla, M. Miiller, A. Lopez, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–9.
[139] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, 2017, pp. 1–16.
[140] F. Codevilla, E. Santana, A. M. López, and A. Gaidon, "Exploring the limitations of behavior cloning for autonomous driving," 2019, *arXiv:1904.08980*. [Online]. Available: https://arxiv.org/abs/1904.08980
[141] C. Paxton, V. Raman, G. D. Hager, and M. Kobilarov, "Combining neural networks and tree search for task and motion planning in challenging environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 6059–6066.
[142] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," 2018, *arXiv:1812.03079*. [Online]. Available: https://arxiv.org/abs/1812.03079
[143] X. Pan, Y. You, Z. Wang, and C. Lu, "Virtual to real reinforcement learning for autonomous driving," 2017, *arXiv:1704.03952*. [Online]. Available: https://arxiv.org/abs/1704.03952
[144] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun, "Driving policy transfer via modularity and abstraction," 2018, *arXiv:1804.09364*. [Online]. Available: https://arxiv.org/abs/1804.09364
[145] M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner, *Autonomous Driving*. Berlin, Germany: Springer, 2016.
[146] European Commission. (2016). *Cooperative Intelligent Transportation Systems—Research Theme Analysis Report*. [Online]. Available: http://www.transport-research.info/sites/default/files/TRIP_C-ITS_Report.pdf
[147] S. A. Bagloee, M. Tavana, M. Asadi, and T. Oliver, "Autonomous vehicles: Challenges, opportunities, and future implications for transportation policies," *J. Mod. Transp.*, vol. 24, no. 4, pp. 284–303, Dec. 2016.
[148] HERE Technologies. (2017). *Consumer Acceptance of Autonomous Vehicles*. [Online]. Available: https://here.com/file/13726/download?token=njs4ZwfW
[149] L. Bosankic. (2017). *How consumers' Perception of Autonomous Cars Will Influence Their Adoption*. [Online]. Available: https://medium.com/@leo_pold_b/how-consumers-perception-of-autonomous-cars-will-influence-their-adoption-ba99e3f64e9a
[150] H. Abraham, B. Reimer, B. Seppelt, C. Fitzgerald, B. Mehler, and J. F. Coughlin. (2017). *Consumer Interest in Automation: Preliminary Observations Exploring a Year's Change*. [Online]. Available: http://agelab.mit.edu/sites/default/files/MIT-NEMPA White Paper-FINAL.pdf
[151] W. Knight. (2016). *An Ambitious Plan to Build a Self-Driving Borg*. [Online]. Available: https://www.technologyreview.com/s/602531/an-ambitious-plan-to-build-a-self-driving-borg/
[152] J. J. Kober and B. A. P. Jan, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.
[153] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.
[154] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine, "Continuous deep q-learning with model-based acceleration," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 2829–2838.
[155] J. T. Barron, D. S. Golland, and N. J. Hay, *Parallelizing Reinforcement Learning*. Berkeley, CA, USA: UC Berkeley, 2009.
[156] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017. *arXiv:1703.03864*. [Online]. Available: https://arxiv.org/abs/1703.03864
[157] S. Yang, W. Wang, C. Liu, W. Deng, and J. K. Hedrick, "Feature analysis and selection for training an end-to-end autonomous vehicle controller using deep learning approach," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 1033–1038.
[158] Y. LeCun, "Generalization and network design strategies," in *Connectionism Perspective*. Amsterdam, The Netherlands: Elsevier, 1989, pp. 143–155.
[159] N. Morgan and H. Bourlard, "Generalization and parameter estimation in feedforward nets: Some experiments," in *Proc. Adv. Neural Inf. Process. Syst.*, 1989, pp. 630–637.

[160] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," in *Neural Networks: Tricks trade*. Berlin, Germany: Springer, 2012, pp. 437–478.

[161] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks trade*. Springer, 1998, pp. 9–50.

[162] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.

[163] C. Raffel. (2015). *Neural Network Hyperparameters*. [Online]. Available: http://colinraffel.com/wiki/neural_network_hyperparameters

[164] Y. Sevchuk. (2016). *Hyperparameter Optimization for Neural Networks*. [Online]. Available: http://neupy.com/2016/12/17/hyperparameter_optimization_for_neural_networks.html

[165] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25, pp. 2960–2968, 2012.

[166] C. E. Rasmussen, "Gaussian processes in machine learning," in *Advanced Lectures on Machine Learning*. Berlin, Germany: Springer, 2004, pp. 63–71.

[167] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2011, pp. 2546–2554.

[168] M. Kumar, G. E. Dahl, V. Vasudevan, and M. Norouzi, "Parallel architecture and hyperparameter search via successive halving and classification," 2018, *arXiv:1805.10255*. [Online]. Available: https://arxiv.org/abs/1805.10255

[169] T. B. Hashimoto, S. Yadlowsky, and J. C. Duchi, "Derivative free optimization via repeated classification," 2018, *arXiv:1804.03761*. [Online]. Available: https://arxiv.org/abs/1804.03761

[170] H. Cai, C. Gan, and S. Han, "Once for all: Train one network and specialize it for efficient deployment," 2019, *arXiv:1908.09791*. [Online]. Available: https://arxiv.org/abs/1908.09791

[171] M. Tan *et al.*, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 2820–2828.

[172] B. Wu *et al.*, "FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 10734–10742.

[173] F. Scheidegger, L. Benini, C. Bekas, and C. Malossi, "Constrained deep neural network architecture search for IoT devices accounting hardware calibration," 2019, *arXiv:1909.10818*. [Online]. Available: https://arxiv.org/abs/1909.10818

[174] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[175] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. 16th Int. Conf. Mach. Learn.*, vol. 3, 1999, pp. 278–287.

[176] A. D. Laud, "Theory and application of reward shaping in reinforcement learning," Ph.D. dissertation, Dept. Comput. Sci., Univ. Illinois, Champaign, IL, USA, 2004.

[177] H. Van Seijen, M. Fatemi, J. Romoff, R. Laroche, T. Barnes, and J. Tsang, "Hybrid reward architecture for reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5392–5402.

[178] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "Safe, multi-agent, reinforcement learning for autonomous driving," 2016, *arXiv:1610.03295*. [Online]. Available: https://arxiv.org/abs/1610.03295

[179] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in ai safety," 2016, *arXiv:1606.06565*. [Online]. Available: https://arxiv.org/abs/1606.06565

[180] S. Russell, "Learning agents for uncertain environments (extended abstract)," in *Proc. 11th Annu. Conf. Comput. Learn. Theory (COLT)*, 1998, pp. 101–103.

[181] J. Z. Kolter, P. Abbeel, and A. Y. Ng, "Hierarchical apprenticeship learning, with application to quadruped locomotion," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 1, 2008, pp. 769–776.

[182] D. Silver, J. A. Bagnell, and A. Stentz, "Learning from demonstration for autonomous navigation in complex unstructured terrain," *Int. J. Robot. Res.*, vol. 29, no. 12, pp. 1565–1592, Oct. 2010.

[183] N. Ratliff, J. A. Bagnell, and S. S. Srinivasa, "Imitation learning for locomotion and manipulation," in *Proc. 7th IEEE-RAS Int. Conf. Humanoid Robots, HUMANOIDS*, Nov. 2007, pp. 392–397.

[184] R. J. Schalkoff, *Artificial Neural Networks*. New York, NY, USA: McGraw-Hill, 1997.

[185] B. D. Ripley, *Artificial Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.

[186] J. A. Marshall, "Neural networks for pattern recognition," *Neural Netw.*, vol. 8, no. 3, pp. 493–494, Jan. 1995.

[187] A. Y. Ng, "Feature selection, L1 vs. L2 regularization, and rotational invariance," in *Proc. 21st Int. Conf. Mach. Learn. (ICML)*, 2004, p. 78.

[188] P. Merolla, R. Appuswamy, J. Arthur, S. K. Esser, and D. Modha, "Deep neural networks are robust to weight binarization and other non-linear distortions," 2016, *arXiv:1606.01981*. [Online]. Available: https://arxiv.org/abs/1606.01981

[189] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

[190] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," 2012, *arXiv:1207.0580*. [Online]. Available: https://arxiv.org/abs/1207.0580

[191] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[192] H. Raza and P. Ioannou, "Vehicle following control design for automated highway systems," *IEEE Control Syst.*, vol. 16, no. 6, pp. 43–60, Dec. 1996.

[193] R. Rajamani, H.-S. Tan, B. Kait Law, and W.-B. Zhang, "Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons," *IEEE Trans. Contr. Syst. Technol.*, vol. 8, no. 4, pp. 695–708, Jul. 2000.

[194] C. Thorpe, T. Jochem, and D. Pomerleau, "The 1997 automated highway free agent demonstration," in *Proc. Conf. Intell. Transp. Syst.*, Nov. 2002, pp. 496–501.

[195] S. Kato, S. Tsugawa, K. Tokuda, T. Matsui, and H. Fujii, "Vehicle control algorithms for cooperative driving with automated vehicles and intervehicle communications," *IEEE Trans. Intell. Transport. Syst.*, vol. 3, no. 3, pp. 155–161, Sep. 2002.

[196] L. Ng, C. M. Clark, and J. P. Huissoon, "Reinforcement learning of adaptive longitudinal vehicle control for dynamic collaborative driving," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2008, pp. 907–912.

[197] C. G. Atkeson, "Using local trajectory optimizers to speed up global optimization in dynamic programming," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 1994, pp. 663–670.

[198] T. Hester *et al.*, "Learning from demonstrations for real world reinforcement learning," 2017, *arXiv:1704.03732*. [Online]. Available: https://arxiv.org/abs/1704.03732

[199] P. Christiano *et al.*, "Transfer from simulation to real world through learning deep inverse dynamics model," 2016, *arXiv:1610.03518*. [Online]. Available: https://arxiv.org/abs/1610.03518

[200] A. A. Rusu, M. Vecerik, T. Rothörl, N. Heess, R. Pascanu, and R. Hadsell, "Sim-to-real robot learning from pixels with progressive nets," 2016, *arXiv:1610.04286*. [Online]. Available: https://arxiv.org/abs/1610.04286

[201] E. Tzeng *et al.*, "Towards adapting deep visuomotor representations from simulated to real environments," 2015, *arXiv:1511.07111*. [Online]. Available: https://arxiv.org/abs/1511.07111

[202] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 23–30.

[203] K. R. Varshney and H. Alemzadeh, "On the safety of machine learning: Cyber–Physical systems, decision sciences, and data products," *Big Data*, vol. 5, no. 3, pp. 246–255, Sep. 2017.

[204] D. Castelvecchi, "Can we open the black box of AI?" *Nature*, vol. 538, no. 7623, pp. 20–23, Oct. 2016.

[205] X. Zhang, M. Clark, K. Rattan, and J. Muse, "Controller verification in adaptive learning systems towards trusted autonomy," in *Proc. ACM/IEEE 6th Int. Conf. Cyber-Phys. Syst. (ICCPS)*, 2015, pp. 31–40.

[206] M. Clark *et al.*, "A study on run time assurance for complex cyber physical systems," Air Force Research Lab Wright-Patterson Afb Oh Aerospace Systems Dir, Wright-Patterson Air Force Base, OH, USA, Tech. Rep., 2013.

[207] S. Jacklin, J. Schumann, P. Gupta, M. Richard, K. Guenther, and F. Soares, "Development of advanced verification and validation procedures and tools for the certification of learning systems in aerospace applications," in *Proc. Infotech Aerospace*, Sep. 2005, p. 6912.

[208] C. Wilkinson, J. Lynch, and R. Bharadwaj, *Final Report, Regulatory Considerations for Adaptive Systems. National Aeronautics and Space Administration*. Hampton, VA, USA: Langley Research Center, 2013.
[209] P. Van Wesel and A. E. Goodloe, "Challenges in the verification of reinforcement learning algorithms," NASA, Washington, DC, USA, Tech. Rep. NASA/TM-2017-219628, 2017.
[210] J. G. Schneider, "Exploiting model uncertainty estimates for safe dynamic control learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 1997, pp. 1047–1053.
[211] J. A. Bagnell, "Learning decisions: Robustness, uncertainty, and appoximation," Ph.D. dissertation, Robot. Inst., Carnegie Mellon Univ., Pittsburgh, PA, USA, Aug. 2004, p. 78.
[212] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proc. 28th Int. Conf. Mach. Learn. (ICML-11)*, 2011, pp. 465–472.
[213] T. M. Moldovan and P. Abbeel, "Safe exploration in Markov decision processes," 2012, *arXiv:1205.4810*. [Online]. Available: https://arxiv.org/abs/1205.4810
[214] X. Xiong, J. Wang, F. Zhang, and K. Li, "Combining deep reinforcement learning and safety based control for autonomous driving," 2016, *arXiv:1612.00147*. [Online]. Available: https://arxiv.org/abs/1612.00147
[215] S. Glaser, B. Vanholme, S. Mammar, D. Gruyer, and L. Nouveliere, "Maneuver–based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction," *IEEE Trans. Intell. Transport. Syst.*, vol. 11, no. 3, pp. 589–606, Sep. 2010.
[216] C. Szegedy *et al.*, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*. [Online]. Available: https://arxiv.org/abs/1312.6199
[217] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 427–436.
[218] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2574–2582.
[219] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*. [Online]. Available: https://arxiv.org/abs/1412.6572
[220] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, "Safety verification of deep neural networks," in *Computer Aided Verification* (Lecture Notes in Computer Science), vol. 10426. Cham, Switzerland: Springer, 2017, pp. 3–29.
[221] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, *arXiv:1607.02533*. [Online]. Available: https://arxiv.org/abs/1607.02533
[222] A. Athalye, L. Engstrom, A. Ilyas, and K. Kwok, "Synthesizing robust adversarial examples," 2017, *arXiv:1707.07397*. [Online]. Available: https://arxiv.org/abs/1707.07397
[223] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.
[224] R. Salay, R. Queiroz, and K. Czarnecki, "An analysis of ISO 26262: Using machine learning safely in automotive software," 2017, *arXiv:1709.02435*. [Online]. Available: https://arxiv.org/abs/1709.02435
[225] *International Organization for Standardization: Road vehicles-functional safety*, Standard ISO26262, International Standard ISO/FDIS, 2011.
[226] F. Falcini, G. Lami, and A. M. Costanza, "Deep learning in automotive software," *IEEE Softw.*, vol. 34, no. 3, pp. 56–63, May 2017.
[227] S. Dixit *et al.*, "Trajectory planning for autonomous high-speed overtaking using MPC with terminal set constraints," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 1061–1068.
[228] S. Dixit *et al.*, "Trajectory planning for autonomous high-speed overtaking in structured environments using robust MPC," *IEEE Trans. Intell. Transp. Syst.*, to be published.
[229] K. Amezquita-Semprun, Y. C. Pradeep, P. C. Chen, W. Chen, and Z. Zhao, "Experimental evaluation of the stimuli-induced equilibrium point concept for automatic ramp merging systems," *IEEE Trans. Intell. Transp. Syst.*, to be published.
[230] V. Milanes, J. Godoy, J. Villagra, and J. Perez, "Automated On–Ramp merging system for congested traffic situations," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 500–508, Jun. 2011.

Sampo Kuutti received the M.Eng. degree in mechanical engineering from the University of Surrey, Guildford, U.K., in 2017, where he is currently pursuing the Ph.D. degree in automotive engineering with the Connected Autonomous Vehicles Lab within the Centre for Automotive Engineering. His research interests include deep learning applied to autonomous vehicles, functional safety validation, and safety and interpretability in machine learning systems.

Richard Bowden (Senior Member, IEEE) is currently a Professor of computer vision and machine learning with the University of Surrey, where he leads the Cognitive Vision Group within the Centre for Vision, Speech and Signal Processing. His research centers on the use of computer vision to locate, track, and understand humans. He is also a fellow of the Higher Education Academy and the International Association of Pattern Recognition (IAPR). In 2013, he was awarded the Royal Society Leverhulme Trust Senior Research Fellowship. He is also an Associate Editor of the *Journals Image and Vision computing* and the IEEE TPAMI.

Yaochu Jin (Fellow, IEEE) is currently a Professor in computational intelligence with the Department of Computer Science, University of Surrey, Guildford, U.K. His main research interests include data-driven surrogate-assisted evolutionary optimization, evolutionary learning, interpretable and secure machine learning, and evolutionary developmental systems.

He is also the Editor-in-Chief of the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS and the Co-Editor-in-Chief of Complex and Intelligent Systems. He is an IEEE Distinguished Lecturer.

Phil Barber was formerly Principal Technical Specialist of capability research with Jaguar Land Rover. For over 30 years in the automotive industry, he has witnessed the introduction of computer controlled by-wire technology and been part of the debate over the safety issues involved in the implementation of real-time vehicle control.

Saber Fallah is currently a Senior Lecturer (Associate Professor) of vehicle and mechatronic systems with the University of Surrey and the Director of Connected Autonomous Vehicle Lab within the Centre for Automotive Engineering, where he leads several research activities funded by the U.K. and European Government (e.g., EPSRC, Innovate U.K., and H2020) in collaboration with major companies active in autonomous vehicle technologies. His research interests include reinforced deep learning, advanced control, optimization, and estimation and their applications to connected autonomous vehicles.