Received: 27 February 2019 | Revised: 24 August 2019 | Accepted: 16 October 2019

DOI: 10.1002/rob.21918

SURVEY ARTICLE

A survey of deep learning techniques for autonomous driving

Sorin Grigorescu | Bogdan Trasnea | Tiberiu Cocias | Gigel Macesanu

Artificial Intelligence, Elektrobit Automotive, Robotics, Vision and Control Laboratory, Transilvania University of Brasov, Brasov, Romania

Correspondence
Sorin Grigorescu, Artificial Intelligence, Elektrobit Automotive, Robotics, Vision and Control Laboratory, Transilvania University of Brasov, 500036 Brasov, Romania.
Email: Sorin.Grigorescu@elektrobit.com

Abstract

The last decade witnessed increasingly rapid progress in self-driving vehicle technology, mainly backed up by advances in the area of deep learning and artificial intelligence (AI). The objective of this paper is to survey the current state-of-the-art on deep learning technologies used in autonomous driving. We start by presenting AI-based self-driving architectures, convolutional and recurrent neural networks, as well as the deep reinforcement learning paradigm. These methodologies form a base for the surveyed driving scene perception, path planning, behavior arbitration, and motion control algorithms. We investigate both the modular perception-planning-action pipeline, where each module is built using deep learning methods, as well as End2End systems, which directly map sensory information to steering commands. Additionally, we tackle current challenges encountered in designing AI architectures for autonomous driving, such as their safety, training data sources, and computational hardware. The comparison presented in this survey helps gain insight into the strengths and limitations of deep learning and AI approaches for autonomous driving and assist with design choices.

KEYWORDS

AI for self-driving vehicles, artificial intelligence, autonomous driving, deep learning for autonomous driving

1 | INTRODUCTION

Over[1] the course of the last decade, deep learning and artificial intelligence (AI) became the main technologies behind many breakthroughs in computer vision (Krizhevsky, Sutskever, & Hinton, 2012), robotics (Andrychowicz et al., 2018), and natural language processing (NLP; Goldberg, 2017). They also have a major impact in the autonomous driving revolution seen today both in academia and industry. Autonomous vehicles (AVs) and self-driving cars began to migrate from laboratory development and testing conditions to driving on public roads. Their deployment in our environmental landscape offers a decrease in road accidents and traffic congestions, as well as an improvement of our mobility in overcrowded cities. The title of "self-driving" may seem self-evident, but there are actually five safety in automotive software (SAE) Levels used to define autonomous driving. The SAE J3016 standard (SAE Committee, 2014) introduces a scale from 0 to 5 for grading vehicle automation. Lower SAE Levels feature basic driver assistance, whilst higher SAE Levels move towards vehicles requiring no human interaction whatsoever. Cars in the Level 5 category require no human input and typically will not even feature steering wheels or foot pedals.

Although most driving scenarios can be relatively simply solved with classical perception, path planning, and motion control methods, the remaining unsolved scenarios are corner cases in which traditional methods fail.

One of the first autonomous cars was developed by Ernst Dickmanns (Dickmanns & Graefe, 1988) in the 1980s. This paved the way for new

The authors are with *Elektrobit Automotive* and the *Robotics, Vision and Control Laboratory* (ROVIS Lab) at the Department of Automation and Information Technology, Transilvania University of Brasov, 500036 Brasov, Romania. See http://rovislab.com/sorin_grigorescu.html.

[1]The articles referenced in this survey can be accessed at the web-page accompanying this paper, available at http://rovislab.com/survey_DL_AD.html

research projects, such as PROMETHEUS, which aimed to develop a fully functional autonomous car. In 1994, the versuchsfahrzeug für autonome mobilität und rechnersehen (VaMP) driverless car managed to drive 1,600 km, out of which 95% were driven autonomously. Similarly, in 1995, carnegie mellon navigation laboratory (CMU NAVLAB) demonstrated autonomous driving on 6,000 km, with 98% driven autonomously. Another important milestone in autonomous driving was the Defense Advanced Research Projects Agency (DARPA) Grand Challenges in 2004 and 2005, as well as the DARPA Urban Challenge in 2007. The goal was for a driverless car to navigate an off-road course as fast as possible, without human intervention. In 2004, none of the 15 vehicles completed the race. Stanley, the winner of the 2005 race, leveraged Machine Learning techniques for navigating the unstructured environment. This was a turning point in self-driving cars development, acknowledging Machine Learning and AI as central components of autonomous driving. The turning point is also notable in this survey paper, since the majority of the surveyed work is dated after 2005.

In this survey, we review the different AI and deep learning technologies used in autonomous driving, and provide a survey on state-of-the-art deep learning and AI methods applied to self-driving cars. We also dedicate complete sections on tackling safety aspects, the challenge of training data sources, and the required computational hardware.

2 | DEEP LEARNING-BASED DECISION-MAKING ARCHITECTURES USED IN SELF-DRIVING CARS

Self-driving cars are autonomous decision-making systems that process streams of observations coming from different on-board sources, such as cameras, radars, light detection and rangings (LiDARs), ultrasonic sensors, global positioning system (GPS) units and/or inertial sensors. These observations are used by the car's computer to make driving decisions. The basic block diagrams of an AI powered autonomous car are shown in Figure 1. The driving decisions are computed either in a modular perception-planning-action pipeline (Figure 1a) or in an End2End learning fashion (Figure 1b), where sensory information is directly mapped to control outputs. The components of the modular pipeline can be designed either based on AI and deep learning methodologies, or using classical nonlearning approaches. Various permutations of learning- and nonlearning-based components are possible (e.g., a deep learning-based object detector provides input to a classical A-star path planning algorithm). A safety monitor is designed to assure the safety of each module.

The modular pipeline in Figure 1a is hierarchically decomposed into four components which can be designed using either deep learning and AI approaches, or classical methods. These components are

- *perception and localization*,
- *high-level path planning*,
- *behavior arbitration, or low-level path planning*,
- *motion controllers*.

On the basis of these four high-level components, we have grouped together relevant deep learning papers describing methods developed for autonomous driving systems. Additional to the reviewed algorithms, we have also grouped relevant articles covering the *safety*, *data sources*, and *hardware* aspects encountered when designing deep learning modules for self-driving cars.

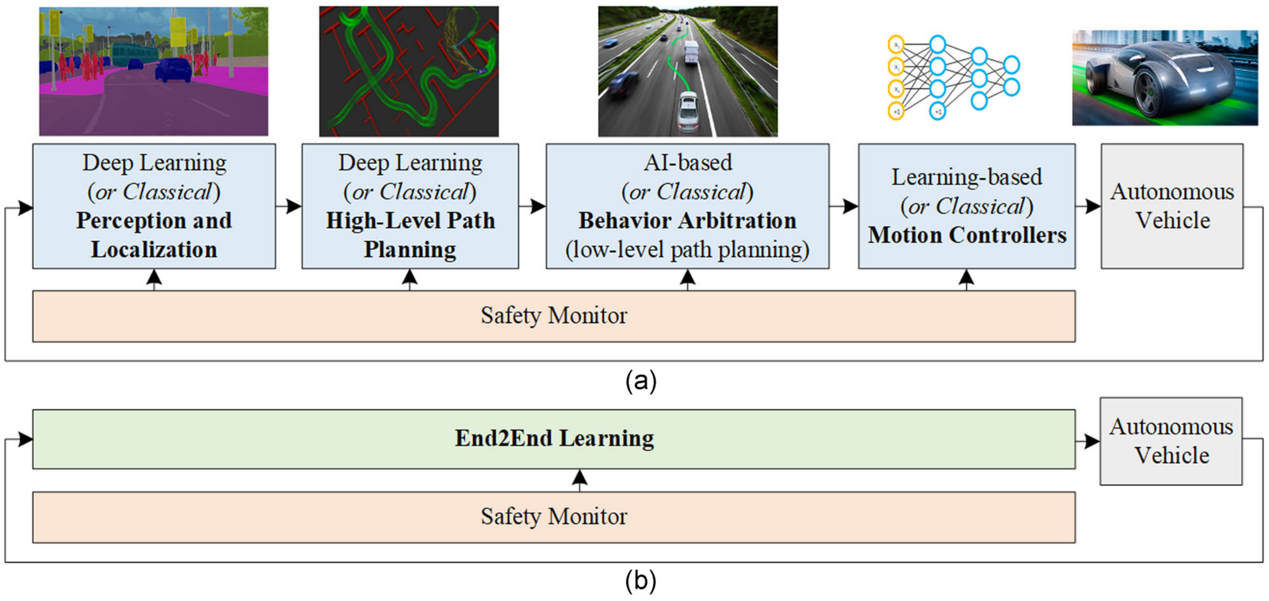
Deep Learning (or Classical) Perception and Localization
Deep Learning (or Classical) High-Level Path Planning
AI-based (or Classical) Behavior Arbitration (low-level path planning)
Learning-based (or Classical) Motion Controllers
Autonomous Vehicle
Safety Monitor
(a)
End2End Learning
Autonomous Vehicle
Safety Monitor
(b)

FIGURE 1 Deep learning-based self-driving car. The architecture can be implemented either as a sequential perception-planning-action pipeline (a) or as an End2End system (b). In the sequential pipeline case, the components can be designed either using AI and deep learning methodologies, or based on classical nonlearning approaches. End2End learning systems are mainly based on deep learning methods. A safety monitor is usually designed to ensure the safety of each module. AI, artificial intelligence [Color figure can be viewed at wileyonlinelibrary.com]

Given a route planned through the road network, the first task of an autonomous car is to understand and localize itself in the surrounding environment. On the basis of this representation, a continuous path is planned and the future actions of the car are determined by the behavior arbitration system. Finally, a motion control system reactively corrects errors generated in the execution of the planned motion. A review of classical non-AI design methodologies for these four components can be found in Paden, Cáp, Yong, Yershov, and Frazzoli (2016).

Following, we will give an introduction of deep learning and AI technologies used in autonomous driving, as well as surveying different methodologies used to design the hierarchical decision-making process described above. Additionally, we provide an overview of End2End learning systems used to encode the hierarchical process into a single deep learning architecture which directly maps sensory observations to control outputs.

3 | OVERVIEW OF DEEP LEARNING TECHNOLOGIES

In this section, we describe the basis of deep learning technologies used in AVs and comment on the capabilities of each paradigm. We focus on *convolutional neural networks* (CNNs), *recurrent neural networks* (RNNs), and *deep reinforcement learning* (DRL), which are the most common deep learning methodologies applied to autonomous driving.

Throughout the survey, we use the following notations to describe time-dependent sequences. The value of a variable is defined either for a single discrete timestep t, written as superscript $\langle t \rangle$, or as a discrete sequence defined in the $\langle t, t+k \rangle$ time interval, where k denotes the length of the sequence. For example, the value of a state variable $\mathbf{z}$ is defined either at discrete time t, as $\mathbf{z}^{\langle t \rangle}$, or within a sequence interval $\mathbf{z}^{\langle t, t+k \rangle}$. Vectors and matrices are indicated by bold symbols.

3.1 | Deep CNNs

CNNs are mainly used for processing spatial information, such as images, and can be viewed as image features extractors and universal nonlinear function approximators (Bengio, Courville, & Vincent, 2013; Lecun, Bottou, Bengio, & Haffner, 1998). Before the rise of deep learning, computer vision systems used to be implemented based on handcrafted features, such as HAAR (Viola & Jones, 2001), local binary patterns (LBPs; Ojala, Pietikäinen, & Harwood, 1996), or histograms of oriented gradients (HoG; Dalal & Triggs, 2005). In comparison to these traditional handcrafted features, CNNs are able to automatically learn a representation of the feature space encoded in the training set.

CNNs can be loosely understood as very approximate analogies to different parts of the mammalian visual cortex (Hubel & Wiesel, 1963). An image formed on the retina is sent to the visual cortex through the thalamus. Each brain hemisphere has its own visual cortex. The visual information is received by the visual cortex in a crossed manner: The left visual cortex receives information from the right eye, whereas the right visual cortex is fed with visual data from the left eye. The information is processed according to the dual flux theory (Goodale & Milner, 1992), which states that the visual flow follows two main fluxes: A *ventral flux*, responsible for visual identification and object recognition, and a *dorsal flux* used for establishing spatial relations between objects. A CNN mimics the functioning of the ventral flux, in which different areas of the brain are sensible to specific features in the visual field. The earlier brain cells in the visual cortex are activated by sharp transitions in the visual field of view, in the same way in which an edge detector highlights sharp transitions between the neighboring pixels in an image. These edges are further used in the brain to approximate object parts and finally to estimate abstract representations of objects.

A CNN is parametrized by its weights vector $\theta = [\mathbf{W}, \mathbf{b}]$, where $\mathbf{W}$ is the set of weights governing the interneural connections and $\mathbf{b}$ is the set of neuron bias values. The set of weights $\mathbf{W}$ is organized as image filters, with coefficients learned during training. Convolutional layers within a CNN exploit local spatial correlations of image pixels to learn translation-invariant convolution filters, which capture discriminant image features.

Consider a multichannel signal representation $\mathbf{M}_k$ in layer k, which is a channelwise integration of signal representations $\mathbf{M}_{k,c}$, where $c \in \mathbb{N}$. A signal representation can be generated in layer $k+1$ as

$$\mathbf{M}_{k+1,l} = \varphi(\mathbf{M}_k * \mathbf{w}_{k,l} + \mathbf{b}_{k,l}), \tag{1}$$

where $\mathbf{w}_{k,l} \in \mathbf{W}$ is a convolutional filter with the same number of channels as $\mathbf{M}_k$, $\mathbf{b}_{k,l} \in \mathbf{b}$ represents the bias, l is a channel index, and $*$ denotes the convolution operation. $\varphi(\cdot)$ is an activation function applied to each pixel in the input signal. Typically, the rectified linear unit (ReLU) is the most commonly used activation function in computer vision applications (Krizhevsky, Sutskever, & Hinton, 2012). The final layer of a CNN is usually a fully connected layer which acts as an object discriminator on a high-level abstract representation of objects.

In a supervised manner, the response $R(\cdot;\theta)$ of a CNN can be trained using a training database $\mathcal{D} = [(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)]$, where $\mathbf{x}_i$ is a data sample, y_i is the corresponding label, and m is the number of training examples. The optimal network parameters can be calculated using *maximum likelihood estimation* (MLE). For the clarity of explanation, we take as example the simple least-squares error function, which can be used to drive the MLE process when training regression estimators:

$$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta; \mathcal{D}) = \arg\min_{\theta} \sum_{i=1}^{m} (R(\mathbf{x}_i; \theta) - y_i)^2. \tag{2}$$

For classification purposes, the least-squares error is usually replaced by the cross-entropy, or the negative log-likelihood loss functions. The optimization problem in Equation (2) is typically solved with stochastic gradient descent (SGD) and the backpropagation algorithm for gradient estimation (Rumelhart et al., 1986). In practice, different variants of SGD are used, such as Adam (Kingma & Ba, 2015) or AdaGrad (Duchi, Hazan, & Singer, 2011).

3.2 | Recurrent neural networks

Among deep learning techniques, *RNNs* are especially good in processing temporal sequence data, such as text, or video streams. Different from conventional neural networks, an RNN contains a time-dependent feedback loop in its memory cell. Given a time-dependent input sequence $[\mathbf{s}^{\langle t-\tau_i\rangle}, ..., \mathbf{s}^{\langle t\rangle}]$ and an output sequence $[\mathbf{z}^{\langle t+1\rangle}, ..., \mathbf{z}^{\langle t+\tau_o\rangle}]$, an RNN can be "unfolded" $\tau_i + \tau_o$ times to generate a loopless network architecture matching the input length, as illustrated in Figure 2. t represents a temporal index, whereas τ_i and τ_o are the lengths of the input and output sequences, respectively. Such neural networks are also encountered under the name of *sequence-to-sequence models*. An unfolded network has $\tau_i + \tau_o + 1$ identical layers, that is, each layer shares the same learned weights. Once unfolded, an RNN can be trained using the backpropagation through time algorithm. When compared with a conventional neural network, the only difference is that the learned weights in each unfolded copy of the network are averaged, thus enabling the network to share the same weights over time.

The main challenge in using basic RNNs is the vanishing gradient encountered during training. The gradient signal can end up being multiplied a large number of times, as many as the number of timesteps. Hence, a traditional RNN is not suitable for capturing long-term dependencies in sequence data. If a network is very deep, or processes long sequences, the gradient of the network's output would have a hard time in propagating back to affect the weights of the earlier layers. Under gradient vanishing, the weights of the network will not be effectively updated, ending up with very small weight values.

Long–short-term memory (LSTM; Hochreiter & Schmidhuber, 1997) networks are nonlinear function approximators for estimating temporal dependencies in sequence data. As opposed to traditional RNNs, LSTMs solve the vanishing gradient problem by incorporating three gates, which control the input, output, and memory state.

Recurrent layers exploit temporal correlations of sequence data to learn time-dependent neural structures. Consider the memory state $\mathbf{c}^{\langle t-1\rangle}$ and the output state $\mathbf{h}^{\langle t-1\rangle}$ in an LSTM network, sampled at timestep $t-1$, as well as the input data $\mathbf{s}^{\langle t\rangle}$ at time t. The opening or closing of a gate is controlled by a sigmoid function $\sigma(\cdot)$ of the current input signal $\mathbf{s}^{\langle t\rangle}$ and the output signal of the last time point $\mathbf{h}^{\langle t-1\rangle}$:

$$\Gamma_u^{\langle t\rangle} = \sigma(\mathbf{W}_u\mathbf{s}^{\langle t\rangle} + \mathbf{U}_u\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_u), \tag{3}$$

$$\Gamma_f^{\langle t\rangle} = \sigma(\mathbf{W}_f\mathbf{s}^{\langle t\rangle} + \mathbf{U}_f\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_f), \tag{4}$$

$$\Gamma_o^{\langle t\rangle} = \sigma(\mathbf{W}_o\mathbf{s}^{\langle t\rangle} + \mathbf{U}_o\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_o), \tag{5}$$

where $\Gamma_u^{\langle t\rangle}$, $\Gamma_f^{\langle t\rangle}$, and $\Gamma_o^{\langle t\rangle}$ are gate functions of the input gate, forget gate, and output gate, respectively. Given current observation, the memory state $\mathbf{c}^{\langle t\rangle}$ will be updated as

$$\mathbf{c}^{\langle t\rangle} = \Gamma_u^{\langle t\rangle} * \tanh(\mathbf{W}_c\mathbf{s}^{\langle t\rangle} + \mathbf{U}_c\mathbf{h}^{\langle t-1\rangle} + \mathbf{b}_c) + \Gamma_f * \mathbf{c}^{\langle t-1\rangle}. \tag{6}$$

The new network output $\mathbf{h}^{\langle t\rangle}$ is computed as

$$\mathbf{h}^{\langle t\rangle} = \Gamma_o^{\langle t\rangle} * \tanh(\mathbf{c}^{\langle t\rangle}). \tag{7}$$

An LSTM network Q is parametrized by $\theta = [\mathbf{W}_i, \mathbf{U}_i, \mathbf{b}_i]$, where $\mathbf{W}_i$ represents the weights of the network's gates and memory cell multiplied with the input state, $\mathbf{U}_i$ are the weights governing the activations, and $\mathbf{b}_i$ denotes the set of neuron bias values. * symbolizes elementwise multiplication.

In a supervised learning setup, given a set of training sequences $\mathcal{D} = [(\mathbf{s}_1^{\langle t-\tau_i,t\rangle}, \mathbf{z}_1^{\langle t+1,t+\tau_o\rangle}), ..., (\mathbf{s}_q^{\langle t-\tau_i,t\rangle}, \mathbf{z}_q^{<t+1,t+\tau_o>})]$, that is, q independent pairs of observed sequences with assignments $\mathbf{z}^{\langle t,t+\tau_o\rangle}$, one can train the response of an LSTM network $Q(\cdot;\theta)$ using MLE:

$$\begin{aligned}\hat{\theta} &= \arg\max_{\theta} \mathcal{L}(\theta; \mathcal{D}) \\ &= \arg\min_{\theta} \sum_{i=1}^{m} l_i\left(Q\left(\mathbf{s}_i^{\langle t-\tau_i,t\rangle}; \theta\right), \mathbf{z}_i^{\langle t+1,t+\tau_o\rangle}\right), \\ &= \arg\min_{\theta} \sum_{i=1}^{m}\sum_{t=1}^{\tau_o} l_i^{\langle t\rangle}\left(Q^{\langle t\rangle}\left(\mathbf{s}_i^{\langle t-\tau_i,t\rangle}; \theta\right), \mathbf{z}_i^{\langle t\rangle}\right),\end{aligned} \tag{8}$$

where an input sequence of observations $\mathbf{s}^{\langle t-\tau_i,t\rangle} = [\mathbf{s}^{\langle t-\tau_i\rangle}, ..., \mathbf{s}^{\langle t-1\rangle}, \mathbf{s}^{\langle t\rangle}]$ is composed of τ_i consecutive data samples, $l(\cdot,\cdot)$ is the logistic regression loss function, and t represents a temporal index.

In RNNs terminology, the optimization procedure in Equation (8) is typically used for training "many-to-many" RNN architectures, such as the one in Figure 2, where the input and output states are represented by temporal sequences of τ_i and τ_o data instances, respectively. This optimization problem is commonly solved using gradient-based methods, like SGD, together with the backpropagation through time algorithm for calculating the network's gradients.

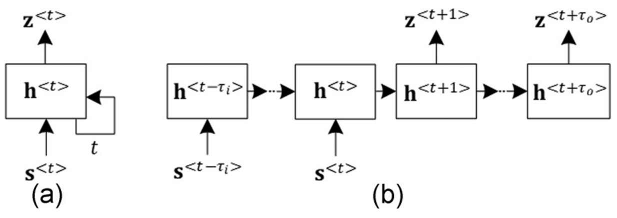

FIGURE 2 A folded (a) and unfolded (b) overtime, many-to-many recurrent neural network. Overtime t, both the input $\mathbf{s}^{\langle t-\tau_i,t\rangle}$ and output $\mathbf{z}^{\langle t+1,t+\tau_o\rangle}$ sequences share the same weights $\mathbf{h}^{\langle\cdot\rangle}$. The architecture is also referred to as *a sequence-to-sequence model*

3.3 | Deep reinforcement learning

In the following, we review the *DRL* concept as an autonomous driving task, using the *partially observable Markov decision process* (POMDP) formalism.

In a POMDP, an agent, which in our case is the self-driving car, senses the environment with observation $\mathbf{I}^{\langle t\rangle}$, performs an action $a^{\langle t\rangle}$ in state $\mathbf{s}^{\langle t\rangle}$, interacts with its environment through a received reward $R^{\langle t+1\rangle}$, and transits to the next state $\mathbf{s}^{\langle t+1\rangle}$ following a transition function $T_{\mathbf{s}^{\langle t\rangle},a^{\langle t\rangle}}^{\mathbf{s}^{\langle t+1\rangle}}$.

In reinforcement learning (RL) based autonomous driving, the task is to learn an optimal driving policy for navigating from state $\mathbf{s}_{\text{start}}^{\langle t\rangle}$ to a destination state $\mathbf{s}_{\text{dest}}^{\langle t+k\rangle}$, given an observation $\mathbf{I}^{\langle t\rangle}$ at time t and the system's state $\mathbf{s}^{\langle t\rangle}$. $\mathbf{I}^{\langle t\rangle}$ represents the observed environment, whereas k is the number of timesteps required for reaching the destination state $\mathbf{s}_{\text{dest}}^{\langle t+k\rangle}$.

In reinforcement learning terminology, the above problem can be modeled as a POMDP $M := (I, S, A, T, R, \gamma)$, where

- I is the set of observations, with $\mathbf{I}^{\langle t\rangle} \in I$ defined as an observation of the environment at time t.
- S represents a finite set of states, $\mathbf{s}^{\langle t\rangle} \in S$ being the state of the agent at time t, commonly defined as the vehicle's position, heading, and velocity.
- A represents a finite set of actions allowing the agent to navigate through the environment defined by $\mathbf{I}^{\langle t\rangle}$, where $a^{\langle t\rangle} \in A$ is the action performed by the agent at time t.
- $T : S \times A \times S \to [0, 1]$ is a stochastic transition function, where $T_{\mathbf{s}^{\langle t\rangle},a^{\langle t\rangle}}^{\mathbf{s}^{\langle t+1\rangle}}$ describes the probability of arriving in state $\mathbf{s}^{\langle t+1\rangle}$, after performing action $a^{\langle t\rangle}$ in state $\mathbf{s}^{\langle t\rangle}$.
- $R : S \times A \times S \to \mathbb{R}$ is a scalar reward function which controls the estimation of a, where $R_{\mathbf{s}^{\langle t\rangle},a^{\langle t\rangle}}^{\mathbf{s}^{\langle t+1\rangle}} \in \mathbb{R}$. For a state transition $\mathbf{s}^{\langle t\rangle} \to \mathbf{s}^{\langle t+1\rangle}$ at time t, we define a scalar reward function $R_{\mathbf{s}^{\langle t\rangle},a^{\langle t\rangle}}^{\mathbf{s}^{\langle t+1\rangle}}$ which quantifies how well did the agent perform in reaching the next state.
- γ is the discount factor controlling the importance of future versus immediate rewards.

Considering the proposed reward function and an arbitrary state trajectory $[\mathbf{s}^{\langle 0\rangle}, \mathbf{s}^{\langle 1\rangle}, ..., \mathbf{s}^{\langle k\rangle}]$ in observation space, at any time $\hat{t} \in [0, 1, ..., k]$, the associated cumulative future discounted reward is defined as

$$R^{\langle \hat{t}\rangle} = \sum_{t=\hat{t}}^{k} \gamma^{\langle t-\hat{t}\rangle} r^{\langle t\rangle}, \tag{9}$$

where the immediate reward at time t is given by $r^{\langle t\rangle}$. In RL theory, the statement in Equation (9) is known as a finite horizon learning episode of sequence length k (Sutton & Barto, 1998).

The objective in RL is to find the desired trajectory policy that maximizes the associated cumulative future reward. We define the optimal action-value function $Q^*(\cdot,\cdot)$ which estimates the maximal future discounted reward when starting in state $\mathbf{s}^{\langle t\rangle}$ and performing actions $[a^{\langle t\rangle}, ..., a^{\langle t+k\rangle}]$:

$$Q^*(\mathbf{s}, a) = \max_{\pi} \mathbb{E}\left[R^{\langle \hat{t}\rangle} | \mathbf{s}^{\langle \hat{t}\rangle} = \mathbf{s}, a^{\langle \hat{t}\rangle} = a, \pi\right], \tag{10}$$

where π is an action policy, viewed as a probability density function over a set of possible actions that can take place in a given state. The optimal action-value function $Q^*(\cdot,\cdot)$ maps a given state to the optimal action policy of the agent in any state:

$$\forall\ \mathbf{s} \in S : \pi^*(\mathbf{s}) = \arg\max_{a \in A} Q^*(\mathbf{s}, a). \tag{11}$$

The optimal action-value function Q^* satisfies the Bellman optimality equation (Bellman, 1957), which is a recursive formulation of Equation (10):

$$\begin{aligned} Q^*(\mathbf{s}, a) &= \sum_{\mathbf{s}} T_{\mathbf{s},a}^{\mathbf{s}'}(R_{\mathbf{s},a}^{\mathbf{s}'} + \gamma \cdot \max_{a'} Q^*(\mathbf{s}', a')) \\ &= \mathbb{E}_{a'}(R_{\mathbf{s},a}^{\mathbf{s}'} + \gamma \cdot \max_{a'} Q^*(\mathbf{s}', a')), \end{aligned} \tag{12}$$

where $\mathbf{s}'$ represents a possible state visited after $\mathbf{s} = \mathbf{s}^{\langle t\rangle}$ and a' is the corresponding action policy. The model-based policy iteration algorithm was introduced in Sutton and Barto (1998), based on the proof that the Bellman equation is a contraction mapping (Watkins & Dayan, 1992) when written as an operator ν:

$$\forall\ Q, \quad \lim_{n\to\infty} \nu^{(n)}(Q) = Q^*. \tag{13}$$

However, the standard reinforcement learning method described above is not feasible in high-dimensional state spaces. In autonomous driving applications, the observation space is mainly composed of sensory information made up of images, radar, LiDAR, and so forth. Instead of the traditional approach, a nonlinear parameterization of Q^* can be encoded in the layers of a deep neural network. In the literature, such a nonlinear approximator is called a deep Q-network (DQN; Mnih et al., 2015) and is used for estimating the approximate action-value function:

$$Q(\mathbf{s}^{\langle t\rangle}, a^{\langle t\rangle}; \Theta) \approx Q^*(\mathbf{s}^{\langle t\rangle}, a^{\langle t\rangle}), \tag{14}$$

where Θ represents the parameters of the DQN.

By taking into account the Bellman optimality equation (12), it is possible to train a DQN in a reinforcement learning manner through the minimization of the mean squared error. The optimal expected Q value can be estimated within a training iteration i based on a set of reference parameters $\bar{\Theta}_i$ calculated in a previous iteration i':

$$y = R_{\mathbf{s},a}^{\mathbf{s}'} + \gamma \cdot \max_{a'} Q(\mathbf{s}', a'; \bar{\Theta}_i), \tag{15}$$

where $\bar{\Theta}_i := \Theta_{i'}$. The new estimated network parameters at training step i are evaluated using the following squared error function:

$$\nabla J_{\hat{\Theta}_i} = \min_{\Theta_i} \mathbb{E}_{\mathbf{s},y,r,\mathbf{s}'}[(y - Q(\mathbf{s}, a; \Theta_i))^2], \tag{16}$$

where $r = R_{\mathbf{s},a}^{\mathbf{s}'}$. On the basis of (16), the MLE function from Equation (8) can be applied for calculating the weights of the DQN. The gradient is approximated with random samples and the back-propagation algorithm, which uses SGD for training:

$$\nabla_{\Theta_i} = \mathbb{E}_{\mathbf{s},a,r,\mathbf{s}'}[(y - Q(\mathbf{s}, a; \Theta_i))\nabla_{\Theta_i}(Q(\mathbf{s}, a; \Theta_i))]. \tag{17}$$

The DRL community has made several independent improvements to the original DQN algorithm (Mnih et al., 2015). A study on how to combine these improvements on DRL has been provided by DeepMind in Hessel et al. (2018), where the combined algorithm, entitled *Rainbow*, was able to outperform the independently competing methods. DeepMind (Hessel et al., 2018) proposes six extensions to the base DQN, each addressing a distinct concern:

- *Double Q-Learning* addresses the overestimation bias and decouples the selection of an action and its evaluation.
- *Prioritized replay* samples more frequently from the data in which there is information to learn.
- *Dueling Networks* aim at enhancing value-based RL.
- *Multistep learning* is used for training speed improvement.
- *Distributional RL* improves the target distribution in the Bellman equation.
- *Noisy Nets* improve the ability of the network to ignore noisy inputs and allows state-conditional exploration.

All of the above complementary improvements have been tested on the Atari 2600 challenge. A good implementation of DQN regarding AVs should start by combining the stated DQN extensions with respect to a desired performance. Given the advancements in DRL, the direct application of the algorithm still needs a training pipeline in which one should simulate and model the desired self-driving car's behavior.

The simulated environment state is not directly accessible to the agent. Instead, sensor readings provide clues about the true state of the environment. To decode the true environment state, it is not sufficient to map a single snapshot of sensors readings. The temporal information should also be included in the network's input, since the environment's state is modified over time. An example of DQN applied to AVs in a simulator can be found in Sallab, Abdou, Perot, and Yogamani (2017a).

DQN has been developed to operate in discrete action spaces. In the case of an autonomous car, the discrete actions would translate to discrete commands, such as turn left, turn right, accelerate, or break. The DQN approach described above has been extended to continuous action spaces based on policy gradient estimation (Lillicrap et al., 2016). The method in Lillicrap et al. (2016) describes a model-free actor-critic algorithm able to learn different continuous control tasks directly from raw pixel inputs. A model-based solution for continuous *Q*-learning is proposed in S. Gu, Lillicrap, Sutskever, and Levine (2016).

Although continuous control with DRL is possible, the most common strategy for DRL in autonomous driving is based on discrete control (Jaritz, Charette, Toromanoff, Perot, & Nashashibi, 2018). The main challenge here is the training, since the agent has to explore its environment, usually through learning from collisions. Such systems, trained solely on simulated data, tend to learn a biased version of the driving environment. A solution here is to use imitation learning (IL) methods, such as inverse reinforcement learning (IRL; Wulfmeier, Wang, & Posner, 2016), to learn from human driving demonstrations without needing to explore unsafe actions.

4 | DEEP LEARNING FOR DRIVING SCENE PERCEPTION AND LOCALIZATION

The self-driving technology enables a vehicle to operate autonomously by perceiving the environment and instrumenting a responsive answer. Following, we give an overview of the top methods used in driving scene understanding, considering camera based versus LiDAR environment perception. We survey object detection and recognition, semantic segmentation and localization in autonomous driving, as well as scene understanding using occupancy maps. Surveys dedicated to autonomous vision and environment perception can be found in Zhu, Yuen, Mihaylova, and Leung (2017) and Janai, Güney, Behl, and Geiger (2017).

4.1 | Sensing hardware: camera versus LiDAR debate

Deep learning methods are particularly well suited for detecting and recognizing objects in two-dimensional (2D) images and 3D point clouds acquired from video cameras and LiDAR devices, respectively.

In the autonomous driving community, 3D perception is mainly based on LiDAR sensors, which provide a direct 3D representation of the surrounding environment in the form of 3D point clouds. The performance of a LiDAR is measured in terms of field of view, range, resolution, and rotation/frame rate. 3D sensors, such as Velodyne®, usually have a 360° horizontal field of view. To operate at high speeds, an AV requires a minimum of 200 m range, allowing the vehicle to react to changes in road conditions in time. The 3D object detection precision is dictated by the resolution of the sensor, with most advanced LiDARs being able to provide a 3-cm accuracy.

Recent debate sparked around camera versus LiDAR sensing technologies. Tesla® and Waymo®, two of the companies leading the development of self-driving technology (O'Kane, 2018), have different philosophies with respect to their main perception sensor, as well as regarding the targeted SAE Level (SAE Committee, 2014). Waymo® is building their vehicles directly as Level 5 systems, with currently more than 10 million miles driven autonomously.[2] On the other hand, Tesla® deploys its AutoPilot as an advanced driver assistance system (ADAS) component, which customers can turn on or off at their convenience. The advantage of Tesla® resides in its large training database, consisting of more than 1 billion driven miles.[3] The database has been acquired by collecting data from customers-owned cars.

The main sensing technologies differ in both companies. Tesla® tries to leverage on its camera systems, whereas Waymo's driving

[2]https://arstechnica.com/cars/2018/10/waymo-has-driven-10-million-miles-on-public-roads-thats-a-big-deal/

[3]https://electrek.co/2018/11/28/tesla-autopilot-1-billion-miles/

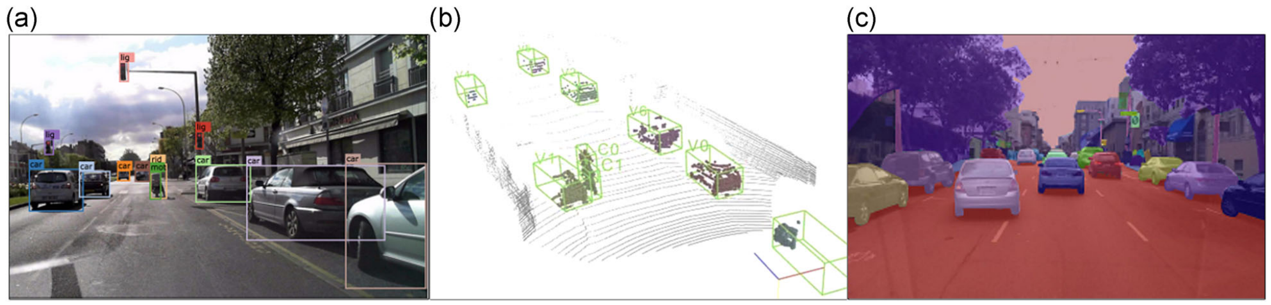

FIGURE 3 Examples of scene perception results. (a) 2D object detection in images, (b) 3D bounding-box detector applied on LiDAR data, and (c) semantic segmentation results on images. 2D, two-dimensional; 3D, three-dimensional [Color figure can be viewed at wileyonlinelibrary.com]

technology relies more on LiDAR sensors.[4] The sensing approaches have advantages and disadvantages. LiDARs have high resolution and precise perception even in the dark, but are vulnerable to bad weather conditions (e.g., heavy rain; Hasirlioglu, Kamann, Doric, & Brandmeier, 2016) and involve moving parts. In contrast, cameras are cost efficient, but lack depth perception and cannot work in the dark. Cameras are also sensitive to bad weather, if the weather conditions are obstructing the field of view.

Researchers at Cornell University tried to replicate LiDAR-like point clouds from visual depth estimation (Wang et al., 2019). An estimated depth map is reprojected into 3D space, with respect to the left sensor's coordinate of a stereo camera. The resulting point cloud is referred to as *pseudo-LiDAR*. The pseudo-LiDAR data can be further fed to 3D deep learning processing methods, such as PointNet (Qi, Su, Mo, & Guibas, 2017) or aggregate view object detection (AVOD; Ku, Mozifian, Lee, Harakeh, & Waslander, 2018). The success of image-based 3D estimation is of high importance to the large-scale deployment of autonomous cars, since the LiDAR is arguably one of the most expensive hardware components in a self-driving vehicle.

Apart from these sensing technologies, radar and ultrasonic sensors are used to enhance perception capabilities. For example, alongside three LiDAR sensors, Waymo also makes use of five radars and eight cameras, whereas Tesla® cars are equipped with eights cameras, 12 ultrasonic sensors, and one forward-facing radar.

4.2 | Driving scene understanding

An autonomous car should be able to detect traffic participants and drivable areas, particularly in urban areas where a wide variety of object appearances and occlusions may appear. Deep learning-based perception, in particular CNNs, became the de facto standard in object detection and recognition, obtaining remarkable results in competitions, such as the ImageNet Large-Scale Visual Recognition Challenge (Russakovsky et al., 2015).

Different neural networks architectures are used to detect objects as 2D regions of interest (Dai, Li, He, & Sun, 2016; Girshick, 2015; Iandola et al., 2016; Law & Deng, 2018; Redmon, Divvala, Girshick, & Farhadi, 2016; S. Zhang, Wen, Bian, Lei, & Li, 2017) or pixelwise segmented areas in images (Badrinarayanan, Kendall, & Cipolla, 2017 ; He, Gkioxari, Dollar, & Girshick, 2017; Treml et al., 2016; H. Zhao, Qi, Shen, Shi, & Jia, 2018), 3D bounding boxes in LiDAR point clouds (Luo, Yang, & Urtasun, 2018; Qi et al., 2017; Zhou & Tuzel, 2018), as well as 3D representations of objects in combined camera-LiDAR data (X. Chen, Ma, Wan, Li, & Xia, 2017; Ku et al., 2018; Qi, Liu, Wu, Su, & Guibas, 2018). Examples of scene perception results are illustrated in Figure 3. Being richer in information, image data are more suited for the object recognition task. However, the real-world 3D positions of the detected objects have to be estimated, since depth information is lost in the projection of the imaged scene onto the imaging sensor.

4.2.1 | Bounding-box-like object detectors

The most popular architectures for 2D object detection in images are single- and double-stage detectors. Popular single-stage detectors are "*You Only Look Once*" (Yolo; Redmon et al., 2016; Redmon & Farhadi, 2017, 2018), the *Single Shot multibox Detector* (SSD; W. Liu et al., 2016), CornerNet (Law & Deng, 2018), and RefineNet (S. Zhang et al., 2017). Double-stage detectors, such as Regions with CNN (R-CNN) (Girshick, Donahue, Darrell, & Malik, 2014), Faster-RCNN (Ren, He, Girshick, & Sun, 2017), or region-based fully convolutional network (R-FCN; Dai et al., 2016), split the object detection process into two parts: region of interest candidates proposals and bounding boxes classification. In general, single-stage detectors do not provide the same performances as double-stage detectors, but are significantly faster.

If in-vehicle computation resources are scarce, one can use detectors, such as SqueezeNet (Iandola et al., 2016 or (J. Li, Peng, & Chang, 2018), which are optimized to run on embedded hardware. These detectors usually have a smaller neural network architecture, making it possible to detect objects using a reduced number of operations, at the cost of detection accuracy.

A comparison between the object detectors described above is given in Figure 4, based on the Pascal visual object classes (VOC) 2012 data set and their measured mean average precision (mAP) with an intersection over union (IoU) value equal to 50 and 75, respectively.

[4]https://www.theverge.com/transportation/2018/4/19/17204044/tesla-waymo-self-driving-car-data-simulation

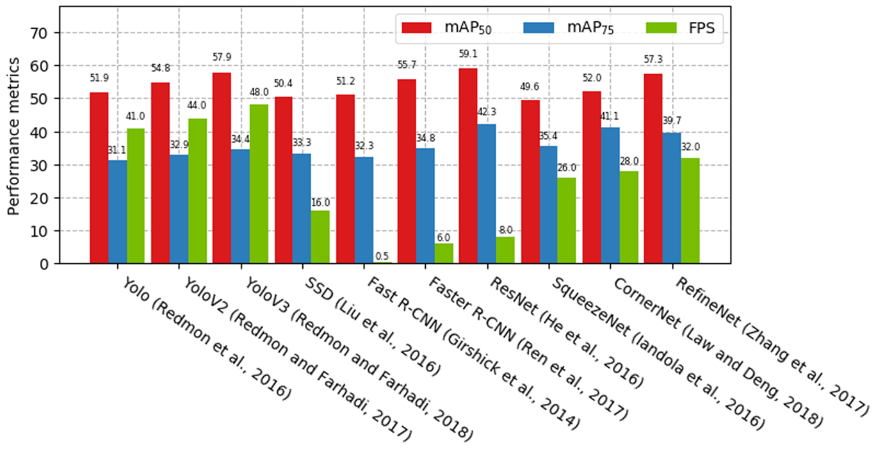

FIGURE 4 Object detection and recognition performance comparison. The evaluation has been performed on the Pascal VOC 2012 benchmarking database. The first four methods on the right represent *single-stage* detectors, whereas the remaining six are *double-stage* detectors. Due to their increased complexity, the runtime performance in frames-per-second (FPS) is lower for the case of double-stage detectors. IoU, intersection over union; mAP, mean average precision; SSD, Single Shot multibox Detector; VOC, visual object classes [Color figure can be viewed at wileyonlinelibrary.com]

A number of publications showcased object detection on raw 3D sensory data, as well as for combined video and LiDAR information. PointNet (Qi et al., 2017) and VoxelNet (Zhou & Tuzel, 2018) are designed to detect objects solely from 3D data, providing also the 3D positions of the objects. However, point clouds alone do not contain the rich visual information available in images. To overcome this, combined camera-LiDAR architectures are used, such as Frustum PointNet (Qi et al., 2018), Multiview 3D networks (MV3D; X. Chen et al., 2017), or RoarNet (Shin, Kwon, & Tomizuka, 2018).

The main disadvantage in using a LiDAR in the sensory suite of a self-driving car is primarily its cost.[5] A solution here would be to use neural network architectures, such as AVOD (Ku et al., 2018), which leverage on LiDAR data only for training, while images are used during training and deployment. At deployment stage, AVOD is able to predict 3D bounding boxes of objects solely from image data. In such a system, a LiDAR sensor is necessary only for training data acquisition, much like the cars used today to gather road data for navigation maps.

4.2.2 | Semantic and instance segmentation

Driving scene understanding can also be achieved using semantic segmentation, representing the categorical labeling of each pixel in an image. In the autonomous driving context, pixels can be marked with categorical labels representing drivable area, pedestrians, traffic participants, buildings, and so forth. It is one of the high-level tasks that paves the way towards complete scene understanding, being used in applications, such as autonomous driving, indoor navigation, or virtual and augmented reality.

Semantic segmentation networks, like SegNet (Badrinarayanan et al., 2017), ICNet (H. Zhao et al., 2018), ENet (Paszke, Chaurasia, Kim, & Culurciello, 2016), AdapNet (Valada, Vertens, Dhall, & Burgard, 2017), or Mask RCNN (He et al., 2017), are mainly encoder–decoder architectures with a pixelwise classification layer. These are based on building blocks from some common network topologies, such as AlexNet (Krizhevsky, Sutskever, & Hinton, 2012), VGG-16 (Simonyan & Zisserman, 2014), GoogLeNet (Szegedy et al., 2015), or ResNet (He, Zhang, Ren, & Sun, 2016).

As in the case of bounding-box detectors, efforts have been made to improve the computation time of these systems on embedded targets. In Treml et al. (2016) and Paszke et al. (2016), the authors proposed approaches to speed up data processing and inference on embedded devices for autonomous driving. Both architectures are light networks providing similar results as SegNet, with a reduced computation cost.

The robustness objective for semantic segmentation was tackled for optimization in AdapNet (Valada et al., 2017). The model is capable of robust segmentation in various environments by adaptively learning features of expert networks based on scene conditions.

A combined bounding-box object detector and semantic segmentation result can be obtained using architectures, such as Mask RCNN (He et al., 2017). The method extends the effectiveness of Faster-RCNN to instance segmentation by adding a branch for predicting an object mask in parallel with the existing branch for bounding-box recognition.

Figure 5 shows tests results performed on four key semantic segmentation networks, based on the CityScapes data set. The per-class mean intersection over union (mIoU) refers to multiclass segmentation, where each pixel is labeled as belonging to a specific object class, whereas per-category mIoU refers to foreground (object)–background (nonobject) segmentation. The input samples have a size of 480 px × 320 px.

4.2.3 | Localization

Localization algorithms aim at calculating the pose (position and orientation) of the AV as it navigates. Although this can be achieved

[5]https://techcrunch.com/2019/03/06/waymo-to-start-selling-standalone-lidar-sensors/

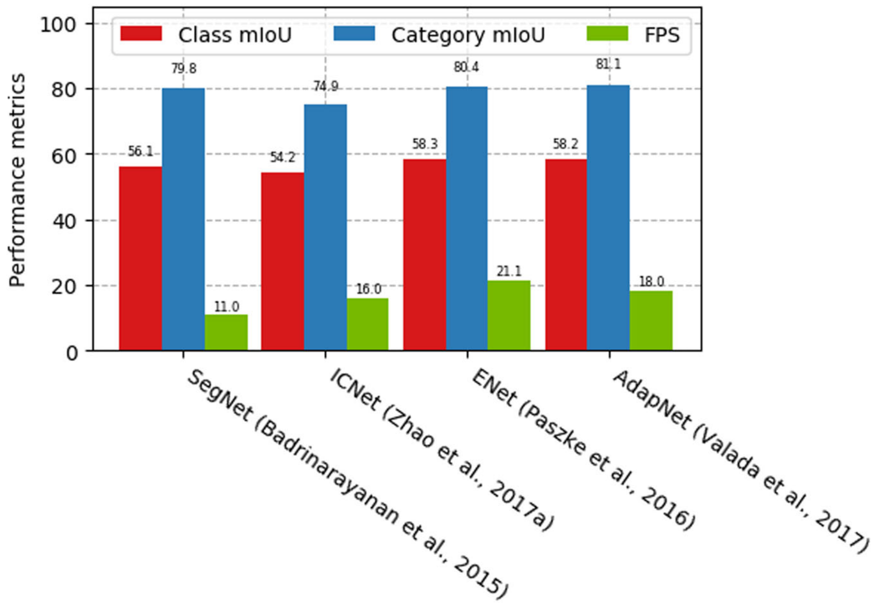

FIGURE 5 Semantic segmentation performance comparison on the CityScapes data set (Cityscapes, 2018). The input samples are 480 px × 320 px images of driving scenes. FPS, frames-per-second; mIoU, mean intersection over union [Color figure can be viewed at wileyonlinelibrary.com]

with systems, such as GPS, in the followings we will focus on deep learning techniques for visual-based localization.

Visual localization, also known as *visual odometry* (VO), is typically determined by matching keypoint landmarks in consecutive video frames. Given the current frame, these keypoints are used as input to a perspective-*n*-point mapping algorithm for computing the pose of the vehicle with respect to the previous frame. Deep learning can be used to improve the accuracy of VO by directly influencing the precision of the keypoints detector. In Barnes, Maddern, Pascoe, and Posner (2018), a deep neural network has been trained for learning keypoints distractors in monocular VO. The so-called learned ephemerality mask acts as a rejection scheme for keypoints outliers which might decrease the vehicle localization's accuracy. The structure of the environment can be mapped incrementally with the computation of the camera pose. These methods belong to the area of *simultaneous localization and mapping* (SLAM). For a survey on classical SLAM techniques, we refer the reader to Bresson, Alsayed, Yu, and Glaser (2017).

Neural networks, such as PoseNet (Kendall, Grimes, & Cipolla, 2015), VLocNet++ (Radwan, Valada, & Burgard, 2018), or the approaches introduced in Walch et al. (2017), Melekhov, Ylioinas, Kannala, and Rahtu (2017), Laskar, Melekhov, Kalia, and Kannala (2017), Brachmann and Rother (2018), or Sarlin, Debraine, Dymczyk, Siegwart, and Cadena (2018), are using image data to estimate the 3D pose of a camera in an End2End fashion. Scene semantics can be derived together with the estimated pose (Radwan et al., 2018).

LiDAR intensity maps are also suited for learning a real-time, calibration-agnostic localization for autonomous cars (Barsan, Wang, Pokrovsky, & Urtasun, 2018). The method uses a deep neural network to build a learned representation of the driving scene from LiDAR sweeps and intensity maps. The localization of the vehicle is obtained through convolutional matching. In Tinchev, Penate-Sanchez, and Fallon (2019), laser scans and a deep neural network are used to learn descriptors for localization in urban and natural environments.

To safely navigate the driving scene, an autonomous car should be able to estimate the motion of the surrounding environment, also known as *scene flow*. Previous LiDAR-based scene flow estimation techniques mainly relied on manually designed features. In recent articles, we have noticed a tendency to replace these classical methods with deep learning architectures able to automatically learn the scene flow. In Ushani and Eustice (2018), an encoding deep network is trained on occupancy grids (OGs) with the purpose of finding matching or nonmatching locations between successive timesteps.

Although much progress has been reported in the area of deep learning-based localization, VO techniques are still dominated by classical keypoints matching algorithms, combined with acceleration data provided by inertial sensors. This is mainly due to the fact that keypoints detectors are computational efficient and can be easily deployed on embedded devices.

4.3 | Perception using occupancy maps

An occupancy map, also known as OG, is a representation of the environment which divides the driving space into a set of cells and calculates the occupancy probability for each cell. Popular in robotics (Garcia-Favrot & Parent, 2009; Thrun, Burgard, & Fox, 2005), the OG representation became a suitable solution for self-driving vehicles. A couple of OG data samples are shown in Figure 6.

Deep learning is used in the context of occupancy maps either for dynamic objects detection and tracking (Ondruska, Dequaire, Wang, & Posner, 2016), probabilistic estimation of the occupancy map surrounding the vehicle (Hoermann, Bach, & Dietmayer, 2017; Ramos, Gehrig, Pinggera, Franke, & Rother, 2016), or for deriving the driving scene context (Marina et al., 2019; Seeger, Müller, & Schwarz, 2016). In the latter case, the OG is constructed by accumulating data over time, whereas a deep neural net is used to

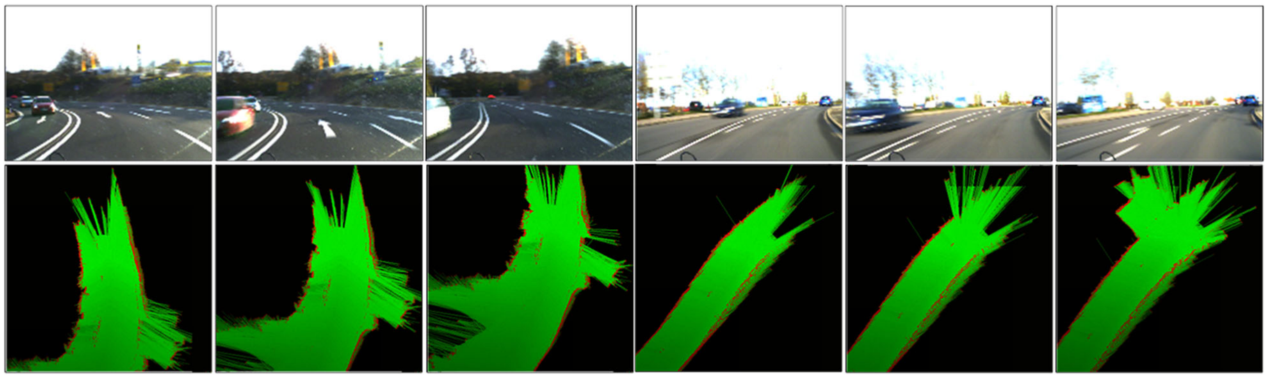

FIGURE 6 Examples of occupancy grids (OGs). The images show a snapshot of the driving environment together with its respective OG (Marina et al., 2019) [Color figure can be viewed at wileyonlinelibrary.com]

label the environment into driving context classes, such as highway driving, parking area, or inner-city driving.

Occupancy maps represent an in-vehicle virtual environment, integrating perceptual information in a form better suited for path planning and motion control. Deep learning plays an important role in the estimation of OG, since the information used to populate the grid cells is inferred from processing image and LiDAR data using scene perception methods, as the ones described in this chapter of the survey.

5 | DEEP LEARNING FOR PATH PLANNING AND BEHAVIOR ARBITRATION

The ability of an autonomous car to find a route between two points, that is, a start position and a desired location, represents *path planning*. According to the path planning process, a self-driving car should consider all possible obstacles that are present in the surrounding environment and calculate a trajectory along a collision-free route. As stated in Shalev-Shwartz, Shammah, and Shashua (2016), autonomous driving is a multiagent setting where the host vehicle must apply sophisticated negotiation skills with other road users when overtaking, giving way, merging, taking left and right turns, all while navigating unstructured urban roadways. The literature findings point to a nontrivial policy that should handle safety in driving. Considering a reward function $R(\bar{s}) = -r$ for an accident event that should be avoided and $R(\bar{s}) \in [-1, 1]$ for the rest of the trajectories, the goal is to learn to perform difficult maneuvers smoothly and safe.

This emerging topic of optimal path planning for autonomous cars should operate at high computation speeds, to obtain short reaction times, while satisfying specific optimization criteria. The survey in Pendleton et al. (2017) provides a general overview of path planning in the automotive context. It addresses the taxonomy aspects of path planning, namely the mission planner, behavior planner, and motion planner. However, Pendleton et al. (2017) do not include a review on deep learning technologies, although the state-of-the-art literature has revealed an increased interest in using deep learning technologies for path planning and behavior arbitration. Following, we discuss two of the most representative deep learning paradigms for path planning, namely IL (Grigorescu, Trasnea, Marina, Vasilcoi, & Cocias, 2019; Rehder, Quehl, & Stiller, 2017; Sun, Peng, Zhan, & Tomizuka, 2018) and DRL-based planning (Paxton, Raman, Hager, & Kobilarov, 2017; L. Yu, Shao, Wei, & Zhou, 2018).

The goal in *IL* (Grigorescu et al., 2019; Rehder et al., 2017; Sun et al., 2018) is to learn the behavior of a human driver from recorded driving experiences (Schwarting, Alonso-Mora, & Rus, 2018). The strategy implies a vehicle teaching process from human demonstration. Thus, the authors employ CNNs to learn planning from imitation. For example, NeuroTrajectory (Grigorescu et al., 2019) is a perception-planning deep neural network that learns the desired state trajectory of the ego-vehicle over a finite prediction horizon. IL can also be framed as an IRL problem, where the goal is to learn the reward function from a human driver (T. Gu, Dolan, & Lee, 2016; Wulfmeier et al., 2016). Such methods use real drivers behaviors to learn reward functions and to generate human-like driving trajectories.

DRL for path planning deals mainly with learning driving trajectories in a simulator (Panov, Yakovlev, & Suvorov, 2018; Paxton et al., 2017; Shalev-Shwartz et al., 2016; L. Yu et al., 2018). The real environmental model is abstracted and transformed into a virtual environment, based on a transfer model. In Shalev-Shwartz et al. (2016), it is stated that the objective function cannot ensure functional safety without causing a serious variance problem. The proposed solution for this issue is to construct a policy function composed of learnable and nonlearnable parts. The learnable policy tries to maximize a reward function (which includes comfort, safety, overtake opportunity, etc.). At the same time, the nonlearnable policy follows the hard constraints of functional safety, while maintaining an acceptable level of comfort.

Both IL and DRL for path planning have advantages and disadvantages. IL has the advantage that it can be trained with data collected from the real-world. Nevertheless, this data is scarce on corner cases (e.g., driving off-lanes, vehicle crashes, etc.), making the trained network's response uncertain when confronted with unseen data. On the other hand, although DRL systems are able to explore different driving situations within a simulated world, these models tend to have a biased behavior when ported to the real-world.

6 | MOTION CONTROLLERS FOR AI-BASED SELF-DRIVING CARS

The motion controller is responsible for computing the longitudinal and lateral steering commands of the vehicle. Learning algorithms are used either as part of *Learning Controllers*, within the motion control module from Figure 1a, or as complete *End2End Control Systems* which directly map sensory data to steering commands, as shown in Figure 1b.

6.1 | Learning controllers

Traditional controllers make use of an a priori model composed of fixed parameters. When robots or other autonomous systems are used in complex environments, such as driving, traditional controllers cannot foresee every possible situation that the system has to cope with. Unlike controllers with fixed parameters, learning controllers make use of training information to learn their models over time. With every gathered batch of training data, the approximation of the true system model becomes more accurate, thus enabling model flexibility, consistent uncertainty estimates and anticipation of repeatable effects and disturbances that cannot be modeled before deployment (Ostafew, Collier, Schoellig, & Barfoot, 2015). Consider the following nonlinear, state-space system:

$$\mathbf{z}^{\langle t+1\rangle} = \mathbf{f}_{\text{true}}(\mathbf{z}^{\langle t\rangle}, \mathbf{u}^{\langle t\rangle}), \tag{18}$$

with observable state $\mathbf{z}^{\langle t\rangle} \in \mathbb{R}^n$ and control input $\mathbf{u}^{\langle t\rangle} \in \mathbb{R}^m$, at discrete time t. The true system $\mathbf{f}_{\text{true}}$ is not known exactly and is approximated by the sum of an a priori model and a learned dynamics model:

$$\mathbf{z}^{\langle t+1\rangle} = \underbrace{\mathbf{f}(\mathbf{z}^{\langle t\rangle}, \mathbf{u}^{\langle t\rangle})}_{\text{a priori model}} + \underbrace{\mathbf{h}(\mathbf{z}^{\langle t\rangle})}_{\text{learned model}}. \tag{19}$$

In previous works, learning controllers have been introduced based on simple function approximators, such as Gaussian process (GP) modeling (Meier, Hennig, & Schaal, 2014; Nguyen-Tuong, Peters, & Seeger, 2008; Ostafew et al., 2015; Ostafew, Schoellig, & Barfoot, 2016) or support vector regression (Sigaud, Salaün, & Padois, 2011).

Learning techniques are commonly used to learn a dynamics model which in turn improves an a priori system model in *iterative learning control* (ILC; Kapania & Gerdes, 2015; Ostafew, Schoellig, & Barfoot, 2013; Panomruttanarug, 2017; Z. Yang, Zhou, Li, & Wang, 2017b) and *model predictive control* (MPC; Drews, Williams, Goldfain, Theodorou, & Rehg, 2017; Lefevre, Carvalho, & Borrelli, 2015; Lefèvre, Carvalho, & Borrelli, 2016; Ostafew et al., 2015, 2016; Pan et al., 2018, 2017; Rosolia, Carvalho, & Borrelli, 2017).

ILC is a method for controlling systems which work in a repetitive mode, such as path tracking in self-driving cars. It has been successfully applied to navigation in off-road terrain (Ostafew et al., 2013), autonomous car parking (Panomruttanarug, 2017), and modeling of steering dynamics in an autonomous race car (Kapania & Gerdes, 2015). Multiple benefits are highlighted, such as the usage of a simple and computationally light feedback controller, as well as a decreased controller design effort (achieved by predicting path disturbances and platform dynamics).

MPC (Rawlings & Mayne, 2009) is a control strategy that computes control actions by solving an optimization problem. It received lots of attention in the last two decades due to its ability to handle complex nonlinear systems with state and input constraints. The central idea behind MPC is to calculate control actions at each sampling time by minimizing a cost function over a short time horizon, while considering observations, input–output constraints and the system's dynamics given by a process model. A general review of MPC techniques for autonomous robots is given in Kamel, Hafez, and Yu (2018).

Learning has been used in conjunction with MPC to learn driving models (Lefevre et al., 2015; Lefèvre et al., 2016), driving dynamics for race cars operating at their handling limits (Drews et al., 2017; Rosolia et al., 2017), as well as to improve path tracking accuracy (Brunner, Rosolia, Gonzales, & Borrelli, 2017; Ostafew et al., 2015, 2016). These methods use learning mechanisms to identify nonlinear dynamics that are used in the MPC's trajectory cost function optimization. This enables one to better predict disturbances and the behavior of the vehicle, leading to optimal comfort and safety constraints applied to the control inputs. Training data are usually in the form of past vehicle states and observations. For example, CNNs can be used to compute a dense OG map in a local robot-centric coordinate system. The grid map is further passed to the MPC's cost function for optimizing the trajectory of the vehicle over a finite prediction horizon.

A major advantage of learning controllers is that they optimally combine traditional model-based control theory with learning algorithms. This makes it possible to still use established methodologies for controller design and stability analysis, together with a robust learning component applied at system identification and prediction levels.

6.2 | End2End learning control

In the context of autonomous driving, End2End learning control is defined as a direct mapping from sensory data to control commands. The inputs are usually from a high-dimensional features space (e.g., images or point clouds). As illustrated in Figure 1b, this is opposed to traditional processing pipelines, where at first objects are detected in the input image, after which a path is planned and finally the computed control values are executed. A summary of some of the most popular End2End learning systems is given in Table 1.

End2End learning can also be formulated as a backpropagation algorithm scaled up to complex models. The paradigm was first introduced in the 1990s, when the autonomous land vehicle in a neural network (ALVINN) system was built (Pomerleau, 1989). ALVINN was designed to follow a predefined road, steering according to the observed road's curvature. The next milestone in End2End driving is considered to be in the mid-2000s, when Darpa Autonomous VEhicle (DAVE) managed to drive through an obstacle-

TABLE 1 Summary of End2End learning methods

Name	Problem space	Neural network architecture	Sensor input	Description
ALVINN (Pomerleau, 1989)	Road following	Three-layer backpropagation network	Camera, laser range finder	ALVINN stands for Autonomous Land Vehicle In a Neural Network. Training has been conducted using simulated road images. Successful tests on the Carnegie Mellon autonomous navigation test vehicle indicate that the network can effectively follow real roads
DAVE (Muller et al., 2006)	DARPA challenge	Six-layer CNN	Raw camera images	A vision-based obstacle avoidance system for off-road mobile robots. The robot is a 50-cm off-road truck, with two front color cameras. A remote computer processes the video and controls the robot via radio
NVIDIA PilotNet (Bojarski et al., 2017)	Autonomous driving in real traffic situations	CNN	Raw camera images	The system automatically learns internal representations of the necessary processing steps, such as detecting useful road features with human steering angle as the training signal
Novel FCN–LSTM (Xu et al., 2017)	Ego-motion prediction	FCN–LSTM	Large-scale video data	A generic vehicle motion model from large-scale crowd-sourced video data is obtained, while developing an end-to-end trainable architecture (FCN–LSTM) for predicting a distribution of future vehicle ego-motion data
Novel C-LSTM (Eraqi et al., 2017)	Steering angle control	C-LSTM	Camera frames, steering wheel angle	C-LSTM is end-to-end trainable, learning both visual and dynamic temporal dependencies of driving. Additionally, the steering angle regression problem is considered classification while imposing a spatial relationship between the output layer neurons
Drive360 (Hecker et al., 2018)	Steering angle and velocity control	CNN + fully connected (FC) + LSTM	Surround-view cameras, controller area network (CAN) bus reader	The sensor setup provides data for a 360° view of the area surrounding the vehicle. A new driving data set is collected, covering diverse scenarios. A novel driving model is developed by integrating the surround-view cameras with the route planner
Deep neural network (DNN) policy (Rausch et al., 2017)	Steering angle control	CNN + FC	Camera images	The trained neural net directly maps pixel data from a front-facing camera to steering commands and does not require any other sensors. We compare the controller performance with the steering behavior of a human driver
DeepPicar (Bechtel et al., 2018)	Steering angle control	CNN	Camera images	DeepPicar is a small-scale replica of a real self-driving car called DAVE-2 by NVIDIA. It uses the same network architecture and can drive itself in real-time using a web camera and a Raspberry Pi 3
TORCS DRL (Sallab et al., 2017b)	Lane keeping and obstacle avoidance	DQN + RNN + CNN	TORCS simulator images	It incorporates recurrent neural networks for information integration, enabling the car to handle partially observable scenarios. It also reduces the computational complexity for deployment on embedded hardware
TORCS E2E (S. Yang, Wang, Liu, Deng, & Hedrick, 2017a)	Steering angle control in a simulated environment (TORCS)	CNN	TORCS simulator images	The image features are split into three categories (sky-, roadside-, and road-related features). Two experimental frameworks are used to investigate the importance of each single feature for training a CNN controller

(Continues)

TABLE 1 (Continued)

Name	Problem space	Neural network architecture	Sensor input	Description
Agile autonomous driving (Pan et al., 2018)	Steering angle and velocity control for aggressive driving	CNN	Raw camera images	A CNN, referred to as the learner, is trained with optimal trajectory examples provided at training time by an MPC controller. The MPC acts as an expert, encoding the scene dynamics into the layers of the neural network
WRC6 AD (Jaritz et al., 2018)	Driving in a racing game	CNN + LSTM Encoder	WRC6 racing game	An asynchronous advantage ActorCritic (A3C) framework is used to learn the car control in a physically and graphically realistic rally game, with the agents evolving simultaneously on different tracks

Abbreviations: C-LSTM, convolutional long–short-term memory; CNN, convolutional neural network; DARPA, Defense Advanced Research Projects Agency; DAVE, Darpa Autonomous VEhicle; DQN, deep Q-network; DRL, deep reinforcement learning; E2E, End2End; FCN, fully convolutional network; LSTM, long–short-term memory; MPC, model predictive control; R-FCN, region-based fully convolutional network; RNN, recurrent neural network; WRC6, World Rally Championship 6.

filled road, after it has been trained on hours of human driving acquired in similar, but not identical, driving scenarios (Muller et al., 2006). Over the last couple of years, the technological advances in computing hardware have facilitated the usage of End2End learning models. The backpropagation algorithm for gradient estimation in deep networks is now efficiently implemented on parallel graphic processing units (GPUs). This kind of processing allows the training of large and complex network architectures, which in turn require huge amounts of training samples (see Section 8).

End2End control papers mainly employ either deep neural networks trained offline on real-world and/or synthetic data (Bechtel et al., 2018; Bojarski et al., 2016; C. Chen, Seff, Kornhauser, & Xiao, 2015; Eraqi et al., 2017; Fridman et al., 2017; Hecker et al., 2018; Rausch et al., 2017; Xu et al., 2017; S. Yang et al., 2017a), or DRL systems trained and evaluated in simulation (Jaritz et al., 2018; Perot, Jaritz, Toromanoff, & Charette, 2017; Sallab et al., 2017b). Methods for porting simulation trained DRL models to real-world driving have also been reported (Wayve, 2018), as well as DRL systems trained directly on real-world image data (Pan et al., 2017, 2018).

End2End methods have been popularized in the last couple of years by NVIDIA®, as part of the *PilotNet* architecture. The approach is to train a CNN which maps raw pixels from a single front-facing camera directly to steering commands (Bojarski et al., 2016). The training data are composed of images and steering commands collected in driving scenarios performed in a diverse set of lighting and weather conditions, as well as on different road types. Before training, the data are enriched using augmentation, adding artificial shifts and rotations to the original data.

PilotNet has 250,000 parameters and approximately 27 million connections. The evaluation is performed in two stages: first in simulation and second in a test car. An *autonomy* performance metric represents the percentage of time when the neural network drives the car:

$$\text{Autonomy} = \left(1 - \frac{(\text{no. of interventions})*6\,\text{s}}{\text{elapsed time (s)}}\right) * 100. \qquad (20)$$

An intervention is considered to take place when the simulated vehicle departs from the center line by more than 1 m, assuming that 6 s is the time needed by a human to retake control of the vehicle and bring it back to the desired state. An autonomy of 98% was reached on a 20-km drive from Holmdel to Atlantic Highlands, NJ. Through training, PilotNet learns how the steering commands are computed by a human driver (Bojarski et al., 2017). The focus is on determining which elements in the input traffic image have the most influence on the network's steering decision. A method for finding the salient object regions in the input image is described, while reaching the conclusion that the low-level features learned by PilotNet are similar to the ones that are relevant to a human driver.

End2End architectures similar to PilotNet, which map visual data to steering commands, have been reported in Rausch et al. (2017), Bechtel et al. (2018), and C. Chen et al. (2015). In Xu et al. (2017), autonomous driving is formulated as a future ego-motion prediction problem. The introduced FCN–LSTM method is designed to jointly train pixel-level supervised tasks using a fully convolutional encoder, together with motion prediction through a temporal encoder. The combination between visual temporal dependencies of the input data has also been considered in Eraqi et al. (2017), where the convolutional long–short-term memory (C-LSTM) network has been proposed for steering control. In Hecker et al. (2018), surround-view cameras were used for End2End learning. The claim is that human drivers also use rear- and side-view mirrors for driving, thus all the information from around the vehicle needs to be gathered and integrated into the network model to output a suitable control command.

To carry out an evaluation of the Tesla® AutoPilot system, Fridman et al. (2017) proposed an End2End CNN framework. It is designed to determine differences between AutoPilot and its own output, taking into consideration edge cases. The network was trained using real data, collected from over 420 hr of real road driving. The comparison between Tesla®'s AutoPilot and the proposed framework was done in real-time on a Tesla® car. The evaluation revealed an accuracy of 90.4% in detecting differences between both systems and the control transfer of the car to a human driver.

Another approach to design End2End driving systems is DRL. This is mainly performed in simulation, where an autonomous agent can safely explore different driving strategies. In Sallab et al. (2017b), a DRL End2End system is used to compute steering command in the TORCS game simulation engine. Considering a more complex virtual environment, Perot et al. (2017) proposed an asynchronous advantage ActorCritic (A3C) method for training a CNN on images and vehicle velocity information. The same idea has been enhanced in Jaritz et al. (2018), having a faster convergence and permissiveness for more generalization. Both articles rely on the following procedure: receiving the current state of the game, deciding on the next control commands and then getting a reward on the next iteration. The experimental setup benefited from a realistic car game, namely World Rally Championship 6, and also from other simulated environments, like TORCS.

The next trend in DRL-based control seems to be the inclusion of classical model-based control techniques, as the ones detailed in Section 6.1. The classical controller provides a stable and deterministic model on top of which the policy of the neural network is estimated. In this way, the hard constraints of the modeled system are transferred into the neural network policy (T. Zhang, Kahn, Levine, & Abbeel, 2016). A DRL policy trained on real-world image data has been proposed in Pan et al. (2017, 2018) for the task of aggressive driving. In this case, a CNN, referred to as the learner, is trained with optimal trajectory examples provided at training time by a model predictive controller.

7 | SAFETY OF DEEP LEARNING IN AUTONOMOUS DRIVING

Safety implies the absence of the conditions that cause a system to be dangerous (Ferrel, 2010). Demonstrating the safety of a system which is running deep learning techniques depends heavily on the type of technique and the application context. Thus, reasoning about the safety of deep learning techniques requires:

- understanding the impact of the possible failures;
- understanding the context within the wider system;
- defining the assumption regarding the system context and the environment in which it will likely be used;
- defining what a safe behavior means, including nonfunctional constraints.

In Burton, Gauerhof, and Heinzemann (2017), an example is mapped on the above requirements with respect to a deep learning component. The problem space for the component is pedestrian detection with CNNs. The top-level task of the system is to locate an object of class person from a distance of 100 m, with a lateral accuracy of ±20 cm, a false negative rate of 1%, and a false positive rate of 5%. The assumptions are that the braking distance and the speed are sufficient to react when detecting persons which are 100 m ahead of the planned trajectory of the vehicle. Alternative sensing methods can be used to reduce the overall false negative and false positive rates of the system to an acceptable level. The context information is that the distance and the accuracy shall be mapped to the dimensions of the image frames presented to the CNN.

There is no commonly agreed definition for the term *safety* in the context of machine learning or deep learning. In Varshney (2016), Varshney defines safety in terms of risk, epistemic uncertainty, and the harm incurred by unwanted outcomes. He then analyzes the choice of cost function and the appropriateness of minimizing the empirical average training cost. Amodei et al. (2016) take into consideration the problem of *accidents* in machine learning systems. Such accidents are defined as unintended and harmful behaviors that may emerge from a poor AI system design. The authors present a list of five practical research problems related to accident risk, categorized according to whether the problem originates from having the wrong objective function (*avoiding side effects* and *avoiding reward hacking*), an objective function that is too expensive to evaluate frequently (*scalable supervision*), or undesirable behavior during the learning process (*safe exploration* and *distributional shift*).

Enlarging the scope of safety, Möller (2012) proposes a decision-theoretic definition of safety that applies to a broad set of domains and systems. They define safety to be the reduction or minimization of risk and epistemic uncertainty associated with unwanted outcomes that are severe enough to be seen as harmful. The keypoints in this definition are: (a) the cost of unwanted outcomes has to be sufficiently high in some human sense for events to be harmful, and (b) safety involves reducing both the probability of expected harms, as well as the possibility of unexpected harms.

Regardless of the above empirical definitions and possible interpretations of safety, the use of deep learning components in safety-critical systems is still an open question. The ISO 26262 standard for functional safety of road vehicles provides a comprehensive set of requirements for assuring safety, but does not address the unique characteristics of deep learning-based software. Salay, Queiroz, and Czarnecki (2017) address this gap by analyzing the places where machine learning can impact the standard and provides recommendations on how to accommodate this impact. These recommendations are focused towards the direction of identifying the hazards, implementing tools and mechanism for fault and failure situations, but also ensuring complete training data sets and designing a multilevel architecture. The usage of specific techniques for various stages within the software development life-cycle is desired.

The ISO 26262 standard recommends the use of a Hazard Analysis and Risk Assessment (HARA) method to identify hazardous events in the system and to specify safety goals that mitigate the hazards. The standard has 10 parts. Our focus is on Part 6: *product development at the software level*, the standard following the well-known V model for engineering. Automotive safety integrity level (ASIL) refers to a risk classification scheme defined in ISO 26262 for an item (e.g., subsystem) in an automotive system.

ASIL represents the degree of rigor required (e.g., testing techniques, types of documentation required, etc.) to reduce risk, where ASIL D represents the highest and ASIL A the lowest risk. If an element is assigned to quality management (QM), it does not require safety management. The ASIL assessed for a given hazard is at first assigned to the safety goal set to address the hazard and is then inherited by the safety requirements derived from that goal (Salay et al., 2017).

According to ISO 26226, a *hazard* is defined as “potential source of harm caused by a malfunctioning behavior, where harm is a physical injury or damage to the health of a person” (Bernd et al., 2012). Nevertheless, a deep learning component can create new types of hazards. An example of such a hazard is usually happening because humans think that the automated driver assistance (often developed using learning techniques) is more reliable than it actually is (Parasuraman & Riley, 1997).

Due to its complexity, a deep learning component can fail in unique ways. For example, in DRL systems, faults in the reward function can negatively affect the trained model (Amodei et al., 2016). In such a case, the automated vehicle figures out that it can avoid getting penalized for driving too close to other vehicles by exploiting certain sensor vulnerabilities so that it *cannot see* how close it is getting. Although hazards such as these may be unique to DRL components, they can be traced to faults, thus fitting within the existing guidelines of ISO 26262.

A key requirement for analyzing the safety of deep learning components is to examine whether immediate human costs of outcomes exceed some harm severity thresholds. Undesired outcomes are truly harmful in a human sense and their effect is felt in near real-time. These outcomes can be classified as safety issues. The cost of deep learning decisions is related to optimization formulations which explicitly include a loss function L. The loss function $L: X \times Y \times Y \rightarrow R$ is defined as the measure of the error incurred by predicting the label of an observation x as $f(x)$, instead of y. Statistical learning calls the risk of f as the expected value of the loss of f under P:

$$R(f) = \int L(x, f(x), y)\, dP(x, y), \tag{21}$$

where $X \times Y$ is a random example space of observations x and labels y, distributed according to a probability distribution $P(X, Y)$. The statistical learning problem consists of finding the function f that optimizes (i.e., minimizes) the risk R (Jose, 2018). For an algorithm's hypothesis h and loss function L, the expected loss on the training set is called the *empirical risk* of h:

$$\mathbf{R}_{\text{emp}}(h) = \frac{1}{m}\sum_{i=1}^{m} L(x^{(i)}, h(x)^{(i)}, y^{(i)}). \tag{22}$$

A machine learning algorithm then optimizes the empirical risk on the expectation that the risk decreases significantly. However, this standard formulation does not consider the issues related to the uncertainty that is relevant for safety. The distribution of the training samples $(x_1, y_1), ..., (x_m, y_m)$ is drawn from the true underlying probability distribution of (X, Y), which may not always be the case. Usually the probability distribution is unknown, precluding the use of domain adaptation techniques (Caruana et al., 2015; Daumé & Marcu, 2006). This is one of the epistemic uncertainties that is relevant for safety because training on a data set of different distributions can cause much harm through bias.

In reality, a machine learning system only encounters a finite number of test samples and an actual operational risk is an empirical quantity on the test set. The operational risk may be much larger than the actual risk for small cardinality test sets, even if h is risk-optimal. This uncertainty caused by the instantiation of the test set can have large safety implications on individual test samples (Varshney & Alemzadeh, 2016).

Faults and failures of a programmed component (e.g., one using a formal algorithm to solve a problem) are totally different from the ones of a deep learning component. Specific faults of a deep learning component can be caused by unreliable or noisy sensor signals (video signal due to bad weather, radar signal due to absorbing construction materials, GPS data, etc.), neural network topology, learning algorithm, training set or unexpected changes in the environment (e.g., unknown driving scenes or accidents on the road). We must mention the first autonomous driving accident, produced by a Tesla® car, where, due to object misclassification errors, the AutoPilot function collided the vehicle into a truck (Levin, 2018). Despite the 130 million miles of testing and evaluation, the accident was caused under extremely rare circumstances, also known as Black Swans, given the height of the truck, its white color under bright sky, combined with the positioning of the vehicle across the road.

Self-driving vehicles must have fail-safe mechanisms, usually encountered under the name of *Safety Monitors*. These must stop the autonomous control software once a failure is detected (Koopman, 2017). Specific fault types and failures have been cataloged for neural networks in Kurd, Kelly, and Austin (2007), Harris (2016), and

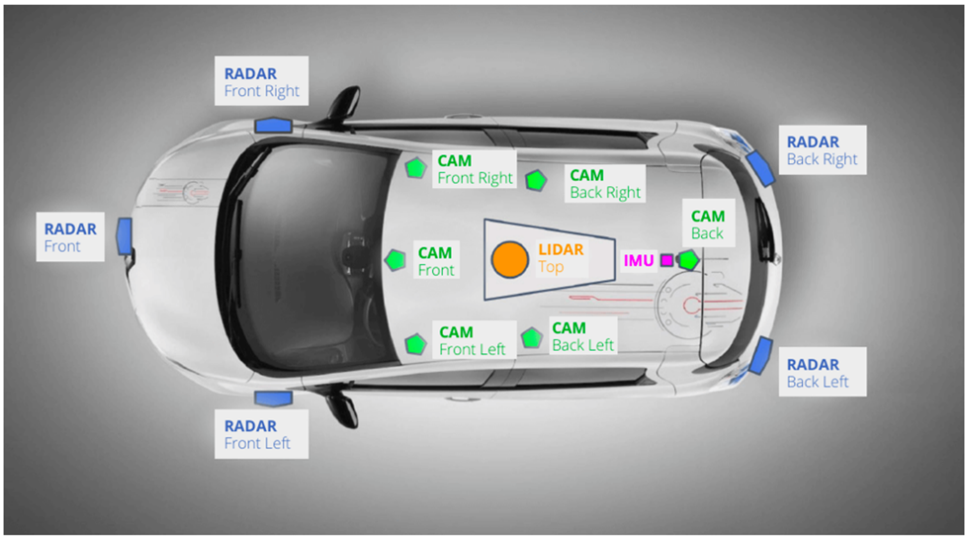

FIGURE 7 Sensor suite of the nuTonomy® self-driving car (Caesar et al., 2019) [Color figure can be viewed at wileyonlinelibrary.com]

McPherson (2018). This led to the development of specific and focused tools and techniques to help finding faults. Chakarov, Nori, Rajamani, Sen, and 2015 Vijaykeerthy (2018) describe a technique for debugging misclassifications due to bad training data, whereas an approach for troubleshooting faults due to complex interactions between linked machine learning components is proposed in Nushi, Kamar, Horvitz, and Kossmann (2017). In Takanami, Sato, and Yang (2000), a white-box technique is used to inject faults onto a neural network by breaking the links or randomly changing the weights.

The training set plays a key role in the safety of the deep learning component. ISO 26262 standard states that the component behavior shall be fully specified and each refinement shall be verified with respect to its specification. This assumption is violated in the case of a deep learning system, where a training set is used instead of a specification. It is not clear how to ensure that the corresponding hazards are always mitigated. The training process is not a verification process since the trained model will be correct by construction with respect to the training set, up to the limits of the model and the learning algorithm (Salay et al., 2017). Effects of this considerations are visible in the commercial AV market, where Black Swan events caused by data not present in the training set may lead to fatalities (McPherson, 2018).

Detailed requirements shall be formulated and traced to hazards. Such a requirement can specify how the training, validation, and testing sets are obtained. Subsequently, the data gathered can be verified with respect to this specification. Furthermore, some specifications, for example, the fact that a vehicle cannot be wider than 3 m, can be used to reject false positive detections. Such properties are used even directly during the training process to improve the accuracy of the model (Katz, Barrett, Dill, Julian, & Kochenderfer, 2017).

Machine learning and deep learning techniques are starting to become effective and reliable even for safety-critical systems, even if the complete safety assurance for this type of systems is still an open question. Current standards and regulation from the automotive industry cannot be fully mapped to such systems, requiring the development of new safety standards targeted for deep learning.

8 | DATA SOURCES FOR TRAINING AUTONOMOUS DRIVING SYSTEMS

Undeniably, the usage of real-world data is a key requirement for training and testing an autonomous driving component. The high amount of data needed in the development stage of such components made data collection on public roads a valuable activity. To obtain a comprehensive description of the driving scene, the vehicle used for data collection is equipped with a variety of sensors, such as radar, LiDAR, GPS, cameras, inertial measurement units (IMU), and ultrasonic sensors. The sensors setup differs from vehicle to vehicle, depending on how the data are planned to be used. A common sensor setup for an AV is presented in Figure 7.

In the last years, mainly due to the large and increasing research interest in AVs, many driving data sets were made public and documented. They vary in size, sensor setup, and data format. The researchers need only to identify the proper data set which best fits their problem space. Janai et al. (2017) published a survey on a broad spectrum of data sets. These data sets address the computer vision field in general, but there are few of them which fit to the autonomous driving topic.

A most comprehensive survey on publicly available data sets for self-driving vehicles algorithms can be found in Yin and Berger (2017). The paper presents 27 available data sets containing data recorded on public roads. The data sets are compared from different perspectives, such that the reader can select the one best suited for his task.

Despite our extensive search, we are yet to find a master data set that combines at least parts of the ones available. The reason may be

TABLE 2 Summary of data sets for training autonomous driving systems

Data set	Problem space	Sensor setup	Size	Location	Traffic condition	License
NuScenes (Caesar et al., 2019)	3D tracking, 3D object detection	Radar, LiDAR, EgoData, GPS, IMU, camera	345 GB (1,000 scenes, clips of 20 s)	Boston, Singapore	Urban	CC BY-NC-SA 3.0
AMUSE (Koschorrek et al., 2013)	SLAM	Omnidirectional camera, IMU, EgoData, GPS	1 TB (7 clips)	Los Angeles	Urban	CC BY-NC-ND 3.0
Ford (Pandey et al., 2011)	3D tracking, 3D object detection	Omnidirectional camera, IMU, LiDAR, GPS	100 GB	Michigan	Urban	Not specified
KITTI (Geiger et al., 2013)	3D tracking, 3D object detection, SLAM	Monocular cameras, IMU LiDAR, GPS	180 GB	Karlsruhe	Urban, rural	CC BY-NC-SA 3.0
Udacity (Udacity, 2018)	3D tracking, 3D object detection	Monocular cameras, IMU, LiDAR, GPS, EgoData	220 GB	Mountain view	Rural	MIT
Cityscapes (Cityscapes, 2018)	Semantic understanding	Color stereo cameras	63 GB (5 clips)	Darmstadt, Zurich, Strasbourg	Urban	CC BY-NC-SA 3.0
Oxford (Maddern et al., 2017)	3D tracking, 3D object detection, SLAM	Stereo and monocular cameras, GPS LiDAR, IMU	23 TB (133 clips)	Oxford	Urban, highway	CC BY-NC-SA 3.0
CamVid (Brostow et al., 2009)	Object detection, segmentation	Monocular color camera	8 GB (4 clips)	Cambridge	Urban	N/A
Daimler pedestrian (Flohr & Gavrila, 2013)	Pedestrian detection, classification, segmentation, path prediction	Stereo and monocular cameras	91 GB (8 clips)	Amsterdam, Beijing	Urban	N/A
Caltech (Dollar et al., 2009)	Tracking, segmentation, object detection	Monocular camera	11 GB	Los Angeles	Urban	N/A

Abbreviations: AMUSE, automotive multisensor data set; CamVid, Cambridge-driving labeled video data set; GPS, global positioning system; IMU, inertial measurement unit; LiDAR, light detection and ranging; SLAM, simultaneous localization and mapping; 3D, three-dimensional.

that there are no standard requirements for the data format and sensor setup. Each data set heavily depends on the objective of the algorithm for which the data were collected. Recently, the companies Scale® and nuTonomy® started to create one of the largest and most detailed self-driving data sets in the market to date.[6] This includes Berkeley DeepDrive (F. Yu et al., 2018), a data set developed by researchers at Berkeley University. More relevant data sets from the literature are pending for merging.[7]

In Fridman et al. (2017), the authors present a study that seeks to collect and analyze large-scale naturalistic data of semi-autonomous driving to better characterize the state-of-the-art of the current technology. The study involved 99 participants, 29 vehicles, 405, 807 miles, and approximately 5.5 billion video frames. Unfortunately, the data collected in this study are not available for the public.

In the remaining of this section we will provide and highlight the distinctive characteristics of the most relevant data sets that are publicly available (Table 2).

KITTI Vision Benchmark data set (*KITTI*; Geiger et al., 2013): Provided by the Karlsruhe Institute of Technology (KIT) from Germany, this data set fits the challenges of benchmarking stereo-vision, optical flow, 3D tracking, 3D object detection, or SLAM algorithms. It is known as the most prestigious data set in the self-driving vehicles domain. To this date it counts more than 2,000 citations in the literature. The data collection vehicle is equipped with multiple high-resolution color and gray-scale stereo cameras, a Velodyne 3D LiDAR and high-precision GPS/IMU sensors. In total, it provides 6 hr of driving data collected in both rural and highway traffic scenarios around Karlsruhe. The data set is provided under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License.

NuScenes data set (Caesar et al., 2019): Constructed by nuTonomy, this data set contains 1,000 driving scenes collected from Boston and Singapore, two known for their dense traffic and highly challenging driving situations. To facilitate common computer vision tasks, such as object detection and tracking, the providers annotated 25 object classes with accurate 3D bounding boxes at 2 Hz over the entire data set. Collection of vehicle data is still in progress. The final data set will include approximately 1.4 million camera images, 400,000 LiDAR sweeps, 1.3 million RADAR sweeps, and 1.1 million object bounding boxes in 40,000 keyframes. The data set is provided under the Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License.

Automotive multisensor data set (*AMUSE*; Koschorrek et al., 2013): Provided by Linköping University of Sweden, it consists of sequences recorded in various environments from a car equipped with an omnidirectional multicamera, height sensors, an IMU, a velocity sensor, and a GPS. The application programming interface (API) for reading these data sets is provided to the public, together with a collection of long multisensor and multicamera data streams stored in the given format. The data set is provided under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unsupported License.

Ford campus vision and LiDAR data set (*Ford*; Pandey et al., 2011): Provided by University of Michigan, this data set was collected using a Ford F250 pickup truck equipped with professional (Applanix POS-LV) and a consumer (Xsens MTi-G) IMUs, a Velodyne LiDAR scanner, two push-broom forward looking Riegl LiDARs and a Point Grey Ladybug3 omnidirectional camera system. The approximately 100 GB of data was recorded around the Ford Research campus and downtown Dearborn, MI in 2009. The data set is well suited to test various autonomous driving and SLAM algorithms.

Udacity data set (Udacity, 2018): The vehicle sensor setup contains monocular color cameras, GPS and IMU sensors, as well as a Velodyne 3D LiDAR. The size of the data set is 223 GB. The data are labeled and the user is provided with the corresponding steering angle that was recorded during the test runs by the human driver.

Cityscapes data set (Cityscapes, 2018): Provided by Daimler AG R&D, Germany; Max Planck Institute for Informatics (MPI-IS), Germany, TU Darmstadt Visual Inference Group, Germany, the Cityscapes data set focuses on semantic understanding of urban street scenes, this being the reason for which it contains only stereo-vision color images. The diversity of the images is very large: 50 cities, different seasons (spring, summer, and fall), various weather conditions, and different scene dynamics. There are 5,000 images with fine annotations and 20,000 images with coarse annotations. Two important challenges have used this data set for benchmarking the development of algorithms for semantic segmentation (H. Zhao, Shi, Qi, Wang, & Jia, 2017) and instance segmentation (S. Liu, Jia, Fidler, & Urtasun, 2017).

The Oxford data set (Maddern et al., 2017): Provided by Oxford University, UK, the data set collection spanned over 1 year, resulting in over 1,000 km of recorded driving with almost 20 million images collected from six cameras mounted to the vehicle, along with LiDAR, GPS, and INS ground truth. Data were collected in all weather conditions, including heavy rain, night, direct sunlight, and snow. One of the particularities of this data set is that the vehicle frequently drove the same route over the period of a year to enable researchers to investigate long-term localization and mapping for AVs in real-world, dynamic urban environments.

The Cambridge-driving Labeled Video data set (*CamVid*; Brostow et al., 2009): Provided by the University of Cambridge, UK, it is one of the most cited data sets from the literature and the first released publicly, containing a collection of videos with object class semantic labels, along with metadata annotations. The database provides ground truth labels that associate each pixel with one of 32 semantic classes. The sensor setup is based on only one monocular camera mounted on the dashboard of the vehicle. The complexity of the scenes is quite low, the vehicle being driven only in urban areas with relatively low-traffic and good-weather conditions.

The Daimler pedestrian benchmark data set (Flohr & Gavrila, 2013): Provided by Daimler AG R&D and University of Amsterdam, this data set fits the topics of pedestrian detection, classification,

[6]https://venturebeat.com/2018/09/14/scale-and-nutonomy-release-nuscenes-a-self-driving-dataset-with-over-1-4-million-images/

[7]https://scale.com/open-datasets

segmentation, and path prediction. Pedestrian data are observed from a traffic vehicle by using only on-board mono and stereo cameras. It is the first data set which contains pedestrians. Recently, the data set was extended with cyclist video samples captured with the same setup (X. Li et al., 2016).

Caltech pedestrian detection data set (*Caltech*; Dollar et al., 2009): Provided by California Institute of Technology, US, the data set contains richly annotated videos, recorded from a moving vehicle, with challenging images of low resolution and frequently occluded people. There is approximately 10 hr of driving scenarios cumulating about 250,000 frames with a total of 350 thousand bounding boxes and 2,300 unique pedestrians annotations. The annotations include both temporal correspondences between bounding boxes and detailed occlusion labels.

Given the variety and complexity of the available databases, choosing one or more to develop and test an autonomous driving component may be difficult. As it can be observed, the sensor setup varies among all the available databases. For localization and vehicle motion, the LiDAR and GPS/IMU sensors are necessary, with the most popular LiDAR sensors used being Velodyne (Velodyne, 2018) and Sick (Sick, 2018). Data recorded from a radar sensor is present only in the NuScenes data set. The radar manufacturers adopt proprietary data formats which are not public. Almost all available data sets include images captured from a video camera, while there is a balance use of monocular and stereo cameras mainly configured to capture gray-scale images. AMUSE and Ford databases are the only ones that use omnidirectional cameras.

Besides raw recorded data, the data sets usually contain miscellaneous files, such as annotations, calibration files, labels, and so forth. To cope with this files, the data set provider must offer tools and software that enable the user to read and postprocess the data. Splitting of the data sets is also an important factor to consider, because some of the data sets (e.g., Caltech, Daimler, and Cityscapes) already provide preprocessed data that is classified in different sets: training, testing, and validation. This enables benchmarking of desired algorithms against similar approaches to be consistent.

Another aspect to consider is the *license* type. The most commonly used license is Creative Commons Attribution-NonCommercial-ShareAlike 3.0. It allows the user to copy and redistribute in any medium or format and also to remix, transform, and build upon the material. KITTI and NuScenes databases are examples of such distribution license. The Oxford database uses a Creative Commons Attribution-Noncommercial 4.0. which, compared with the first license type, does not force the user to distribute his contributions under the same license as the database. Opposite to that the AMUSE database is licensed under Creative Commons Attribution-Noncommercial-noDerivs 3.0 which makes the database illegal to distribute if modifications of the material are made.

With very few exceptions, the data sets are collected from a single city, which is usually around university campuses or company locations in Europe, the US, or Asia. Germany is the most active country for driving recording vehicles. Unfortunately, all available data sets together cover a very small portion of the world map. One reason for this is the memory size of the data which is in direct relation with the sensor setup and the quality. For example, the Ford data set takes around 30 GB for each driven kilometers, which means that covering an entire city will take hundreds of TeraBytes of driving data. The majority of the available data sets consider sunny, daylight, and urban conditions, these being ideal operating conditions for autonomous driving systems.

9 | COMPUTATIONAL HARDWARE AND DEPLOYMENT

Deploying deep learning algorithms on target edge devices is not a trivial task. The main limitations when it comes to vehicles are the price, performance issues, and power consumption. Therefore, embedded platforms are becoming essential for integration of AI algorithms inside vehicles due to their portability, versatility, and energy efficiency.

The market leader in providing hardware solutions for deploying deep learning algorithms inside autonomous cars is NVIDIA®. DRIVE PX (NVIDIA) is an AI car computer which was designed to enable the automakers to focus directly on the software for AVs.

The newest version of DrivePX architecture is based on two Tegra X2 (NVIDIA) systems on a chip (SoCs). Each SoC contains two Denve (NVIDIA) cores, four ARM A57 cores, and a GPU from the Pascal (NVIDIA) generation. NVIDIA® DRIVE PX is capable to perform real-time environment perception, path planning, and localization. It combines deep learning, sensor fusion, and surround vision to improve the driving experience.

Introduced in September 2018, NVIDIA® DRIVE AGX developer kit platform was presented as the world's most advanced self-driving car platform (NVIDIA), being based on the Volta Technology (NVIDIA). It is available in two different configurations, namely DRIVE AGX Xavier and DRIVE AGX Pegasus.

DRIVE AGX Xavier is a scalable open platform that can serve as an AI brain for self-driving vehicles, and is an energy-efficient computing platform, with 30 trillion operations/second, while meeting automotive standards, like the ISO 26262 functional safety specification. NVIDIA® DRIVE AGX Pegasus improves the performance with an architecture which is built on two NVIDIA® Xavier processors and two state-of-the-art TensorCore GPUs.

A hardware platform used by the car makers for ADASs is the R-Car V3H SoC platform from Renesas Autonomy (NVIDIA). This SoC provides the possibility to implement high-performance computer vision with low power consumption. R-Car V3H is optimized for applications that involve the usage of stereo cameras, containing dedicated hardware for CNNs, dense optical flow, stereo-vision, and object classification. The hardware features four 1.0 GHz Arm Cortex-A53 MPCore cores, which makes R-Car V3H a suitable hardware platform which can be used to deploy trained inference engines for solving specific deep learning tasks inside the automotive domain.

Renesas also provides a similar SoC, called R-Car H3 (NVIDIA) which delivers improved computing capabilities and compliance with functional safety standards. Equipped with new CPU cores (Arm Cortex-A57), it can be used as an embedded platform for deploying various deep learning algorithms, compared with R-Car V3H, which is only optimized for CNNs.

A field-programmable gate array (FPGA) is another viable solution, showing great improvements in both performance and power consumption in deep learning applications. The suitability of the FPGAs for running deep learning algorithms can be analyzed from four major perspectives: efficiency and power, raw computing power, flexibility, and functional safety. Our study is based on the research published by Intel (Nurvitadhi et al., 2017), Microsoft (Ovtcharov et al., 2015), and UCLA (Cong et al., 2018).

By reducing the latency in deep learning applications, FPGAs provide additional raw computing power. The memory bottlenecks, associated with external memory accesses, are reduced or even eliminated by the high amount of chip cache memory. In addition, FPGAs have the advantages of supporting a full range of data types, together with custom user-defined types.

FPGAs are optimized when it comes to efficiency and power consumption. The studies presented by manufacturers, like Microsoft and Xilinx, show that GPUs can consume upon 10 times more power than FPGAs when processing algorithms with the same computation complexity, demonstrating that FPGAs can be a much more suitable solution for deep learning applications in the automotive field.

In terms of flexibility, FPGAs are built with multiple architectures, which are a mix of hardware programmable resources, digital signal processors, and Processor Block RAM (BRAM) components. This architecture flexibility is suitable for deep and sparse neural networks, which are the state-of-the-art for the current machine learning applications. Another advantage is the possibility of connecting to various input and output peripheral devices, like sensors, network elements, and storage devices.

In the automotive field, functional safety is one of the most important challenges. FPGAs have been designed to meet the safety requirements for a wide range of applications, including ADAS. When compared with GPUs, which were originally built for graphics and high-performance computing systems, where functional safety is not necessary, FPGAs provide a significant advantage in developing driver assistance systems.

10 | DISCUSSION AND CONCLUSIONS

We have identified seven major areas that form open challenges in the field of autonomous driving. We believe that deep learning and AI will play a key role in overcoming these challenges:

Perception: In order for an autonomous car to safely navigate the driving scene, it must be able to understand its surroundings. Deep learning is the main technology behind a large number of perception systems. Although great progress has been reported with respect to accuracy in object detection and recognition (Z.-Q. Zhao, Zheng, Xu, & Wu, 2018), current systems are mainly designed to calculate 2D or 3D bounding boxes for a couple of trained object classes, or to provide a segmented image of the driving environment. Future methods for perception should focus on increasing the levels of recognized details, making it possible to perceive and track more objects in real-time. Furthermore, additional work is required for bridging the gap between image- and LiDAR-based 3D perception (Wang et al., 2019), enabling the computer vision community to close the current debate on camera versus LiDAR as main perception sensors.

Short- to middle-term reasoning: Additional to a robust and accurate perception system, an AV should be able to reason its driving behavior over a short (milliseconds) to middle (seconds to minutes) time horizon (Pendleton et al., 2017). AI and deep learning are promising tools that can be used for the high- and low-level path planning required for navigating the miriad of driving scenarios. Currently, the largest portions of papers in deep learning for self-driving cars are focused mainly on perception and End2End learning (Shalev-Shwartz et al., 2016; T. Zhang et al., 2016). Over the next period, we expect deep learning to play a significant role in the area of local trajectory estimation and planning. We consider long-term reasoning as solved, as provided by navigation systems. These are standard methods for selecting a route through the road network, from the car's current position to destination (Pendleton et al., 2017).

Availability of training data: "Data is the new oil" became lately one of the most popular quotes in the automotive industry. The effectiveness of deep learning systems is directly tied to the availability of training data. As a rule of thumb, current deep learning methods are also evaluated based on the quality of training data (Janai et al., 2017). The better the quality of the data, the higher the accuracy of the algorithm. The daily data recorded by an AV are on the order of petabytes. This poses challenges on the parallelization of the training procedure, as well as on the storage infrastructure. Simulation environments have been used in the last couple of years for bridging the gap between scarce data and the deep learning's hunger for training examples. There is still a gap to be filled between the accuracy of a simulated world and real-world driving.

Learning corner cases: Most driving scenarios are considered solvable with classical methodologies. However, the remaining unsolved scenarios are corner cases which, until now, required the reasoning and intelligence of a human driver. To overcome corner cases, the generalization power of deep learning algorithms should be increased. Generalization in deep learning is of special importance in learning hazardous situations that can lead to accidents, especially due to the fact that training data for such corner cases is scarce. This implies also the design of *one-shot* and *low-shot* learning methods that can be trained a reduced number of training examples.

Learning-based control methods: Classical controllers make use of an a priori model composed of fixed parameters. In a complex case, such as autonomous driving, these controllers cannot anticipate all driving situations. The effectiveness of deep learning components to adapt based on past experiences can also be used to learn the parameters of the car's control system, thus better approximating the underlaying true system model (Ostafew, 2016; Ostafew et al., 2016).

Functional safety: The usage of deep learning in safety-critical systems is still an open debate, efforts being made to bring the computational intelligence and functional safety communities closer to each other. Current safety standards, such as the ISO 26262, do not accommodate machine learning software (Salay et al., 2017). Although new data-driven design methodologies have been proposed, there are still opened issues on the explainability, stability, or classification robustness of deep neural networks.

Real-time computing and communication: Finally, real-time requirements have to be fulfilled for processing the large amounts of data gathered from the car's sensors suite, as well as for updating the parameters of deep learning systems over high-speed communication lines (Nurvitadhi et al., 2017). These real-time constraints can be backed up by advances in semiconductor chips dedicated for self-driving cars, as well as by the rise of 5 G communication networks.

10.1 | Final notes

AV technology has seen a rapid progress in the past decade, especially due to advances in the area of AI and deep learning. Current AI methodologies are nowadays either used or taken into consideration when designing different components for a self-driving car. Deep learning approaches have influenced not only the design of traditional perception-planning-action pipelines, but have also enabled End2End learning systems, able do directly map sensory information to steering commands.

Driverless cars are complex systems which have to safely drive passengers or cargo from a starting location to destination. Several challenges are encountered with the advent of AI-based AVs deployment on public roads. A major challenge is the difficulty in proving the functional safety of these vehicles, given the current formalism and explainability of neural networks. On top of this, deep learning systems rely on large training databases and require extensive computational hardware.

This paper has provided a survey on deep learning technologies used in autonomous driving. The survey of performance and computational requirements serves as a reference for system-level design of AI-based self-driving vehicles.

ACKNOWLEDGMENTS

The authors would like to thank Elektrobit Automotive for the infrastructure and research support.

ORCID

Sorin Grigorescu http://orcid.org/0000-0003-4763-5540
Bogdan Trasnea http://orcid.org/0000-0001-6169-1181
Tiberiu Cocias http://orcid.org/0000-0003-4311-0018
Gigel Macesanu http://orcid.org/0000-0002-9906-501X

REFERENCES

Amodei, D., Olah, C., Steinhardt, J., Christiano, P. F., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. *arXiv preprint*, 1606.06565.

Andrychowicz, M., Baker, B., Chociej, M., Jozefowicz, R., McGrew, B., Pachocki, J., … Zaremba, W. (2018). Learning dexterous in-hand manipulation. *arXiv preprint*, 1812.06489.

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A deep convolutional encoder–decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2481–2495.

Barnes, D., Maddern, W., Pascoe, G., & Posner, I. (2018). Driven to distraction: Self-supervised distractor learning for robust monocular visual odometry in urban environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE.

Barsan, I. A., Wang, S., Pokrovsky, A., & Urtasun, R. (2018). Learning to localize using a LiDAR intensity map. In *Proceedings of the 2nd Conference on Robot Learning (CoRL)*.

Bechtel, M. G., McEllhiney, E., & Yun, H. (2018). DeepPicar: A low-cost deep neural network-based autonomous car. In *The 24th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)* (pp. 1–12).

Bellman, R. (1957). *Dynamic programming*. Princeton, NJ, USA: Princeton University Press.

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(8), 1798–1828.

Bernd, S., Detlev, R., Susanne, E., Ulf, W., Wolfgang, B., & Carsten, P. (2012). Challenges in applying the ISO 26262 for driver assistance systems. In *Schwerpunkt Vernetzung, 5. Tagung Fahrerassistenz*.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., … Zhao, J. (2016). End to End learning for self-driving cars. arXiv preprint 1604.07316.

Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., & Muller, U. (2017). Explaining how a deep neural network trained with end-to-end learning steers a car. arXiv preprint 1704.07911.

Brachmann, E., & Rother, C. (2018). Learning less is more—6D camera localization via 3D surface regression. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018*.

Bresson, G., Alsayed, Z., Yu, L., & Glaser, S. (2017). Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, *2*(3), 194–220.

Brostow, G. J., Fauqueur, J., & Cipolla, R. (2009). Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*, *30*, 88–97.

Brunner, M., Rosolia, U., Gonzales, J., & Borrelli, F. (2017). Repetitive learning model predictive control: An autonomous racing example. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)* (pp. 2545–2550).

Burton, S., Gauerhof, L., & Heinzemann, C. (2017). Making the case for safety of machine learning in highly automated driving. *Lecture Notes in Computer Science*, *10489*, 5–16.

Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., … Beijbom, O. (2019). nuScenes: A multimodal dataset for autonomous driving. arXiv preprint 1903.11027.

Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., & Elhadad, N. (2015). Intelligible models for HealthCare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1721–1730).

Chakarov, A., Nori, A., Rajamani, S., Sen, S., & Vijaykeerthy, D. (2018). Debugging machine learning tasks. arXiv preprint 1603.07292.

Chen, C., Seff, A., Kornhauser, A. L., & Xiao, J. (2015). DeepDriving: Learning affordance for direct perception in autonomous driving. In *2015 IEEE International Conference on Computer Vision (ICCV)* (pp. 2722–2730).

Chen, X., Ma, H., Wan, J., Li, B., & Xia, T. (2017). Multi-view 3D object detection network for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2017*.

Cityscapes. (2018). *Cityscapes data collection*. Retrieved from https://www.cityscapes-dataset.com/

Cong, J., Fang, Z., Lo, M., Wang, H., Xu, J., & Zhang, S. (2018). Understanding performance differences of FPGAs and GPUs: (Abtract only). In *Proceedings of the 2018 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '18)* (p. 288). New York, NY: ACM.

Dai, J., Li, Y., He, K., & Sun, J. (2016). R-FCN: Object detection via region-based fully convolutional networks. *Advances in Neural Information Processing Systems NIPS 2016*, 379–387.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR 2005*.

Daumé, H., III, & Marcu, D. (2006). Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, *26*(1), 101–126.

Dickmanns, E., & Graefe, V. (1988). Dynamic monocular machine vision. *Machine Vision and Applications*, *1*, 223–240.

Dollar, P., Wojek, C., Schiele, B., & Perona, P. (2009). Pedestrian detection: A benchmark. In *2009 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 304–311).

Drews, P., Williams, G., Goldfain, B., Theodorou, E. A., & Rehg, J. M. (2017). Aggressive deep driving: Combining convolutional neural networks and model predictive control. *Conference on Robot Learning*, *78*, 133–142.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, *12*, 2121–2159.

Eraqi, H. M., Moustafa, M. N., & Honer, J. (2017). End-to-end deep learning for steering autonomous vehicles considering temporal dependencies. *Machine Learning for Intelligent Transportation Systems Workshop in the 31st Conference on Neural Information Processing Systems NIPS 2017*.

Faria, J. M. (2018). Machine Learning Safety: An Overview. *Safety-Critical Systems Club*.

Ferrel, T. (2010). *Engineering safety-critical systems in the 21st century*.

Flohr, F., & Gavrila, D. M. (2013). Daimler pedestrian segmentation benchmark dataset. In *Proceedings of the British Machine Vision Conference*.

Fridman, L., Brown, D. E., Glazer, M., Angell, W., Dodd, S., Jenik, B., … Reimer, B. (2017). MIT autonomous vehicle technology study: Large-scale deep learning based analysis of driver behavior and interaction with automation. In *IEEE Access 2017*.

Garcia-Favrot, O., & Parent, M. (2009). Laser scanner based SLAM in real road and traffic environment. In *IEEE International Conference on Robotics and Automation (ICRA09). Workshop on Safe Navigation in Open and Dynamic Environments Application to Autonomous Vehicles*.

Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, *32*(11), 1231–1237.

Girshick, R. (2015). Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1440–1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR '14)* (pp. 580–587). Washington, DC: IEEE Computer Society.

Goldberg, Y., & Hirst, G. (2017). Neural network methods for natural language processing. In Morgan, & Claypool (Eds.), *Synthesis lectures on human language technologies*, 37.

Goodale, M. A., & Milner, A. (1992). Separate visual pathways for perception and action. *Trends in Neurosciences*, *15*(1), 20–25.

Grigorescu, S., Trasnea, B., Marina, L., Vasilcoi, A., & Cocias, T. (2019). NeuroTrajectory: A neuroevolutionary approach to local state trajectory learning for autonomous vehicles. *IEEE Robotics and Automation Letters*, *4*(4), 3441–3448.

Gu, S., Lillicrap, T., Sutskever, I., & Levine, S. (2016). Continuous deep Q-learning with model-based acceleration. In *International Conference on Machine Learning ICML 2016* (Vol. 48, pp. 2829–2838).

Gu, T., Dolan, J. M., & Lee, J. (2016). Human-like planning of swerve maneuvers for autonomous vehicles. In *2016 IEEE Intelligent Vehicles Symposium (IV)* (pp. 716–721).

Harris, M. (2016). Google reports self-driving car mistakes: 272 Failures and 13 near misses. *The Guardian*. https://www.theguardian.com/technology/2016/jan/12/google-self-driving-cars-mistakes-data-reports

Hasirlioglu, S., Kamann, A., Doric, I., & Brandmeier, T. (2016). Test methodology for rain influence on automotive surround sensors. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)* (pp. 2242–2247).

He, K., Gkioxari, G., Dollar, P., & Girshick, R. B. (2017). Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 2980–2988).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778).

Hecker, S., Dai, D., & VanGool, L. (2018). End-to-end learning of driving models with surround-view cameras and route planners. In *European Conference on Computer Vision (ECCV)*.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., … Silver, D. (2018). Rainbow: Combining improvements in deep reinforcement learning. *Artificial Intelligence 2018?*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hoermann, S., Bach, M., & Dietmayer, K. (2017). Dynamic occupancy grid prediction for urban autonomous driving: Deep learning approach with fully automatic labeling. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Hubel, D. H., & Wiesel, T. N. (1963). Shape and arrangement of columns in cat's striate cortex. *The Journal of Physiology*, *165*(3), 559–568.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 Mb model size. arXiv preprint 1602.07360.

Janai, J., Güney, F., Behl, A., & Geiger, A. (2017). Computer vision for autonomous vehicles: Problems, datasets and state-of-the-art. *ArXiv preprint, 1704.05519*.

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

Kamel, M., Hafez, A., & Yu, X. (2018). A review on motion control of unmanned ground and aerial vehicles based on model predictive control techniques. *Engineering Science and Military Technologies*, *2*, 10–23.

Kapania, N. R., & Gerdes, J. C. (2015). Path tracking of highly dynamic autonomous vehicle trajectories via iterative learning control. In *2015 American Control Conference (ACC)* (pp. 2753–2758).

Katz, G., Barrett, C. W., Dill, D. L., Julian, K., & Kochenderfer, M. J. (2017). Reluplex: An efficient SMT solver for verifying deep neural networks. In *CAV*.

Kendall, A., Grimes, M., & Cipolla, R. (2015). PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)* (pp. 2938–2946). Washington, DC: IEEE Computer Society.

Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J. M., … Shah, A. (2018). *Learning to drive in a day*. ArXiv preprint, 1807.00412.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In *Third International Conference on Learning Representations (ICLR 2015)*. San Diego, CA.

Koopman, P. (2017). Challenges in autonomous vehicle validation: Keynote presentation abstract. In *Proceedings of the 1st International Workshop on Safe Control of Connected and Autonomous Vehicles*.

Koschorrek, P., Piccini, T., Öberg, P., Felsberg, M., Nielsen, L., & Mester, R. (2013). A multi-sensor traffic scene dataset with omnidirectional video. In *Ground Truth—What is a Good Dataset? CVPR Workshop 2013*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems (NIPS)* (25, pp. 1097–1105). Harrahs and Harveys, Lake Tahoe, USA: Curran Associates Inc.

Ku, J., Mozifian, M., Lee, J., Harakeh, A., & Waslander, S. L. (2018). Joint 3D proposal generation and object detection from view aggregation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2018*. IEEE.

Kurd, Z., Kelly, T., & Austin, J. (2007). Developing artificial neural networks for safety critical systems. *Neural Computing and Applications*, *16*(1), 11–19.

Laskar, Z., Melekhov, I., Kalia, S., & Kannala, J. (2017). Camera relocalization by computing pairwise relative poses using convolutional neural network. In *The IEEE International Conference on Computer Vision (ICCV)*.

Law, H., & Deng, J. (2018). Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 734–750).

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Lefevre, S., Carvalho, A., & Borrelli, F. (2015). Autonomous car following: A learning-based approach. In *2015 IEEE Intelligent Vehicles Symposium (IV)* (pp. 920–926).

Lefèvre, S., Carvalho, A., & Borrelli, F. (2016). A learning-based framework for velocity control in autonomous driving. *IEEE Transactions on Automation Science and Engineering*, *13*(1), 32–42.

Levin, S. (2018). *Tesla fatal crash: 'Autopilot' mode sped up car before driver killed, report finds*. The Guardian. https://www.theguardian.com/technology/2016/jun/30/tesla-autopilot-death-self-driving-car-elon-musk

Li, J., Peng, K., & Chang, C.-C. (2018). An efficient object detection algorithm based on compressed networks. *Symmetry*, *10*(7), 235.

Li, X., Flohr, F., Yang, Y., Xiong, H., Braun, M., Pan, S., … Gavrila, D. M. (2016). A new benchmark for vision-based cyclist detection. In *2016 IEEE Intelligent Vehicles Symposium (IV)* (pp. 1028–1033).

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., … Wierstra, D. (2016). *Continuous control with deep reinforcement learning*. ArXiv preprint, 1509.02971.

Liu, S., Jia, J., Fidler, S., & Urtasun, R. (2017). SGN: Sequential Grouping Networks for Instance Segmentation. *IEEE International Conference on Computer Vision ICCV 2017*, 3516–3524.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In *European Conference on Computer Vision* (pp. 21–37). Springer.

Luo, W., Yang, B., & Urtasun, R. (2018). Fast and furious: Real time end-to-end 3D detection, tracking and motion forecasting with a single convolutional net. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2018)*.

Maddern, W., Pascoe, G., Linegar, C., & Newman, P. (2017). 1 Year, 1000km: The Oxford RobotCar dataset. *The International Journal of Robotics Research (IJRR)*, *36*(1), 3–15.

Marina, L., Trasnea, B., Cocias, T., Vasilcoi, A., Moldoveanu, F., & Grigorescu, S. (2019). Deep grid net (DGN): A deep learning system for real-time driving context understanding. In *International Conference on Robotic Computing (ICRC 2019)*. Naples, Italy.

McPherson, J. (2018). How Uber's self-driving technology could have failed in the fatal tempe crash. *Forbes*. https://www.forbes.com/sites/jimmcpherson/2018/03/20/uber-autonomous-crash-death/

Meier, F., Hennig, P., & Schaal, S. (2014). Efficient Bayesian local model learning for control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)* (pp. 2244–2249). IEEE.

Melekhov, I., Ylioinas, J., Kannala, J., & Rahtu, E. (2017). Image-based localization using hourglass networks. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)* (pp. 870–877).

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Möller, N. (2012). *The concepts of risk and safety*. Dordrecht, Netherlands: Springer.

Muller, U., Ben, J., Cosatto, E., Flepp, B., & Cun, Y. L. (2006). Off-road obstacle avoidance through end-to-end learning. *Advances in neural information processing systems NIPS 2006*, 739–746.

Nguyen-Tuong, D., Peters, J., & Seeger, M. (2008). Local Gaussian process regression for real time online model learning. In *Proceedings of the Neural Information Processing Systems Conference* (pp. 1193–1200).

Nurvitadhi, E., Venkatesh, G., Sim, J., Marr, D., Huang, R., Ong Gee Hock, J., … Boudoukh, G. (2017). Can FPGAs beat GPUs in accelerating next-generation deep neural networks? In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA '17)* (pp. 5–14). New York, NY: ACM.

Nushi, B., Kamar, E., Horvitz, E., & Kossmann, D. (2017). On human intellect and machine failures: Troubleshooting integrative machine learning systems. In *AAAI*.

NVIDIA. *Denver core*. Retrieved from https://en.wikichip.org/wiki/nvidia/microarchitectures/denver

NVIDIA. *NVIDIA AI car computer drive PX*. Retrieved from https://www.nvidia.com/en-au/self-driving-cars/drive-px/

NVIDIA. *NVIDIA drive AGX*. Retrieved from https://www.nvidia.com/en-us/self-driving-cars/drive-platform/hardware/

NVIDIA. *NVIDIA Volta*. Retrieved from https://www.nvidia.com/en-us/data-center/volta-gpu-architecture/

NVIDIA. *Pascal microarchitecture*. Retrieved from https://www.nvidia.com/en-us/data-center/pascal-gpu-architecture/

NVIDIA. *Tegra X2*. Retrieved from https://devblogs.nvidia.com/jetson-tx2-delivers-twice-intelligence-edge/

Ojala, T., Pietikäinen, M., & Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, *29*(1), 51–59.

O'Kane, S. (2018). How Tesla and Waymo are tackling a major problem for self-driving cars: Data. *Transportation*. https://mobility21.cmu.edu/how-tesla-and-waymo-are-tackling-a-major-problem-for-self-driving-cars-data/

Ondruska, P., Dequaire, J., Wang, D. Z., & Posner, I. (2016). End-to-end tracking and semantic segmentation using recurrent neural networks. ArXiv preprint, 1604.05091.

Ostafew, C. J., Schoellig, A. P., & Barfoot, T. D. (2013). Visual teach and repeat, repeat, repeat: Iterative learning control to improve mobile robot path tracking in challenging outdoor environments. *IEEE/RSJ International Conference on Intelligent Robots and Systems 2013*, 176–181.

Ostafew, C. J. (2016). *Learning-based control for autonomous mobile robots* (Ph.D. thesis). Canada: University of Toronto.

Ostafew, C. J., Collier, J., Schoellig, A. P., & Barfoot, T. D. (2015). Learning-based nonlinear model predictive control to improve vision-based mobile robot path tracking. *Journal of Field Robotics*, *33*(1), 133–152.

Ostafew, C. J., Schoellig, A. P., & Barfoot, T. D. (2016). Robust constrained learning-based NMPC enabling reliable mobile robot path tracking. *International Journal of Robotics Research*, *35*(13), 1547–1563.

Ovtcharov, K., Ruwase, O., Kim, J.-Y., Fowers, J., Strauss, K., & Chung, E. (2015). Accelerating deep convolutional neural networks using specialized hardware. *Microsoft whitepaper*.

Paden, B., Cáp, M., Yong, S. Z., Yershov, D. S., & Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving Urban vehicles. *IEEE Transactions on Intelligent Vehicles*, *1*(1), 33–55.

Pan, Y., Cheng, C., Saigol, K., Lee, K., Yan, X., Theodorou, E., & Boots, B. (2018). Agile off-road autonomous driving using end-to-end deep imitation learning. *Robotics: Science and Systems 2018*, (pp. 1–10). Pittsburgh, Pennsylvania, USA.

Pan, Y., Cheng, C.-A., Saigol, K., Lee, K., Yan, X., Theodorou, E. A., & Boots, B. (2017). Learning deep neural network control policies for agile off-road autonomous driving. *31st Conference on Neural Information Processing Systems NIPS 2017*. Long Beach, CA, USA.

Pandey, G., McBride, J. R., & Eustice, R. M. (2011). Ford campus vision and LiDAR data set. *International Journal of Robotics Research*, *30*(13), 1543–1552.

Panomruttanarug, B. (2017). Application of iterative learning control in tracking a Dubin's path in parallel parking. *International Journal of Automotive Technology*, *18*(6), 1099–1107.

Panov, A. I., Yakovlev, K. S., & Suvorov, R. (2018). Grid path planning with deep reinforcement learning: Preliminary results. In *8th Annual International Conference on Biologically Inspired Cognitive Architectures (BICA 2017). Procedia Computer Science*, *123*, 347–353.

Parasuraman, R., & Riley, V. (1997). Humans and automation: Use, misuse, disuse, abuse. *Human Factors*, *39*(2), 230–253.

Paszke, A., Chaurasia, A., Kim, S., & Culurciello, E. (2016). Enet: A deep neural network architecture for real-time semantic segmentation. arXiv preprint, 1606.02147.

Paxton, C., Raman, V., Hager, G. D., & Kobilarov, M. (2017). Combining neural networks and tree search for task and motion planning in challenging environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. abs/1703.07887.

Pendleton, S. D., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y. H., & Ang, M. H. (2017). Perception, planning, control, and coordination for autonomous vehicles. *Machines*, *5*(1), 1–54.

Perot, E., Jaritz, M., Toromanoff, M., & Charette, R. D. (2017). End-to-end driving in a realistic racing game with deep reinforcement learning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (pp. 474–475).

Pomerleau, D. A. (1989). ALVINN: An autonomous land vehicle in a neural network. *Advances in neural information processing systems NIPS 1989*, 305–313.

Qi, C. R., Liu, W., Wu, C., Su, H., & Guibas, L. J. (2018). Frustum PointNets for 3D object detection from RGB-D data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018*.

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Varshney, K. P., & Alemzadeh, H. (2016). On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big Data*, 5.

Radwan, N., Valada, A., & Burgard, W. (2018). VLocNet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters*, *3*(4), 4407–4414.

Ramos, S., Gehrig, S. K., Pinggera, P., Franke, U., & Rother, C. (2016). Detecting unexpected obstacles for self-driving cars: Fusing deep learning and geometric modeling. In *IEEE Intelligent Vehicles Symposium* (Vol. 4).

Rausch, V., Hansen, A., Solowjow, E., Liu, C., Kreuzer, E., & Hedrick, J. K. (2017). Learning a deep neural net policy for end-to-end control of autonomous vehicles. In *2017 American Control Conference (ACC)* (pp. 4914–4919).

Rawlings, J., & Mayne, D. (2009). *Model predictive control: Theory and design*. Madison, WI, USA: Nob Hill Publishing.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 779–788).

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint 1804.02767.

Rehder, E., Quehl, J., & Stiller, C. (2017). Driving like a human: Imitation learning for path planning using convolutional neural networks. In *International Conference on Robotics and Automation Workshops*.

Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*, 1137–1149.

Renesas. *R-Car H3*. Retrieved from https://www.renesas.com/sg/en/solutions/automotive/soc/r-car-h3.html/

Renesas. *R-Car V3H*. Retrieved from https://www.renesas.com/eu/en/solutions/automotive/soc/r-car-v3h.html/

Rosolia, U., Carvalho, A., & Borrelli, F. (2017). Autonomous racing using learning model predictive control. In *2017 American Control Conference (ACC)* (pp. 5115–5120).

Rumelhart, D. E., McClelland, J. L., & PDP Research Group, C. (Eds.), (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*. Cambridge, MA: The MIT Press.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., … Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211–252.

SAE Committee. (2014). *Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems*. https://www.sae.org/standards/content/j3016_201806/

Salay, R., Queiroz, R., & Czarnecki, K. (2017). *An analysis of ISO 26262: Machine learning and safety in automotive software* (SAE Technical Paper). https://www.sae.org/publications/technical-papers/content/2018-01-1075/

Sallab, A. E., Abdou, M., Perot, E., & Yogamani, S. (2017a). Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, *2017*(19), 70–76.

Sallab, A. E., Abdou, M., Perot, E., & Yogamani, S. (2017b). Deep reinforcement learning framework for autonomous driving. *CoRR*.

Sarlin, P., Debraine, F., Dymczyk, M., Siegwart, R., & Cadena, C. (2018). Leveraging deep visual descriptors for hierarchical efficient localization. In *Proceedings of the 2nd Conference on Robot Learning (CoRL)*.

Schwarting, W., Alonso-Mora, J., & Rus, D. (2018). Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, *1*, 187–210.

Seeger, C., Müller, A., & Schwarz, L. (2016). Towards road type classification with occupancy grids. In *Intelligent Vehicles Symposium—Workshop: DeepDriving—Learning Representations for Intelligent Vehicles IEEE*. Gothenburg, Sweden.

Shalev-Shwartz, S., Shammah, S., & Shashua, A. (2016). *Safe, multi-agent, reinforcement learning for autonomous driving*. ArXiv preprint, 1610.03295.

Shin, K., Kwon, Y. P., & Tomizuka, M. (2018). RoarNet: A robust 3D object detection based on region approximation refinement. *2019 IEEE Intelligent Vehicles Symposium (IV)*, 2510–2515.

Sick. (2018). *Sick LiDAR for data collection*. Retrieved from https://www.sick.com/

Sigaud, O., Salaün, C., & Padois, V. (2011). On-line regression algorithms for learning mechanical models of robots: A survey. *Robotics and Autonomous Systems*, *59*(12), 1115–1129.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations 2015*.

Sun, L., Peng, C., Zhan, W., & Tomizuka, M. (2018). A fast integrated planning and control framework for autonomous driving via imitation learning. In *ASME 2018 Dynamic Systems and Control Conference* (Vol. 3).

Sutton, R., & Barto, A. (1998). *Introduction to reinforcement learning*. Cambridge, MA: The MIT Press.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2015). Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Takanami, I., Sato, M., & Yang, Y. P. (2000). A fault-value injection approach for multiple-weight-fault tolerance of MNNs. In *Proceedings of the IEEE-INNS-ENNS* (Vol. 3, pp. 515–520).

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic robotics (Intelligent robotics and autonomous agents).* Cambridge, MA: The MIT Press.

Tinchev, G., Penate-Sanchez, A., & Fallon, M. (2019). Learning to see the wood for the trees: Deep laser localization in urban and natural environments on a CPU. *IEEE Robotics and Automation Letters*, *4*(2), 1327–1334.

Treml, M., Arjona-Medina, J. A., Unterthiner, T., Durgesh, R., Friedmann, F., Schuberth, P., … Hochreiter, S. (2016). *Speeding up semantic segmentation for autonomous driving.* Whitepaper.

Udacity. (2018). *Udacity data collection.* Retrieved from http://academictorrents.com/collection/self-driving-cars

Ushani, A. K., & Eustice, R. M. (2018). Feature learning for scene flow estimation from LiDAR. In *Proceedings of the 2nd Conference on Robot Learning (CoRL)* (Vol. 87, pp. 283–292).

Valada, A., Vertens, J., Dhall, A., & Burgard, W. (2017). AdapNet: Adaptive semantic segmentation in adverse environmental conditions. In *2017 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 4644–4651).

Varshney, K. R. (2016). Engineering safety in machine learning. In *2016 Information Theory and Applications Workshop (ITA)* (pp. 1–5).

Velodyne (2018). *Velodyne LiDAR for data collection.* https://velodynelidar.com/

Viola, P. A., & Jones, M. J. (2001). Rapid object detection using a boosted cascade of simple features. In *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), with CD-ROM, December 8–14, 2001* (pp. 511–518). Kauai, HI.

Walch, F., Hazirbas, C., Leal-Taixé, L., Sattler, T., Hilsenbeck, S., & Cremers, D. (2017). Image-based localization using LSTMs for structured feature correlation. In *2017 IEEE International Conference on Computer Vision (ICCV)* (pp. 627–637).

Wang, Y., Chao, W.-L., Garg, D., Hariharan, B., Campbell, M., & Weinberger, K. (2019). Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2019).*

Watkins, C., & Dayan, P. (1992). Q-learning. *Machine Learning*, *8*(3), 279–292.

Wulfmeier, M., Wang, D. Z., & Posner, I. (2016). Watch this: Scalable cost-function learning for path planning in urban environments. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* abs/1607.02329.

Xu, H., Gao, Y., Yu, F., & Darrell, T. (2017). End-to-end learning of driving models from large-scale video datasets. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Yang, S., Wang, W., Liu, C., Deng, K., & Hedrick, J. K. (2017a). Feature analysis and selection for training an end-to-end autonomous vehicle controller using the deep learning approach. In *2017 IEEE Intelligent Vehicles Symposium* (Vol. 1).

Yang, Z., Zhou, F., Li, Y., & Wang, Y. (2017b). A novel iterative learning path-tracking control for nonholonomic mobile robots against initial shifts. *International Journal of Advanced Robotic Systems*, *14*, 172988141771063.

Yin, H., & Berger, C. (2017). When to use what data set for your self-driving car algorithm: An overview of publicly available driving datasets. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)* (pp. 1–8).

Yu, F., Xian, W., Chen, Y., Liu, F., Liao, M., Madhavan, V., & Darrell, T. (2018). BDD100K: A diverse driving video database with scalable annotation tooling. ArXiv preprint, 1805.04687.

Yu, L., Shao, X., Wei, Y., & Zhou, K. (2018). Intelligent land-vehicle model transfer trajectory planning method based on deep reinforcement learning. *Sensors (Basel, Switzerland)*, *18*, 1–22.

Zhang, S., Wen, L., Bian, X., Lei, Z., & Li, S. Z. (2017). Single-shot refinement neural network for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Zhang, T., Kahn, G., Levine, S., & Abbeel, P. (2016). Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search. In *2016 IEEE International Conference on Robotics and Automation (ICRA).*

Zhao, H., Qi, X., Shen, X., Shi, J., & Jia, J. (2018). ICNet for real-time semantic segmentation on high-resolution images. In *European Conference on Computer Vision* (pp. 418–434).

Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 6230–6239).

Zhao, Z. Q., Zheng, P., Xu, S. T., & Wu, X. (2018). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, *30*(11), 3212–3232.

Zhou, Y., & Tuzel, O. (2018). VoxelNet: End-to-end learning for point cloud based 3D object detection. In *IEEE Conference on Computer Vision and Pattern Recognition 2018* (pp. 4490–4499).

Zhu, H., Yuen, K.-V., Mihaylova, L. S., & Leung, H. (2017). Overview of environment perception for intelligent vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *18*, 2584–2601.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section.

How to cite this article: Grigorescu S, Trasnea B, Cocias T, Macesanu G. A survey of deep learning techniques for autonomous driving. *J Field Robotics*. 2019;1–25. https://doi.org/10.1002/rob.21918