IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.Doi Number

A Survey on Deep Learning for Steering Angle Prediction in Autonomous Vehicles

Usman Gidado Manzo[1,4], Haruna Chiroma[2,*], Nahla Aljojo[3], Saidu Abubakar[4], Segun I. Popoola[5], Mohammed Ali Al-Garadi[6]

[1]Computer Department, ETF-Community Education Resource Centre Gombe, Nigeria
[1]Mathematical Science Department, Abubakar Tafawa Balewa University Bauchi, Nigeria
[2]Future Technology Research Center, National Yunlin University of Science and Technology, Yunlin, Douliu, Taiwan
[3]Department of Information System and Technology, College of Computer Science and Engineering, University of Jeddah, Jeddah, Saudi Arabia
[4]Mathematical Science Department, Abubakar Tafawa Balewa University Bauchi, Nigeria
[5]Department of Engineering, Manchester Metropolitan University, Manchester, UK
[6]Department of Biomedical Informatics, Emory University, Atlanta, USA

[*]Corresponding author: Haruna Chiroma (freedonchi@yahoo.com).

ABSTRACT Steering angle prediction is critical in the control of Autonomous Vehicles (AVs) and has attracted the attention of researchers, manufacturers, and insurance companies in the automotive industry. Different Deep Learning (DL) architectures have been applied to predict the steering angle of AVs in various scenarios. A survey on steering angle prediction based on deep learning algorithms can help expert researchers identify those areas that require development. Also, novice researchers can use the survey as a starting point. In this paper, we present a broad study on the recent advances made in DL architectures that covers the steering angle prediction of AVs. A new comprehensive taxonomy of the application of DL in steering angle prediction of AVs is created. The survey presents a concise research summary synthesis, and analysis. It is found that most researchers depend on Convolutional Neural Network (CNN) over other DL architectures in predicting the steering angle of autonomous driving vehicles. Also identified are open research problems. The prominent challenge facing DL-based steering angle prediction of AVs is lack of sufficient real-world datasets, which means that researchers largely depend on data generated from simulated environments. Lastly, alternative viewpoints to solve the identified open research challenges are proposed, pointing towards promising future research directions.

INDEX TERMS Convolutional Neural Network; Deep Learning; Deep Neural Network; Steering Angle Prediction; Autonomous Vehicles.

I. INTRODUCTION

Human errors account for more than 90% of car accidents. In comparison, mechanical failures are responsible for only 2% [1]. These statistics prompted the idea of proposing Autonomous Vehicles (AVs) to eliminate human errors. AVs are now becoming an innocuous substitute for human drivers, which saves the lives of thousands of people every year. Among the most remarkable of the ongoing research efforts to manage diversified challenges facing AVs are recognition of humans, traffic, road and lanes, steering controls, and path planning. A large amount of different sensor data is gathered and processed to address these challenges [2].

Currently, there are more than 1,400 autonomous cars, trucks, and other vehicles in the testing phase initiated by more than 80 companies across 36 states in the United States of America. California is listed among those states that have deployed AVs on public roads. AV control has made some headway in recent years, and many auto vendors have pledged commercial production on a large scale within a period of two to three years [3]. AVs currently have a massive impact on the automotive industry [1]. In AV steering control, lateral and longitudinal motions constitute the major components of vehicle motion control. Steering of the vehicle, control of the lateral motion of AVs that aim at controlling the position of the vehicle in the lane, and other lateral actions like changing of pathway and avoidance of collision while manipulating pedals of the vehicle are aspects of longitudinal motion [4]. AVs possess a wide scope paired with the tendency to commit fewer errors than human drivers [5].

Deep Learning (DL) has made a positive impact on the control of AVs, particularly in terms of the steering angle prediction due to its ability to effectively process unlabeled raw data. DL understands the world through analyzing the context of a scene, while focusing on essential objects and observing them at hierarchical levels – from small objects with higher resolution to large objects with lower resolution. Therefore, when analyzing a scene, DL is reasonably insensitive to variations of environmental conditions, yet requires a large amount of high-quality data to achieve high accuracy [6]. Neural networks can learn complex interactions between features, which is beneficial for autonomous driving in dynamic environments [7]. Steering a car through traffic constitutes a complex task that is very hard to cast into algorithms. Thus, researchers turn to train

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

Artificial Neural Networks (ANN) with a stream of data generated by front-facing cameras yielding associated steering angles [8].

The benefits of steering angle prediction using the DL approach are that it has tolerance for mistakes, the ability to quickly identify errors and better capability in managing unpredictable situations [9]. Different studies have applied various algorithms to achieve these goals. For instance, in the literature [10], [11], and [12], the authors propose a Genetic Algorithm (GA) for steering control. Linhui and Lie [13] worked on control of unmanned vehicles using fuzzy logic via GA. Layne and Passino [14] worked on fuzzy logic model-based control learning for cargo ship steering. Cao et al. [15] worked on a system for controlling brake pressure based on fuzzy logic using steering angle and yaw speed. Lee [16] proposed a steering autopilot control algorithm for four-wheel-steering passenger vehicles. A major setback of fuzzy logic is that the use of fuzzy logic-based controllers for rear-end collision avoidance depends on the number of fuzzy rules, and an extreme amount of such has direct bias on its efficiency [17].

Many studies have applied DL in predicting the steering angle of AVs [2], [18], [19], [7], [20]. Kuutti, Bowden [4] and Oussama and Mohamed [21] surveyed DL applications in AV control. Our survey differs from the previous survey papers as it includes the following: a detailed taxonomy on the use of DL in steering angle prediction of AVs; the classification of projects based on DL architecture; synthesis and analysis based on publication trend, frequency of DL architecture in predicting the steering angle of AVs, and per-year analysis of the DL architectures; different DL frameworks and libraries for implementing the prediction of steering angle using DL and the merits and demerits of each; the limitation associated with each project; prominence of simulators and driving scenarios; and lastly, different challenges encountered in the previous surveys and future research directions based on the identified challenges.

In this paper, we attempt to provide a broad and in-depth review, synthesis, and analysis of the recent advances on the steering angle prediction of AVs using DL approaches. This study will benefit novice researchers and developers who are interested in this research area and can use our literature survey as initial reading material. Also, it will help expert readers who can use the study to propose novel approaches for steering angle prediction by adopting DL architecture. Thus, interested readers can use this survey to compare various applications of DL in steering angle prediction of AVs.

The scope of this survey is limited to steering angle prediction approaches of AVs that are developed based on DL architectures. In summary, the contributions of our survey paper include:

- a new taxonomy that integrates the architectures, platforms, libraries, simulators, and optimizers of DL-based steering angle prediction in AVs;
- analysis and synthesis of the literature on DL architectures for steering angle prediction in AVs;
- Current challenges of DL application to steering angle prediction in AVs;
- A new perspective to DL approaches for steering angle prediction in AVs with prospects for future research development.

The remaining parts of the paper are organized as follows: Section II presents the taxonomy of DL architectures for steering angle prediction in AVs; Sections III, IV, V, and VI present the concepts of DL architectures, steering angle prediction of AV, applications of DL architectures to steering angle prediction, and driving scenarios in AVs and simulated environment, respectively; Section VII discusses DL frameworks and libraries; Section VIII highlights optimizers used in various projects; Section IX, presents a general analysis of the applications of DL to steering angle prediction in AVs; Section X discusses challenges and future research prospects, while the conclusions are drawn in section XI.

II. TAXONOMY OF DEEP LEARNING ARCHITECTURES FOR STEERING ANGLE PREDICTION OF AUTONOMOUS VEHICLES

Taxonomy enables researchers to identify classifications used in a specific research area based on the most accurate information [22]. It organizes the body of knowledge in the field of research to identify the respective researchers in the various fields [23]. The taxonomy also determines a number of open problems that share vital characteristics that can be addressed through similar approaches [24]. The proposed taxonomy allows readers to gain a quick understanding of the existing issues and pave the way to developing solutions that have yet to be investigated. As the taxonomy identifies open problems, it also helps in opening new areas of research [25]. DL for steering angle prediction of AVs can be classified into the following four major categories: DL architectures, DL framework, AVs simulators, and optimizer which is derived from the literature that applies DL in steering angle prediction of AVs, as shown in Figure 1.

III. DEEP LEARNING ARCHITECTURES

DL is part of the broad field of artificial intelligence that refers to the science and engineering of building machines with the intelligence and ability to achieve goals like humans, as stated by John McCarthy who coined this term in the 1950s [26]. DL led to several practical applications and innovations in various domains. The automotive industry and development of AVs constitutes the domain where DL has made a considerable impact. DL, as a branch of Machine Learning (ML), proffers solutions to AV [27]. It is considered as a means of automating the process of predictive analytics [28]. Different DL architectures that are applied in steering angle prediction of AVs are discussed in this section so that the readers can become acquainted with DL architectures operations and understand how these DL algorithms operate to achieve their goal. These DL architectures will be discussed as follows:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

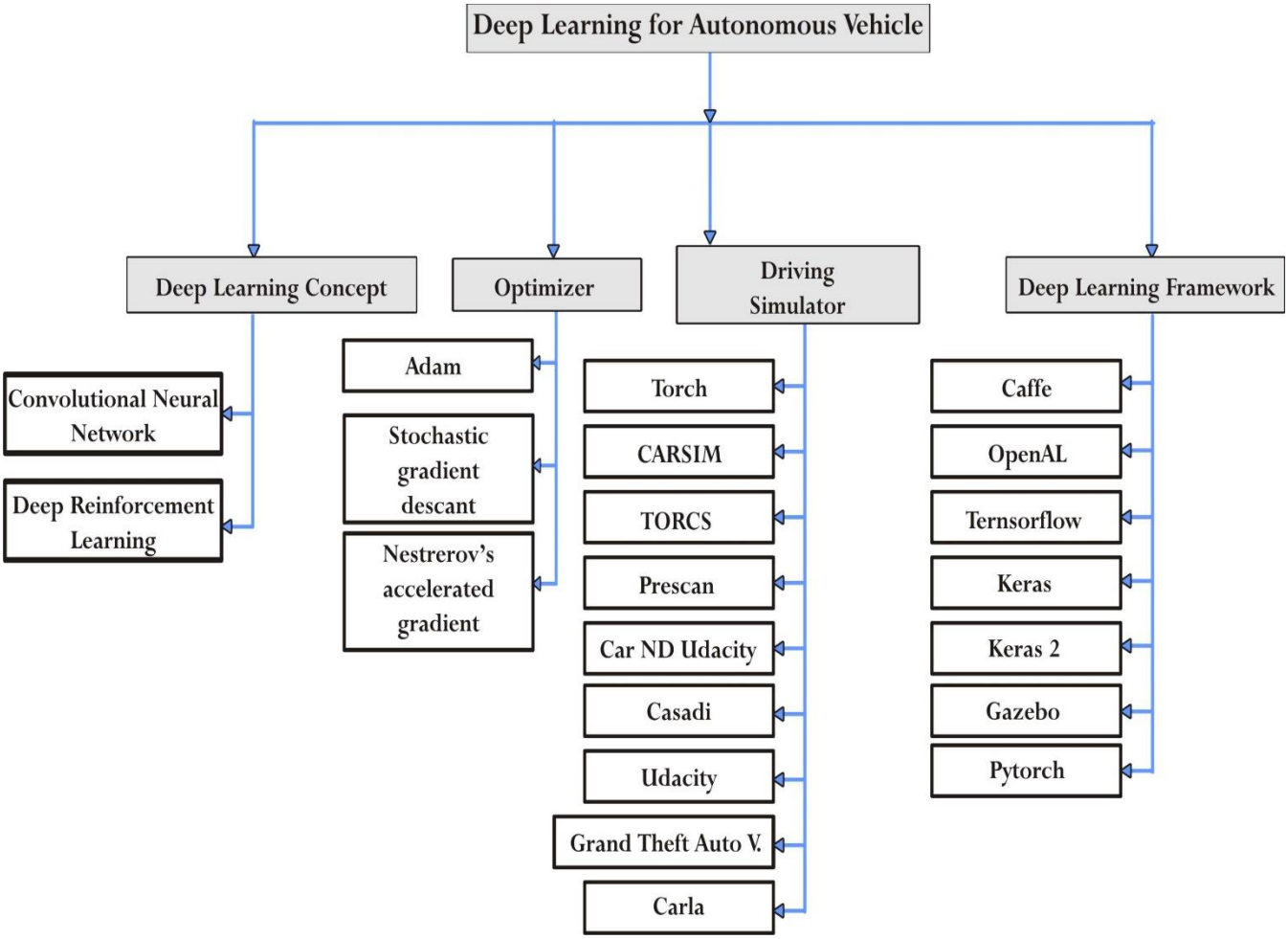

FIGURE 1. Taxonomy of steering angle prediction of AVs

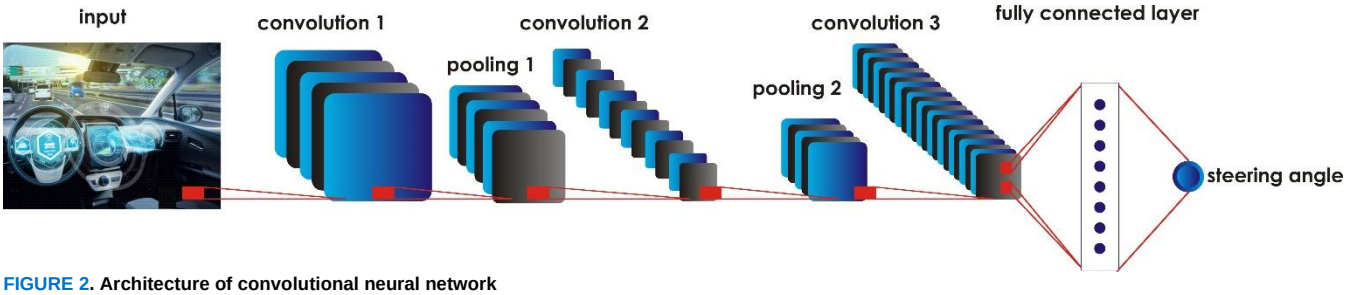

FIGURE 2. Architecture of convolutional neural network

A. CONVOLUTIONAL NEURAL NETWORK

Convolutional neural network (CNN) is a DL architecture that comprises multiple layers [2], [20]. CNN is the most attractive DL architecture in use due to its efficiency in solving image processing problems. [29] Mathematically, convolution defines a procedure by which one real-valued function works on another real-valued feature to yield a new real-valued role. Let one real-valued function be $f(t)$ and other be $g(t)$, where t signifies consistent time. Let the convolution of the two be signified as $s(t)$. The convolution process is commonly indicated as * [29]:

$$f(t) * g(t) = (f * g)(\mathrm{t}) = \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau \quad (1)$$

Similarly, convolution is a commutative operation that implies, $(f * g)$ which is same as $(\mathrm{g} * \mathrm{f})$, [29]:

$$(f * g)(t) = (\mathrm{g} * \mathrm{f})(t) = \int_{-\infty}^{\infty} g(t)\, f(t-\tau) d\tau \quad (2)$$

Similar equations can likewise be expressed for discrete processes that are ordinarily worked within ML applications. The distinct partners of the two equations can be represented as $f(k)$ and $g(k)$, where k signifies a discrete example of time [29]:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

$$f(k) * g(k) = (f * g)(k) = \sum_{\delta=-\infty}^{\infty} f(\delta) g(k-\delta) \quad (3)$$

and [29]:

$$(f * g)(k) = (g * k)(k) = \sum_{\delta=-\infty}^{\infty} g(\delta) f(k-\delta) \quad (4)$$

These definitions are fundamentally the same as the definition of connection that exists between two functions. In any case, the critical distinction here is the indication of δ in the case of discrete convolution, and τ in the case of ceaseless convolution is an inverse among f and g.
However, in the case where the sign is flipped, similar equations denote a correlation operation. The inversion sign allows one of the functions to be replaced in time before it is increased in a point-wise manner by another function. Here, a reversal has the effect of expansion, and after convolution, the effect is not equal to the consequences of the connection. The convolution possesses intriguing features from the point of Fourier transform in the frequency area, and they are intensely utilized in signal handling applications. The architecture of CNN is shown in Figure 2. The CNN building block consists of the following three layers: the convolution layer, the ReLu, and the pooling layer. The convolution layers comprise a sequence of kernels, whereby each kernel is applied to the real figure. ReLu Unit, as its name implies, rectifies the output of the convolutional layer and converts all values that are negative to 0, thus the function [29]:

$$\mathrm{f(x)} = \max(0, x) \quad (5)$$

In the pooling layer, the maximum value of the block replaces the entire block as it performs downsampling through swapping the larger-sized blocks. Those blocks with a single value aim at drastically reducing the dimension of the data that flows into the network while retaining important information that has been captured as a result of the convolution operations. Max-pooling is a commonly used pooling method. The three layers make up the essential CNN building block. Numerous kinds of blocks can be used in a particular CNN. The earlier presented layers are aimed at a specific spatial part of the input and are transformed with the convolutional kernels.
The max-pooling operation is expressed as [30]:

$$P4^k = \mathrm{MaxPool}(C3^k) P4^k_{i,j} = \max\begin{pmatrix} C3^k_{(2i,2j)}, & C3^k_{(2i,+1,2j)} \\ C3^k_{(2i,2j+1)}, & C3^k_{(2i,+1,2j+1)} \end{pmatrix}, \quad (6)$$

where (i, j) are indices of k[th] feature map of the output, and k is the feature map index, while layer 5 $C5$ is the third convolution layer that produces 120, output feature maps and is expressed as [30]:

$$C5^k_{i,j} = \sigma\left(\sum_{d=0}^{15} \sum_{m=0}^{4} \sum_{n=0}^{4} w^{k,d}_{m,n} * P4^d_{i+m,j+n} + b^k\right) \quad (7)$$

where C5k denotes the 120 output feature maps of convolution layers $C5$ of size $1\,X\,1$, k is the index of the feature map's output, $(m\,X\,n)$ are the filter weight indices, while d is the number of channel in input and (i, j) are the output's indices. As it is$1\,X\,1$, the index (i, j) remains (0,0) for every filter. Equation 13 can be abridged, as the filter size is equal to the size of the input, in order to avoid the occurrence of convolution stride [30].

$$C5^k = \sigma\left(\sum_{d=0}^{15} \sum_{m=0}^{4} \sum_{n=0}^{4} \quad w^{k,d}_{m,n} * P4^d_{m,n} + b^k\right) \quad (8)$$

At the sixth layer (layer 6) we have the fully connected layer which comprises of 10 neurons for ten classes, which is mathematically expressed as [30]:

$$F6^k = \sum_{i=1}^{120} w^k_i * C5^i \quad (9)$$

([31] & [32])

$$\Phi(\upsilon) = \max\{\upsilon, 0\} \text{ (Rectfied Linear Unit [ReLU])} \quad (10)$$

$$\Phi(\upsilon) = \max\{\min[\upsilon, 1], -1\} \text{ (hard tanh)} \quad (11)$$

The ReLU and hard tanh activation functions have replaced the sigmoid and soft tanh activation functions in new neural networks due to the ease in training multilayered neural networks with these activation keys [32].

[2] TCNN is a type of DL network that performs dimension reduction of higher dimensional input via convolution. CNN yields excellent performance, especially on larger data sets. When utilized in an autonomous driving vehicle, it learns all the essential images of the road features without the need of a guide for optimization. CNN serves as (the controller) maps frames from car camera to steering angle to control the AV. CNNs has performed brilliantly on other applications such as classification of images, detection of objects, steering angle prediction (Rausch, Hansen [2], Pan, Cheng [18] & Do, Duong [20]) speech recognition [2], object recognition [20], natural language processing (Wani, Bhat [30] & Do, Duong [20]), and text processing [30].

B. DEEP REINFORCEMENT LEARNING

Deep Reinforcement Learning (DRL) is a combination of DL and Reinforcement Learning (RL) and deals with sequences of decision making. It has been used to solve a variety of complex decision making tasks that were previously out of reach of the ML (François-Lavet, Henderson [33]. DRL constitutes a reward-driven process involving trial and error whereby the system learns to interact with a complex environment to accomplish a rewarding outcome referred to as RL. The trial and error process stems from the need to maximize expected rewards over time in RL. It can serve as a pursuit gateway for creating rightly intelligent agents like autonomous cars, game-playing algorithms, and smart robots that interact with the environment [31]. Figure 3 illustrates a DRL parameter

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

taking action upon studying the environment: the agent interacts with the environment, makes an observation, and takes intelligent action.

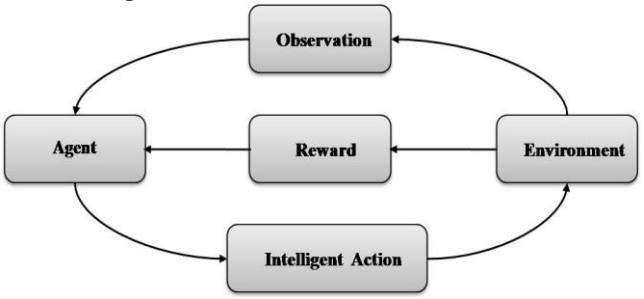

FIGURE 3. The architecture of the deep reinforcement learning environment

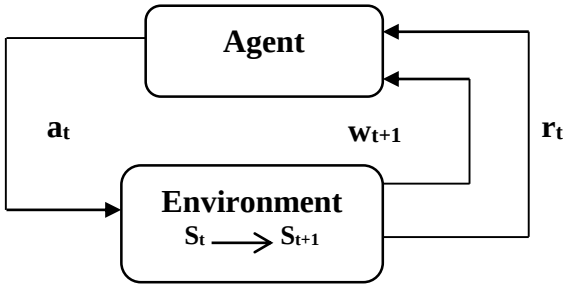

FIGURE 4. Agent-environment interaction in RL [33]

Definition: A discrete-time stochastic control process is Markovian if it possesses the Markov property [33]:

$$P\left(\omega_{t+1}\,|\omega_t, a_t\right) = \mathbb{P}\left(\omega_{t+1}\,|\omega_t, a_{t,\ldots,}\omega_0, a_0\right), \text{and}$$

$\mathbb{P}\left(r_t\,|\omega_t, a_t\right) = \mathbb{P}\left(r_t\,|\omega_t, a_{t,\ldots,}\omega_0, a_0\right)$. The Markov property refers to the future process that exclusively depends on the current observation, whereby the agent is not interested in looking at the complete history.

In the expected return, the case of the RL agent whose goal is to find a policy $\pi(s,a) \in \Pi$ is considered to optimize an expected return $V\,\pi\,(S) : S \rightarrow R$ which is also referred to as V −value function such that [33]:

$$\mathrm{V}^{\pi}(s) = \mathrm{E}\left[\sum_{\mathrm{k}=0}^{\infty} \mathrm{r}^{\mathrm{k}} \mathrm{r}_{\mathrm{t+k}}\, \mathrm{S}_{\mathrm{t}} = \mathrm{S}, \pi\right], \tag{12}$$

Where: E

- $\mathrm{r}_t = {}_{\mathrm{a}\sim\pi(\mathrm{s_t.})}\, \mathrm{s}(\mathrm{s_t}.\,\mathrm{a}, \mathrm{s_{t+1}}),$
- $\mathrm{P}\left(\mathrm{s}_{t+1}|s_t, a_t\right) = \mathrm{T}\left(s_t, a_t, \mathrm{st}+1\right) \mathrm{with a_t} \sim \pi\left(\mathrm{s_t}.\right),$

From the definition of the expected return, the expected optimal return is conveyed as [33]:

$$\mathrm{V}^{*}(s) = {}^{\max}_{\pi\in}\mathrm{V}^{\pi(s)}. \tag{13}$$

In addition to the V-value function, several other functions of interest can be introduced as the Q-value function $Q\pi\,(s,a) : S \times A \rightarrow R$ is defined as [33]:

$$\mathrm{Q}^{\pi}(s,a) = \mathrm{E}\left[\sum_{\mathrm{k}=0}^{\infty} \mathrm{r}^{\mathrm{k}} \mathrm{r}_{\mathrm{t+k}}\, \mathrm{S}_{\mathrm{t}} = \mathrm{S}, \mathrm{at} = \mathrm{a}, \pi\right], \tag{14}$$

Equation 14 may be rewritten recursively in the case of Markov decision process using Bellman's equation [33]:

$$\mathrm{Q}^{\pi}(s,a) = \sum_{s'\in S} \mathrm{T}(\mathrm{s}, \mathrm{a}, \mathrm{s}') + \mathrm{rQ}^{\pi}\,(s, a = \pi(s')) \tag{15}$$

Likewise, , the optimal Q −value function $Q*(s,a)$ can also be defined as [33]:

$$\mathrm{Q}^{*}(s,a) = {}^{\max}_{\pi\in\Pi}\, \mathrm{Q}^{\pi(s,a)}. \tag{16}$$

The accuracy of the Q −value function as related to the V −value function is that the optimal policy can be derived right from $\mathrm{Q}^{*}(s,a)$ [33]:

$$\pi^{*}(s) = {}^{\mathrm{argmax}}_{\mathrm{a}\in A}\ \mathrm{Q}^{*(s,\ a)}. \tag{17}$$

The optimal V-value function $V*(s)$ is the probable cut-rated return in a given state s while following the policy $\pi*$ later. Function $V*(s)$ is the expected discount reward when in a given state s while developing the policy π *subsequently. It is also likely to define the critical function [33]:

$$\mathrm{A}^{\pi}(s,a) = \mathrm{Q}^{\pi}(s,a), -\mathrm{V}^{\pi}(\mathrm{s}). \tag{18}$$

The quantity describes how worthy action a is when equated with the probable yield while following direct policy π. [34] DRL is obtained from deep neural networks to represent the state or observation and approximate any of the components of reinforcement learning like value function $v(s;\theta)$ or ^ $q(s,a;\theta)$, policy $\pi(a \mid s;\theta)$, and the model, which is the state transition function and reward function. Here, the parameters θ are the weights in the deep neural networks [35]. DRL algorithms achieve excellent replay based on experience in various challenging domains, utilize much memory and computation per real interaction, and also require off-policy learning algorithms, which can be updated using data generated by an old policy.

IV. STEERING CONTROL OF AUTONOMOUS VEHICLES

An autonomous car is equipped with a camera that visualizes the road ahead and outputs the angle of the steering wheel [36]. The front steering angle is controlled using the steering indirectly. The mechanism extending from the steering wheel to the front wheel is referred to as the steering system. The motion equations are generated for the steering systems, which are used when examining the characteristics of the steering system due to the vehicle motion. In the usual driving condition of the vehicle, the steering wheel control is done by the driver's hand [37]. [38] reported that the rotation of the steering wheel is transferrable through the shaft

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

steering wheel. Various models as described in (Smolyakov, Frolov [39], Woo, Yu [40], Wang, Wen [41], Sharma, Tewolde [42] & Simmons, Adwani [43]) predict continuous steering commands from raw input pixels applying various approaches; some achieve this through an end-to-end method. Even though these models offer a high level of accuracy, what happens on the various layers of the network remains unknown, which makes it a prerequisite for the car manufacturing companies and their first-tier suppliers to apprehend and lawfully verify that these approaches yield the correct output before they can be adopted for commercialized AVs [44]. Picturing what the model perceives on various layers is crucial in order to develop improved networks and avoid a trial and error approach. Recently, various methods have been proposed to visualize the activation layer and highlight vital regions of an input image using occlusion techniques (Zeiler and Fergus [45] & [46]).

Predicting the steering angle of an autonomous vehicle is also carried out through reinforcement learning. Steering angles can be predicted using inverse turning radius$\hat{u}_t = r_t^{-1}$, where $r_t$is the turning radius at each time step t instead of using steering angle commands. This depends on the steering geometry of the vehicle and can result in numerical instability when predicting steering angle commands at near zero. The relationship between the inverse turning radius u_t and the command of steering angle Θ_t can be estimated by Ackerman's steering geometry [47] & [48] as:

$$\theta_t = f_{\text{steers}}(u_t) = u_t d_w K_s\left(1 + K_{\text{slip}} v t^2\right) \tag{19}$$

where Θ_t is in degrees and v_t in (m/s) is a steering angle and velocity at time t respectively, and K_s, K_{slip} and d_w are vehicle-definite parameters. K_s is the steering ratio between the wheel's turn, K_{slip} denotes the relative motion that is between the surface of the road and the wheel, while d_w denotes the length between the rear wheels and the front. Each raw input image is down-sampled and resized to 80×160×3 with the nearest neighbor algorithm, 80 representing the dimension along the road section, 160 representing the dimension perpendicular to a road segment and 3 the layers of colors. This is done to minimize computational cost [48].

In place of images that have various aspect ratios, their heights are cropped to match the rate before performing down sampling. The mean RGB value is subtracted when compared with the training set from each pixel [49] & [50], achieving zero-centered inputs that are initially in different scales. The driving dataset does not show different levels as the camera gains in advance or automatically as it is usually calibrated to capture images with higher quality in a specific dynamic environment [48].

Given a smoothing factor of $0 \leq \propto_s \leq 1$, the simple exponential smoothing method is specified as [48]:

$$\begin{pmatrix}\hat{\theta}_t \\ \hat{u}_t\end{pmatrix} = \alpha_s \begin{pmatrix}\theta_t \\ u_t\end{pmatrix} + (1-\alpha_s)\begin{pmatrix}\hat{\theta}_{t-1} \\ \hat{u}_{t-1}\end{pmatrix} \tag{20}$$

where $\hat{\theta}_t$ and $\hat{v}_t$ are the level time series that describe the past load, which are realizations of some unknown stochastic processes of $\Theta_t$and v_t, respectively. Note that the parameters are the same as the real-time series when $\propto_s = 1$, while the values of $\propto_s$ closer to zero have a more significant smoothing effect and are less in response to recent changes.

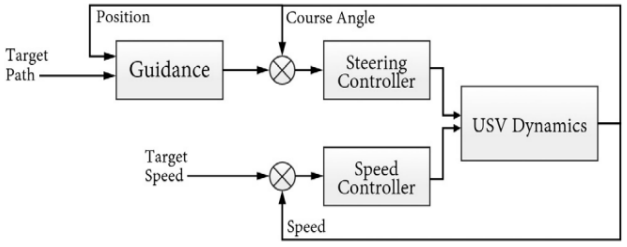

FIGURE 5. Block diagram from the unmanned surface vehicle path following system [40].

Figure 5 shows a schematic diagram of the Unmanned Surface Vehicle (USV) path-following system for training. As shown, a dynamic system should be implemented to describe the behavior of the vehicle's dynamic, while a guidance block is utilized to determine the desired course angle.

$\boldsymbol{R}(\boldsymbol{\eta})$ is a rotation matrix from a body fixed-frame to an inheritance frame [40]:

$$\dot{\eta} = \boldsymbol{R}(\eta)v \tag{21}$$

[40]: $v = (u, v, r)^T$ (22)

[40]: $\eta = (x_{i,} y_{i,} \psi)^T$ (23)

[40] $$\boldsymbol{R}(\boldsymbol{\eta}) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{24}$$

The 3DOF horizontal planar dynamic model of the USV can be described as follows [40]:

$$\boldsymbol{M}\dot{v} + C(v)v + D(v)v = f, \tag{25}$$

where $\boldsymbol{M}$ is the mass matrix, $C(v)$ is the Coriolis and centripetal and centripetal matrix, $D(v)$ is the damping matrix, and f is the control forces and moment. Since the target USV is a differential thruster type, the control forces and moment f can be expressed as [40]:

$$f = \begin{bmatrix}\tau_X \\ \tau_Y \\ \tau_N\end{bmatrix} = \begin{bmatrix} T_{port} + T_{stbd} \\ o \\ \left(T_{port} + T_{stbd}\right).\frac{B}{2}\end{bmatrix} \tag{26}$$

In equation (26), T_{port} and T_{stbd} represent the thrust force of the port side and the starboard side thruster, respectively, and B refers to the beam of the target USV. An actuator is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

implemented based on the model in [40]; thus, the thrust force can be directly calculated from the thruster's RPM δn as utilized in Equation (27) [40]:

$$T = 3.54 \times 10^{-5}\, \delta n^2 + 0.084\delta n - 3.798, \quad (27)$$

where the RPM δn of whatever port or the starboard side is determined by the steering command δn_d and the speed command $\delta n_{m,}$ which are inversely calculated from (28) and (29), respectively. In (30), the steering command δn_d is defined as the difference of RPM, which is between δn_{port} and δn_{stbd} and normalized by the maximum RPM value δn_{max}. Similarly, the speed command is defined as a mean value of RPMs that has been normalized as used in Equation (31) [40]:

$$\delta n_d = (\delta n_{port} - \delta n_{stbd})/(2\delta n_{max}) \quad (28)$$

$$[40]:\ \delta n_m = (\delta n_{port} + \delta n_{stbd})/(2\delta n_{max}) \quad (29)$$

[40] In the view of the actuator dynamics that have been presented in the absolute value of the RPM, $\delta \dot{n}$ are saturated as δn_{max} and $\delta \dot{n}_{max,}$ respectively.

Amid the various dynamic models suggested as a model for simulation, dynamics was adopted as the model for the simulation. Due to the simplicity of the selected model, the time for the calculation is sufficient for running millions of steps of simulation from the repetitive training process while maintaining an appropriate level of dynamics prediction accuracy. For the linearized maneuvering dynamics, the speed dynamic model is illustrated as in Equation (32), and the steering dynamics can be expressed as in Equation (33) [40]:

$$\dot{u} = a_u u + b_{u^u \tau_X} + b_{u_{bias}} \quad (30)$$

$$[40]\ \begin{bmatrix}\dot{v}\\ \dot{r}\end{bmatrix} = A\begin{bmatrix}u\\ r\end{bmatrix} + B\tau_N + B_{bias,} \quad (31)$$

where the matrix A is a 2-by-2 system matrix, the matrix B is a 2-by-1 control matrix, and $B_{bias,}$ is a 2-by-1 matrix for inclusion of the bias term. According to the system identification result in the unknown parameters in (34) and (35), it can be described as in (36) and (37) [40]:

$$\dot{u} = -1.3191_u + 0.0028_{u\tau_X} + 0.6836 \quad (32)$$

[40]:

$$\begin{bmatrix}\dot{v}\\ \dot{r}\end{bmatrix} = \begin{bmatrix}0.0161 & -0.0052\\ 8.2861 & -0.9860\end{bmatrix}\begin{bmatrix}v\\ r\end{bmatrix} + \begin{bmatrix}0.0002\\ 0.0307\end{bmatrix}\tau_N + \begin{bmatrix}0.0068\\ 1.3276\end{bmatrix} \quad (33)$$

[32] defines a vector field for a linear path as in (34). According to the equation, the maximum deviation of the course angle from the angle's path is restricted to χ^{∞} [32]:

$$\chi^d = \chi^{\infty}.tan^{-1(ke_y)} + x^{path} \quad (34)$$

A target path for USV can be represented in various forms, such as polynomial spline (usually used with the Serret-Frenet frame) or a set of waypoints. According to this approach, the target path is represented through utilizing a line of sight (LOS) from the previous waypoint $W_{k-1}(x_{k-1,}\ y_{k-1})$ towards the current waypoint $W_k(x_{k,}\ y_k)$. When d is defined as a Euclidean distance between W_{k-1} and W_k, and the difference between the d and along-track error e_x has become smaller than the particular threshold distance value $d_{th,}$ as $d - e_x < d_{th,}$ the target waypoint is changed to the next waypoint. First, the direction of the path (X_{path}), the direction from $W_{(k-1)}$ to the USV(dW_{k-1}) and the distance from the previous waypoint to USV (dW_{k-1}) can be calculated as follows [40]:

$$X_{path} = atan\left(\frac{y_k - y_{k-1}}{x_k - x_{k-1}}\right)$$
$$Xw_{k-1} = atan\left(\frac{y_{usu} - y_{k-1}}{x_{usu} - x_{k-1}}\right)$$
$$dw_{k-1} = \sqrt{(y_{usu} - y_{k-1})^2 + (x_{usu} - x_{k-1})^2} \quad (35)$$

Using the geometric relationship, we can now calculate along-track error e_x and the cross-track error e_y by utilizing (36) [40]:

$$e_x = \cos(X_{path} - Xw_{k-1}).dw_{k-1}$$
$$e_x = \sin(X_{path} - Xw_{k-1}).dw_{k-1} \quad (36)$$

When the error variables are identified, the desired course angle can be calculated by utilizing (37). In an RL problem, the goal is to find an optimal policy π^* that maximizes the accumulated discounted reward R_t as in (35) [51]. In equation 37, γ is known as a discount factor, which weights the future error and has a value between 0 and 1.

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2\ldots\ldots} = \sum_{k=1}^{\infty} \gamma k_{t+k+1} \quad (37)$$

A policy π can be evaluated by utilizing two value functions. A (state) value function $V^{\pi}(s)$ is defined as the expectation of the accumulated discounted reward $(s_{t,}a_t)$ while maintaining the policy. Likewise, an action-value function is defined as a value function for the particular state and action pair $(s_{t,}a_t)$ [40]:

$$V^{\pi}(s_t) = \mathbb{E}_{\pi}\left[R_t \,|s_t\right]$$
$$= \mathbb{E}_{\pi}\left[\sum_{k=1}^{\infty} \gamma^{k} r_{t+k+1} \,|s_t\right] \quad (38)$$

$$[40]\quad Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}\left[R_t \,|s_{t,}a_t\right]$$
$$= \mathbb{E}_{\pi}\left[\sum_{k=1}^{\infty} \gamma^{k} r_{t+k+1} \,|s_{t,}a_t\right] \quad (39)$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

According to the definition of the value functions and the optimal policy π^*, the optimal policy π^* always satisfies the following conditions [40]:

$$\pi^* = \arg\max_{\pi} V^{\pi}(\mathrm{s_t}) = \arg\max_{\pi} Q^{\pi}(\mathrm{s_t}, \mathrm{a_t}) \quad (40)$$

To solve the RL problem, researchers usually use a neural network as an approximator of the value functions. However, the learning process completed through updating the temporal difference update of the RL algorithm and the training of the neural network approximator for the value function approximation usually interfere with each other, thus hindering the learning process from being settled. This phenomenon is referred to as the inference problem [52] and is considered as the major obstacle to applying RL to a real world problem. In RL dealing with action-value function$Q^{\pi}(s_t, a_t)$, updating the $Q-$value is done by utilizing the Belman equation 45. If it is an assumption that the target policy is deterministic, the inner expectation can be eliminated, as shown in Equation (45). Given that the expectation is dependent on the environment, the policy Q^{μ} can be learned off-policy, which means that the exploration can be separated from the learning process [53] as:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t,} s_{t+1}} \sim E\left[r_{(s_t, a_t)} \gamma^{\mathbb{E}_{a_{t+1}} \sim \pi}\left[Q^{\pi}(s_{t+1,} a_{t+1})\right]\right] \quad (41)$$

[53]:

$$Q^{\mu}(s_t, a_t) = \mathbb{E}_{r_{t,} s_{t+1}} \sim E\left[r_{(s_t, a_t)} + \gamma\, Q^{\mu}(s_{t+1,} mu(s_{t+1}))\right] \quad (42)$$

When a neural network-based function approximator is parameterized by θ^Q, the approximator can be optimized by minimizing the loss function $L(\theta^Q)$ in (48) [53]:

$$L(\theta^Q) = \mathbb{E}_{s_{t\sim p^{\beta}, a_{t\sim\beta}, r_{t\sim E}}}\left[\left(Q\left(s_{t,} a_t \,|\theta^Q\right) - y_t\right)^2\right], \quad (43)$$

where y_t is known as the temporal difference target and is defined as [53]:

$$y_t = r(s_t, a_t) + \gamma^Q(s_{t+1,}\, \mu(s_{t+1}) \,|\theta^Q) \quad (44)$$

By adopting the newly developed techniques in DRL such as experience replay, separate target network, and batch normalization, the deep deterministic policy gradient algorithm [54] can treat large-scale neural network approximators. To update the networks, the critic network is updated using the gradient of the loss function $L(\theta^Q)$ in (43), while the actor network utilizes a deterministic policy gradient [53]. This can be obtained through using Equation (45) [40]:

$$\nabla_{\theta^{\mu}} J \approx \mathbb{E}s_t \sim p^{\beta}\left[\nabla_{\theta^{\mu}} Q(s, a|\theta^Q) \,|s = s_{t,} a = \mu(s_t \,|\theta^{\mu})\right] = \mathbb{E}s_t \sim p^{\beta}\left[\nabla_a Q\left(s, a \,|\, \theta^Q\right) \,|s = s_{t,} a = \mu(s_t) \nabla_{\theta^{\mu}} \mu\left(s \,|\theta^{\mu}\right) \,|s = s_{t,}\right] \quad (45)$$

According to the state space, $s \in \mathbb{s}$ is defined as (46) [40]:

$$\mathbb{s} = \left\{\tilde{X}, \dot{\tilde{X}}, e_y, \dot{e}_y, \delta n_d\right\}, \quad (46)$$

where $\tilde{\chi}$ is the difference between the course angle of the USV and the desired course angle calculated from the vector field guidance method as $\tilde{\chi} = \tilde{\chi} d, e_y$ is the cross-track error, and δn_d is a steering command of the USV. Since $\tilde{\chi}$ only gives angular positional error information to the controller, the variable e_y is added to consider the relative positional error information in generating the control input. Also, δn_d is integrated into the state space to provide the current steering command information to avert the chattering phenomenon. Since the RL based controller is designed as a steering controller, the action space $a \,\epsilon\, \mathbb{A}$ expressed as (47) [40]:

$$\mathbb{A} = \{\delta n_d\}, \quad (47)$$

where δn_d is a steering control command that defines the RPM commands of the main thrusters as in (44). Since the goal of the path-following problem is to minimize the cross-track error and the course angle error without producing chattering, partial reward functions are defined as (46)-(48) [40]:

$$r_{\tilde{X}} = \begin{cases} e^{-k_1 \cdot |\tilde{X}|} & \text{if } |\tilde{\chi}| < 90^o \\ -e^{-k_1 \cdot (\tilde{X} - 180} & \text{if } \tilde{\chi} \geq 90^o \\ -e^{-k_1 \cdot (\tilde{X} - 180} & \text{if } \tilde{\chi} \leq -90^o \end{cases} \quad (48)$$

[40]: $r_{e_y} = e^{-k_1 \cdot |e_y|}$ (49)

[40]: $r_{\sigma_\delta} = e^{-k_3 \cdot \sigma_{\delta,}}$ (50)

where χ, e_y, and σ_δ are the course angle error., cross track error, and standard deviation of the recent 20 steering commands history values. The partial selected reward function is 0.1, 0.2 and 0.3 for k_1, k_2, and k_3, respectively [40]:

$$r = w_{xe} r_{xe} + w_{e_y} r_{e_y} + w_{\sigma_\delta} r_{\sigma_\delta} \quad (51)$$

Autonomous Land Vehicle in a Neural Network (ALVINN) is a 3-layer back propagation network designed to move on the road as proposed in [55]. It constitutes a connectionist approach to the navigational task of road following, which is the main success of the training using simulated images. ANN displays promising performance and

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

flexibility in various domains that are characterized by high degrees of noise and variability, such as handwriting character recognition ([56] & [57]) and speech recognition [58]. Specifically, ALVINN was designed to control the NAVLAB, the Carnegie Mellon autonomous navigation test vehicle. ALVINN's network training is performed using artificial road "snapshots" and the Warp back propagation simulator as described in [59].

The Radio Detection and Ranging System (RADAR) is equipped with short-range and long-range sensors. Since its invention in the early 1930s, radar technology has inspired a considerable amount of inventions of every day impact [60]. Light Detection and Ranging (LIDAR) is an on-board sensor used in obtaining information about the environment to ensure that the autonomous vehicle's dynamical safety is translated to avoid single tire lift-off [61].

AVs primarily rely on on-board sensors for decision making and utilize safety information and guidance from the surrounding environment [62]. Autonomous vehicles used to require several onboard sensing and monitoring devices in order to obtain data of the surrounding environment [63]. Also, they sense the world through different mounted sensors mounted, and the information is processed in a perception block, which involves processing sensor data to meaningful information [64].

In a related work, [65] applies deep sensor fusion for developing AVs control. In [66] LIDAR and stereo vision sensor data are applied for developing the control of AVs. Similarly, [67] applies multiple sensor fusion in controlling autonomous vehicle. In [68] multi-sensor fusion system is applied for the development of autonomous vehicle control.

In [69], cameras mounted on AVs play the role of the sensors and also detect the white lines on the road and pedestrians and help control the vehicle directly. Today researchers like [8] achieve steering angle prediction without synchronizing the steering wheel sensor with the camera sensor, as the information of the steering wheel angle is communicated through the vehicle controller area network.

Traditional and end-to-end learning approaches

The traditional approach comprises several tasks such as image capturing from the vehicle camera [69], lane detection ([70] & [71]), path planning ([72] & [73]), control logic ([74] & [75]), and steering angle prediction [76]. The lane markings are usually detected using image processing techniques such as color enhancement, edge detection, etc. Path planning and control logic are performed based on detecting lane markings in the preliminary stage. The performance depends on feature extraction and image data interpretation. However, the manually defined rules and features are not optimal. Errors can also accrue from previous stages of processing, leading to inaccurate final results. At the same time, an end-to-end learning approach of AVs has proved surprisingly powerful and resolves the problems related to lane marking detection, path planning, and control by simultaneously optimizing all the processing steps [76].

V. APPLICATIONS OF DEEP LEARNING ARCHITECTURES IN AUTONOMOUS VEHICLE STEERING CONTROL

In this section, the applications of the DL architecture are discussed as follows:

A. CONVOLUTIONAL NEURAL NETWORK IN AUTONOMOUS VEHICLE STEERING ANGLE PREDICTION

In this section we describe the applications of CNN in the steering control of AVs to demonstrate the importance of CNN in AVs. For example, [77] proposed utilizing CNN to map raw pixels from a single camera to directly steer a car in an end-to-end manner. The CNN was able to learn significant features of the road from a very sparse training signal. However, robustness and visualization of internal processing steps of the network was poor. [78] applied CNN for steering angles prediction from images captured ahead of the road in an end-to-end manner. The result showed that CNN was able to identify objects and learned other features like lane markings, road edges, other cars on the road, bush lining, and typical vehicles that are additional features which would be difficult to anticipate and programmed by engineers. However, less training data were used that may not have covered various driving scenarios sufficiently.

[2] proposed CNN for end-to-end steering control of AVs where it served as the controller. Its performance was compared with the human driver's steering behavior. The result showed that the CNN's steering angles were superior to the human driver's steering angles. The limitations of the study include a smaller training dataset and less stability of the end-to-end system. [19] proposed CNN to resolve autonomous lateral control. CNN generated a proper steering angle that enabled the vehicle to complete laps with no human intervention. The performance of CNN in controlling the steering was evaluated on unknown tracks. For single lane unknown tracks, the model steered the vehicle effectively for 89.02% of the time. However, the amount of training data was small and did not cover many scenarios. [20] proposed CNN for steering angles of an autonomous vehicle. CNN was trained using data collected from the vehicle platform that was built with 1/10 scale RC car, Raspberry Pi 3 model B computer. The frames were collected from the front camera to generate steering commands. Manual and automated driving was compared. The trial results demonstrated effectiveness and robustness of the automated driving when completing the lane keeping task. However, it had slow camera latency performance, that is about 300-350 milliseconds.

[42] proposed CNN to implement the longitudinal and lateral control of vehicles through training two separate models to predict the speed and steering angle. CNN used 5*5 kernel and 2*2 max-pooling for each convolution. The model with dual action performed better: it attained autonomy of up to 100% on the e-road track and accomplished complete laps without crossing over the lane markings. However, too much

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

thread consumed much of the available memory space, and less data was used for the training. Gathering additional data from various tracks and environments to train CNN may produce better results. [79] proposed CNN based closed-loop feedback DAVE-2SKY to predict steering wheel angles for lateral control of AVs. The proposed CNN, DAVE-2SKY was compared with traditional CNN-based approaches. It performed well as it was able to control the steering wheel angle for lateral control of the autonomous vehicle and it accomplished robust control of the steering even in partially observable situations. This shows the prospect of fully intelligent autonomous driving vehicles controlled by CNN through an end-to-end steering controller. However, the simulation results revealed that it was unable to perform proper lateral control in the lane keeping task.

[39] proposed CNN to predict the steering angle. The CNN design was carried out with the intention of minimizing the training parameters. It consisted of a sequence of convolutional layers and fully connected layers where the output of the last layer was used as a prediction value of the steering angle. CNN was able to reduce a large number of parameters, avoid overfitting, and predict steering angles with 78.5% accuracy. However, there was high dispersion due to the use of the small size of the dataset. [76] proposed CNN for end-to-end learning approaches to solve the lane-keeping problem through producing the most suitable steering angles. CNN was applied to predict the steering angles, and its performance was compared to the performance of a human driver referred to as the ground truth. The CNN model produced accurate steering angles of the vehicle. However, a small size driving dataset was used that contained fewer driving scenarios, which did not include night driving scenarios.

[48] proposed CNN to predict steering angle commands. The CNN utilized a visual attention model that was augmented with an added layer of causal filtering. The CNN was tested using three real large-scale datasets for driving that contained more than 16 hours of video frames. The CNN performance was compared to the performance of a human driver. The result showed that the attention model was able to highlight image regions that influenced the output of the network. However, the method suffered from low spatial resolution deconvolution, which divides the attention map from capturing objects like cars and lane markings. [80] proposed CNN to develop DeepPicar called Raspberry pi3 (B), which involved the replication of real autonomous vehicles to predict steering angles. The CNN Raspberry pi3 (B) on NVIDIA performance was compared with Intel UP and NVIDIA Jeston TX2. The result showed that the new CNN Raspberry pi3 outperformed other models in terms of speed and also had the lowest cost. However, overfitting issues, CPU thermal throttling degraded the performance in the case of processing multiple DNN models in the network simultaneously, and it had high energy consumption. [44] proposed CNN with a generic visualization model that utilized attention heatmaps (AHs) by stressing image regions that were most relevant for steering control. This was to predict longitudinal and lateral control that involved steering, acceleration, and braking. CNN predicted output (inverse occlusion) was compared with a fully occluded image instead of the original image. The method remains an unusual visualization method used to better understand and improve the learning process of end-to-end control signals, i.e. lateral and longitudinal control. However, the work failed to extend the temporal analysis and investigate the generic metric that describes the robustness of the model.

[81] proposed CNN to produce end-to-end lateral control using a single short-range fisheye camera to solve lateral control of the AV. The CNN performance was compared with average multiple trained algorithms referred to as bagging. The result showed that the trained end-to-end CNN was capable of controlling vehicles autonomously with more than 99% accuracy on urban roads and was validated on a real car, in both open road and challenging scenarios like sharp turns and working zone areas of the test track. However, longitudinal control was not included, and better performance can be attained through improving the neural network architecture. [41] used CNN to propose a novel navigation command that utilizes the current position of the vehicle to calculate the subgoal angle for an end-to-end driving model to increase the quality of the steering angle prediction. The subgoal angle significantly boosted the performance of the driving model through the increasing quality of the steering angle prediction. The result also yielded a stable performance of complex tasks from the developed angle branched architecture. However, only RGB image was used as input, and the vehicle still failed along the way. [18] proposed CNN for steering control using an end-to-end system with sub-networks to solve continuous steering and throttle commands. The method depended on the experiment to validate current imitation learning theory, and the approach did not require state estimation or on-the-fly planning to steer the vehicle. CNN yielded generalized features that were more robust to covariate shift. It performed fast off-road navigation autonomously; however, it was limited to off-road navigation. The summary of the CNN applications in steering angle prediction is presented in Table

TABLE 1
SUMMARY OF CNN APPLICATION IN AUTONOMOUS VEHICLE STEERING CONTROL

Reference	Methodology	Objective	The algorithm used for evaluation	Result	Limitation
[77]	CNN	To solve end-to-end steering commands	Not applicable	Learns significant features from sparse training steering signal	Robustness and visualization of internal processing steps were not good

[78]	CNN	To solve end-to-end steering commands	Not applicable	The developed PilotNet learns to identify other features and predict steering angles	Small size training data were used
[2]	CNN	To solve end-to-end steering control	Human driver	The CNN steering angles were better than the human driver's steering angles	Small size training data
[19]	CNN	To solve lateral control	Not applicable	The trained model controlled the car on unknown tracks and was capable of doing laps without failing	Small amount of training data which did not cover much scenarios
[20]	CNN	To obtain a steering command.	Manual driving	The results demonstrated the efficiency and robustness of the autopilot model in lane-keeping tasks	Camera latency
[42]	CNN	To achieve longitudinal and lateral control	Not applicable	100% autonomy was attained on the e-road track; it accomplished complete laps without going over lane markings	Consumed memory space
[79]	CNN-based closed-loop feedback DAVE-2SKY	To achieve steering wheel angle for lateral control	Traditional CNN	Accomplished robust steering control even in partially observed scenarios; sufficiently intelligent autonomous driving vehicles	The simulation could not perform proper lateral control in lane-keeping.
[39]	CNN	To predict steering angle	Not applicable	A large number of parameters were reduced, overfitting was avoided, and steering angles were predicted	High dispersion due the small size of data
[76]	CNN	To predict steering angle for lane-keeping	Steering angles of a human driver	The CNN model yielded accurate relative steering angles	Small driving dataset, few scenarios not including night driving.
[48]	CNN	To predict steering control and remove spurious images	CNN with no attention	The attention model was able to highlight image regions that potentially influenced the output of the network	Low spatial resolution, deconvolution
[80]	CNN Raspberry pi3	To develop a low-cost device for predicting steering angles	Intel UP and NVIDIA Jeston TX2	Outperformed other models in speed; most cost-efficient	Overfitting issue, CPU thermal throttling, high-power consumption
[44]	CNN	To achieve lateral and longitudinal control	One fully occluded image	Occlusion technique was applied to a neural network to predict steering angles	Failed to extend the temporal analysis to the archive robustness of the model
[81]	CNN	To perform lateral control	Bagging	Achieved autonomy of >99% on urban roads	Could not avoid an obstacle
[82]	CNN	To predict steering angle	Branched model	Significantly boosted the performance of steering angle prediction	Only RGB image was used as observation input; vehicle made false stops
[18]	CNN	To obtain continuous steering and throttle commands to mimic MPC	CNN policy and its CNN sub-network	Performed fast off-road navigation autonomously at high speed	Study is limited to off-road navigation

B. THE APPLICATIONS OF DEEP REINFORCEMENT LEARNING IN AUTONOMOUS VEHICLE STEERING CONTROL

The application of DRL in steering control is highly limited in the literature. [40] proposed a DRL-based controller for path following of an unmanned surface vehicle to predict the steering angle. The DRL uses a deep deterministic policy gradient (DDPG) algorithm that functions as an actor-critic based RL algorithm for the controller. The DRL is used to self-develop the vehicle's path-following capability and to develop a steering angle controller. Four DDPG (A, B, C, & D) were compared with the benchmark controllers PID and DQN. DDPGD performed better, and the unmanned surface vehicle was able to have self-learning ability. This capability can be applied to solve problems dealing with uncertain environmental conditions. However, unmolded dynamic terms or environmental disturbances were not covered, and the vehicle did not properly follow its target path. Table 2 presents a summary of the study.

TABLE 2
SUMMARY OF THE APPLICATION OF DRL IN AUTONOMOUS VEHICLE STEERING CONTROL

Reference	Methodology	Objective	The algorithm used for evaluation	Result	Limitation
[40]	DRL	To self-develop vehicle's path following capability through interacting with nearby environment.	DDPG A, B, C, & D	The unmanned surface vehicle demonstrated self-learning ability; can be applied to deal with uncertain environmental conditions	Susceptible to environmental disturbances

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

C. APPLICATIONS OF HYBRID METHODS IN AUTONOMOUS VEHICLE STEERING CONTROL

In AVs steering control hybrid DL algorithms were applied for steering control. Hybrid algorithms typically solve each other's limitations to produce a more powerful and robust structure of the DL algorithm. For example, [83] hybridized CNN and long-short term memory (LSTM)/recurrent neural network (RNN) to predict the steering angle. Three DNN-based steering angle prediction algorithms were independently designed by Chauffeur [84], Autumn [84], and Rambo [85]. The Chauffeur model included one CNN for extracting features from image and LSTM/RNN model for steering angle prediction; the Autumn model had five (5) CNNs layers connected together and an LSTM/RNN layer, and the Rambo model had three (3) CNNs layers whose output was combined at the final layer. The Chauffeur model resisted snowy images; the Rambo model fought with foggy images, and only the Autumn model performed well across different scenarios. The CNN model extracted features from captured image, with one LSTM predicting the steering angles. The proposed CNN and LSTM/RNN solved asymmetry, weighted model fusion and steering angles prediction. However, it was affected by failure rates. [41] hybridized CNN and state-transitive LSTM (CNN-LSTM) with multi-auxiliary task to improve speed and steering angles prediction. CNN and LSTM were combined through sharing the CNN's feature layer parameters with an end-to-end autonomous master model. CNN was used for image classification, and LSTM addressed the time sequence problem and predicted the steering angles. The end-to-end autonomous model utilized the auxiliary task to simultaneously predict the steering angle and speed. The result indicated that the performance of the proposed CNN-LSTM was better than CNN and state-transitive LSTM. However, the study only covered limited driving scenarios. In another study, [43] combined DNN and CNN (DNN-CNN) to address steering control and speed command problems. CNN used four hidden layers, while in the DNN Dropout and ReLu activation were used at each hidden layer. DNN-CNN was applied to control the steering. The results showed that DNN-CNN performed better than DNN. The proposed finite state machine yielded less autonomous behaviors. The summary of the studies that used the hybrid DL algorithm for steering control is presented in Table 3.

VI. DRIVING SCENARIOS FOR AUTONOMOUS VEHICLES IN SIMULATED ENVIRONMENT

Driving simulators are application software that places the driver in an imitated environment that resembles a real driving environment. Unlike aircraft simulators, driving simulators bolster considerably higher than driver training. Researchers and engineers now utilize driving simulators with advanced features in the design of vehicles, smart highway design, and studies of human factors like behaviors of the human driver under the influence of alcohol and drugs, and severe weather conditions. It ensures a safer environment for testing that can be controlled under repeated measurements and is also cost-effective. Engineers and researchers acknowledge that the estimations acquired can assist them to predict equal estimates in the real world that lead to a superior comprehension of the mind-boggling driver-vehicle-roadway cooperation in dangerous driving circumstances. In consequence, these studies help reduce traffic-related injuries and deaths on the highways [39].

Driving simulators are possibly the best state-of-the-art software of computer-aided kinematic and dynamic simulation and can be regarded as one of the biggest triumphs in the field, as considered by Jia [86]. Although various driving simulators have been developed, in this paper we consider only the driving simulators used in the project we have reviewed; for example, the Car Simulator (CARSIM) [2], the Open Racing Car Simulator (TORCS) [19], [42] Prescan [79], CarND Udacity [39], Gazebo [18], Udacity [83], and Grand Theft Auto V (GTAV) [41]. However, other researchers did not provide information on the driving simulators used in the project. The summary of the simulators with the corresponding description is presented in Table 4. It is found in the literature that different projects used different scenarios depending on the objective of the project. Table 5 shows different scenarios from various projects depicted in the AV-simulated environment.

TABLE 3
STEERING CONTROL METHODS USING HYBRID DEEP LEARNING ALGORITHM

Reference	Methodology	Objective	The algorithm used for evaluation	Result	Limitation
[83]	DNN, CNN and LSTM/RNN	To increase resiliency of the CNN	Chauffeur, Autumn, and Rambo	Weighted model fusion on average achieved 40% accuracy as compared to separate algorithms	High rates of failure

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

[41]	CNN-LSTM	To improve steering angle prediction and vehicle speed	CNN and traditional LSTM	The proposed method is robust and effective	Covered limited driving scenarios
[43]	DNN-CNN	Not applicable	DNN-CNN	Results revealed that DNN-CNN outperformed DNN	The projected finite state machine contained fewer autonomous behaviors

TABLE 4
DRIVING SIMULATORS AND DESCRIPTIONS

Reference	Driving Simulator	Description
[77]	Torch 7	Scientific computing framework for extensive support within the machine learning framework
[2]	CARSIM	Provides labeled data for the training
[19]	TORCS 1.3.7	Used as research platform; runs on linux; used for car racing; allows users to develop their own vehicle controllers without human intervention
[42]	TORCS	Used as research platform; runs on linux; used for car racing; human driver controls the vehicle in the simulation and enables the collection of data
[79]	PreScan	Physics-based simulation platform utilized in the automotive industry to develop advanced driver assistance systems based on sensor technologies such as laser/LiDAR, GPS and camera
[39]	CarND Udacity	Flexible software; allows simulation of car movement in manual and automatic mode
[18]	Gazebo	Built using Robot operating system in Ubuntu; safety control module; evaluated performance of tracks in software before real track test
[83]	Udacity	Allows flexible simulation environment
[41]	Grand Theft Auto V. (GTAV)	Action adventure gaming platform for driving
[81]	Grand Theft Auto (GTA)	Video game platform for driving
[41]	Carla	Open source urban driving simulator; provides rich scene and numerous sensor information comprising camera, depth, LIDAR measurements, etc.

TABLE 5
DRIVING SCENARIOS FROM DIFFERENT PROJECTS

Reference	**Driving scenario**
[77]	Operated in diverse road conditions like highways, local and residential roads in rain, sun, fog, day and night with lighting and cloudy conditions
[2]	Road course, traffic, wind and screen captures
[19]	Lane markings in tracks mimic similar to real world; not all traffic involved; other cars controlled by the games AI engine created nuisance and instigated crashes
[20]	Varied widespread driving conditions
[42]	40-meter long course with right and left-hand curves and traffic lights
[79]	Valet parking scenario
[39]	Modeled on one-hour driving in different styles in manual mode; tried not to collide with objects and not to move out of track; allows modeling styles that can be implemented on special polygons in real world only
[18]	Salient features: boundaries of tracks and parts of buildings
[40]	Based on path following
[76]	Considers straight paths and daytime driving only.
[48]	Videos captures highway driving in clear weather and daytime; various road types like residential with and without lane markings; .also contains all activities of the driver like staying in and switching lanes.
[80]	Smooth track with curves of 10 minutes at 50% throttle.
[44]	Contains divers' situations, marked with respective turning angles of steering wheel of up to 38k sequential frames; traffic lights were removed
[81]	More than 10,000km and 200 hours of driving on open road with various kinds of urban streets, highways and country roads; 2000 frames × 350 episodes and 50% of weather types in town used in the training environment; two-lane dual carriage way

TABLE 6
DEEP LEARNING FRAMEWORK AND LIBRARY

Reference	**Deep Learning framework**	**Operating System**	**Hardware Technology**
[77]	-	-	GPU
[2]	Caffe	NA	CPU
[19]	OpenAL 1.17.2	Ubuntu 14.04 LTS	GPU, NVIDIA CUDA
[20]	Tensorflow	NA	CPU, GPU
[42]	NA	NA	GPU
[79]	Caffe	NA	NA
[39]	Keras2 & Tensorflow	NA	CPU, NVIDIA GTX 1080Ti
[18]	-	Ubuntu	CPU, GPU
[43]	Keras	Raspbian	-
[80]	Tensorflow	Ubuntu MATE 16.04	CPU/GPU
[44]	Keras/Tensorflow	-	GPU (Nvidia Tesla K80)
[87]	Pytorch	-	GPU (NVIDIA GeForce GTX Titan X

VII. DEEP LEARNING FRAMEWORK AND LIBRARY

Numerous ML frameworks and libraries offer the possibility of utilizing GPU accelerators to speed up the learning

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

process with supported interfaces; some allow the use of optimized libraries like CUDA (cuDNN) and OpenCL to improve performance. The primary feature of multiple-core accelerators is the extreme parallel architecture, which enables GPUs to speed up computations that involve matrix-based operations. Software development in the ML community is vastly dynamic and has different layers of abstraction [27]. Also, advanced DL platforms are now becoming fashionable, most of all open source. Giant companies such as Google, Microsoft, Apple, and NVIDIA are investing in DL technologies to aid software and hardware innovations that will further advance DL performance that can be used for the next generation of smart-world products [88]. Table 6 presents different DL frameworks and descriptions that have been utilized to implement various projects of steering angle prediction using DL, focusing on those projects that make the DL framework information available.

A. KERAS

Keras is a high-level neural networks API written in phyton and capable of running on top of Tensorflow, CNTK or Theano. It has been developed with the focus of enabling fast experiments¬. being able to move from the idea to the result with the least possible delay is the key to doing proper research [89]. Keras utilized a DL library that allows for easy and fast prototyping (Hatcher and Yu [90]; Nguyen, Dlugolinsky [27]) through user friendliness, modularity and extensibility (Nguyen, Dlugolinsky [27]; Hatcher and Yu [90]). It also supports both CNN and RNN as well as combinations of the two. Keras also runs flawlessly on CPU and GPU (Nguyen, Dlugolinsky [27]; Hatcher and Yu [90]). However, keras is not a DL framework on its own and requires the use of Tensorflow, Keras with its high-level API integrates with Tensorflow, Theano, and CNTK and is friendly to developers and can incorporate other common ML packages like scikit-learn in Python (7). It has been widely embraced by researchers and industries over the last years. The latest keras version is 2.2.5 and implements the 2.2.* API [89].

Merits [90]:

- It is an open-source and fast-evolving tool and possesses backend tools from influential companies like Microsoft and Google.
- It is an accessible API for DL with good documentation.
- It offers a convenient way to rapidly define DL models on top of backend, for example, Tensorflow, CNTK, Theano. It wraps backend libraries, thereby abstracting their capabilities and hiding their complexity.

Demerits [27]:

- Modularity and simplicity come at the prize of being less flexible, which is not optimal for research in new architectures.
- Multi-GPU does not work 100% in terms of efficiency and user-friendliness, as pointed out by several benchmarks that used it with a Tensorflow backend.

B. CAFFE

Caffe is a DL framework prepared with expression, modularity and speed in mind. Caffe was developed by Yangqing Jia Berkeley Artificial Intelligence Research (BAIR) [90] by community [86] and the BVLC (Berkeley Vision and Learning Center) at UC Berkeley to offer expressive architecture and GPU support and primarily image classification in 2014 [36]. In Caffe DNNs are defined layer by layer. Accepted data sources for Caffe includes LevelDB or LMDB and Hierarchical Data Format (HDF5)− efficient databases with common image formats like JPEG, GIF, TIFF, PNG, PDF. Common and normalization layers provide numerous data vector processing and normalization operations. New layers have to be written in C++ CUDA [90]; [27]. Custom layers are also braced in Python but are less efficient [27]. Caffe can also be run in command line, MatLab and Python interfaces. It also runs on mobile platforms and bare CUDA devices, in addition to its extended use in the Apache Hadoop ecosystem using Spark, among others. As part of Facebook Open Source and Research, Caffe2 was built from the earlier Caffe project that implemented more Python API which supports Windows, Mac OS X, Linux Android, iOS, and other platforms [90].

Merits [27]:

- It is easy to code with MatLab and Phyton's API/CLI.
- Its interface is suitable in image processing using CNN.
- Caffe Model Zoo contains pre-trained networks for fine-tuning.

Demerits [27]:

- Its custom layers have to be written in C++.
- No further active developments; the latest available version of Caffe is 1.0 (April 2017) and later merged to become Pytorch.
- Definition of the static model graph does not fit numerous RNNs applications that require variable sized inputs.
- Caffe prototxt files for model definition are overly cumbersome for modular DNN models and very deep in comparison with other frameworks like ResNet or GoogLeNet.

C. TENSORFLOW

Tensorflow is an open-source software used in numerical computation that involves data flow graphs ([27]&[90]). It constitutes an end-to-end platform that has a comprehensive, flexible ecosystem of tools, libraries, and community resources that allow researchers and developers to meet up with the state-of-the-art deployment of powered ML applications. [91] Tensorflow was developed and maintained by the Google Brain team that is within Google's Machine Intelligence research organization for DL and ML [27], [90]. It is currently released under the Apache 2.0 open source license although released by Google in 2015; version 1.0.0 was released in 2017 [90]. Tensorflow is intended for extensive-scale distributed training and inference and is meant for research, production system and development.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

Merits [27]:

- It provides a basis for DL research and development through its numerical library for data flow programming.
- It outperforms other DL tools; it is a fast evolving, open source supported by Google.
- It is efficient in multi-GPU settings, allows GPU/CPU computing, mobile computing, and higher scalability of computation through machines and substantial data sets.
- Comfortable building and deployment in the cloud, browser, or on the device, no matter the language used and the model's training using intuitive APIs like Keras with smooth execution, allowing for fast model iteration and easy debugging [91].
- Simple and flexible architecture that takes new ideas from concept to code to develop state-of-the-art models [91].
- Its API can be distributed across multiple GPUs, multiple machines, or TPUs. For training models, the API can also enable the distribution of existing models and training code with minimal code changes [91].

Demerits

- All computational flow must be developed as a static graph, even though the Tensorflow Fold package in Google-AI-blog 2017 tried to lessen the problem [92].
- Remains lower-level API that is difficult to utilize directly in creating DL models [27].

VIII. OPTIMIZER

In this section, we present the solvers used for optimizing DL architectures during the training of different projects for steering angle prediction of AVs. Among them the Adam solver is most popular, as clearly shown in the literature survey. Numerous solvers are available for updating weights and bias after each iteration. Different researchers compare different solvers and analyze how they influence the training process. Each model is trained multiple times with different solvers to find the best optimizer for use in the final DL model. The typical solvers used by researchers include the Adam solver derived from adaptive moment estimation, stochastic gradient descent (SGD) solver, and Nesterov's accelerated gradient (NAG) solver. The Adam solver computes adaptive learning rates for each parameter and is a gradient-based optimization method. [93] Optimizers help in calculating adaptive learning rates of neural networks [94].

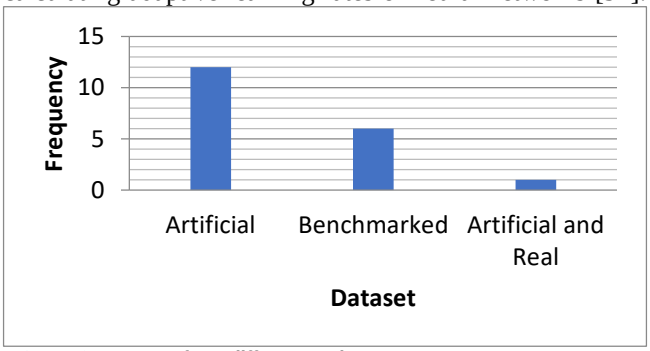

FIGURE 6. Datasets from different projects

Also, the solver selection is said to be an important factor in training neural nets [2]. [87] Adam is a technique for efficient stochastic optimization, which only requires first-order gradients with minor memory requirements.

As shown in Table 7, 'Y' indicates the use of a particular solver otherwise indicates non-usage of the solver.. As shown, only one research discussed all the three solvers used at the preliminary experiment stage for comparison. Other researchers discussed only the best solver used in the study.

TABLE 7
THE SUMMARY OF THE SOLVERS USED IN DIFFERENT PROJECTS

Reference	Adam	NAG	SGD
[2]	Y	Y	Y
[20]	-	Y	-
[39]	Y	-	-
[18]	Y	-	-
[40]	Y	-	-
[41]	Y	-	-
[48]	Y	-	-
[44]	Y	-	-
[41]	Y	-	-

IX. ANALYSIS AND DISCUSSION

This survey presents recent advances in applying DL algorithms to improve the accuracy of steering angle prediction of AVs (as shown in Tables 1, 2, 3 and 4). Driving simulators and corresponding descriptions (see Table 5), driving scenarios from different projects (Table 6), DL framework and libraries (Table 7) and solvers used in different projects (Table 8) are discussed in the survey. The CNN, DRL and hybrid algorithms are the main DL algorithms that received remarkable attention from the research community. It has been found that the DL algorithms can considerably improve the steering angle prediction of autonomous vehicles.

These DL algorithms work very well in new and unknown scenarios when the dataset is very large, i.e. have been trained on many hours of driving in different scenarios. There has been a breakthrough in the improvement of DL in AVs as many manufacturers, insurance companies and researchers are actively involved, and the field is gaining increased attention as a leading future technology.

Most researchers used an artificial dataset, which is the dataset that is collected from driving simulators. Only one single project applied a real dataset to experiment on the application of DL on steering angle prediction. Real dataset refers to data collected from actual driving experiments in the real word and is typically difficult to collect and process. On the other hand, some researchers used benchmark data. Benchmark dataset refers to real data that are collected and stored in a public repository for researchers to use for research purposes – either freely available or subscription-based. As shown in Figure 6, only those projects that reveal the type of data have been extracted and presented. Although, real dataset is currently not readily available,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

researchers are working with companies and regularly provide benchmark datasets for continued research in the field. The top ten datasets are Astyx dataset Hires 2019, Barkeley DeepDrive, Landmarks, Landmark-v2, Level5, nuscenes Dataset, Open Images v5, Oxford Radar Robot Car Dataset, Pandaset and Waymo Open Dataset. The percentage spread of the datasets is shown in Figure 7, indicating which artificial dataset has the highest percentage.

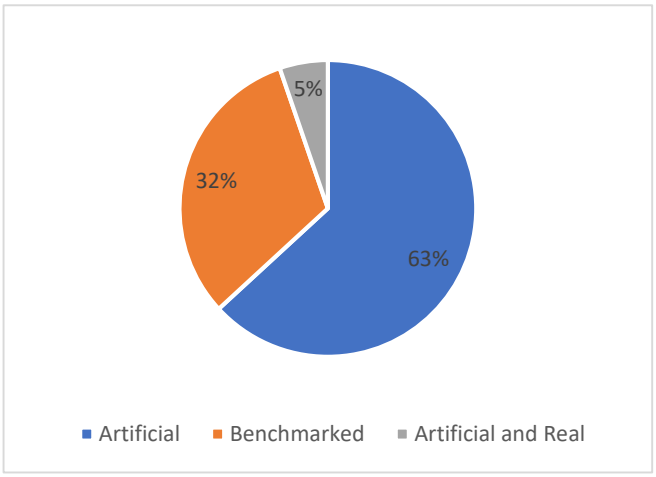

FIGURE 7. The percentage of the dataset used by researchers

Figure 8 depicts the publication trend starting from 2016 up to 2019, with 2018 recording the highest number of publications in the domain. The number of publications increased until 2018 and declined thereafter.

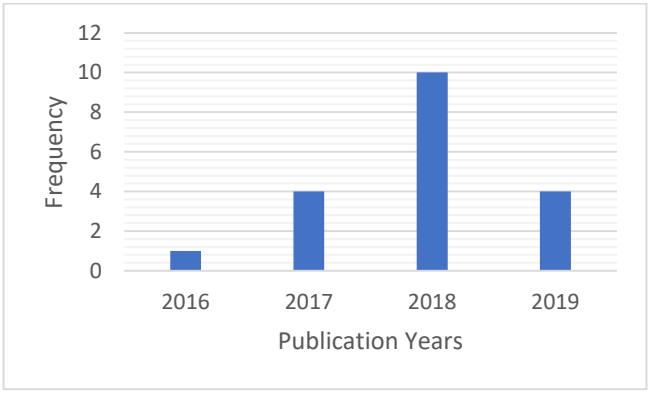

FIGURE 8. Publication trend

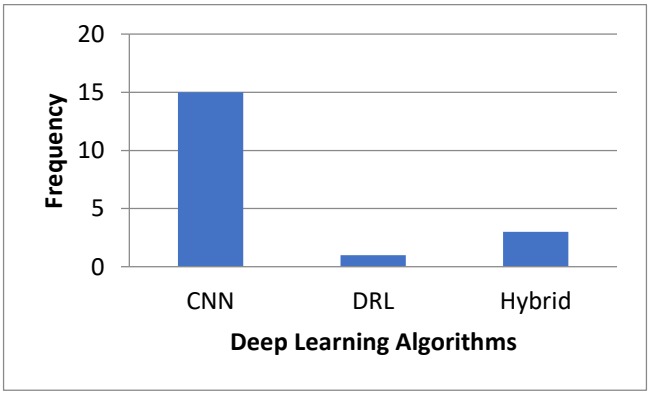

FIGURE 9. Deep learning architectures

Figure 9 presents the frequency of the DL architectures used in the steering angle prediction of AVs and clearly illustrates their popularity. CNN is a highly popular architecture, signifying its importance in steering angle prediction of AVs. The likely reason for CNN receiving such unprecedented attention from researchers lies in its ability to work effectively on images, which it does better than other DL architectures; it manages a high amount of images captured by the cameras in the AVs. The network takes in images as the input, trains and predicts the steering angle. On the other hand, hybrid DL architecture is also gaining momentum due to its advantages over single algorithms as two or more hybrid algorithms can complement each other to improve effectiveness and efficiency. This is achieved by eliminating the limitation of the individual algorithms. In terms of percentage, the CNN takes 79% as shown in Figure 10 of the total adoption in AVs steering angle prediction.

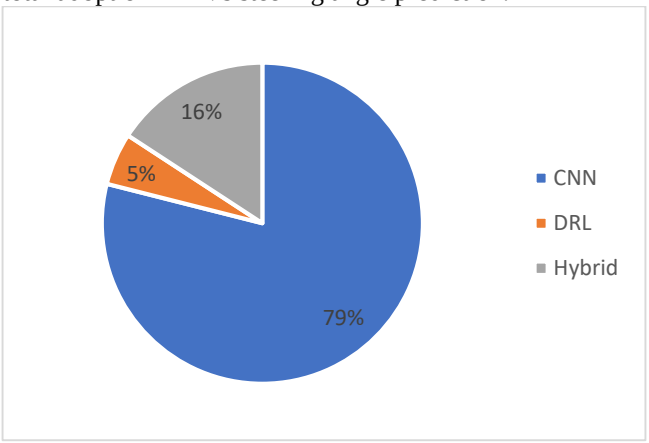

Figure 10. Deep learning architectures popularity

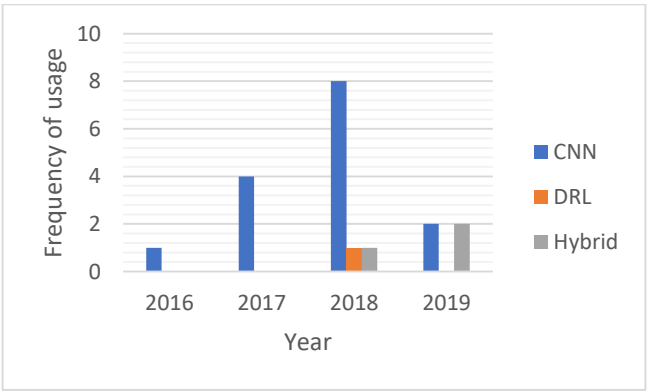

FIGURE 11. The frequency of different deep learning architecture per year

Figure 11 shows the different architectures of the DL used in each year. For example, only CNN was applied in the steering angle prediction of AVs in 2016; however, in 2018 other types of DL architectures such as DRL and Hybrid algorithms were used as well. CNN has appeared since 2016while hybrid has started to gain more attention in 2018. Kuutti, Bowden [4] Argued that the application of DL in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

autonomous vehicle control is becoming increasingly popular as DL algorithms have shown promising results in solving complex and non-linear control problems, in addition to their ability to apply learnt rules in new scenarios.

X. CHALLENGES AND FUTURE RESEARCH PROSPECTS

Despite the successes recorded in the steering control of AVs, there are still challenges lingering in the literature that require further study. In this section, we outline the challenges revealed in the literature survey and suggest possible approaches in solving the identified problems in the future. The challenges and the possible methods for addressing them are discussed as follows:

A. SYNTHETIC DATASET

From the survey conducted, Table 8 revealed that researchers in this domain heavily rely on artificial datasets to experiment with the application of DL in steering control of AVs. This is in agreement with the argument presented in [40] that most researchers in the domain used artificial datasets. The use of artificial datasets has challenges in conducting experiment with the aim of deploying the results in a real-world environment. Despite the fact that the simulated environment models the real-world environment, the scenario may differ from the simulated environment due to the peculiarities of the unexpected events likely to occur in the real world that are not captured in the artificial dataset generated from the simulated environment. Therefore, the experiment conducted in a simulated environment using an artificial dataset may not necessarily work in the real-world environment as the artificial dataset is not generated from a real-world event. We suggest researchers to collaborate with autonomous vehicle manufacturers like Mercedes Benz, Baidu, Volkswagen, Waymo, etc., to build a public repository of real-world datasets on steering control of AVs.

B. SLOW IN UNKNOWN ENVIRONMENT

It is found that CNN driving at higher speed remains a challenge. Researchers argue that CNN takes time to recover from bad mistakes, and it is found to be slow in responding to unknown scenarios [41]. Therefore, we propose a hybrid of CNN and another DL algorithm to overcome its shortcoming in steering control at high speed.

C. HIGH COMPUTATIONAL COST AND MEMORY CONSUMPTION

In respect to the DL structure, CNN requires high computational cost and consumes a lot of memory space. This constitutes a challenge to steering control of the AVs as it has negative implications on both the hardware and software, which in turn makes the vehicle more expensive. Researchers should investigate different approaches for reducing high computational cost and memory consumption of the DL structure applied in controlling the steering of AVs [91].

D. YET TO ATTAIN OPTIMUM STEERING PERFORMANCE

The optimum performance of steering control via DL has not yet been achieved, and there is still room for improvement to attain the optimum performance. Better performance of steering control can be attained through improving the neural network architecture, specifically through batch normalization and skip connections. In addition, it can be extended to take care of obstacle avoidance through investigating mediated perception techniques using conditional networks or through adding a temporal aspect to the steering angle prediction [81].

E. THE DNN HIGH FAILURE RATE

The DNN is found to be vulnerable to a relatively high rate of failures in controlling the steering of AVs [83]. Its optimal structure can significantly reduce steering control failure, which remains an unresolved problem. [83] suggested the extensive investigation of diversifying the DNN structure to solve its failure rate problem in steering control.

F. INSUFFICIENT TRAINING DATASET

It could be observed from the literature that using a sufficient dataset for training the DL algorithm to control the steering angle of the AVs has remained a challenge. However, the DL algorithms highly require a very large dataset for better performance. The performance of the DL algorithms increases as the data size increases. Therefore, a limited dataset limits the performance of the DL algorithms, thereby also limiting the efficiency, effectiveness and robustness of the steering angle prediction of the AVs. As such, we suggest building a large-scale dataset repository that is freely available or subscription-based to provide researchers access to large scale dataset on AVs steering control.

G. EGO-VEHICLE FAILS

The prediction of the steering angle of AVs can still be improved: the ego-vehicle still fails although it recovers from mistakes; it is not robust. The network cannot use RGB image as sole input and cannot directly extract with both semantic and depth information [41]. We suggest that researchers propose a new DL model for steering control of AVs with the capability to handle ego-vehicle failure and interact directly with semantic and depth information.

H. LIMITED COMBINATION OF DEEP LEARNING WITH CONTROL THEORIES

From the survey it can be observed that the studies mainly depend on the DL algorithms for steering control of the AVs, thus neglecting the classical control theories. That has deprived the DL-based controllers from the benefits of the classical control theories. Therefore, [2] suggested the combination of the classical control theory with DL algorithm to produce a new generation of controllers with an improved performance.

I. LONGITUDINAL CONTROL ISSUE

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

There is the challenge of lateral and longitudinal control in autonomous vehicle steering control. Adding autonomous longitudinal control remains an unresolved problem [19]. Therefore, researchers should improve the system by gathering more data from various tracks containing various driving scenarios (road conditions) to achieve successful control [42].

J. NIGHT DRIVING

The AVs are expected to drive during day and night time when deployed as commercial vehicles on public roads. However, only few works such as [77] and [41] include night driving in their respective studies, while most researchers have ignored night driving in their scenario, as reported in [76]. If AVs only rely on daytime driving, their scope of activities remains. We therefore suggest that researchers include both driving scenarios in future studies.

K. LACK OF TRANSPARENCY AND BLACK-BOX NATURE OF DEEP LEARNING

The DL has a "black box" nature ([95], [96] & [97]) and lacks transparency−a feature for which it has been criticized since it was proposed. This lack of transparency is partly due to its reliance on multiple nonlinear transformations. As such, nonlinearities can be challenging when tracing the consecutive layers of weights back to the input data to identify the features that provide the most significant decision or contribution [98]. As a result, it is difficult to ascertain that the driving prediction features are scientifically relevant to the data, such as gray images. Also, a model with brilliant performance may still have limited experimental utility. The issue of enhancing the interpretability of DL has in recent years become an uprising and expanding area of research, with several studies attempting to extract information on the essential features of driving prediction using various approaches ([99], [100], [101], [102]). However, the existing works only provide a mere overview of the problem [103].

Furthermore, ([104], [105], [106]) highlighted unanswered questions and issues like explaining what happens in the black box, how it works, and how comprehensive a model it is. Black boxes can be dangerous, and therefore delegating decisions to it without the possibility of interpretation may be risky as it can give rise to discrimination and trust issues. We suggest that future researchers apply explainable artificial intelligence to mitigate the issue of lack of interpretability in DL for steering angle prediction of AVs, in addition to answering open research questions.

XI. CONCLUSION

This paper presents a review of the steering control of AVs via DL architectures. Taxonomy, synthesis and analysis of the approaches that applied DL for the control of AVs are presented. The literature review has covered the following DL architectures in steering control: deep neural network, convolutional neural network, hybrid DL and deep reinforcement learning. It was found in the review that the methods used by the researchers mainly aim at providing proper predictions of steering angle. The review indicated that the convolutional neural network is very important in the domain and more heavily relied on by researchers than any other DL architecture. Research challenges and possible approaches for solving the outlined challenges were pointed out. Our review can serve as a starting point for new researchers who are interested in steering control of AVs via DL as well as expert researchers in AV steering control who can use it as a benchmark for proposing new steering control algorithms.

REFERENCES

1. Singh, S., *Critical reasons for crashes investigated in the national motor vehicle crash causation survey*. 2015.
2. Rausch, V., et al. *Learning a deep neural net policy for end-to-end control of autonomous vehicles*. in *2017 American Control Conference (ACC)*. 2017. IEEE.
3. Etherington, D., *Over 1,400 self-driving vehicles are now in testing by 80+ companies across the US.* 2020.
4. Kuutti, S., et al., *A Survey of Deep Learning Applications to Autonomous Vehicle Control*. Vol. PP. 2019.
5. Shreyas, V., et al., *Self-driving Cars: An Overview of Various Autonomous Driving Systems*, in *Advances in Data and Information Sciences*. 2020, Springer. p. 361-371.
6. Russakovsky, O., et al., *Imagenet large scale visual recognition challenge.* International journal of computer vision, 2015. **115**(3): p. 211-252.
7. Mohseni, F., S. Voronov, and E. Frisk, *Deep Learning Model Predictive Control for Autonomous Driving in Unknown Environments.* IFAC-PapersOnLine, 2018. **51**(22): p. 447-452.
8. Eraqi, H.M., M.N. Moustafa, and J. Honer, *End-to-end deep learning for steering autonomous vehicles considering temporal dependencies.* arXiv preprint arXiv:1710.03804, 2017.
9. Islam, M.Z., et al., *Self-driving car: a step to the future of mobility*. 2018, BRAC University.
10. Krogh, B. and C. Thorpe. *Integrated path planning and dynamic steering control for autonomous vehicles*. in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*. 1986. IEEE.
11. McGookin, E.W., et al., *Ship steering control system optimisation using genetic algorithms.* Control Engineering Practice, 2000. **8**(4): p. 429-443.
12. Tan, H.-S., et al., *Development of an automated steering vehicle based on roadway magnets-a case study of mechatronic system design.* IEEE/ASME transactions on mechatronics, 1999. **4**(3): p. 258-272.
13. Linhui, G. and W.R.Z.M.G. Lie, *Study on Lateral Fuzzy Control of Unmanned Vehicles Via Genetic Algorithms [J].* Journal of Mechanical Engineering, 2012. **6**.
14. Layne, J.R. and K.M. Passino, *Fuzzy model reference learning control for cargo ship steering.* IEEE Control Systems Magazine, 1993. **13**(6): p. 23-34.
15. Cao, C.-T., et al., *System for controlling brake pressure based on fuzzy logic using steering angle and yaw speed*. 1997, Google Patents.
16. Lee, A., *A preview steering autopilot control algorithm for four-wheel-steering passenger vehicles.* 1992.
17. Khadim, S., et al., *A non-cooperative rear-end collision avoidance scheme for non-connected and heterogeneous environment.* Computer Communications, 2020. **150**: p. 828-840.
18. Pan, Y., et al. *Agile autonomous driving using end-to-end deep imitation learning*. in *Robotics: science and systems*. 2018.

19. Sharma, S., G. Tewolde, and J. Kwon. *Behavioral Cloning for Lateral Motion Control of Autonomous Vehicles Using Deep Learning*. in *2018 IEEE International Conference on Electro/Information Technology (EIT)*. 2018. IEEE.
20. Do, T.-D., et al. *Real-Time Self-Driving Car Navigation Using Deep Neural Network*. in *2018 4th International Conference on Green Technology and Sustainable Development (GTSD)*. 2018. IEEE.
21. Oussama, A. and T. Mohamed. *A Literature Review of Steering Angle Prediction Algorithms for Self-driving Cars*. in *International Conference on Advanced Intelligent Systems for Sustainable Development*. 2019. Springer.
22. Arnold, R. and D. Fletcher, *A research synthesis and taxonomic classification of the organizational stressors encountered by sport performers.* Journal of sport and exercise psychology, 2012. **34**(3): p. 397-429.
23. Quinn, A.J. and B.B. Bederson. *Human computation: a survey and taxonomy of a growing field*. in *Proceedings of the SIGCHI conference on human factors in computing systems*. 2011.
24. Lei, Y., et al., *A review on empirical mode decomposition in fault diagnosis of rotating machinery.* Mechanical systems and signal processing, 2013. **35**(1-2): p. 108-126.
25. Pitropakis, N., et al., *A taxonomy and survey of attacks against machine learning.* Computer Science Review, 2019. **34**: p. 100199.
26. Hubel, D.H. and T.N. Wiesel, *Receptive fields, binocular interaction and functional architecture in the cat's visual cortex.* The Journal of physiology, 1962. **160**(1): p. 106-154.
27. Nguyen, G., et al., *Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey*. Artificial Intelligence Review, 2019. **52**(1): p. 77-124.
28. Jia, Y.a.S., Evan and Donahue, Jeff and Karayev, Sergey and Long, Jonathan and Girshick, Ross and Guadarrama, Sergio and Darrell, Trevor, *Caffe.* arXiv preprint arXiv:1408.5093, 2014.
29. Joshi, A.V., *Machine Learning and Artificial Intelligence*. 2019, Springer.
30. Wani, M.A., et al., *Advances in Deep Learning*. Vol. 57. 2020: Springer.
31. Aggarwal, C.C., *Neural networks and deep learning.* Cham: Springer International Publishing, 2018.
32. Nelson, D.R., et al., *Vector field path following for miniature air vehicles.* IEEE Transactions on Robotics, 2007. **23**(3): p. 519-529.
33. François-Lavet, V., et al., *An introduction to deep reinforcement learning.* Foundations and Trends® in Machine Learning, 2018. **11**(3-4): p. 219-354.
34. Li, Y., *Deep reinforcement learning: An overview.* arXiv preprint arXiv:1701.07274, 2017.
35. Mnih, V., et al., *Human-level control through deep reinforcement learning.* Nature, 2015. **518**(7540): p. 529-533.
36. Jia, Y., et al. *Caffe: Convolutional architecture for fast feature embedding*. in *Proceedings of the 22nd ACM international conference on Multimedia*. 2014. ACM.
37. Abe, M. and W. Manning, *Chapter 5 - Steering System and Vehicle Dynamics*, in *Vehicle Handling Dynamics*, M. Abe and W. Manning, Editors. 2009, Butterworth-Heinemann: Oxford. p. 149-164.
38. Abe, M., *Chapter 5 - Steering System and Vehicle Dynamics*, in *Vehicle Handling Dynamics (Second Edition)*, M. Abe, Editor. 2015, Butterworth-Heinemann. p. 139-152.
39. Smolyakov, M., et al. *Self-Driving Car Steering Angle Prediction Based On Deep Neural Network An Example Of CarND Udacity Simulator*. in *2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT)*. 2018. IEEE.
40. Woo, J., C. Yu, and N. Kim, *Deep reinforcement learning-based controller for path following of an unmanned surface vehicle.* Ocean Engineering, 2019. **183**: p. 155-166.
41. Wang, D., et al., *End-to-End Self-Driving Using Deep Neural Networks with Multi-auxiliary Tasks.* Automotive Innovation, 2019: p. 1-10.
42. Sharma, S., G. Tewolde, and J. Kwon. *Lateral and Longitudinal Motion Control of Autonomous Vehicles using Deep Learning*. in *2019 IEEE International Conference on Electro Information Technology (EIT)*. 2019. IEEE.
43. Simmons, B., et al. *Training a Remote-Control Car to Autonomously Lane-Follow using End-to-End Neural Networks*. in *2019 53rd Annual Conference on Information Sciences and Systems (CISS)*. 2019. IEEE.
44. Mund, S., et al. *Visualizing the Learning Progress of Self-Driving Cars*. in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018. IEEE.
45. Zeiler, M.D. and R. Fergus. *Visualizing and understanding convolutional networks*. in *European conference on computer vision*. 2014. Springer.
46. Zeiler, M.D., G.W. Taylor, and R. Fergus. *Adaptive deconvolutional networks for mid and high level feature learning*. in *2011 International Conference on Computer Vision*. 2011. IEEE.
47. Rajamani, R., *Vehicle dynamics and control*. 2011: Springer Science & Business Media.
48. Kim, J. and J. Canny. *Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention*. in *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017.
49. He, K., et al. *Deep residual learning for image recognition*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
50. Simonyan, K. and A. Zisserman, *Very deep convolutional networks for large-scale image recognition.* arXiv preprint arXiv:1409.1556, 2014.
51. Sutton, R.S. and A.G. Barto, *Introduction to reinforcement learning. Vol. 135*. 1998, MIT press Cambridge.
52. Carreras, M., P. Ridao, and A. El-Fakdi. *Semi-online neural-Q/spl I.bar/leaming for real-time robot learning*. in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*. 2003.
53. Silver, D., et al. *Deterministic policy gradient algorithms*. 2014.
54. Lillicrap, T.P., et al., *Continuous control with deep reinforcement learning*. 2017, Google Patents.
55. Pomerleau, D.A. *Alvinn: An autonomous land vehicle in a neural network*. in *Advances in neural information processing systems*. 1989.
56. Jackel, L., et al. *An application of neural net chips: Handwritten digit recognition*. in *ICNN proceeding*. 1988.
57. Lee, D.-S. *Neural network models and their application to handwritten digit recognition*. in *IEEE 1988 International Conference on Neural Networks*. 1988. IEEE.
58. Pomerleau, D.A., et al., *Neural network simulation at Warp speed: How we got 17 million connections per second*. 1988, CARNEGIE-MELLON UNIV PITTSBURGH PA ARTIFICIAL INTELLIGENCE AND PSYCHOLOGY ….
59. Waibel, A., et al. *Phoneme recognition: neural networks vs. hidden Markov models vs. hidden Markov models*. in *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing*. 1988. Citeseer.
60. Dickmann, J., et al. *Radar contribution to highly automated driving*. in *2014 44th European Microwave Conference*. 2014. IEEE.
61. Liu, J., et al. *A multi-stage optimization formulation for MPC-based obstacle avoidance in autonomous vehicles using a LIDAR sensor*. in *Dynamic Systems and Control Conference*. 2014. American Society of Mechanical Engineers.
62. Fang, Y., *Connected Vehicles Make Transportation Faster, Safer, Smarter, and Greener!* IEEE Transactions on Vehicular Technology, 2015. **64**(12): p. 5409-5410.
63. Alkheir, A.A., M. Aloqaily, and H.T. Mouftah, *Connected and autonomous electric vehicles (caevs).* IT Professional, 2018. **20**(6): p. 54-61.
64. Kocić, J., N. Jovičić, and V. Drndarević, *An End-to-End Deep Neural Network for Autonomous Driving Designed for*

Embedded Automotive Platforms. Sensors, 2019. **19**(9): p. 2064.
65. Xu, D., D. Anguelov, and A. Jain. *Pointfusion: Deep sensor fusion for 3d bounding box estimation*. in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018.
66. Oh, S.-I. and H.-B. Kang, *Fast occupancy grid filtering using grid cell clusters from LIDAR and stereo vision sensor data.* IEEE Sensors Journal, 2016. **16**(19): p. 7258-7266.
67. Chavez-Garcia, R.O. and O. Aycard, *Multiple sensor fusion and classification for moving object detection and tracking.* IEEE Transactions on Intelligent Transportation Systems, 2015. **17**(2): p. 525-534.
68. Cho, H., et al. *A multi-sensor fusion system for moving object detection and tracking in urban driving environments*. in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014. IEEE.
69. Nose, Y., et al. *A Study on a Lane Keeping System using CNN for Online Learning of Steering Control from Real Time Images*. in *2019 34th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*. 2019. IEEE.
70. Humaidi, A.J. and M.A. Fadhel. *Performance comparison for lane detection and tracking with two different techniques*. in *2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA)*. 2016. IEEE.
71. Zhao, J., B. Xie, and X. Huang. *Real-time lane departure and front collision warning system on an FPGA*. in *2014 IEEE High Performance Extreme Computing Conference (HPEC)*. 2014. IEEE.
72. Li, C., et al. *A model based path planning algorithm for self-driving cars in dynamic environment*. in *2015 Chinese Automation Congress (CAC)*. 2015. IEEE.
73. Yoon, S., et al., *Recursive path planning using reduced states for car-like vehicles on grid maps.* IEEE Transactions on Intelligent Transportation Systems, 2015. **16**(5): p. 2797-2813.
74. Wang, D. and F. Qi. *Trajectory planning for a four-wheel-steering vehicle*. in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*. 2001. IEEE.
75. Kong, J., et al. *Kinematic and dynamic vehicle models for autonomous driving control design*. in *2015 IEEE Intelligent Vehicles Symposium (IV)*. 2015. IEEE.
76. Chen, Z. and X. Huang. *End-to-end learning for lane keeping of self-driving cars*. in *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017.
77. Bojarski, M., et al., *End to end learning for self-driving cars.* arXiv preprint arXiv:1604.07316, 2016.
78. Bojarski, M., et al., *Explaining how a deep neural network trained with end-to-end learning steers a car.* arXiv preprint arXiv:1704.07911, 2017.
79. Jhung, J., et al. *End-to-End Steering Controller with CNN-based Closed-loop Feedback for Autonomous Vehicles*. in *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018. IEEE.
80. Bechtel, M.G., et al. *Deeppicar: A low-cost deep neural network-based autonomous car*. in *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 2018. IEEE.
81. Toromanoff, M., et al. *End to End Vehicle Lateral Control Using a Single Fisheye Camera*. in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018.
82. Wang, Q., et al., *End-to-End Autonomous Driving: An Angle Branched Network Approach.* IEEE Transactions on Vehicular Technology, 2019. **68**(12): p. 11599-11610.
83. Wu, A., et al. *Model Fusion: Weighted N-Version Programming for Resilient Autonomous Vehicle Steering Control*. in *2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*. 2018. IEEE.
84. Zhang, M., et al. *DeepRoad: GAN-based metamorphic testing and input validation framework for autonomous driving systems*. in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 2018.
85. Yan, M., L. Wang, and A. Fei, *ARTDL: Adaptive Random Testing for Deep Learning Systems.* IEEE Access, 2019.
86. Jia, Y.S., Evan Donahue, Jeff Karayev, Sergey Long, Jonathan Girshick, Ross Guadarrama, Sergio and Darrell, Trevor, *Caffe: Convolutional Architecture for Fast Feature Embedding.* arXiv preprint arXiv:1408.5093, 2014.
87. Kingma, D.P. and J. Ba, *Adam: A method for stochastic optimization.* arXiv preprint arXiv:1412.6980, 2014.
88. Alpha, C.T., *Train and run faster than never before.* 2020.
89. About keras. (2020, January 20). Keras.Io. https://keras.io/
90. Hatcher, W.G. and W. Yu, *A survey of deep learning: platforms, applications and emerging research trends.* IEEE Access, 2018. **6**: p. 24411-24432.
91. Why TensorFlow. (2019, December 18). tensorflow.org. https://www.tensorflow.org/
92. Patel, A.B., M.T. Nguyen, and R. Baraniuk. *A probabilistic framework for deep learning*. in *Advances in neural information processing systems*. 2016.
93. Kermani, B.G., S.S. Schiffman, and H.T. Nagle, *Performance of the Levenberg–Marquardt neural network training method in electronic nose applications.* Sensors and Actuators B: Chemical, 2005. **110**(1): p. 13-22.
94. Sparks, E.R., et al. *Keystoneml: Optimizing pipelines for large-scale advanced analytics*. in *2017 IEEE 33rd international conference on data engineering (ICDE)*. 2017. IEEE.
95. Poznyak, T.I., I. Chairez Oria, and A.S. Poznyak, *Chapter3 - Background on dynamic neural networks*, in *Ozonation and Biodegradation in Environmental Engineering*, T.I. Poznyak, I. Chairez Oria, and A.S. Poznyak, Editors. 2019, Elsevier. p. 57-74.
96. Alain, G. and Y. Bengio, *Understanding intermediate layers using linear classifier probes.* arXiv preprint arXiv:1610.01644, 2016.
97. Yosinski, J., et al., *Understanding neural networks through deep visualization.* arXiv preprint arXiv:1506.06579, 2015.
98. Suk, H.-I., et al., *Latent feature representation with stacked auto-encoder for AD/MCI diagnosis.* Brain Structure and Function, 2015. **220**(2): p. 841-859.
99. Deshpande, G., et al., *Fully connected cascade artificial neural network architecture for attention deficit hyperactivity disorder classification from functional magnetic resonance imaging data.* IEEE transactions on cybernetics, 2015. **45**(12): p. 2668-2679.
100. Kim, J., et al., *Deep neural network with weight sparsity control and pre-training extracts hierarchical features and enhances classification performance: Evidence from whole-brain resting-state functional connectivity patterns of schizophrenia.* Neuroimage, 2016. **124**: p. 127-146.
101. Liu, S., et al. *Early diagnosis of Alzheimer's disease with deep learning*. in *2014 IEEE 11th international symposium on biomedical imaging (ISBI)*. 2014. IEEE.
102. Suk, H.-I., et al., *State-space model with deep learning for functional dynamics estimation in resting-state fMRI.* NeuroImage, 2016. **129**: p. 292-307.
103. Lipton, Z.C., *The mythos of model interpretability.* Queue, 2018. **16**(3): p. 31-57.
104. Doshi-Velez, F. and B. Kim, *Towards a rigorous science of interpretable machine learning.* arXiv preprint arXiv:1702.08608, 2017.
105. Hofman, J.M., A. Sharma, and D.J. Watts, *Prediction and explanation in social systems.* Science, 2017. **355**(6324): p. 486-488.
106. Weller, A., *Challenges for transparency.* arXiv preprint arXiv:1708.01870, 2017.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3017883, IEEE Access

USMAN GIDADO MANZO is currently Resource person (Senior Education Officer) at Computer Department, ETF-Community Education Resource Centre Gombe, since 2012. He received his B. Sc. Computer Science degree in 2011 from Gombe state university, Gombe state, Nigeria. He is currently undergoing his MSc Computer Science under the supervision of Dr. Haruna Chiroma at Abubakar Tafawa Balewa University, Bauchi, Nigeria. His current research topic is improving steering angle prediction of self-driving vehicles using hybrid deep learning algorithms. He is a member of Computer Professional (Registration Council) of Nigeria (CPN).

HARUNA CHIROMA received the B.Tech. degree from Abubakar Tafawa Balewa University, the M.Sc. degree from Bayero University Kano, and the Ph.D. degree from the University of Malaya, all in computer science, and the Post-Graduate Diploma in education from Usman Danfodio University. He was a Visiting Senior Lecturer with Abubakar Tafawa Balewa University, Bauchi, and a Lecturer with the Federal College of Education, Gombe. As a teacher, he has developed interest in advanced learning technology. He has published more than 100 academic articles in international refereed ISI WoS Journals, Edited Books, Conference Proceedings, and Local Journals. He participated in the 2017 and 2018 QS world universities ranking evaluation of world universities research strengths in computer science. His research interests include deep learning, nature-inspired algorithms for machine learning, with special focus on their applications to internet of vehicles, autonomous vehicles, the Internet of Things, big data analytics, edge computing, cybersecurity, fog computing, and cloud computing. He has served in various capacities in more than 20 international conferences across the world. He is a member of the ACM, INNS, NCS, IAENG, and the Association for Computing Machinery (ACM). He is an Associate Editor of the IEEE Access. He is a Leading Volume Editor of an edited book Advances on Computational Intelligence in Energy-the Applications of Nature-Inspired & Metaheuristic Algorithms in Energy (Heidelberg, Berlin: Springer), renowned series of Lecture Notes in Energy.

NAHLA ALJOJO received the bachelor's degree in computer science from King Abdulaziz University, Jidda, Saudi Arabia, the master's degree in Computer System and Information Technology from Washington International University in Prussia, PA, USA, and the Ph.D. in Computing from the University of Portsmouth. She was an Associate Professor with the Information System Department, Faculty of Computing and Information Technology, King Abdulaziz University. She is currently an Associate Professor with the Information Systems Department, Faculty of Computing and Information Technology, University of Jeddah, Jeddah. Her research interests include adaptivity in web-based educational systems, E-business, Leadership's studies, information security, E-leaning, and Education.

SAIDU ABUBAKAR obtained B.Tech computer science in 2014 and currently pursuing his MSc. Computer Science all in Abubakar Tafawa Balewa University, Bauchi, Nigeria in the year 2014. He is currently working on Collision detection of Internet of Vehicles and deep learning.

SEGUN I. POPOOLA received the B.Tech. degree (Hons.) in electronic and electrical engineering from the Ladoke Akintola University of Technology, Ogbomoso, Nigeria, and the M.Eng. degree in information and communication engineering from the Department of Electrical and Information Engineering, Covenant University, Ota, Nigeria. He is currently pursuing the Ph.D. degree with the School of Engineering, Faculty of Science and Engineering, Manchester Metropolitan University, Manchester, U.K. He was a Lecturer with the Department of Electrical and Information Engineering, Covenant University. He has authored and coauthored more than 50 academic papers published in international peer-reviewed journals and conference proceedings. His research interests include wireless communications, machine learning, radio propagation modeling, the Internet of Things (IoT), cybersecurity, and smart and connected cities (SCC). He is a member of the International Association of Engineers (IAENG) and a Registered Engineer with the Council for the Regulation of Engineering in Nigeria (COREN). He received the best Graduating Student Award from the Department of Electronic and Electrical Engineering, Faculty of Engineering and Technology (FET) in conjunction with the Nigerian Society of Engineers (NSE).

MOHAMMED ALI AL-GARADI received the PhD. Degree from the Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia. He has published several articles in academic journals indexed in well-reputed databases such as ISI and Scopus. His research interests include cybersecurity, online social networking, and machine learning text mining, deep learning and IoT.