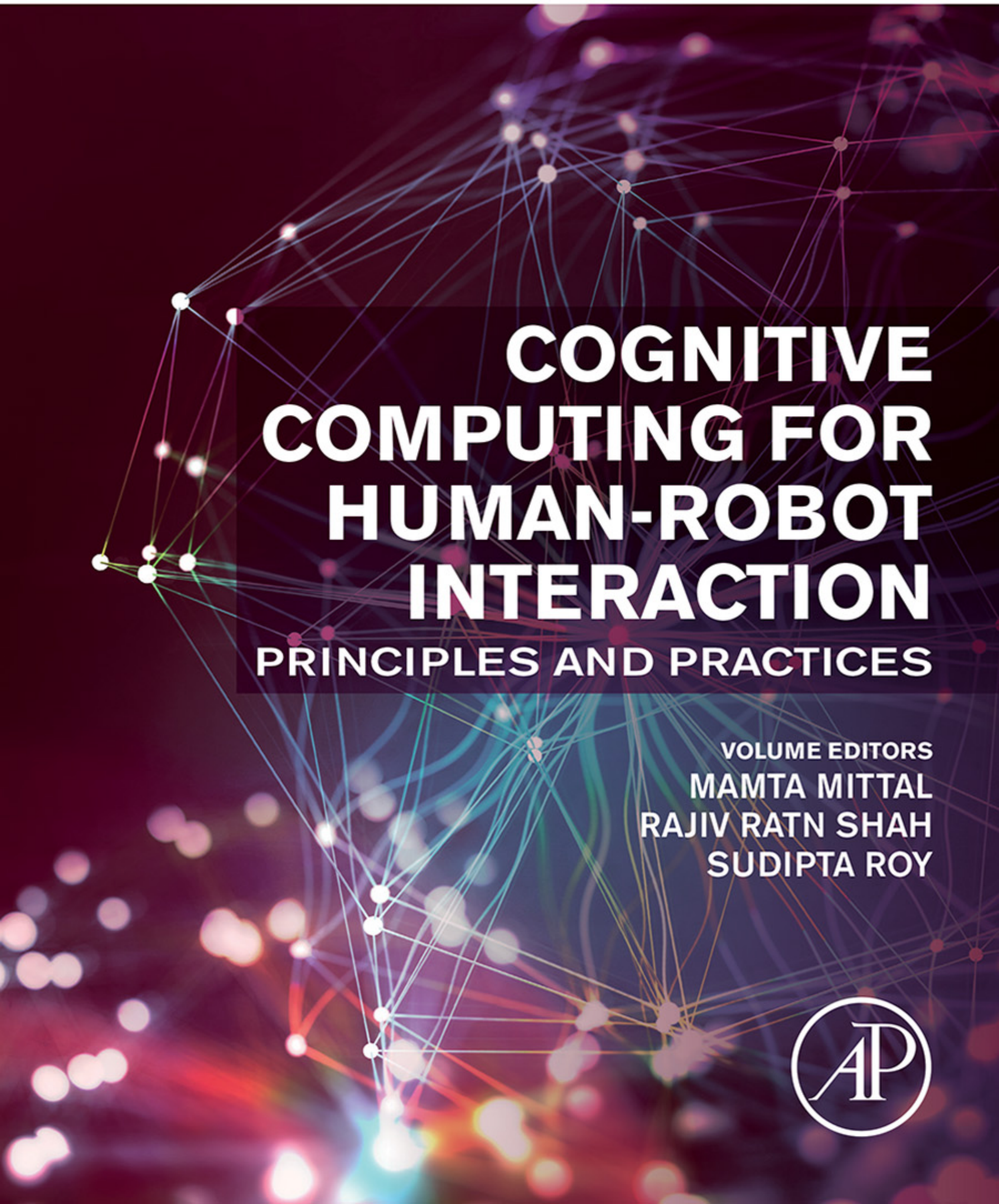
COGNITIVE DATA SCIENCE IN SUSTAINABLE COMPUTING

SERIES EDITOR: ARUN KUMAR SANGAIAH



COGNITIVE COMPUTING FOR HUMAN-ROBOT INTERACTION

PRINCIPLES AND PRACTICES

VOLUME EDITORS

MAMTA MITTAL

RAJIV RATN SHAH

SUDIPTA ROY

Cognitive Computing for Human-Robot Interaction

Principles and Practices

Cognitive Data Science in Sustainable Computing

Cognitive Computing for Human-Robot Interaction

Principles and Practices

Edited by

Mamta Mittal
Department of Computer Science and Engineering, G. B. Pant Government Engineering College, New Delhi, India

Rajiv Ratn Shah
Department of Computer Science and Engineering (joint appointment with the Department of Human-centered Design), IIIT-Delhi, New Delhi, India

Sudipta Roy
PRTTL, Washington University in Saint Louis, Saint Louis, MO, United States

Series Editor

Arun Kumar Sangaiah
School of Computing Science and Engineering,Vellore Institute of Technology (VIT), Vellore, India

ELSEVIER

ACADEMIC PRESS
An imprint of Elsevier

Academic Press is an imprint of Elsevier
125 London Wall, London EC2Y 5AS, United Kingdom
525 B Street, Suite 1650, San Diego, CA 92101, United States
50 Hampshire Street, 5th Floor, Cambridge, MA 02139, United States
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, United Kingdom

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Cataloging-in-Publication Data
A catalog record for this book is available from the Library of Congress

ISBN: 978-0-323-85769-7

Publisher: Mara Conner
Acquisitions Editor: Sonnini R. Yura
Editorial Project Manager: Rachel Pomery
Production Project Manager: Niranjan Bhaskaran
Cover Designer: Matthew Limbert

Typeset by MPS Limited, Chennai, India

About the editors

Mamta Mittal graduated in Computer Science & Engineering from Kurukshetra University Kurukshetra in 2001 and received a Master's degree (Honors) in Computer Science & Engineering from YMCA, Faridabad. She completed her PhD at Thapar University Patiala in Computer Science and Engineering. Her research areas include data mining, big data, and machine learning. She has been teaching for the last 18 years with an emphasis on data mining, machine learning, soft computing, and data structure. She is a lifetime member of CSI. She has published and communicated more than 70 research papers in SCI, SCIE, and Scopus indexed Journals and attended many workshops, FDPs, and Seminars. She has filed nine patents in the area of Artificial Intelligence and Deep Learning, out of which two are granted and many are published online. Presently, she is working at G.B. PANT Government Engineering College, Okhla, New Delhi (under Government of NCT Delhi) and supervising PhD candidates of Guru Gobind Singh Indraprastha University, Dwarka, New Delhi. She is the main editor of *Data Intensive Computing Applications for Big Data* published by IOS press, the Netherlands and also the editor of *Big Data Processing Using Spark in Cloud* by Springer and many more. She is the managing editor of the International Journal of Sensors, Wireless Communications and Control Published by Bentham Science. She is working on DST approved Project "Development of IoT based hybrid navigation module for mid-sized autonomous vehicles" with a research grant of 25 Lakhs and Handing Pradhan Mantri YUVA project for Entrepreneur Cell Activity as Faculty Coordinator/Representative from G. B. Pant Govt. Engineering College. She is the reviewer of many reputed journals (Including Wiley, IEEE Access, Elsevier, etc.) and has chaired several conferences.

Rajiv Ratn Shah currently works as an Assistant Professor in the Department of Computer Science and Engineering (joint appointment with the Department of Human-centered Design) at IIIT-Delhi. He is also the director of the MIDAS Lab at IIIT-Delhi. He received his PhD in Computer Science from the National University of Singapore, Singapore. Before joining IIIT-Delhi, he worked as a research fellow in Living Analytics Research Center at the Singapore Management University, Singapore. Before completing his PhD, he received his MTech and MCA degrees in Computer

Applications from the Delhi Technological University, Delhi and Jawaharlal Nehru University, Delhi, respectively. He has also received his BSc in Mathematics (Honors) from the Banaras Hindu University, Varanasi. Dr. Shah is the recipient of several awards, including the prestigious Heidelberg Laureate Forum and European Research Consortium for Informatics and Mathematics fellowships. He won the best student poster award at the 33rd AAAI Conference on Artificial Intelligence in Honolulu, HI, United States and won the best poster runner-up award at the 20th IEEE International Symposium on Multimedia (ISM) conference in Taichung, Taiwan. Recently, we also won the best poster and best industry paper awards at the 5th IEEE International Conference on Multimedia Big Data (BigMM) conference. He is also the winner of the 1st ACM India Student Chapter Grand Challenge 2019. He has also received the best paper award in the IWGS workshop at the ACM SIGSPATIAL conference 2016, San Francisco, CA, United States and was the runner-up in the Grand Challenge competition of ACM International Conference on Multimedia 2015, Brisbane, QLD, Australia. He is involved in organizing and reviewing many top-tier international conferences and journals. He is TPC cochair for IEEE BigMM 2019 and BigMM 2020. He has also organized the Multimodal Representation, Retrieval, and Analysis of Multimedia Content (MR2AMC) workshop in conjunction with the first IEEE MIPR 2018 and 20th IEEE ISM conferences. His research interests include multimedia content processing, natural language processing, image processing, multimodal computing, data science, social media computing, and the Internet of Things.

Sudipta Roy received his PhD in Computer Science & Engineering from the Department of Computer Science and Engineering, University of Calcutta. He is the author of more than 40 publications in refereed national/international journals and conferences, including IEEE, Springer, and Elsevier. He is the author of one book and many book chapters. He holds a US patent in medical image processing and filed an Indian patent in smart agricultural system. He has served as a reviewer of many international journals including IEEE, Springer, Elsevier, and IET, and international conferences. He has served international advisory committee member and program committee member of AICAE-2019, INDIACom-2019, CAAI 2018, ICAITA-2018, ICSESS-2018, INDIACom-2018, ISICO-2017, AICE-2017, and many more conferences. He is serving as an associate editor of IEEE Access, IEEE, and International Journal of Computer Vision and Image Processing, IGI Global Journal. He has more than 5 years of experience in teaching and research. His fields of research interests are healthcare image analysis, image processing, steganography, artificial intelligence, big data analysis, machine learning, and big data technologies. Currently, he is working at PRTTL, Washington University in St. Louis, Saint Louis, MO, United States.

Chapter 7

Cognitive computing in autonomous vehicles

Atharva Sandeep Vidwans
Pune, India

Introduction

Cognitive computing is a combination of cognitive and computer science. It represents self-learning systems without human interaction based on human cognition. It is a modern computing technique that works based on neural networks (NN). Being the building block that is based on human cognition, NN is called the brain of Cognitive Computing. It mimics the functioning of the central nervous system in human beings. It generates an output depending on the input given to the NN, which is a function of the input data. The detailed working of NN is explained later in the chapter. In the past decade, Computing Hardware is advancing with the developments in cognitive computing algorithms. Thus, due to a large amount of available data to train the models, robust hardware and efficient algorithms have facilitated the application of complex algorithms for problems with ease. They have led developments in many fields, from medical and robotics to internet security.

Cognitive computing has led to the development of a new field. Unlike previous computing techniques, it is not about solving complex algorithms or doing fast calculations but finding new relations in the data. That is finding patterns in the data, the quality which was lacking in previous techniques.

So, what is cognitive computing?

It is the use of computerized models to simulate the process similar to human thoughts in complex situations in which decision-making is crucial, or tasks which require human-level expertise. Thus, cognitive computing can be defined as a problem-solving approach that uses hardware or software to approximate the form or function of the natural cognitive process.

Cognitive Computing for Human-Robot Interaction. DOI: https://doi.org/10.1016/B978-0-323-85769-7.00008-2

Natural cognition

Learning is what a system must do to improve its performance of the system, based on experience rather than hard programming. Constant learning is a critical defining feature in cognitive computing.

How the system processes or acquires the natural stimuli from the environment is called perception. This data acquired by the perception is used for the learning task.

Cognitive computing consists of two fundamental approaches:

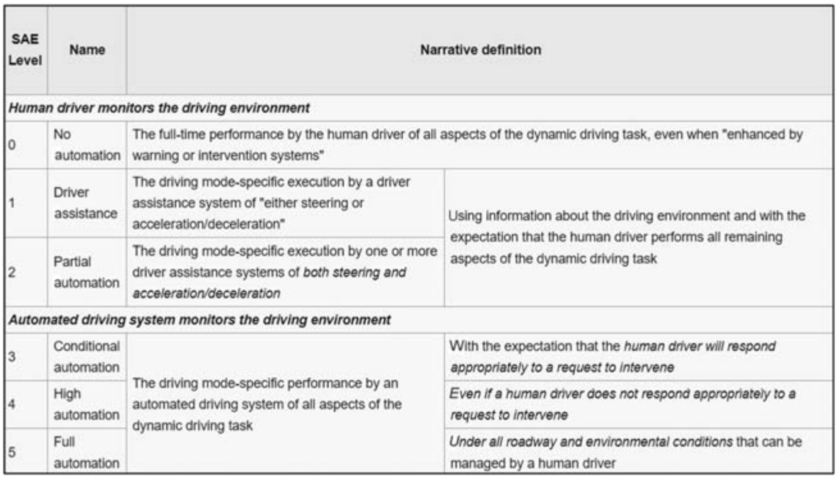
1. Neuromorphic architecture approach: It is an approach in which there is specialized hardware that simulates neural activity that is called neuromorphic components or systems. The basic idea is to create neural network models after biological systems or components, such as neurons or synapses. They consist of hardware with a Neuro synaptic chip system that is currently developed by IBM containing upto 10 billion neurons and a hundred trillion synapses that consume around 1 kW of power called system of neuromorphic adaptive plastic scalable electronics (SyNAPSE) board.
2. Software centric approach: this method emulates neural function in software in on conventional von Neumann hardware architecture.

Many times, the term "Cognitive Computing" is used interchangeably with artificial intelligence (AI) that is the umbrella term for all the technologies related to decision-making that rely on data without direct human intervention. Several AI techniques are used to build cognitive models that mimic thought processes that include but are not limited to machine learning, deep learning, NN, neuro linguist programming (NLP).

Introduction to autonomous driving

Taxonomy for autonomous vehicle

Due to the recent advancements in cognitive computing techniques, they are used for a wide range of applications. One such prominent application is autonomous driving. By drawing the understanding from biological cognition and behavior generation in our brain, human drivers, and autonomous vehicles (AVs) coexist better. Cognitive computing in autonomous driving is based on the fact that neither the humans nor the AVs are immune to failure. Rather, it is been observed that humans and self-driving are prone to fail at different things. Thus can be used together no nullify the effects, creating safe and secure travel. Neither the machine nor the human is the best driver, but the combination of both. Therefore, fusing the strength from both sides and removing the lacunas is a viable option. It is quite natural to train the cognitive model that can estimate the risk and possibly avoid one (Nützel, 2018).



SAE Level	Name	Narrative definition	
Human driver monitors the driving environment			
0	No automation	The full-time performance by the human driver of all aspects of the dynamic driving task, even when "enhanced by warning or intervention systems"	
1	Driver assistance	The driving mode-specific execution by a driver assistance system of "either steering or acceleration/deceleration"	Using information about the driving environment and with the expectation that the human driver performs all remaining aspects of the dynamic driving task
2	Partial automation	The driving mode-specific execution by one or more driver assistance systems of *both steering and acceleration/deceleration*	
Automated driving system monitors the driving environment			
3	Conditional automation	The driving mode-specific performance by an automated driving system of all aspects of the dynamic driving task	With the expectation that the *human driver will respond appropriately to a request to intervene*
4	High automation		*Even if a human driver does not respond appropriately to a request to intervene*
5	Full automation		*Under all roadway and environmental conditions* that can be managed by a human driver

FIGURE 7.1 SAE (J3016) automation levels taxonomy.

Society of automotive engineers (SAE) has proposed different classes in AVs depending on their level of autonomy. They range from level 0, meaning no autonomy or completely manual, to level 5 which means complete and unrestricted vehicle autonomy. All six classes for autonomous autonomy are illustrated in the table below (Fig. 7.1).

Before studying the taxonomy of AVs it is necessary to go through some of the technical terms which will help to define the classes in taxonomy.

1. Driving task: driving task mainly consists of three subcategories (Nützel, 2018), namely:
   a. Perceiving the environment–This is mainly concerned with perceiving the environment in which we are driving, which includes tracking the vehicle motion and identifying the various elements around the vehicle like the road surface or the road signs and moving objects.
   b. Motion planning—motion planning task consists of how the vehicle is going to reach from point A to point B.
   c. Controlling the vehicle—After the path is defined and the waypoints are fixed then the vehicle needs to be maneuvered to follow that specific path to reach our final destination.
2. Operation design domain (ODD): It constitutes the operating conditions wherein which the vehicle is designed to perform it may include, environmental condition, road condition, weather conditions, or even time of the day. Thus for a self-driving car, the ODD needs to be defined perfectly as it is crucial for the safety of the system. ODD needs to be planned out carefully in advance based on the fact that what the environmental conditions and other parameters could be during the drive.

AVs are classified based on the amount of degree of automation and the degree of human action and human attention required. For the first three classes that is 0, 1, and 2 the human driver has to monitor the environment and then take suitable decisions. The control is with the human driver and the vehicle can just assist the driver. But for the next three classes that is for 3, 4, and 5 the vehicle takes care of the perception task for the human that is the AV monitors the driving environment. As seen from the above figure, there are six levels for an AV (Nützel, 2018).

1. Level 0: For level 0 there is no vehicle autonomy and the vehicle is completely controlled by a human. Vehicles falling into this category shows absolutely no intelligence. They are completely controlled by the human driver.
2. Level 1: For level 1, the vehicle can be programmed such that it shows either longitudinal control like acceleration and braking or, lateral control like steering. Thus the vehicle can be programmed for either the longitudinal or lateral control but not both. Still, the complete control of the vehicle is with the human. The Level 1 control is for just assisting the driver. This class is called as "Driver Assistance". For example, adaptive cruise control in which the system can keep up the speed but the driver has to steer, lane keeping assistance which can help to stay in the lane or warn in case of a drift.
3. Level 2: The second level automation is such that the AV can have more than one vehicle assistance. As against in level 1, in which the vehicle can only have either the lateral or the longitudinal control, in level 2 the AV can perform both the vehicle assistance, lateral and longitudinal control. So it can be said that the vehicle is driving itself partially based on environmental parameters. But still, the human needs to be attentive and need to give feedback to the vehicle. Thus the crucial decision-making and environment monitoring are still up to the human driver. This level is called as "Partial Automation". For example, GM super cruise, Nissan's pro pilot assist.
4. Level 3: level 3 automation is called as "Conditional Automation". In level 3 automation the system can perform object and event detection and response (OEDR) to a certain degree in addition to the level 2 control tasks. Even though the system can perform OEDR the control of the vehicle needs to be taken by the human driver in case of emergency or failure. The difference between level 2 and level 3 is that the attention of the driver is reduced and the vehicle can alert the driver in case of emergency or critical decision-making. But this creates controversy as it is not always possible for the vehicle to know that which is the critical situation. For example Audi A8 Sedan.
5. Level 4: In level 4 autonomy the system is such that it can execute a low-risk task in case the driver does not intervene during an emergency

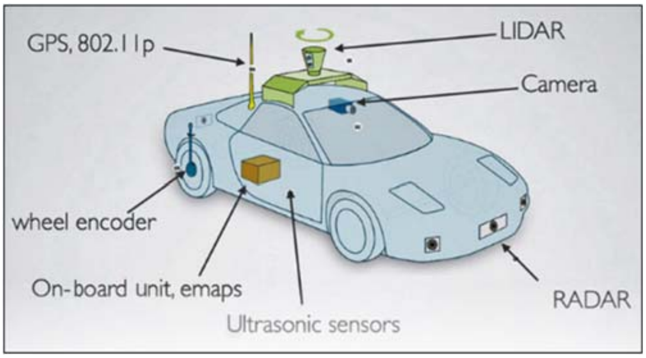


FIGURE 7.2 Components of autonomous vehicle.

situation, thus mitigating the damage. This type of control, thus, called as "High Automation". Level 4 vehicles can handle emergencies on their own but up to some extent and still requires driver intervention for some situations. This type of automation is such that it permits the driver to check the mobile phones or even watch movies. But still, the Level 4 has some lacunas, as it has a limited ODD. For example, WAYMO has achieved such a type of autonomy for limited ODD without the need for a human driver.

6. Level 5: level 5 autonomy called as "Full Automation" is a class in which the vehicle supports unlimited ODD and can operate in any conditions with full automation without human intervention. Such level 5 AVs are still not developed and require extremely high accuracy of hardware and software. Currently, every industry is jostling for this type of autonomy. And this is where cognitive computing can help to reach level 5 autonomy (Fig. 7.2).

Components of autonomous vehicle

Sensors

Even the best quality perception algorithms are restricted by their sensor data. A sensor is any device that measures a property of an environment or measures any change to that property over time. Sensors are mainly of two categories depending on what type information they record, they are as follows:

1. Exteroceptive: where "Extero" means surrounding or outside. Meaning they measure the surrounding property. For example, a camera, LIDAR, etc.
2. Proprioceptive: proprioceptive sensors are the ones that measure the property of own that is this case property of the vehicle. For example, IMU, Wheel Encoder, etc.

Different types of exteroceptive and proprioceptive sensors used in AVs are as follows:

1. Camera: the camera is the most common exteroceptive sensor used in an AV. They are passive light-collecting sensors that are good in collecting rich details of the environment. Camera selection is an important requirement for a good vision task. It is based on, camera resolution, field of view, dynamic range which is the difference between the light and the dark pixel in the Image, Focal length, etc. Stereo cameras are used for depth estimation. In stereo cameras, the images from the two cameras overlap and create a disparity map from which the depth of the object from the camera can be known.
2. LIDAR: light detection and Ranging is an important type of sensor as high accuracy vision task cannot be achieved using just the camera. In LIDAR the sensor shoots light beams in all directions and records the time taken by the light beam to return to the sensor after reflecting from the object. Thus by measuring the time we can know the distance traveled by the light as the velocity of light is known in almost all cases and varies slightly. Apparatus includes spinning element shooting light beams and gives the output in the form of a 3D point cloud. Unlike a camera, it is not affected by ambient light. LIDAR is selected based on the following factors, the number of beams in which the sensor shoots, points per second collection. The rotation rate is also an important parameter as higher the rate faster are the points updated, and also the field of view. Displayed below is a typical Image of point cloud generated using LIDAR.
3. RADAR: radio detection and ranging is the most common sensor and been there longer than the LIDAR sensor and is more robust. As they consist of radio waves they are not affected by weather conditions like precipitation, snowfall, etc. Thus making them more reliable in adverse weather. Radars are selected based on detection range, the field of view, position, and speed accuracy.
4. Ultrasonic sensors: ultrasonic sensors are also called as sound navigation and ranging (SONAR) which are low-cost sensors compared to others. They are mainly used for detecting closer objects, thus they are used predominantly during parking etc. Like RADAR they are unaffected by adverse weather conditions. SONAR sensors are selected by the parameters such as, range, the field of view, and cost.
5. Global navigation satellite system (GNSS) and inertial measurement unit (IMU): GNSS is a type of proprioceptive sensor which measures the position of the vehicle. Global positioning system is one of the GNSS which is widely used in AVs. They are used to measure the velocity and heading direction. The IMU is used to measure the angular velocity and the acceleration of the vehicle. By using these sensors together we can get a complete orientation of the vehicle.
6. Wheel encoder: Wheel encoder or wheel odometry sensor is used to track the heading direction rate and the rate of rotation of wheels (Fig. 7.3).

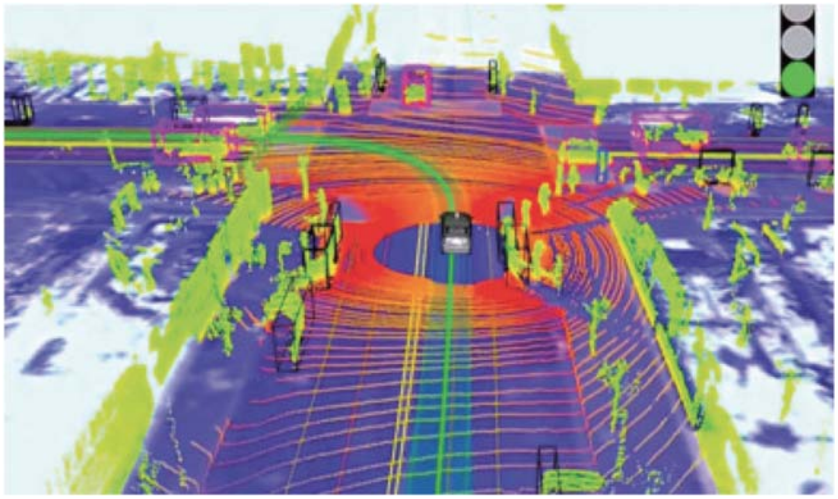


FIGURE 7.3 3D point cloud.

Sensor fusion: no sensor used individually provides enough information to achieve higher accuracy for the self-driving vehicle. Thus the fusion of sensor data and GPS is used that enables in determining the exact position and other objects around the self-driving vehicle with an accuracy of 10 cm. Sensor fusion can overcome certain drawbacks of the individual sensor. The most common example is the fusion of RADAR and camera. The camera can give greater accuracy when the weather is clean and clear. But during high fog or precipitation conditions where the visibility is hardly in few meters, the camera does not perform well due to unclear vision and obstructions. Thus in this case a RADAR can be used which enables the AV to see through the environment with enhanced visibility.

Actuators

Actuators of an AV are similar to what muscles are for the human body. They are responsible for moving and controlling the self-driving vehicle. Different actuation methods used for steering, braking, acceleration use an electric motor with gear reduction, or mechanical linkages or hydraulics to maneuver the vehicle.

Cognitive computing algorithms in the development of AVs

Types of learning algorithms

This part of the chapter will present general algorithms along with the cognitive algorithms used in AVs for different tasks. Cognitive computing is based on multi-core programming of CPUs, GPUs, TPUs, and Neuromorphic chips along with software development environment with intrinsic support for parallel computing and different software libraries. Before designing any algorithm it is necessary to consider certain events, a few of which are, landscape, different traffic rules in different countries, such as left-hand side and right-hand side driving, road signs, etc. Each of these parameters can affect the algorithm widely. However, as stated above the accuracy and development of any algorithm are restricted by the type of sensor used, such as camera, LIDAR, RADAR, etc. and the collected data. This data is then feed as input to the AI algorithms (Gudivada, Irfan, Fathi, & Rao, 2016).

Before diving deep into AI algorithms used in self-driving vehicles, it is necessary to know their classification. AI algorithms are classified as supervised, unsupervised, and reinforcement learning algorithms. In supervised learning, there is a set of associations between learning data and the labels, that is along with the data, the labels are fed to the AI model. The new data is repeatedly fed to the model with labels and based on that the weights are modified, to get a perfect model for prediction. For example, this process is similar to a child learning to identify new objects from the surrounding. The training dataset consists of two sets of information, first is the input data x, and expected output y. Thus (x, y) forms one example. Training dataset consists of n such pairs: $\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\ldots (x_n, y_n)\}$. Some of the examples of supervised learning algorithms are decision tree, NN, Bayesian classification, etc. (Gudivada et al., 2016). In unsupervised learning algorithms, the labels are not provided thus the model draws the inference from just the input data and groups the data into different clusters based on their characteristic pattern and hidden structures. K-means clustering, genetic algorithms are different unsupervised learning algorithms. Reinforcement learning is a type of learning which is inspired from day to day learning processes of a human being. Usually, in the unknown situation or when the outcomes are not known we the humans tend to take a certain action, and then based on the output of those actions, we decide whether to repeat that action or not. Thus similarly, in reinforcement learning algorithms the theoretical system agent takes some actions and based on those actions it gets some reward. For each correct action, it is awarded some positive reward. Thus the agent performs further actions such that the notion of reward is maximized in the environment. There are basically two types of reinforcement learning based on the fact that whether the agent is rewarded or penalized called positive reinforcement and negative reinforcement learning. For cognitive analysis, unsupervised learning algorithms have a certain edge over supervised algorithms in some scenarios such as, when we do not know the pattern in the data a priori. In those cases, unsupervised algorithms are used to generate the training data which then can be used to train supervised learning algorithms. Ensemble learning is a technique in which two or more algorithms are optimally combined to give better results than using individual algorithms (Gudivada et al., 2016).

Thus any machine learning algorithm can be classified into one of the above categories. These algorithms are used in AV to perform different tasks. Let us dive deep into the cognitive computing algorithms used AVs.

Cognitive computing algorithms used in autonomous vehicle

In this section different machine learning algorithms are discussed and compared. Before directly diving into different algorithms let us understand a general view of input, output, and the AI model.

The input consists of multiple rows of data which are called features. Each example represents a data point. The number of rows of data is guided by the number of a feature used for training the model. At times there may be a large number of features to train. It is not possible every time to train the model for all the features as the process is quite time consuming, it is necessary to select the most important features. Finding good features is an art necessary to increase the accuracy of the algorithm and to reduce the time consumption. The data is denoted by X which has n × m dimensions, given by $X = \{x_1, x_2, x_3 \ldots x_n\}$. Where x is a feature vector defined as, $x_n = \{x_{1n}, x_{2n}, x_{3n} \ldots x_{mn}\}$, where m is the number of features. For supervised algorithms, the input consists of labels given by Y which is an $n \times 1$ dimension vector, where Y is defined as $\{y_1, y_2, y_3 \ldots y_n\}$, where n is a positive integer equal to the number of training examples. The most challenging part of the machine learning algorithms is to select the model for mapping feature vector to target values. There are a lot of models available, thus selecting a proper model for the required task is critical. Parameters that define the model architecture are called "Hyperparameters" and the process to select an ideal structure of the model is called "Hyperparameter tuning". It is an important step which is used as a fine tuning technique to increase the accuracy of the model. Most of the times it is a trial and error process to select proper set of Hyperparameters. During training of the model, the weight is tweaked such that the prediction matches the expected outcome. These weights are given by $\theta = \{\theta_0, \theta_1, \theta_2 \ldots \theta_n\}$.

The autonomous driving pipeline is decomposed into four different modules which can use AI or deep learning approach or traditional approach independently (Grigorescu et al., 2020):

1. Perception and localization
2. High-level planning
3. Low-level planning
4. Decision-making (Fig. 7.4)

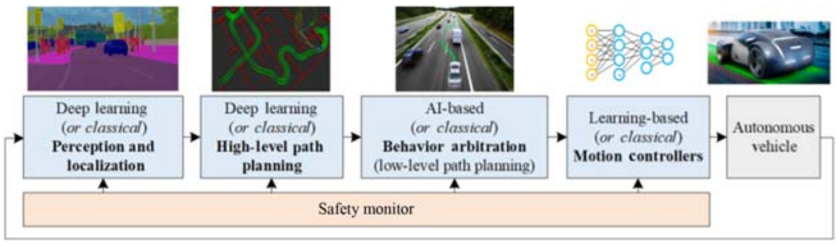


FIGURE 7.4 Architecture of deep learning based self-driving vehicle.

Let us discuss the tasks and the machine learning algorithms used in detail.

Machine learning

Neural networks

Artificial neural networks or neural nets are a type of supervised machine learning algorithm. It is biologically inspired by the neurons in the human brain. The main component of a NN is called a perceptron. It is predominantly used for the task of classification. Overall this algorithm is divided into different phases. The starting phase is known as the training phase. In this phase, the NN is fed different inputs and their corresponding labels. The learning phase itself is subdivided into two stages called feedforward and backpropagation. During the feed-forward step, the different weights are multiplied by their neuron value. Then passed through a perceptron which classifies the output into given classes depending on the activation function like the ReLU function, and then the prediction is calculated as output. After this feed-forward step the there is a backpropagation step. The cost function is shown below compares the true output value with the predicted value, given by the equation (Nützel, 2018; Gudivada et al., 2016) (Fig. 7.5). Where, $J(w)$ is the cost function which uses mean squared error to calculate the difference between the actual and predicted value. Here, n is the number of training examples. This cost function is optimized using an optimizer like Adams or gradient descent. The cost function has a bell shaped curve, thus consists of only one global minimum. In some cases, when the cost function is complex it can have multiple local minima. Thus it can happen that sometimes the optimizer can get stuck at a certain local minimum (Fig. 7.6).

The backpropagation updates the weights and biases of the NN. The Hyperparameter known as the learning rate is adjusted to control the weight change during one single iteration known as "Update Strength". Increasing the learning rate may cause higher oscillations in getting to the optimum point but reduces the time of training. Whereas, lowering the rate causes increased time consumption for the learning. Thus the value of the learning rate needs to be adjusted to suit our purpose by trial and error. The updated weights are given by the equation (Nützel, 2018),

$$\mathrm{w}_{(\mathrm{j,new})} = - \propto \frac{\partial \mathrm{J}}{\partial \mathrm{w}_{\mathrm{j}}} + \mathrm{w}_{\mathrm{j}} \tag{7.2}$$

Where, α is the learning rate and w_j is the current weight and $w_{(j,\text{new})}$ is the new weight after optimizing cost function j w.r.t initial weight in the previous layers. Thus repeating these steps for n number of examples the final weights are used for prediction purposes, which is the second stage of NN, that is testing phase. In this phase, the NN is tested on unseen data and the final results are analyzed.

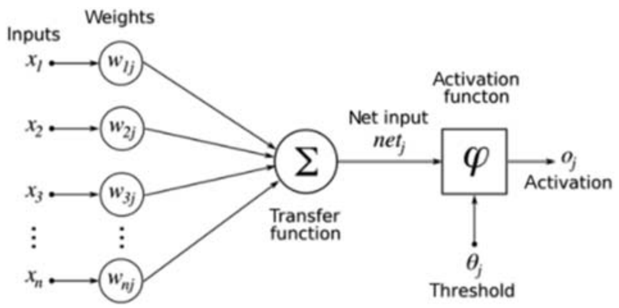


FIGURE 7.5 Diagram of artificial neural network.

$$J(w) = \left(\frac{1}{n}\right) \sum_{i}^{n} \left(\text{true}^{(i)} - \text{prediction}^{(i)}\right)^2 \tag{7.1}$$

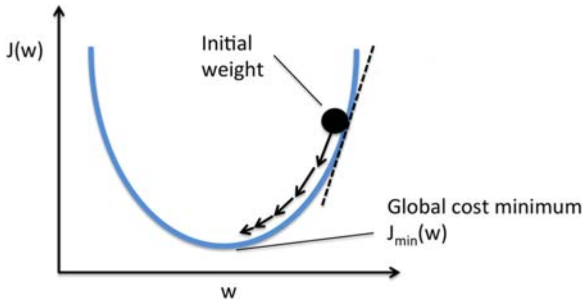


FIGURE 7.6 Cost function of a neural network having single global optimum.

Deep convolutional neural network

Traditionally, detection of objects was done in two phases, namely, feature detection which was a handcrafted approach followed by classification using ML algorithm such as SVM. But by the advent and development of NN, this traditional approach was replaced by a much accurate and automated machine approach called deep convolutional neural network (Deep CNN) (Grigorescu et al., 2020).

In the deep learning approach, the two steps of feature extraction and classification in the traditional methods are combined to form a single approach. CNN computes the feature map using the corresponding kernel by iterating the kernel over the complete image. Thus the output is the image with features whose shape and size depends on the kernel. A kernel is nothing but a square matrix that is convoluted over the complete image (Grigorescu et al., 2020) (Figs. 7.7 and 7.8).

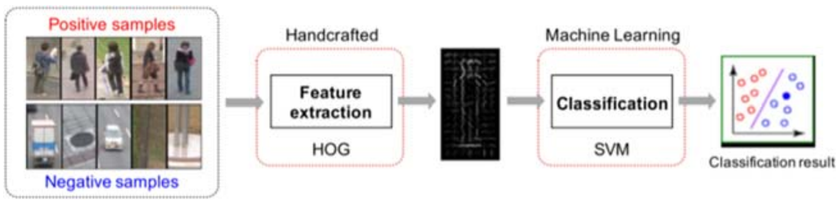


FIGURE 7.7 Traditional machine learning approach.

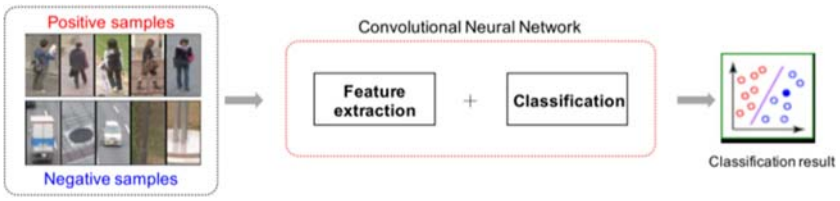


FIGURE 7.8 Modified approach using cognitive computing.

A basic example of a kernel is given by:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{7.3}$$

$$G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{7.4}$$

The above kernels shown are basic 3×3 kernel which detect the edges in x and y-direction. G_x is called a gradient component in x-direction and G_y is called the gradient component in the y-direction. As shown above G_x detects the vertical edge whereas, G_y detects edges in y-direction.

The result of using the above kernel on an image is given below (Fig. 7.9).

This feature extracted image is feed to the deep neural network for prediction purposes. The total CNN architecture looks like this (Fig. 7.10):

In the first layer, different kernels are applied which extract different features from the image, for example, curves, horizontal lines, vertical lines, circles, etc. Then these features are fed to the deep NN. During testing of the algorithm when the image contains those certain features, the specific neuron with that feature fires up. The detection accuracy of the deep CNN is much more than traditional handcrafted feature detection technique. Deep CNN, performs better than traditional techniques. Apart from this, deep CNN is flexible. It can be applied to various tasks just by changing the structure of the model. This is one of the advantages that cognitive computing provides over the traditional approach (Fujiyoshi, Hirakawa, Yamashita, 2019).

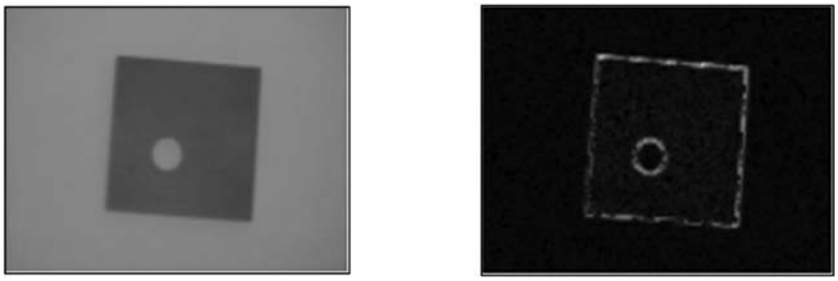


FIGURE 7.9 Before and after images of applying the kernel.

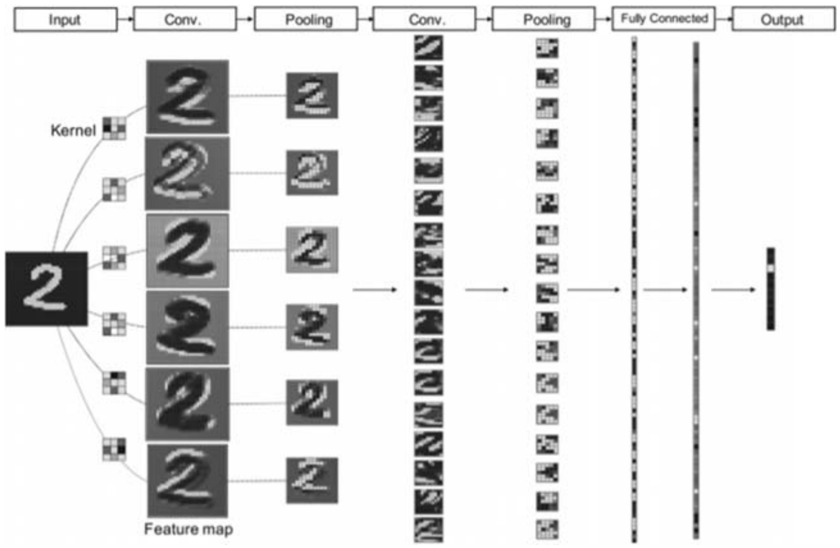


FIGURE 7.10 Architecture of a deep CNN model.

Recurrent neural network

Amongst all the types of NN, Recurrent neural networks (RNNs) are particularly good in the processing sequence of data, like music, audio, video, speech recognition, that is predicting the pattern in time series. In AV, they are mainly used to predict the motion of the object. Unlike other NN, RNNs have feedback in the hidden layer. Due to this, during the next iteration, the data from the previous layer is also included in the computation thus enabling the RNNs to consider data from not only that instance but also from the previous instances. Below is the diagram of a full RNN model (Grigorescu, Trasnea, Cocias, & Macesanu, 2020; Yao, 2018) (Figs. 7.11 and 7.12).

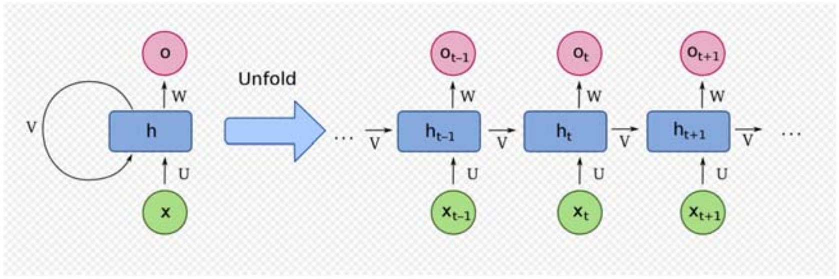


FIGURE 7.11 Folded and unfolded recurrent neural network.

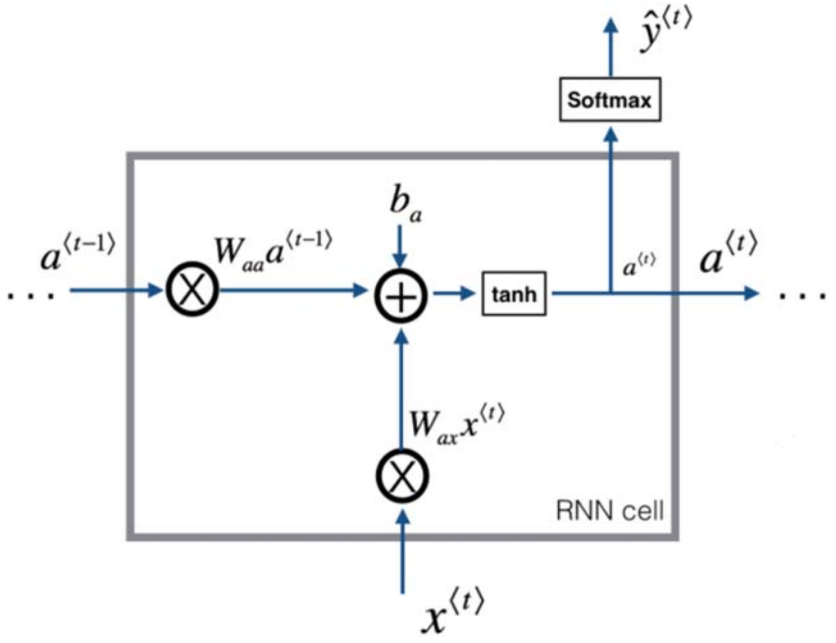


FIGURE 7.12 Recurrent neural network cell for a single time step.

In the above figure W_{ax} is the parameters that govern the connection from the x to the hidden layer. W_{aa} is the parameter that governs the horizontal connection. W_{ya} is the parameter that governs the output predictions. The value of W_{ax}, W_{aa}, and W_{ya} is the same in every time step. The value $a^{<0>}$ is considered as zero. The y and the a values for the current layer are given by the equation,

$$a{<}t{>} = g_1(W_{\mathrm{aa}}a^{<t-1>} + W_{\mathrm{ax}}x^{<t>} + b_a) \tag{7.5}$$

$$y{<}t{>} = g_2(W_{\mathrm{ya}}a^{<t>} + b_y) \tag{7.6}$$

Where g_1 and g_2 are the activation functions. The most commonly used activation function tanh for g_1 and sigmoid for g_2.

By using the above equations y and a can be found out in the RNN model. Even though RNN uses the data from the previous time step it is prone to some major well-known problem. The main challenge in using RNNs is its vanishing gradient problem. As the number of time steps increases the data from the previous time steps goes on reducing to the point that during the backpropagation the change in the first layer is almost negligible. This problem can be countered by using long short term memory networks (LSTM). LSTM networks are a nonlinear function approximator for estimating the dependencies in the data. As opposed to the traditional RNN, LSTM solves the vanishing gradient problem using three gates as shown below [https://jmyao17.github.io/Machine_Learning/Sequence/RNN-1.html, Oct 2020 (accessed 4.10.2020)] (Fig. 7.13).

Forget gate: When the subject or the pattern in the video changes it is necessary to delete or forget the previously stored memory, this is where the forget gate comes into the picture. In LSTM forget gate let us do this:

$$\Gamma_f^{<t>} = \sigma(W_f\left[a^{<t-1>}, x^{<t>}\right] + b_f) \tag{7.7}$$

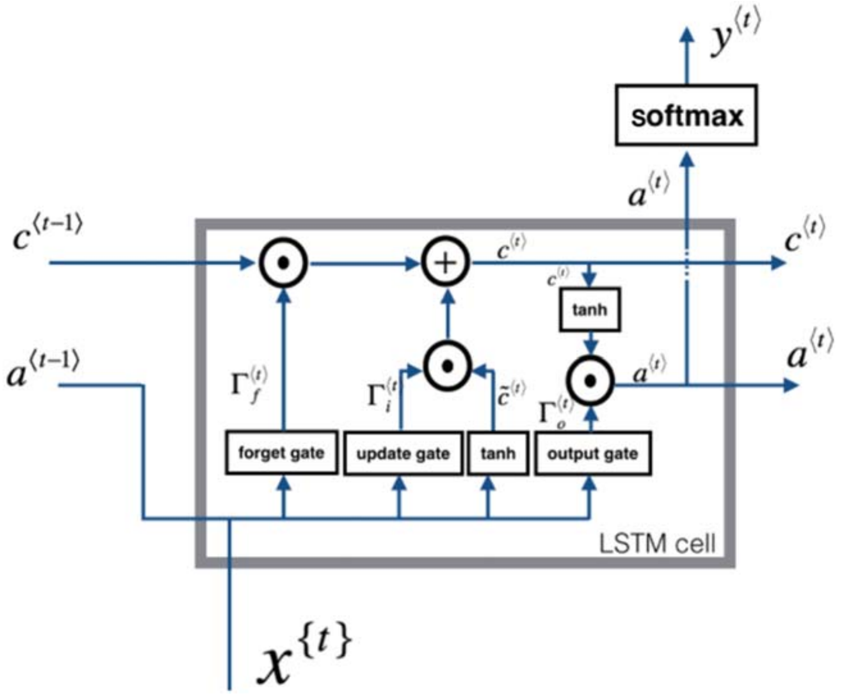


FIGURE 7.13 Long short term memory network cell.

Update gate: Called the input gate, as the name suggests, it updates the content which needs to be reflected in the new subject. To update the new subject it is necessary to create a new vector of numbers that can be added to the previous state. It is given by the equations as follows,

$$\Gamma_u^{<t>} = \tanh(W_u\left[a^{<t-1>}, x^{<t>}\right] + b_u) \tag{7.8}$$

$$\tilde{c}^{<t>} = \tanh(W_c\left[a^{<t-1>}, x^{<t>}\right] + b_c) \tag{7.9}$$

$$c^{<t>} = \Gamma_f^{<t>} * c^{<t-1>} + \Gamma_u^{<t>} * \tilde{c}^{<t>} \tag{7.10}$$

Output state: To decide which output to use the following two formulas can be applied in the model. The first equation decides what to output using a sigmoid function and the second equation multiplies that by the tanh of the previous state.

$$\Gamma_o^{<t>} = \sigma(W_o\left[a^{<t-1>}, x^{<t>}\right] + b_0) \tag{7.11}$$

$$a^{<t>} = \Gamma_o^{<t>} \times \tanh(c^{<t>}) \tag{7.12}$$

Another algorithm which is used along with LSTM is GRU. It achieves the same function as LSTM but uses only two gates instead of three.

Deep reinforcement learning

Deep reinforcement learning (DRL) have made a breakthrough in various decision-making tasks like defeating the professional human player in "DOTA", amazing victory in Atari games, deep mind victory in "Alpha Go". This section reviews the DRL in autonomous driving task, using partially observable markov decision process, which is an agent-based process. In the reinforcement learning algorithm, the agent which is the AV senses the environment with an observation $I^{<t>}$, based on the observation performs the action $a^{<t>}$ for the state $s^{<t>}$, based on the output for that state it receives reward $R^{<t+1>}$ and the state changes from $s^{<t>}$ to $s^{<t+1>}$ using a transition function $T^{s^{<t+1>}}$ and the same process is executed till the system reaches the destination state $s_{\text{dest}}^{<t+k>}$. The agent aims to maximize the reward. Thus the agent learns the optimal driving policy in order to maximize the reward. The above logic can be expressed in model form as (Grigorescu, Trasnea, Cocias, & Macesanu, 2020; Bhatt, 2018),

$$M = (I, S, A, T, R, \Upsilon)$$

Where, I is the set of observation of the environment at any instance of time t denoted by, $I^{<t>}$.

S is the current state of the agent at time t denoted by, $s^{<t>}$

An is the action taken from the observation $I^{<t>}$ for state $s^{<t>}$ denoted by, $a^{<t>}$

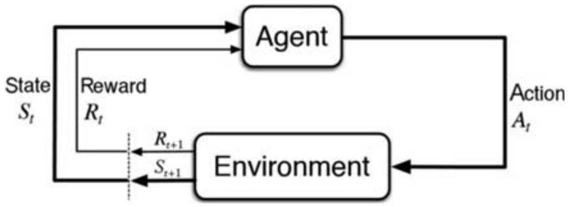


FIGURE 7.14 Block diagram of deep reinforcement learning.

R is the reward which the agent gets based on the action taken denoted by, $R^{<t+1>}$

Υ is the discount factor that takes into consideration the importance of future events to the current events (Fig. 7.14).

The agent selection of action is modeled as per the map called policy denoted by π.

$$\pi = A^{*}S \rightarrow [0, 1] \tag{7.13}$$

$$\pi(a, s) = P(a_t = a, s_t = s) \tag{7.14}$$

Which is state and action at a given time instance t. Thus the policy-map gives the probability of taking an action a in the state s. A state value function denoted by $V_\pi(s)$ follows a policy π starting from the initial state S_0. It measures how good it is to be in a state s following policy π. Thus $V_\pi(s)$ is given by,

$$V_\pi(s) = E\left[\sum_{t=0}^{\infty} \gamma^t \times r_t\right] \tag{7.15}$$

Where, $V_\pi(s)$ is the summation of all the rewards in the future multiplied by the discount factor which decides how good a state is. Thus the algorithm finds the optimum policy with maximum expected returns. The optimum action Value function is denoted by $Q \times (.,.)$ satisfies the Bellman optimality equation, which is a recursive formula given by,

$$Q^{\times}(s, a) = \sum_{s} T_{s,a}^{s'}\left(R_{s,a}^{s'} + \gamma \max Q^{*}(s', a')\right) \tag{7.16}$$

In autonomous driving applications, the observation space is mainly composed of sensor information which is made up of data from LIDAR, SONAR, camera, etc. Thus instead of the traditional approach, a non-linear Q* can be encoded in the deep learning model. Such a model is called DRL. In DRL the deep learning technique is combined with reinforcement learning to give ground-breaking results and solve the problems like never before. Although continuous control is possible with DRL, a discrete approach is used. The main problem is the training, as the agent has to learn through exploring the environment, usually through the collisions. The solution here is to use an imitation learning method called inverse reinforcement learning to learn from human driving demonstrations without the need to go to unsafe methods.

DRL suffers from a major problem known as the credit assignment problem. As the in a single episode even though all the actions are correct but due to the last few actions the agent fails, the complete reward for that episode reduces thus deducing the agent that all the actions in that single episode must be bad actions intern reducing the probability of taking those actions in the future. The solution to this is the agent should able to recognize what part of the episode is causing the less reward and eliminate that part instead of eliminating the whole episode.

End-to-end learning method

In most research in CNN and cognitive computing techniques in autonomous driving, the environment around the vehicle is recognized by using the camera or the LIDAR sensor, appropriate motion planning, and control the throttle positions. Thus they mainly use processes such as recognition of the object, segmentation, etc. But due to the progress in CNN, it is possible to directly control the valves directly from the Input Image. In this method, the network is trained using a dashboard camera or a similar method like a simulator to train the CNN model to control the vehicle under a real environment. End-to-end learning has one advantage as it does not require to completely understand its surrounding environment. The inputs of the camera are given to the CNN model and output steering angle directory and throttle angles i.e. using a driving simulator to train the CNN model and use the trained model to control the throttle and steering angles (Grigorescu et al. 2020; Fujiyoshi et al. 2019).

The structure comprises of five layers of convolutional layers along with the pooling layers and three layers of fully connected neurons. Shown below is the simple pipeline, combined pipeline with cognitive computing and combined with end-to-end approach (Figs. 7.15–7.17).

As can be seen from the above image that the end-to-end system replaces almost all the tasks in autonomous driving by the CNN model. Training of the end-to-end system is given in the following figure (Fig. 7.18).

Even with these techniques end-to-end has some drawbacks. One of the challenges in end-to-end learning is it requires a lot of driving data for training before going into the actual environment. For smaller datasets usually, the other methods are employed instead of end-to-end learning. As for smaller datasets, the accuracy of end-to-end drops drastically and thus are not

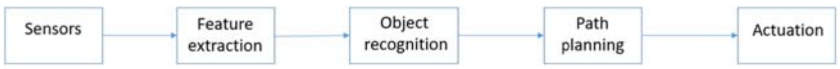


FIGURE 7.15 Simple pipeline.

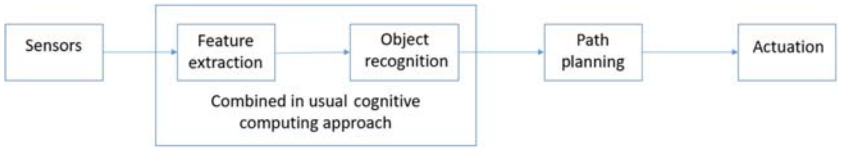


FIGURE 7.16 Combined with cognitive computing.

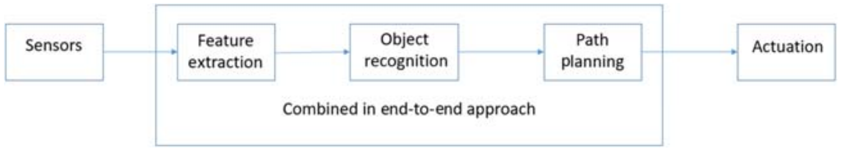


FIGURE 7.17 End-to-end learning.

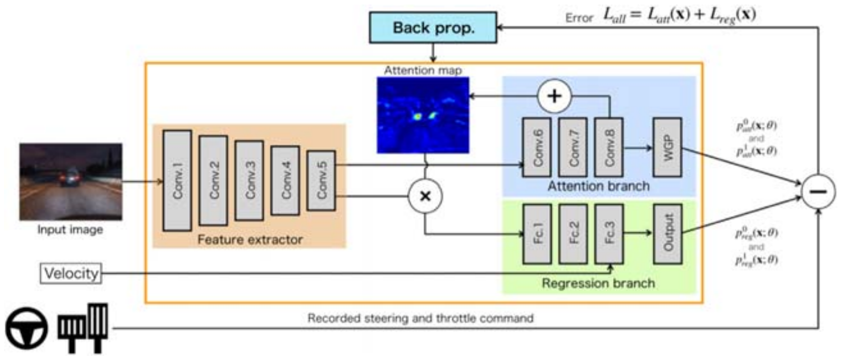

Back prop.
Error $L_{all} = L_{att}(\mathbf{x}) + L_{reg}(\mathbf{x})$
Attention map
Input image
Conv.1
Conv.2
Conv.3
Conv.4
Conv.5
Feature extractor
Conv.6
Conv.7
Conv.8
WGP
Attention branch
Fc.1
Fc.2
Fc.3
Output
Regression branch
Velocity
$p^0_{att}(\mathbf{x};\theta)$ and $p^1_{att}(\mathbf{x};\theta)$
$p^0_{reg}(\mathbf{x};\theta)$ and $p^1_{reg}(\mathbf{x};\theta)$
Recorded steering and throttle command

FIGURE 7.18 Regression type attention branch network (end-to-end).

preferred for smaller or medium datasets. Another disadvantage for end-to-end is that it eliminates the handcrafted ML models. When data is small the model does not get insight from only the data and hand designing a component in the model would be a way to inject manual knowledge. As a learning algorithm has two main sources of knowledge i.e. data and second is hand-design feature or components. Thus when we have large data there is less need to hand-design but when the data is small the only way to inject knowledge is by carefully hand designing the components (Fujiyoshi et al. 2019).

Deep driving

Perception and localization

CNN in object detection

In conventional machine learning approach uses a raster scan two-class classifier for detecting the object, which is based on the principle that the aspect

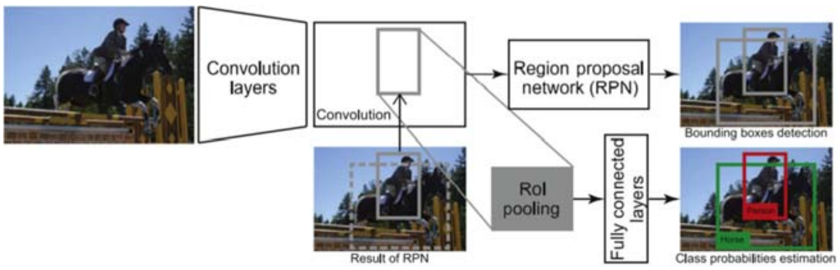


FIGURE 7.19 R-CNN structure.

ratio of the object which is to be detected does not changes, it will detect only certain categories learned as a positive sample. In some cases when there is no object in the window the algorithm searches for an object to classification. On the contrary, in the object through CNN objects with different aspects are detected, thus multiple class detection of object is possible by using region proposal network (RPN). A much faster class of CNN which is R-CNN simultaneously detects objects and detects the object class of those objects. In RPN an object is detected using raster scanning of K different windows of different sizes in the focused area known as an anchor. The region of the anchor is input to the RPN and the score of object likeliness is output. The region of the anchor is input to the connected network for object recognition when it is determined to be an object by RPN. Thus the RPN method has made it possible to recognize multiple objects in the image. Below is the block diagram for R-CNN architecture (Fujiyoshi et al.) (Fig. 7.19).

Another method was developed known as you only look once (YOLO) is a technique that detects multiple objects by giving the whole input image to Convolutional Neural Network without raster scanning. In this, the image is divided locally in 7×7 grid. Firstly a feature map is generated through CNN and then the feature map directly feed to the fully connected layers. The output obtained displays the score of the object category at each smaller grid of the two object rectangles. YOLO has an advantage as the object detection can be done in real-time. Below shows the process of object detection using YOLO (Fig. 7.20).

CNN in semantic segmentation

Semantic segmentation is defined as linking each pixel in an image to the class label. Semantic segmentation is considered a difficult task in the computer vision. But it is seen that the NN has shown much more promising results like increased accuracy, higher performance compared to traditional machine learning algorithms. A fully convolutional network (FCN) is a technique that enables end-to-end learning and can obtain segmentation using only CNN.

A typical structure of FCN is given below (Fujiyoshi et al.) (Fig. 7.21).

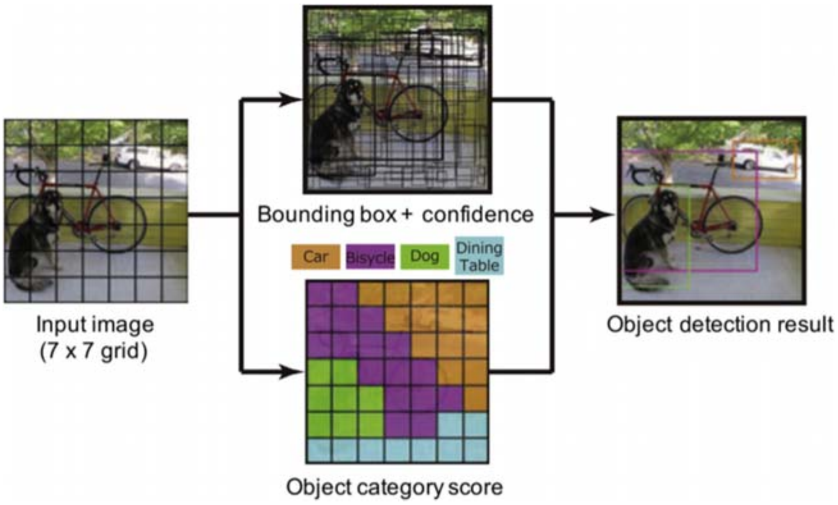


FIGURE 7.20 You only look once structure.

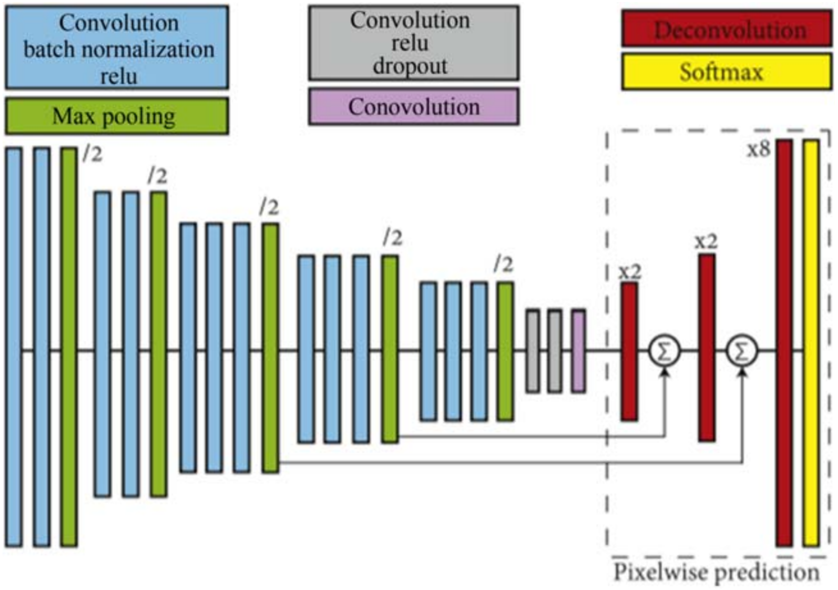


FIGURE 7.21 Fully connected network structure.

Localization

Localization algorithms are used to locate the position of the AVs as it navigates. The traditional approach to solving the localization problem is by using GPS. But GPS is not quite accurate, giving an accuracy of 2−10 m that is we cannot rely on GPS for a lane change or such tasks that require higher accuracy. Visualize localization known as visual odometry (VO) uses deep learning to localize the AV giving and accuracy of 2–10 cm. This is done by matching key-points landmarks in consecutive video frames. The key-points are input to the n-point mapping algorithm which detects the pose of the vehicle. The structure of the environment can be mapped with the computation of the camera pose. These methods are called simultaneous localization and mapping (SLAM). NN such as PosNet, VLocNet++ uses the image data to estimate the pose 3D pose of the camera. (Bugala, 2018; Grigorescu et al., 2020).

For the driving of AVs, it is necessary to correctly estimate the motion and trajectory of the surrounding objects. This process is called as scene flow. Traditionally used methods of scene flow needs to design feature manually. This is replaced by current DL approach which automatically learns the scene flow. Even though there is a lot of improvement in DL the traditional VO algorithms are not completely supplanted by the DLs. This is because the key-point detector technique is computationally efficient and can be deployed on embedded devices without many efforts.

Route planning algorithm

One of the important and the first decision was taken by the AV is to select a route from the current location to the destination based on criteria of distance, time consumed, or fuel consumption. Heuristic algorithms are used to find the distance between two locations via the cost of intermediate nodes, they are as follows, Bellman-ford is used when non-negative edges are only used, Dijkstra's algorithms are used for a known topology, and A* algorithm which is the most common algorithm for path selection. Traditional methods used for the route planning purpose are as follows (Bugala, 2018):

The non-linear approach tracks the trajectory of the vehicle and maintains the steering such that it follows the desired trajectory.

Model predictive control (MPC) consists algorithm for precise prediction of vehicle trajectory using traffic optimization and prediction techniques.

Feedback-feed-forward control is a technique that estimates the steering angle which is required to follow the curvature in a feed-forward loop such as to minimize the feed-forward error signal.

Shared control or parallel autonomy provides an additional level of safety by taking over the task of the driver in dangerous conditions which is referred to as interleaved autonomy.

Modern algorithms are based on a machine learning model called the end-to-end learning models. These models replace the traditional models for Route Planning. These end-to-end methods apply to single components of the system (as distinguished from the end-to-end autonomy described further on), for example to keep the vehicle within a lane. Schwarting et al. (2018).

End-to-end planning is a holistic approach, that is without any breakdown into components responsible for some part of the system. There are CNN networks that learn to carry out perception and generation of routes completely based on driving sequence and LIDAR data.

Decision-making

Driving decision-making (DDM) algorithms are an important part of autonomous driving. They are responsible for making strategic decisions during driving that is whether to change the lane, to overtake, or even decision based on road signs, based on various data from the input sensors. There are a lot of decision-making algorithms available that are applied based on the capability or necessity of the algorithm. Even though it is possible to present some algorithms, which are stated below (Bugala, 2018):

1. Partially observable markov decision process
2. Support vector machine with particle swarm optimization
3. Markov decision process with reinforcement learning
4. Deep reinforcement learning

Conclusion, discussion, and further research

AVs as discussed combine different techniques and algorithms using sensor data to solve problems in self-driving such as perception, localization, planning, etc. For this cognitive computing, techniques have proved to have higher accuracy and show higher performance compared to traditional approach machine learning approach as discussed (Fig. 7.22).

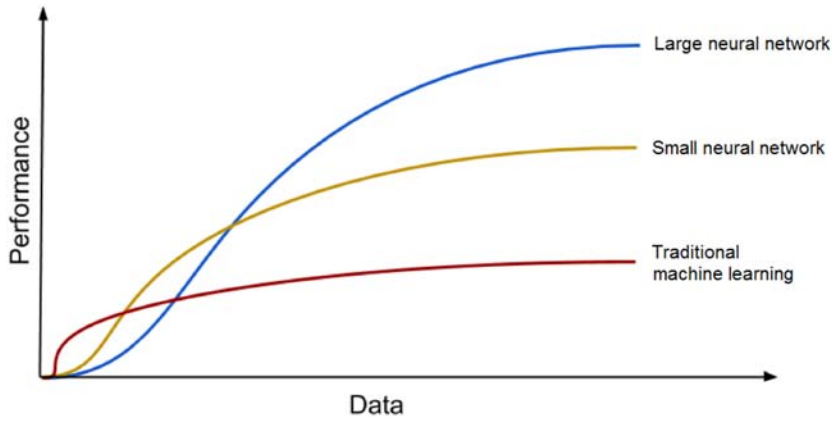


FIGURE 7.22 Comparison of different traditional machine learning algorithms and cognitive computing algorithms.

Traditional computing techniques even though it requires a small amount of data, generally shows a plateau in the performance even though a large dataset is available and is fed to the model. This is not the case with cognitive computing algorithms. For a smaller dataset, traditional algorithms prove much more useful compared to deep learning models. This is due to the fact that cognitive models require a large amount of data to function accurately. As seen from the graph, it is evident that as the data size increases the accuracy and performance of the model also increases. This useful characteristic of cognitive algorithms is used widely in AVs. For self-driving vehicles, the data collected through the different sensors is in a large amount. This data consists of data from actual driving of a human entity and also from simulators such as CARLA. In such cases, using traditional machine learning models would reduce the maximum achievable accuracy thus reducing the performance of the vehicle. As against this, if deep learning models are used, due to a higher available of data the maximum achievable accuracy can be increased to a greater extent, far beyond the conventional models.

Based on the fact in machine learning, which is "Garbage in, Garbage out," states that even though the data is available in larger quantities, it needs to be accurate and bias free, or at least the bias needs to be minimal if possible. Thus, the selection of data is a major criterion for AVs which affects the model accuracy greatly.

To summarize, cognitive computing techniques provide a greater advantage over conventional algorithms in AVs. But it is important to understand when to utilize traditional machine learning algorithms and when to opt for cognitive computing algorithms. Though deep learning provides greater advantage over other machine learning algorithms in some situations, replacing all the algorithms with cognitive computing algorithms does not seem to be a viable option. Definitely replacing most of them will prove to be a great deal of improvement. Even though using these techniques and by current technology, it is not possible to achieve full autonomy. Current developments in deep learning models like "Border Paris Method" provides an improvement over backpropagation task in deep learning by eliminating irrelevant learning patterns. Along with this, it has no problem in finding global maxima. It is called bi-propagation, quite similar to backpropagation but with the difference that is, the learning takes place separately for each layer. Thus, the hidden layers are no longer hidden. Nevertheless, technological progress will enable us to continuously change the upgrade, the choice of algorithms, and replace the traditional ones with novel methods thus reaching level 5 autonomy in AVs.

References

Bugala, M., (2018). Algorithm applied in autonomous vehicle system.

Fujiyoshi, H., Hirakawa, T., & Yamashita, T. (2019). Deep learning-based image recognition for autonomous driving. *International Association of Traffic and Safety Sciences Research*, *43* (4), 244–252.

Grigorescu, S., Trasnea, B., Cocias, T., & Macesanu, G. (2020). A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, *37*(3), 362–386. Available from https://doi.org/10.1002/rob.21918.

Gudivada, V. N., Irfan, M. T., Fathi, E., & Rao, D. L. (2016). Cognitive analytics: Going beyond big data analysis and machine learning. *Handbook of Statistics*, *35*, 169–205, https://doi.org/10.1016/bs.host.2016.07.010.

Nützel, T., (2018). *AI-based movement planning for autonomous and teleoperated vehicles including the development of a simulation environment and an intelligent agent*. Technical University of Munich.

Schwarting, W., Alonso-Mora, J., & Rus, D. (2018). Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, *1*, 187–210. Available from https://doi.org/10.1146/annurev-control-060117-105157.

Yao, J., Research Associate Michigan State University (2018), Available at https://jmyao17.github.io/Machine_Learning/Sequence/RNN-1.html, (accessed 4.10.2020).

Bhatt, S. (2018) Reinforcement Learning, Available at https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html, (accessed 4.10.2020).

Further reading

Available from https://towardsdatascience.com/artificial-intelligence-vs-machine-learning-vs-deep-learning-2210ba8cc4ac, Nov 2020 (accessed 20.11.2020).

Available from https://dzone.com/articles/comparison-between-deep-learning-vs-machine-learni, Nov 2020 (accessed 20.11.2020).

Available from https://www.sciencedirect.com/science/article/pii/S2095809916309432, (accessed 5.10.2020).

Available from https://www.embedded.com/the-role-of-artificial-intelligence-in-autonomous-vehicles/, (accessed 10.11.2020).

Available from https://www.robsonforensic.com/articles/autonomous-vehicles-sensors-expert/, (accessed 10.11.2020).

Bussemaker, K.J., (2014). *Sensing requirement for an automated vehicle for highway and rural environment*, Delft University of Technology.

Chen, S., Jian, Z., Huang, Y., Chen, Y., Zhou, Z., & Zheng, N. (2019). Autonomous driving: cognitive construction and situation understanding. *Science China Information Sciences*, *62*, 81101. Available from https://doi.org/10.1007/s11432-018-9850-9.

Jeffery Roderick Norman Forbes. (2002). Rei*nforcement learning for autonomous vehi*cles (p. 110p) Berkley: University of California.

Kuutti, S., Bowden, R., Jin, Y., Barber, P., & Fallah, S. (2021). A survey of deep learning applications to autonomous vehicle control. *IEEE Transactions on Intelligent Transportation Systems*, *22*(2), 712–733. Available from https://doi.org/10.1109/TITS.2019.2962338.

Ni, J., Chen, Y., Chen, Y., Zhu, J., Ali, D., & Cao, W. (2020). A survey on theories and applications for self-driving cars based on deep learning methods. *Applied Sciences*, *10*(8), 2749. Available from https://doi.org/10.3390/app10082749.

O' Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Velasco Hernandez, G., Krpalkova, L., Riordan, D., & Walsh, J. (2019). *Deep learning vs. traditional computer vision*. Tralee: IMaR Technology Gateway, Institute of Technology Tralee.

Pendleton, S. D., Andersen, H., Du, X., Shen, X., Meghjani, M., Eng, Y. H., Rus, D., & Ang, M. H. (2017). Perception, planning, control, and coordination for autonomous vehicles. *Machines*, *5*(1), 6. Available from https://doi.org/10.3390/machines5010006.

Pu, Y., Apel, D. B., Liu, V., & Mitri, H. (2019). Machine learning methods for rockburst prediction-state-of-the-art review. *International Journal of Mining Science and Technology*, *29*(4), 565−570. Available from https://doi.org/10.1016/j.ijmst.2019.06.009.

Rao, Q., & Frtunikj, J. (2018). Deep learning for self-driving cars: chances and challenges, in: 2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS), pp. 35-38.

Simhambhatla, R., Okiah, K., Kuchkula, S., & Slater, R. (2019). Self-driving cars: evaluation of deep learning techniques for object detection in different driving conditions. *SMU Data Science Review*, *2*(1), Article 23. Available from. Available from https://scholar.smu.edu/datasciencereview/vol2/iss1/23.

COGNITIVE DATA SCIENCE IN **SUSTAINABLE COMPUTING**

SERIES EDITOR: **ARUN KUMAR SANGAIAH**

COGNITIVE COMPUTING FOR HUMAN-ROBOT INTERACTION

PRINCIPLES AND PRACTICES

VOLUME EDITORS
MAMTA MITTAL, RAJIV RATN SHAH AND SUDIPTA ROY

Cognitive Computing for Human-Robot Interaction: Principles and Practices explores the efforts that should ultimately enable society to take advantage of the often-heralded potential of robots to provide economical and sustainable computing applications.

This book discusses each of these applications, presents working implementations, and combines coherent and original deliberative architecture for human–robot interactions (HRI). Supported by experimental results, it shows how explicit knowledge management promises to be instrumental in building richer and more natural HRI, by pushing for pervasive, human-level semantics within the robot's deliberative system for sustainable computing applications.

This book will be of special interest to academics, postgraduate students, and researchers working in the area of artificial intelligence and machine learning.

KEY FEATURES

- Introduces several new contributions to the representation and management of humans in autonomous robotic systems;
- Explores the potential of cognitive computing, robots, and HRI to generate a deeper understanding and to provide a better contribution from robots to society;
- Engages with the potential repercussions of cognitive computing and HRI in the real world.

ISBN 978-0-323-85769-7

ELSEVIER

ACADEMIC PRESS
An imprint of Elsevier
elsevier.com/books-and-journals