Autonomous Driving Systems

Subjects: Transportation

Contributor: Carlos Francisco Moreno-Garcia

Autonomous vehicles are increasingly becoming a necesAutonomous vehicles are increasingly becoming a necessary trend towards building the smart cities of the future. Numerous proposals have been presented in recent years to tackle particular aspects of the working pipeline towards creating a functional end-to-end system, such as object detection, tracking, path planning, sentiment or intent detection, amongst others. Nevertheless, few efforts have been made to systematically compile all of these systems into a single proposal that also considers the real challenges these systems will have on the road, such as real-time computation, hardware capabilities, etc.

Keywords: autonomous vehicle ; autonomous driving system

1. Introduction

The National Highway Traffic Safety Administration (NHTSA) reported that 94% of severe crashes on the road are caused by human error [1]. In this regard, the rise of the autonomous vehicle (AV) has a huge potential to decrease these accidents and make the road much safer. Therefore, the implementation of robust and secure systems is paramount for the proper design of an autonomous driving system (ADS) pipeline. This field has been widely investigated for many years. Both academia and industry investigations have achieved breakthroughs and state-of-the-art results in the last years. In 2004, the Defense Advanced Research Projects Agency (DARPA) held the Grand Challenge competition for American AVs (DARPA, *The Grand Challenge*, DARPA, https://www.darpa.mil/about-us/timeline/-grand-challenge-for-autonomous-vehicles); however, no team could complete the designated 150 miles route. One year later, at the same event, all but one of the finalists surpassed the best record (7.32 miles) from the previous year, and five teams were able to finish the 132 miles desert terrain route without human intervention. In 2007, during the third edition of this challenge (DARPA, *DARPA Urban Challenge*, DARPA, https://www.darpa.mil/about-us/timeline/darpa-urban-challenge), 60 miles of the urban course were proposed. Only six teams managed to complete the entire track successfully. These three competitions were very significant events up to this time, so the results have inspired universities and large corporations to improve the state of the art of in ADSs in different environments and conditions.

Apart from the DARPA challenges, many other remarkable competitions have been held until today. For instance, Hyundai Motor Company has been researching AVs for a long time. They aim to expand future vehicle research at universities and improve Korea's automotive industry's technological development, and thus they have organized four AVs competitions (Hyundai NGV Industry-Academy-Research Cooperation, *Autonomous Vehicle Competition*, Hyundai NGV, http://www.hyundai-ngv.com/en/tech_coop/sub05.do) since 2010. Similarly, the Society of Autonomous Engineers (SAE) and General Motors have partnered to sponsor the AutoDrive Challenge (SAE International and GM, *AutoDrive Challenge*, SAE, https://www.sae.org/attend/student-events/autodrive-challenge/), which consisted of a three-year (2018–2020) competition for students to complete an urban driving course. Furthermore, Indy Autonomous Challenge (Indianapolis Motor Speedway, *Indy Autonomous Challenge*, Indianapolis Motor Speedway, https://www.indyautonomouschallenge.com/) is a new high-speed autonomous race held at the Indianapolis Motor Speedway. Every team must use the specific Dallara-produced IL-15 that has been retrofitted with hardware and controls to compete. It has been noted that numerous universities and companies worldwide are very active in these kinds of events, which indubitably will accelerate research on this field.

From the academic perspective, several survey papers in this area have taken different approaches to review this field. Rangesh et al. [2] compared several representative online Multi-Object Tracking (MOT) for AVs. Later on, the authors proposed M^3OT (i.e., a 3D variation of MOT), an approach suitable for tracking objects in the real world. It is worth mentioning that M^3OT can work with any type and number of cameras as long as they are calibrated. Huang et al. [3] spanned the state-of-the-art technology in significant fields of self-driving systems, such as perception, mapping and localization, prediction, planning and control, simulation, etc. Yurtsever et al. [4] attempt to provide a structured and

comprehensive overview of state-of-the-art automated driving-related hardware-software practices, high-level systems architectures, present challenges, datasets, and tools to ADS.

Badue et al. [5] published survey research on self-driving cars based on the aforementioned DARPA challenges. The authors presented a detailed description of the self-driving vehicle developed at the Universidade Federal do Espírito Santo in Brazil, named Intelligent Autonomous Robotics Automobile (IARA). This paper presents a complete overview of each of the aspects that compose the matured used architecture of self-driving cars nowadays, dividing it into two systems, i.e., perception and decision-making. At the same time, each design is composed of a variety of subsystems. Achieving its objective by describing each of the technologies superficially, this work represents a notable introduction to the interested reader. However, for a lettered person in the subject, this work's scope might fall short because it does not present the state-of-the-art technologies for driverless cars' different paradigms.

Furthermore, the study of these techniques has been extended to other application domains beyond AVs on the highway. For instance, Szymak et al. [6] presented a comparative study where different deep learning architectures were tested to perform object classification in underwater AVs' video image processing. In this study, the authors evaluated different architectures' performance to classify objects (i.e., fish, other vehicles, divers and obstacles) and detect abandoned munitions' corrosion. Interestingly, they were able to deduct that pre-trained algorithms have higher probabilities of success than tailor-made approaches. This rationale will be explained and supported throughout our work.

2. Methods to Address the Pipeline of End-To-End ADS

This section will discuss the latest literature methods to address the different stages that constitute an end-to-end ADS. This section is divided into four main categories. The first one presents image pre-processing and instance segmentation, in which the main aim is to locate all legal boundaries and other objects of reference. It is inferred that although one vehicle is physically capable of driving on a sidewalk, it is not correct for it to do so, and therefore the ADS needs to recognize the road lanes. Nonetheless, not all road lanes are used for driving in the required direction (i.e., the single direction only, cycling/pedestrian lanes). Thus, this step plays a crucial role in detecting which road lanes shall the vehicle occupy. The second one focuses on object detection within the semantic segmenting boundaries of the route. For instance, algorithms such as YOLO are used to recognize each object, and this data is fused with LiDAR point clouds using an MV3D-like architecture [7]. Processes such as these help the vehicle draw prism points on each detected object and know where its center of mass relies upon and the volume being handled within the lane space. Human and sentiment detection techniques (for driver, passengers, and pedestrians) are discussed in the third section. Finally, in the fourth category, cloud computing and parallelization are discussed to speed up and improve the pipeline. For instance, these techniques can train all object detection models, communication of different vehicles, etc.

2.1. Image Pre-Processing and Instance Segmentation

Chen et al. [8] proposed a CNN model for object localization using camera and LiDAR data for 3D multi-view object detection, or MV3D. In this method, three CNNs are deployed in parallel. Two of them are fed with the point cloud LiDAR data (one obtained from a BEV and the other from the front view). The third CNN uses raw RGB images from a camera located at the front, on the sides, and the car's back for a 360-degree view. Once each architecture finishes, a deconvolution process occurs to identify where the tracks are located and their shapes. Like an RCNN architecture, a decoder-like function outputs decoding fed to the architecture performing the localization task and shared between the other parallel architectures. This process generates a 3D object proposal of the classes, combining a region-wise feature obtained via ROI pooling caused by each architecture. The output is a 3D bounding box around the recognized objects. It is important to note that this method can realize the item's class, position, and volume. This method performs with a 14.9 % higher accuracy than state-of-the-art methods, comparable in runtime. As a result, this method is computationally demanding, and deducting the 3D bounding box will require a specific and robust hardware architecture for the algorithms to be deployed correctly and in real-time scenarios.

Zhou et al. [9] proposed VoxelNet, an end-to-end trainable deep architecture for point cloud-based 3D detection. In contrast to the previous two methods, this is a LiDAR-only based detection, which relies on the object size, occlusion state, and other superficial information. To improve the localization and detection, the authors proposed the Multimodal Voxel for 3D Object Detection Network (MVX-Net) framework. In this method, fused augmented LiDAR point clouds with semantic image features are used at early stages to improve 3D object detection. Moreover, two fusion techniques were developed to extend the VoxelNet. The first one is PointFusion, which combines features at early stages to add 3D points to an image feature and capture dense context, extracting features from a 2D detector (i.e., a camera) and then jointly inputting them into the VoxelNet model. The second technique is called VoxelFusion and employs non-empty 3D voxels to

be projected on the image plane. Features are extracted and fed into the voxel feature encoder used by a 3D RPN to deduce the bounding boxes. Authors state that the VoxelFusion technique delivers a slightly inferior performance compared to the PointFusion one; however, it has more efficiency in memory consumption. The authors evaluated the KITTI 3D object detection benchmark model and concluded that their approach demonstrated better results than the single modality VoxelNet [10].

Avoiding person-to-person contact is the best way to control and prevent the virus' transmission amidst the COVID-19 epidemic that affected the world in 2020. Still, people require food and other basic goods to address their essential needs. Liu et al. [11] developed Hercules, an autonomous logistic vehicle used for contactless delivery, to address this issue. It has an autonomous navigation capability and a stable hardware-software system to manage the operations. Its maximum payload and capacity are 1000 kg and 3 m^3, sufficient for most of the people delivery's requirements, and uses 3D object detection with LiDAR point clouds to recognize and classify objects. Multiple calibrated LiDARs provide data that are fused as input into the 3D object detector of the VoxelNet. Authors consider that the real-time performance deserved much more attention than accuracy for this particular application. To improve efficiency, they replaced dense convolutional layers with spatially sparse convolutions to obtain inference time boosting. Another critical part of Hercules' perception task was to build the 3D map of the environment with 3D point clouds from the LiDARs and the readings from the inertial measurement unit. Objects in motion should be treated differently than static entities. Thus, this task is very crucial for motion planning in AVs. Hercules' solution is called Ego-motion Compensated Moving-object Detection, which detects objects based on the consecutive point cloud frames. The autonomous navigation technology was reported to work well on the state-of-the-art scenarios.

The Intelligent Autonomous Robotic Automobile (IARA) was presented in Moraes et al. [12] and Cardoso et al. [13]. It is a vehicle that uses a suite of sensors (including LiDAR and odometry sensors) to perform various tasks. The two papers present an NN-aided approach to state-of-the-art navigation, mapping, and localization methods. The first publication presents a novel, image-based real-time path planner. Using a CNN, the algorithm can deduce a path from images the on-board camera sends. This path (a cubic spline) is then transformed to the car's coordinates, and thus the route can be followed. The second publication presents a new approach for real-time inference of occupancy maps using deep learning. The network proposed in this paper, NeuralMapper, takes the LiDAR data as input and creates an occupancy grid map. It further uses a NN to infer certain parts of the route where LiDAR data is not very accurate, creating a completer and more efficient map.

If a car detects an obstacle, it brakes and corrects its course, but it does so when the interruption has already occurred. Analyzing human gestures can help avoid sudden collisions; however, deep learning models do not have enough data to train them. For this reason, Cruise's AVs [14] use motion capture to understand human gestures technology, which is a technique that video game developers use to create and animate characters. With this technology, the necessary characteristics would be extracted to train the deep learning models and be one step closer to predicting pedestrians' or drivers' sudden movements.

There are two types of mo-cap systems optical and non-optical. The visual uses cameras distributed over a sizeable grid-like structure that surrounds a stage; the video streams from these cameras can triangulate the 3D positions of visible markers on a full-body suit worn by an actor. In a non-optical way, this system [15] uses a sensor-based version of motion capture instead, which relies on microelectromechanical systems (MEMS) [16][17], which are portable, wireless, and do not require dedicated studio space. That gives a lot of flexibility and allows to take it out of the studio and collect real data of world locations.

Barea et al. presented an integrated CNN for multisensory 3D vehicle detection in a real autonomous driving environment [18]. As the title expresses, the proposal of their paper aims to present an outstanding architecture based on the combination of state of the art methods for object detection such as YOLO and Mask R-CNN for 3D segmentation besides a LiDAR point cloud. Figure 1 shows the results achieved in this paper, integrating Mask R-CNN, YOLOv3 and cloud point LiDAR. It can be seen that an accurate bird eye view, which corresponded with the object detection in the superior image, is created. Moreover, an accurate mapping of the road's current state, with the vital information of the 3D sizing of the moving and static objects in the ego car's peripheric view, is obtained with complete navigation details. The unique architectures applied in this proposal achieved high performance in the KITTY test set.

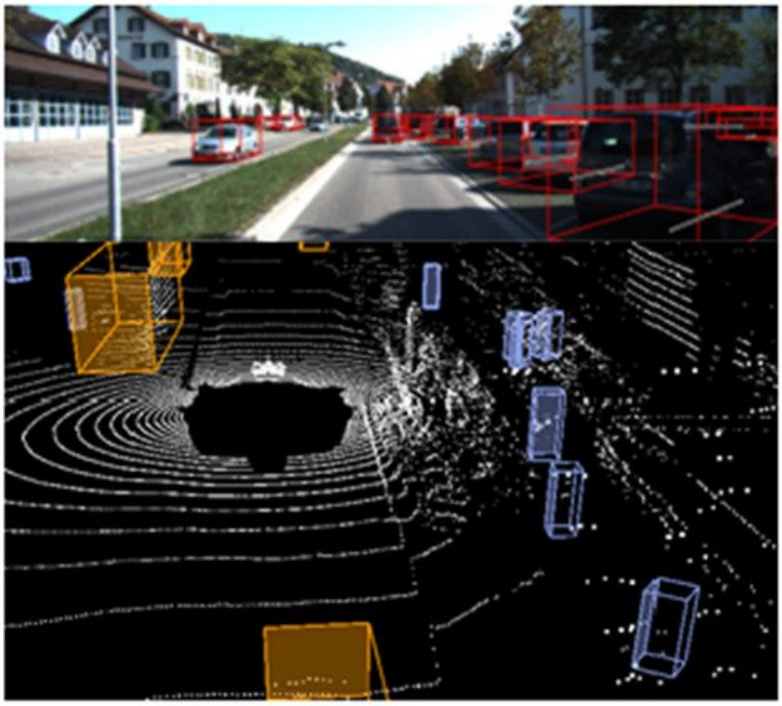

Figure 1. LiDAR's birds-eye view plus boxing object detection.

This work proves that individually each architecture cannot overcome itself. Still, in the correct combination, new opportunities might be found. For the reader, in machine learning and similar technologies, the novel results are achieved based on experimenting and not in a fixed formula. Thus, the following question arises: What would be the best set combination for acquiring the correct data for a completely autonomous car?

As mentioned previously, autonomous driving is a challenging problem addressed with various technologies, methods and algorithms. Still, it seems that there's always space for more efforts due to the rigorous task of taking decisions in an always-changing and mostly unpredictable environment, especially on busy streets and highways. The following paper [19] discusses how image segmentation allows the class to classify the road's drivable and non-drivable regions. To do so, a Mask R-CNN architecture was trained to differentiate these possibilities. The architecture was trained and tested in the Berkeley Deep Drive (BDD100k) dataset, with 100k on road pictures, achieving a final mAP of 0.79 (a sample of the results is provided in Figure 2). To identify the drivable areas, the model determines the ego car's current line, making the difference with the alternatives. The implementation problem was broken into various subproblems; road object detection to declare a line as non-drivable, segmentation breaking a 2D Image into depth-based layers, and lane maker detector to separate each of the lanes.

Figure 2. Line segmentation based on Mask R-CNN.

This paper also explores new possibilities demonstrating that the architectures mentioned in the previous part of this paper can be used in different ways to achieve similar objectives, which provides more information to the ego car, to be able to make better decisions by evaluating the state of the environment. On the other hand, this paper does not mention the time it takes for the algorithm to drop a given state's results.

2.2. Driver Assistance and Predicting Driver Patterns

One characteristic that differentiates a good driver from a bad one is the capability of choosing the best possible action in the proper time and execute it correctly, at all times, even in the rarest and most dangerous circumstances. In those cases, being a skilled driver can save lives. In the context of AVs, this is a real threat because, as mentioned before, one of the biggest challenges for self-driving cars is the unpredictability of the always-changing road conditions and human behaviors. At its beginnings, AVs were based on precarious algorithms, based on state conditionals, with the first driver-assisted systems, acceleration and deceleration, and conditional automation. The human intervenes only when needed. These old approaches were not able to counter the odds of real-time decisions (WIRED Brand Lab, *A brief history of Autonomous Vehicle Technology*, WIRED, 2019, A Brief History of Autonomous Vehicle Technology|WIRED).

Nowadays, state-of-the-art aims to evolve to fully automated systems, where non-human intervention is needed. But one barrier that challenges all the developments is how a human perceives the driving and, consequently, decides how a computer does. Many background context conditions are known for humans that can be subjective and for computers becomes complicated to understand; for example, the local cultural manners of driving often define what to expect from other cars, which differ in geographic location. Another condition is the passengers' feel. This concept means that it might be the same for a computer to avoid an object a second before the collision; therefore, this move can be made smoothly. Conversely, this decision for the passengers may seem uncomfortable, causing other issues, such as a human hitting the window. Therefore, the autonomous system should consider the passengers' inner motions, thus being dynamic and based on human behavior to improve it, without realizing actions that might confront what a human expects from another car.

The mentioned considerations are causes and circumstances that must be considered for a full automotive driving system. Thus, the decision-making part of an autonomous system is a critical part of the process. Driving patterns are ways in which a human driver behaves. These patterns are developed through experience and practice. Humans can predict how the environment can change within these driving patterns, and based on these predictions, a decision is made. Various methods and solutions make use of different NN architectures [20]. Some of them are explained in the following sections.

2.2.1. Behavioral Cloning (BC)

BC consists of a CNN that uses the real driving data obtained through the vehicle's camera and computer when driving during different situations. Then, the network learns the response of the driver to obstacles in other locations. According to [21], a whole model can be trained using only raw data and expect to have a steering output. This approach is called end-to-end learning, but electromobility and autonomous driving named BC are much more appealing. End-to-end learning is called that way because there are no subprocesses when expecting and output from the NN. These subprocesses include SLAM, Path planning, object recognition, etc. None of these techniques is used for the vehicle's motion planning; the only information used is data taken from feature extraction in images taken when the car was being driven. These images have the steering angle and gas pedal position as a label for the dataset. Thus, when seeing a similar scene to the ones in its training data set, then the gas and steering values are close to those trained on.

It could be very promising that there are no subprocesses required for achieving autonomous driving. Using raw images, then the only two values needed for driving are the ones generated. But the reality is much more discouraging. It would require enormous amounts of data, which is not at hand. Because of the model generalization, millions of different driving scenarios need to be recorded but have not yet been taken. BC works only during certain situations, but it works encouragingly. BC is a method that deserves to be discussed in the future once more data is available. For the time being, end-to-end learning is used and applied in natural language processing because of its capability of finding sense in sentences, compared to traditional methods. Data in this language is large enough for end-to-end learning to be applicable, not the case for autonomous driving in which research is being developed but years away from practical applications.

2.2.2. Reinforcement Learning (RL)

One of the most famous machine learning paradigms is RL; this type of algorithm has been used in different ways to process the information given by the sensors to output the best actions. This broad field of study aims to replicate with algorithms how living beings learn through the process of gaining experience by exploring their environment and realizing which activities are suitable and which are not, based on rewards, concerning a specific objective or task inherent to the environment. RL has been widely used as a control algorithm to determine the best policy or a simple word. This strategy

will determine the best possible action for any presented state or circumstances that the environment presents [22]. Figure 3 shows the basic flow of this type of algorithms.

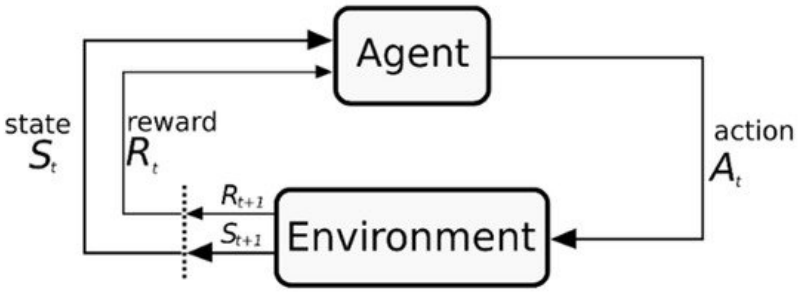

Figure 3. Basic Reinforcement Learning (RL) process, agent-environment relation (image under creative commons license).

The necessary process for an RL to work is as follows. An agent that is the learning object interacts with the environment, which can be virtual or real, the agent performs actions (a) given an environment state s , that will affect the environment, in exchange, the environment will feedback the agent with a new state s ′ and a reward r . If the action were right, then the reward would be positive. Contrarily if the move were wrong, the reward would be negative; in other words, a punishment. The purpose of the agent is to maximize rewards. The process aims to create a system that behaves in the best possible way to achieve a goal.

With a good overview of RL and without deepening the algorithm's details because of this work's scope, this paradigm's application to AVs can be discussed. This type of algorithm is used in the second part of the vehicle's decisions process. First, it has to acquire data of the environment, in this case, the road and the objects such as other cars interacting within it, via the already mentioned and studied approaches such as LiDAR and object detection algorithms via image processing. Based on these data, advanced RL versions based on NNs, such as deep RL, deep Q-learning, and deep deterministic policy gradient (amongst others), have been used for AVs' decision-making processes. These deep versions of RL present the advantage of a more robust, more powerful capability of learning behaviors, understanding patterns, and predicting situations in comparison with their non-deep versions.

An example is control of speed and lane change decisions [23]. As the authors mention, a known problem of decision taking on AVs is that the methods are designed for a specific task. An example would be the Intelligent Driver Model (IDM), which is applied to decide only when to change lines. Therefore, there's a need for a more global algorithm to take care of AVs' multiple decision needs. This work proposes that the car uses a 27-input vector providing information on the road and eight possible surrounding cars to make lane change decisions and speed control. However, in the same DQL algorithm, they trained two agents to address each task [24].

One perceived disadvantage of RL algorithms in autonomous cars is that they cannot predict other road users' intentions but evaluate all the objects' current state. These kinds of learning do not have long-term future recognition of the environment. This problem is intrinsic in the math that the algorithms are based on because Markov's decision process takes decisions based on the last observations state.

2.2.3. LSTM Based Models

Currently, there are many approaches to road prediction using LSTM architectures for these types of forecasts. Bai et al. [25] suggested spatiotemporal LSTM architecture for motion planning in AVs. It uses convolutional layers for feature extraction in an image, and it runs through an LSTM to find sequences in those images, something similar to a CRNN architecture. Once this data is acquired, a CNN architecture is applied to extract spatiotemporal information from multiple frames. Lastly, a fully connected layer is used to extract the trajectory taken by peripheral vehicles. This trained model can be of service for detecting multiple object trajectories. It is considered an end-to-end algorithm since all this raw data enters this architecture, and it outputs a solution. Additionally, it is quite demanding in terms of training and testing data.

Atlché et al. [26] propose using an LSTM architecture for lane changing detection on highways, using data such as velocity and lateral displacement from vehicles. It was trained using the NGSIM US-101 dataset and trained two different LSTM models, one for lateral displacement and another for linear velocity. It performed very rapidly with 70 cm of error in displacement and 3 m / s error for speed, with 10 s ahead of raw predictions. This is a less direct approach since it requires the labeling of data. It can be assumed that they are using a Frenét like topology since they are only considering linear velocity and lateral displacement in their model.

2.3. Human Sentiment Analysis

One of the most critical objectives for intelligent transportation applications is road safety, and unfortunately, one part of traffic accidents is human error. A driver's behavior could affect its way of driving. For that reason, a monitoring system inside an intelligent vehicle would provide relevant information and indicate if the driver's action is allowed. Otherwise, the car would be able to correct it. This monitoring system used cameras inside the cabin of a vehicle to detect face sentiment. Then, CNNs are applied for action recognition. To be more specific, R-CNNs are the ones that select the most informative regions from the drivers [27].

These NNs are trained and tested with an extensive data set to detect any change in gaze and facial emotions, thus discerning regular and aggressive driving [28] or distracted driving actions [29]. In some other cases, there want to predict drivers' behavior; to make this possible, CRNNs are used. Kim et al. [30] proposed a line-segment feature analysis-convolutional recurrent neural network (LFA-CRNN) to analyze drivers' facial expressions indicating pain to prevent automobile accidents and respond to emergency health-risk situations. The LFA-CRNN showed an accuracy of 97.4% in contrast with 98.21% and 97.4% of CRNN and AlexNet.

Other systems also analyze people's voice tones inside the cabin, such as Affectiva Automotive[2], which understands what is happening inside a vehicle. This technology works as both a driver and monitoring tool, ensuring that the safety drivers keep their eyes on the road even as the self-driving software drives the car. An emotional tracker ensures robot taxi passengers feel safe during the trip. The system monitors levels of driver fatigue or drowsiness, driver distraction, understanding driver mood and reactions, and could enhance fleet management. To solve these tasks, the software uses some deep learning architectures that have been mentioned in the previous sections. For instance, for face detection, tracking and classification are used CNNs, in specific RPNs; and for voice-activity detection and classification RNNs, in specific LSTM.

2.4. Using the Cloud with Autonomous Driving Systems

To solve the limitations posed by cloud systems, there are different solutions in the literature. In this section, some of the solutions proposed are discussed. According to Liu et al. [31], the cloud has to provide distributed computing, distributed storage, and heterogeneous computing. It is expected that these systems are tailored together. However, this creates problems with resource sharing because it has to be copied between each application, reducing available space and speed. In this paper, the authors propose a unified infrastructure that complies with the services mentioned. The system developed was reliable, had low latency, and a high-throughput autonomous driving cloud. These practices are useful if anyone wants to build a system in which several AVs share the compiled information.

Kumar et al. [32] developed a cloud-assisted system that involved several vehicles communicating with each other and with several sensors located in the environment. To avoid bandwidth limitations, they used a request-based system, which allowed the car to have information about different locations at various resolutions. They used the Octree representation, which is a 3D-point cloud that can be divided recursively into eight. For more information, the reader is referred to [32].

3. Future Trends

Research on AVs has shown great promise, and there have been so many breakthroughs in recent years. However, there are still many challenges that the community needs to overcome to provide the safest and robust vehicles. All the sensors involved in ADS produce a lot of data from their surroundings, and they are required to be processed in real-time so that the AV can make a correct decision; this is very critical because a small delay could make a considerable change. In this section, some future ADS trends, which can improve performance and reduce response times, will be presented.

3.1. Cloud Computing

Nowadays, cloud computing is becoming a trend for many aspects of our everyday life. More and more businesses are migrating several of their systems to the cloud because of its availability, security, scalability, among others. When discussing businesses' status, payments, or stock inventory, the amount of time required to get a result may be enough if a human perceives it as instantaneous or even tolerated if a small delay occurs. However, in application domains such as autonomous driving, real-time results are a must if accidents or life-threatening scenarios have to be avoided. Thus, the need to explore concepts such as edge and fog within the cloud computing domain. Edge and fog computing consists of moving the location where the information is processed closer to where it is generated [33]. Any device with an internet connection that can store and process data can be used as a fog node (Cisco, "Fog Computing and the Internet of Things: Extend the Cloud to Where the Things Are", *Cisco,* computing-overview.pdf (cisco.com)). This way, the cloud server might

rely on these nodes for most analysis, reducing latency and increasing privacy. The authors considered three characteristics of an application that needs fog computing:

- Data is gathered on an edge. In this case, an AV or a group of AVs can be considered an extreme edge.
- There is a lot of data being generated in a large geographic area.
- The information gathered needs to be analyzed, and it has to provoke a response in less than a second.

An average cable provider in 2008 allowed 16 Mbps (Jakob Nielsen, Nielsen's Law of Internet Bandwidth, *Nielsen Norman Group*, Nielsen's Law of Internet Bandwidth (nngroup.com)). Two years later, bandwidth increased to 31 Mbps. The last measurement was from 2019 with 325 Mbps. This behavior can be described by Nielsen's law, which says that each year, bandwidth increases by approximately 50%. Some authors even compare Nielsen's law with Moore's law, which describes computer power. Despite recent advances in bandwidth capacity, edge computing is still the most critical step for real-time cloud analysis. As shown in [34], a DDNN that involves the edge brings several benefits, such as fault tolerance, privacy, and reduced communication costs.

All and all, it is concluded that edge and fog computing are the best ways to implement a NN in the cloud for applications that require fast analysis like AVs. Despite requiring more powerful end devices, or edge servers, these techniques reduce network traffic, allowing faster analysis of information, more privacy, and fewer transmission errors.

3.2. Parallelization of Neural Networks

The review of deep learning NNs presented in Section 2 showed that the accuracy of models could be improved by increasing the number of parameters and the scale (i.e., nodes, layers, etc.). However, these actions also derive slower training times, which is inefficient for adaptive autonomous driving scenarios. Therefore, it has been acknowledged that deep learning models need to be parallelized to accelerate training and deployment. Indeed, deep learning models can be trained and executed in multiple GPUs. The key consideration to take advantage of their parallelization is to know how to divide the GPUs' tasks. Hence, three approaches should be considered if anyone wants to train parallelized models: data parallelism, model parallelism, and data-model parallelism [33].

- Data parallelism: Data parallelism is a different kind of parallelism that, instead of relying on the process or task concurrency, is related to both the flow and the information structure. Each GPU uses the same model to trains on different data subsets. If there are too many computational nodes, it is necessary to reduce the learning rate to keep a smooth training process.
- Model parallelism: In model parallelism, each computational node is responsible for parts of the model by training the same data samples. The computational nodes communicate between them when a neuron's input is from another computational node's output. However, the performance is worse than data parallelism, and the performance of the network will be decreased if it has too many nodes.
- Data-model parallelism: Both previous models have disadvantages, but they also have positive characteristics. Model parallelism could get good performance with many neuron activities, and data parallelism is efficient with many weights.

In summary, data-model parallelism combines the best of the first two approaches and is more suited for AV tasks as long as there is a clear idea of where to apply each step. For instance, the convolutional layer of a CNN could use data parallelism (since it contains about 90% of the computation and 10% of the parameters). In contrast, the fully connected layer could only rely on model parallelism because it contains 90% of the parameters and 10% of the computation.

3.3. Parallelization of the Whole System

One thing that is rarely discussed in research is the likelihood of the proposed solution being applicable in real-life settings. Factors such as the time required for the whole data to be processed and if that time is good enough for real-time applications are commonly discussed for a system to be deployed in the real world. However, hardware requirements for the processing are acceptable for these applications. Code parallelization that accelerates the clocks in the processor at just the right rate speed (but not too much so that hardware operates at dangerous temperatures) needs to be acknowledged well. In this regard, thermal throttling can be considered a viable solution. Thermal throttling is a safety protocol that the processor starts once the system reaches a determined temperature. It starts to diverge power from processing the data towards the cooling system techniques to avoid any hardware malfunctions.

Although this technique is beneficial for the hardware, it derives from slower data processing (almost half of its capabilities). For example, using thermal throttling in image processing might drop the frame rate from 30 FPS to 15 FPS (or less); this quality drop in autonomous driving could even be dangerous as it could result in lower recognition rates and maximum risk of accidents. This issue can be addressed using proper parallelization techniques in CPU and GPU architectures. It is implemented for data to be transferred smoothly throughout the system without frame drops and appropriate speed for real-time applications without compromising the hardware's integrity. For instance, CUDA libraries from NVIDIA (NVIDIA, "CUDA Toolkit 10.1 original Archive", *NVIDIA developer*, CUDA Toolkit 10.1 original Archive|NVIDIA Developer) have excellent parallelization frameworks to segment complex instructions into smaller ones; all of them are run concurrently in different GPU CUDA cores, giving the system the robustness and speed it requires. From Google, TensorFlow API also has adequate parallelization protocols for NN training and deployment in GPU architectures. It can be used in object detection, instance segmentation, and RNN prediction tasks. The robotics operating system provides parallelization options in CPU for the SLAM, path planning, communication protocols, and wireless data transfer for multiple path planning operations in the Frenét coordinate system.

3.4. First Thoughts towards an Approach for LSTM-Based Path Planning Prediction

After considering the conditions mentioned above, we propose a novel LSTM approach to predict both lateral displacement and linear velocity in a Frenét coordinate system. Many techniques have been proposed to achieve this (as described in Section 2.5 and Section 3). Still, they are considered isolated ways to solve the problem as they do not consider integrating the overall pipeline. The reader must recall that vast amounts of data are required to perform this task and acknowledge that a critical concept is often ignored in this phase, i.e., causality. It is essential to consider a high chance of a reckless driver making a poor judgment that could affect the AV, or that a single car's predictions would not change because another care changes the whole driving scenario to poor or unexpected choices. Even deployed systems, such as the Ego Car (described in Section 3), would be part of this causality scenario. Therefore, using the output from the Frenét coordinate system that was taken from object classification via YOLO and boundary classification with instance segmentation, a bidirectional LSTM can be fed with the Frenét data, both for lateral displacements and linear velocities.

These are four different LSTM models that feed their weights and share weights between lateral displacements and linear velocities. As seen in Figure 4, the LSTM architecture is represented graphically.

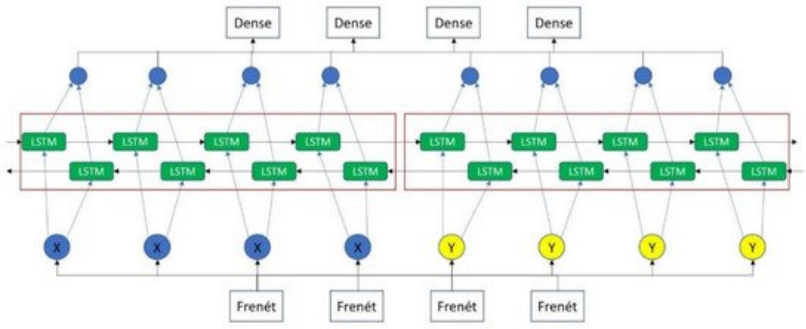

Figure 4. LSTM representation for the prediction task.

Every Frenét point of each car will be fed to the model, including the Ego Car points. Thus, the prediction will consider every variable in the scenario and change its predictions if the causality demands it. This will derive in a new Frenét prediction point for every neuron in the LSTM architecture, and so the position, orientation, and velocity of every point can be predicted. As a result, the car's volume can be included in every future time instance. This output data can be used for decision-making in an AV, reducing the number of accidents enormously since causality is part now of the overall algorithm deployment and execution.

References

1. Singh, S. Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey; National Hi ghway Traffic Safety Administration; U.S. Department of Transportation, National Highway Traffic Safety Administration: Washington, DC, USA, 2015.
2. Rangesh, A.; Trivedi, M.M. No Blind Spots: Full-Surround Multi-Object Tracking for Autonomous Vehicles Using Camer as and LiDARs. IEEE Trans. Intell. Veh. 2019, 4, 588–599.

3. Huang, Y.; Chen, Y. Autonomous Driving with Deep Learning: A Survey of State-of-art Technologies. arXiv 2020, arXiv:2006.06091.
4. Yurtsever, E.; Lambert, J.; Carballo, A.; Takeda, K. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. IEEE Access 2020, 8, 58443–58469.
5. Badue, C.; Guidolini, R.; Vivacqua, R.; Azevedo, P.; Brito, V.; Forechi, A.; Ferreira, A. Self-Driving Cars: A Survey. arXiv 2019, arXiv:1901.04407.
6. Szymak, P.; Piskur, P.; Naus, K. The Effectiveness of Using a Pretrained Deep Learning Neural Networks for Object Classification in Underwater Video. Remote. Sens. 2020, 12, 3020.
7. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-View 3D Object Detection Network for Autonomous Driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017.
8. Chen, C.; Qin, C.; Qiu, H.; Tarroni, G.; Duan, J.; Bai, W.; Rueckert, D. Deep Learning for Cardiac Image Segmentation: A Review. Front. Cardiovasc. Med. 2020, 7, 25.
9. Zhou, Y.; Tuzel, O. VoxelNet End-to-End Learning for Point Cloud Based 3D Object Detection. arXiv 2017, arXiv:1711.06396.
10. Zhou, Y.; Tuzel, O. MVX-Net: Multimodal VoxelNet for 3D Object Detection. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019.
11. Liu, T.; Liao, Q.; Gan, L.; Ma, F.; Cheng, J.; Xie, X.; Wang, Z.; Chen, Y.; Zhu, Y.; Zhang, S.; et al. Hercules: An Autonomous Logistic Vehicle for Contact-less Goods Transportation During the Covid-19 Outbreak. arXiv 2020, arXiv:2004.07480.
12. Moraes, G.; Mozart, A.; Azevedo, P.; Piumbini, M.; Cardoso, V.B.; Oliveira-Santos, T.; De Souza, A.F.; Badue, C. Image-Based Real-Time Path Generation Using Deep Neural Networks. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020.
13. Cardoso, V.B.; Oliveira, A.S.; Forechi, A.; Azevedo, P.; Mutz, F.; Oliveira-Santos, T.; Badue, C.; De Souza, A.F. A Large-Scale Mapping Method Based on Deep Neural Networks Applied to Self-Driving Car Localization. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020.
14. Tabian, I.; Fu, H.; Khodaei, Z.S. A Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures. Sensors 2019, 19, 4933.
15. Weaver, C. Self-Driving Cars Learn to Read the Body Language of People on the Street, IEEE SPECTRUM. Available online: (accessed on 17 October 2020).
16. Aranzeta-Ojeda, L.; Moreno-García, C.F.; Granados-Reyes, A.; Bustamante-Bello, R. Design, Development and Testing of a Low-Cost, High Sensitivity System for Neurodegenerative Disease Detection and Characterization. In Proceedings of the International Conference on Microtechnologies and Medical Biology (MMB), Lucerne, Switzerland, 4–6 May 2011; pp. 64–65.
17. Bustamante-Bello, R.; Aranzeta-Ojeda, L.; Moreno-Garcia, C.F. Design and Development of a Low-Cost, High Sensitivity Device for Neurodegenerative Disease Detection. In Proceedings of the 24th IEEE International Conference Micro Electro Mechanical Systems (MEMS), Cancun, Mexico, 23–27 January 2011.
18. Barea, R.; Bergasa, L.M.; Romera, E.; López-Guillén, E.; Perez, O.; Tradacete, M.; López, J. Integrating State-of-the-Art CNNs for Multi-Sensor 3D Vehicle Detection in Real Autonomous Driving Environments. In Proceedings of the IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; Volume 1.
19. Shah, N.; Shankar, A.; Park, J.-h. Detecting Drivable Area for Autonomous Vehicles. arXiv 2020, arXiv:1911.02740.
20. Meng, X.; Lee, K.K.; Xu, Y. Human Driving Behavior Recognition Based on Hidden Markov Models. In Proceedings of the 2006 IEEE International Conference on Robotics and Biomimetics, Kunming, China, 17–20 December 2006.
21. Curiel-Ramirez, L.A.; Ramirez-Mendoza, R.A.; Bautista-Montesano, R.; Bustamante-Bello, M.R.; Gonzalez-Hernandez, H.G.; Reyes-Avedaño, J.A.; Gallardo-Medina, E.C. End-to-End Automated Guided Modular Vehicle. Appl. Sci. 2020, 10, 4400.
22. Sutton, R.S.; Barto, A.G. Reinforcement Learning: An Introduction, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2018.
23. Hoel, C.J.; Wolff, K.; Laine, L. Automated Speed and Lane Change Decision Making using Deep Reinforcement Learning. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018.
24. El Sallab, A.; Abdou, M.; Perot, E.; Yogamani, S. Deep Reinforcement Learning framework for Autonomous Driving. Electron. Imaging 2017, 2017, 70–76.

25. Bai, Z.; Cai, B. Deep Learning-Based Motion Planning for Autonomous Vehicle Using Spatiotemporal LSTM Network. arXiv 2019, arXiv:1903.01712.
26. Atlché, F.; de la Fortelle, A. An LSTM Network for Highway Trajectory Prediction. arXiv 2018, arXiv:1801.07962.
27. Yan, S.; Teng, Y.; Smith, J.; Zhang, B. Driver behavior recognition based on deep convolutional neural networks. In Proceedings of the 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), Changsha, China, 13–15 August 2016; pp. 636–641.
28. Naqvi, R.A.; Arsalan, M.; Rehman, A.; Rehman, A.U.; Loh, W.-K.; Paul, A. Deep Learning-Based Drivers Emotion Classification System in Time Series Data for Remote Applications. Remote. Sens. 2020, 12, 587.
29. Hu, Y.; Lu, M.; Lu, X. Feature refinement for image-based driver action recognition via multi-scale attention convolutional neural network. Signal Process. Image Commun. 2020, 81, 115697.
30. Kim, C.-M.; Hong, E.J.; Chung, K.; Park, R.C. Driver Facial Expression Analysis Using LFA-CRNN-Based Feature Extraction for Health-Risk Decisions. Appl. Sci. 2020, 10, 2956.
31. Liu, S.; Tang, J.; Wang, C.; Wang, Q.; Gaudiot, J.L. Implementing a Cloud Platform for Autonomous Driving. arXiv 2017, arXiv:1704.02696.
32. Kumar, S.; Gollakota, S.; Katabi, D. A Cloud-Assisted Design for Autonomous Driving. In Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC); ACM: Helsinki, Finland, 2012.
33. Li, X.; Zhang, G.; Li, K.; Wang, Z. Chapter 4: Deep Learning and Its Parallelization. In Big Data: Principles and Paradigms; Morgan Kauffman: Burlington, MA, USA, 2016.
34. Teerapittayanon, S.; McDanel, B.; Kung, H.T. Distributed Deep Neural Networks over the Cloud, the Edge and End Devices. In Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS), Atlanta, GA, USA, 5–8 June 2017.