Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Real-Time Vehicle and Lane Detection using Modified OverFeat CNN: A Comprehensive Study on Robustness and Performance in Autonomous Driving

Monowar Hossain Saikat[1], Sonjoy Paul Avi[2], Kazi Toriqul Islam[3], Tanjida Tahmina[4] , Md Shahriar Abdullah[5] and Touhid Imam[6] ✉

[12]*Department of Civil & Environmental Engineering, Lamar University, Texas, USA*
[3]*Department of Engineering Management. Trine University. 1 University Ave, Angola, IN 46703, USA*
[4]*Department of Manufacturing and Industrial Engineering, University of Texas Rio Grande Valley, Edinburg, TX, USA*
[5]*Department of Civil and Environmental Engineering, Lamar University, TX USA*
[6]*Department of Computer Science, University of South Dakota, Vermillion, South Dakota, USA*
Corresponding Author: Touhid Imam, **E-mail**: Touhid.Imam@coyotes.usd.edu

| ABSTRACT
This examination researches the use of profound learning methods, explicitly utilizing Convolutional Brain Organizations (CNNs), for ongoing recognition of vehicles and path limits in roadway driving situations. The study investigates the performance of a modified Over Feat CNN architecture by making use of a comprehensive dataset that includes annotated frames captured by a variety of sensors, including cameras, LIDAR, radar, and GPS. The framework shows heartiness in identifying vehicles and anticipating path shapes in 3D while accomplishing functional rates of north of 10 Hz on different GPU setups. Vehicle bounding box predictions with high accuracy, resistance to occlusions, and efficient lane boundary identification are key findings. Quiet, the exploration underlines the likely materialness of this framework in the space of independent driving, introducing a promising road for future improvements in this field.

| KEYWORDS
Real-Time Vehicle; Lane Detection; Modified OverFeat CNN; Robustness; Autonomous Driving

| ARTICLE INFORMATION
ACCEPTED: 01 April 2024 **PUBLISHED:** 11 April 2024 **DOI:** 10.32996/jcsts.2024.6.2.4

1. Introduction

Ever since the DARPA Grand Challenges introduced the concept of vehicles there has been a surge, in applications and research related to self-driving cars. One key aspect of this technology is the driving environments it can operate in with highways and urban roads representing two contrasting scenarios. Highways are generally more predictable and well organized with maintained road surfaces and clearly marked lanes. On the hand urban driving involves greater unpredictability with various objects on the road inconsistent lane markings and complex traffic flow patterns. The structured nature of highways has allowed for some implementations of autonomous driving technology. Many car manufacturers are now focusing on developing highway auto pilot systems that aim to reduce driver stress and fatigue while providing safety features. Advanced driver assistance systems (ADAS) can currently help keep cars within their lanes and detect vehicles ahead. However human drivers still bear responsibility for any obstacles or serious incidents by always keeping their hands on the steering wheel.

The performance gap between auto pilot systems and fully autonomous vehicles, like those developed by Google is largely influenced by financial considerations. Today self-driving vehicles are equipped with sensors, like LIDAR, radar and high precision GPS. These sensors work in conjunction, with maps to ensure reliable autonomous navigation. In today's production-grade

autonomous cars, critical sensors include radar, sonar, and cameras. Long-range vehicle detection normally needs radar, but local automobile detection can be done with sonar. Computer vision may play an important role in lane detection as well as redundant object detection at intermediate distances. Radar works quite well for detecting automobiles but has difficulties differentiating between different metal items and hence can report false positives on objects such as tin cans. Also, radar offers minimal orientation information and has a bigger variation in the lateral position of objects, making localization problematic on acute bends.

The efficacy of sonar is both reduced at fast speeds and, even at modest speeds, restricted to a working distance of about 2 meters. Compared to sonar and radar, cameras provide a richer collection of characteristics for a fraction of the expense. By advancing computer vision, cameras might serve as a dependable, redundant sensor for autonomous driving. Despite its potential, computer vision has yet to occupy a substantial role in today's self-driving automobiles. Classic computer vision systems just have not delivered the robustness necessary for production-grade automotive; these techniques require substantial manual engineering, road modeling, and special case handling. Considering the apparently unlimited variety of driving situations, environments, and unanticipated impediments, the effort of scaling classic computer vision to robust, human-level performance would be enormous and likely unachievable.

Deep learning, or neural networks, is an alternate method for computer vision. It offers tremendous potential as a remedy for the inadequacies of standard computer vision. Recent research in the subject has enhanced the practicality of deep learning applications to tackle complicated, real-world issues, and the industry has responded by expanding the use of such technologies. Deep learning is data-focused, requiring extensive computing but little hand-engineering. In the last several years, an increase in accessible storage and computation capabilities has enabled deep learning to achieve success in supervised perception tasks, such as image detection. A neural network, after training for days or even weeks on a big data set, can be capable of inference in real-time with a model size that is no greater than a few hundred MB [1]. State-of-the-art neural networks for computer vision require huge training sets paired with extended networks capable of simulating such immense amounts of data. For example, the ILSRVC data set, where neural networks obtain top performance, comprises 1.2 million images in over 1000 categories. By leveraging expensive existing sensors that are already employed for self-driving applications, such as LIDAR and precise GPS,[2] and calibrating them with cameras, we may produce a video data set comprising labeled lane markings and annotated cars with location and relative speed. By constructing a labeled data set in all sorts of driving scenarios (rain, snow, night, day, etc.), we can test neural networks on this data to see if they are resilient in every driving environment and situation for which we have training data. In this study, we give an empirical assessment of the data set we collected. In addition, we discuss the neural network that we employed for identifying lanes and automobiles, as illustrated in Figure 1.

2. Related Work

In the rapidly evolving landscape of autonomous driving, Computer Vision plays a pivotal role, albeit with certain limitations necessitating complementary sensor fusion and road models for enhanced precision. Noteworthy studies have employed diverse approaches, such as reinforcement learning in highway scenarios, where S. Nageshrao et al. demonstrated autonomous vehicles' decision-making prowess. P. Chuan-Hsian and C. -S. Sea's research showcased Dark net outperforming Tensor Flow in vehicle detection accuracy. J. Wang et al. addressed highway driving challenges through supervised and reinforcement learning, incorporating LSTM for improved performance. G. Prabhakar et al. developed a deep learning system for obstacle detection, while A. A. Hasanaath proposed a real-time road condition monitoring mechanism achieving high accuracy. Z. Wei's computer vision system excelled in lane change detection, and K. Muhammad's survey offered insights into deep learning architectures' reliability in autonomous driving. Additionally, studies by Yang et al., Dhawan et al., and Yi et al. focused on workload detection, traffic sign classification, and personalized driving state recognition, respectively. Emphasizing the importance of road infrastructure, a study targeted road markings' damage detection using computer vision, utilizing deep learning for improved F1-scores in Japanese and Spanish images, albeit with a call for more extensive image collection for further advancements in the field.

3. Methodology

3.1 Real-time vehicle detection

Convolutional neural networks (CNNs) have had the largest success in image recognition in the previous 3 years. From these image recognition systems, several detection networks were developed, leading to further advances in image detection. While the advances have been startling, not much emphasis has been paid to the real-time detection speed necessary for applications. In this study, we demonstrate a detection system capable of running at better than 10Hz Hz using nothing but a laptop GPU. Due to the needs of highway driving, we need to verify that the system utilized can identify automobiles more than 100m away and can work at rates greater than 10 Hz; this distance demands higher picture resolutions than are typically used, which in our instance are 640 × 480. We employ the Over feat CNN detector, which is extremely scalable and replicates a sliding window detector in a single forward pass in the network by efficiently recycling convolutional findings on each layer. Other detection techniques, such as R-CNN, rely on selecting as many as 1000 candidate windows, where each is evaluated independently and does not reuse

convolutional findings. In our implementation, we make a few modest modifications to over feat's labels in order to manage automobile occlusions, predict lanes, and increase performance during inference. We will first offer a quick description of the original implementation and then discuss the adjustments. Over feat converts an image recognition CNN into a "sliding window" detector by giving a bigger resolution image and transforming the fully connected layers into convolutional layers.

Then, after converting the fully connected layer, which would have produced a single final feature vector, to a convolutional layer, a grid of final feature vectors is formed. Each of the resulting feature vectors reflects a slightly different context viewpoint inside the original pixel space. To determine the stride of this window in pixel space, it is straightforward to simply multiply the strides on each convolutional or pool layer together. The network we employed has a stride size of 32 pixels. Each final feature vector in this grid may predict the presence of an item; if an object is discovered, those same characteristics are then utilized to predict a single bounding box using regression. The classifier will predict no object if it cannot detect any part of an item within the whole input view. This produces large ambiguities for the classifier, which can only predict a single object, as two separate items might readily appear in the context view of the final feature vector, which is typically larger than 50% of the input picture resolution.

The network we utilized has a context view of 355×355 pixels in size. To guarantee that all objects in the picture are classified at least once, multiple distinct context views of the image are obtained by employing skip-gram kernels to minimize the stride of the context views and by using up to four different scales of the input image. The classifier is then trained to activate when an object occurs anywhere inside its whole context view. In the original Over feat study, this results in 1575 alternative context views (or final feature vectors), where each one is likely to become active (form a bounding box). This presents two challenges for our empirical examination. Due to the L2 loss between the predicted bounding box and actual bounding proposed by Sermonette et al., the ambiguity of having two valid bounding box locations to predict when two objects appear is incorrectly handled by the network by predicting a box in the center of the two objects to minimize its expected loss.

These boxes tend to present a difficulty for the bounding box merging method, which wrongly thinks that there must be a third item between the two ground truth objects. This might cause difficulties for an ADAS system that incorrectly believes there is a car when there is not, and emergency breaking is wrongly applied. In addition, the merging algorithm, used solely during inference, runs in O(n 2) where n is the number of bounding boxes suggested. Because the bounding box merging is not as easily parallelizable as CNN, this merging may become the bottleneck of a real-time system in the case of an ineffective implementation or too many predicted bounding boxes.

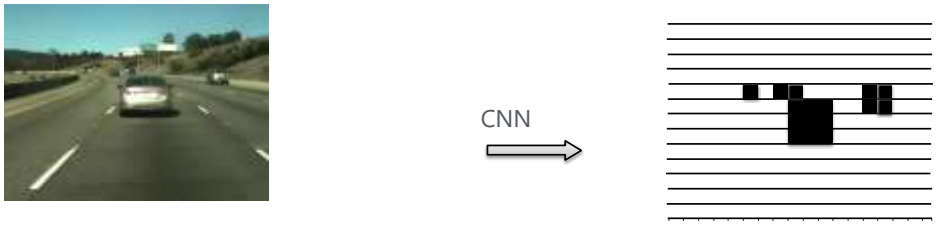

Fig. 1: mask detector.

In our assessments, we apply a mask detector as published in Szegedy et al. [10] to ameliorate some of the difficulties with over feat as discussed above. Szegedy et al. offer a CNN that takes an image as input and generates an object mask through regression, indicating the object's position. The idea of a mask detector is given in Fig. 2. To differentiate multiple nearby objects, various part-detectors generate object masks, from which bounding boxes are subsequently derived. The detector they suggest must take numerous crops of the image, then run multiple CNNs for each portion on every crop. Their resulting implementation takes around 5-6 seconds per frame per class using a 12-core system, which would be too sluggish for our application.

We combine these methods by employing the efficient "sliding window" detector of over feat to generate an object mask and perform bounding box regression. This is demonstrated in Fig. 2. In this method, we employ a single picture resolution of 640 × 480 with no skip-gram kernels. To ease the ambiguity problem and limit the number of bounding boxes predicted, we adjust the detector on the top layer to only activate within a 4 × 4-pixel region at the center of its context view, as shown in the first box in Fig 2. Because it's exceedingly unlikely that any two separate objects bounding boxes occur in a 4 × 4-pixel region, compared to the complete context view with over feat, the bounding box regressor will no longer have to arbitrarily select between two legitimate objects in its context view.

In addition, because the need for the detector to fire is stronger, this yields much fewer bounding boxes, which greatly decreases our run-time performance during inference. Although these adjustments improved, ambiguity was still a common problem on the

boundary of bounding boxes in the circumstances of occlusion. This uncertainty results in a false bounding box being predicted between the two ground-truth bounding boxes. To fix this problem, the bounding boxes were initially shortened by 75% before constructing the detection mask label. This introduced the additional criterion that the center 4×4-pixel section of the detector window had to be within the center region of the object before activation.

Our modifications to the network happen on the dense layers, which are changed to convolution, as reported in sermonette et al. [1]. When using our higher picture sizes of 640 × 480 this transforms the prior final feature response maps of size 1 × 1 × 4096 to 20 × 15 × 4096. As indicated previously, each of these feature vectors views a context region of 355355 pixels, and the stride between them is 32 × 32 pixels; nevertheless, we want each to generate predictions at a resolution of 4 × 4 pixels, which would leave gaps in our input image. To remedy this, we use each 4096 features as input to 64 SoftMax classifiers, which are arranged in an 8×8 grid and each predict if an item is within a distinct 4×4-pixel zone. This enables the 4096-feature vector to span the full stride size of 32 × 32 pixels; the ultimate result is a grid mask detector of size 160 × 120 where each. element is 4×4 pixels, which covers the whole input picture of size 640 × 480.

3.2 Lane Detection

The CNN utilized for vehicle detection may be readily extended for lane border detection by adding class. Whereas the regression for the vehicle class predicts a five-dimensional value (four for the bounding box and one for depth), the lane regression predicts six dimensions. Similar to the vehicle detector, the first four dimensions represent the two endpoints of a local line segment of the lane boundary. The remaining two dimensions reflect the depth of the endpoints concerning the camera. Fig. 3 visualizes the lane boundary ground truth label placed on an example picture. The green tiles show sites where the detector is trained to fire, and the line segments indicated by the regression labels are explicitly drawn. The line segments have their ends connected to make continuous splines.

The depth of the line segments is color-coded so that the closest segments are red, and the furthest ones are blue. Due to our data gathering methods for lane labels, we can extract ground truth in spite of objects that occlude them. This requires the neural network to learn more than a basic paint detector and must utilize context to forecast lanes where there are occlusions. Like the vehicle detector, we employ L1 loss to train the regressor. We employ mini-batch stochastic gradient descent for optimization. The learning rate is regulated via a variant of the momentum scheduler [11]. To get semantic lane information, we employ DBSCAN to cluster the line segments into lanes. our lane predictions after DBSCAN clustering. Different lanes are indicated by different colors. Since our regressor produces depths as well, we can anticipate the lane shapes in 3D using inverse camera perspective mapping.

3.3 Experiment Setup

3.3.1 Data Collection

Our research vehicle is a 2014 Infiniti Q50. The car currently employs the following sensors: 6x Point Grey Flea3 cameras, 1x Velodyne HDL32E lidar, and 1x Novatel SP receiver We also have access to the Q50 built-in Continental mid-range radar system. The sensors are linked to a Linux PC with a Core i7- 4770k CPU. Once the raw films are acquired, we annotate the 3D sites for cars and lanes, as well as the relative speed of all the cars.

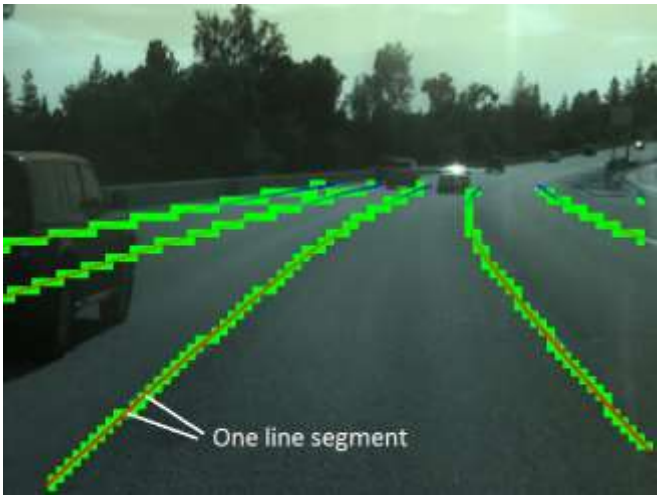

Fig. 2: Example of lane boundary ground truth

To collect vehicle annotations, we follow the typical technique of using Amazon Mechanical Turk to acquire precise bounding box coordinates inside pixel space. Then, we match bounding boxes with radar returns to obtain the distance and relative speed of the vehicles. Unlike automobiles that may be tagged using bounding boxes, highway lane boundaries sometimes need to be marked as curves of various forms. This makes frame-level tagging not only tedious and wasteful but also prone to human mistakes. Fortunately, lane markers may be treated as static objects that do not alter their geolocations very often. We follow the procedure described in to build LIDAR maps of the environment using Velodyne and GNSS equipment. Using these maps, labeling is simple. First, we filter the 3D point clouds based on lidar return intensity and position to derive the left and right bounds of the ego-lane. Then, we duplicate the left and right ego-lane bounds to obtain initial predictions for all the lane boundaries.

A human annotator inspects the created lane boundaries and makes appropriate corrections using our 3D labeling tool. For completeness, we describe each of these processes in depth. 1) Ego-lane border generation: Since we do not change lanes during data collection trips, the GPS trajectory of our research car already offers a fair approximation of the shape of the road. We can then simply determine the ego-lane boundaries using a few heuristic filters. Noting that lane limits on highways are frequently indicated with retro-reflective materials, we first filter out low-reflectivity surfaces such as asphalt in our 3D point cloud maps and only examine spots with high enough laser return intensities. We next filter out other reflective surfaces, such as autos and traffic signs, by only considering points whose heights are close enough to the ground plane.

Lastly, assuming our car drives close to the center of the lane, we filter out ground paint other than the ego-lane boundaries, such as other lane boundaries, carpool signs, or directional signs, by only considering markings whose absolute lateral distances from the car are smaller than 2.2 meters and greater than 1.4 meters. We may also identify the left border from the right one using the sign of the lateral distance. After obtaining the points in the left and right bounds, we fit a piecewise linear curve comparable to the GPS trajectory to each boundary. 2) Semi-automatic development of various lane boundaries: We note that the width of lanes during a single data collection session is constant most of the time, with occasional exceptions such as mergers and splits. Therefore, if we predefine the number of lanes to the left and right of the automobile for a single run, we may make a decent first approximation of all the lane boundaries by pushing the auto-generated ego-lane boundaries laterally by multiples of the lane width. We will then rely on human annotators to address the exception instances.

At the time of this writing, our annotated data set consists of 14 days of driving in the San Francisco Bay Area during the months of April and June for a few hours each day. The vehicle-annotated data is captured at 1/3Hz and comprises nearly 17 thousand frames with 140 thousand bounding boxes. The lane-annotated data is captured at 5 Hz and comprises about 616 thousand frames. During training, translation and seven different perspective distortions are performed on the raw data sets. Fig. 3 shows an example picture after perspective distortions are applied. Note that we apply the same perspective distortion to the ground truth labels so that they correspond appropriately with the distorted picture.

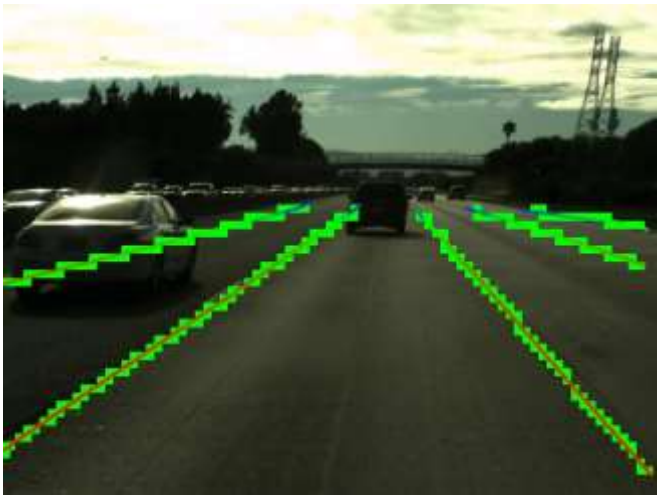

Fig. 3: Image after perspective distortion

4. Results

When used on a desktop PC with a GTX 780 Ti, the detection network has the capability to operate at 44 Hz. The network can run at 2:5 Hz with a mobile GPU like the Tegra K1, and the system should be able to run at 5 Hz with the Nvidia PX1 chipset, according to expectations.Our path discovery test set contains 22 video cuts got from both left and right cameras across 11 particular

information assortment meetings, summarizing to around 50 minutes of driving film. This assessment evaluates the identification results for four path limits: the outer edges of the two lanes that are adjacent, in addition to the left and right borders of the ego lane. Every path limit's appraisal is additionally arranged by longitudinal distances, traversing from 15 to 80 meters in front of the vehicle, divided at 5-meter spans. As a result, there are a maximum of 4 * 14 = 56 positions available for evaluating detection results. Using a greedy nearest neighbor matching strategy, the prediction and ground truth points at these locations are paired together. Genuine up-sides, bogus up-sides, and misleading negatives are organized at every evaluation point utilizing a standard convention: A genuine positive is enrolled when the matched forecast and ground truth shift by under 0.5 meters. Assuming that the matched expectation and ground truth contrast by more than 0.5 meters, both misleading positive and bogus negative counts are expanded.

Figure 4 presents a visual portrayal of this assessment approach inside a solitary picture. Blue dots denote true positives, red dots denote false positives, and yellow dots denote false negatives. In Figure 8, the joined accuracy, review, and F1 score across all test recordings are exhibited. Explicitly concerning the inner self path limits, we accomplish a 100 percent F1 score inside a 50-meter range. Notwithstanding, review begins declining remarkably past 65 meters because of the picture goal's powerlessness to catch the path markings' width at that distance. The closest point has a lower recall when it comes to the adjacent lanes because it is out of the camera's field of view.

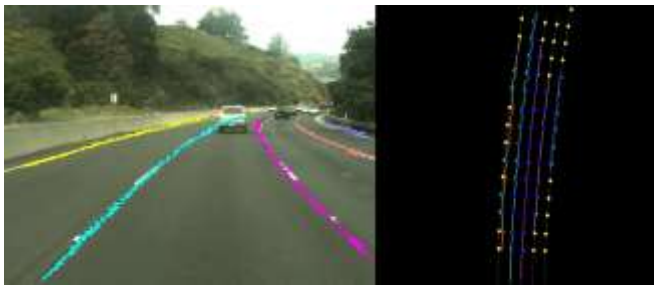

Fig. 4: Left: lane prediction on test image. Right: Lane detection evaluated in 3D

like these. It seems like these clasps show the indicator's crude recognitions with no extra sifting or street models, isn't that so? It's interesting that the network only looked at cars going the same way from the back, which could explain why cars crossing the highway barrier might get missed.

That is fabulous information! Publicly releasing the code for the vehicle and path identifier on GitHub permits others to investigate, use, and possibly add to the turn of events. Forking the storehouse from the first Caffe code base by the BVLC bunch exhibits the cooperative idea of such activities, empowering further headways and local area contribution.

5. Conclusion

By utilizing cameras, lidar, radar, and GPS, we produced a highway data collection consisting of 17 thousand picture frames with vehicle bounding boxes and over 616 thousand image frames with lane annotations. We then trained on this data using a CNN architecture capable of recognizing all lanes and automobiles in a single forward pass. Using a single GTX 780 Ti, our system runs at 44 Hz, which is more than acceptable for real-time use. Our results suggest existing CNN algorithms are capable of good performance in highway lane and vehicle detection. Future work will focus on gathering frame-level annotations that will allow us to create new neural networks capable of using temporal information across frames.

This analysis demonstrates a real-time vehicle and lane detection system using a modified version of the over Feat CNN. The system achieves high accuracy and can run at better than 10 Hz on a laptop GPU. It is robust to occlusions and can predict lane shapes in 3D. The authors have collected a large dataset of annotated data to train and test their system. This system has the potential to be used in self-driving cars.

Here are some key takeaways from this analysis:

Over Feat CNN can be used for both vehicle and lane detection.
The system is robust to occlusions and can predict lane shapes in 3D.
The authors have collected a large dataset of annotated data.
The system has the potential to be used in self-driving cars.

Overall, this paper presents a promising approach for real-time vehicle and lane detection. The system is accurate, robust, and efficient, and it has the potential to revolutionize the way we drive.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Adnan A., Mahbubur R G. M., Hossain M. M., Mim M. S. and Rahman M. K., (2022) A Deep Learning Based Autonomous Electric Vehicle on Unstructured Road Conditions, 2022 IEEE 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE), Penang, Malaysia, 2022, pp. 105-110, doi: 10.1109/ISCAIE54458.2022.9794498.
[2] Dhawan, K., R, S.P. & R. K., N. (2023) Identification of traffic signs for advanced driving assistance systems in smart cities using deep learning. Multimed Tools Appl 82, 26465–26480 (2023). https://doi.org/10.1007/s11042-023-14823-1
[3] Demetriou, A., Alfsvåg, H. and Rahrovani, S. et al. (2023) A Deep Learning Framework for Generation and Analysis of Driving Scenario Trajectories. SN COMPUT. SCI. 4, 251 (2023). https://doi.org/10.1007/s42979-023-01714-3
[4] Giunchiglia, E., Stoian, M.C., Khan, S. et al. (2023) ROAD-R: the autonomous driving dataset with logical requirements. Mach Learn 112, 3261–3291 (2023). https://doi.org/10.1007/s10994-023-06322-z
[5] Nadeem, H., Javed, K., Nadeem, Z., Khan, M. J., Rubab, S., Yon, D. K., & Naqvi, R. A. (2023). Road Feature Detection for Advance Driver Assistance System Using Deep Learning. Sensors, 23(9), [4466]. https://doi.org/10.3390/s23094466
[6] Tu, J., Mei, G. & Piccialli, F. (2022) An Efficient Deep Learning Approach Using Improved Generative Adversarial Networks for Incomplete Information Completion of Self-driving Vehicles. J Grid Computing 20, 21 (2022). https://doi.org/10.1007/s10723-022-09610-5
[7] Wang, D., Wang, C. and Wang, Y. et al. (2021) An Autonomous Driving Approach Based on Trajectory Learning Using Deep Neural Networks. Int.J Automot. Technol. 22, 1517–1528 (2021). https://doi.org/10.1007/s12239-021-0131-2
[8] Yang, Y., Sun, H., Liu, T., Huang, GB., and Sourina, O. (2015). Driver Workload Detection in On-Road Driving Environment Using Machine Learning. In: Cao, J., Mao, K., Cambria, E., Man, Z., Toh, KA. (eds) Proceedings of ELM-2014 2. Proceedings in Adaptation, Learning and Optimization, 4. Springer, Cham. https://doi.org/10.1007/978-3-319-14066-7_37
[9] Yi, D., Su, J., Liu, C., Quddus, M., & Chen, W-H. (2019). A machine learning based personalized system for driving state recognition. Transportation Research Part C: Emerging Technologies, 105, 241-261. https://doi.org/10.1016/j.trc.2019.05.042