Activation Function: Key to Cloning from Human Learning to Deep Learning

Pranit Gopaldas Shah[1], Hiral Pranit Shah[2]
[1] Dept. of Computer Science & Engineering,
Parul Institute of Engineering & Technology, Vadodara, India.
[2] Senior Data Scientist, TeerHub Technology Private Limited, Vadodara, India.
pranit.shah@gmail.com, Hiral@teerhubtech.com

Abstract. Maneuvering a steady on-road obstacle at high speed involves taking multiple decisions in split seconds. An inaccurate decision may result in crash. One of the key decision that needs to be taken is can the on-road steady obstacle be surpassed. The model learns to clone the drivers behavior of maneuvering a non-surpass-able obstacle and pass through a surpass-able obstacle. No data with labels of "surpass-able" and "non-surpass-able" was provided during training. We have development an array of test cases to verify the robustness of CNN models used in autonomous driving. Experimenting between activation functions and dropouts the model achieves an accuracy of 87.33% and run time of 4478 seconds with input of only 4881 images (training + testing). The model is trained for limited on-road steady obstacles. This paper provides a unique method to verify the robustness of CNN models for obstacle mitigation in autonomous vehicles.

Keywords: BC, end-to-end learning, saliency map, computer vision, behavioral cloning, autonomous vehicles, self-driving, obstacle mitigation.

1 Introduction

Affordability of powerful computational hardware and advances in deep learning techniques has made vision-based autonomous driving an active research focus within the transport industry. There are considerable drawbacks in the techniques to overcome, even though the research worldwide has already taken giant leap. Foremost downside is the inability to explicitly model each possible scenario. Driving needs responding to a large variety of complex environmental conditions and agent behaviors.

End-to-end method and perception-driven method are the two popular vision-based paradigms for self-driving cars. A perception-based method lacks self-learning ability and all features including task plans are manually hand crafted. This is the major disadvantage of perception-based method.

End-to-end Behavior cloning (Off-policy imitation learning) provides an alternative to traditional modular approach by simultaneously learning both perception and control using deep network.

Maneuvering each and every steady on-road obstacle either surpass able or non-surpass-able is cost intensive. Again maneuvering a steady on-road obstacle at high speed involves taking multiple decisions in split seconds. An inaccurate decision may result in crash. One of the key decision that needs to be taken is can the steady on-road obstacle be surpassed.

Clearly, an autonomous driving vehicle successfully navigating through the streets should be able to follow the roadway as well as maneuver only in cases required. If the autonomous vehicle can surpass a steady on-road obstacle without being unstable it must do so. Therefore, we here in propose an improved convolution neural network model. Overall, this research work makes the following contributions:

- Provides evaluation method for popular learning models by defining test cases to mitigate an on-road obstacle.
- Identify, evaluate and validate the configuration producing optimum results by performing combination of activation function and dropout.
- Improve prediction accuracy by validating statistical data with visual saliency map.

The paper has been organized as follows: Section 2 gives an overview of related work done in past and present. Section 3 describes the methodology used. Section 4 Experimental Setup, Results and Discussion. Finally, Section 5 concludes this paper.

2 Related Work

Perception-driven method ad End-to-end method are the two popular vision-based paradigms for self-driving cars. Both the perception and end-to-end methods have been reviewed extensive through literature study and presented here. However, the key aspect of autonomous driving is the problem of object detection itself. Hence in the later part of this section we review our study on object detection in the context of Deep Learning.

2.1 Perception-driven Learning Methods

Traditional Perception based method has made remarkable achievements, in past decades, in the field of self-driving cars. Several methods of detections have been proposed to generate a description of the local environment. Depending on the technique used current detection research can be broadly classified as shown in Figure 1 below

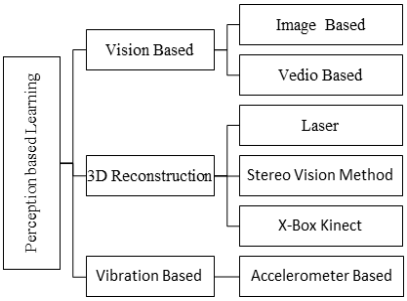


Fig. 1: Perception-driven learning

A. Vision Based Detection:

Object location through enclosing boxes has been adopted by researchers as a part of object detection task. This bounding box can be anything from a steady road signs, traffic signs, moving cars, moving bicycles, etc. The model is trained with labeled data of objects. The key factor in case of self-driving car is an ability to identify if there is an obstacle and at what distance. It is immaterial to have the exact location of the bounding box.

Kwang Eun An [10] used Deep Convolutional Neural Network (DCNN) for image classification into obstacle/pothole or non-pothole. The method compared Inception_v4, Inception_ResNet_v2, ResNet_v2_152 and MobileNet_v1 for comparison between color and grayscale images. This method is limited in its efficiency in processing single frame of image.

Canny edge detection or Hough transformation area is implemented to locate lane position through many lane detection methods. There are no specific geometric restrictions required to identify uneven lane boundaries in these methods.

B. 3D Reconstruction:

There are three major approaches for 3D reconstruction of obstacle/pothole each with its own drawback.

1) Chang et al. [19] used a Grid based processing technique. Here a surface receives laser incidents and digitally implements the bounced back pulse to generate a precise. The output was accurate, however this was expensive. Li et al. [20] used infrared laser line projectors, a digital camera and a multi-view coplanar scheme for calibrating the lasers. The method plotted more feature points in the cameras point of view and was much more cost effective.

2) Wang [21] used a series of cameras. This method generated a 3D surface model through a series of 2D images captured. High computation requirement was key backdrop.

3) Xbox Kinect sensor was used by Joubert et al. [22] and Moazzam et al. [23]. The method could not minimize the error and power for computing, even though equipment price was minimized.

C. Vibration based detection:

Umang Bhatt [26] combined accelerometer, gyroscope, location and speed data to classify road condition and detect potholes/obstacles. SVM with radial basic function (RBF) kernel was used for road condition classification. SVM and gradient boost were used for pothole/non-pothole classification. Failure attributed includes inability to accurately classify between good and bad road due to insufficient data for all road

types. The key backdrop was inability to distinguish between a bump, a manhole or a pothole.

One key takeaway from all the above work done in perception based learning provides proactive data to the driver regarding the obstacle. These methods do not provide any method for routing through obstacle. Again, there is not classification if an obstacle can be surpassed or not.

2.2 End-to-end Learning Methods

Pomerleau [9] pioneered end-to-end training of neural network to steer a car. In 1989 Pomerleau built the Autonomous Land Vehicle in a Neural Network (ALVINN) system.

In the scenario of autonomous driving one of the key requirement is an ability to identify salient objects.

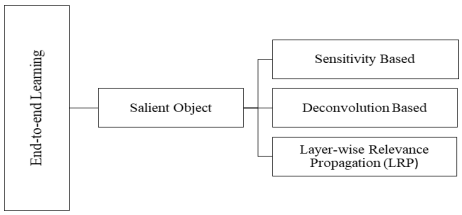


Fig. 2: End-to-end Learning

A. Sensitivity Based:

After 25 years of advances in data and computational power DAVE-2 in [3] demonstrated the potential of end-to-end learning. A CNN based model introducing PilotNet Network Architecture designed by NVIDIA. The key differentiator for this architecture was its ability to identify salient objects without the need for hand-coded rules and instead learn by observing. However Dave's PilotNet model admits that the convolutional layers were chosen empirically and hence the performance was not sufficiently reliable to provide a full alternative to more modular approaches to off-road driving.

B. Deconvolution Based:

Matthew [28] presented a method for mid and high-level feature learning like corners, junctions and object parts. Mattew [28] resolves the two fundamental problems found in image descriptors like SIFT, HOG or edge gradient calculators followed by some histogram or pooling operation. First being invariance to orientation and scale. Secondly a CNN models inefficiency in training each model with respect to input. Visualization available is for one activation per layer.

C. Layer-wise Relevance Propagation:

Alexander Binder [29] implemented Layer-wise relevance propagation to compute scores for image pixels and image regions denoting the impact of the particular image region on the prediction of the classifier for one particular test image. Alexander Binder [29] demonstrated controlling the noisiness of the heatmap, however an optimal trade between numerical stability and sparsity/meaningfulness of the heatmap was kept as item for future work.

Salient object based methods does identify the key features that impact the steering angles. However these methods have not been explicitly developed/tested for identifying if an obstacle can be surpassed or not.

To overcome the above mentioned limitations, we propose to perform extensive training and testing for a neural network to clone an obstacle mitigation policy. Even though there is a great deal of work and literature on the task of steering angle prediction, our goal is not to propose yet another method for prediction, but rather provide a different perspective on on-road steady obstacles mitigation model. A model to not only detect an on-road steady obstacle but also to predict the obstacle can be surpassed to avoid unnecessary maneuvering.

3 Methodology

This section provides the details on CNN models used for validation and steps performed on data for accurate prediction.

3.1 Preprocessing

Network Model continuously predicts the steering angle to clear all test cases with an input of raw pixels incorporating attention in an end-to-end manner. It's important that our experiments and results are independent of car geometry, hence we represent steering command as inverse turning radius r_t^{-1} (r is turning radius at time stamp t). We use inverse to prevent numerical instability, singularity, and smooth transitions through zero from left turns to right turns. The relation between steering angle θ_t and inverse turning radius can be given as

$$\theta_t = f_{Steers}(u_t) = u_t d_w K_s (1 + K_{slip} v_t^2) . \quad (1)$$

Where θ_t in degree and v_t (m/s) is a steering angle and velocity at time t, respectively. K_s, K_{slip} and d_w are vehicle-specific parameters. K_s is steering ratio between the turn of the steering and the turn of the wheels. K_{slip} represents the relative motion between the front and rear wheels.

3.2 Data Bias Removal

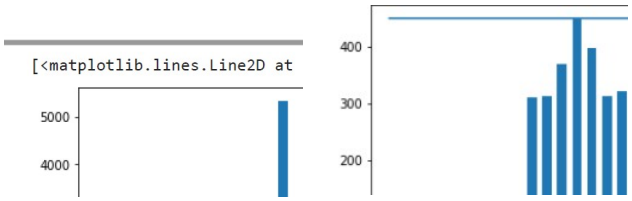


Fig. 3: Left: Data with 5000+ images having '0' steering. Right: Non-biased data with sample reduced to 400+ per bin to have a uniformly distributed steering.

The general tendency of every driver is to drive as steady as possible without lot of maneuvering. However in case of behavioral cloning if the driver drives steady during training then all the model will learn is to maintain a zero steering angle. Such a training would generate results bias to zero output. In order to avoid output bias towards a –ve/+ve steering angle driving the training is done for complete track in clockwise and then anticlockwise direction over track. In fact additional recovery training tracks are also done where the car is take off the center lane and then recovered to back. The data bias is removed by trimming samples per bin as shown in Figure 3.

3.3 Encoder model: Convolution to clone obstacle mitigation behavior

Convolutional Neural Networks (CNN) are adaptations of multi-layer perceptron and are biologically-inspired. In context of self-driving cars, CNNs are able to learn the entire task of lane and road following without manual decomposition into road or lane marking detection, semantic abstraction, path planning and control. The network learns to detect the outline of a road without the need of explicit labels during training. The following research implements variants of the CNN architecture established by NVIDIA for self-driving cars, the PilotNet[3] and DroNet[32].

4 Experimental Setup, Results and Discussion

This section presents the basic description of experimental setup for data collection, training and testing. We elaborate further on the configuration of hardware and software used. Later we enlist the training cases and test cases for model evaluation and the evaluation criteria. Eventually we enlist the results from our experiment.

Model training and testing is performed in a virtual environment Unity 2017 in the interest of research cost. Other software tools include Visual studio for Unity, Atom, Jupyter Notebook and Anaconda. GitHub for online code repository and Google Colab platform for online code execution. Programming is done in c# and python. Multiple packages including OpenCV, numpy, matplotlib, Keras (model, optimizer, layers), pandas and sklearn are used. Hardware includes my laptop with Intel i5 processor@1.27GHz, 8GB RAM and Intel HD Graphics family with 2GB RAM.

Virtual 3D models of non-surpass-able and surpass-able obstacles are created in unity as shown in Figure 4 below



Fig. 4: Left: Car model alongside non-surpass-able obstacle. Right: Car model alongside surpass-able obstacle.

3 Cameras are mounted in front of the model car to capture left, center and right images.

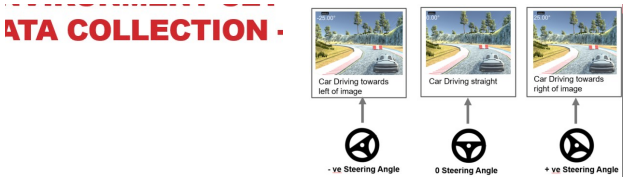


Fig. 5: Left: Front camera to capture images. Right: Steering angle and car driving direction

4.1 Training and Test cases

Our network model is trained on only 4 cases and is expected to pass 10 test cases based on training. These training cases are to train the network to clone driver's behavior while driving through both non-surpass-able and surpass-able obstacles. Training cases are as listed in Figure 6 below

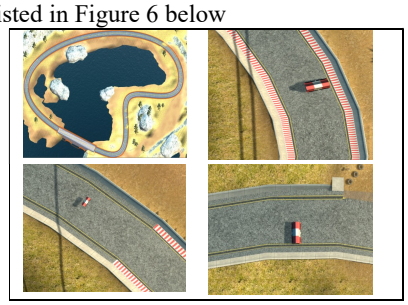


Fig. 6: Top Left: Training Case-TRC01: The track is empty without any obstacles. The model is trained on this empty track. Top Right: TRC02: The track has a non-surpass-able obstacle on the right of simulated car. The model it trained to maneuver through the obstacle. Bottom Left: TRC03: The track has a surpass-able obstacle in the centre of track. The model it trained to pass through the obstacle. Bottom Right: TRC04: The track has a non-surpass-able obstacle on the left of simulated car. The model it trained to maneuver through the obstacle.

Test case are to verify the robustness of networks ability to learning to maneuver through varied configurations of obstacles. Test cases are as listed in Figure 7 below

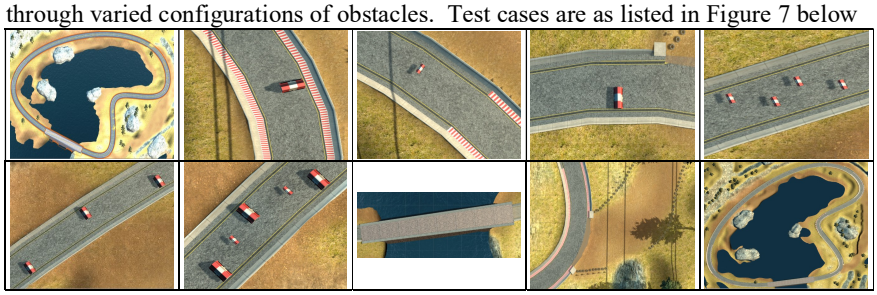


Fig. 7: Starting from Top Left 1st: Test Case-TC01: The track is empty without any obstacles. The model is supposed to drive through the entire track to call it has passes this test case. 2nd:TC02: The track has a non-surpass-able obstacle on the right of simulated car. The model is supposed to maneuver through the obstacle and retain a steady drive to call it has passes this

test case. 3rd:TC03: The track has a surpass-able obstacle. The model is supposed to pass through the obstacle with minimum vehicle instability and maintain a steady drive to call it has passes this test case. 4th:TC04: The track has a non-surpass-able obstacle on the left of simulated car. The model is supposed to maneuver through the obstacle and retain a steady drive to call it has passes this test case. 5th:TC05: The track has an array of multiple surpass-able obstacles. The model is supposed to pass through the obstacles with minimum vehicle instability and maintain a steady drive to call it has passes this test case. Starting Bottom Left 1st:TC06: The model is supposed to maneuver through an array of left and right non-surpass-able obstacles and retain a steady drive to call it has passes this test case. 2nd:TC07: The track has an array of both multiple left and right non-surpass-able and surpass-able obstacles. The model is supposed to pass through surpass-able obstacles and maneuver through the obstacle to call it pass. 3rd:TC08: In this case an unknown bridge, which has a different track, has to be crossed to call this case passes. 4th:TC09: In this case an unknown unpaved path has to be avoided by the model. The model has not been trained for this behavior. 5th:TC10: In model is expected to clear all the obstacles, pass through unknown bridge and unpaved path all in a single stretch.

4.2 Model Accuracy Calculation

Each model is trained in 4 cases and tested for 10 cases. The model accuracy is defined as follows
Number of test cases model passed: P
Score for a test cases in which model self-recovered: SR
Model Accuracy: MA
Total test cases: T
Number of obstacles correctly maneuvered or surpassed: SR_o
Total maneuverable and surpass-able obstacles: T_o

$$MA = ((P+SR)/ T)/100 . \quad (2)$$

$$SR = SR_o/T_o . \quad (3)$$

4.3 Dataset Characteristics

Right size of data set is the key to accurate predictable solution. Initial training was started with 30,000 images, however due to resource constraints, we reached an optimal data size of 9970 images that produced reliable results. Zero steering bias images are removed from input images. Data augmentation is performed to increase the data size and accuracy. The final data set is classified using test_train_split functionality in sklearn library. All models are trained using 3904 samples and validated with 977 samples as shown in Table 1 below

Table 1. Data Characteristics for different models.

Model	Input Data	Input Data after removing bias		Trainable parameters
		Training Samples	Validation Samples	
DroNet	9930	3904	977	311777
PilotNet	9930	3904	977	252219

This paper represents two deep learning models tested with a combination of activation function and dropout on same database. Table 2 below shows model used, code assigned for ease, configurations used and val_loss achieved. Table 2 below clearly indicated that model P1, PilotNet with elu and No Dropout, achieved the best val_loss of 0.0250.

Table 2. Configuration and val_loss comparison for different models.

Model	Code	Configuration	val_loss
DroNet	D1	Dropout + Relu	0.0295
	D2	No Dropout + elu	0.0731
PilotNet	P1	No Dropout+ elu	0.0250
	P2	Dropout + elu	0.0454
	P3	No Dropout + Relu	0.0276
	P4	No Dropout + Sigmoid	0.0934
	P5	No Dropout + Softmax	0.0863

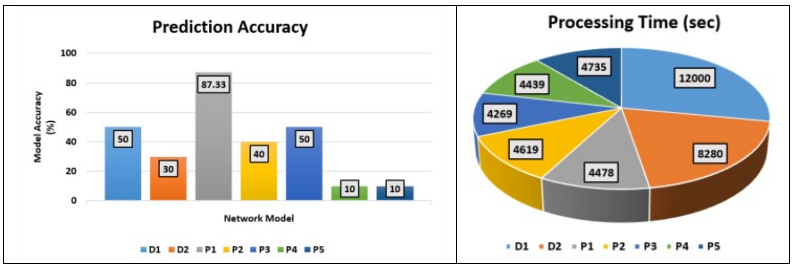
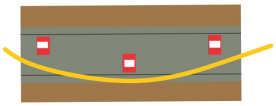
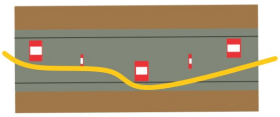


Fig. 8: Left: Prediction Accuracy. Right: Processing time

Figure 8 Left and right shows the prediction accuracy and model running time when using different configurations on two learning models using same dataset. Model P1, PilotNet with elu and no dropout, achieves highest prediction accuracy of 87.33%. Model P4 and P5 has the least accuracy of 10%. The highest accuracy achieved by both DroNet model D1 and D2 are 50% and 30% respectively with heavy processing time of 12000s and 8280s respectively. Model P1 achieves the highest accuracy consuming the processing time of 4478 seconds.

Model P1, PilotNet with elu and no-dropout, performed a self-recovery in test cases 6 and 7 as listed in Table 5 below

Table 3. Path Followed and individual test case accuracy for Model P1.

	Test Case 6	Test Case 7
Path Followed		
Description	The model maneuvers 1st, skips 2nd by going off the track and recovers back post	The model maneuvers 1st, pass through 2nd, skips 3rd by going off the track and

	skipping 3rd obstacle.	recovers back post skipping 4th and 5th obstacle.
SR calculation	$SR_o = 1$; $T_o = 3$; $SR_6 = 0.33$	$SR_o = 2$; $T_o = 5$; $SR_7 = 0.4$

Overall Model Accuracy for model P1 is calculated as below

$$MA_{P1} = ((P_{P1}+SR_{P1})/ T)/100$$
$$= ((8 + (0.33+0.4))/10)/100$$
$$= 87.33\%$$

Saliency Map is generated to co-related and validate the statistical results obtained. PilotNet models are taken for saliency mapping for identification of left non-surpass-able, surpass-able and right non-surpass-able obstacles. As depicted in table below model P4 and P5 has failed completely which co-relates with only prediction accuracy of 10%. Model P2 has wrongly detected the saliency for surpass-able obstacle and has not detected the lane boundaries at all. Model P1 perform better than P3 with better saliency map both for obstacles and lanes as depicted in table 6

Table 4. Saliency Map for PilotNet based Models.

Model	Non-surpass-able left(alpha=0.004)	Surpass-able (alpha=0.003)	Non-surpass-able right(alpha=0.0045)
P2			
P3			
P1			
P4			
P5			

5 Conclusion

In this paper, we presented and compared two most popular autonomous driving methods including DroNet and PilotNet. We experimented with combinations of different activation functions with-without dropout. The experiment has demonstrated the PilotNet model P1 is able to learn the entire task of non-surpass-able obstacle maneuvering and passing through a surpass-able obstacle. The experiment has provided us with a clear insight into effect of each activation function and dropout on steering angle prediction. PilotNet model P1 has the highest prediction accuracy, lowest val_loss and reasonable processing time with best visual saliency map for obstacles with current dataset. The experiment has clearly concluded that PilotNet,

with activation function elu without dropout, outperforms all other models and configurations.

The system learned to mitigate through an obstacle without the need of explicit surpass-able and non-surpass-able obstacle labeling during training.

In the future work, we would like to optimize PilotNet to improve prediction accuracy. We would also like to introduce a custom-Net that would outperform all current autonomous driving methods.

References

1. R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," IEEE transactions on pattern analysis and machine intelligence, vol. 38, no. 1, pp. 142–158, 2016.
2. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779– 788.
3. Bojarski, Mariusz, PhilipYeres, AnnaChoromanska, KrzysztofChoromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. "Explaining how a deep neural network trained with end-to-end learning steers a car." arXiv preprint arXiv:1704.07911 (2017).
4. Bratko, Ivan, Tanja Urbani, and Claude Sammut. "Behavioural cloning: phenomena, results and problems." IFAC Proceedings Volumes 28, no. 21 (1995): 143-149.
5. Kumaar, Saumya and Navaneethkrishnan, B and Hegde, Sinchana and Raja, Pragadeesh and Vishwanath, Ravi M (2019) Towards Behavioural Cloning for Autonomous Driving. In: IEEE INTERNATIONAL CONFERENCE ON ROBOTIC COMPUTING (IRC 2019), 25-27 Feb. 2019, Naples, Italy, pp. 560-567.
6. Bojarski, Mariusz & Testa, Davide & Dworakowski, Daniel & Firner, Bernhard & Flepp, Beat & Goyal, Prasoon & Jackel, Lawrence & Monfort, Mathew & Muller, Urs & Zhang, Jiakai & Zhang, Xin & Zhao, Jake & Zieba, Karol. (2016). End to End Learning for Self-Driving Cars.
7. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 25, pages 1097–1105. Curran Associates, Inc., 2012. URL: http://papers.nips.cc/paper/ 4824-imagenet-classification-with-deep-convolutional-neural-networks. pdf.
8. Net-Scale Technologies, Inc. Autonomous off-road vehicle control using end-to-end learning, July 2004. Final technical report. URL: http://net-scale.com/doc/net-scale-dave-report.pdf.
9. Dean A. Pomerleau. ALVINN, an autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, 1989.
10. An, Kwang & Lee, Sung & Ryu, Seung-ki & Seo, Dongmahn. (2018). Detecting a pothole using deep convolutional neural network models for an adaptive shock observing in a vehicle driving. 1-2. 10.1109/ICCE.2018.8326142.
11. Srivastava, Sumit & Sharma, Ayush & Balot, Harsh. (2018). Analysis and Improvements on Current Pothole Detection Techniques. 1-4. 10.1109/ICSCEE.2018.8538390.
12. Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using b-snake," Image and Vision computing, vol. 22, no. 4, pp. 269–280, 2004.
13. A. A. Assidiq, O. O. Khalifa, M. R. Islam, and S. Khan, "Real time lane detection for autonomous vehicles," in Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on. IEEE, 2008, pp. 82–88.

14. Y. Li, A. Iqbal, and N. R. Gans, “Multiple lane boundary detection using a combination of low-level image features,” in Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on. IEEE, 2014, pp. 1682–1687.
15. J. He, H. Rong, J. Gong, and W. Huang, “A lane detection method for lane departure warning system,” in Optoelectronics and Image Processing (ICOIP), 2010 International Conference on, vol. 1. IEEE, 2010, pp. 28–31.
16. Z. Nan, P. Wei, L. Xu, and N. Zheng, “Efficient lane boundary detection with spatial-temporal knowledge filtering,” Sensors, vol. 16, no. 8, p. 1276, 2016.
17. E. Buza S. Omanovic and A. Huseinovic: Pothole Detection with Image Processing and Spectral Clustering. In 2nd International Conference on Information Technology and Computer Networks, Pages 48–53, 2013.
18. K. C. P. Wang: Challenges and feasibility for comprehensive automated survey of pavement conditions, In 8th International Conference on Applications of Advanced Technologies in Transportation Engineering (2004), Pages 531-53
19. Z. Hou, K. C. P. Wang, and W. Gong: Experimentation of 3D pavement imaging through stereovision, In International Conference on Transportation Engineering (2007), Pages 376-381.
20. D. Joubert, A. Tyatyantsi, J. Mphahlehle, and V.Manchidi: Pothole tagging system, In 4th Robotics and Mechatronics Conference of South Africa(2011), Pages 1-4.
21. Moazzam, K. Kamal, S. Mathavan, S. Usman, and M. Rahman: Metrology and visualization of potholes using the Microsoft Kinect sensor, In 16th International IEEE Annual Conference on Intelligent Transportation Systems(2013), Pages 1284-1291.
22. Shah, Pranit & Pandya, Krishna & Shah, Harsh & Gandhi, Jay. (2019). Survey on Vision based Hand Gesture Recognition. International Journal of Computer Sciences and Engineering. 7. 281-288. 10.26438/ijcse/v7i5.281288.
23. Li, Q., Yao, M., Yao, X and Xu, B. (2009): A real-time 3D Scanning System for pavement distortion inspection, measurement science and technology,Pages 15702-15709.
24. Bhatt, Umang & Mani, Shouvik & Xi, Edgar & Kolter, J.. (2017). Intelligent Pothole Detection and Road Cond0ition Assessment.
25. B. Huval, T. Wang, S. Tandon, J. Kiske, W. Song, J. Pazhayampallil, M. Andriluka, P Rajpurkar, T. Migimatsu, R. Cheng-Yue et al., “An empirical evaluation of deep learning on highway driving,” arXiv preprint arXiv:1504.01716, 2015.
26. C. Chen, A. Seff, A. Kornhauser, and J. Xiao, “Deepdriving: Learning affordance for direct perception in autonomous driving,” in Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 2722–2730.
27. Binder, Alexander & Lapuschkin, Sebastian & Montavon, Gregoire & Müller, Klaus-Robert & Samek, Wojciech. (2016). Layer-Wise Relevance Propagation for Deep Neural Network Architectures. 913-922. 10.1007/978-981-10-0557-2_87.
28. Zhao, Zhong-Qiu & Zheng, Peng & Xu, Shou-Tao & Wu, Xindong. (2019). Object Detection With Deep Learning: A Review. IEEE Transactions on Neural Networks and Learning Systems. PP. 1-21. 10.1109/TNNLS.2018.2876865
29. Wang, Wenguan & Lai, Qiuxia & Fu, Huazhu & Shen, Jianbing & Ling, Haibin. (2019). Salient Object Detection in the Deep Learning Era: An In-Depth Survey.
30. A. Loquercio, A. I. Maqueda, C. R. del-Blanco and D. Scaramuzza, "DroNet: Learning to Fly by Driving," in IEEE Robotics and Automation Letters, vol. 3, no. 2, pp. 1088-1095, April 2018.
31. Pranit Shah, Krishna Pandya, Harsh Shah, Jay Gandhi, “Survey on Vision based Hand Gesture Recognition,” International Journal of Computer Sciences and Engineering, Vol.7, Issue.5, pp.281-288, 2019