Simulation of Self-driving Car using Deep Learning

Aman Bhalla
Computer Science and Engineering
National Institute of Technology, Raipur
Raipur, India
amanbhalla50@gmail.com

Munipalle Sai Nikhila
Computer Science and Engineering
National Institute of Technology, Raipur
Raipur, India
nikhilamunipalli2008@gmail.com

Pradeep Singh
Computer Science and Engineering
National Institute of Technology, Raipur
Raipur, India
psingh.cs@nitrr.ac.in

2020 3rd International Conference on Intelligent Sustainable Systems (ICISS) | 978-1-7281-7089-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICISS49785.2020.9315968

***Abstract*— The rapid development of Artificial Intelligence has revolutionized the area of autonomous vehicles by incorporating complex models and algorithms. Self-driving cars are always one of the biggest inventions in computer science and robotic intelligence. Highly robust algorithms that facilitate the functioning of these vehicles will reduce many problems associated with driving such as the drunken driver problem. In this paper our aim is to build a Deep Learning model that can drive the car autonomously which can adapt well to the real-time tracks and does not require any manual feature extraction. This research work proposes a computer vision model that learns from video data. It involves image processing, image augmentation, behavioural cloning and convolutional neural network model. The neural network architecture is used to detect path in a video segment, linings of roads, locations of obstacles, and behavioural cloning is used for the model to learn from human actions in the video.**

Keywords— Self-driving cars, Deep Learning, Computer Vision, Behavioural Cloning, Image Augmentation

I. INTRODUCTION

With the rapid developments in technology, computers are leaving behind the old notion of 'Computers are dumb machines.' Among all the achievements obtained in various fields of computer science, an autonomous vehicle has been one of the biggest and most important invention. It has been a topic of research for many years and numerous algorithms involving advanced concepts of artificial intelligence. Popularity of these cars are increasing tremendously. Recent studies show that around 10 Million self-driving cars would be on-road in near future[1]. Companies such as Audi, Volvo,Ford, Google, General Motors, BMW, and Tesla incorporated this technology in their latest releases. Nowadays, many autonomous vehicles are found in busy towns such as Columbus and Ohio. However, these involve a supervising driver to assist the agent. According to level of human interference required, self-driving cars are categorized into 5 levels by the Society of Automotive Engineers (SAE) [2]. These are:

Level 0: This vehicle is operated completely manually. Every single decision is taken by human.

Level 1: ADAS (Advanced Driver Assistance System) supports the human driver with either steering, braking or speed. ADAS provides rear-view cameras and moving seat warning options to alert drivers while driving off the road.

Level 2: Similar to ADAS, vehicles of this level provide the functionality of indicating car movements, detection of neighbouring vehicles, etc.

Level 3: Vehicles of this level perform all driving activities independently. However, in restricted conditions, such as car parking, the human driver should be able to take over the control.

Level 4: Cars categorized under level 4 can perform all driving tasks and in bound situations controlling the driving environment. These are accurate enough in most of the conditions and demand very less human intervention.

Level 5: The future cars or those grouped under this level serve as virtual drivers and keep moving in all informed situations. They have the capability to take independent decisions in known or unknown scenarios.

Until 2018, self-driving cars were in level 3. Development of Reinforcement Learning models gave scope for self-exploration and reward maximization. This has instantly changed the state-of-the-art to level 4. Now the main focus of research is to develop vehicles of level 5 where human behaviour understanding is considered for an AI agent.

Our approach is a deep learning model which is based on 'Behavioural Cloning' [3], a concept in which the agent learns from human behaviour. This can be recognized as a level 5 algorithm. In this paper development of CNN [4] model and training with the data using behavioural cloning is performed. Image pre-processing and image augmentation is also performed to provide more data to the model. A comparative analysis is also performed with the other deep learning models on the simulator provided by the Udacity [5]. The rest of this paper is organized as follows. Section II reviews some related works and Section III explains the methodology while results are presented in Section IV. The conclusion of this paper is given in Section V.

II. RELATED WORK

With the rapid developments in computer technology, there has been a great increase in the capabilities of computers. Now, the computers are capable of performing Terabytes of calculation in just a few minutes. This has paved a path for the growth of Machine Learning amongst the community. An extensive study of the research done in the field of Self-

Driving cars is conducted and explored. Reinforcement Learning and Deep Learning are the two main approaches involved in solving the problem of self-driving cars. However, this research work focuses on the Deep Learning based approach since it is capable of capturing the style and expertise of the driver through a concept known as Behavioural Cloning [3].

With the onset of powerful Graphical Processing Units (GPUs), Deep Learning has become a choice of technology for researchers as well as for developers working in the field of self-driving cars. With the combination of Deep Learning and OpenCV[6], there is a huge opportunity for building robust autonomous vehicles. Fujiyoshi et al. demonstrated how Deep Learning involving CNNs are being used in the tasks of Image Recognition, Object Detection and Semantic Segmentation [7]. Further, the paper presents how Deep Learning is being used currently to infer the control values of an autonomous car through technique known as end-to-end. It also presents visual explanation for such a technique which can boost the confidence of passengers of autonomous cars. Kulkarni et al. demonstrated how Faster Region based Convolutional Network (R-CNN) [8] can be used for detecting traffic lights efficiently [9]. Bojarski et al. achieved an autonomy value of 90% in virtual environments using his end-to-end approach consisting of a Convolutional Neural Network (CNN) [4][10]. This approach demonstrated how a single-camera can be used to feed images to CNN to get the steering angles accurately. Jain built a working model of the self-driving car using Raspberry Pi and Lidar sensor and showed how CNNs can be used to drive a real-world model of a car without any human intervention [11]. Kim et al. collected the data from a game using end-to-end method and employed an autonomous driving technique announced by Nvidia [12]. Further, authors have used AlexNet, one of the most popular CNN [4] models. Kang et al. did a thorough study of 37 publicly-available datasets and 22 virtual testing environments, which serves as a guide for researchers and developers involved in the designing of self-driving cars [13]. It included various datasets like Berkeley DeepDrive Video dataset (BDDV) [14], Cambridge-driving Labeled Video Database (CamVid) [15]etc. The various virtual testing environments studied in the above mentioned paper included AirSim (Microsoft) [16], CARLA [17], TORCS [18] and several others.

III. PROPOSED WORK

The objective behind this approach is to leverage the power of Supervised Learning in conjunction with Deep Learning to achieve our end goal. We focused on a concept called 'Behavioural Cloning' [3], which, essentially, is a technique of teaching the machine to learn from a human subject. By Behavioral cloning human car driving actions are captured and along with the situation that gave rise to the action. A log of these records is used as input to the machine learning [3]. The simulator Udacity [5] is opensource and available with Training Mode or Autonomous Mode. In the training mode, manual car drive is used to record the driving behavior. The recorded images and the behaviour are used to train deep learning model. Once trained the Autonomous Mode is used for testing the machine learning models to see how the model can drive on new road. The simulator is used and first training data is generate data using behaviour cloning. The image data and actions are used to build and train the model. Different deep learning models CNN, ResNet50, DenseNet 201, VGG16, VGG19 are used for the evaluation purpose. The proposed modified CNN has achieved the better result in comparison of ResNet50, DenseNet 201, VGG16, VGG19. Convolutional Neural Network (ConvNet/CNN) is discussed next in the paper.

A. CONCEPTS INVOLVED

Convolutional Neural Network (ConvNet/CNN): Inspired by the organization of Visual Cortex, CNNs take an image as input, assigns importance to different features/objects of the image, thereby learning about its characteristics. It involves a series of convolution operations applied to the input image [4].

$$Dim(H_1, W_1, D_1) = \left(\frac{H+2Z_p-k_1}{Z_s}+1, \frac{W+2Z_p-k_2}{Z_s}+1, K_D\right) \quad (1)$$

Where H_1,W_1 , and D_1 are height ,width and number of kernels as output calculate by the input tensor H is height, W is width Z_p is padding,$K_1$and K_2 are height and width of kernel, Zs is stride using equation (1).

Flatten layer: In order to pass on information from a 2D network to a fully connected layer, a conversion to lower dimension is required. Flatten layer generally follows a convolution 2D layer and precedes a dense layer as an intermediate converter.

Image Augmentation: Using the technique of Image Augmentation, a huge amount of training data can be generated through different ways of processing or combination of multiple processing, such as random rotation, shifts, shear and flips, etc [19].

Loss function: Loss function is simply a measure of how wrong the predictions of the model are. There are several loss functions including Mean Square Error, Cross Entropy Loss etc. The Mean Square Error is measured as the Mean of the square of the difference between the actual observation and the predicted value.

$$MSE = \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{n} \quad (2)$$

Mathematically, Mean Square Error is defined using the equation (2).

Activation functions: Activation functions are used to convert an input signal into an output signal. These are what empowers the Neural Networks by allowing them to understand linear as well as non-linear relationships in the data. There are several activation functions such as Sigmoid, ReLu [20], Tanh, Softmax etc.

ReLU: ReLU (Rectified Linear Unit) is a half-rectified activation function that is most widely used in convolutional neural networks. Here, both the function and its derivative are monotonic. However, it converts negative values immediately to zero. This decreases the learnability of the model[20].

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ x & for\ x \geq 0 \end{cases} \qquad (3)$$

Equation (3) is used to define ReLU function mathematically.

ELU: ELU (Exponential Linear Unit) activation function is similar to RELU [20] activation function, but helps the model to converge to zero much faster and produce more accurate results [21].

It can be mathematically stated as:

$$R(z) = \begin{cases} z & z > 0 \\ \alpha(e^z - 1) & z \leq 0 \end{cases} \qquad (4)$$

ELU is defined mathematically using equation (4).

Optimisers: Optimisers control learning of the model, by updating the model parameters in response to the output of the loss function.

Adam: There are several optimisers but the most notable one is adam. The reason adam is preferred over classical stochastic gradient descent is that stochastic gradient descent maintains a single learning rate for all the weight updates whereas in adam the learning rate is unique for each weight update and updated separately [22].

Other bechmark deep learning algorithms are used in the experiment for the evaluation purpose are ResNet50, DenseNet 201, VGG16, VGG19.
ResNet 50: ResNet is a powerful Deep Neural Network and won the first place on ImageNet [23] detection, ImageNet [23] localization, COCO (Common Objects in Context) [24] detection and COCO [24] segmentation in ILSVRC and COCO 2015. ResNet stands for Residual Network. Each layer feeds its output to the next layer and also to the layer 2-3 hops away from it. ResNet 50 is used in this paper, which is a 50-layer Residual Network [25]. Figure 1 represents the block diagram of Residual Network.

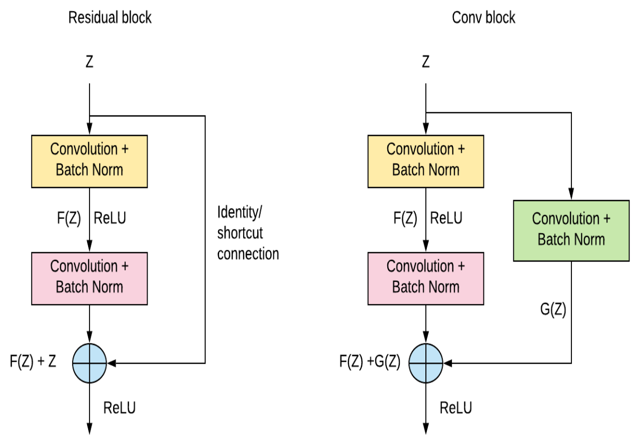

Figure 1: Residual Networks [25]

DenseNet 201: DenseNet stands for Dense Convolutional Network. In DenseNet, each layer receives feature maps from all the previous layers. This makes the network quite compact. DenseNet 201 is a 201 layers deep network that is trained on more than a million images from Imagenet [23][26]. Concept behind DenseNet has been explained in figure 2.

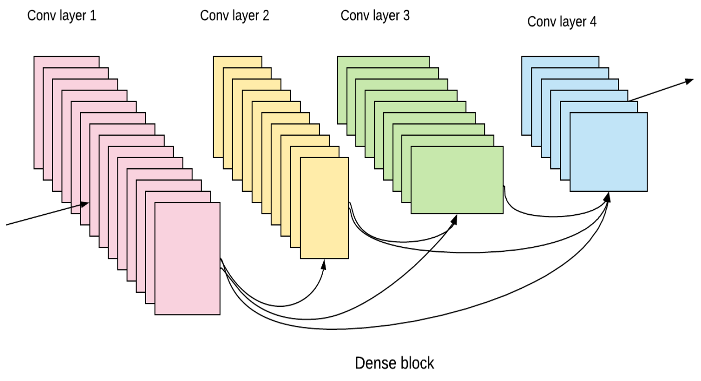

Figure 2: DenseBlock [26]

VGG: VGGNet is invented by Visual Geometry Group(VGG) from University of Oxford and was the 1st runner up in ILSVRC 2014. VGG16 is 16 layers deep while VGG19 is 19 layers deep. The notable characteristic of VGG networks is the use of 3x3 convolution filters in the Convolution layers, thereby allowing the network to be deeper [27].

B. MODEL ARCHITECTURE

We experimented with various different architectures of deep learning and finally adopted to change Nvidia's model proposed in [10]. The architecture of the proposed model with total numbers of parameter is shown in table 1. The model consist of five 2D-Convolutional Layers, one flatten layer and four Dense Layers. The first 2D-Convolutional layer consists of 24 filters, each of size (5, 5) with stride equal to (2, 2). The second layer is again a 2D-Convolutional layer consisting of 36 filters, each of size (5, 5) with stride equal to (2, 2). The third 2D-Convolutional layer consists of 48 filters, each of size (5, 5). The fourth 2D-Convolutional layer consists of 48 filters, each of size (3, 3) and stride equal to the default (1, 1). The fifth layer is the last 2D-Convolutional layer consisting of 64 filters, each of size (3, 3). The output from the last Convolutional Layer is flattened in order to make it compatible with the input size that the Dense layer expects. Next are the 3 Dense layers of size 32, 50 and 10 respectively followed a Dense output layer with 1 node which predicts the Steering Angle corresponding the frame fed to the Model. Table 1 shows the CNN model with its parameters:

TABLE 1: PROPOSED MODEL AND PARAMETERS

LAYER	SIZE	# PARAMETERS
Input	66 x 200 x 3	0
2D Convolutional layer 1	31 x 98 x 24	1824
2D Convolutional layer 2	14 x 47 x 36	21636
2D Convolutional layer 3	5 x 22 x 48	43248
2D Convolutional layer 4	3 x 20 x 48	20784
2D Convolutional layer 5	1 x 18 x 64	27712
Flatten layer	1152	0
Dense layer 1	32	36896
Dense layer 2	50	1650
Dense layer 3	10	510

C TRAINING & VALIDATION

This section consists of training a Deep-Learning model using the dataset gathered by our experiment as human subject, whose behaviour is to be cloned. This dataset was recorded using the Udacity Self-Driving Car Simulator [5]. The simulator consists of two modes:

1. Training Mode - It is used for recording the dataset
2. Autonomous Mode - It is used for testing the model.

The simulator is based on Unity Engine. The car has three cameras attached to it: one in the front, and two at each side of the car. Figure 3 shows the training track and testing track in generated in Udacity simulator.

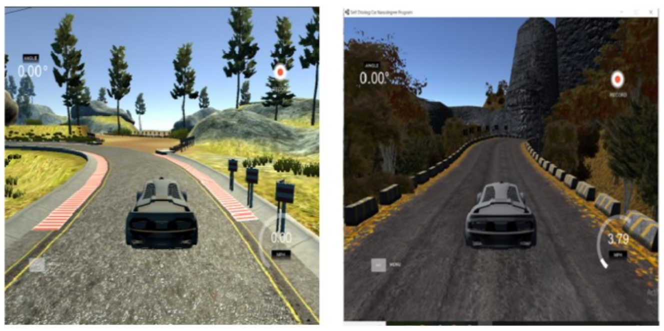

Figure 3: Left image Training track and test track the right image [5].

Path provided in the simulator during the training phase involves a one-way track, which had multiple curvatures, obstacles, bridges, and rivers. Testing track is an unknown mountain area, which has up-down scenarios and non-uniform curvatures.

The first mode, Training Mode, is used to gather the dataset. Therefore, we used this mode to gather our dataset. We drove multiple rounds each in forward as well as in backward direction on the first track. Initially the dataset contained around 4053 frames along with the respective labels. Since, our dataset was highly skewed towards 0 degree, we under sampled it to get a more balanced dataset. We selected a threshold value of 350 i.e. corresponding to each angle we can have a maximum of 350 frames. This threshold was selected keeping in mind the data-imbalance condition as well as to maintain a decent count of the frames available for training our model. The dataset was then split into Train Set and Validation set in the ratio of 4:1.

After having our dataset in place, the next step was to preprocess the data before providing it to CNN [4] based model. To preprocess the data we harnessed the power of OpenCV 2 [6] library.

The preprocessing applied to the input data were cropping the image, converting the Channel from RGB to YUV, applying Gaussian Blur, resizing the image and normalization.

Therefore, each frame, that is passed to the model is first preprocessed as explained in the pseudo-code below:

Algorithm 1 Deep Learning Image Preprocessing

```
Require: img_matrix: Image as a matrix
    1.  Function preprocess(img_matrix)
    2.  img_matrix ← img_matrix[60:137, :, :]
    3.  img_matrix ← YUV(img_matrix)
    4.  img_matrix ← GaussianBlur(img_matrix, (3, 3), 0)
    5.  img_matrix ← resize(img_matrix), (200, 66))
    6.  img_matrix ← img_matrix/255
    7.  Return img_matrix
    8.  End function
```

Conversion of the channel is performed to get the pixels in YUV format. This is performed because YUV channel is more efficient and reduces the bandwidth more than what RGB can capture [28].

In Gaussian Blur, the image is convolved using a Gaussian filter to minimise the noise in the image [19]. Gaussian filter basically uses Gaussian function, which is applied to each pixel in the image.

The Gaussian function for two-dimensions is shown in equation (5):

$$G(x,y) = \frac{e^{-(x^2+y^2)/2\sigma^2}}{2\Pi\sigma^2} \quad (5)$$

Further, normalization was performed to reduce each pixel to a similar data distribution which allows faster convergence of the model.

The formula for Normalization is illustrated in equation (6):

$$input\ para = input\ para/255 \quad (6)$$

Our next step was to leverage the power of a concept known as Image Augmentation, which in the past has proved to be highly beneficial for any CNN [4] based model. Zooming() function is used for producing a new image from the given image by either zooming in or zooming out. Panning() function is used for producing translation in the image. brightness () function is used to multiply each intensity value of pixels in Image to brighten or darken the image. flipping () function flips the image Horizontally as well as the corresponding Steering Angle. In our model we used ELU activation function [21]. Adam optimizer [22] is used with default parameters along with Mean Square Error as the regression loss function. Batch size of 32 was selected. The pseudo-code for the process of Image Augmentation is described in algorithm 2:

Algorithm 2 Deep Learning Image Augmentation

```
Require: img_matrix: Image as a matrix; steer: Steering Angle
1.  Function augment(img_matrix, steer)
2.  number ← Random()
3.  if number > 0.6 then
4.  img_matrix← zooming(img_matrix)
5.  if number > 0.6 then
6.  img_matrix← panning(img_matrix)
7.  if number > 0.6 then
8.  img_matrix← brightness(img_matrix)
9.  if number > 0.6 then
10. img_matrix, steer ← flipping(img_matrix, steer)
11. Return img_matrix, steer
12. End function
```

D. TESTING

After training our model with the proposed CNN architecture with the preprocessed image, the model is connected to the simulator to predict the steering angles in real-time. The frames were pre-processed, and the model passed back the Steering Angles and Throttle values back to the simulator in real-time. We trained our model using the frames obtained from the first track. For testing the model, we allowed the model to drive the car in a track it had never seen before, i.e different track.Throttle value is calculated using the equation (7):

$$Throttle = 1.0 - Speed/SpeedLimit \quad (7)$$

The entire process can be summarized using the workflow given in Figure 4.

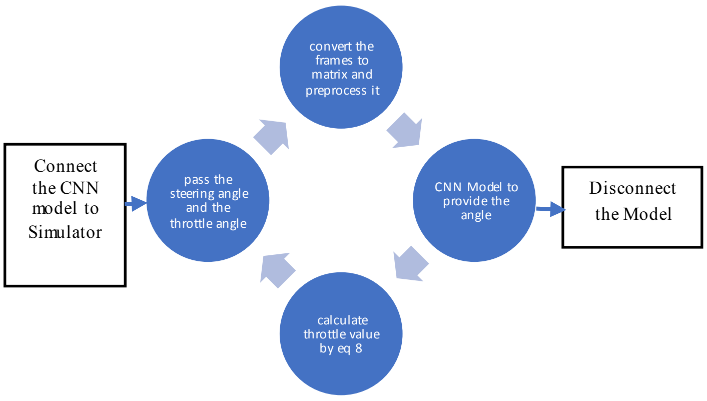
Connect the CNN model to Simulator
pass the steering angle and the throttle angle
convert the frames to matrix and preprocess it
CNN Model to provide the angle
Disconnect the Model
calculate throttle value by eq 8

Figure 4: Workflow diagram of the model with simulator

IV. EXPERIMENTAL RESULTS

The experimental results of proposed model are astonishing. The model is trained on the frames from the first track and tested by driving autonomously on an unseen second track. Around 1500 images were passed to the generator function (for performing image augmentation during the training). The Deep Learning model was trained for 20 epochs using Mean Square Error as loss function and optimiser was selected to be adam optimiser [22]. It took around 3240.2 seconds for the entire process on Kaggle's powerful k80 GPU [29]. The Loss of training and validation is shown in the figure5. The Validation loss keeps on decreasing and both the validation & Training loss seems to decrease as the epoch count increases and converges at around 17 epoch, which is a good sign and indicates the model has learnt the correlations well and has not over-fit on the training set. The least MSE loss obtained is 0.029 on the validation set. The car remained stable for most of the duration and drove with perfection. This approach is indeed a good method for implementing Self-Driving Car in simulation. It requires no separate feature extraction like identification of the edges of road or identification of the lanes etc.

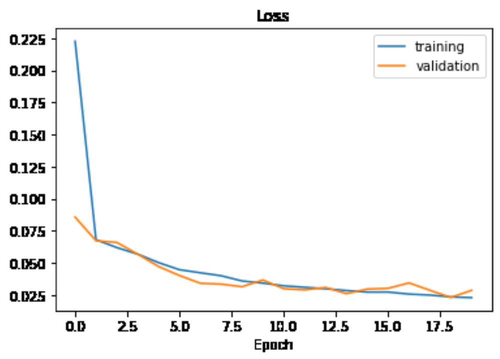

Figure 5: Loss vs. Epoch plot. The graph can be seen converging at around 17th epoch.

Further, we experimented with different activation functions like ReLU [20], Leaky ReLU [20] and ELU [21], but ELU [21] seemed to perform better than the rest. So, ELU is used in the experiment. Further standard models like Resnet 50 [25], DenseNet 201 [26], VGG16 [27]and VGG19 [27] as feature extractors followed by 1 or 2 dense layers are also trained and tested. The comparative analysis of the proposed model with other benchmark models is shown in table 2. The Mean Square Error is measured as the difference between the actual observation and the predicted value for all the models. The proposed systems have limitations as the proposed model is trained with limited data on certain tracks. For real life we require no error and potential for working in different weather condition and situations.

TABLE 2: COMPARATIVE RESULTS ANALYSIS OF MODELS

MODEL	LOSS
Resnet50	**0.03109**
DenseNet 201	**0.02590**
VGG16	**0.05303**
VGG19	**0.03153**
Proposed Model	**0.02914**

V. CONCLUSION

In this paper, we presented an approach for building of self-driving car. This approach is based on the concept of Behavioural Cloning. This approach is end-to-end and doesn't require any separate manual work like feature extraction or connecting different modules for efficient working. Udacity self-driving car simulator is used for simulation and various standard models like Resnet 50, DenseNet 201, and VGG19 for the comparison. The proposed model is Convolution based with five 2D-Convolutional Layers, one flatten layer and four Dense Layers. Compared to other deep learning models the proposed model has outperformed. The work presented in this paper can be realized in real-world to build vehicles capable of driving autonomously. As future work, additional training data of real tracks in various weather conditions and situations will be used to increase the robustness of our system.

REFERENCES

[1] "10 Million Self-Driving Cars Will Hit The Road By 2020 -- Here's How To Profit." .

[2] W. Society for International Engineers, "SAE International Releases Updated Visual Chart for Its 'Levels of Driving Automation' Standard for Self-Driving Vehicles," *SAE.org News*, 2018. .

[3] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," in *IJCAI International Joint Conference on Artificial Intelligence*, 2018, vol. 2018-July, pp. 4950–4957, doi: 10.24963/ijcai.2018/687.

[4] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," pp. 1–11, 2015.

[5] "GitHub - udacity/self-driving-car-sim: A self-driving car simulator built with Unity." .

[6] "OpenCV." .

[7] H. Fujiyoshi, T. Hirakawa, and T. Yamashita, "Deep learning-based image recognition for autonomous driving," *IATSS Research*. Elsevier B.V., Dec. 2019, doi: 10.1016/j.iatssr.2019.11.008.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, doi: 10.1109/TPAMI.2016.2577031.

[9] R. Kulkarni, S. Dhavalikar, and S. Bangar, "Traffic Light Detection and Recognition for Self Driving Cars Using Deep Learning," *Proc. - 2018 4th Int. Conf. Comput. Commun. Control Autom. ICCUBEA 2018*, pp. 1–4, 2019, doi: 10.1109/ICCUBEA.2018.8697819.

[10] M. Bojarski *et al.*, "End to End Learning for Self-Driving Cars," Apr. 2016.

[11] A. K. Jain, "Working model of Self-driving car using Convolutional Neural Network, Raspberry Pi and Arduino," in *Proceedings of the 2nd International Conference on Electronics, Communication and Aerospace Technology, ICECA 2018*, Sep. 2018, pp. 1630–1635, doi: 10.1109/ICECA.2018.8474620.

[12] J. Kim, G. Lim, Y. Kim, B. Kim, and C. Bae, "Deep Learning Algorithm using Virtual Environment Data for Self-driving Car," in *1st International Conference on Artificial Intelligence in Information and Communication, ICAIIC 2019*, Mar. 2019, pp. 444–448, doi: 10.1109/ICAIIC.2019.8669037.

[13] Y. Kang, H. Yin, and C. Berger, "Test Your Self-Driving Algorithm: An Overview of Publicly Available Driving Datasets and Virtual Testing Environments," *IEEE Trans. Intell. Veh.*, vol. 4, no. 2, pp. 171–185, Mar. 2019, doi: 10.1109/tiv.2018.2886678.

[14] F. Yu *et al.*, "BDD100K: A Diverse Driving Video Database with Scalable Annotation Tooling," May 2018.

[15] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognit. Lett.*, vol. 30, no. 2, pp. 88–97, Jan. 2009, doi: 10.1016/j.patrec.2008.04.005.

[16] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles," 2018, pp. 621–635.

[17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," Nov. 2017.

[18] B. Wymann, C. Dimitrakakis, A. Sumner, E. Espié, and C. Guionneau, "TORCS: The open racing car simulator," 2015.

[19] E. S. Gedraite and M. Hadad, "Investigation on the effect of a Gaussian Blur in image filtering and segmentation," *Proc. Elmar - Int. Symp. Electron. Mar.*, no. August, pp. 393–396, 2011.

[20] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation Functions: Comparison of trends in Practice and Research for Deep Learning," Nov. 2018.

[21] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," Nov. 2015.

[22] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," 2015.

[23] J. Deng, W. Dong, R. Socher, L.-J. Li, Kai Li, and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," Mar. 2010, pp. 248–255, doi: 10.1109/cvpr.2009.5206848.

[24] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," in

Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), May 2014, vol. 8693 LNCS, no. PART 5, pp. 740–755, doi: 10.1007/978-3-319-10602-1_48.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Dec. 2016, vol. 2016-December, pp. 770–778, doi: 10.1109/CVPR.2016.90.

[26] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, Nov. 2017, vol. 2017-January, pp. 2261–2269, doi: 10.1109/CVPR.2017.243.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[28] M. Podpora, G. Paweł, K. Korba´s, and A. Kawala-Janik, "YUV vs RGB-Choosing a Color Space for Human-Machine Interaction," doi: 10.15439/2014F206.

[29] "Kaggle: Your Home for Data Science." .