Autonomous DeepRacer Using Deep Learning

Rishabh Sharma
University at Buffalo, NY, USA.
rs278@buffalo.edu

Vinay Krishna Sudarshana
University at Buffalo, NY,USA
vinaykri@buffalo.edu

Akhil Reddy
University at Buffalo, NY, USA.
apallamr@buffalo.edu

Saurabh Mahindre
University at Buffalo, NY, USA.
smahindr@buffalo.edu

***Abstract*— Autonomous vehicles caught attraction since the first project developed in 1980s. Due to increase in road traffic, accidents increased majorly due to human errors. To overcome this issue, till now various models has been proposed including reinforcement learning and supervised learning which showed good results in highly complex environments. Deep learning is one of the methods which not only showed the promising results but also showed good results in generalization and for new scenarios. Testing the developed models for millions of miles raises safety concerns and testing costs. In this project we investigated lateral control of vehicle in a simulated environment using various deep learning models. To learn the model, data was generated by manually driving the vehicle in a simulated environment in various tracks and deployed on the same environment for testing. In this we were able to successfully deploy the model and drive the vehicle autonomously. This project also demonstrated the strengths and limitations of the deep supervised learning.**

Keywords*— Autonomous vehicles, Deep supervised learning, Simulation, ROS, Neural networks, Artificial Intelligence.

I. Introduction

From decades, due to rise in traffic, road accidents increased substantially due to vehicle failure or human errors. Whereas, human errors are the major cause for these accidents. Vehicle failures can be handled by improved vehicle design or strict traffic rules, on the other hand human errors are subjective and hard to control. This led to rise in study of autonomous vehicles, where a computer can learn the behavior of the environment and drive accordingly with more precision and minimal cost. One of the first projects on autonomous driving was initiated in 1980s by Carnegie Mellon University in structured environments [1]. Since then these projects were promoted by DARPA and perused as a major research field.

To drive a vehicle autonomously, there has been many methods proposed till now i.e. supervised learning, reinforcement learning etc. These methods have their own advantages and disadvantages with respect to each other. Deep supervised learning is one of the methods which showed promising results in the field of autonomous driving. Deep learning not only showed state-of-art results in AVs but also in speech recognition and image recognition [2] [3]. The ability to make a learning model generalized for new environments and environment makes deep learning more reliable than other methods.

Testing and training an algorithm requires millions of miles to drive before deploying which is costly and unsafe. To overcome this, simulated environment can be used to train the algorithms and deploy it on physical environment. Due to complexity of the real environment training on virtual environment and deployment on real world still have the reality gap.

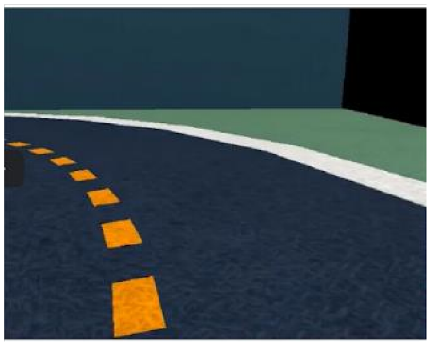


Figure 1.1. Front facing camera. The yellow line is the centre line of the road in the simulated environment whereas white line is the curb of the road.

In this project we trained a vehicle on a simulated environment using supervised deep learning on various tracks and tested on the same. This environment was simulated using ROS and a visualization tool, RViz. We used a single camera on the front of the vehicle to acquire RGB images for training as shown in Figure 1.1. Along with this, we also collected corresponding steering labels data. This data was used to train a deep neural network (NN) for steering angle prediction. The learned model is deployed in the same simulated environment for steering angle prediction which is further published on steering ROS topics to drive the vehicle autonomously on the track.

II. Background

The inception of the Autonomous cars begins in the year 1926 with world's first Radio controlled car was designed and developed "The Car" which were first rolled out in early 20th century, on the streets of the Dayton, Ohio [4]. The deep learning modules that were first drawn attention on Autonomous Driving car technologies dates back to the early 80's and for the highway Driving, which were introduced for having the idea of safe and prominent transport of the mankind which would help in planning [5] [6]. An autonomous car can operate without human control and does not require any human intervention have made certain things clear that autonomous vehicles can sense their local environment, classify different kinds of objects that they detect around them, can interpret sensory information to identify appropriate navigation paths whilst obeying transportation rules [7]. Since then, a constant and exponential work has been contributed towards this field of

Artificial Intelligence (AI). The focus of the autonomous vehicles has been on vision guided systems using LIDAR, RADAR, GPS and Computer Vision applications. These were then developed into the autonomous technologies present in modern cars like adaptive cruise control, lane parking, steer assist etc. And, in the future, we will see the fully autonomous cars will be a reality, based on looking at the present trends in the Autonomous vehicles industry giants like Google, Tesla, Baidu etc. Due to rise in traffic no. of accidents has risen and leading to over 3000 deaths, per day [8]. They have also stated that if certain measures, that could curb them, are not taken then there would be more than 2.4 million accidents every year. Which is why many companies and AI researchers are investing and contributing to this Autonomous vehicle field of AI, to develop and prototype the AV's for a safer, prominent and efficient transportation along with certain added advantages like the government spending's would be at a very minimally catered at this disposal.

We all know that the deep learning techniques have made rapid progress in conditional image generation. While Deep convolutional Neural Networks have greatly improved the ability for computers to see and understand images in recent years, thus the deep learning has started to see its development, importance, use in the field of autonomous vehicles in an exponential manner as it was consistently proving and surprising everyone with its results. The first Autonomous land vehicle that used Deep learning to follow the road was a simple 3layered backpropagation network, takes images from a camera and a laser range finder as input and produces as output the direction the vehicle should travel in order to follow the road [9]. Since then many researches have contributed and have proven certain state-of-the-art results. The control perspective of the autonomous vehicles was then categorized into two tasks "Lateral Control" "Longitudinal Control" where the former deals with the steering of the vehicles and the later deals with the throttle and brakes of the vehicle. The lateral control is responsible for the lane kept-assist, lane changes and collision avoidance maneuvers and similarly for the longitudinal control, it is the sole responsible for maintaining a safe distance from other vehicles and also the acceleration, braking so as to maintain the desirable velocity on the roads [10]. Our implementation certainly deals only with the "lateral control" similar to the method introduced by NVIDIA in which they used images as an input from three cameras that were placed in the front of the vehicle [11]. These images are labelled along with the steering angles in radians that were captured along with the road views to train the network. Whereas in our method we used only one camera to train the NN for lateral control. Their architecture consisted of Input layer, Normalization layer, 5 convolutional layers, 3 fully connected layers and an output layer.

Our implementation is similar to the paper cited above (NVIDIA) with certain modifications in the architecture and the data collection. In our method we used dropouts and maxpooling layers which is implemented to avoid overfitting and reduce computational cost. We used the *AWS robomaker sample Deep Racer application* as our platform for training and testing [12]. Our network used only one camera to capture the images where respective steering angles were recorded by monitoring Ackerman ROS topics while we were manually driving the car. The complete details about collecting data, training, and model's performance during testing is discussed in the later part of the paper.

III. METHODOLOGY

In this work, we make use of the simulation environment in the *AWS Robomaker sample application for Deep Racer*. Although the original objective of the Deep racer application is to train a reinforcement learning (RL) model to drive a car around a track, the application also provides the option to locally run the simulation. We use this functionality so that we can use the same application to use Deep Learning instead of Reinforcement Learning. For our work, what we need is a simulation environment with track and a car on it which has a front-facing camera that gives us the video feed of the part of the environment that the car sees at any point. We plan to make use of these images as the data and the steering input to the car as the supervision to train a Deep Learning model to drive a car around a track.

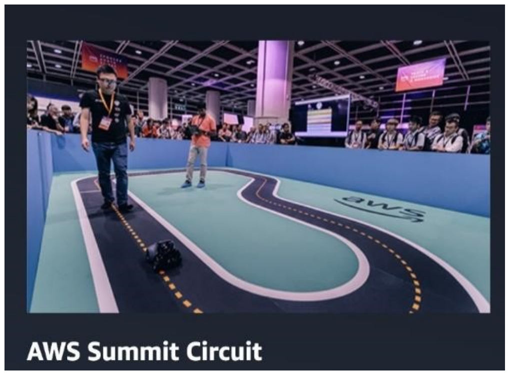


Figure 3.1. AWS Summit circuit deepracer competition in 2018.

In *AWS Robomake*r package there are three different tracks in which we can train a model. These three tracks include, *easy_track*, *medium_track* and *hard_track*. Because easy track was just a patch of road, it cannot be used for training or testing the model whereas *medium_track* and *hard_track* is a full closed-loop racing track in which the car can be driven infinitely. Because *medium_track* is unidirectional, training requires to drive the vehicle in both directions for full autonomy. To avoid that we used *hard_track* which is the toughest track in the package available to train our model. It's a replica of the real track that was used in the AWS summer circuit 2018 as shown in Figure 3.1. Figure 3.2 shows the *hard_track* that we use as part of our work. This track has ridges on both sides of the track i.e. white lines, if the car goes out of either of the ridges on the sides, returning to the track is hard.

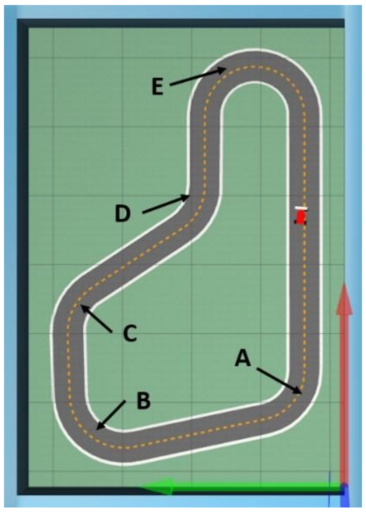


Figure 3.2. Top view of the hard_track in AWS Robomaker deepracer [12]. A, B, C, D and E are the turns with different difficulty level. Grey color shows the road with white borders. Blue boundary of the image represents the wall in the simulation environment.

There are walls on all sides of the track which bound the car within a rectangular space. This track has total five turns i.e. A, B, C, D and E as shown in Figure 3.2. with variable difficulty level. E is the steepest turn in the whole track which makes it the hardest, whereas A, B, C and D are slightly less difficult. The simulation environment uses Gazebo and ROS for rendering and control of the car and its environment respectively.

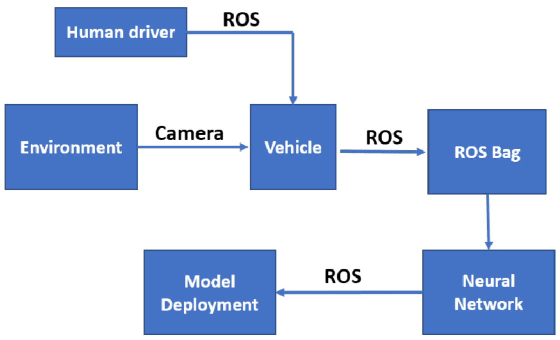


Figure 3.3. Project's Methodology. Vehicle is driven (while collecting training data) by the human driver in the simulated environment in which images from the front camera with it's corresponding steering label is colleted using rosbag. This data is futher fed inside the NN to train a model which is further deployed in the same environment.

We wrote a ROS node that we can use to control the car using keyboard. In the ROS node we used the keystroke input and mapped it to the corresponding steering angle and publish it to a ROS topic which the application uses to render the changes to the car's steering angle. The front facing is also published by the application in a ROS topic. We used that ROS node to manually drive around the track for 1 hour and used ROS bag to record the front-facing images and the steering angle inputs. This is the data that we will be using to train the Deep Learning model as shown in Figure 3.3. The synchronization of the image with the steering angle that was given as input was achieved by changing the frequencies at which the data was being published in the ROS topics.

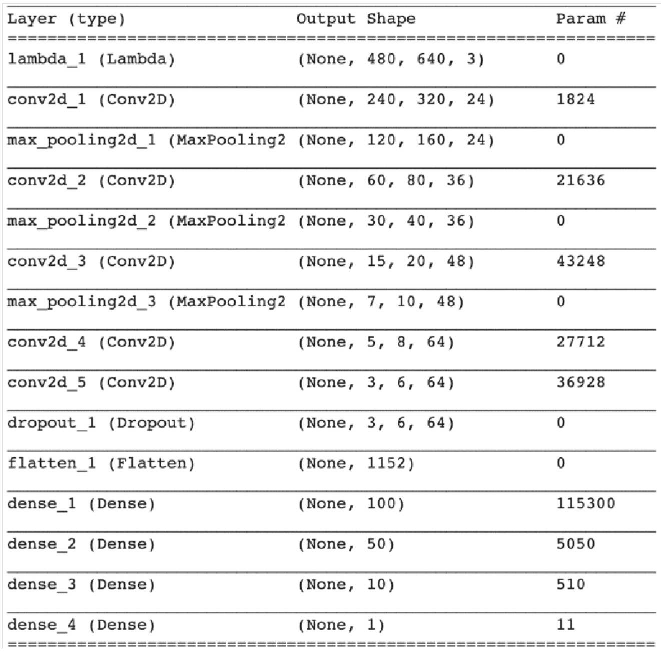

```
Layer (type)                 Output Shape              Param #
=================================================================
lambda_1 (Lambda)            (None, 480, 640, 3)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 240, 320, 24)      1824
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 120, 160, 24)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 60, 80, 36)        21636
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 30, 40, 36)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 20, 48)        43248
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 7, 10, 48)         0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 5, 8, 64)          27712
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 6, 64)          36928
_________________________________________________________________
dropout_1 (Dropout)          (None, 3, 6, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 100)               115300
_________________________________________________________________
dense_2 (Dense)              (None, 50)                5050
_________________________________________________________________
dense_3 (Dense)              (None, 10)                510
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 11
=================================================================
```

Figure 3.4. Neural network architecture with five convolutional layers, 3 maxpooling and one dropout.

After collecting data from the ROS bag, we had a training set of RGB images as seen by the race vehicle and their corresponding steering values. We trained a deep neural network to be able to learn from these annotated images and predict steering values. We used a deep neural network with 5 convolution layers combined with max-pooling layers and dropout normalization as shown in Figure 3.4. For model evaluation, we divided the dataset of images into a training and validation set of 80% and 20%. The dataset consisted of images of driving in a clockwise and counterclockwise direction on the racetrack. In order to remove any underlying bias due to the different number of samples for right and left turns, we randomly sampled an equal number of images from both scenarios. The evaluation metric used was mean squared error (MSE) for steering values and Adam optimizer for gradient descent was used for training. Before deploying the model into the environment we visually verified the predicted steering angles for a set of images from the validation data set. The model weights were then stored in an H5 file format and then deployed to the race vehicle to be used to perform real-time prediction of steering angles from the camera feed on the same track.

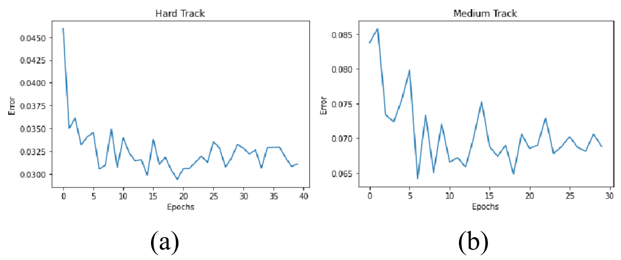
IV. Results & Discussion

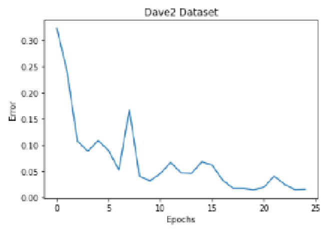
The performance of models over different epochs can be seen in Figure 4.1. As seen, the training loss keeps reducing even for large numbers of epochs but the validation loss plateaus after 5 epochs.

Table 1. Performance of three different models with different epochs. Green row represents our model and optimized epochs which showed best results.

Model	Epochs	Data	Train Error	Test Error
5 Conv. Layers (Nvidia)	3	Simulation (Hard track)	0.0285	0.0361
5 Conv. Layers (Nvidia)	7	Simulation (Hard track)	0.0104	0.0305
3 Conv. Layers (UC Berkeley)	7	Simulation (Hard Track)	0.03	0.09
17 epochs	17	Dave-2 Dataset (2700 Images)	0.0334	0.0332

We also observed that even after the validation error stopped decreasing the model was still learning to make difficult maneuvers like steep U-turns. This can be seen from the fact that from 5 epochs to 12 epochs the validation error decreased by 0.001 but the model was unable to perform steep U-turns at 5 epochs but did so perfectly at 12 epochs. This could suggest that MSE might not be the best metric to optimize the model. In order to get an objective evaluation of the location of the race car while it is autonomous we recorded the distance of the car from the center of the track and observed the histogram of deviations from the center. The observations suggest that the racecar did not deviate far from the center for the entirety of the race track.





(c)

Figure 4.1. Performance of model over epochs for (a) Hard Track (b) Medium Track and (c) Dave2 (Real-world dataset)

To evaluate our results, we compared the results from our architecture with another shallower architecture with 3 convolutional layers, combined with max-pooling layers and no dropout normalization as seen in Figure 4.1 [13].

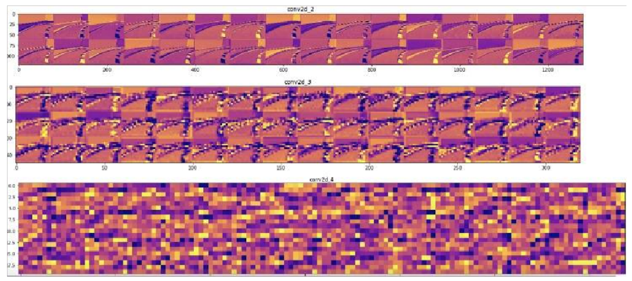


Figure 4.2. Feature maps of the first few convolutional layers. These reveal the important road features learnt.

The comparison of MSE values and real-time results can be seen in Table 1. The results on a real-world driving dataset (Dave-2) also showed good RMSE values using the network trained on our simulated images.

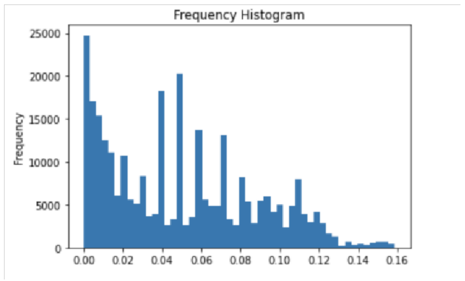


Figure 4.3. Histogram of the distance between vehicle and center line of the track during deployment.

Visualizing the few layers revealed that the first two layers learn track outlines and centre lines from the images as shown in Figure 4.2. We could remove weights from the final layers and utilize the weights from the weights learned from the first two layers to perform transfer learning and use the same network in a different environment or track. Further to quantify the results we also calculated the distance between the vehicle and the center line during deployment as shown in Figure 4.3.

V. Conclusion & Future work

The simulation environment i.e. *AWS robomaker sample Deep Racer application* proved to be a viable interface for building a Deep Learning autonomous driving model to train and test upon even though it was originally developed for Reinforcement Learning. The simulation environment is costeffective and safe because it avoids cost which is associated with data collection and deployment in the real world. Having said that, we do acknowledge the fact that there is a reality-gap which does not include the complexity of the real environment.

Deep Learning showed promising results in predicting steering angles using a single front-facing camera and training data which was obtained by driving the car manually for as low as 60 minutes. For the model's architecture A, which is modified from the model used in [11] it was observed that the car learns to go around the track while staying inside the track. As compared to other models and human driving, the car was able to turn successfully on the steepest curve E as shown in Figure 3.2.

Future work involves end-to-end longitudinal and lateral control which includes obstacle avoidance, lane changing, near collision avoidance and considering dynamic objects like pedestrians and other vehicles. The learnt model in the simulation environment can further be deployed on a Tamiya TT02 remote control car. Further model can be tested for generalization and new scenarios.

REFERENCES

[1] C. Thorpe, M. Herbert, T. Kanade, and S. Shafter, "Toward autonomous driving: the cmu navlab. ii. architecture and systems," *IEEE expert,* vol. 6, no. 4, pp. 44-52, 1991.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097-1105.

[3] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine,* vol. 29, no. 6, pp. 82-97, 2012.

[4] L. Torres-Quevedo, &quot;The RCA radio-controlled car,&quot; 1904.

[5] M. H. Charles Thorpe, Takeo Kanade, and Steven Shafer, *Towards Autonomous Driving*. 1991.

[6] E. D. D. a. A. Zapp, *Autonomous high speed road vehicle guidance by computer vision1,”*. 1987.

[7] M. Campbell, Magnus Egerstedt, Jonathan P., *Autonomous driving in urban environments: approaches, lessons and challenges.* 2010.

[8] P. Deshpande, ""Road Safety and Accident Prevention in India: A review," 2014.

[9] A. D. A. Pomerleau, *An autonomous land vehicle in a neural network*. 1988.

[10] R. B. Sampo Kuutti, Yaochu Jin, Phil Barber, and Saber Fallah, "A Survey of Deep Learning Applications to Autonomous Vehicle Control," 2019.

[11] "End to End Learning for Self-Driving Cars," 2016.

[12] B. Balaji *et al.*, "DeepRacer: Educational Autonomous Racing Platform for Experimentation with Sim2Real Reinforcement Learning," *arXiv preprint arXiv:1911.01562,* 2019.

[13] V. Rausch, A. Hansen, E. Solowjow, C. Liu, E. Kreuzer, and J. K. Hedrick, "Learning a deep neural net policy for end-to-end control of autonomous vehicles," in *2017 American Control Conference (ACC)*, 2017: IEEE, pp. 4914-4919.