Behavioral Cloning for Autonomous Cars

Khawaja Ghulam Alamdar
School of Science & Engineering
Habib University - Karachi, Pakistan
ka05058@st.habib.edu.pk

Muhammad Ammar Khan
School of Science & Engineering
Habib University - Karachi, Pakistan
mk04366@st.habib.edu.pk

Aiman Junaid
School of Science & Engineering
Habib University - Karachi, Pakistan
aj05161@st.habib.edu.pk

***Abstract*—This paper proposes two techniques to enhance performance of self driving cars on the Udacity Self-driving dataset by including the temporal information of dataset. In the first part we propose using a moving average filter to make the steering angles in dataset smoother and remove random fluctuations that make our model learn on not so predictable data. In the second part we apply LSTM layers to the SOTA NVIDIA model, to observe what size of input images to LSTM improves the performance of NVIDIA, if it does at all.**

***Index Terms*—LSTM, Autonomous cars, Self Driving**

I. Introduction

With the growing phase of autonomous learning, self-driving car is emerging as a center of focus for automobile industries. Behavioral cloning is a method by which human sub cognitive skills can be captured and reproduced in a computer program. In recent years, several deep learning-based behavioral cloning approaches have been developed in the context of self-driving cars. The current trend of the automotive industry combined with research by the major tech companies has proved that self-driving vehicles are the future. This paper will focus on how an end-to-end model is implemented. This paper focuses on including the temporal information into our model as well and to analyze its results.
.

A. Evolution of Self-Driving Cars

The rapid development in the field of Artificial Intelligence (AI) has promoted the progress of self-driving cars. The market demand and economic value of self-driving cars are increasingly prominent[1]. At present, more and more enterprises and scientific research institutions have invested in this field. Google, Tesla, Apple, Nissan, Audi, General Motors, BMW, Ford, Honda, Toyota, Mercedes, NVIDIA, and Volkswagen have participated in the research and development of self-driving cars [2].

Google is one of the finest leaders in self-driving cars, based on its solid foundation in artificial intelligence. Google tested two self driving cars on the road in June 2015. So far with the development, Google vehicles have accumulated more than 3.2 million km of tests, becoming the closest to the actual use. Tesla is another company which has made significant progress in the field of self-driving cars.It was the first company to devote self-driving technology to production. Followed by the Tesla models series, its "auto-pilot" technology has made major breakthroughs in recent years.
There are several other Car and Internet companies like Zenuity, established by the collaboration of Sweden, Volvo and Autoliv which is committed to the security of self-driving cars and have devoted their researches towards this growing field.[3]

II. Literature Review

A. Deep Learning Methods for Self-Driving Cars

1) Convolutional Neural Networks (CNN)

A CNN is a kind of artificial neural network utilized in image recognition and processing that is explicitly intended to deal with pixel information. CNN has their ”neurons” orchestrated more like those of the frontal lobe of a human brain, the zone liable for preparing visual boosts in people. Therefore, we would be able to learn features much faster inside a picture and be able to make decisions based on that as well.[4] The CNN has yielded outstanding results in computer image and general image classification tasks recently. Because many self-driving cars technologies rely on image feature representation, they can be easily realized based on CNNs, such as obstacle detection, scene classification, and lane recognition.

2) NVIDIA CNN Architecture

DARPA Autonomous Vehicle (DAVE) established the potential of end-to-end learning and was used to validate the DARPA Learning Applied to Ground Robots (LAGR) program. However, DAVE's success was not reliable enough to support a full alternative to more modular approaches to off-road driving: its average space between crashes was about 20 m. To cater this problem, NVIDIA proposed a new application which was build on DAVE and created a strong system that is capable of learning the whole task of line and path monitoring, without the need for manual decomposition, marking, semantic abstraction, path planning, and control.[5]

This CNN arose from the motivation to achieve a robust system for autonomous driving on public roads. The NVIDIA model is the first empirical demonstrated that CNN are capable of learning the whole task of tracking lines and path without manual marking. This model is also GPU accelerated so therefore, does much faster processing but at the cost of expense.

3) Recurrent Neural Networks (RNNs) and the Need for Long Short Term Memory (LSTM) Networks

   A normal CNN generates output solely based on the input images. The current output is only dependent on the current input frame. However, the frames we're inputting is a time series data, and the images are related to each other. Every next frame is some transformation of the previous frame. When we apply CNNs to these frames, we lose the temporal information that could, hypothetically help us improve our results.
   To capture this temporal information we can use Recurrent Neural Networks (RNNs). Compared to simple feedforward networks, RNNs area capable of learning complex temporal dynamics on sequence data [6].
   One problem with RNN is that it cannot connect information too far due to gradient vanishing [7,8]. This issue can be overcomes by using LSTM units for a RNN network [9]. "LSTM has a special cell state, served as a conveyor belt that could allow information to flow without many interactions [7]." Put simply, it has gates that allow it to figure out when to remember some temporal information and when to forget.
   Now our input is still images so we can't deploy LSTM directly. We could flatten the images but then we would lose spatial information. So for this purpose we use 3D CNN which is the same as 2D convolution but also has a third temporal direction along which it convolves [7]. Then we can simply flatten the data, feed it into LSTM and then into fully connected layers to get the steering angle. [7] reported that its results actually deteriorated with LSTM layers to some extent instead of improving.
   An earlier paper [10] however, report very promising results using CNN+LSTM hybrids as suggested by [7].It claimed that it improved steering erro (RMSE) by 35 percent over recent methods (2017 and before), and led to a more stable steering by 87 percent [10].
   A similar study was conducted by Gu et al [11], where their proposed model outperformed several baseline models in driving action prediction on the Waymo Open Dataset in 2020 [11].
   Using all the evidence presented, it seems a good approach to utilize the time information as well on self driving to improve the results.

This paper provides a analysis on the implementation, comparison and results of two architectures namely NVIDIA Architecture and LSTM-CNN Hybrid model in the second part of section III.

III. Proposed Methodology

A. Filtering to Smoothen angles

Our first task relates to pre-possessing the data-set. As shown in the figure below, the steering angles are heavily biased towards the zero steering angle.

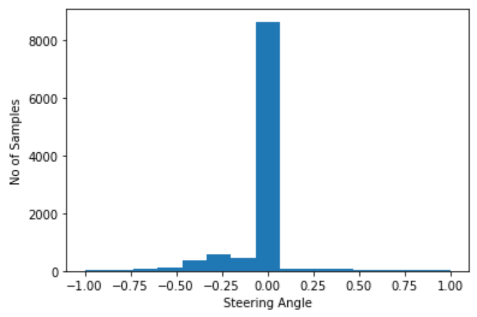


Figure 1. Histogram of Steering Angles

According to our literature review, it will take 100s of epochs to learn. Another approach commonly used is data augmentation. Using data augmentation, we can augment the frames with less representation by varying its brightness, shadow, blurring et cetra. This makes our steering angle distribution more like Gaussian resulting in less bias. However, as we did not have enough training computing power, we could not afford to have more data. So what we did is that analyzed the dataset further. What we see is a lot of random fluctuations between steering angles as the dataset was trained driven using keyboard or joystick. Now the main issue between human driving a real car and driving in simulations is that human driving a car using a steering wheel is continuous. That is , he continuously adjusts the steering angle according to the road curvature. The Udacity simulator track has mostly curvature and very rarely does it have perfectly straight roads.

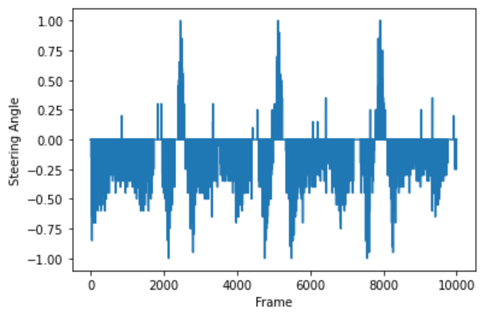


Figure 2. Steering Angle over frames(time)

Now compare this with a driving a car in game. You don't continuously press turning button when roads curvature arrive, but you press button discretely to adjust course and stay on the road. Thus, similar images have mostly zeros and some angles. This makes our model heavily biased towards steering angle of 0 or moving straight.

Now what we propose as an alternative to data augmentation, is smoothing the steering angles. We achieve this by a moving average filter that acts as a low pass filter and removes the highly fluctuating angles. We have used a filter with ’l’ parameters/length and all parameters as 1 for now. We can try using Gaussian and other filter as proposed in V-A.

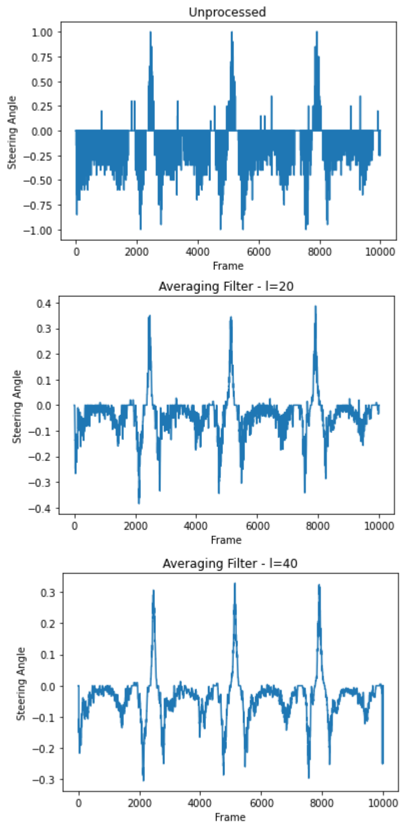


Figure 3. Affect of Moving Average Filter

Here, the issue is that since we’re changing input labels when we change our length of filter, our input isn’t constant hence we can’t compare the loss of different models with different values of ’l’ filters. So we can only analytically observe how well the car is driving using the Simulator.

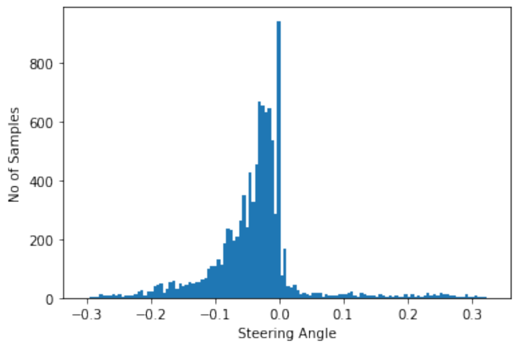


Figure 4. Histogram After Moving Average Filter

As you can see in the histogram, the steering angle distribution becomes unbiased. However, our right turns are still in low quantity. This could improve by having larger data-set. We could not do so because of limited computing power.

B. NVIDIA Architecture

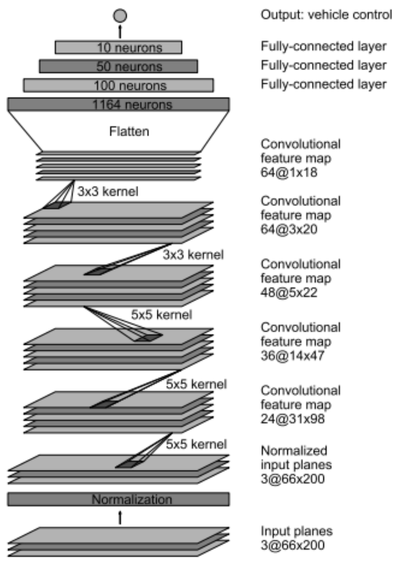


Figure 5. NVIDIA Architecture

This section outlines the network architecture released by NVIDIA for self-driving cars as shown in Fig 5.The network consists of 9 layers, including a normalization layer, 5 convolutional layers and 3 fully connected layers. The input image is split into YUV planes and passed to the network.The first layer of the network performs image normalization.Image preprocessing normalizes the images before feeding it into the architecture and therefore this layer has been omitted from the actual implementation. The five convolutional layers were designed to perform feature extraction. We use strided convolutions in the first three convolutional layers with a 2×2 stride and a 5×5 kernel and a non-strided convolution with a 3×3 kernel size in the last two convolutional layers.

We follow the five convolutional layers with three fully connected layers which then outputs the steering angles.In order to reduce over-fitting Dropout layer(0.2) was used.An Adam optimizer was used for optimization with a learning rate of 0.0002. [4]

C. LSTM-CNN Hybrid Control

The second part of our study focuses on application of LSTM layers on SOTA models. For our case we chose the NVIDIA model. The problem with using normal CNNs, as explained in II-B,is that they do not cater the time information, which is very important as the frames we get as input data are a time series, so each upcoming frame is somehow a new version of previous frame and is hence related. To take this temporal info into account, we used LSTM based RNN. We just inserted a LSTM layer in NVIDIA model, between the CNN layers and Dense layers.

As discussed in II-B, we need 3D convolution layers to be compatible with LSTM layers. This is especially simple in keras. We used the TimeDistributed() function to add a temporal component to out 2D conv layers and make it essentially act as a 3D conv layer. [12]

Note that for this network, we have to pre-process the input into time divided batches, so we need to change input size from (B,H,W,C) to (B,N,H,W,C), where N is a hyper-parameter denoting the number of consecutive images that are input to the LSTM network in one go. We ran our code for different values of N with fixed l (parameter of averaging filter for pre-processing of input images). We then compared our MSE loss for different values of N to the original NVIDIA model we fit on the same dataset. We could not use larger values of N that could've given better comparative results because of limitations on computing power as LSTM networks are time intensive.

Also, note that the steering angles are preprocessed the same way as done in part A using an averaging filter with value l=60, as that worked best as explained next.

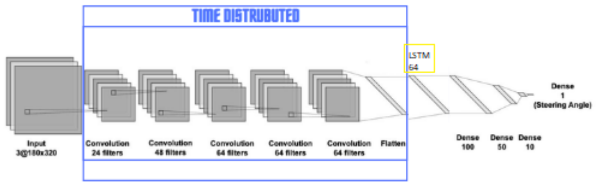


Figure 6. NVIDIA+LSTM Architecture

IV. Results

A. Filtering to Smoothen angles

Since the input to different lengths of filters are different, we can't effectively compare losses between them to determine which works best. The losses only give us the idea that this filtered data is being learned well by model with that particular value of 'l'. The losses at different values of l (filter length) are shown below.

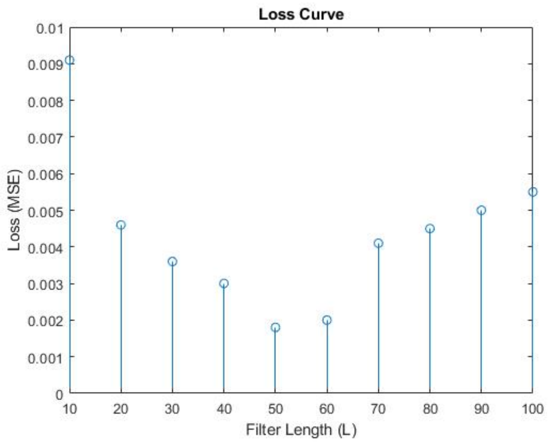


Figure 7. Loss with Different Filter lengths

Analytically, we ran all the models on Udacity Simulator to observe which filter worked the best.

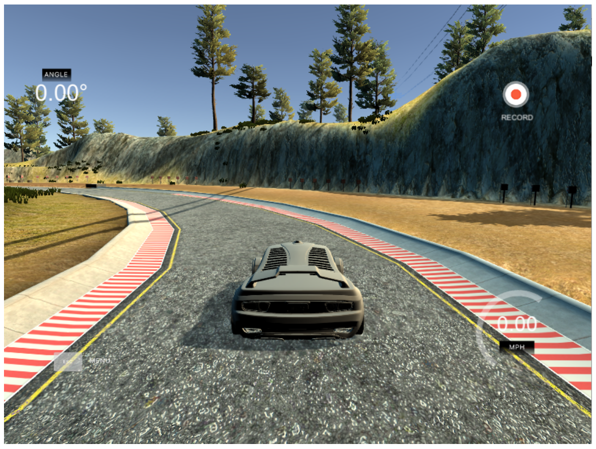


Figure 8. Udacity Simulator Layout

We observed the simulations for filter length (l) between 10 and 100. The simulations were run multiple times for each value of l, and average was taken to calculate the average time a car stays on track without crashing. The results are summarized in the figure below.

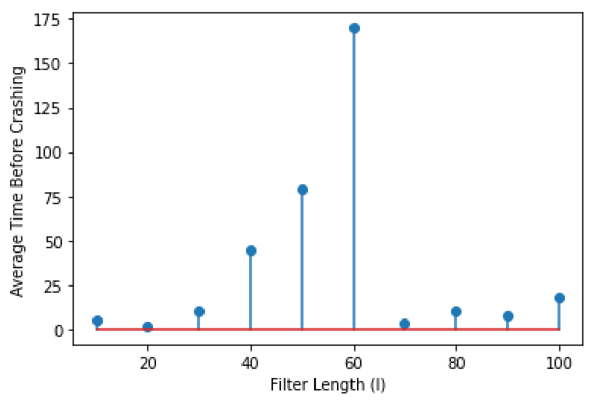


Figure 9. Performance of Car on Udacity Simulator with Different Filter Lengths

As sown, best results were achieved with a length 60 averaging filter, which stayed on track for about 3 minutes on average. The car also turned smoothly without any sharp edges. This is especially impressive given the small amount of dataset (3.5k images) and the amount of epochs (only 10).

B. LSTM-CNN Hybrid Control

The bottleneck we faced while testing different LSTM networks was the high time complexity which did allow us to use more complicated LSTM models like with inputs of a large number of temporal images (the parameter N as described). We could not go above N=5 and N=1 to N=4 gave almost similar losses. The MSE loss for NVIDIA model and NVIDIA-LSTM hybrid is shown in the figure below.

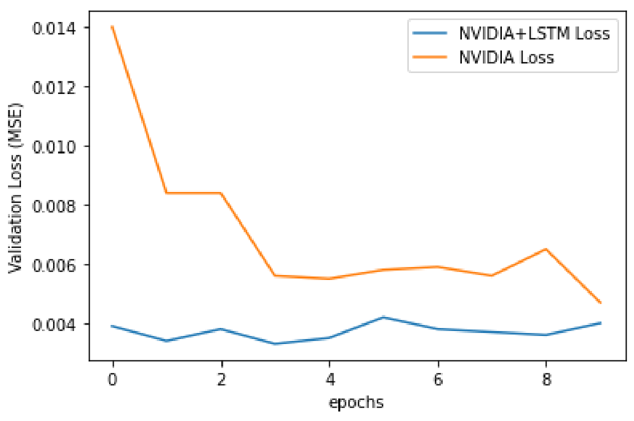


Figure 10. NVIDIA vs LVIDIA+LSTM Loss

Over 10 epochs, the MSE loss of NVIDIA+LSTM model is 2/3rds less than that of the NVIDIA. This shows that our LSTM has improved the performance of the NVIDIA model. Do note that this is when LSTM input size is just 5 temporal images where the time difference between the sequential frames is pretty less. Increasing this number can theoretically improve the results significantly but again due to computational limitations, we were unable to try that.

Another thing to notice here is that NVIDIA+LSTM model's loss stays almost constant and is not really decreasing. This loss could then be further reduced by tuning the hyper-parameter values like number of neurons of LSTM and/or the learning rate of our optimizer which in this case was Adam.

V. Future Potential

A. Gaussian and other Filters to Smoothen angles

In section III-A, we proposed an averaging filter will all coefficients equal to 1. But a more realistic idea could be used of a Gaussian filter with weighted coefficients. Thus it decreases the impact the neighbouring steering angles have on the computation of the current steering angle. The coefficients used by us and Gaussian coefficients (for variance =1) are shown below.

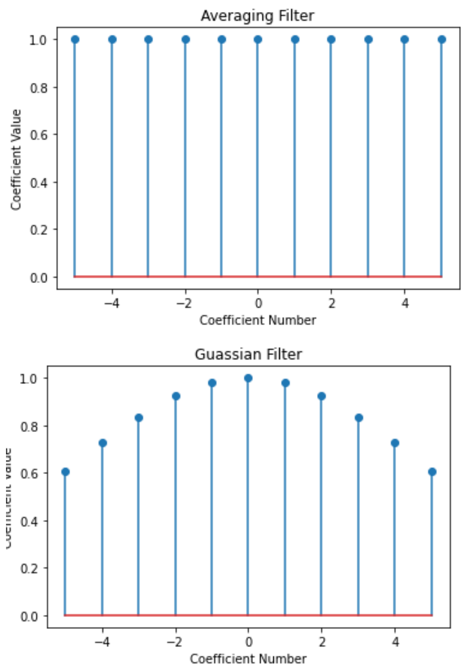
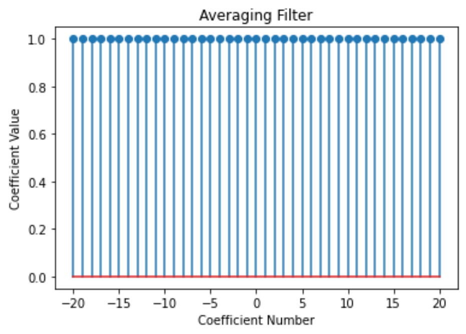


Figure 11. Normal Averaging vs Gaussian Filter Coefficients for l = 11

Similarly, we could vary the value of variance of the Gaussian filter to see which filter performs best on simulation.



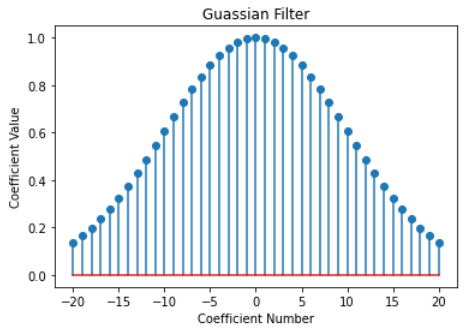


Figure 12. Normal Averaging vs Gaussian Filter Coefficients for l =40, Sigma = 10

We tried training and simulating a Gaussian filtered model and achieved excellent results compared to our previous findings and the car was able to stay on road for more than 10 minutes overall. We also believe that tuning it's hyper-parameters will enable us to see even better results.

B. LSTM-CNN Cooperative Control

[13] suggest an improved method of LSTM-CNN cooperative control. The basic idea is that a car moving in-front of our car is sharing data with us, so we have future info as well. This is simulated in a train environment by taking 2 batches for LSTM, one from our car and one batch from a car T frames ahead of us. [13] showed that it has shown promising results and values of T as a hyper-parameter can be tuned further. They proposed that it works best for a certain value of T and performs less for other values.

This can also help us in the case where the present image is affected by noise or contains less useful information for example if it is burnt by direct sunlight [13]. In these situations, correlation between current frame, the past frames (because of LSTM) and the future frame from the car in-front, can be useful to decide the steering value.

To test the robustness of the algorithm, so that its efficiency remains good and well, we wanted to implement a network that can alter its parameters based on how close T is compared to optimal T so the car can benefit from the car in-front of it to some extend despite the distance of the car from us. We did the preprocessing to simulate the images coming from the car T frames infront, the the processing but our RAM was crashing. Given we take 5 images from present and 5 from future cars, we send 10 images as input to out network. So in essence, you can say we have 10 times the original input image frames. Thus, we were not properly able to experiment with this technique but it seems an area with great potential for research.

C. GANs for Unsupervised Learning

Since we encountered numerous problems regarding the data set, we can easily use the generator to augment the images using advanced GANs and extract dataset for different road conditions (e.g., snowy, rainy) and our network can be easily trained on such conditions as well. Therefore, this idea can be used in two ways: either to use the generator's feature to enhance our current network or use the discriminator to implement semi-supervised learning.
The emphasis is placed on the fact that, while collecting data from the simulator is relatively simple, when it comes to the practical design of autonomous vehicles, this further generation of dataset that can prove to be super-realistic can be groundbreaking in the sense that a small dataset can be pre-processed in this manner and an authentic dataset can be generated. Augmentation techniques may allow researchers to generate images with certain road conditions that are difficult for them to generate [14] thus, GAN can be used to train a network that is universal. Object detection may also enable stop signals, automobiles and pedestrians, road breakage/accidents, and speed breakers will all be easily identified by autonomous automobiles. This will ensure that the envisaged network can adequately cater to unusual events. Comparisons of generated images and the original one can be seen from figure below.

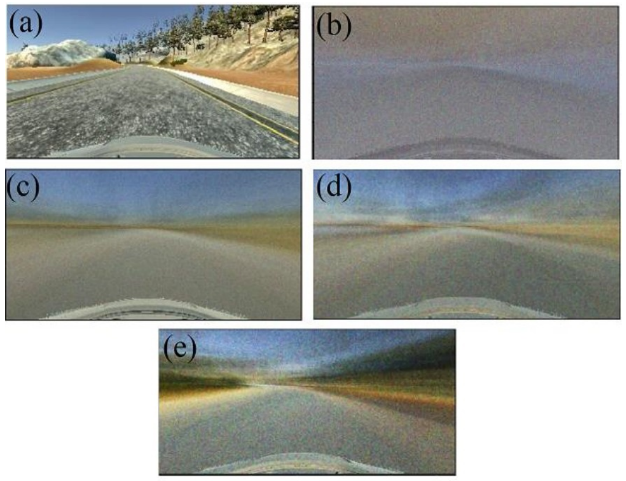


Figure 13. (a) Original Image, (b) Generator image (2000 epochs), (c) Generator image (5000 epochs) (d) Generator image (7500 epochs) (e) Generator image (10000 epochs)

Within the ten-thousand epochs, we can see that the imitation at the car's bonnet is pretty strong, and that the environment is gradually building with development in the road and off-road texture, as we can easily tell in (e) as to which section perfectly depicts the road and which does not.

Reiterating that this GAN was rather straightforward, we can easily tune the hyper-parameters to be more complicated, and you'll get a better overall simulation of the scenario, but at the cost of additional time-space complexity.

VI. Conclusion

Throughout our research, it has been observed that in the case of behavioral cloning for driving a car, we are obliged to be restricted by its constrain-ted data-set and even in practical scenario's, we would observe a car to stay at a steering angle for a longer period and thus, we would need such drastic pre-processing on the labels of the data-set and due to this reason, we believe that the research on this particular topic is very unique compared to other Computer Vision research topics. Furthermore, we were able to convince our results through the simulations on the aspect that pre-processing labels would not affect the overall result of the autonomous car but rather (in our case) result in smoother turns. We were also able to theoretically convince that LSTM could prove to be a breakthrough in the future and can be thought through if communications between traveling cars can be a possibility with the development of security and hardware with time. Overall, the results we obtained and the proposed techniques worked out great for us and we will be looking onto implementing the future aspects of this project later in our life.

References

[1]"Economic Effects of Automated Vehicles - Lewis M. Clements, Kara M. Kockelman, 2017", SAGE Journals, 2021. [Online]. Available: https://journals.sagepub.com/doi/abs/10.3141/2606-14. [Accessed: 31-May- 2021].

[2]N. A. Greenblatt, "Self-driving cars and the law," in IEEE Spectrum, vol. 53, no. 2, pp. 46-51, Feb. 2016, doi: 10.1109/MSPEC.2016.7419800.

[3]E. Coelingh, J. Nilsson and J. Buffum, "Driving tests for self-driving cars," in IEEE Spectrum, vol. 55, no. 3, pp. 40-45, March 2018, doi: 10.1109/MSPEC.2018.8302386.

[4] M. Bojarski et al., "End to End Learning for Self-Driving Cars", arXiv.org, 2021. [Online]. Available: https://arxiv.org/abs/1604.07316. [Accessed: 30- May- 2021].

[5] Chirag Sharma, "Self Driving Car using Deep Learning Technique", International Journal of Engineering Research and, vol. 9, no. 06, 2020. Available: 10.17577/ijertv9is060247.

[6] Y. Miao, M. Gowayyed and F. Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), 2015, pp. 167-174, doi: 10.1109/ASRU.2015.7404790.

[7] S. Du, H. Guo and A. Simpson. Self-driving car steering angle prediction based on image recognition, 2019. available at: arXivpreprintarXiv:1912.05440

[8] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," Neural Networks, IEEE Transactions on, vol. 5, no. 2, pp. 157–166, 1994.

[9] S. Hochreiter and J. Schmidhuber, "Long short- ¨ term memory," Ne

[10] H. Eraqi, M. Mohamed and J. Honer, "End-to-End Deep Learning for Steering Autonomous Vehicles Considering Temporal Dependencies," 31st Conference on Neural Information Processing Systems (NIPS), vol 32, pp. 1-8, 2017.

[11] Z. Gu, Z. Li, X. Di and R Shi, "An LSTM-Based Autonomous Driving Model Using a Waymo Open Dataset," Appl. Sci., vol. 10, no. 2046, 2020. https://doi.org/10.3390/app10062046

[12] TimeDistributed layer, Keras.io. Accessed on: May 20, 2021. [website]. Available: https://keras.io/api/layers/recurrentlayers/timedistributed/

[13] R. Valiente, M. Zaman, S. Ozer and Y. P. Fallah, "Controlling Steering Angle for Cooperative Self-driving Vehiclzing CNN and LSTM-based Deep Networks," 2019 IEEE Intelligent Vehicles Symposium (IV), 2019, pp. 2423-2428, doi: 10.1109/IVS.2019.8814260.

[14] Uřičář, M., Křížek, P., Hurych, D., Sobh, I., Yogamani, S., Denny, P. (2019). Yes, we gan: Applying adversarial techniques for autonomous driving. Electronic Imaging, 2019(15), 48-1.