Supervised Learning for Autonomous Driving

Greg Katz, Abhishek Roushan, Abhijeet Shenoi

***Abstract*—In this work, we demonstrate end-to-end autonomous driving in a simulation environment by commanding steering and throttle control inputs from raw images. Supervised learning was used to map images to steering and throttle inputs based on example data from a human driver. The training data was augmented with adjusted images which allow the model to generalize better. Model selection was used to determine an appropriate neural network architecture and this process is documented. The capability of the model to drive autonomously in a real-time closed-loop simulation was evaluated. Further, the effect of smoothing data input and the augmentation of the data was explored. This work demonstrates the capability of a deep convolutional neural network with and without a Long Short-Term Memory layer to accomplish aspects of autonomous driving using only direct image data as input.**

I. Introduction

A world with exclusively autonomously driven cars is an inexorable possibility. Studies such as [10] have shown that having a central system that can route and control all cars on the road vastly reduces the average travel time of a commuter. As self-driving car technology improves, it is likely that more cars on the road will be equipped with the technology to autonomously drive, but more importantly, drive safely and reliably.

Our project explores the possibility of employing behavior cloning to construct a model to achieve steering and throttle prediction using only input images. The goal is to provide a proof of concept, that image data can indeed provide the information required to drive a car autonomously.

As a secondary unfold, the project explores the notion of what successful autonomous driving entails. Various metrics indicating the distance from the center of the lane and the average speed while staying on the track were evaluated. The project drew inspiration from a challenge introduced in the Udacity self-driving car nanodegree [4], having the prime objective of autonomous driving by predicting steering. We organize this paper as follows.

Section II contains previous work in the same domain and an overview of current methods and the state of the art. Section III contains information about the data-set and features appropriated for our project. Section IV contains the details about the methodology of our project and the investigations we carried out. Section V contains the experiments performed for optimizing our design followed by critical discussion. Section VI contains the conclusions, and we culminate with a discussion of future work in Section VII.

II. Related Work

The first step towards the area of autonomous vehicle navigation was taken by Pomerleau while designing the Autonomous land vehicle in Neural Network (ALVINN)[6]. Although the computing power at the time was limited, the network demonstrated the use input pixels for autonomous vehicle navigation.

The Udacity self-driving car nanodegree program [4] served as the starting point for this project. The Udacity setup was immensely successful in highway lane following situations and flat driving tracks.

Recently, the use of deep CNNs and RNNs for video classification, object detection etc. have shown the value of more complex neural network architecture. Besides CNN and/or RNN methods, research has shown progress in the video prediction domain. For example Comma.ai [7] proposed Variational Auto-Encoder (VAE) and Generative Adverserial Network (GAN) approach to learn a driving simulator.

Another related work is done by Shuyang et.al [8] which deploys steering angle prediction methodology from image recognition. The project uses 3D convolution layers to predict steering angles and automatic extraction of features to make the prediction. This research most closely resembles our work.

Furthermore, Reinforcement Learning (RL) has proven effective for autonomous driving. Ahmad et.al [9] have proposed one such approach. The underscored research incorporates Recurrent Neural Networks (RNN) for information integration, which enables the car to handle partially observable scenarios. In the case of self driving cars, it is easier to specify a reward for good driving rather than to design a loss function which captures this information. Therefore, RL algorithms provide a distinct advantage over supervised learning approaches to the autonomous driving problem.

III. Dataset and Features

The data-set is collected by simulation of a car driving around a designated track.

Consequently, the recorded images associated with a steering angle and throttle input serve as features to train on. The details are as follows.

A. Simulator Data Collection

Udacity provided the basic simulation framework. The simulation environment is built in Unity [5]. In the training mode, the simulator records steering angle, throttle, and images (160×320) from three simulated dashboard cameras while a human drives the car around the track.

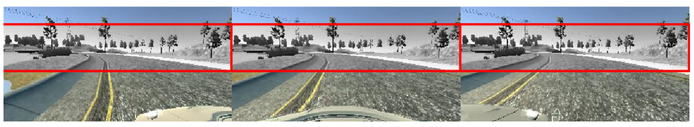


Fig. 1: Input images

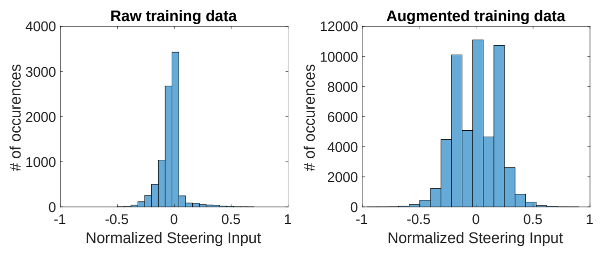


Fig. 2: Comparison of Raw and augmented steering angles

We also perform image cropping corresponding to regions useful for learning road turns (Figure 1).

Images and steering angles while driving the car around the track were recorded, and subsequently, the image-steering angle examples were used to train different models.

Data was collected using a joystick controller for several laps around the track. The final training data set consisted of 8662 steering angle samples with three images (center, left, right camera for each step) and a throttle per steering angle recorded.

B. Data Pre-processing

In the driving mode for data collection, the vehicle was driven in the center of the lane. The large majority of the training data corresponds to zero steering input. We found for training on this data set, the car did not have enough examples to learn how to return to the center of the road if for any reason it drifted apart from center. In order to augment the data, images from a simulated left and right camera were added. For these images, a steering correction factor was applied to the steering angle.

Additionally to generalize the data set further, all images were flipped horizontally and the steering angle negated for these examples.

After augmentation, our data set consists of 51972 examples. Figure 2 shows the steering inputs before and after the data augmentation. Further, all images are normalized to -1 to 1 range, cropped to focus on the road and converted to grayscale.

IV. Methods

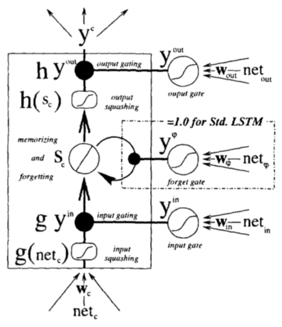
The majority of this project deals with deep convolutional neural networks. Additionally, the effect of adding LSTM's in the network was also studied.

A. Convolutional Neural Networks

CNN's are a type of feed forward neural network, that use shared weights to allow for spatial invariance, and to reduce the number of total parameters. A single layer consists of a set of neurons which are connected only to select neurons in the subsequent layer. They are trained by backpropogation.

B. LSTM

LSTM's are a type of Recurrent Neural Network (RNN). A single LSTM 'cell' consists of its three gates: an input gate, an output gate, and a forget gate. The cell itself has a state which can be used to remember the information. Each of the gates alter the cell state by way of weights and transfer functions. These weights are updated using backpropagation. [3] contains the details of the update rules, and the equations for the update of the cell state, also reproduced in Figure 3.



$$y^{out_j}(t) = f_{out_j}\left(\sum_m w_{out_j m}\, y^m(t-1)\right), \quad (1)$$

$$y^{in_j}(t) = f_{in_j}\left(\sum_m w_{in_j m}\, y^m(t-1)\right). \quad (2)$$

$s_c(t-1)$ $(t > 0)$:

$$s_{c_j^v}(t) = s_{c_j^v}(t-1) + y^{in_j}(t)\, g(net_{c_j^v}(t)), \quad (3)$$

$$y^{c_j^v}(t) = y^{out_j}(t)\, h(s_{c_j^v}(t)). \quad (4)$$

$$y^k(t) = f_k\left(\sum_m w_{km}\, y^m(t-1)\right), \quad (5)$$

Fig. 3: LSTM implementation details

V. Experiments and Discussion

Using the generated data-set we proceeded to train prospective models to evaluate the best candidate.

A. Model Selection

Based on the related works and primary insights, we chose a total of 5 different model types: Linear regression model, Shallow Neural Network, Simple Convolutional Neural Network (CNN), Deep CNN, NVIDIA CNN model ([2]), and NVIDIA CNN with Long Short-Term Memory layer (LSTM). The models were compared to ascertain the best option. On a single data set, each model was trained using backpropagation with minibatch Adam optimization [1] and a Mean Squared Error (MSE) loss function over an 80-20 train-test split. We then compared the models based on the following:

1) Error vs Epoch Analysis: The test and validation errors are recorded after each training epoch. The optimization is considered converged when the error between iterations levels off. Additionally the plot is used to look for over-fitting which would be indicated by a large gap between training error and validation error and/or an increase in validation error with epoch. In the case where validation error increases, early stopping can be used to reduce over-fitting.

2) Learning Curves: The learning curves are used to analyze bias and variance effects in the model training. A large difference between training and validation errors, indicates that the model does not generalize well. A downward trend on the learning curve indicates that additional data could result in a better model. The results indicate the NVIDIA architecture,

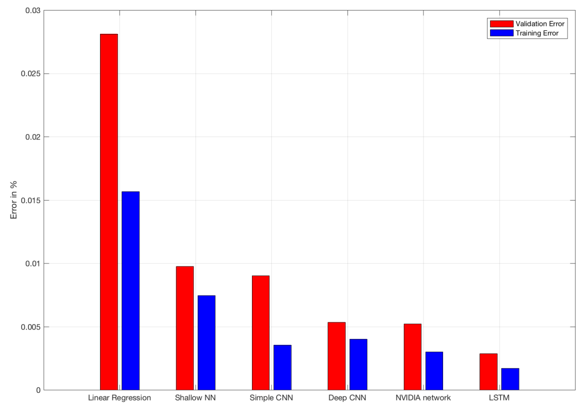


Fig. 4: Optimal Errors at final epoch

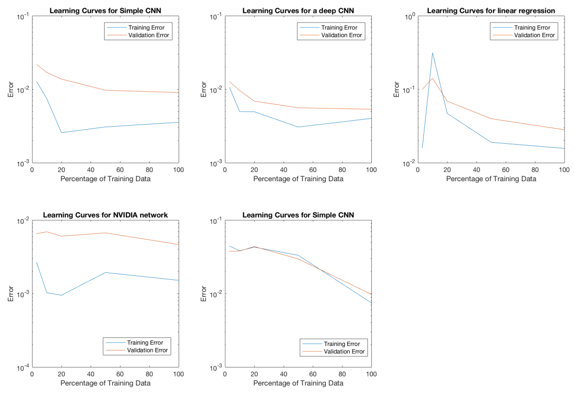


Fig. 5: Learning curve for all models

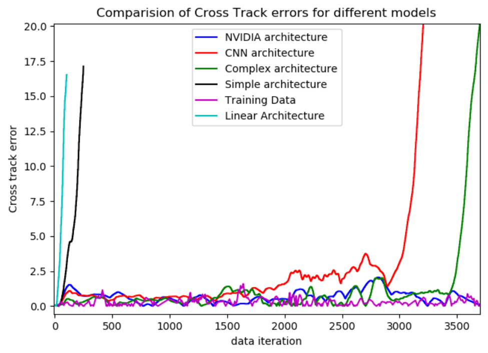
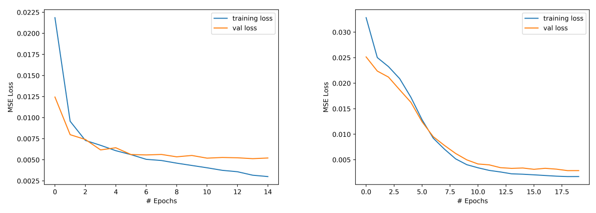


Fig. 6: Models Cross track errors



(a) NVIDIA (b) LSTM

Fig. 7: Training Curves

adapted from [2], performs best on the training as well as the validation set. Also, the downward trend on the validation set indicates that more data will help the network generalize better. Hence, we expanded the size of our data set to further train our selected models.

3) Cross track Error analysis: We derive the ”Cross track errors”, the distance from the center of the track, while autonomously driving the car around the track. We use this as a figure of merit for our trained network and the plots are shown in Figure 6. It is evident that the selected architecture performs closest to the training data.

The NVIDIA architecture proved to be the most promising, based on Figure 4, and the best on track performance. This architecture was adopted for all further work. In addition to this, it seemed likely that sequences of images would have further information to help us predict steering angle more accurately, and further predict the throttle as well. We included an LSTM layer within the NVIDIA model architecture and compared the effect of its addition.

B. Refining the model

Upon selecting the network architecture we carry forward by optimizing our meta parameters and preprocessing. In this phase we found that grayscale images allow for steering prediction at the same level of full colour images, with the added benefit of lower training time. The NVIDIA network tended to overfit after 5 epochs of training. Early stopping was used, and the weights before the increase of validation error were used for testing. A batch size of 100 was found to give good convergence, and training within a reasonable time. Training took on average around three minutes per epoch.

To combat the overfitting in the LSTM network, dropout, with a probability of 0.2, was used in all the fully connected layers. This dramatically reduced the overfitting and led to a validation error lower than the NVIDIA network. Using a time-series of 3 images per input showed the best results on our test set. The training time was significantly higher than the NVIDIA network, at ten minutes per epoch. Example training curves for both models are shown in Figure 7.

We evaluate our models on how successful they are in completing the track autonomously and how well the driving behavior matches the training data. In regard to autonomous driving our metric was the maximum speed at which the car could reliably complete the track. The higher the speed, the smaller the margin for error on steering angle, so a more accurate model can complete the track at higher speeds than a less accurate one. Additionally, since we did not have a second track available in our simulation environment, the testing was done on the training track driven in reverse to test the ability of the models to generalize to unseen inputs.

C. Training Data vs Prediction

Before driving autonomously, the first test was to run the center images from the training data through the model and compare the prediction to the label. We expect a good match

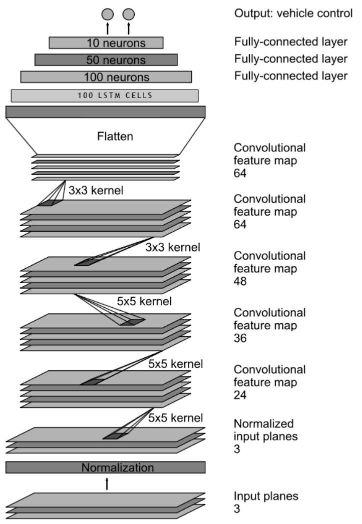


Fig. 8: Network architecture from NVIDIA with LSTM layer

because we are comparing predictions with data used in training the model. However as you can see in Figure 9, the model prediction is quite noisy compared to the labeled data. We hypothesized this might be caused by similar images with very different steering angles leading to inconsistent predictions. To mitigate this effect, when recording our second data set joystick inputs were smoothed to provide a cleaner signal for training. Figure 9 shows the effect of this smoothing in a percentile plot. As it turns out, the smoothing did not reduce the noise in the prediction. Figure 9 shows that the noise remains with the smoothed data. In fact, root mean square error (RMSE) increased from 0.059 to 0.073 after smoothing the training data.

A second hypothesis for the noisy steering predictions was that the left and right camera images used to augment the dataset lead to very similar images with very different steering labels. So we trained a model on just the center images and ran this test one more time. This did mitigate the noise as can be seen in the final plot of Figure 9. This time the RMSE was reduced to 0.040. However, this model was not able to complete a lap around the track even at low speeds. The conclusion is that the data augmentation improves success in autonomous driving at the expense of making it more difficult for the model to produce smooth and sensible steering predictions for every image.

D. *Autonomous Driving*

In regard to driving autonomously, both models were able to complete the track in forward and reverse. Our initial goal was drive the track at 10 mph and this worked so well that we extended the goal to 30 mph. This goal proved a challenge just out of reach. At high speeds the models become unstable and oscillate until leaving the track. At somewhat lower speeds, however, the autonomous driving behavior of the cars was stable. Table I shows the maximum stable speed achieved by each model on each track. The LSTM model performed somewhat better than the NVIDIA model, which is due to the ability to speed up on the straights and slow down on the turns. Subjectively, our observation was that the NVIDIA model actually drove more smoothly and consistently in the center of the track.

To examine more closely the autonomous driving performance, we compared the steering angles and cross track error of the training data and the autonomous driving. Figure 10 shows these results for the NVIDIA model driving at a constant 20 mph. This is different from Figure 9 observation

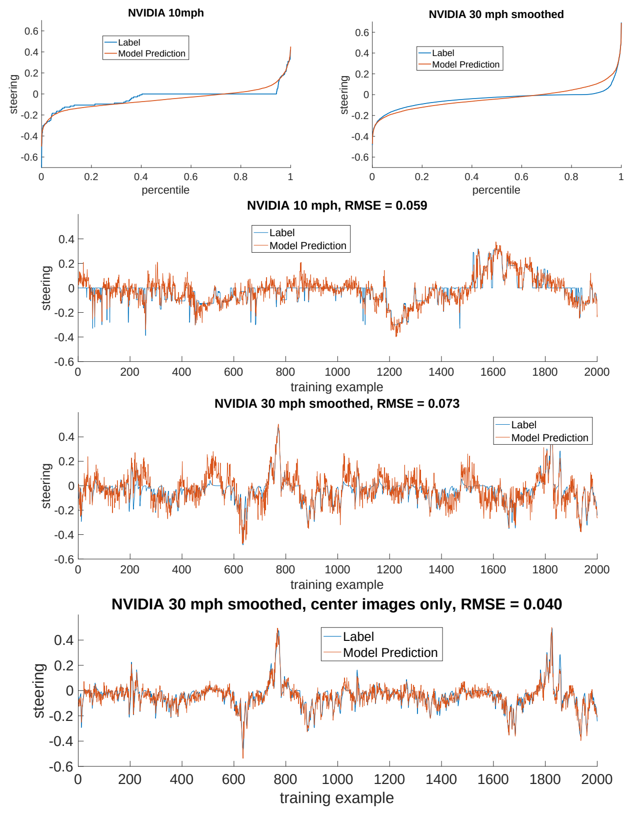


Fig. 9: Steering angle Predictions on center camera test data

Model	Forward Track	Reverse Track
NVIDIA	20	15
LSTM	22	25

TABLE I: Models Max stable speed (mph)

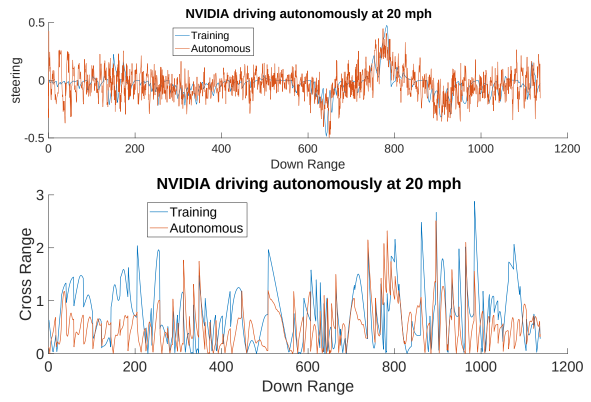


Fig. 10: NVIDIA Autonomous Driving

in that this data is from driving autonomously so the images are generated in a feedback loop by the simulation based on the way the car is driving. Thus in order to make a useful comparison. we plot the same against the longitudinal position on the track which we call the ”Down Range”. The results show that the model is even noisier in autonomous driving. However, if we look at the ”Cross Range” plot which measures how far the car is from the center of the road, we see that the autonomous driving performs quite well. The noisy steering angle seems to be filtered out by the vehicle dynamics in the simulation and does not show up in the position. The cross range of the autonomous driving stays near or below the cross range of the training data and has a similar variability and frequency.

E. LSTM Model and Throttle Control

The results discussed so far have focused on the NVIDIA model. We now turn our attention to the LSTM model. We expected this model to perform better because it has more information, using the last 3 images for its prediction rather than 1. However, Figure 11 shows that the steering angle is actually even noisier than the NVIDIA model. This matches our subjective observations.

What allows the LSTM to drive faster on average than the NVIDIA model, despite the noisier steering angle, is that the LSTM model controls throttle in addition to steering angle. For the NVIDIA model we used a PID controller for throttle input to maintain a constant speed (provided by Udacity). We hypothesized that individual images are not sufficient to reliably predict a throttle input, but since the LSTM processes temporal input, it may have some ability to predict a throttle from a series of images. Figure 11 shows the a comparison of the speed of the training data with the LSTM model driving autonomously. Mysteriously, there is a general bias in the autonomous data being slower than the training data, but it did show the ability to accelerate on the straights and slow down for the turns. Since the turns are usually where the car drives off the road, this allowed the LSTM to be successful getting around the track at a higher average speed.

VI. Conclusion

In this work we have demonstrated autonomous steering and throttle control in a closed-loop, real-time simulation using only raw images as input. Furthermore, generalization to new images was shown by successfully driving the track in reverse. This proof-of-concept shows the power of deep convolutional neural networks with long short-term memory to learn the necessary visual features without traditional visual processing steps such as lane detection. The simulation, with a rich visual representation, suggests further work in a higher fidelity simulation with more varied terrain would be fruitful.

The results of this research confirmed that the NVIDIA network is an excellent network architecture for training raw images to predict steering angle. We also found that augmenting data with left and right cameras did improve driving stability but at the expense of a noisier steering input. We found that smoothing the steering input in the training data was not necessary and did not improve performance. We saw that although the steering was noisy the behavior of the car as measured by position relative to the center of the road was a good match with the training data. Finally, we found that adding an LSTM layer allowed for a usable throttle control prediction, though at the expense of some accuracy in the steering prediction.

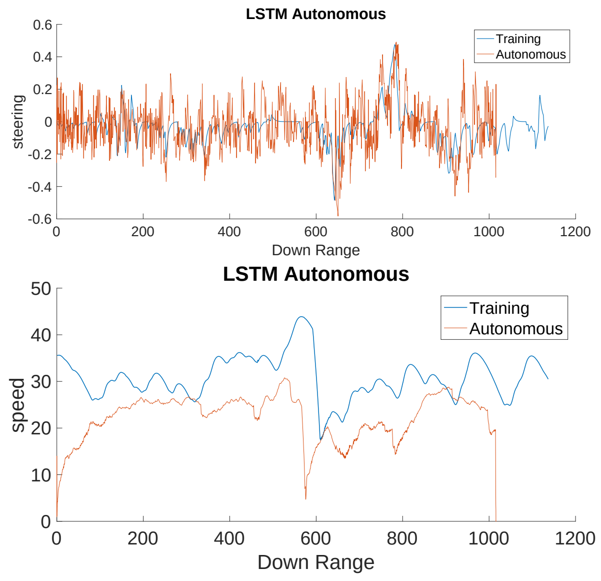


Fig. 11: LSTM Autonomous Driving - Car drove off the road at about 1000 Down Range

Although the goal of autonomous driving in this simulation was accomplished, there were limitations in the speed of stable driving and in the noisy behavior of the steering prediction. The future work section will discuss some ways in which the results could be improved upon moving forward.

VII. Future Work

This work has been a proof of concept, but there are several ways in which we would like to extend this research in the future. One of the most challenging aspects of this task is determining the connection between error in model training and performance in the autonomous driving task. We found, for example, that the LSTM model had a lower validation error but was less stable in autonomous driving. Thus, minimizing validation error on the training data is not enough. Thorough testing in autonomous driving scenarios is necessary to a achieve reliable, high-performance controller. To optimize autonomous driving performance, a better autonomous driving test suite would be invaluable. Instead of simply determining the fastest speed to complete the track, an improvement would be to test the car in hundreds or thousands of automatically generated test scenarios and trenchantly analyze failure cases.

A second important extension to this work would be to use a simulation environment with more richness and variety to demonstrate the potential of extending this to real world driving. Finally, autonomous driving is well-suited for reinforcement learning. The controller developed in this work could be used as a starting point for optimization by reinforcement learning.

VIII. Personal Contributions

A. *Greg Katz*

Modified Unity simulator to output vehicle global position, increase max speed, and allow manual override in autonomous mode. Set up joystick and recorded training data. Adapted script to train neural network using Keras. Created CNN and LSTM models for training. Adapted script to drive autonomously using predictions from neural network. Took lead in testing and evaulation of models in autonomous driving scenario.

B. *Abhishek Roushan*

Modified training model to train with split fraction data for learning curves; Initial training of network varying batch-size and epochs to arrive at convergence; Function modeling for calculation of Cross track errors, average velocity and maximum off track distance. Performed initial Image processing by grayscale training on NVIDIA architecture. Network training on GPU cluster.

C. *Abhijeet Shenoi*

Modified the simulator code to allow for programmatic startup; Created the deep CNN network model. Trained all networks using the oat GPU cluster. Implemented the LSTM network. Added dropout. Optimised hyper-parameters for the NVIDIA and LSTM networks. Generated plots of learning curves, errors vs. epoch, training error curves. Implemented code to test the effect of augmentation of input image data.

References

[1] Kingma, Diederik, and Jimmy Ba. ”Adam: A method for stochastic optimization.” arXiv preprint arXiv:1412.6980 (2014).

[2] Bojarski, Mariusz and Del Testa, Davide and Dworakowski, Daniel and Firner, Bernhard and Flepp, Beat and Goyal, Prasoon and Jackel, Lawrence D and Monfort, Mathew and Muller, Urs and Zhang, Jiakai and others. End to end learning for self-driving cars

[3] Gers, F.A., Schmidhuber, J. and Cummins, F., 1999. Learning to forget: Continual prediction with LSTM.

[4] Udacity-An open source self-driving car https://github.com/udacity/self-driving-car-sim

[5] https://unity3d.com

[6] ALVINN, an autonomous land vehicle in a Neural Network http://repository.cmu.edu/cgi/viewcontent.cgi?article=2874&context=compsci

[7] Learning a driving simulator https://arxiv.org/abs/1608.01230

[8] Shuyang Du, Haoli Guo, Andrew Simpson: Self-Driving Car Steering Angle Prediction Based on Image Recognition. http://cs231n.stanford.edu/reports/2017/pdfs/626.pdf

[9] Ahmad El Sallab, Mohammed Abdou, Etienne Perot, Senthil Yogamani: Deep Reinforcement Learning framework for Autonomous Driving https://arxiv.org/abs/1704.02532

[10] Tim Roughgarden: Selfish Routing and the Price of Anarchy

[11] Chollet, François and others: Keras, 2015 https://github.com/fchollet/keras

[12] Martn Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Rafal Jozefowicz, Yangqing Jia, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Man, Mike Schuster, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vigas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.