Convolutional Architectures for Self Driving Cars

Gaurav Bansal (gbansal@us.toyota-itc.com) **Dhruv Choudhary** (choudharydhruv@gmail.com)

Abstract

Self-driving vehicle control system would have to determine steering wheel angle, brakes, and acceleration in any driving environment. Convolutional Neural Networks (CNN) have been recently proposed as an effective solution for predicting steering wheel angles for self-driving cars. In this project, we have formulated steering angle prediction as a regression problem and have used open-source driving data sets released by Udacity to evaluate various CNN architectures. We have devised a novel technique of using image sharing between vehicles via Vehicle-to-Vehicle (V2V) communications. Our results show that by employing novel image sharing technique, we are able to reduce the validation loss by almost five times as compared to the baseline and achieve performance with mean error of 3.5 degrees in the steering wheel angle prediction.

1. Introduction

Self-Driving cars can have a huge impact on the society. The traditional robotics approach for designing self-driving software has four main components: Localization, Mapping, Perception and Path-Planning. Data from various on-board sensors (cameras, radars, Lidars, GPS, HD maps etc.) is used for realizing such a solution. This approach has severe challenges in many scenarios such as: bad-weather conditions, non line-of-sight view (driving around intersections), long range sensing etc. There has been a recent push to use deep learning for designing end-to-end self-driving systems. This approach could learn from the data collected by human driving and effectively try to emulate human driving behavior.

In this project, we use a convolutional architecture based deep learning solution for designing self-driving software. Specifically, we use camera data from vehicles to train on the steering wheel angle. We have considered this as a regression problem for steering angle prediction. We use an open-sourced driving dataset released by Udacity [2]. We have cropped images in the driving dataset to focus on the section of the image most relevant to steering wheel angle learning. We have used techniques to augment the data collected from additional cameras on the car. We have used image flipping to further augment our dataset. Our baseline results show that Convolutional Neural Networks (CNN)-based deep learning solutions can be designed to effectively predict steering wheel angle.

We have also proposed a novel approach of using future image as input to the CNN by sensor data sharing between vehicles using vehicle-to-vehicle (V2V) communications. Automotive industry have been working on V2V communication technologies for many years and deployments have already started in US (GM's Cadillac CTS), Japan (Toyota Lexus, Prius and other models), and Europe (recent announcement from VW) [10], [11] and [12]. These deployments allow vehicles to share 300-400 bytes of messages, at a periodic 10 Hz rate with a transmission range of 300-500m. Additionally, Cellular industry is getting ready for deploying 5G technologies with automotive as one of the main use cases. 5G-based V2V communications will enable vehicles to share huge amounts of data (Gbps and higher), with ultra-low latency (10 msec and below) and high reliability (Packet Error Ratio of 5% and below) [6], [13], and [14]. This would allow vehicles to share raw camera images which could be used for deep learning applications. Specifically, in this project we have created a virtual vehicle that is moving two-seconds ahead of the ego vehicle (Note, ego vehicle in this report refers to self-driving vehicle for which we are predicting steering wheel angle) and sharing raw camera images. We show that by using these future images from the vehicle ahead, we are able to reduce the validation loss by almost five times. To the best of our knowledge, this is the first time when raw image sharing between vehicles is used for deep learning based self-driving car systems.

The rest of the report is organized as follows. In Section 2, we provide details on related work. We present our data processing, augmentation and proposal for using future image data in Section 3. Deep learning network architecture is presented in Section 4. Evaluation and visualization of both baseline and network using future images is provided in Section 5. Finally, the conclusions and discussion on future work is presented in Section 6.

2. Related Work

Bojarski et al. [1] (NVIDIA team) have used CNNs to train input camera images to predict the steering wheel angle. They have formulated the steering wheel prediction as a regression problem and have used three cameras (center, left and right) to augment the data set during training, and thus generalize learning. The center camera sees the middle of the road, left and right cameras are tilted sideways. Correction factor is added to the steering angles corresponding to images collected from left and right cameras. Data augmentation techniques such as adding random rotations to the steering angle have also been applied. Deep network architecture uses five convolutional layers followed by five fully-connected layers.

Udacity launched a similar self-driving car challenge [3] for using camera data to predict steering wheel angle. Udacity has open-sourced driving dataset collected for this challenge. Data-sets has images from 3 cameras (as in [1]) and steering wheel, torque and brake data. Many solutions by teams participating in this challenge used CNN architectures similar to the one used by Bojarski et al. [1]. They report that deeper architectures (such as ResNet, VGGNet) perform worse for this regression problem than relatively shallower CNN architecture in [1]. The final leaderboard and achieved validation accuracies are available at [9]. The winning team [8] used a combination of 3D CNNs and LSTMs to achieve mean error of 2.6 degrees error. While this approach is accurate, it is computationally quite expensive. The second position team [7] used a relatively simpler approach while achieving impressive performance of 3.3 degrees. They propose using CNNs similar to the ones used by Bojarski et al. [1] but instead of using raw images as input, they pass in the delta between consecutive images in the dataset. This is based on the intuition that the displacement of image features is more valuable than the image itself. But they strictly use past images in the dataset. Our sensor data sharing is inspired from this approach, but we take the delta between images in the future rather than the past.

Currently around 100 people die every day in car accidents in US. Automotive industry believes that V2V communications, which allows vehicles to share data among each other can prevent more than 80% of vehicle related crashes. In this regard, V2V deployments are already undergoing in US and Japan [10], [11], and [12]. Research in 5G technologies, holds great promise for enabling high data rate and ultra low latency for V2V communications. Recently, many 5G demonstrations have been conducted by Nokia, Ericsson, Intel [13] etc. Self-Driving cars are frequently mentioned as one of the prominent use cases for 5G [14].

3. Dataset and Features

Udacity has released data sets from 5 trips with a total drive time of 1694 seconds (28.23 minutes) [2]. Test vehicle has 3 cameras mounted as in [1]. Camera images are collected at a rate of around 20 Hz. Steering wheel angle, brake, acceleration, GPS data was also recorded in the experiments. The image size is 480 X 640 X 3 pixels and total data set is 3.63 GBytes.

Figure 1 plots the time series of recorded steering angles across the 5 trips. The trips were taken at different locations with the following attributes:

Trip#1: Direct sunlight, moderate turns in the beginning.
Trip#2: Shadows, tight turns, curvy roads.
Trip#3: Moderate turns, shadows.
Trip#4: Tight turns, elevation.
Trip#5: Traffic, fairly straight road, multiple lanes.

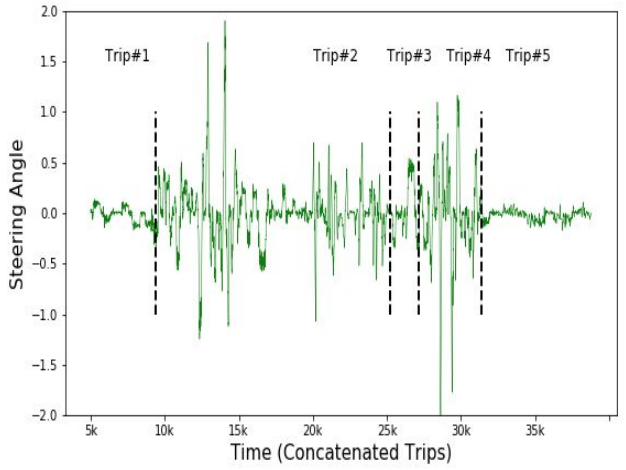


Figure 1: Time Series of Steering Angles across 5 trips.

It should be noted that although the training set has 33.8k images, there are only few images in the dataset with very steep turns. This will be a challenge during training because most examples have low to moderate steering angles, making it hard to learn and predict steep left and right turns.

The dataset is a series of images and there is high correlation between adjacent samples, thus it is important during training to shuffle the training images. Moreover the validation strategy has to be chosen carefully. For e.g. if we choose the last part of the dataset as validation (Trip 5), the images are from a fairly straight road and does not have any steep turns. If we randomize the whole dataset and choose a validation set, we might get very similar images in the validation and training sets. Thus it will get harder to detect if the network starts overfitting. Thus we choose the last 20% of each trip for cross validation. This allows us to check how well the model generalizes to unseen images and also helps us capture some steep turns from Trip 2 and Trip 4.

3.1 Data Processing and Normalization

We normalized and zero-centered image data (x / 255.0-0.5). We carefully analyzed the image dataset and cropped top 1/2 (240 pixels) of the images as it does not have the information on the road (mostly consists of trees, sky etc.). Figure 2 shows two example images from the dataset (top portion above blue line has been cropped).

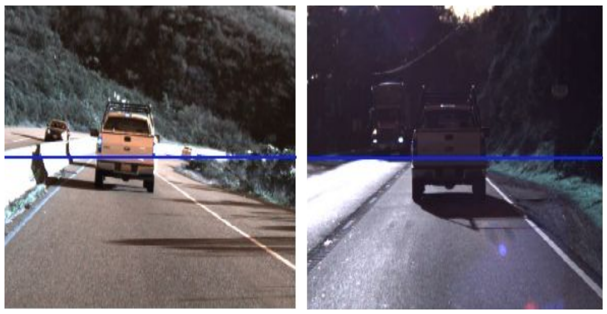


Figure 2: Cropped Images in the original datasets

We also removed images with steering angle values below a certain threshold (.05 radians). This removes noise in the data collection process and helps balance the dataset better because a large part of the data involves straight driving. Figure 3 shows the original distribution (using images from center camera) of steering wheel angles. The distribution is not normal and hence we would not expect mean squared error (MSE) loss to work well in this situation. In Figure 4, we removed images with corresponding steering angle value below 0.05 radians and it shows a much balanced distribution.

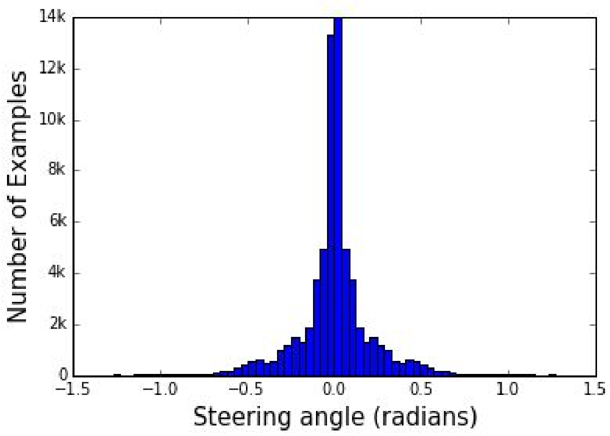


Figure 3: Distribution of Steering Angles in the original data-set

3.2 Data Augmentation

We have used data from left and right cameras to augment the data collected by center camera. We adjust the steering angle by a fixed threshold to account for the positioning difference between these cameras. These off-center cameras enable training for recovery paths for situations when car might weave from center of the road. Specifically, we adjust the steering angles for images from left and right cameras as:
steering_left = steering_center + STEER_CORRECTION
steering_right = steering_left - STEER_CORRECTION

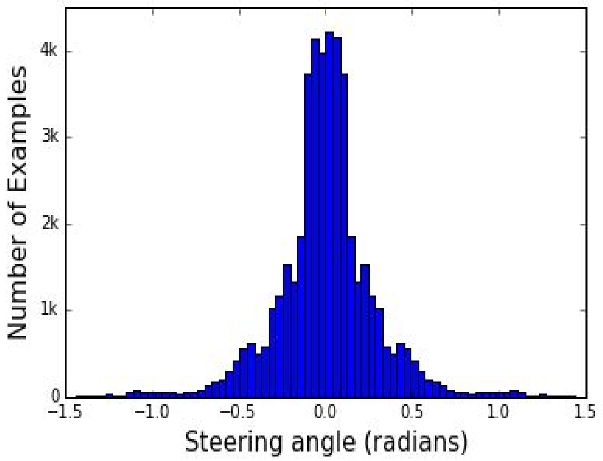


Figure 4: Distribution of Steering Angles after removing images corresponding to low steering angles

We do not have the ground truth for this correction factor and have treated it as a hyperparameter. For the baseline network we set STEER_CORRECTION to a value of 0.1 radians after cross validation.

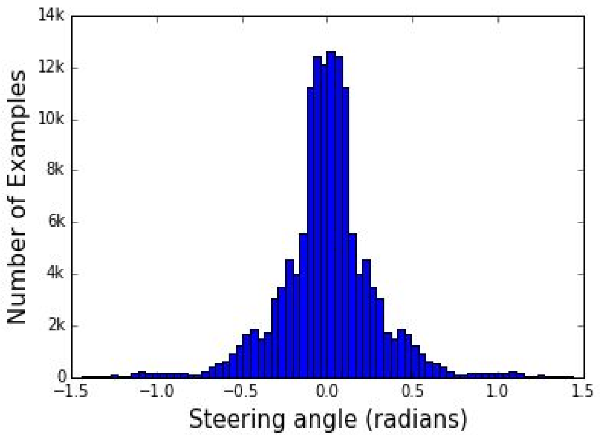


Figure 5: Distribution of Steering Angles using images from all three cameras and after applying STEER_CORRECTION for Left and Right Cameras.

3.3 Future Images

In this project, we use a novel idea of sensor data sharing between vehicles. There has been lot of progress in new 5G standardization by the cellular industry [6]. Specifically, wireless technologies have been designed that can enable sharing Gbps data between vehicles at very low latency (10 msec and below). These developments enable the use of raw images from lead vehicle for end to end self-driving applications.

Note, we do not propose using steering wheel angle from vehicle ahead for end-to-end self driving applications. Lead vehicle can easily share steering angle data (requires very low bandwidth) and theoretically same steering angle can be used by ego vehicle when it arrives at the exact same location. However, state-of-art GPS receivers have significant errors in localization. Thus precise location of the lead and the ego vehicle is unknown. Recently, LiDARs are used for precise localization, however LiDAR can be an expensive sensor and not able to provide localization in bad weather (rain, snow etc.) conditions. Hence in this work we focus on image sharing, which can be used by CNN architecture without the knowledge of exact location of the lead and the ego vehicle.

For this work we did not have access to data-sets where two vehicles were driving on the road at the same time. We created a virtual vehicle trace by shifting the data set of the original vehicle in time. We followed the Two-Second driving principle used by various states in US, which recommends drivers to maintain two second gap between vehicles for safe driving [16]. The camera data in Udacity data-sets was recorded at roughly 20 Hz. Hence, we created a new virtual vehicle by shifting the original vehicle 2 seconds ahead in time.

By using this technique, at any time t, ego vehicle can receive Image(t+40) from the virtual vehicle. Note, that ego vehicle can store images received from the virtual vehicle and use them at a later time. We are assuming that only images from center camera are being shared. We did not use flipped images data augmentation for images received from the virtual vehicle.

Figure 6 and 7 below shows two images at time t as seen by the host vehicle and virtual vehicle (2-second ahead) respectively. As we can see from these figures, by obtaining future image from the virtual vehicle, the ego vehicle can know about the upcoming turn earlier. Future images can be given as input to CNN and enable improved steering angle prediction.

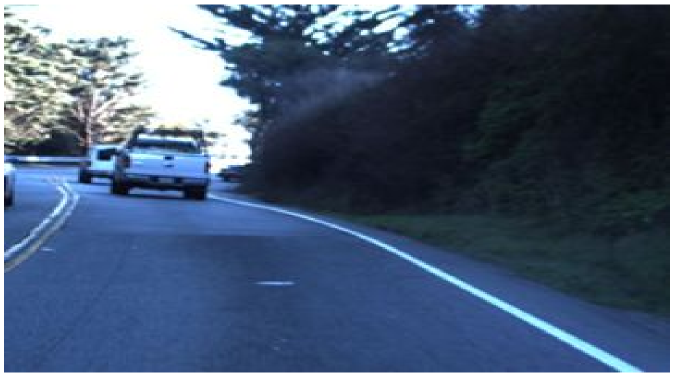


Figure 6: Center Image seen by ego vehicle at time t

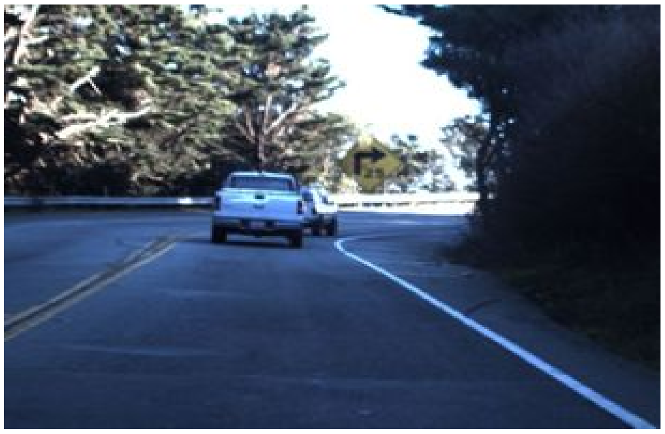


Figure 7: Center Image seen by virtual vehicle (Two-second ahead) at time t.

4. Methods

We currently use the CNN architecture as in [1], while formulating steering angle prediction as a regression problem. Network has ten layers, and we have extensively used Batch Normalization and Dropout (which was not used in [1]). The 10-layer network architecture is:

Layer	Type	Size
1	Conv	5 x 5, 24
2	Conv	5 x 5, 36
3	Conv	5 x 5, 48
4	Conv	3 x 3, 64
5	Conv	3 x 3, 64
6	FC	1164
7	FC	100

8	FC	50
9	FC	10
10	FC	1

Table 1: CNN architecture used for steering angle prediction

The CNN architecture is also illustrated in the figure below:

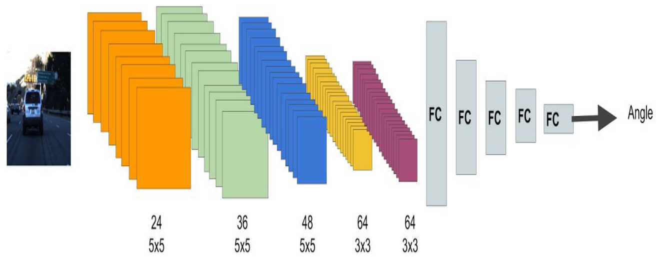


Figure 8: CNN architecture used for steering angle prediction

We are using Batch Normalization after all the 5 Conv Layers and Dropout after first four fully connected layers. We identify setting Dropout layer Keep_probability parameter to 0.2 provides the best performance for our model. We have used ReLu nonlinearity after all the layers (except the last one). We uses a mean squared error (MSE) loss without any regularization. All these design choices were derived using extensive experimentation with a cross validation set consisting of 20% of the data. As few examples, we have tried placing Batch Normalization layer before the ReLU activations, but the configuration performs significantly worse. Hence, we decided to use Batch Normalization after the ReLU activations. Dropout with probability 0.5 also performs worse than probability of 0.2. We also experimented with different number of convolutional and fully-connected layers. Our results show that best performance is achieved by using 5-Conv and 5-FC layer architecture.

For baseline we provide the current image as input to the network. For the case with future images, we provide the differential input at time t: Image(t) - Image (t+x). Here, Image (t+x) is coming from a virtual vehicle two seconds ahead of the ego vehicle. Images from the vehicle ahead can be stored and x can have any value between 1 and 40 (40 is calculated using 20 Hz camera rate).

5. Evaluation

In this section, we first present evaluation and visualization for our baseline network. We will then present evaluation while using future image data as input to the network.

5.1 Baseline Network

We simulated the baseline network described in Section 4. We use Keras [4] for all our experiments with a Tensor Flow backend. The network is trained using the MSE Loss function. We used the Adam Optimizer with a learning rate of 1e-3 and no decay rate. Our minibatch size is 64.

We train the network for 20 epochs and used early stopping to choose the model at 18th epoch. We achieved a validation loss of 0.0179, which is equivalent to 7.6 degrees mean error in the steering angle prediction. We have plotted the validation and training loss for baseline network in Figure 9. It should be noted that the training loss is higher than the validation loss. The reason for this is that the training data has a lot more steep turns and we do not predict well for such steep turns. Thus the training loss is skewed by these turns. Validation loss has relatively fewer steep turns.

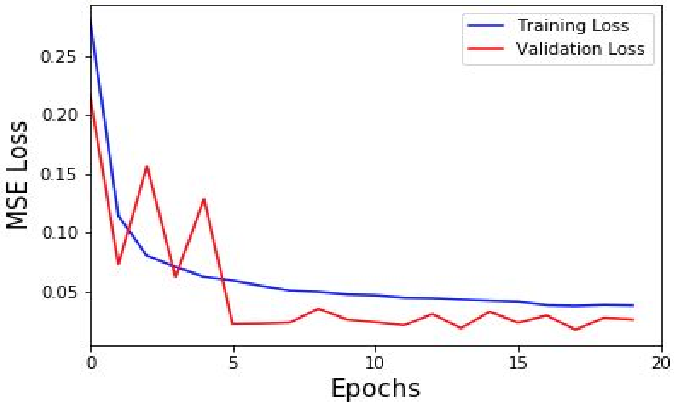


Figure 9: Loss function for the baseline network

5.2 Visualization

We plot the learned weights of 24 filters in the first convolutional layer of the baseline network in Figure 10. We can see that each filter is learning to detect a different pattern in the input image.

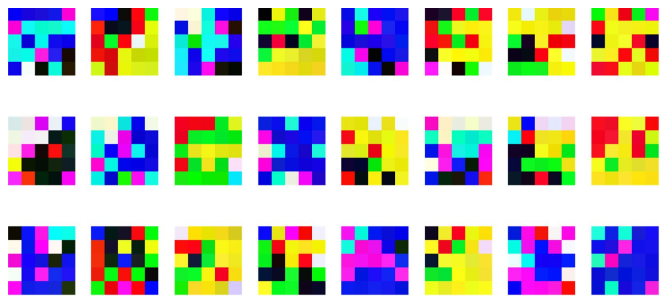


Figure 10: Visualizations of the First Layer Weights

We plot the activations at the first convolutional layer for each of the 24 filters in Figure 11. We plot two images, one with the road turning (left image) and one with a straight road (right image). We can see that for both the images, different filters get activated which is in-line with the expectation from a network learning to predict steering angle from driving data. We also found that some of the filters were not detecting an obvious pattern, we did not have the time to evaluate this and leave it as future work. It is also interesting that the network learns to detect the lane markers. This is because most of the dataset has clear lane markings. This means that in areas without well defined lanes the predictions may be quite off.

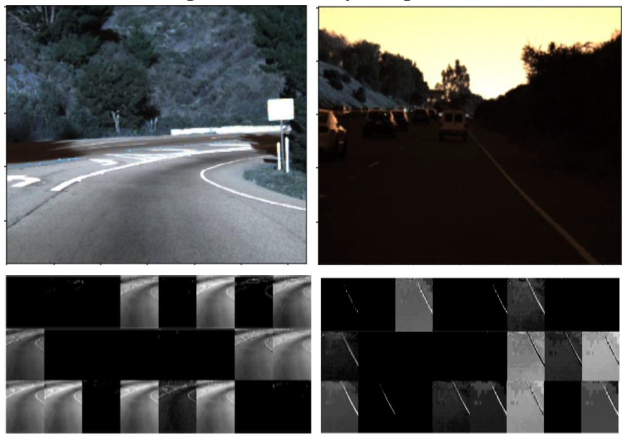


Figure 11: Activation Maps For 1st Convolutional Layer

In Figure 12, we plot the predicted and measured (as used by the human driver) steering angle across the whole dataset (training and validation). We can see that the baseline network tracks the measured steering angle quite well for moderate turns, but for steep turns it is not able to predict the huge turn (i.e. large steering angle). Trip 2 and Trip 4 specifically have some very tight turns and the highest errors in the prediction come from these trips.

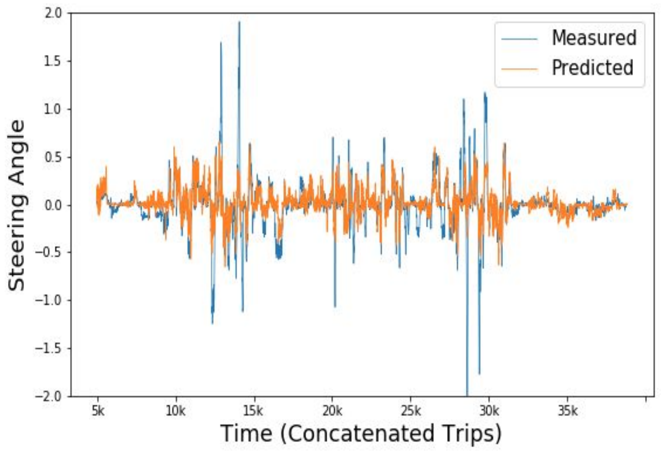


Figure 12: Measured and predicted angles across the whole dataset.

We show various example images from the validation set with values of true and predicted steering angle (in red text in the top left corner of images) in Figure 13. It can be seen from these various scenarios that our baseline network is able to learn from the dataset and able to predict steering angles close to the true angle used by human driver. Figure 14 shows challenging scenarios with images having very steep turns. As we can see the predictions with baseline network are not very accurate for these turns.

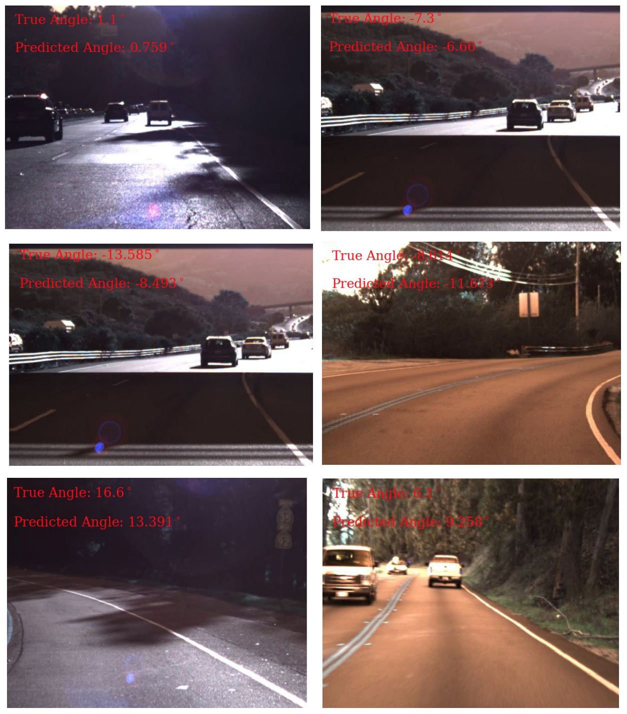


Figure 13: Example images from the validation set

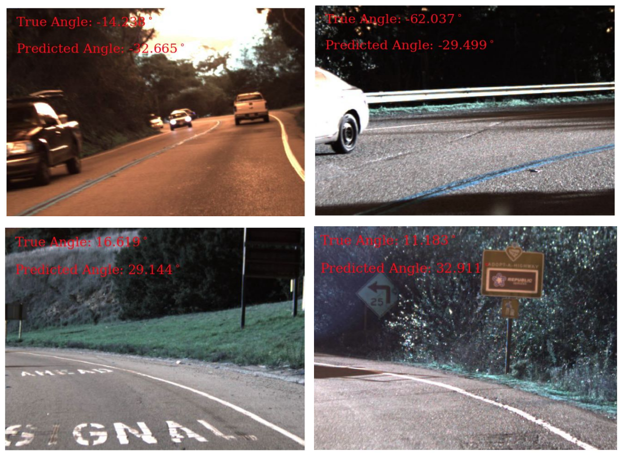

True Angle: -14.238 °
Predicted Angle: -32.665 °
True Angle: -62.037 °
Predicted Angle: -29.499 °
True Angle: 16.619 °
Predicted Angle: 29.144 °
True Angle: 11.183 °
Predicted Angle: 32.911 °

Figure 14: Examples images of sharp turns.

5.3 Future Images

In this section, we used the images coming from the vehicle ahead using V2V communications. As described in Section 4, at time t ego vehicle will have access to images from t+1 to t+40 which were received from virtual vehicle in previous two seconds of driving. We did a cross-validation and found x = 20 (i.e., the location when lead vehicle was 1 second ahead) to give the best performance. In Figure 15, we plot the training and validation loss for the case with future images. At end of epoch 7, we are able to obtain a validation loss of .0038 (corresponds to mean steering angle error of 3.5 degrees).

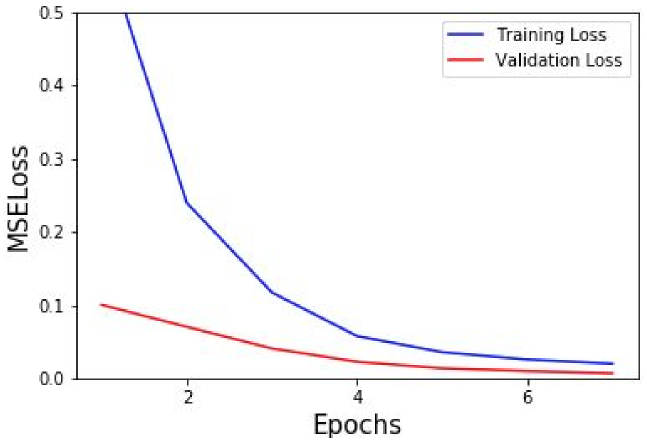
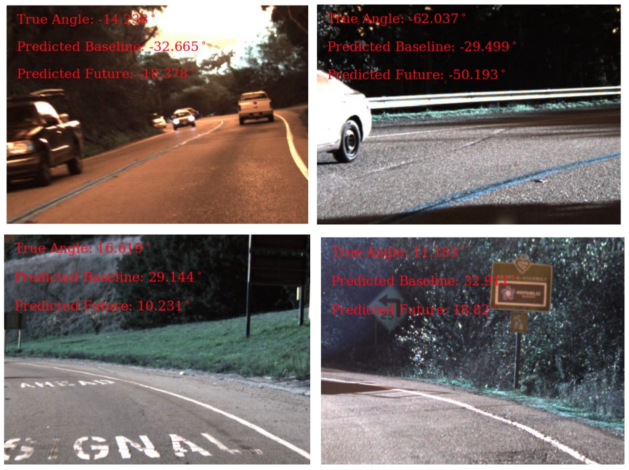


Figure 15: Loss function for the network using future images

We have provided the predicted steering angles with the future image technique in Figure 16 for the same example images with sharp turns as in Figure 14. We can see that the prediction of steering angles in sharp turns is significantly improved by using future image technique.



True Angle: -14.238 °
Predicted Baseline: -32.665 °
Predicted Future: -18.378 °
True Angle: -62.037 °
Predicted Baseline: -29.499 °
Predicted Future: -50.193 °
True Angle: 16.619 °
Predicted Baseline: 29.144 °
Predicted Future: 10.231 °
True Angle: 11.183 °
Predicted Baseline: 32.911 °
Predicted Future: 18.82 °
SIGNAL

Figure 16: Visualization of sharp turn scenarios

6. Conclusions and Future Work

In this project, we have used CNN-based 10 layer architecture to train camera data for predicting steering wheel angle for self-driving car applications. We have used various techniques for data-processing (image cropping, removing of low steering angles) and data augmentation (flipping images, using data from left and right cameras). We have extensively used Batch Normalization and Dropout layers in the CNN architecture. Our baseline network is able to train well and achieves a validation loss of .0179 (7.6 degree of mean error in steering angle). We have provided various visualization plots showing the performance and insights of our baseline network.

We have proposed a novel technique of using future images by using recent advances in 5G V2V communications. To the best of our knowledge, this is the first time V2V data has been used in deep learning architecture for self-driving cars. Using future images, we are able to obtain a validation loss of .0038 (3.5 degree of mean error in steering angle). Even though this accuracy is quite far from accuracies that would be required from deployment self-driving systems. However, we think that given the driving data-set is only of 28.33 minutes, our proposed technique of using future images with deep learning architecture shows potential of being used in practice.

Our main focus in the project has been to run the baseline network and conduct first experiments with future images. We realize that both the baseline network and future image input can be significantly optimized, and performance can be further improved. Optical flow has been used in the Udacity challenges to input previous images to the network [5]. Optical flow while using future images could be an interesting area of research. In Udacity challenges, CNN-LSTM network seem to improve performance over the CNN-based network used in this project. Future images with CNN-LSTM architecture could also be a good avenue for research.

Finally, we opensource all our scripts used in this project [15]. The dataset are not part of this repository and we refer the reader to [2] for the datasets.

References

[1] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba “End to end Learning for Self-Driving Cars”, CoRR, vol. arXiv:1604.07316, 2016.

[2] Udacity Open-source Data sets: https://github.com/udacity/self-driving-car

[3] Udacity Self-Driving Car Challenge#2:

https://medium.com/udacity/challenge-2-using-deep-learning-to-predict-steering-angles-f42004a36ff3

[4] Keras Documentation: https://keras.io/

[5] P. Weinzaepfel, J. Revaud, Z. Harchaoui, C. Schmid, “DeepFlow: Large Displacement Optical Flow with Deep Matching”, 2013 IEEE International Conference on Computer Vision (ICCV '13).

[6] J. Andrews, S. Buzzi, W. Choi, S. Hanly, A. Lozano, A. Soong, J. Zhang, “What will 5G be?”, IEEE Journal on Selected Areas in Communications, Vol. 32, Issue 6, June 2014.

[7] 2nd Rank in Udacity Challenge, Team rambo. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/rambo

[8] 1st Rank in Udacity Challenge, Team komanda. https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/komanda

[9] Udacity Challenge Final Ranking and Leaderboard https://github.com/udacity/self-driving-car/tree/master/challenges/challenge-2

[10] V2V Safety Technology Now Standard on Cadillac CTS Sedans http://media.cadillac.com/media/us/en/cadillac/news.detail.html/content/Pages/news/us/en/2017/mar/0309-v2v.html

[11] In Japan, Priuses can talk to other Priuses https://techcrunch.com/2016/08/16/in-japan-priuses-can-talk-to-other-priuses/

[12] First Toyota cars to include V2V and V2I communication by the end of 2015 http://sites.ieee.org/connected-vehicles/2015/09/30/first-toyota-cars-to-include-v2v-and-v2i-communication-by-the-end-of-2015/

[13] Intel showed how 5G networking will power VR and self-driving cars http://www.pcworld.com/article/3175745/internet-of-things/intel-showed-how-5g-networking-will-power-vr-and-self-driving-cars.html

[14] BMW: 5G is key to self-driving car deployment http://www.computerworlduk.com/it-vendors/bmw-5g-could-be-key-self-driving-car-deployment-3501253/

[15] https://github.com/choudharydhruv/cs231n_project

[16] https://en.wikipedia.org/wiki/Two-second_rule