Machine Learning Engineer Nanodegree
Capstone Project : DeepTesla

Guangwei Wang

March 15, 2017

1 Definition

Project Overview

This project is based on Course MIT 6.S094: Deep Learning for Self-Driving Cars [1].

There is an unprecedented interest, activity, and excitement in the field of intelligent vehicles [2]. The culmination of research efforts of the past decades in a broad range of disciplines, including vehicle control, robotics, sensing, machine perception, navigation, mapping, machine learning, embedded systems, human-machine interactivity, and human factors, has realized practical and affordable systems for various automated features in automobiles.

In vehicle control task, generally, the output of an obstacle avoidance system is a set of possible steering angles that direct the robot toward traversable regions. One straightforward idea is to view the entire problem of mapping input an image frame from a front camera to possible steering angles as a single indivisible task to be learned from End-to-End [3]. In this approach, the autonomous driving system is often learned from data rather than manually designed, mainly due to sheer complexity of manually developing a such system video data[4].

The End-to-End learning can be traced back to late 80's [5], in which ALVINN (Autonomous Land Vehicle In a Neural Network) with a single hidden layer neural network is designed for lane tracking. It takes images from a camera and a laser range finder as input and produces the steering wheel angle as output to follow the road. Later, a CNN (convolutional neural network) was trained to to emulate the obstacle avoidance behavior of a human driver for an off-road robots [3]. A fully integrated real-time system for autonomous off-road navigation that uses end-to-end learning from onboard sensors, operator input, and stereo cameras to avoid myopic behavior [6]. Recently, a deeper CNN [7] and CNN combined with LSTM [8] method are used for lane tracking based only on a front camera.

In all these cases, deep neural network has been found to be surprisingly effective at learning a complex mapping from a raw image to control. In this project, the End-to-End learning approach will be adopted to learn a model from Tesla dataset for steering wheel angle control.

Figure 1: Video frame snapshot

Table 1: The CSV data format

ts_micro	frame_index	wheel
1464305394391807	0	-0.5
1464305394425141	1	-0.5
1464305394458474	2	-0.5

Problem Statement

The goal of this project is to predict the steering wheel angel from Tesla dataset based on the video of the forward roadway.

Datasets and Inputs

This dataset can be found in this github [9].

There are 10 videos in this dataset, each video is encoded in H264/MKV with the resolution of 1280x720 and 30 fps. One corresponding csv file records the steering wheel angle along each frame of these videos. One video frame sample is shown in Fig. 1.

Wheel value was extracted from the in-vehicle CAN, and the corresponding steering value linking with each video frame is recorded in a csv file. The csv data format is expressed in Table. 1, in which *ts_micro* is time stamp, $frame\_index$ is frame number, *wheel* is steering wheel angle (Based on the horizontal direction, + denotes clockwise, - denotes counterclockwise).

Problem Definition

Let x_t denotes the t-th frame of the dataset, $X_t = \{x_{t-n+1}, x_{t-n+2}, ..., x_t\}$ denotes a n frames long video. $Y_t = \{s_{y-n+1}, y_{t-n+2}, ..., y_t\}$ expresses steering wheel angles associated with these

Figure 2: Typical driving scenarios

frame. Training a model to drive the vehicle can be defined as estimating the function $F : \mathbb{R}^{1280x720x3xn} \times \mathbb{R}^n \rightarrow \mathbb{R}^{1280x720x3}$ that predicts $x_{t+1} = f(X_t, Y_t)$.

Note that function $f(X_t, Y_t)$ in this definition can be approximated in a straightforward way with the proposed End-to-End method.

Evaluation Metrics

This is a regression problem, MSE metrics is adopted here.

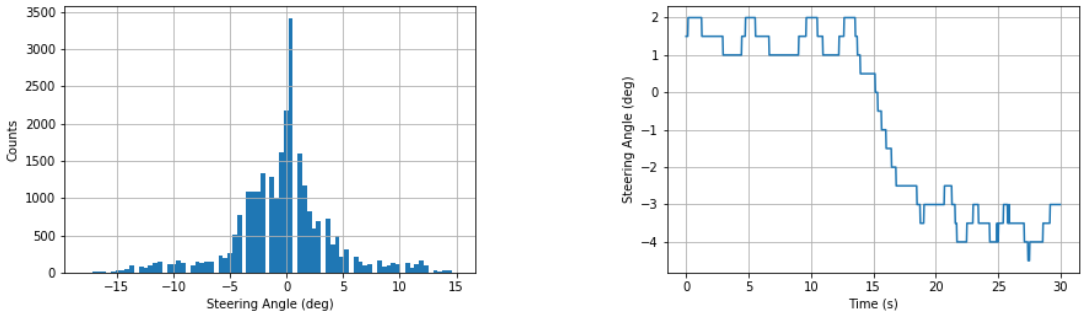
$$MSE = \frac{1}{N} \sum_{i=1}^{N} \sqrt{y_p - y} \tag{1}$$

where, y_p is the predicted steering wheel angle, y denotes the reference steering wheel angle, and the sample number is N. The lower MSE is preferred.

2 Analysis

Data Exploration

These videos are captured by a centered front-facing camera. the ROI (region of interest) lies in the lower half except for the bottom part contains vehicle head, in where it contains lane marking of the road. The upper half is the sky states. In these videos, there are different road scenarios, i.e., straight, cure, upslope and downslope, in which there contains sunny and cloudy weather. Great brightness caused by the sun glare andblurry lane lines are observed .Some typical driving scenarios are given in Fig. 2. With a glance of the steering data, there are 27,000 samples, and they are located in [-18, 15]deg.



(a) (b)

Figure 3: (a) Histogram of Steering Angle; (b) Driving operation snippet

Exploratory Visualization

The histogram of steering wheel angle is expressed in Fig. 3(a), in which we can see, it is in normal distribution. Moreover, a snippet of driving operation with 30 seconds long is shown in Fig. 3(b). The steering angle with human driving is discrete and noncontinuous, however, it cannot be considered with a classification problem due to the curse of dimension.

Algorithms and Techniques

To employ End-to-End method, we need to extract information from raw pixes firstly. CNN can succeed in the role with the computational ability increasing.

Generally, CNN consists of several convolutional and max pooling layers, which can extract various features from raw pixes, and then the high-level reasoning in the neural network will be done via fully connected layers. To increase performance of the training model, the following parameters will be tuned:

- **Preprocess parameters**: image size, ROI.
- **Hyper parameters**: number of epochs, batch size, optimizer type, learning rate.
- **Network architecture**: number of layers, layer types, layer parameters.

Benchmark

It is hard to find a benchmark for this project. Fortunately, there is an example model on the course demonstration website [9]. This network has an input layer of size (200, 66, 3) – representing a width of 200, a height of 66, and 3 channels (red-green-blue). Following that, there are three convolutional layers and pooling layers, and a single output neuron. The benchmark model architecture can be found in Table. 2.

After 10 epochs training with about 25 minutes, MSE value with 4.26 on training and 4.20 on valid data are achieved, as shown in Fig. 4.

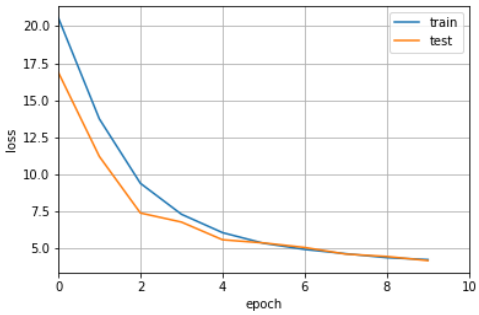


Figure 4: benchmark training results with MSE metirc

3 Methodology

Data Preprocessing

The data preprocessing code is written in preprocessing.py file and called in the DeepTesla.ipynb notebook, the following steps are included:

- Crop off 1/2 from the top and cut bottom 150px, which contains the head of car.
- Reshape images size into 66x200x3 and keep their original height/width ratio.
- Use epoch 1-9 as training data, select the remaining epoch 10 as test data. The input is each 66x200x3 pixes, and the corresponding steering angle is output.
- Normalize the input training data divided by 255.
- Split training data into train data and valid data with shuffled order.

After data preprocessing, there are 24,300 training and 2,700 test samples with 66x200x3 pixes.

Implementation

To improve performance based on benchmark model shown in above, more complex model structure is employed in this section. More filters are used in each convolutional layer to extract richer features from raw pixes. This uses Adam optimizer, and training data are shuffled in each batch. Total parameters in this model is 1,428,081. The detailed architecture is described in Table. 2.

Specially, ReLU [10] is selected as the activation function defined as $f(x) = max(0, x)$, it has been used in convolutional networks more effectively than the traditional sigmoid activation function. Dropout [11] regularization method is used in the last convolutional layer and fully connected layer for reducing overfitting. Its key idea is to randomly drop units (along with their connections) from the neural network during training.

This model is trained on a GPU Intensive workloads, which has 61G memory and 12G GPU memory. The code is implemented in python notebook with some local functions,

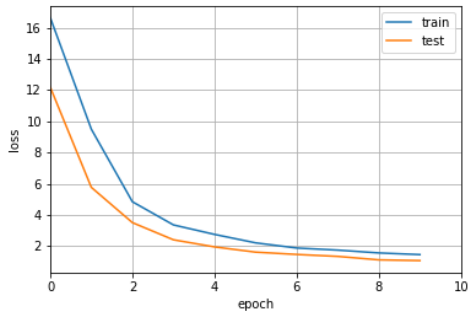


Figure 5: Training results with MSE metirc

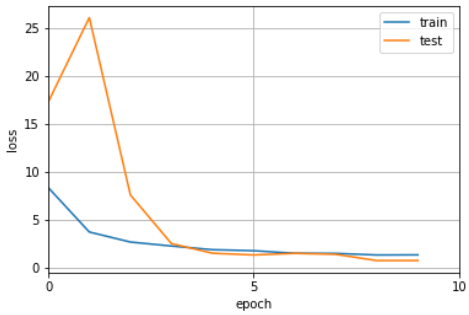


Figure 6: Refined training results with MSE metirc

which can be found in [12]. The training time is about 90 minutes With 10 epochs iterating, MSE with 1.46 on training and 1.07 on valid data are obtained, see Fig. 5.

Refinement

The following steps are done to refine the trained model further:

1. More Dropout are applied to prevent overfitting.

2. Implement He normal initializer [13]. It draws samples from a truncated normal distribution centered on 0 with stddev $= \sqrt{2/n_{in}}$, where n_{in} is the number of input units in the weight tensor.

3. Add batch normalization [14]. Normalize the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.

The refined model structure can be found in Table. 2. With about 4 hours training and 10 epochs iterating, a loss value 1.36 on training and 0.78 on valid data are achieved.

The interesting thing is the loss value on valid dataset increases dramatically on the 2rd epoch, the reason might be caused by dropout method, which stops too many neurons and the model has not adapt to the unknown variables. However, with the above methods

Table 2: Model Architecture

	Benchmark		Model		Refined Model	
	Output Shape	Param	Output Shape	Param	Output Shape	Param
Normalization	(None, 66, 200, 3)		(None, 66, 200, 3)		(None, 66, 200, 3)	
Conv-1	(None, 66, 200, 8)	224	(None, 32, 198, 32)	896	(None, 64, 198, 32)	896
					BatchNormalization	128
Maxpooling	(None, 33, 100, 8)		(None, 32, 99, 32)		(None, 32, 99, 32)	
Conv-2	(None, 33, 100, 8)	584	(None, 30, 97, 48)	13872	(None, 30, 97, 48)	13872
					BatchNormalization	192
Maxpooling	(None, 16, 50, 8)		(None, 15, 48, 48)		(None, 15, 48, 48)	
Conv-3	(None, 16, 50, 8)	584	(None, 13, 46, 64)	27712	(None, 13, 46, 64)	27712
					BatchNormalization	256
Maxpooling	(None, 8, 25, 8)		(None, 6, 23, 64)		(None, 6, 23, 64)	
					Dropout 0.25	
Conv-4	(None, 8, 25, 8)	584	(None, 4, 21, 128)	73856	(None, 4, 21, 128)	73856
					BatchNormalization	512
Maxpooling	(None, 2, 8, 8)		(None, 2, 10, 128)		(None, 2, 10, 128)	
			Dropout 0.25			
Flatten	(None, 128)		(None, 2560)		(None, 2560)	
Dense-1	(None, 1)	129	(None, 512)	1311232	(None, 512)	1311232
					BatchNormalization	2048
			Dropout 0.5			
Dense-2			(None, 1)	513	(None, 1)	513
Total params	2,105		1,428,081		1,431,217	

implemented, the loss value decreases below 2 within 5 epochs iterating, see Fig. 6, and the MSE value keeps insensible decrease, but the valid metric is not rise. It illustrates that the model has been almost converged under this model structure and overfitting problem is prevented.

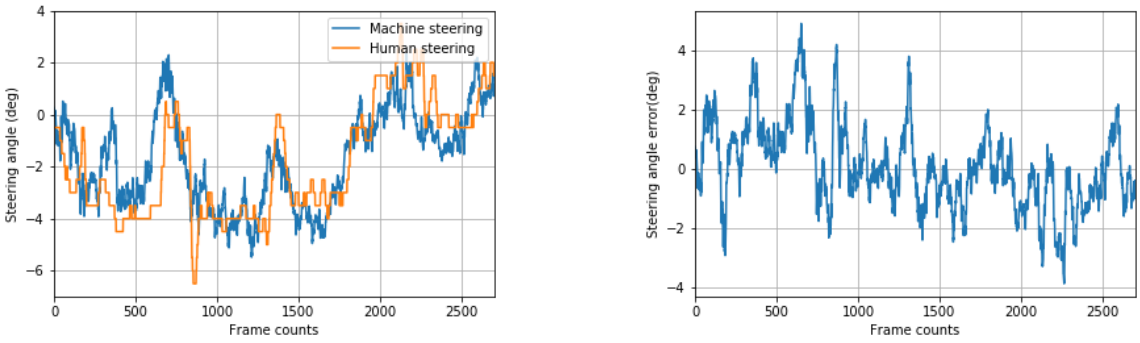
4 Results

Model Evaluation and Validation

The epoch 10 of the dataset is used for evaluating the revised model. The steering angle calculated by this trained model is compared with the human driving performance, shown as Fig. 7, from which we can see the driving trends is similar and the steering error is constrained. One thing can be confirmed is that the trained model do not make obvious mistakes at least.

Justification

To compare with the benchmark established earlier, their performances can be found in Figs. 4 and 6. It is obvious that the converge speed and final MSE value have a great improvement. However, the final solution is not convincing enough to implement this model in real world.



(a) label 1

(b) label 2

Figure 7: Test results. Left: machine steering performance compared with human driver; right: steering angle in each frame of the test epoch.

5 Conclusion

An autonomous vehicle steering control with End-to-End method is implemented in this project. CNN algorithm is adopted to extract features from raw pixes, and then the steering angle is generated with a fully connected layer. To improve performance of the trained model, several methods are implemented. i.e. ReLU, batch normalization, weight initialization, dropout. The final test results have a reasonable performance compared with human driving behavior. The most difficult thing is the burden computational task. It costs so long time (several hours) to verify each code modification. With a fine tuned parameters and decorated model structure, a better performance can be expected.

Improvement

There are still a lot to be improved further for a better steering angle tracking performance. However, they have to be presented as future work under the burden computational tasks.

1. More filters in Convnets, more epochs iterating with higher dropout value. These could extract more features from raw pixes to cope with various driving condition. for instance, line mark missing, light variance, etc.

2. The parameters in optimizer. Such as learning rate, decay rate. Higher coverage speed and stable performance can be expected with these fine tuned parameters.

3. Long short-term memory (LSTM) [15]. A recurrent neural network (RNN) architecture might work on autonomous vehicle due to driving behaviors always are smooth expect for unexpected situation, the previous condition can be referable for the next moment.

4. Image preprocessing. We can revise the brightness and contrast of input images to simulate various driving conditions.

5. generative adversarial networks (GAN). This method can be used for training driving simulator and has been obtained a beautiful results [16], in which the MSE in the order of 10^{-2} is achieved.

References

[1] L. Fridman, "6.s094: Deep learning for self-driving cars," http://selfdrivingcars.mit.edu/, 2016.

[2] E. Ohn-Bar and M. M. Trivedi, "Looking at humans in the age of self-driving and highly automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 90–104, 2016.

[3] Y. LeCun, U. Muller, J. Ben, E. Cosatto, and B. Flepp, "Off-road obstacle avoidance through end-to-end learning," in *NIPS*, 2005, pp. 739–746.

[4] J. Zhang and K. Cho, "Query-efficient imitation learning for end-to-end autonomous driving," *arXiv preprint arXiv:1605.06450*, 2016.

[5] D. A. Pomerleau, "Alvinn, an autonomous land vehicle in a neural network," Carnegie Mellon University, Computer Science Department, Tech. Rep., 1989.

[6] M. Bajracharya, A. Howard, L. H. Matthies, B. Tang, and M. Turmon, "Autonomous off-road navigation with end-to-end learning for the lagr program," *Journal of Field Robotics*, vol. 26, no. 1, pp. 3–25, 2009.

[7] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[8] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," *arXiv preprint arXiv:1612.01079*, 2016.

[9] L. Fridman, "End-to-end learning from tesla autopilot driving data," https://github.com/lexfridman/deeptesla, 2016.

[10] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[11] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[12] G. Wang, "Capstone-deeptesla," https://github.com/gwwang16/Machine-Learning/tree/master/Projects/Capstone-DeepTesla, 2017.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[16] E. Santana and G. Hotz, "Learning a driving simulator," *arXiv preprint arXiv:1608.01230*, 2016.