2017 IEEE Intelligent Vehicles Symposium (IV)
June 11-14, 2017, Redondo Beach, CA, USA

End-to-End Learning for Lane Keeping of Self-Driving Cars

Zhilu Chen and Xinming Huang, *Senior Member, IEEE*

***Abstract*—Lane keeping is an important feature for self-driving cars. This paper presents an end-to-end learning approach to obtain the proper steering angle to maintain the car in the lane. The convolutional neural network (CNN) model takes raw image frames as input and outputs the steering angles accordingly. The model is trained and evaluated using the comma.ai dataset, which contains the front view image frames and the steering angle data captured when driving on the road. Unlike the traditional approach that manually decomposes the autonomous driving problem into technical components such as lane detection, path planning and steering control, the end-to-end model can directly steer the vehicle from the front view camera data after training. It learns how to keep in lane from human driving data. Further discussion of this end-to-end approach and its limitation are also provided.**

I. Introduction

Lane keeping is a fundamental feature for self-driving cars. Despite many sensors installed on autonomous cars such as radar, LiDAR, ultrasonic sensor and infrared cameras, the ordinary color cameras are still very important for their low cost and ability to obtain rich information. Given an image captured by camera, one of the most important tasks for a self-driving car is to find the proper vehicle control input to maintain it in lane. The traditional approach divides the task into several parts such as lane detection [1], [2], path planning [3], [4] and control logic [5], [6], and they are often researched separately. The lane markings are usually detected by some image processing techniques such as color enhancement, Hough transform, edge detection, etc. Path planning and control logic are then performed based on the lane markings detected in the first stage. In this approach, its performance highly relies on the feature extraction and interpretation of the image data. Often the manually defined features and rules are not optimal. Errors can also accumulate from previous processing stage to next stage, leaving the final result inaccurate. On the other hand, an end-to-end learning approach for self-driving cars has been demonstrated in [7] using convolutional neural networks (CNNs). The end-to-end learning takes the raw image as input and outputs the control signal automatically. The model is self-optimized based on the training data and there is no manually defined rules. These become the two major advantages of end-to-end learning: better performance and less manual effort. Because the model is self-optimized based on the data to give maximum overall performance, the intermediate parameters are self adjusted to be optimal. Moreover, there is no need to detect and recognize certain categories of pre-defined objects, to label those objects during training or to design control logic based on observation of these objects. As a result, less manual efforts are required. Figure 1 compares the traditional approach with the end-to-end learning approach.

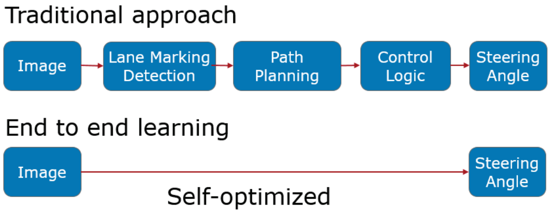


Figure 1. Comparison between the traditional approach and end-to-end learning.

This paper presents the end-to-end learning approach to produce the proper steering angle from camera image data aimed at maintaining the self-driving car in lane. The model is trained and evaluated using comma.ai dataset, which contains image frames and the steering angle data captured when driving. The rest of the paper is organized as follows. Section II provides the details of our implementation, including data pre-processing and CNN architecture. The evaluation results are presented in Section III, followed by discussions in Section IV and conclusions in Section V.

II. Implementation Details

A. Data pre-processing

The data used in this paper are from comma.ai driving dataset. The dataset contains 7.25 hours of driving data, including 11 video clips recorded at 20 Hz and some other measurements such as steering angle, speed, GPS data, etc. The image frames are of size 320 × 160 pixels, and are cropped from original video frames. The original frames are not provided by the dataset. An example of the frame from the dataset is shown in Figure 2. For lane keeping, only the image frames and the steering angle data are used. The steering angle data are recorded at 100 Hz, and they are

*Research supported by US NSF Grant No. CNS-1626236 and The MathWorks Inc. The authors are with the Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, Worcester, MA 01609, USA. The corresponding author is X. Huang (email: xhuang@wpi.edu).

aligned with the image frames using the alignment stamps provided by the dataset. In case there are multiple steering angle instances correspond to the same image frame, their average is used to form an one-to-one mapping between each image frame and its corresponding steer angle.

Figure 2. An example of image frame from the dataset.

Before training the CNN model, the data need to be further processed. First of all, to simplify the problem, driving at night is not considered in this paper and all four clips recorded at night are not considered. Second, the data contains many scenarios such as driving forward, changing lanes, making turns, driving on straight or curved roads, driving in normal speed or moving slowly in a traffic jam, etc. To train a lane keeping model, the data that meet the following criteria are selected: driving in normal speed, no lane changes or turns, and both straight and curved roads. After data selection, the remaining data are from 7 video clips with a total of about 2.5 hours. At last, five video clips containing 152K frames are used for training and two video clips containing 25K frames are used for test.

During the training stage, one important issue needs to be addressed that the data used for training is highly unbalanced as shown in Figure 3. As highway roads tend to be mostly straight and the portion of curved road is at a small percentage, the trained model based on these unbalanced data may tend to driving straight while still have low losses. To remove such bias, the data of curved roads are up-sampled by five, where curved roads are defined by where the absolute steering angle values are larger than five degrees. The data are then randomly shuffled before training.

B. *CNN implementation details*

The CNN architecture that we proposed is shown in Figure 4, which is similar to that in [7] and [8] but is much simpler. The loss layer used during training is Euclidean loss, which computes the sum of squares of differences between predicted steering angle and ground truth steering angle: $\frac{1}{2N}\sum_{i=1}^{N}||x_i^1 - x_i^2||_2^2$. The CNN model is trained using Caffe [9].

The CNN model consists of three convolutional layers and two fully connected layers. The input layer is raw RGB image, and output layer is the predicted steering angle for the input image. The first convolutional layer uses a 9×9 kernel and a 4×4 stride. The following two convolutional layers use a 5×5 kernel and a 2×2 stride. The convolutional layers are mainly for feature extraction and the fully connected layers are mainly for steering angle prediction, but there is no clear boundary between them since the model is trained end-to-end. Dropout layers are used for preventing over-fitting. There are no pooling layers because the feature maps are small.

The CNN architecture, as well as the hyper-parameters used, can be further tuned through more experiments. Over-

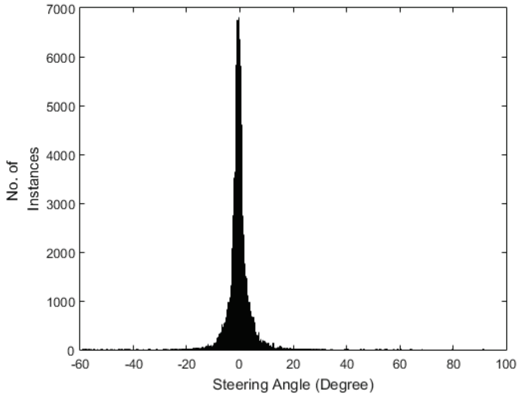


Figure 3. Histogram of steering angles in training data.

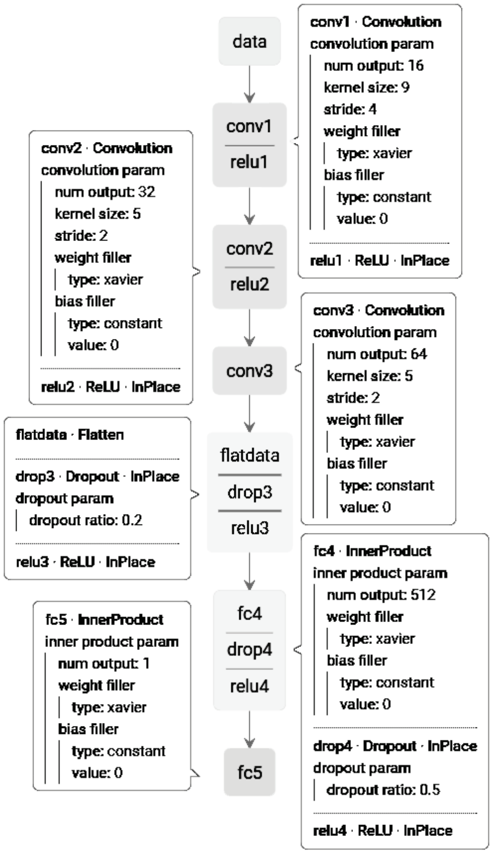


Figure 4. The proposed CNN architecture for deep learning.

all, the CNN architecture is not the major concern of this work for the following of two reasons. First, we feel the dataset is too small. Despite the training and testing data contain more than 170K frames that equals to about 2.5 hours driving, it is actually insufficient to train a generic lane keeping model that uses raw image as input. The appearance of the roads can be very complex due to different curves, road markings, lighting conditions, etc. In fact, the proportion of data for curved roads is relatively small, with only about 20 minutes of driving. Training a model that gives a continuous value as predicted steering angle, these amount of data is not sufficient. The other reason is that tuning a model requires proper evaluation metric, which is also limited by the current dataset. The details of the evaluation method will be discussed in Section IV.

III. Evaluation

The trained model is evaluated using two test video clips containing 25K frames. For each frame, the predicted steering angle is compared with the ground truth value. The histogram of the error is shown in Figure 5. The standard deviation of the error is 3.26, the mean absolute error is 2.42, and the unit is degree. To better understand the errors, the predicted angle and ground truth angle are compared in each frame and the results cam be visualized.

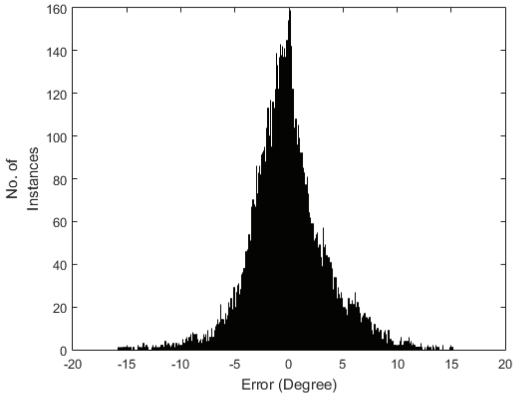
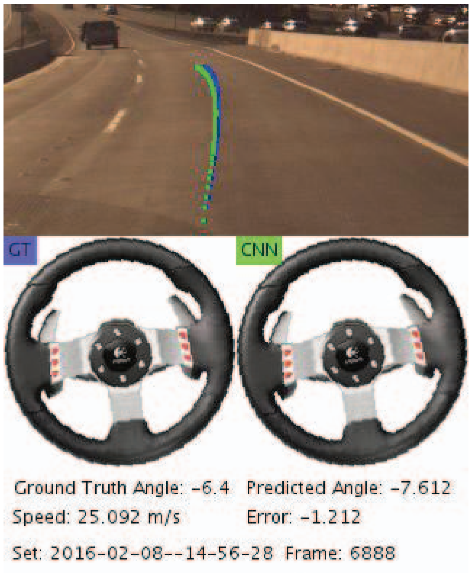


Figure 5. Histogram of error of predicted steering angles during test.

Figure 6 shows an example frame along with the ground truth angle and predicted angle. The projected paths for both angles are plotted using the same approximation as in [8]. The path using ground truth angle is in blue and the path using predicted angle is in green. The simulated steering wheels for both angles are also drawn for better visualization.

Figure 7 also visualizes the feature maps from the first two convolutional layers. The top-right 4×4 cells are results from the first convolutional layer, and the bottom 4×8 cells are results from the second convolutional layer. As expected, the convolutional layers automatically learned to extract the lane markings as a kind of feature during training. The model does not use any manually defined or hand-crafted features, since it can learn useful features from the data automatically .



Ground Truth Angle: -6.4 Predicted Angle: -7.612
Speed: 25.092 m/s Error: -1.212
Set: 2016-02-08--14-56-28 Frame: 6888

Figure 6. An example frame with the ground truth angle, predicted angle and their respective projected path

IV. Discussion

A. Evaluation

As an evaluation metric, computing the differences of ground truth angle and predicted angle is actually questionable. Firstly, the ground truth provided by the human driver is not globally optimal. The human driver cannot maintain the vehicle in the center of the lane all the time. As long as the vehicle stays in lane, the predicted angles are fine and do not have to be exactly the same as the human driver. Secondly, both the vehicle movement and the steering control are continuous, thus the frame by frame evaluation is not appropriate. Let's consider two scenarios if the road is straight. In the first scenarios, the steering angle turns to the left a bit, then quickly turns to the right a bit to maintain the vehicle in the lane. This process can be repeated. In the second scenario, the steering angle turns to the left a bit and stay at that angle for a period of time, then it turns to the right a bit and stays for a while. In the second scenario, the vehicle actually would drive out of the lane most of the time. In these two scenarios, the histogram of the errors, mean absolute error, standard deviation of the error are the same. However, the first scenario is fine while the second one is completely unacceptable. Figure 8 shows an example of the disadvantage of this type of frame by frame evaluation. The frames and their predicted angles are from the test dataset. These 5 frames are put in chronological order. We can see that the middle frame has a huge error of 10 degrees. However, the recorded ground truth does not seem correct in this frame. By looking at the previous and following frames, we find out that the ground truth in this

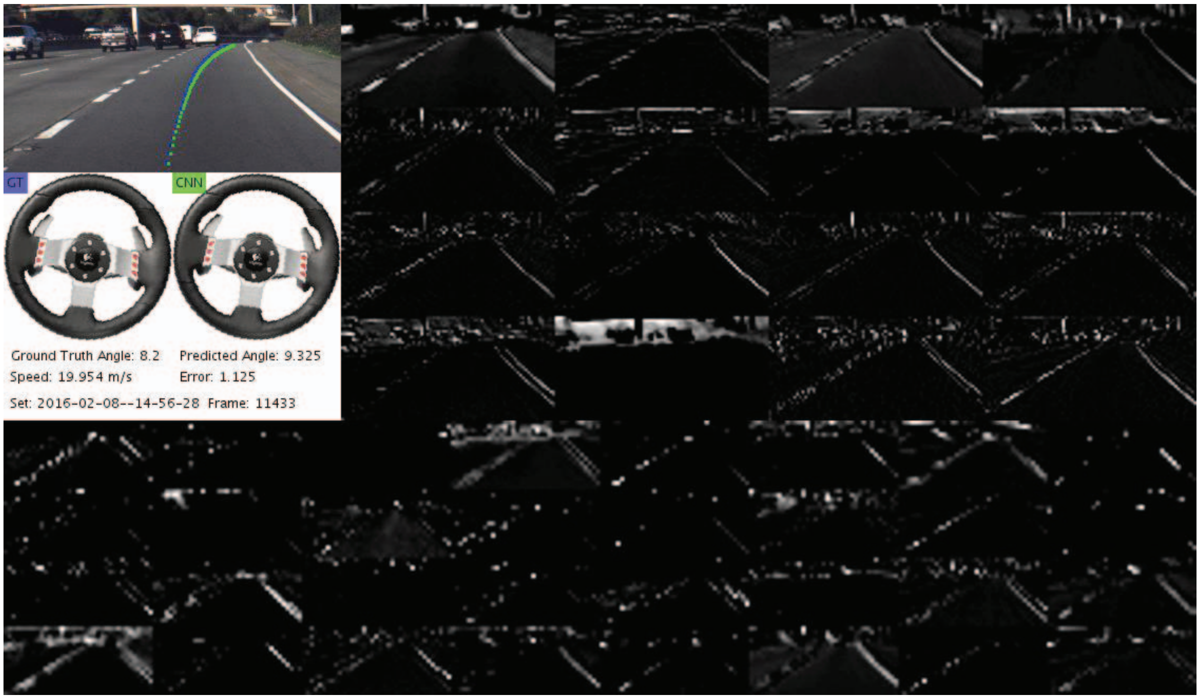


Figure 7. Visualization of the results from first two convolutional layers.

frame is transitioning from left to right. This example shows that evaluating the error frame by frame is not appropriate.

To solve this problem, a simulator is needed to provide feedback based on the predicted angle. The simulator should be able to generate the frames and simulate the vehicle movement realistically. The frames should be generated according to the vehicle position and orientation. One way to do so is using a virtual game engine, such as described in [10], [11]. The advantage of using a virtual engine is that there are built-in physics simulation and 3D rendering mechanism. The vehicle movement simulation and frame generation can be done realistically. Besides, the ground truth information is very rich in the virtual world. Information such as vehicle position, orientation and velocity can be easily obtained, so do other objects. The disadvantage is that the frames are computer generated graphics, but not real images captured from driving in the real world. Although they look very realistic with the state of the art game engine, the details and variations they provide still cannot match the data from real world.

Alternatively, we can generate the next frames according to controls inputs using recorded frames, i.e., data captured in real world. This can be achieved by either learning approach [12] or 3D image projection approach [7]. The learning approach learns auto-encoders to embedding road frames, and learns a transition model in the embedded space. The next few frames can be generated based on the current frame image and the current control inputs. On the other hand, the 3D image projection approach assumes the ground is a flat surface, and solves the 3D geometry [13] to generated the next frame based on the actual recorded frame, through predicted camera shift and rotation. The camera shift and rotation can be obtained from vehicle movement simulation, which can be computed using vehicle kinematic or dynamic models [5], [6].

B. Data augmentation

Since we are not supposed to drive off the lane when recording, the data obtained from human driving are lack of error correction process. The human driver is able to maintain the vehicle within the lane, but a model trained on such data is not robust to errors and the vehicle may slowly drift away. To train a model that can correct small errors such as vehicle shifts and rotations, the error correction data must be provided during training. One solution is to perform data augmentation by randomly creating some shifts and rotations, which generates corresponding frames based on the 3D geometry described above. The correction control input can be computed again using the vehicle kinematic or dynamic models. The comma.ai dataset does not contain the original sized frames or camera calibration parameters. Therefore simulator and data augmentation is not included in this paper. Our current work collects real world data using multiple camera. All the aforementioned techniques will be incorporated in the future work.

V. Conclusions

This paper presents the end-to-end learning approach to lane keeping for self-driving cars that can automatically produce proper steering angles from image frames captured

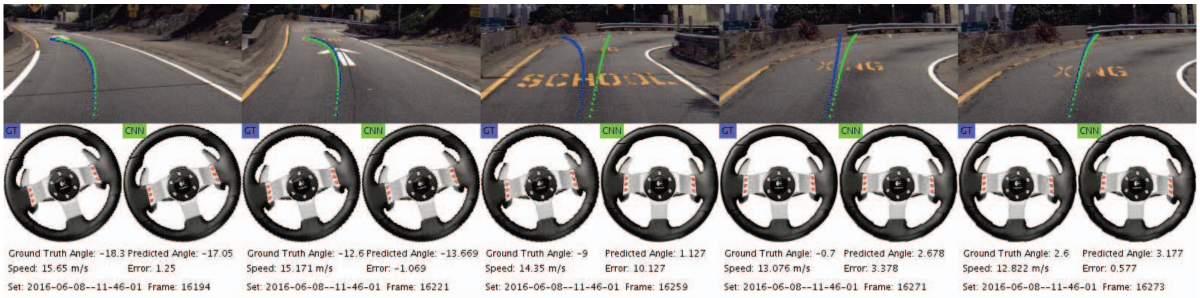


Figure 8. An example of the disadvantage of frame by frame evaluation with 5 consecutive frames: the error in the middle frame is false

by the front-view camera. The CNN model is trained and evaluated using comma.ai dataset, which contains image frames and the steering angle data captured from road driving. The test results show that the model can produce relatively accurate steering of vehicle. Further discussions on evaluation and data augmentations are also presented for future improvement.

References

[1] J. Zhao, B. Xie, and X. Huang, "Real-time lane departure and front collision warning system on an fpga," in *2014 IEEE High Performance Extreme Computing Conference (HPEC)*, Sept 2014, pp. 1–5.

[2] A. J. Humaidi and M. A. Fadhel, "Performance comparison for lane detection and tracking with two different techniques," in *2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA)*, May 2016, pp. 1–6.

[3] C. Li, J. Wang, X. Wang, and Y. Zhang, "A model based path planning algorithm for self-driving cars in dynamic environment," in *2015 Chinese Automation Congress (CAC)*, Nov 2015, pp. 1123–1128.

[4] S. Yoon, S. E. Yoon, U. Lee, and D. H. Shim, "Recursive path planning using reduced states for car-like vehicles on grid maps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2797–2813, Oct 2015.

[5] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 1094–1099.

[6] D. Wang and F. Qi, "Trajectory planning for a four-wheel-steering vehicle," in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation*, vol. 4, 2001, pp. 3320–3325 vol.4.

[7] 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. abs/1604.07316, 2016. [Online]. Available: http://arxiv.org/abs/1604.07316

[8] *The comma.ai driving dataset*. [Online]. Available: https://github.com/commaai/research

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[10] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European Conference on Computer Vision (ECCV)*, ser. LNCS, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., vol. 9906. Springer International Publishing, 2016, pp. 102–118.

[11] S. Minhas, A. Hernández-Sabaté, S. Ehsan, K. Díaz-Chito, A. Leonardis, A. M. López, and K. D. McDonald-Maier, *LEE: A Photorealistic Virtual Environment for Assessing Driver-Vehicle Interactions in Self-driving Mode*. Cham: Springer International Publishing, 2016, pp. 894–900.

[12] E. Santana and G. Hotz, "Learning a driving simulator," *CoRR*, vol. abs/1608.01230, 2016. [Online]. Available: http://arxiv.org/abs/1608.01230

[13] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.