Real-Time Control Using Convolution Neural Network for Self-Driving Cars

Woraphicha Dangskul
Department of Electronics
School of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
60010895@kmitl.ac.th

Kunanon Phattaravatin
Department of Electronics
School of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
60010120@kmitl.ac.th

Kiattisak Rattanaporn
Department of Electronics
School of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
60010101@kmitl.ac.th

Yuttana Kidjaidure*
Department of Electronics
School of Engineering
King Mongkut's Institute of Technology Ladkrabang
Bangkok, Thailand
yuttana.ki@kmitl.ac.th

2021 7th International Conference on Engineering, Applied Sciences and Technology (ICEAST) | 978-1-6654-4122-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICEAST52143.2021.9426255

***Abstract*— In this paper, we perform an Autonomous deep learning robot using an end-to-end system. The system operates as the controller for navigating and driving automatically. The deep learning robot used Convolution Neural Network (CNN). The CNN architecture is Mobile net with Softmax activation function. The Softmax activation function predicts the probability of steering angles. In the training phase, the CNN model learns from images and steering angles that are collected during the driving. In the testing phase, we apply the diversified environment to the trained CNN model. The CNN model accuracy is up to 85.03%. The results showed that the CNN is able to learn the diversified tasks of lanes and roads following with and without lane marking, direction planning and automatically control. Also, the CNN can replace the conventional PID controller.**

Keywords— self-driving robot, Deep Learning, controller, Convolution Neural Network, MobileNet

I. Introduction

Recent theoretical developments have revealed that CNN was used in diversified tasks pattern recognition is the major function of the self-driving car task. The leap of CNN is feature that learned images and controls the steering angle.[10] One primary problem to adopting CNN for predict the steering control is the regression method. The regression method is needed to improve when facing with uncertain environment. The smooth steering angle is required. In this work, we replace the regression method [10] with logistic regression. Applying CNN to steering control [10], The common problem was the stability of steering angle changing.

From previous research, in this study we deal with the above problem with using Gaussian area concept to predict steering angle. Though the prediction method is changed. But the concept such as using the convolution kernel to scan entire images still remains.

II. Relate work

In 2016, NVIDIA introduced End to End Learning for Self-Driving Cars [10]. Training CNN to map pixels from the front camera on the car straight to control steering angle. This concept has a limitation on continuously predicting steering angle. Conventional self-driving car researches generally predict steering control base on the lane marking. In the other research, Byambaa Dorj researched in Highly Curved Lane Detection Algorithms Based on Kalman Filter [1] this research base on lane tracking. Lane tracking and path planning are necessary for adopting CNN to control steering angle directly.

III. Collecting data

Training data was collected by driving the car in Fig. 2. directly from human control. The data was collected to JetsonNano in two types of data including thousands of images and steering angle. The procedure of data collection were collected from many driving situations with and without lane markings.

A. Images

The image was collected and then converted color space from BGR to RGB and resizing the image from 1280×720 pixels to 160×160 pixels. Overall process focuses on reducing size of the image corresponding to the frame rate of the self-driving car system.

B. Steering angle

The recorded steering angle was transformed into 5 classes are "Left", "MidLeft", "Forward", "MidRight" and "Right". Each class is divided into intervals of 30 degrees per class see in Fig. 3.

IV. CNN Architecture

A. CNN Description

We trained the weights of our network to minimize root mean square between output from network and practical value. The network was shown in Table. I

The model was designed for resolution 160×160 pixels. Mobile net performs feature extraction. We customize the prediction layer by using Softmax activation function. Softmax activation function predicts the probability class. The prediction layer had 5 nodes with Softmax activation function.

Training process compiled with learning rate 0.001 and categorical cross entropy loss function.

TABLE I. CNN Architecture. The network has 1.8 million parameters

Mobilenet
Input layer
Convolutional layer
Depthwise n Convolution layer
Batch Normalization
ReLU
(n = 1,2,3,…,13 layers)
Global Average Pooling layer
Reshape layer
Dropout layer
ReLU layer
Reshape layer
Softmax layer
Reshape layer
Output

* Corresponding author.

978-1-6654-4122-3/21/$31.00 ©2021 IEEE

V. Self Driving Control Architecture

A. Controller

The conventional controller such as PID controllers may lead to poor handling performance because they need to set up with the specific features, hardware parameters. On the other hand, the specific features for CNNs are not required. Currently the CNN has end-to-end systems e.g., feature extraction, automatically feedback adjustment. So, the CNN can be used in place of a conventional controller.

The related work in [10] describes that a CNN is able to go beyond pattern recognition. It learns the entire processing pipeline needed to steer an automobile. The difference is the CNN does not require specific human designed features such as lane marking, guard rails, obstacle identification system.

In Fig. 1., the CNN block diagram below is applied to replace the conventional controller. Furthermore, the CNN can apply on real time processing with or without lane marking.

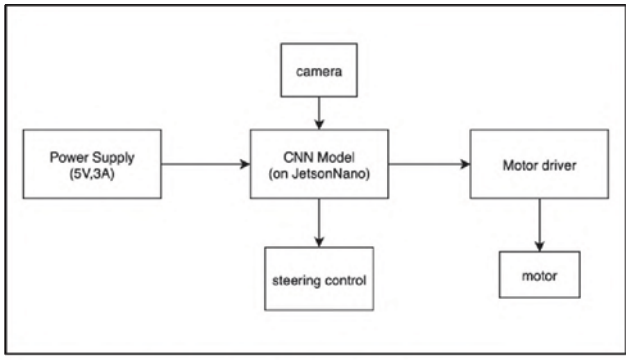

Fig. 1. The system block diagram

B. Hardware optimization

A processor with high processing speed and high performance to communicate and respond rapidly is needed in self driving car for driving and steering motor control. So we design the end-end system which requires the power about 15 Watts.

- The processor for the self-driving car is the Jetson Nano, the high-speed processor. The Jetson Nano is used to compute CNN models through an Nvidia GPU.
- The controller selected for building the self driving car is the STM32, which is a relatively large industrial controller supported by the Mbed Studio. The main function of the controller is communication between the motor drive system and JetsonNano.
- The motor driver chip for building the self driving car is the L293D. It can be used to drive four small DC motors and a servo motor. Using this chip can typically power up to 36 volts. Also the chip can supply a maximum current of 600mA per channel.
- The power supply for the self driving car is an 11.4 Volt 2200 mAh Lithium polymer battery (Lipo battery), which supplies power to all systems. All systems are placed on the car shown in Fig. 2. There are camera, JetsonNano, Printed Circuit Board (PCB) for the controller and power management built-in, and Lipo battery.

Fig. 2. The self-driving car when applying all systems and ready to drive.

VI. Methods and Experiment

A. Experiment set up

First, the Lipo battery is applied to the self driving car and set up an alarm system to detect when the voltage in each battery cell was lower than the threshold. Next step the joystick must be connected to the self driving car via Bluetooth protocol. PCB is designed to combine all systems, including the motor driver, power management system, servo control system, and the main processing unit. The steering angle was initially started at 0 degree.

B. Data collection

A data set, captured image, the steering angle is collected while controlling the car through a joystick. Thousands of images are saved into JetsonNano that runs on Ubuntu 14.0 operating system. The range of servo motor angle is between -70 to 70 degrees. In this section, the designed software is used to collect the angle for each image. The steering angle is attended into Comma Separated Value (CSV) files.

C. Output data for classification

The steering angles collected from a joystick are in between -75 to 75 degrees. We will segment the steering angle corresponding with image data into 5 classes shown in Fig. 3. below. These five classes and their centers are shown in Table. II. This step was calculated on Google Colab notebooks.

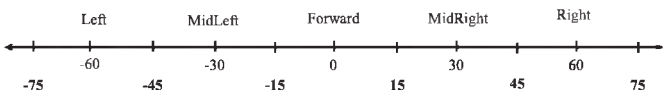

Fig. 3. the intervals to mapping angles to the classes.

D. Image preprocessing

To increase the self driving car performance, the image was resized to require fewer the storage on JetsonNano. The CNN was designed with 160 x 160 pixels size for pushing the higher frame rate as fast as possible. The image normalization has been done before applied to the model.

E. Data pre processing

We shuffled data from CSV file that contains the image path and steering angle. The data is divided into training and testing set with the ratio of 80% and 20% respectively.

F. Training model

We train the network about 128 epochs. From Fig. 4. the accuracy of the training dataset is 94.09% and the accuracy of the testing dataset is 85.03%.

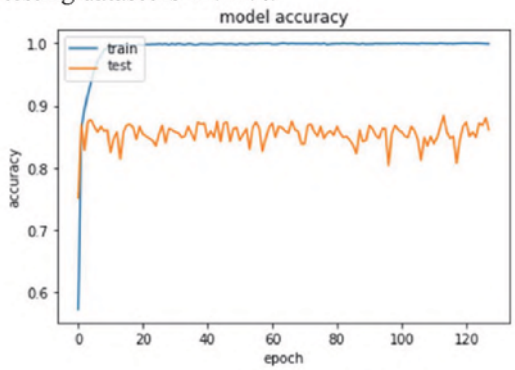

Fig. 4. The output accuracy of 128 epochs

G. Experiment Results

The image was applied to the trained model that has acceptable performance. We test the experiments in 2 phases, software simulation and on road testing. The CNN model has the softmax activation function, the purpose of these activation function is used to predict probability of each classes (P_i).

In the first phase, the images were applied to the CNN in order to obtain the probability of each class (P_i) corresponding to the steering class angle. The prediction angle from each class is calculated by equation (1). This equation is used to compute the output steering angle by multiplying the probability of each class (Pi) with degrees (Wi) corresponding to Table. II. If the training data is unbalanced such as the number of right class images is greater than the other classes. Degree (W_i) should be adjusted shown in Table. II. The trained model

$$\text{Angle (degree)} = \sum_{i=0}^{4} (P_i \cdot W_i) \qquad (1)$$

In the second phase, the trained model from the first phase is embedded into the car and takes out for the road test. The challenge of this method was the situation when the car was facing with the road corner. We tested the trained model with images in Fig. 5. The predicted steering angle are 67,68,69 degrees respectively. The results encourage that the CNN is able to learn diversified tasks of lanes and roads following with and without lane marking.

The accuracies were calculated from the training and testing phase which are 94.05% and 85.03% respectively. Comparing with the work from Chishti. S. in 2018 [4], the CNN model was trained with 89% accuracy and 73% test accuracy.

Left picture Engr Akram, Daulat Beg Oldi 2020, Daulat Beg Oldi: World's Tallest Airstrip, India's Pride, China's Eyesore. <https://www.myishasmehfil.com>. Digital Image.

Fig. 5. The Unseen environment image (Left)
the unseen environment image without lane marking (Middle)
the unseen environment image with lane marking (Right)

TABLE II. CLASS MEANING AND GAIN FOR MAPPING TO PREDICTED ANGLE

Class (Probability)	Meaning	Degrees (Wi)
Class 0	Left	-60
Class 1	MidLeft	-30
Class 2	Forward	0
Class 3	MidRight	30
Class 4	Right	60

VII. Conclusion

This paper used CNN as a controller. There is no theoretical comparing between using conventional controllers and the CNN in self-driving car, but the similarity between them is their feedback system but in different types. The conventional ones obtain the errors from measured data while the errors in CNN are not directly fed back from the output errors, but they are hidden in images captured from the camera. The correction from the images behaves like human controlling. The experiment results in Fig. 5. proved that the CNN model can replace the conventional PID controller. The CNN is able to learn the diversified tasks of lanes and roads following with and without lane marking, direction planning and automatically control. We have empirically demonstrated that the car is sufficient to operate in diverse environments. To improve performance from the previous works [10],[4],[5] which are the regression model, this work used the classification model instead. The steering angle outputs were smoothed by the Softmax activation function. In addition, error standard of self driving car might provide an important area for future research.

Acknowledgment

In this project, we would like to thank Robot Club Engineering at KMITL for the mechanism suggestions and hardware knowledge.

References

[1] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, and H. Adam, (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[2] C. C. J. Kuo, (2016). Understanding convolutional neural networks with a mathematical model. Journal of Visual Communication and Image Representation, 41, 406-413.

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

[4] S. O. Chishti, S. Riaz, M. BilalZaib, and M. Nauman, (2018, November). Self-driving cars using CNN and Q-learning. In 2018

IEEE 21st International Multi-Topic Conference (INMIC) (pp. 1-7). IEEE.

[5] M. Muratov, A. Mehar, W. S. Lee, M. Szpakowicz, O. E. Umolu, J. M. Bobadilla, and A. Kuwajerwala, (2020). Experiments in Autonomous Driving Through Imitation Learning. arXiv preprint arXiv:2011.12460.

[6] J. del Egido Sierra, L. M. B. Pascual, E. R. Carmena, C. G. Huélamo, J. A. Ruiz, and R. B. Navarro, Self-Driving a Car in simulation through a CNN.

[7] D. Ansari, and G. Kochar, (2019, November). Simulation of Steering a Self-Driving Car Using 1) PID Controller 2) Neural Network. In 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT) (pp. 212-218). IEEE.

[8] D. Scherer, A. Müller, and S. Behnke, (2010). Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. ICANN.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

[10] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, and K. Zieba, (2016). End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316.

[11] N. Fernandez, (2018). Two-stream convolutional networks for end-to-end learning of self-driving cars. arXiv preprint arXiv:1811.05785.