End-to-end autonomous driving based on the convolution neural network model

Yuanfang Zhao and Yunli Chen
*Beijing University of Technology, Beijing, China
E-mail: zhaoyuanfang@bjut.edu.cn
E-mail: yunlichen@bjut.edu.cn

***Abstract*—In the control algorithm of autopilot system, the Deep Learning method plays a vital role. Since the convolutional neural network (CNN) model used in automatic driving has a huge amount of parameters and the training results are prone to over-fitting, an excellent model is necessary. In this paper, an end-to-end control method was proposed to apply a convolutional neural network with a new network structure to control the steering angle and speed of the vehicle and reach the goal of automatic vehicle driving. The experimental results show that it not only greatly reduces the number of parameters, but also keeps the error rate of the experimental results at the low level.**

I. INTRODUCTION

The field of artificial intelligence develops rapidly, among which autonomous driving is the research content that people pay close attention to. The typical methods of autonomous driving technology include three schemes. The first method is indirect perception. This method includes target detection, target tracking, scene semantics segmentation and correction, three-dimensional reconstruction and other computer vision sub-tasks. By synthesizing the detection results of multiple sub-tasks, a complete environment representation is established. The advantage of this method is that the module is clear and the coupling degree is low, while the disadvantage is that the redundancy is high, which will result in waste of resources. The second scheme is direct perception, which optimizes indirect perception and learns the key indicators of driving directly. This model simplifies the operation, but it has some limitations in unstructured traffic scenarios. The third scheme is end-to-end control. It mainly uses in-depth learning technology. The aim of end-to-end control method is to establish a direct mapping relationship between sensor and driving action. The end-to-end structure achieves good results. This method directly uses the learning method to supervise and learn driving action. Fig. 1 illustrates how the basic end-to-end solution works.

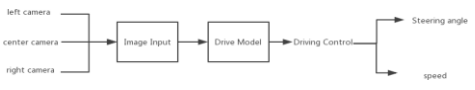

Fig. 1: Basic end-to-end structure

These three typical methods of automatic driving have some representative results. At present, the three methods are constantly being studied and developed. In this paper we adopt end-to-end structure. It is unavoidable to use the neural network model in the end-to-end control of this automatic driving method. In recent years, a variety of neural network models have been developed and widely used, among which are AlexNet [1], VGGNet [2], GoogleNet [3], ResNet [4] and so on. They are the most typical models. Current research [5- 15] is mostly based on the improvement and innovation of these models.

We propose a new deep model learning architecture for autopilot technology based on the traditional learning model. Our learning model is universal because it learns the predicted future motion path given the current proxy state. Finally, we use end-to-end Control method and deep learning algorithm to achieve automobile driving. The number of parameters can be greatly reduced, and the parameters can be effectively transferred and used through the improved neural network model. Our paper has made two new contributions: Firstly, we designed a novel CNN model, which has a simpler model structure, a smoother operation process and can output multi-mode prediction. Secondly, we produce large data sets in order to learn general motion models from vehicles with heterogeneous actuators.

The rest of the paper is organized as follows: In Section II, the related work on automatic driving are analyzed. We present the proposed work of our model in Section III. In Section IV, the criteria for data sets, parameters of the model and the test results are given, and finally, we conclude the paper in Section V.

II. RELATED WORK

Samples were collected by camera [6]. One of the cameras is used to match the input image with the steering angle. The other two cameras are used to input negative samples to correct vehicle deviation. Referring to the VGG16 of CNN model, the automatic driving operation is completed. Mainly based on the data provided, paper [16] predicts and simulates the driver's behavior in the training data under the given current state. They proposed two methods: automatic encoder to reduce dimensionality and RNN conversion learning. However, this regression method is difficult to understand the

different turning situations in the same scenario. There is still a unique method, which is different from the previous two methods [6,16]: by putting forward a set of scene description indicators which can be used for automatic driving, and by accurately learning the values of these indicators, the final decision can be made [7]. The framework used in this paper is CNN model based on Caffe. However, due to the complexity of environmental factors, the calculation dimension is high and the redundancy is high. Paper [8] adopted FCN-LSTM architecture, in which FCN uses AlexNet pretrained on ImageNet, removes pooling5, and uses dilated convolutions for FC6 and FC7. Semantic segmentation is used as side-task, and the performance of the model is improved. In recent years, Deep Learning-based deep reinforcement learning method is used to complete automatic driving [17-20]. This method fully considers the relationship between agent and environment. Unlike CNN, its output is not a direct decision- making state, but a reward for the corresponding behavior of output decision-making. These methods are innovations on CNN model, but they do not solve some drawbacks of CNN model itself.

From the past research results, it is wise to apply CNN to automatic driving. But up to now, the CNN network model used for automatic driving takes a long time in the training process, and is prone to over-fitting, which reduces the training accuracy. Usually dropout, regularization and other methods are also needed to reduce the error rate in the training process which increases a lot of calculation process. In our experiments, we will use a new network structure to avoid these problems.

III. PROPOSED WORK

We first describe the overall advantages of the global average pooling layer in the automatic driving model, and then we propose a specific novel convolutional neural network architecture for automatic driving.

Our experimental model is based on AlexNet model. The global average pooling layer(GAP) is used to replace the most of the full connection layer(FC).

The generalization ability of the whole network model is affected by the full connection layer. In this paper, we use a strategy called global average pooling layer to take the place of full connection layer in our driving model. This layer appears after the last convolution operation, so the feature of black box in the full connection layer is removed directly, and the actual category meaning of each channel is given directly. The global average pooling layer is mainly used to solve the problem of full connection. The feature map of the last layer is pooled to form feature points. These feature points are composed into the final feature vector for softmax calculation. Feature mapping can be easily interpreted as related category mapping. The advantage of the global average pooling layer is that the relationship between each category and feature map is more intuitive, feature map is easier to be converted into classification probability because there is no parameter to be adjusted in GAP, and the over-fitting problem is avoided; GAP aggregates spatial information, so it is more robust for spatial conversion of input.

Our network architecture is shown in the fig.2. The network layer consists of seven layers, including one normalization layer and five convolution layers. The last pooling layer in the convolution layer is the global average pooling layer. At the end of the model, we still need a full connection layer to integrate all the features.

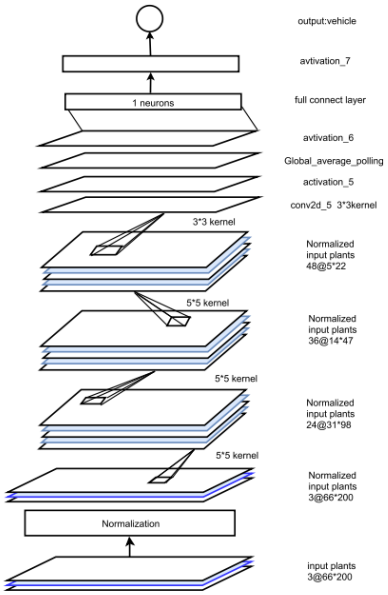

Fig. 2: proposed CNN architecture.

The first layer of network structure is to pack our graphics data into "layer" objects. In data normalization with

$$x=x(x-min)/(max-min) \quad (1)$$

Max and Min are the maximum and minimum values in the whole data, and X is the current pixel value to be converted. The implementation of standardization in the network allows the standardization plan to be changed through the network structural and accelerated through GPU processing.

We use step-wise convolution in the first three convolution layers, including 2 *2 strides and 5 *5 cores, and 3 *3 non- step convolution in the last two convolution layers.

In this layers. For example, to take a two-dimensional image I as input, we need a two-dimensional convolution kernel K:

$$S(i,\ j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \quad (2)$$

We use the local receptive field and weight sharing characteristics of convolutional neural network to enhance some features of the original signal, reduce noise and obtain local receptive features. The pooling layer is connected behind the convolution layer for down-sampling. On the one hand, the resolution of the image can be reduced and the number of parameters can be reduced. On the other hand, the robustness of translation and deformation can be obtained.

Following the five convolution layers, the final stage is the global average pooling layer, whose work is to take the global average of the generated feature map. The operation structure is shown in Fig. 3. The global average of the feature map is used to output a value, which is to change a tensor of W * H * D into a tensor of 1 * 1 * D.

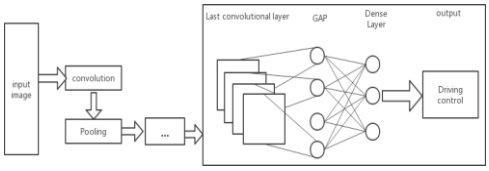

Fig. 3: The data flow of CNN with GAP

IV. EXPERIMENT

The main purpose of our experiment was to reduce the number of parameters and increase the speed of the operation while ensuring the accuracy of the experiment. In this section, we first prepare the criteria for data sets, and then use a deep learning model based on heuristics and standard depth. Additional experiments have been carried out to assess the usability of our model. In addition, we demonstrate the excellent capability of our model to process data. Finally, we use a car to demonstrate the performance of our model in a simulated environment.

A. Data set

Our dataset is from the simulation platform. The platform is Udacity. It contains two operation scenarios, i.e. Training and Autonomous. We collect data under the Training interface and verify the fluency of the model under the Autonomous interface. We get the data from three directions of the car through the simulation vehicle, which are from the front, the left and the right, just like the fig. 4. It includes many driving scenes, such as lake scenes, jungle scenes and so on. We use different driving scenarios to form different data sets. Different datasets have different sizes and goals.

Fig. 4: The input data from three directions of the vehicle, they are left, middle and right, respectively.

A large number of datasets from the simulation platform can be collected, and then according to the dataset the universal driving model is learned and trained. For different training scenarios, datasets are collected by repeated operations. We make the amount of data enough to support the experiment to train a precise model. We will collect the data in two formats: picture format and data format. It is convenient for data importing and model learning.

Compared with other experiments, our experimental data format is simple, no data enhancement and other operations are needed, which simplifies the experimental process. In addition, the accuracy of the data also has a very important impact on the accuracy of the model, so it is necessary to ensure that there is no dirty data in the training data and that every picture follows the correct driving strategy.

B. Selection of experimental parameters

In the experiment, CNN is trained by the extended Adam algorithm of stochastic gradient descent(SGD) algorithm. Adam algorithm is superior to traditional SGD algorithm in that Adam designs independent adaptive learning rates for different parameters by calculating the first and second moment estimates of gradients. During the training process, each batch contains 20032 images, and the total number of iterations (epoch) is 8. We select ELUS as the activation function. Elus activation function, like ReLU function, has no parameters. The converges of ELUS function is faster than ReLU function. Better results can be achieved without batch processing than with batch processing when using ELUS function. The ELUS function without batch processing is more effective than the ReLU function with batch processing. The ELUS function is given as follows:

$$f(x) = \begin{cases} x(x \geq 0) \\ a(e^x - 1)(otherwise) \end{cases} \quad (3)$$

when x is less than 0, more complex transformations are made.

C. Comparison with the model based on FC

After the model training, we get the change of loss and validation loss as shown in Fig. 5. Both of parameters descend smoothly. It shows that our model has been learning effectively without using Dropout and regularization methods, and the occurrence of over-fitting is avoided effectively.

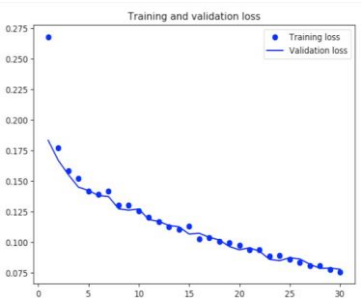

Fig. 5: The training and validation loss of the model with GAP layer

The method proposed in this paper with global average pooling layer is called model B. Comparing with the model in this paper, the convolutional neural network with full connection layer is called model A. Figure 6 shows the loss of model A and model B. Figure 7 shows the variation of error rates under different neural models.

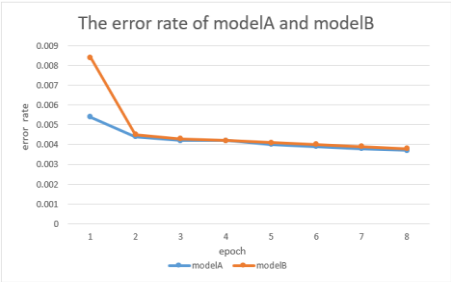

Fig. 6: The error rate of model A and model B in 8 epochs of training is shown.

Fig. 7: The loss of model A and model B in 8 epochs of training is shown.

From fig. 6, we can see that the error rate of network model B and network model A is almost the same under the same iteration times, but model B has a faster error reduction under the global level pooling layer, which indicates that model B has a stronger generalization ability. In fig. 7, Compared with the model with full connection layer, the loss value of our model is in a steady state of decline.

Table 1 lists the parameters, model file size, training time per step and error rate of network models A and B under various experimental conditions. As can be seen from table 1, the training time of model B is slightly better than that of model A, and the computational complexity of model B is smaller in terms of the computational complexity in the training stage. In terms of model file size, model B is reduced by 30% compared with model A.

	parameters	Model file size /(KB)	Training time per step/(ms)	Test Error
Model A	2116983	9	228	0.37%
Model B	131413	6	219	0.38%

TABLE1. model A compare to model B

D. Visualization

We input data into the CNN model to get the response of neurons to the input image. We can visualize these outputs, and the visualization results are shown in Fig. 8. We can get some filling block and boundary characteristics after the first layer convolution calculation. The convolution kernel basically retains all the information of the image at this stage. The middle layer learns some texture features so that the features are closer to the classifier level and the features needed for driving decisions can be obtained. With the increase of layers, the output content of convolution core becomes more and more abstract and less information is retained.

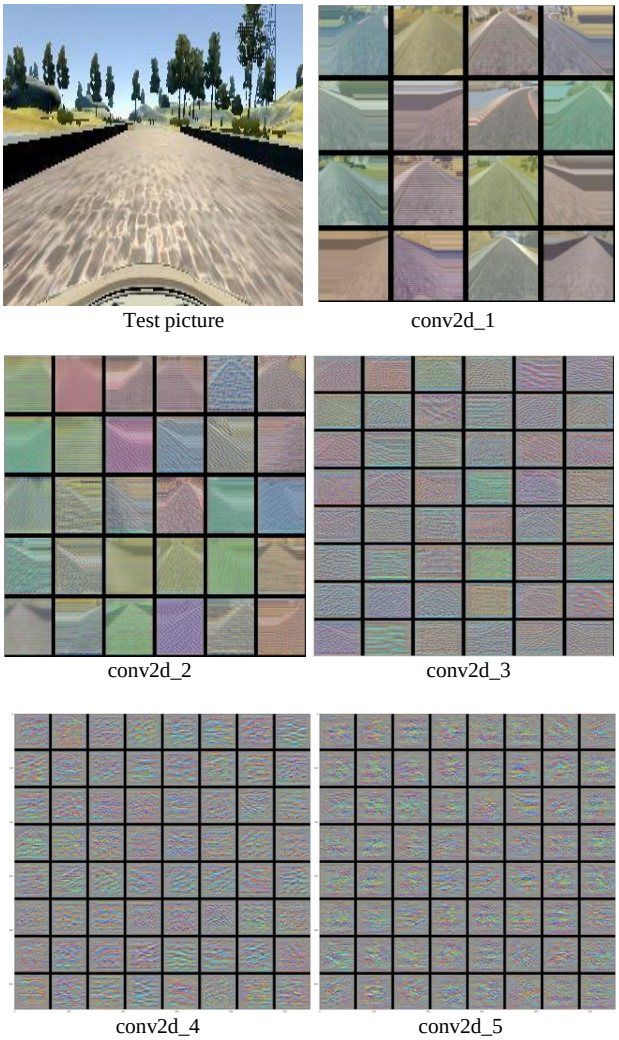

Fig. 8: Visualizing convnet filters. The first one is the road test map, followed by Filter patterns for layer conv2d_1 to conv2d_5.

We visualize class activation maps (CAM) to determine the importance of each location to each category. Since the original road information is retained in the graph, we can superimpose the heatmap with the original image to determine the strong response of neurons to those road markers when estimating. From Fig. 9, we can see that neurons respond strongly to the location of road markings and obstacles. It indicates that they have learned to avoid these obstacles when making decisions. The thermodynamic response is more and more intense with the deepening of training.

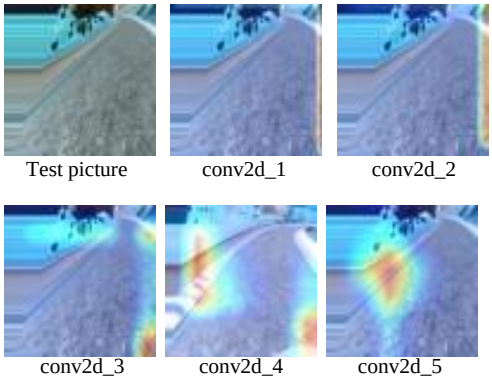

Fig. 9: The CAM of our experiment. The first one is the road test map, followed by the class activation thermodynamic map of the first to fifth test images.

To verify the generality of our approach, we tested different scenarios other than datasets (see fig. 10). The speed and steering angle of the simulated vehicle are predicted based on the model whether in lake or jungle scenarios. Although our model is actually trained under a unified standard data set, it shows robustness to random scenes. Our car has made correct driving decisions under different obstacles and different environmental conditions. The availability of our model is fully proved.

Fig. 10: The result predicted steering angle and speed of car after training. The first three pictures are about the operation of the lake field under different road conditions, while the last three pictures are about the operation of different road conditions in the jungle environment.

V. CONCLUSION

Experiments showed that our CNN can accomplish lane detection and real-time path planning very smoothly. In this paper, we introduce a method of learning general-purpose driver model for large data sets with end-to-end trainable architecture. The car can complete the task of automatic driving in different environments by processing images. CNN can learn meaningful road features from the data. For example, systematic learning detects road contours during training without requiring explicit labels. More work is needed to improve the efficiency of model training and the convergence speed in the training process.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in International Conference on Neural Information Processing Systems, 2012.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Computer Science, 2014.

[3] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015, pp. 1–9.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016, pp. 770– 778.

[5] M. Liu, J. Niu, and X. Wang, "An autopilot system based on ros distributed architecture and deep learning," Jul 2017, pp. 1229–1234.

[6] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, and K. Zieba, "End to end learning for self-driving cars," 2016.

[7] C. Sun, J. M. U. Vianney, and D. Cao, "DeepDriving: Learning Affordance for direct perception for autonomous driving," 2019.

[8] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," Jul 2017, pp. 3530–3538.

[9] J. Kim, G. Lim, Y. Kim, B. Kim, and C. Bae, "Deep learning algorithm using virtual environment data for self-driving car," Feb 2019, pp. 444–448.

[10] W.-Y. Lin, W.-H. Hsu, and Y.-Y. Chiang, "A combination of feedback control and vision-based deep learning mechanism for guiding self-driving cars," Dec 2018, pp. 262–266.

[11] R. Aihara and Y. Fujimoto, "Free-space estimation for self-driving system using millimeter wave radar and convolutional neural network," in 2019 IEEE International Conference on Mechatronics (ICM), vol. 1, March 2019, pp. 467–470.

[12] N. Fanani, M. Ochs, A. Sturck, and R. Mester, "Cnn-based multi-frame imo detection from a monocular camera," Jun 2018, pp. 957–964.

[13] N. Patel, A. Choromaska, P. Krishnamurthy, and F. Khorrami, "Sensor modality fusion with cnns for ugv autonomous driving in indoor environments," Sept 2017, pp. 1531–1536.

[14] Y. Hu, W. Zhan, and M. Tomizuka, "Probabilistic prediction of vehicle semantic intention and motion," in 2018 IEEE Intelligent Vehicles Symposium (IV), June 2018, pp. 307–313.

[15] P. Viswanath, S. Nagori, M. Mody, M. Mathew, and P. Swami, "End to end learning based self-driving using jacintonet," Sept 2018, pp. 1–4.

[16] E. Santana and G. Hotz, "Learning a driving simulator," March 2016.

[17] T. Okuyama, T. Gonsalves, and J. Upadhay, "Autonomous driving system based on deep q learnig," in 2018 International Conference on Intelligent Autonomous Systems (ICoIAS), March 2018, pp. 201–205.

[18] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," 2016.

[19] M. Wei, S. Wang, J. Zheng, and D. Chen, "Ugv navigation optimization aided by reinforcement learning-based path tracking," IEEE Access, 2016.

[20] E. Bejar and A. Moran, "Backing up control of a self-driving truck-trailer vehicle with deep reinforcement learning and fuzzy logic," Dec 2018, pp. 202–207.