International Core Journal of Engineering Volume 6 Issue 3, 2020
ISSN: 2414-1895 DOI: 10.6919/ICJE.202003_6(3).0030

End-to-end Learning Based Behaviror Clone

Honghui Yuan

School of Shanghai Maritime University, Shanghai 201306, China

yhh@mpig.com.cn

Abstract

In recent years, although the topic of unmanned driving continues, it is still a challenge to realize unmanned driving in an unknown environment. In order to realize auto driving in an unknown environment, FCN-BiLSTM framework was proposed and end-to-end training was adopted. Road images collected by the simulator were input into the network, and the data of the steering Angle of the car was output, with the deep learning network in the middle, so as to control the car simulator and realize the smooth driving of the car. BiLSTM network is composed of forward LSTM network and backward LSTM network, which is used to deal with the dependence between before and after images. Experimental results show that in Udacity open source unmanned driving simulator, the combination of convolutional neural network and bidirectional long and short-term memory neural network provides an effective scheme for the control of unmanned driving.

Keywords

End-to-end; Behavioral cloning ; Automatic drive; BiLSTM.

1. Introduction

At the beginning of 2016, the first upsurge in the field of automatic drive was set off in China in more than 20 years. The accumulation of technologies in the past few decades, together with the current wind gap of artificial intelligence, led to the development of automatic drive technology.

Automatic drive is the technology that allows the car to have its own environmental awareness, path planning and autonomous vehicle control, namely the imitation of human driving or autonomous driving conducted by the computer.

The traditional autonomous driving method is to find a collision free path from the initial state to the target state according to certain evaluation conditions in a dynamic environment with obstacles.The decision-making of autonomous vehicle includes global navigation planning, driving behavior decision-making and motion trajectory planning.However, end-to-end deep learning does not require manual disassembly of the problem. It only requires the acquisition of road condition images through the car's camera, and then recording the control parameters of the car. The image and control parameters are taken as input to train the deep neural network, and finally the control parameters are output to achieve the purpose of driving. In fact, automatic drive based on end-to-end learning is a very simple automatic drive model. It ignores the data collected by other sensors other than the camera and only makes operational decisions based on the camera data.

The earliest attempts at end-to-end driving models date back at least to the ALVINN model of 1989[1]. NVIDIA lighter with a three layers of back propagation neural connection network, using the simulation training road map, the input of monocular camera images, radar data, and the intensity of a feedback, output a vector indicating the direction and the intensity of the input to the next moment feedback, which makes the vehicle can go along the road. [2] proposed the direct perception method to estimate the driving affordability.A simple controller can drive itself by learning to map the input image to perceptual indicators related to the affordability of road or traffic conditions, such as the car's Angle relative to the road, the distance to lane markings, and the distance to cars in the current lane and adjacent lanes. [3] the input is the image obtained by the camera on the unmanned vehicle, the output is the steering Angle, and the learning is conducted through a ConvNet in the middle.But it can only be driven on a flat, barrier-free track and is difficult to make decisions in the face of complex driving conditions. Comma. ai trains cars using simulated road conditions to clone driver behavior and planning skills.It uses the variational automatic encoder (VAE) and the generated adversarial network (GAN) to realize the cost function of road video prediction, and on this basis combined with the recursive neural network (RNN), this method can predict the multi-frame realistic picture[4]. [5] defines how to from the Angle of vision, automatic driving through deep learning, not just in the road conditions can be achieved under the condition of simple lanes to follow, but also can deal with more complex environment, and proved that this model can effectively achieve the goal of automatic driving, implied to study the image at the same time the key information. And joined the short - and long-term memory network (LSTM) for modeling the time series, which can make use of the history of the driver information. A multi-task learning demonstration framework is proposed to achieve the goal of unmanned driving with guided assisted learning[6].

The end-to-end model has been developing towards more and more elaborate due to the rapid development of deep learning technology.From the initial three-tier network, it was gradually equipped with the latest modules and techniques.With the support of these latest technologies, the end-to-end driving model has basically realized the functions of straight and curve driving, speed control and so on.

2. Model Architecture Design

The input of the deep learning network designed in this paper is the image captured by three cameras of the unmanned vehicle, and the output is the steering Angle.Minimizing loss function is the ultimate goal of network optimization

2.1 FCN-BiLSTM Architecture

Convolutional neural network has 15 layers, including 11 convolutional layers, 3 pooling layers and 1 full connection layer. There are two layers of BiLSTM embedded between convolutional neural network and full connection network.After preprocessing, the size of the original image was adjusted to 66x200, and then it was taken as the input of the network. Then the image was input into the convolutional layer, and features were extracted through the convolutional layer. Finally, the output of the network was the size of the turning Angle.Since the features of adjacent areas are similar and almost unchanged, the image input into the pooling layer can select the pixel that can best represent the features, which not only reduces the amount of data, but also retains the features of the image.

The pooling layer is followed by two layers of bidirectional long - and short-term memory neural network. Finally, a value representing the turning Angle is output through the full connection layer.The frame of FCN-BiLSTM is shown in figure 1:

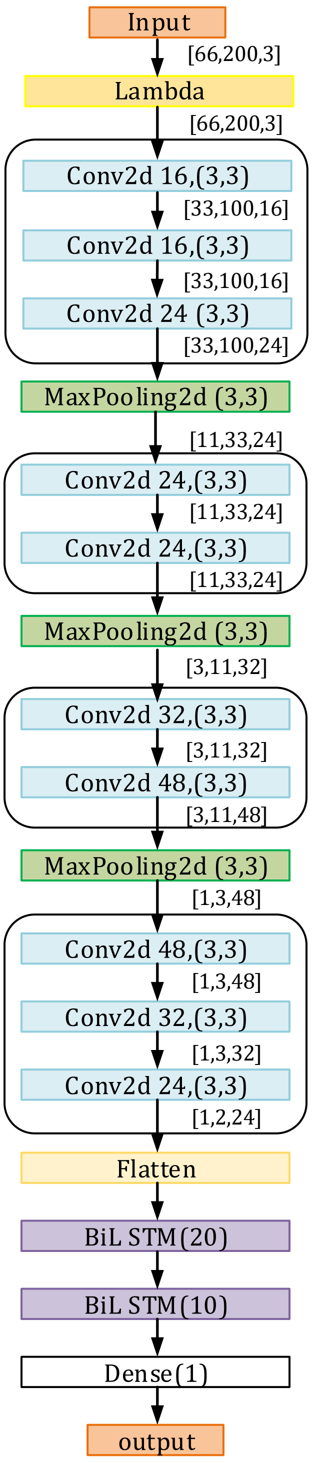

Figure 1. FCN-BiLSTM Architecture

2.1.1 Convolutional Neural Network Architecture

Convolutional neural network is a kind of feedforward neural network that includes convolution calculation and has deep structure. It is one of the representative algorithms of deep learning.A complete convolutional neural network is usually composed of convolution layer, pooling layer, excitation layer and full connection layer.The convolutional layer is the core of convolutional neural network.Each convolution layer is composed of many convolution units. Convolution kernel is convolved with the input image, which means to slide the convolution kernel on the image, that is, to dot the convolution kernel with its corresponding receptor field, to find out the corresponding mapping activation. Finally, parameters of each convolution unit can be calculated by back propagation algorithm.The purpose of convolution operation is to extract different features of input. The first convolutional layer may only extract some low-level features, such as edges, lines and corners, etc., while more multilayer networks can iteratively extract more complex features from low-level features.

The size of the output image is not only related to the size of the input image and the size of the convolution kernel, but also related to the depth, step size and zero fill of the input image.Depth is the number of convolution kernels, and the number of convolution kernels is usually set to 2^n.Step size is the step size of each dimension of the image when you convolve it, it's a one-dimensional vector.When we do convolution, in order to prevent the edge information from being missed, we usually fill the edge with zero.Zero fill defines the space between the element border and the element content. When the step length is 1 and the size of the convolution kernel is$\mathrm{F} \times \mathrm{F}$, the zero fill is$\frac{F-1}{2}$.

Remember that the size of the input image is $W_1 \times H_1 \times D_1$, the size of the convolution kernel is F×F, the number is K, and the size of the output image is $W_2 \times H_2 \times D_2,$, the calculation formula is as follows:

$$W_2 = \frac{W_1 - F + 2P}{S} + 1 \tag{1}$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1 \tag{2}$$

$$D_2 = K \tag{3}$$

The size of the input image in this algorithm is 66×200×3. In the first convolutional layer, the size of the convolution kernel is F=5, the number is K=16, and the step length is S=2. The zero filling takes the form of "SAME" and fills until the filter can reach the image edge.According to the formula, the size of the output image is 33×100×16.The output of other convolution layers can be calculated in the same way.

There are two methods for pooling: maximum pooling and average pooling.Features have been extracted from the previous convolutional layer. The features of adjacent regions are similar and almost unchanged. Pooling only selects the pixels that can best represent the features, reduces the amount of data, and retains the features. The pooling layer can be described as blurring the image, missing some less important features, namely downsampling.

Convolution neural network has three characteristics: local perception, weight sharing and multi-ple convolution kernel. Local perception is actually convolution kernel and image convolution, each convolution kernel only covers local features, so local perception.Convolutional neural network is a process from part to whole, while traditional neural network is a whole process.The number of parameters of a traditional neural network is very large, for example, 1000×1000 pixel picture, which requires 10^{12} parameters.In addition to the full connection layer, the parameters of the convolution layer depend entirely on the size of the convolution kernel.For example, a 10×10 convolution kernel requires only 100 parameters.Compared with traditional neural network, the whole picture can share a set of parameters of convolution kernel, and the number of parameters and computation is greatly reduced.

2.1.2 Activation Function

The main function of activation function is to add nonlinear factors to solve the problem of inadequate expression ability of linear model, which plays a crucial role in the whole neural network.The commonly used activation functions in neural networks include Sigmoid, Tanh, ReLU and ELU.

This algorithm uses ELU as the activation function.It attempts to accelerate learning by averaging the output of the activation function to near zero.At the same time, it can avoid the problem of disappearing gradient by positive identification.The expression of ELU function is as follows:

$$\mathrm{f(x)} = \begin{cases} x & , \ \text{if } \mathrm{x} > 0 \\ (e^x - 1), & \text{otherwise} \end{cases} \tag{4}$$

α is a constant and can be selected by cross-validation.

Compared with Sigmoid, Tanh and ReLU, ELU has the following advantages:

(1) High computational efficiency.Sigmoid and other activation functions are adopted, which requires a large amount of exponential calculation. When calculating the error gradient through back propagation, the derivation involves division, which requires a relatively large amount of calculation.

(2) There will be no gradient disappearance.When the sigmoid function is approaching the saturation zone, the transformation is too slow, and the derivative tends to 0, which will cause information loss, so that the training of deep network cannot be completed.When the ELU function is propagated back, the gradient will neither disappear nor explode.

(3) This method is used to speed up the convergence.

(4) ELU function is an improved version of ReLU function. Compared with ReLU function, ELU function also has corresponding output when the input is negative.Moreover, this part of the output has certain anti-interference ability, which can eliminate the death of ReLU neurons.

2.1.3 Bidirectional Long and Short Term Memory Neural Network

The long and short term memory neural network (LSTM) can be regarded as a representative recursive neural network, which can learn long-term dependence problems.The core of the long and short term memory neural network is the introduction of a new state C_t , which is used to store the things that want to be remembered, and the addition of three gates.

It‘s structure is shown in the following figure:

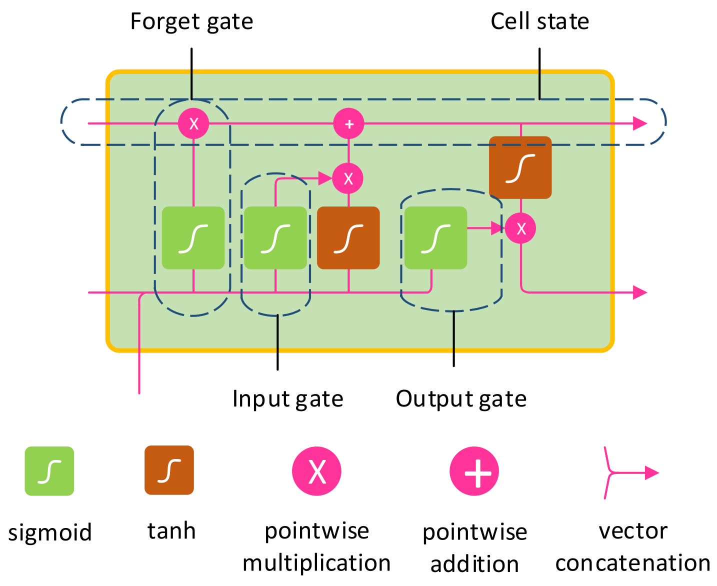

Figure 2. LSTM Architecture

(1) Forget gate: when do you need to forget your previous state.

Forget gate will read h_{t-1} and x_t, and output a value between 0 and 1 to the number in C_{t-1}, “1”means complete retention, “0”means complete abandonment.The formula is as follows:

$$f_t = \delta(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{5}$$

(2) Input gate: decide when to add a new state

The input gate first needs to find the state to be updated, and then update the state to state C_t. The specific formula is shown as follows:

$$i_t = \delta(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{6}$$

$$C_t^{\sim} = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{7}$$

The tanh layer creates a new state value vector—C_t ,that will be added to the state.

After the forgetting gate finds the information f_t that needs to be forgotten, it multiplies it with the old state, discarding the information that needs to be discarded.The result is then added to $i_t \times C_t$ to obtain the new information, which completes the status update of C_t.

$$C_t = f_t * C_{t-1} + i_t * C_t^{\sim} \tag{8}$$

(3)Output gate: decide when you need to put state and input together for output.

In the output gate, a Sigmoid layer is used to determine which part of the information will be output. and then the C_t state is processed through the Tanh layer to get a value between -1 and 1, and it is multiplied with the Sigmoid gate output, finally output the final desired output.

$$O_t = \delta(W_o[h_{t-1}, x_t] + b_0) \tag{9}$$

$$h_t = O_t * \tanh(C_t) \tag{10}$$

BiLSTM is composed of single layer long term and short term memory (LSTM) from front to back and from back to front.

2.2 Cost Function

Since the turning Angle is a continuous value, the mean square deviation is used to measure the loss function.Mean square error refers to the expected value of the deviation square of the network output steering Angle and the real steering Angle difference when driving.The mean square error can evaluate the degree of change of the data. The smaller the mean square error, the better the accuracy of the prediction model to describe the experimental data.N is the number of training samples, $\mathrm{f}(x_i)$ is the value of the steering Angle output by the network, and y_i is the value of the real steering Angle when driving a human vehicle.The loss function is expressed as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f(x_i) - \ y_i \ )^2 \tag{11}$$

2.3 Optimization Method

This algorithm adopts Adam optimization algorithm.Adam is different from the classical stochastic gradient descent method.Stochastic gradient descent maintains a single learning rate for all weight updates, and the learning rate does not change during training.Each network weight maintains a learning rate and is adjusted individually as learning progresses.This method calculates the adaptive learning rates of different parameters from the budget of the first and second moment of the gradient.

Adam algorithm is a kind of accelerated gradient descent algorithm based on the gradient descent method of driving quantity.Compared with the gradient descent method that drives quantities, both the accelerated gradient descent algorithm and Adam algorithm have improved ideas on how to make the learning on the horizontal axis faster and the learning on the vertical axis slower.The accelerated gradient descent algorithm and Adam algorithm introduce the square gradient and correct the deviation of the speed on the basis of the gradient descent method.The specific calculation formula is as follows:

$$V_{W^L} = \beta_1 V_{\omega^L} + (1\text{-}\beta_1)\frac{\partial J}{\partial \omega^L} \tag{12}$$

$$V_{\omega^L}^{corrected} = \frac{V_{\omega^l}}{1-(\beta_1)^2} \tag{13}$$

$$S_{\omega^L} = \beta_2 S_{\omega^L} + (1-\beta_2)(\frac{\partial J}{\partial \omega^L})^2 \tag{14}$$

$$S_{\omega^L}^{corrected} = \frac{S_{\omega^l}}{1-(\beta_2)^2} \tag{15}$$

$$\omega^L = \omega^l - \alpha\frac{V_{\omega^L}^{corrected}}{\sqrt{S_{\omega^L}^{corrected}} + \varepsilon} \tag{16}$$

3. Simulator

At present, a lot of training of driverless software takes place in the virtual environment, mainly because the cost of building a driverless car is very expensive, and it is difficult to find a real road to test the driverless car in real life. Secondly, we cannot guarantee the safety of driverless cars 100%, and the traffic rules in real life are very complex, so it is necessary to choose the open source simulator.

To investigate end-to-end behavioral cloning, we did it through Udacity's open source emulator. During the experiment, the simulator can collect road condition image data and control data. Before training the unmanned driving model, it is necessary to collect the training data, then take the road condition image data as the input of the model, and the output of the model is the control data for the road condition image.

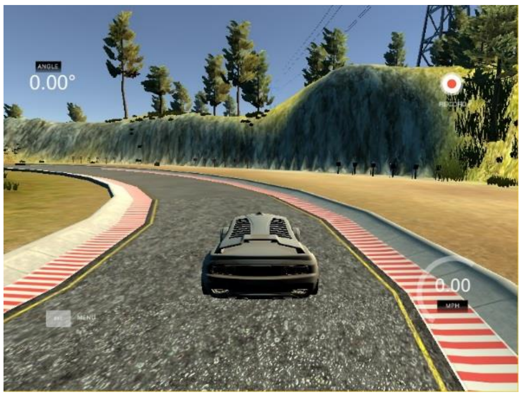

Figure 3. Udacity simulator interface

4. Experiment

4.1 Data Collection

Udacity's driving simulator offers two different test tracks, the lakeside trail and the jungle trail, from which all the sample data was collected. The car can drive around the track in training mode, which records the car's steering Angle and images from three separate cameras on the left, center and right.

Using three cameras, data can be obtained from the left and right sides of the car, and the model can be corrected by adding an offset to the left and right turning angles, allowing the car to drive in the center of the road. The number of data sets was 24108 and the sample size was 160x320.

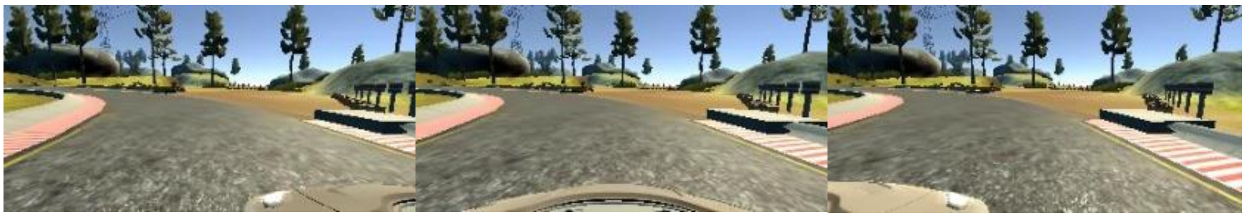

Figure 4. Images collected by three cameras

4.2 Data Preprocessing

In order to reduce overfitting and improve the driving ability of our model in unfamiliar environments, we can enhance our data through some proven image enhancement techniques. One method we used was to randomly adjust the brightness of the image, first converting the image to HSV color space, then magnifying or shrinking the V channel through random factors, and finally converting the image back to RGB color space. Another method is to flip the image around the vertical axis and reduce the error of the steering Angle. First, the same number of left and right steering angles can be obtained in the training data set, so as to reduce the possible error of the car when turning left and right. The original size of the image in the dataset we collected was 160x320. First, the first 40 pixels and the second 20 pixels of the image were cropped to remove the noise from the sky or tree tops and the car hood from the top of the image, and the image size was adjusted to 66x200.

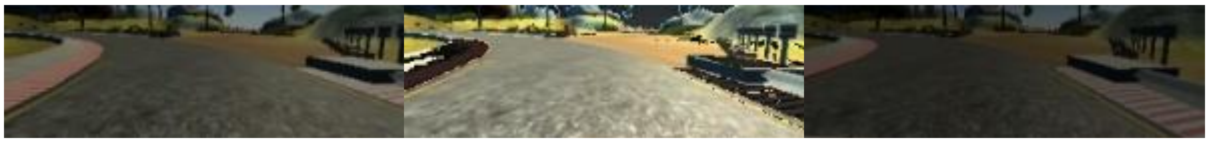
Figure 5. The preprocessed image

4.3 Result

The data provided by Udacity only contains the turning Angle of the center image, so in order to effectively use the left image and the right image during training, the left image is offset by2.75and the right image is offset by 2.75.When the steering Angle is 0, the car will go straight. When the turning Angle is negative, the car turns left; When the turn Angle is positive, the car turns right. Since the distance between the three cameras is unknown, the Angle is assumed to be 0.275 after repeated experiments. Due to the large proportion of small angles in the data provided, in order to prevent the car from driving straight, the number of angles in the range of [-0.1,0.1] should be limited to half of any given batch. At the same time, in order to make the right turn data in the training data distributed evenly, the image is flipped horizontally and the symbol of the turning Angle is changed. In addition, the photos taken by the three cameras were randomly selected to create the balanced data set.

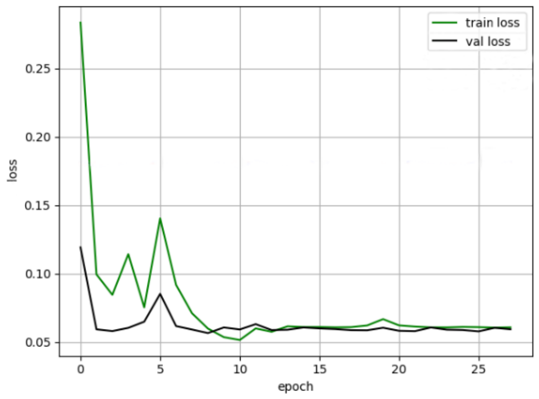

Figure 6. Cost Function

In this paper, FCN-BiLSTM network was used for training. The sample size of each round was 24000 and the batch size was 64.When the number of iterations was set as 30, the model overfitted, so the final number of iterations was set as 28, and after 28 rounds of iterations, the error of the verification set was no longer reduced, and was stable at about 0.06. The loss changes of the training set and the cross-validation set in the training process are shown in figure 6.

5. Conclusion

The model successfully navigated two different test tracks, lakeside trail and jungle trail. During the test of the lakeside trail, the car was uneven on the curve. Although the data set was taken from the lakeside trail, the results were still good when testing the jungle trail, with cars driving in the center of the road. Although the data set only comes from the track lakeside trail, the unmanned vehicle can not only drive steadily on the track of the lakeside trail, but also drive normally in the track center of the jungle trail.

References

[1] D. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, Advances in Neural Information Processing Systems 1. Morgan Kaufmann, 1989.

[2] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In Proceedings of the IEEE International Conference on Computer Vision, pages 2722–2730, 2015.

[3] 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.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202003_6(3).0030

[4] Santana E, Hotz G. Learning a driving simulator [J]. arXiv preprint arXiv:1608.01230, 2016.

[5] H. Xu, Y. Gao, F. Yu, and T. Darrell. End-to-end Learning of Driving Models from Large-scale Video Datasets. ArXiv e-prints, Dec. 2016.

[6] Mehta A, Subramanian A, Subramanian A. Learning end-to-end autonomous driving using guided auxiliary supervision [J]. arXiv preprint arXiv:1808.10393, 2018.

[7] Bansal M, Krizhevsky A, Ogale A. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst [J]. arXiv preprint arXiv:1812.03079, 2018.

[8] Mehta A, Subramanian A, Subramanian A. Learning end-to-end autonomous driving using guided auxiliary supervision [J]. arXiv preprint arXiv:1808.10393, 2018.