Journal of Computers Vol. 30 No. 4, 2019, pp. 78-89
doi:10.3966/199115992019083004007

Shallow Neural Network Based Vision Mobile Robot Control

Zhe Wu[1], Yiqin Yang[1], Qingyang Xu[1*], Fabao Yan[1]

[1] School of Mechanical, Electrical & Information Engineering, Shandong University, Weihai, Shandong 264209, China
{guiyuanwwu, Yangyiqinsdu}@163.com, {qingyangxu, hjc-8555}@sdu.edu.cn

Received 7 October 2017; Revised 28 February 2018; Accepted 31 March 2018

Abstract. Although deep neural network has many advantages, the main problem is the large requirement of computing power. The performance of shallow network is inferior to deep neural network, but the requirement of computing power is smaller. Therefore, we present an improved shallow neural network based on the deep learning techniques. The new techniques will improve the shallow neural network. This work takes a mobile robot as experimental platform using a camera module and a Raspberry Pi. The proposed shallow neural network is used to study driving a robot as a simulation research of unmanned vehicle, and then the robot can be steered by the trained neural network. In the training mode, the camera module provides the images needed for training the shallow neural network, and the robot will predict the movements and direction using the camera images according to the trained model in the autonomous mode. The experiment exhibits the effectiveness of the proposed model.

Keywords: batch normalization, mobile robot, Raspberry Pi, shallow neural network

1 Introduction

Machine learning includes kinds of subject including artificial neural network [2], fuzzy system [2-3] *etc*. Artificial neural network is a main branch of machine learning. The early model of artificial neural network is introduced by McCulloch and Pitts in 1943. This model could compute any arithmetic or logical function. In 1949, the famous Hebb learning rule is proposed according to the mechanism of biological neurons learning. The Hebb rule is often paraphrased as "Neurons that fire together wire together." The first practical application of artificial neural network came with the invention of the perceptron network. This network demonstrated its ability in pattern recognition. Unfortunately, the networks suffered from the inherent limitations. The key problem was that a single-layer neural network is incapable of solving the exclusive-or (XOR) problem, and also the computation power of the computer was not enough to effectively handle the large scale neural networks at that time. Influenced by this, many researchers believed that further research on neural networks was unpromising. In the following decade, some new concepts were introduced and responsible for the rebirth of neural networks. The most important one is the back propagation algorithm. This algorithm is a generalized form of the delta rule for training multilayer perceptron networks which provided a solution to handle the XOR problem. A typical perception includes input layer, hidden layer and an output layer. The input feature vector is functioned by the activation function (such as sigmoid, tanh) of hidden layer, and the output layer will get the final result. One long-term goal of artificial neural network research is to handle highly complex tasks, such as perception (vision, audition), reasoning, intelligent control, and other artificially intelligent behaviors. The artificial neural network with more layers can learn highly complex functions with minimal need for prior knowledge, and with minimal human intervention. The implication of the multi-layer perceptron is that the number of layers of the neural network directly determines its ability to characterize the reality. However, for deep architectures, the optimization of deep neural network is more and more hard to train, and it is easily to fall into the local optimal solution [4]. For a limited training data, the performance of deep neural network is also worse than shallow network and is easy to overfit. The learning speed is

* Corresponding Author

slower when the weights are far away from the output layer. It is an unignored problem that the gradient disappears is more serious as the number of layers increases [5-6]. In 2006, Hinton makes use of the pre-training method to alleviate the local optimal solution problem. The hidden layer is increased to seven layers, the neural network has a real sense of depth. Although the deep neural network has kinds of applications [7-8], there are many drawbacks for the deep structure. In order to overcome the gradient disappears, ReLU [6], dropout [9] and other transfer function instead of sigmoid are used in deep architectures.

The deep learning and unmanned vehicle have become a hot topic [10-11]. More and more companies are dedicated to developing driverless technologies such as Tesla and Nvidia. Deep neural network has been widely used in many aspects of vehicle control. NVIDIA has developed a self-driving car called DAVE-2 [12, 14], which drove on real roads.DAVE-2's network consists of 9 layers, 27 million connections and 250k parameters. However, it is unsuitable for embedding system due to the large scale computation power requirement. So now there is an urgent need for a neural network model that does not require too much computational power to do some researches. Indeed, several teams are already using improved neural network technology such as MIT RACECAR [13] and Jetson RACECAR.However they still use deep neural network which also have the large requirement of computing power.

Although the shallow architectures have some shortcomings [15], if we adopt some effect improvement measures, an effect algorithm can be designed for the embedding system. Therefore, based on some skills of deep architectures, an improved shallow neural network is proposed. This model is running on a Raspberry Pi with limited computation power. Details about this shallow network are shown in the next section.

2 Shallow Neural Network

2.1 Different Neural Network Models

In order to indicate the proposed model, four different shallow neural networks are described to exhibit the evolution of the final neural network, including a traditional BP neural network [16] and some improved shallow neural networks as Fig. 1. Model 1 is a traditional BP neural network. In model 2, a softmax classifier takes place a traditional BP classifier. In model 3, the Relu function is adopted as the activation function of hidden layer to improve the performance of the model. In model 4, a batch normalization layer is added to adjust the distribution of the input data based on the model 3. Although, it is a shallow neural network, it is easy to extend to a deep neural network or a sub-deep neural network. Therefore, if the computation power is increase, a deep neural network or a sub-deep neural network can be constructed immediately according the model 4 as Fig. 1(e).

2.2 Algorithm of Shallow Neural Network

For the traditional BP neural network (model 1), the calculation process is as follows [17].

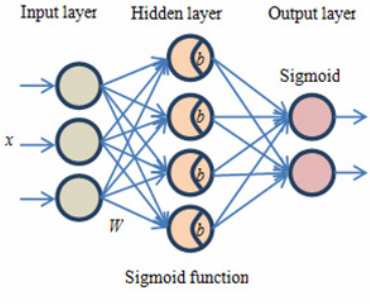
$$z^{(2)} = W^{(1)}x + b^{(1)}. \tag{1}$$

$$a^{(2)} = f\left(z^{(2)}\right). \tag{2}$$

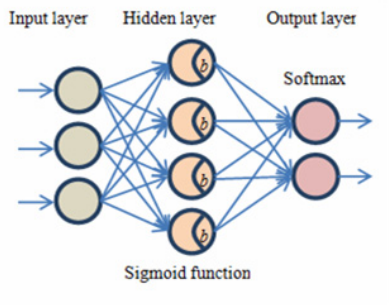
$$z^{(3)} = W^{(2)}\mathrm{a}^{(2)} + b^{(2)}. \tag{3}$$

$$o = f(z^{(3)}). \tag{4}$$

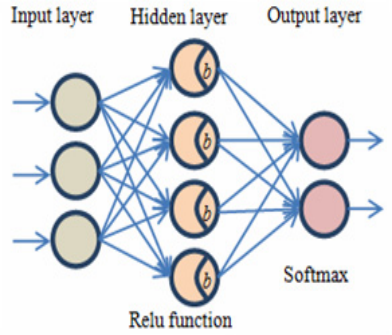
(x, y) is the pair of training data set, x is the input data and y is the expected data. W_{ij}^{l} is the weight between the jth neuron of layer l and ith neuron of l+1 layer, and b_i is the ith bias value of l layer. z_i is the sum of ith neuron input at layer l. $a_i^{(l)}$ is the activation value of ith neuron at layer l. f() is the activation function. o is the output of the neural network



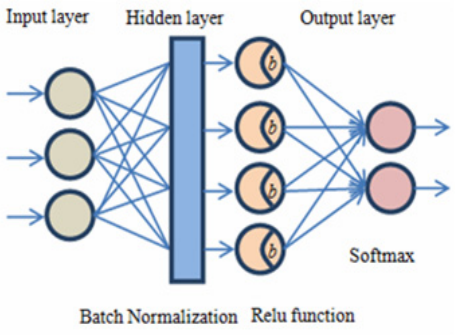
(a) Model 1- traditional BP



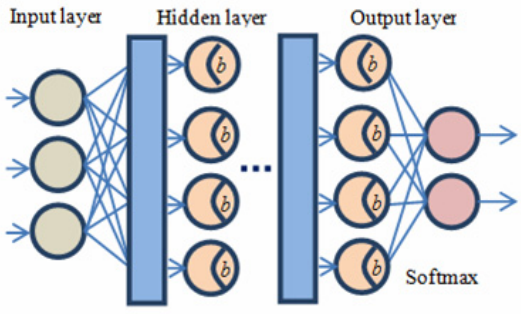
(b) Model 2



(c) Model 3



(d) Model 4



(e) Model 5 deep structure

Fig. 1. Four neural network models

The training of BP is based on the back propagation algorithm. The cost function of m samples, including weight attenuation term [18-19], can be described by equation (5).

$$\begin{aligned} J(W,\mathrm{b}) &= [\frac{1}{m}\sum_{i=1}^{m} J(W,b;x^{(i)},y^{(i)})] + \frac{\lambda}{2}\sum_{l=1}^{nl-1}\sum_{i=1}^{Sl}\sum_{j=1}^{Sl+1}(W_{ji}^{(l)})^2 \\ &= [\frac{1}{\mathrm{m}}\sum_{i=1}^{m}(\frac{1}{2}\| o(x^{(i)}) - y^{(i)} \|^2)] + \frac{\lambda}{2}\sum_{l=1}^{nl-1}\sum_{i=1}^{Sl}\sum_{j=1}^{Sl+1}(W_{ji}^{(l)})^2 \end{aligned} . \quad \textbf{(5)}$$

λ is the regularization parameter. m is the output neuron number.

According to the back propagation algorithm, the partial derivative can be computed as following formula (6, 7).

$$\nabla_{W^{(1)}} J(W,b) = \delta^{(l+1)}(a^{(l)})^T . \quad \textbf{(6)}$$

$$\nabla_{b^{(l)}} J(W,b) = \delta^{(l+1)}. \tag{7}$$

$\nabla_{W^{(1)}} J(W,b)$ is partial derivative value for a single sample. δ is the residual item of corresponding neuron.

The output layer residual item δ_i can be calculated as following,

$$\delta^{(l)} = -(y-o)\cdot f'(z^{(l)}). \tag{8}$$

For the hidden layer, the error item δ_i of lth layer can be calculated as following,

$$\delta^{(l)} = ((W^{(l)T}\delta^{(l+1)}))\cdot f'(z^{(l)}). \tag{9}$$

Then the weight and bias value can be updated by the following equation (10, 11),

$$W^{(l)} = W^{(l)} - \alpha[(\frac{1}{m}\Delta W^{(l)}) + \lambda W^{(l)}]. \tag{10}$$

$$b^{(l)} = b^{(l)} - \alpha[\frac{1}{m}\Delta b^{(l)}]. \tag{11}$$

α is the learning rate.

For a practical problem, output is always a multi-classification problem. Therefore, a softmax regression model is adopted as the output layer in model 2, due to softmax regression model is the generalization of logistic regression model on multi-classification problem. The cost function will be improved as equation (12).

$$J(\theta) = -\frac{1}{\mathrm{m}}[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\{y^{(i)} = j\}\cdot \log\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}]. \tag{12}$$

$1\{\cdot\}$ is indicative function. $\frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}$ is the softmax function.

Partial derivative is as following formula (13),

$$\nabla_{\theta_j} J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[x^{(i)}(1\{y^{(i)} = j\} - p(y^{(i)} = j \mid x^{(i)};\theta))]. \tag{13}$$

$$p(y^{(i)} = j \mid x^{(i)};\theta) = \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}. \tag{14}$$

The sigmoid function is the preferred activation function for hidden layer in traditional neural network. However, the sigmoid function has some drawbacks especially for the partial derivative computation. Fig. 2 shows curves of the sigmoid function, a Relu function and the derivative of sigmoid function.

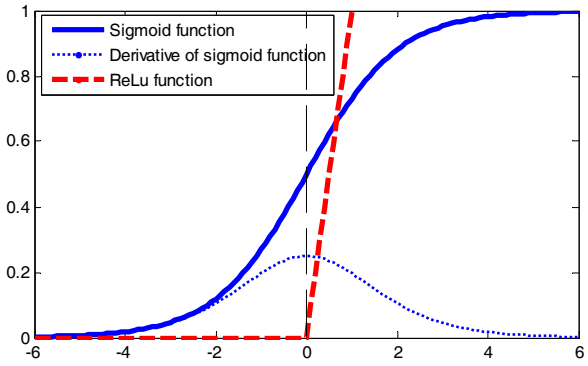


Fig. 2. Activation function comparison diagram

Form Fig. 2 we can see that the derivative value of sigmoid is compressed to 0~0.25. In the error propagation process, the residual comes from the multiplication of derivative value of sigmoid and sum of upper layer errors. By this calculation, the errors will be compressed to a smaller value than 1/4 of aggregate error. This will become more and more serious with the increasing layer in the backward propagation. It is the famous " gradient diffusion" problem. Therefore, the layers closing to the input layer will not get enough training signal.

In deep structures, the ReLu function is always adopted as the activation function as model 3.

$$f(z) = \max(0, z) . \tag{15}$$

The advantage of Relu is that it is simple for calculation as Fig. 2, without saturated, fast convergence and easier to find the extreme point of function. Therefore, the Relu function is always adopted in the deep structures. The output layer residual item δ_i improves as following due to $f'(z)$=1.

$$\delta^{(l)} = -(y - a^{(l)}) . \tag{16}$$

For the hidden layer, the error item δ_i of lth layer can be improved as following,

$$\delta^{(l)} = ((W^{(l)T}\delta^{(l+1)})) . \tag{17}$$

In deep neural network, the output of the current layer will be the input of the next layer. Therefore, the input of the deep layer will change in larger due to the changing of the previous layer output caused by the parameter adjustment. It is hard to train the nonlinear model. Therefore, a batch normalization [20] layer is added in order to adjust the distribution of input data and also playing a regularization effect at the same time which reduce the requirement of the dropout strategy. In model 4, a batch normalization layer is added following the input layer. The batch normalization includes a Gaussian normalization and a linear transformation, and they are all smooth and derivable. It is a very important property for parameter learning in back propagation process. The operation of batch normalization can be described as following.

$$\mu_\beta \leftarrow \frac{1}{\mathrm{m}}\sum_{i=1}^{m} x_i \quad \text{//mean} . \tag{18}$$

$$\sigma_\beta^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\beta)^2 \quad \text{//variance} . \tag{19}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \varepsilon}} \quad \text{//normalize} . \tag{20}$$

$$y_i \leftarrow \gamma\hat{x}_i + \beta \quad \text{//scale and shift} . \tag{21}$$

x is the mini-bath input data of batch normalization layer. u and σ are the mean and variance value of the mini-batch input. γ and β are the linear transformation parameters. y is the adjusted data.

$$\frac{\partial l}{\partial \hat{x}_i} = \frac{\partial l}{\partial y_i}\frac{\partial y_i}{\partial \hat{x}_i} = \frac{\partial l}{\partial y_i}\cdot\gamma . \tag{22}$$

$$\frac{\partial l}{\partial \sigma_\beta^2} = \sum\nolimits_{i=1}^{m}\frac{\partial l}{\partial \hat{x}_i}\cdot(x_i - \mu_\beta)\cdot\frac{-1}{2}(\sigma_\beta^2 + \varepsilon)^{-3/2} . \tag{23}$$

$$\frac{\partial l}{\partial \mu_\beta} = (\sum\nolimits_{i=1}^{m}\frac{\partial l}{\partial \hat{x}_i}\cdot\frac{-1}{\sqrt{\sigma_\beta^2 + \varepsilon}}) + \frac{\partial l}{\partial \sigma_\beta^2}\cdot\frac{\sum_{i=1}^{m}-2(x_i - \mu_\beta)}{m} . \tag{24}$$

$$\frac{\partial l}{\partial x_i}=\frac{\partial l}{\partial \hat{x}_i}\cdot\frac{1}{\sqrt{\sigma_\beta^2+\varepsilon}}+\frac{\partial l}{\partial \sigma_\beta^2}\cdot\frac{2(x_i-\mu_\beta)}{m}+\frac{\partial l}{\partial \mu_\beta}\cdot\frac{1}{m}. \tag{25}$$

$$\frac{\partial l}{\partial \gamma}=\sum_{i=1}^{m}\frac{\partial l}{\partial y_i}\cdot\hat{x}_i\ . \tag{26}$$

$$\frac{\partial l}{\partial \beta}=\sum_{i=1}^{m}\frac{\partial l}{\partial y_i}. \tag{27}$$

In the weight updating, the parameters are changed along with the negative gradient direction to minimize the loss function.

$$x=x\text{-}\alpha\cdot\frac{\partial l}{\partial x}. \tag{28}$$

α is the learning rate. As long as the learning rate is low enough, it is always possible to get a positive progress on the parameter regulation for the entire data set. Momentum item is always used in deep neural network due to its better convergence speed [21]. The gradient update is realized based on the momentum method in model 4 as following.

$$v\ =\ u\cdot v\ \text{-}\ \alpha\cdot\frac{\partial l}{\partial x}. \tag{29}$$

$$x=x+v. \tag{30}$$

v is the regulation parameter, and u is a hyper parameter which always called momentum coefficient. The parameter u effectively suppresses the speed and reduces the kinetic energy of the system. It is often helpful to have an anneal strategy for the learning rate over time when train the deep neural network. If the learning rate is high, the kinetic energy of the system will be larger and the parameters will irregularly fluctuate. In model 4, an adaptive learning rate is adopted according to the iteration numbers.

$$\alpha'=k\alpha\ . \tag{31}$$

3 Experiment

3.1 Hardware Configuration

In this project, the main components include a small chassis of car, a Raspberry Pi 3B [22], a 5V power bank, a L293D motor driver IC and a camera fixed on the top of the car. The motor driver L293D controls the two motors which are for the forward control and steering angle control. The front motor controls the left/right direction and the back motor is for the forward/reverse direction control. The Raspberry Pi camera module installed on the top of the car captures the images needed for neural network training and autonomous navigation.

If the image of camera contains a large region of the car, we should adjust the position of the camera to get enough road information. Then the images coming from the camera are used as the inputs of the neural network for training and prediction.

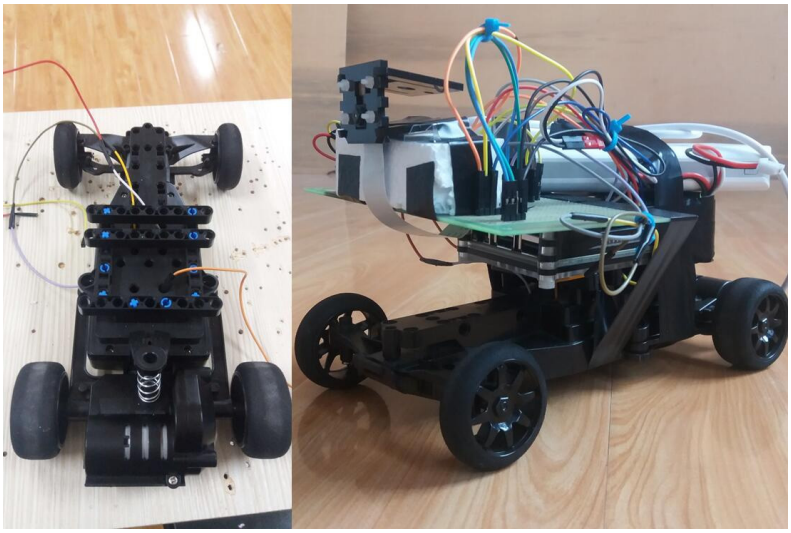


Fig. 3. The structure of the autonomous car

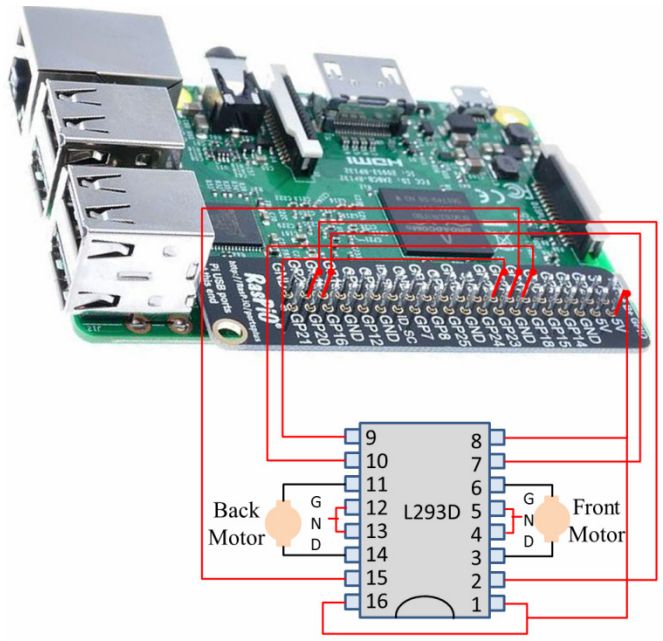


Fig. 4. The hardware block diagram

3.2 Parameter Configuration

The experiment includes two stages. The first one is data collection and model training. The second is the prediction and autonomous navigation. In data collection section, the images required can be captured by the Raspberry Pi camera. The images are captured along with the corresponding function (forward, left, right) key to assist us in their classification later. The car is steered on various circumstance and captures around 1600 images for the training data set.

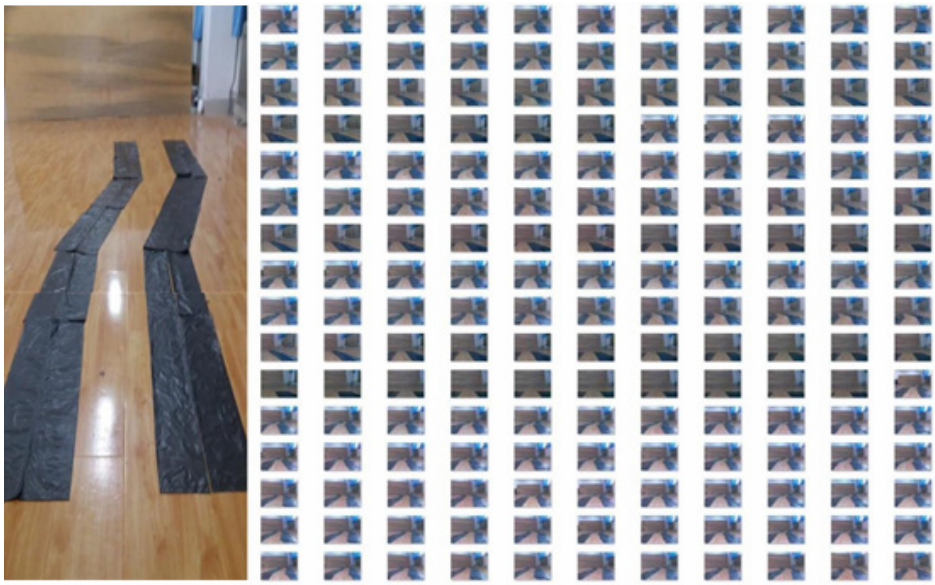


Fig. 5. The experimental site and samples

The RGB image size is 75*75. There are some preprocessing techniques such as the image cropping, the conversion of RGB image to gray picture, etc. The images are transferred to the neural network when the preprocessing is finish. The input layer has 5625 neurons according to the image size, the neuron number of hidden layer is 60 and the output layer size is 3. The regularization parameter λ is 0.1, the batch size is 200 which decide the computation of loss value in the forward and backward calculation, the momentum coefficient is 0.9 and the learning rate attenuation coefficient is 0.95.

3.3 Training Process

The four models are trained by the samples, and we get the corresponding convergence curve of the loss value.

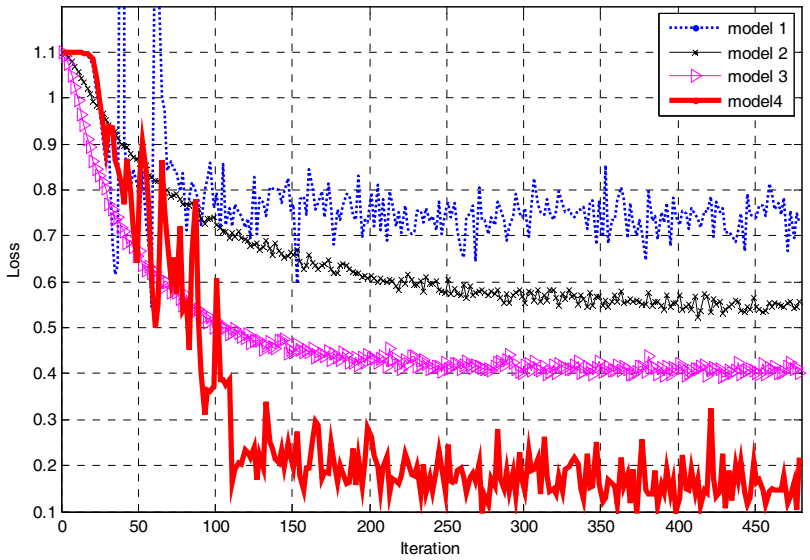


Fig. 6. The convergence curve of loss

From Fig. 6, we can see that the convergence speed of model 1 is slower than other models and the final loss value is bigger which may be caused by the drawbacks of sigmoid function. The convergence speeds of model 2 and model 3 have certain promotion and have a lower final convergent value. Model 4 has the best learning ability including the convergent speed and final loss value. In model 4, a Relu

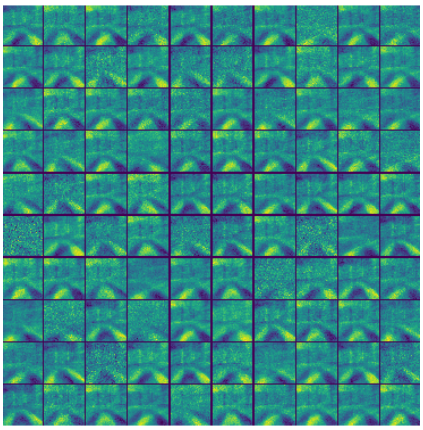
activation function and the batch normalization technique are adopted to overcome the gradient diffusion problem and adjust the distribution of input data.

Table 1 shows the accuracy statistics of the four models. Model 4 has the best capability. The table shows the progressive improvement process of different models. The traditional BP neural network (Model 1) has the worst performance. With the improvement of the models, the performance of the neural network becomes better and better.

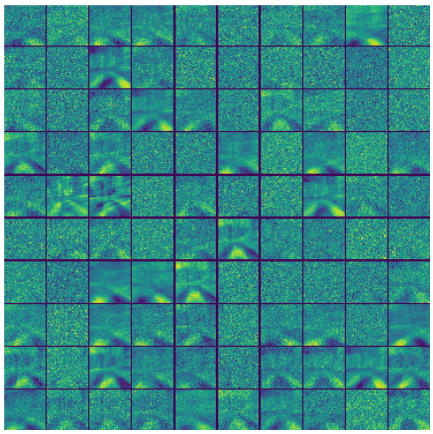
Table 1. The accuracy statistics of four models

Model	Accuracy
Model 1	35.5%
Model 2	41.5%
Model 3	70.5%
Model 4	95.75%

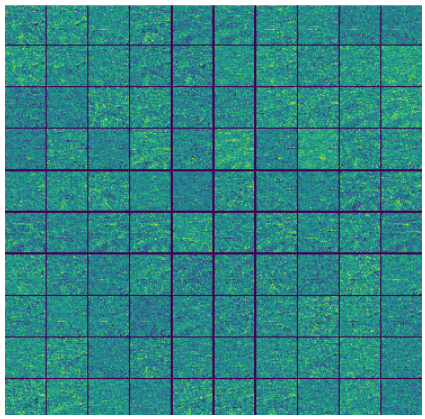
In order to exhibit the information learning of the models, we have a visualization of the weight between the input and hidden layer as Fig. 7. We can see that it is the features of tracks coming from different models. Fig. 7(a) is the visualization of model 1 and model 2. It is the feature of tracks. However, there is not obvious characteristics for different neurons of hidden layer. In model 3, a Relu function is used in hidden layer which is a truncation function. Therefore, the main features are extracted. The function of feature extraction looks like the characteristics of the deep neural network. Although, there are some difference for the features, it is not obvious. For model 4, there are not obvious features by eye due to adding a batch normalization after input layer. The input images are changed. Therefore, the features is not the original one. But the data distribution are adjusted with better characteristics which is benefit to network training.



(a) Model 1, 2 visualization



(b) Model 3 visualization



(c) Model 4 visualization

Fig. 7. The visualization of the weight

3.4 Experiments

Once we have the trained model, we can run the car autonomously. The forward surrounding images are captured by the camera module and input to the trained model which will predict the next navigation action. The front and the back motors are controlled based on the predicted action.

In order to exhibit the steering effect of the autonomously car, a red pen is fixed at the end of the car, and the track are covered by the A4 paper which also can testify the robustness of the model by color changing of the track. There will be a red trajectory recording on the paper when the car is running on the track. The direction of the track is also changed to test the generalization ability of the model.

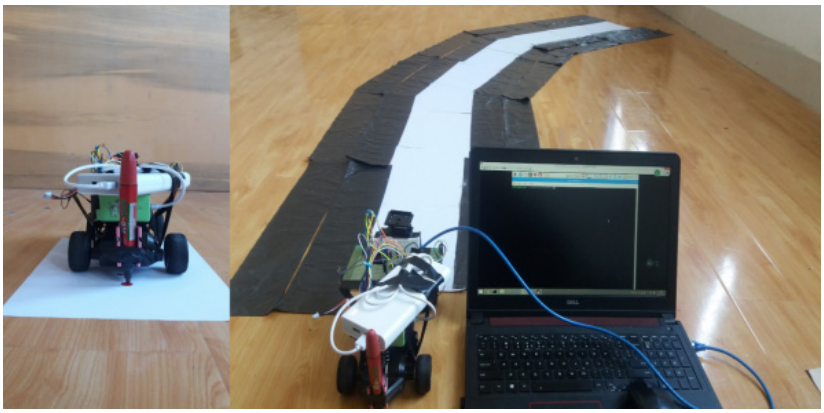


Fig. 8. The experiment scheme

(a) (b) (c)

Fig. 9. The navigation result

Fig. 9 shows the steering effect when the car navigates on the track. The first one is the navigation effect of model 1, model 2 and model 3. We can see that the car drives out the track. The second and third one is the navigation effect of model 4. Fig. 9(b) is the navigation process and Fig. 9(c) is the final trajectory exhibition. Although, the direction of the track is changed and the floor is covered by A4 paper, the small car drives along the center of the track. It means that model 4 is effective to learn the driving characteristics and have better robustness.

4 Conclusions

Considering the computation power of the Raspberry Pi, a shallow neural network is proposed based on the techniques of deep learning. Four models are described to exhibit the improvement progress of the shallow neural network. A Relu function is adopted as the activation function of hidden layer to overcome the gradient diffusion. Combing the batch normalization technique and adopting the softmax function as the classifier, the shallow neural network is constructed based on the deep structure. The shallow one can be extended to a deep one easily. According to the training result, the accuracy is about 98.5%. The car can navigate along the track correctly despite the track is changed. Comparing with conventional BP neural network, the accuracy and performance are improved greatly.

The following work will be the lightweight deep neural network construction and binarization of the neural network which will be better used for embedding system. We also plan to improve prediction accuracy and speed control by providing more complex data closer to the actual road conditions.

Acknowledgements

This work is supported by National Nature Science Foundation of China (61603214, 61573213), and the Natural Science Foundation of Shandong Province (ZR2015PF009).

References

[1] M.J. Er, Y. Zhang, N. Wang, M. Pratama, Attention pooling-based convolutional neural network for sentence modelling, Information Sciences 373(2016) 388-403.

[2] N. Wang, M.J. Er, Direct adaptive fuzzy tracking control of marine vehicles with fully unknown parametric dynamics and uncertainties, IEEE Transactions on Control Systems Technology 24(5)(2016) 1845-1852.

[3] X. Xiang, C. Yu, Q. Zhang, Robust fuzzy 3D path following for autonomous underwater vehicle subject to uncertainties, Computers & Operations Research 84(2017) 165-177.

[4] J.U.R. Schmidhuber, Deep learning in neural networks: an overview, Neural networks, 61(2015) 85-117.

[5] Y. Wu, S. Zhang, Y. Zhang, Y. Bengio, R.R. Salakhutdinov, On multiplicative integration with recurrent neural networks, in: Proc. Advances in Neural Information Processing Systems, 2016.

[6] B. Neyshabur, Y. Wu, R.R. Salakhutdinov, N. Srebro, Path-normalized optimization of recurrent neural networks with ReLU activations, in: Proc. Advances in Neural Information Processing Systems, 2016.

[7] L. Yu, Z. Wang, S. Tian, F. Ye, J. Ding, J. Kong, Convolutional neural networks for water body extraction from landsat imagery, International Journal of Computational Intelligence and Applications 16(01)(2017) 1750001.

[8] Z. Wu, X. Ding, G. Zhang, A novel method for classification of ECG arrhythmias using deep belief networks, International Journal of Computational Intelligence and Applications 15(04)(2016) 1650021.

[9] N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach Learn. Res. 15(1)(2014) 1928-1958.

[10] J. Cui, R. Bachmayer, W. Huang, B. Deyoung, Wave height measurement using a short-range FMCW radar for unmanned surface craft, in: Proc. OCEANS'15 MTS/IEEE Washington, 2015.

[11] N. Wang, S. Lv, M.J. Er, W. Chen, Fast and accurate trajectory tracking control of an autonomous surface vehicle with unmodeled dynamics and disturbances, IEEE Transactions on Intelligent Vehicles 1(3)(2016) 230-243.

[12] 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, K. Zieba, End to end learning for self-driving cars, arXiv: 1604.07316, 2016.

[13] R. Shin, S. Karaman, A. Ander, M.T. Boulet, J. Connor, K.L. Gregson, W. Guerra, O.R. Guldner, M. Mubarik, B. Plancher, R. Shin, J. Vivilecchia, Project based, collaborative, algorithmic robotics for high school students: programming self driving race cars at mit, in: 2017 IEEE Integrated STEM Education Conference (ISEC) (ISEC'17), 2017.

[14] K. Pei, Y. Cao, J. Yang, S. Jana, DeepXplore: automated whitebox testing of deep learning systems, arXiv: 1705.06640, 2017.

[15] H. Mhaskar, T. Poggio, Deep vs. shallow networks: an approximation theory perspective, arXiv: 1608.03287, 2016.

[16] S. Zhang, J. Lv, X. Yuan, S. Yin, BP neural network with genetic algorithm optimization for prediction of geo-stress State from wellbore pressures, International Journal of Computational Intelligence and Applications 15(03)(2016) 1650013.

[17] F. Yu, X. Xu, A short-term load forecasting model of natural gas based on optimized genetic algorithm and improved BP neural network, Appl. Energ. 134(134)(2014) 102-113.

[18] T. Salimans, D.P. Kingma, Weight normalization: a simple reparameterization to accelerate training of deep neural networks, in: Proc. Advances in Neural Information Processing Systems, 2016.

[19] D. Wu, F. Ren, W. Zhang, An energy optimal thrust allocation method for the marine dynamic positioning system based on adaptive hybrid artificial bee colony algorithm, Ocean Eng 118(2016) 216-226.

[20] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, CoRR. abs/1502.03167 (2015).

[21] H. Zhang, D. Xu, Boundedness and convergence of split-complex back-propagation algorithm with momentum and penalty, Neural Process. Lett. 39(3)(2014) 297-307.

[22] A.J. Lewis, M. Campbell, P. Stavroulakis, Performance evaluation of a cheap, open source, digital environmental monitor based on the raspberry Pi, Measurement 87(2016) 228-235.