Research and Implementation of CNN Based on TensorFlow

Liang Yu¹, Binbin Li¹,* and Bin Jiao¹
¹School of Electric engineering, Shanghai Dianji University, Shanghai, China

*Corresponding author e-mail: libinbin2000@126.com

Abstract. TensorFlow is Google's open source machine learning and deep learning framework, which is convenient and flexible to build the current mainstream deep learning model. Convolutional neural network is a classical model of deep learning, the advantage lies in its powerful feature extraction capabilities of convolutional blocks. Based on the TensorFlow platform, a convolutional neural network model with two-convolution-layers was built. The model was trained and tested with the MNIST data set. The test accuracy rate could reach 99.15%, and compared with the rate of 98.69% with only one-convolution-layer model, which shows that the two-convolution-layers convolutional neural network model has a better ability of feature extraction and classification decision-making.

1. Introduction
In the wave of artificial intelligence, the technological revolution brought by deep learning has spread widely. As a branch of machine learning, deep learning can not only learn the relationship between features and tasks, but also automatically extract more complex features from simple samples. Machine learning points the way for artificial intelligence, while deep learning makes machine learning truly implementation [1]. In the recent few years, deep learning has been extended to various fields of machine learning, bringing great convenience to people's lives.

At present, the mainstream open source deep learning framework mainly includes Google's TensorFlow, Microsoft's CNTK, Baidu's PaddlePaddle, California University's Caffe, Montreal University's Theano, Facebook's Torch and so on. TensorFlow officially opened source in November 2015. It is the second generation deep learning system. Compared with the first generation of DistBelief, it has faster calculation speed, more supported computer platforms and deep learning algorithms [2]. And the stability of the system is also higher, so it is greatly concerned and used by users. TensorFlow is a relatively high-level machine learning library, and the supported languages are C++ and Python. Users do not have to write complex code to build a neural network structure. The optimization algorithms and functions to be used in modeling can simply call the corresponding functions in the TensorFlow library, which greatly reduces the threshold for deep learning. With the rapid development of computer technology and the tremendous improvement of computing power, ordinary computers can also build deep learning models on the TensorFlow platform, which reduces the cost of deep learning and makes it easier to verify algorithms [3].

2. Deep learning model
2.1. CNN principle
Convolutional neural network (CNN) is a deep neural network. The main idea of CNN can be summarized as two points: sparse connectivity and shared weights. A typical convolutional neural
network consists of an input layer, a convolutional layer, a pooling layer, a fully connected layer, and an output layer. It is mainly divided into two processes: forward and back propagation. The former finally outputs the prediction result through the network structure, and the latter performs parameter adjustment according to the difference between the prediction result and the actual value. The input layer is image data that needs to be input, and it is generally a matrix of $N \times r \times r$. The convolutional layer is a fixed-size filter (convolution kernel) convoluted with the image of the previous layer to extract eigenvalues in the image. The principle is as shown in equation (1):

$$x^l_j = f \left( \sum_{i\in S_j} x^{l-1}_i \ast \omega^l_{ij} + b^l_j \right)$$  \hspace{1cm} (1)$$

$S_j$ represents a set of input images, $x^{l-1}_i$ represents a feature map of the $i$-th input of the $(l-1)$-th layer, $\omega^l_{ij}$ represents the convolution kernel from an $i$-th input feature map of the $(l-1)$-th layer to the $j$-th output feature map of the $l$-th layer, $b^l_j$ represents the bias value corresponding to the $j$-th output feature map of the $l$-th layer, $f(\cdot)$ represents the activation function, $\ast$ represents the convolution operation, the result is the $j$-th feature map $x^l_j$ of the $l$-th layer.

The pooling layer also called the down-pooling layer, it is a special convolutional layer. It generally has two forms: max pooling and average pooling. The principle is similar to equation (1), except that the activation function becomes a pooling function. The purpose of the transformation is to reduce the dimension of the feature map, without changing the number of feature maps, but to change its size, thereby reducing the parameters in the fully connected layer and speeding up the calculation.

The fully connected layer is connected to the convolutional layer and the ordinary layer. From the beginning of the input to the fully connected layer called convolutional layers with the "convolution block"; from the fully connected layer to the output which are all ordinary layers of the final "NN block". The fully connected layer will extract the high-dimensional data obtained from the parent layer (convolution layer) as input in a flat form similar to that in Figure 2, perform some nonlinear transformation (activation function), and then input the result into the subsequent system of ordinary layers. Further decisions (prediction and classification) are made in these layers. Therefore, "convolution block" becomes a feature extractor, and the "NN block" is called a decision-making classifier.

The loss function is a reflection of the degree of model fitting. The cross entropy loss function can solve the vanishing problem of gradient perfectly. The principle is shown in equations (2) [4].

$$L(y, G(x)) = -[y \ln(G(x)) + (1 - y) \ln(1 - G(x))], \quad G(x) = (v_1, \ldots, v_k)^T, \quad \sum_k v_k = 1$$  \hspace{1cm} (2)$$

The back propagation optimization method of convolutional neural network adopts the adaptive learning rate Adam algorithm, which is an extension of the stochastic gradient descent algorithm, and is the most widely used and generally best algorithm in deep learning applications. The error is backpropagated by the Adam algorithm, and the parameter values of the layers of the convolutional neural network are updated layer by layer. The principle of the simplified version is shown in equations (4) [5].

$$\Delta = \beta_1 \Delta + (1 - \beta_1) \Delta w_t, \nabla^2 = \beta_2 \nabla^2 + (1 - \beta_2) \nabla^2 w_t, \quad \Delta^t w_t = \frac{\Delta}{\sqrt{\nabla^t + \eta}}$$  \hspace{1cm} (3)$$

2.2. CNN classification model

Figure 1 is a schematic structural diagram of a convolutional neural network classification model. It can be seen from the figure that it mainly includes an input layer, a two-convolutional -layers, a two-pooling-layer, a fully connected layer, a Dropout layer, a Softmax and an output layer. The convolution layer convolves the data of the upper layer through the convolution kernel, and then uses the activation function to obtain the feature map of the convolution layer; the pooling layer is to find the maximum value through each $2 \times 2$ neighbourhood in the feature map of the upper layer, so that the dimension of the feature map after pooling is half of the upper layer; the fully connected layer is to tile the 64 feature maps of the second pooling layer into a vector; The Dropout layer is to prevent overfitting and improve the generalization ability of this model. Finally, the data is classified by the softmax regression model to output the category.
3. Research and Implementation on TensorFlow——Taking MNIST data set as an Example

3.1. TensorFlow and TensorBoard

There are three models in TensorFlow, which respectively are the calculation model: calculation graph, data model: tensor, running model: session. The data in TensorFlow is represented by a Tensor data structure, Flow represents the flow and calculation of data. It can use the feed (or fetch) to assign (or get) the data in the tensor.

TensorFlow is a deep learning programming system that expresses computations in the form of a computational graph. The TensorFlow program development flow chart is shown in Figure 2 [6], which is mainly divided into two stages: construction and execution graph.

TensorBoard is a visual tool corresponding to the TensorFlow calculation graph. It can visualize the output log files during the running process of the TensorFlow program, and effectively display the calculation graph and the trend of various parameter indicators with time during operation in TensorFlow, and facilitate the understanding and debugging of the program.

3.2. MNIST data set

There are two types of images in the MNIST handwritten digit recognition data set: one is 60,000 training sample sets, including 55,000 training samples and labels, 5000 training verification samples and labels, and the other is 10,000 test samples and labels. Each sample represents a picture, each picture is represented by a $28 \times 28$ matrix, which is tiled into a 724-dimensional row of data. The label value is a 10-dimensional vector, which is a one-hot representation of the 0-9 category number. Feature extraction and classification decision-making are the focus of handwritten digit recognition [7].

3.3. Program implementation steps

The program was written on Anaconda, the most popular open source Python data science platform, and the Python development environment is Spyder (Scientific Python Development Environment).

Figure 3 is a computational graph of the entire model structure displayed by the TensorBoard, including two convolutional layers Conv1 and Conv2, two pooling layers Pool1 and Pool2, one fully connected layer Fc1, one output layer Fc2, and Dropout is added between the Fc1 and Fc2 layers to prevent data overfitting; accuracy indicates the accuracy of the model using the test sample to predict the result and compare with the label value, cross_entropy is called the cross entropy loss function, indicating the error between the model prediction result and the label value; Adam_train is an optimization training algorithm.
3.3.1. Input layer and Initialize the weight and bias value. Two placeholders \( x \) and \( y \) are defined as inputs of the training and test sets, and the 784-dimension data are converted into a \( 28 \times 28 \) matrix image. The weight is initialized with a truncated normal distribution, and the bias value is initialized to a constant of 0.1.

3.3.2. Convolution and pooling layer. When defining the convolution operation, the step size of the convolution kernel is one both in the \( x \) and \( y \) direction. The padding='SAME'. When defining the pooling operation, the pooling window size is \( 2 \times 2 \); The weight value of the first convolution layer are initialized, the sampling window size is \( 5 \times 5 \), 32 convolution kernels extract features from one plane, and each convolution kernel has an bias value; convolve the input image \( x_{\text{image}} \) with the weight vector, plus the bias value, then applied to the relu activation function; Finally maximize the pooling of the output; similarly initialize the weight and bias value of the second convolutional layer, except that 64 convolution kernels extract features from 32 planes.

3.3.3. Fully connected layer. It can be seen from the previous procedure that the initial \( 28 \times 28 \) picture is still \( 28 \times 28 \) after the first convolution because of the SAME PADDING. Feature map becomes \( 14 \times 14 \) after the first pooling, because of the pooled \( 2 \times 2 \) window and the step size is \( 2 \); similarly after the second convolution, it becomes \( 14 \times 14 \) and after the second pooling becomes \( 7 \times 7 \). Finally, after the above operation, \( 64 \times 7 \times 7 \) planes are obtained, which is used as the input of the next fully connected layer. Initialize the weight of the fully connected layer, set 1024 neurons, 1024 bias values, and flatten the 64 feature maps into \( 64 \times 7 \times 7 \)-dimensional data, and then multiply them by the weight vector, plus the offset value. Then applied to the relu activation function to find the output of the fully connected layer.

3.3.4. Dropout layer. Perform a dropout operation to prevent data from overfitting, and use keep_prob to indicate the output probability of the neuron and tf.nn.dropout() function to achieve dropout.

3.3.5. Softmax output layer. Initialize the second fully connected layer, set 10 output neurons, and output the final classification result using the tf.nn.softmax() regression model function.

3.3.6. Loss function and Optimization training. Select a cross entropy loss function: tf.nn.softmax_cross_entropy_with_logits(), then ptimize and minimize the loss after constructing the loss and use the tf.train.AdamOptimizer() function algorithm for optimization training.

3.3.7. Accuracy. Store the classification results in a boolean list and use the argmax() function to return the position of the largest value in the tensor. And use euqal () comparison function to determine whether the result is correct, and finally obtain the classification accuracy.
3.3.8. Session execution. Due to the huge training data, the ideas of batch training are used to set up 100 training charts per batch. After training 550 times, all sample data can be trained. Sessions are created, variables are initialized, training data are fed, and train model 10000 steps, record the parameters and the accuracy of the test set per 100 steps.

3.4. Result analysis

Figure 4 shows the output of the final test after 10,000 trainings. The left picture shows a two-layer network with an accuracy of 99.15%, which is 0.46 percentage points higher than the accuracy of 98.69% of the one-layer. Figure 7 demonstrates the relation curve between the training times and the accuracy of CNN model, which is derived from the visualization tool TensorBoard, where the abscissa is the number of trainings and the ordinate is the recognition accuracy. It can be seen from Figure 5 that as the number of training increases, the accuracy gradually increases and finally stabilizes. The two-layer convolutional layer neural network, the accuracy rate reached 90% when trained to 200-300 times; when trained to 1100 times of data, the accuracy rate is 97%; when training to 6000 times, the accuracy rate is basically stable at around 99%. It can be clearly seen from the figure that the accuracy curve of the two-convolution-layers is generally higher than that of one-convolution layer; and the error of the two models is decreasing with the number of trainings, but the error curve of the two-convolution-layers neural network is significantly lower than the error of the one-convolution-layer. The above data shows that the two-convolution-layers neural network model has higher recognition accuracy than the one-convolution-layer, faster convergence, stronger ability of feature extraction and classification decision-making.

![Figure 4. Test results (two convolutional layers on the left, the one on the right )](image)

![Figure 5. Relationship diagram between training times and accuracy and error](image)

4. Conclusion

In the Spyder Python language development environment, this paper writes a program and calls the open source TensorFlow library to establish a deep learning model: CNN. The MNIST data set is used as an example to train and test the model. With the visual tool TensorBoard, the paper display this deep learning model structure, test results and trend curves, verify the validity of the model. Compared
with the classical convolutional neural network, a layer of convolution and pooling operations are added, and the second convolution pooling operation is convenient for deeper levels to extract a richer feature quantity, and it finally achieve a recognition accuracy of 99.15%.

References
[1] Zeyu Zheng., TensorFlow actual Google deep learning framework [M]. Electronic Industry Press., 2018:1-4,287.
[2] Tao Jing, Yongai Zhang., Digital recognition based on deep learning under the TensorFlow platform [J]. Information technology and network security,2018,37(04):74-78.
[3] Jiahui Xiao., The application prospect of clustering algorithm in TensorFlow platform is discussed [J]. Digital technology and applications, 2018,36(05):150+187.
[4] Ian Goodfellow, Youshua Bengio, Aaron Courville., DEEP LEARNING[M]. Posts and Telecom Press, 2018:201-203.
[5] Yujian He., Python works with machine learning [M]. Electronic Industry Press, 2017:200,221-222.
[6] Shanjie Han, Shizhe Tan., Design and implementation of a deep learning model for stock forecasting based on TensorFlow [J]. Computer applications and software, 2018,35(06):267-271+291.
[7] Yuan Xing., Application of deep learning in handwritten numeral recognition [D]. Suzhou University, 2017.
[8] Yun Xu, Haowei Yuan, Zhi Li. Handwritten numeral recognition in convolutional neural network and TensorFlow [J]. Shanghai electric technology,2018,11(01):31-34+61.
[9] Hao Chen, Hai Guo, Daquan Liu, Jiaqi Zhang., Handwritten numeral recognition system based on TensorFlow [J]. Information communication,2018(03):108-110.
[10] Panhai Zheng, Ling Guo, Libing Ding., Research and implementation of convolutional neural network based on TensorFlow [J/OL]. Electronic technology and software engineering, 2018(18):20-22[2018-10-11].
[11] Zhao J., Comparison and analysis of different deep convolutional neural networks based on TensorFlow [J]. Electronics World,2018(06):25-26.
[12] Yanlu Hou, Shifei Ding, Tongfeng Sun., The mixed depth learning model C-RF and its application in handwritten numeral recognition [J]. data acquisition and processing, 2018,33(02):343-350.
[13] F. Ertam and G. Aydn, "Data classification with deep learning using Tensorflow," 2017 International Conference on Computer Science and Engineering (UBMK), Antalya, 2017, pp. 755-758.
[14] T. Makkar, Y. Kumar, A. K. Dubey, Á. Rocha and A. Goyal, "Analogizing time complexity of KNN and CNN in recognizing handwritten digits," 2017 Fourth International Conference on Image Information Processing (ICIIP), Shimla, 2017, pp. 1-6.
[15] M. A. R. Ali, S. Ahmed and M. A. Hasan, "Isolated Bangla handwritten character recognition with convolutional neural network," 2017 20th International Conference of Computer and Information Technology (ICCIT), Dhaka, 2017, pp. 1-6.
[16] S. Iamsa-at and P. Horata, "Handwritten Character Recognition Using Histograms of Oriented Gradient Features in Deep Learning of Artificial Neural Network," 2013 International Conference on IT Convergence and Security (ICITCS), Macao, 2013, pp. 1-5.