Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/6/20230897

Recognition of handwritten digital neural network construction and improvement

Zhenqun Shao

Faculty of Engineering, University of Bristol, Bristol, England, United Kingdom BS8 1TH

Wohelijiaweishi@iCloud.com

Abstract. More and more neural network algorithms are being invented and improved, but the most basic algorithms are still useful in learning or practice areas. This essay is about building a digital recognition neural network from scratch, whose implementation has a completely original core code and details and precisely elaborates the whole process of basic neural network construction through detailed mathematical derivation, combined with the idea of programming. Furthermore, this project deeply researched and analyzed recognition accuracy according to the identified data of the neural network built for the project, then improved the algorithm used by increasing the accuracy from 86.93% to 99.1% and finally successfully explained the role of raw data preprocessing.

Keywords: Neural Networks, Recognizing Handwritten Digits, Backpropagation, One-hot Encoding, Python Implementing Neural Networks.

1. Introduction

This article examines the use of the Python base package to implement a neural network for recognizing handwritten digits. The development of neural networks is very rapid, and the theory of neural networks in the learning process is mainly presented in mathematical form, and the realization of a complex neural network can be used by the built package. At present, backpropagation neural networks have been used in various aspects, such as military, medical, biotechnology, etc. However, there is very little information on the implementation of neural networks in basic code. Since David E. Rumelhart proposed this theory, its development has been quite obvious, and many variants have increased the rate of calculation according to its extended variants. At present, the main focus of the paper is how to apply backpropagation, such as physics, biotechnology and other fields, but there is little in-depth analysis of the method of algorithm implementation and the reasons for the specific implementation details [1][2]. This paper applies this technique to image recognition, using easy-to-understand language, transforming the mathematical theory of neural network construction into a form that is easy to write in Python code, explaining the important and necessary characteristics of image recognition algorithms in the early stages of processing images. It expands the scope of neural network applications and enables beginners to learn and deeply understand backpropagation algorithms. This article combines graphics and text to explain the computational process of each element in a multi-layer neural network. In terms of specific implementation, explain the propagation syntax of Python to implement matrix fast operations, one-hot

DOI: 10.54254/2755-2721/6/20230897

encoding of data processing, etc. Beginners can combine code to deeply understand the algorithm and facilitate learning neural networks.

2. Building a neural network

Learning how to build a neural network from scratch is the foundation of machine learning, and it also tests the ability to use computer languages. This time, Python and Mnist datasets, as well as the backwards propagation algorithm, are used to construct a neural network capable of recognizing handwritten words [3].

2.1. Data and data pre-processing

The dataset used for this project is Nmist, which contains 60,000 training data and 10,000 testing data [3]. A large amount of data facilitates the training of the underlying neural network. First, the dataset consists of images followed by a number tag to record what number it is. The part of preprocessing the picture requires three steps, respectively to tenser, normalize , flatten, and one-hot encoding.

Normalize

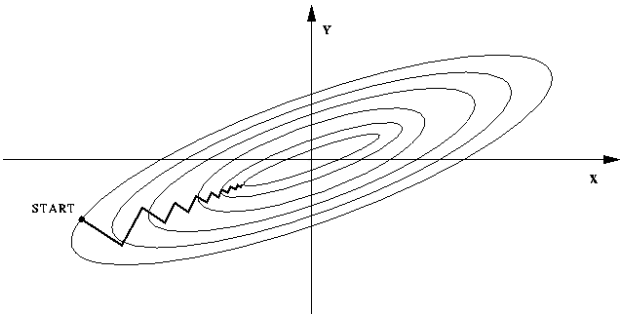

Figure 1. Before normalize [4].

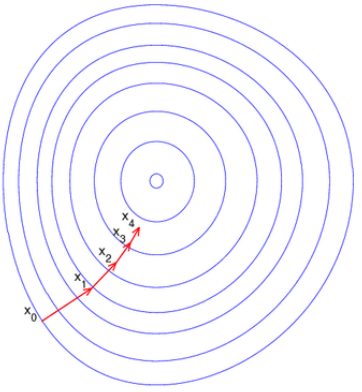

Figure 2. After normalize [4].

The data of the image is often centerless, so its descending gradient is an ellipse (figure1). When the neural network tries to move in the vertical direction of the gradient tangent, the repeated reciprocating motion will reduce the learning efficiency. After normalization, the gradient of the data closes to a circle (figure 2). This is very helpful to get a more accurate neural network [5]. The picture in the data is composed of 28*28 pixels. It cannot be used directly by the neural network. It needs to be converted into a one-dimensional vector before it can be passed into the neural network one by one.

DOI: 10.54254/2755-2721/6/20230897

2.2. *Model and algorithm*

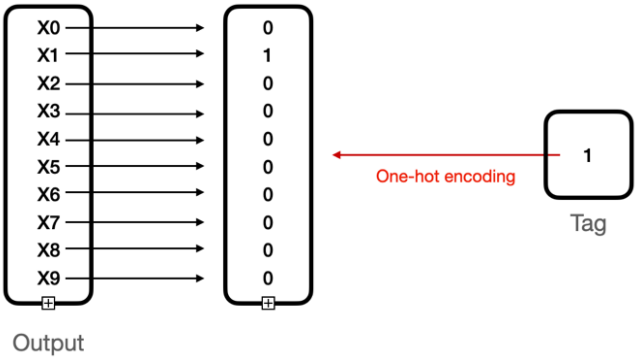

Figure 3. One-hot encoding.

One-hot encoding can convert decimal 0-9 numbers into 10 of 1-bit outputs. This matches with the output of the neural network to be designed next, which has ten outputs, each representing the likelihood of being recognized as a particular number [6][7]. As shown in Figure 3, when the tag is 1 for one-hot encoding, only the second output value is 1.

2.2.1. *Forward propagation*

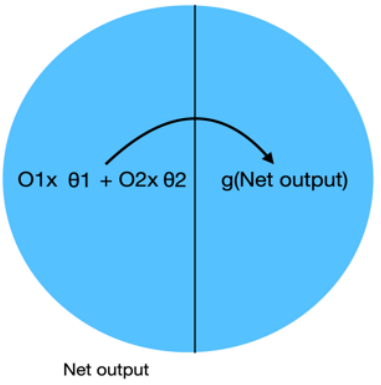

Figure 4. Forward propagation.

Each layer of neural network should have input value, net-output, output value and weight coefficient. Only the weighted output is called net-output (figure 4). It needs to go through the activation function to get a number less than 1 and greater than 0 to be the output of this layer of the neural network [8][9]. Activation function:

$$h_\theta(x) = \frac{1}{1+e^{-x}} \tag{1}$$

In general, the final output of a node should be:

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \tag{2}$$

other nodes as well

Thus, the output of this layer can be the input of the next layer. Forward propagation was built like this. At the end of network, 10 output illustrate which number it could be.

DOI: 10.54254/2755-2721/6/20230897

2.2.2. Backward propagation Cost refers to the difference between this judgment and the real value. To close this gap by changing the weight, you can find the appropriate weight by finding the gradient of the partial derivative of the weight with respect to the cost (figure 5). In other words, the relationship between the cost and the change of the weight is obtained, and because of the expectation of the reduction of the cost, it is obtained how the weight should change [8][9]. According to the chain rule:

$$\frac{\partial C_{total}}{\partial \theta_1} = \frac{\partial C_{total}}{\partial Out_1} \times \frac{\partial Out_1}{\partial Net_1} \times \frac{\partial Net_1}{\partial \theta_1} [7] \tag{3}$$

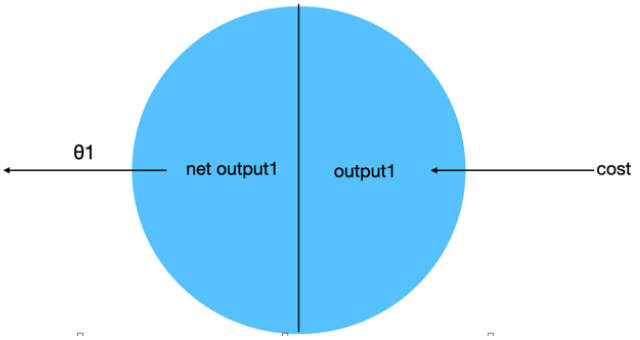

Figure 5. Backward propagation.

It is also needed to find the partial derivatives with respect to the weights of the latter layer (figure 6). The only difference is that all costs need to be considered.

$$\frac{\partial C_{total}}{\partial C_{Last}} = \frac{\partial C_{total}}{\partial Out_{Last}} \times \frac{\partial Out_{Last}}{\partial Net_{Last}} \times \frac{\partial Net_{Last}}{\partial \theta_{Last}} \frac{\partial C_{total}}{\partial Out_{Last}} = \sum_{i=0}^{9} \frac{\partial C_i}{\partial Out_{Last}} \tag{4}$$

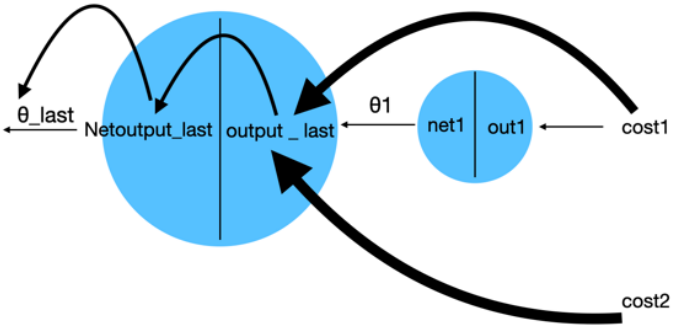

Figure 6. Further backward propagation.

2.3. Programming

For programming, the layer can be defined as a class. Any neural network can be constructed by instantiating multiple classes of layer. When the backpropagation function is called, the parameter is passed as the partial derivative of the cost to the layer output, because according to the chain rule, it is only necessary to multiply the past parameter by the partial derivative of the output of this layer to the weight, which is the partial derivative of cost to the weight. In more detail, the calculation of the matrix is the key to saving training time [10]. According to the broadcasting principle of Python, matrices of different shapes are compatible with each other.

For example, a 4*3 matrix can be added to a 1*3 matrix, python will automatically add the corresponding dimensions and repeat the operation [11] (figure7).

DOI: 10.54254/2755-2721/6/20230897

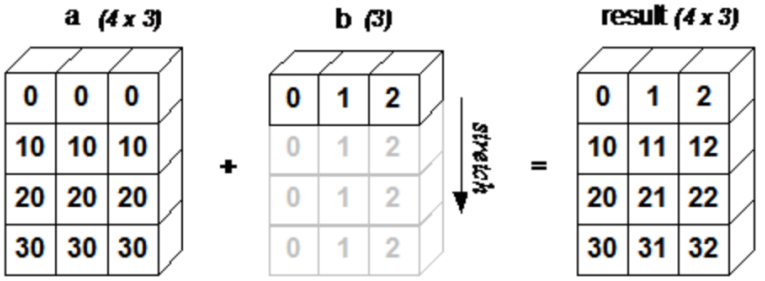

Figure 7. Broadcast [11].

3. Training and Improving neural networks

3.1. Analysis

After many experiments, the neural network's recognition of 3 and 8 is 86.93% and 87.17%, respectively, which is much lower than the recognition rate of other figures (table 1). Number 3 is always recognized as number 5 (figure 8). Preliminarily, this is because the handwriting of 3 and 5 is relatively the same, and there is only some difference in the link position of the first line. If 3 is shifted to the left, it is very likely to be recognized as 5.

Table 1. Experimental results.

Number	Number of digitals in one test	Wrong number	Percentage of wrong	Correct percentage
0	980	48	4.90%	95.10%
1	1135	38	3.35%	96.65%
2	1032	54	5.23%	94.77%
3	1010	132	13.07%	86.93%
4	982	76	7.74%	92.26%
5	892	63	7.06%	92.94%
6	958	50	5.22%	94.78
7	1028	77	7.49%	92.51%
8	974	125	12.83%	87.17%
9	1009	73	7.23%	92.77%

Figure 8. Number 3 and 5.

3.2. Hypothesis and verification

The number was rotated randomly fronm 0 to 90 degrees (figure 9) and trained as the input value [12]. The accuracy of the final recognition was greatly improved to 99.1%, and there was no inaccurate recognition of a single word. Thus, processing of the raw data can improve the accuracy of the digit recognition of the neural network.

DOI: 10.54254/2755-2721/6/20230897

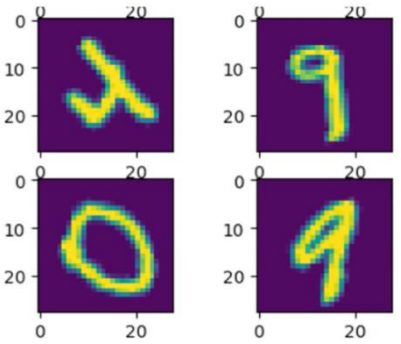

Figure 9. Digital result after rotation.

4. Conclusion

In summary, this paper presents a simple and convenient neural network for recognizing handwritten digits. In this process, the method of back-propagation derivation is emphasized, and the broadcast feature of Python is used to solve the effect of long fitting time and data preprocessing to improve the recognition accuracy which goes from 86.93% increase to 99.1%. It is necessary to preprocess the graphics as much as possible to improve the accuracy of recognition. Although this paper discusses the problems of data preprocessing and introduces an improvement method, it does not delve into the algorithm and improvement principle of processing in depth. At the same time, this article does not involve the most cutting-edge technology to solve the image recognition problem, nor can it most intuitively explain the cutting-edge technology. Future research will study cutting-edge algorithms in more depth and strive to solve problems with the most advanced technology.

Acknowledgment

The reason for this article is that I am first thanked by Professor who led me through my study of neural networks. Secondly, Teacher Ming Liu, who let me discover my own shortcomings and gave me guidance and encouragement, so that I could complete this article.

References

[1] Behler, J. (2021). Four generations of high-dimensional neural network potentials. Chemical Reviews, 121(16), 10037-10072.

[2] Marvin, G., & Alam, M. G. R. (2022, February). Explainable Augmented Intelligence and Deep Transfer Learning for Pediatric Pulmonary Health Evaluation. In 2022 International Conference on Innovations in Science, Engineering and Technology (ICISET) (pp. 272-277). IEEE.

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

[4] Ljtyxl, (2019) Jacobian and Hessian, LM optimization methods https://blog.csdn.net/u014033218/article/details/88680720

[5] He, C. L., Zhang, P., Dong, J. X., Suen, C. Y., & Bui, T. D. (2005, August). The role of size normalization on the recognition rate of handwritten numerals. In The 1st IAPR TC3 NNLPAR workshop (pp. 8-12).

[6] Seger, Cedric. (2018). "An investigation of categorical variable encoding techniques in machine learning: binary versus one-hot and feature hashing."

[7] Okada, S., Ohzeki, M., & Taguchi, S. (2019). Efficient partition of integer optimization problems with one-hot encoding. Scientific reports, 9(1), 1-12.

[8] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. nature, 323(6088), 533-536.

[9] Matt Mazur. (2021) A Step by Step Backpropagation Example

DOI: 10.54254/2755-2721/6/20230897

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/
[10] Erik Learned-Miller Vector, Matrix, and Tensor Derivatives, (2015) http://cs231n.stanford.edu/vecDerivs.pdf
[11] NumPy documentation Version: 1.23 https://numpy.org/doc/stable/user/basics.broadcasting.html
[12] Weiler, M., Hamprecht, F. A., & Storath, M. (2018). Learning steerable filters for rotation equivariant cnns. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 849-858).