Advances in Intelligent Systems Research, volume 161
International Conference on Transportation & Logistics, Information & Communication, Smart City (TLICSC 2018)

Recognition and Optimization Algorithm of MNIST Dataset Based on LeNet5 Network Structure

Hailong Xi[a], Haiyan Liu, Yu Zhang
Army Armored Force Academy, Beijing 100072, China
[a]hailong006@foxmail.com

Abstract. This paper analyzes the content composition and detailed structure of the MNIST dataset, introduces the overall level of the traditional LeNet5 network, and proposes an improved algorithm for its hidden layer, optimization function, activation function, etc. of the network layer structure. Optimization, and the effectiveness of the algorithm is verified by experiments. This paper has certain reference value for traditional neural network optimization and improving the correct classification rate of MNIST data sets.

Keywords: LeNet5 network, MNIST data set, optimization function.

1. MNIST Dataset

The MNIST dataset is derived from the National Institute of Standards and Technology (NIST), and it contains 70,000 handwritten digits of grayscale images, each of which is represented by 28 x 28 pixels. The training set consists of 250 handwritten numbers which the high school students and the Census Bureau staff each account for half. The test set is also the same proportion of handwritten digital data. The dataset contains a total of four parts:

Table 1. MNIST dataset content map

Training set images: train-images-idx3	ubyte.gz (9.9 MB, 47MB after decompression, containing 60,000 tags)
Training set labels: train-labels-idx1	ubyte.gz (29 KB, 60KB after decompression, containing 60,000 tags)
Test set images: t10k-images-idx3	ubyte.gz (1.6MB, 7.8MB after decompression, containing 10,000 tags)
Test set labels: t10k-labels-idx1	ubyte.gz (5KB, 10KB after decompression, containing 10,000 tags

The internal file structure of the MNIST training set is as follows:

Table 2. MNIST training set file structure diagram

TRAINING SET LABEL FILE (train-labels-idx1-ubyte):			
[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
....			
xxxx	unsigned byte	??	label
The labels values are 0 to 9.			

From the file structure we can see that the data is stored in binary form, each tag value is 1 number between 0 and 9, and the real data is only [value], and the other [type] are only description items, but not a real data file.

2. The Model of LeNet5

2.1 Model Structure.

The LeNet5 model is a neural network structure of handwritten font recognition proposed by Yann LeCun in 1998. It was applied to bank check digital recognition early and achieved good results. The model framework is as follows:

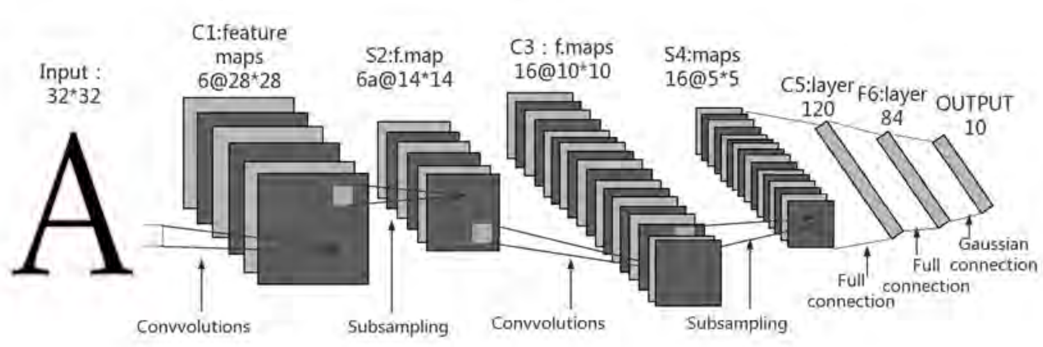


Fig. 1 LeNet5 neural network structure

The LeNet5 network consists of 8 layers, convolutional layers and pooled layers cross-combination, and finally outputs predicted values through the fully connected layer, each of these layers contains trainable connection weights. The CNN passes through the local receptive field (ie, the neuron input of the upper layer of the local domain accepted by each layer of neurons), the weight sharing (the set of weights of the local receptive field at different positions of the image is set to the same value) and the downsampling (ie, reduce the resolution of the feature map by the pooling layer, reduce the sensitivity to output displacement and deformation) to achieve displacement, scaling, and deformation invariance.

C1 layer: Select 6 5×5 convolution kernel pairs to extract 32×32 input features, and obtain 28×28=784 neurons ;

S2 layer: Using a pooling block of size (2, 2), compressing the image to 14×14;

C3 layer: The 5×5 convolution kernel is still used. After compressing a single picture to 10×10, multiple pictures are weighted and combined:

$$P_i = w_{1i} \cdot x_{1i} + w_{2i} \cdot x_{2i} + \cdots + w_{25i} \cdot x_{25i} \tag{1}$$

and get the result of 16*(10*10) through function mapping:

$$P = P_1 + P_1 + \cdots + P_6 \tag{2}$$

$$P = WX \tag{3}$$

$$Out = f(P + b) \tag{4}$$

Where b is the bias term and f is the activation function. The total number of convolution parameters required to obtain 16 feature maps is 16*(6*(5*5))=16*6*(5*5).

S4 layer: use 2*2 pooling block to maximize the pooling of C3 layer output, reduce the number of neurons to 16*5*5=400.

C5 layer: The convolution operation of the upper layer output is performed by a 5*5 convolution kernel, and 120 1*1 neurons are obtained.

The F6 layer is a fully connected layer with a tangent function and the last layer is an output layer. The above 8 layers make up the entire structure of the LeNet5 network.

2.2 Training Process.

Lenet5's training process is mainly divided into two parts. One is forward propagation. A training data (trnData, trnLabel) is taken from the training set, and inpute trnData into the neural network. After multiple convolutions and pooling, the actual data output_real is obtained; the second is backpropagation, and the error function of the actual label reaLabel and the ideal label trnLabel is calculated. The back propagation uses the method of minimizing the error, and the interlayer parameters are adjusted according to the error between each layer and the upper layer.

3. Experimental Design

Hardware condition: CPU Inter Core i7-7700HQ@2.8GHz

Operating system: Windows10

Software and environment: Pycharm2017, TensorFlow.

3.1 Better Performance Activation Function—Leaky ReLU.

Traditional activation functions use the Sigmoid function and the Tanh function, and the former is even called the core of the neural network. The typical feature of the Sigmoid function is that the gain of the center signal is large, but the nonlinear region is small. The spatial mapping of the signal has a good effect when the features are not very different. However, the characteristics of such two ends tend to be saturated, which makes them slower in gradient extraction and prone to gradient disappearance, thereby limiting network performance. The Tanh function expression is:

$$f(z) = \tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} \tag{5}$$

$$\tanh(x) = 2sigmoid(2x) - 1 \tag{6}$$

The Tanh function works well when the feature difference is obvious, and its feature effect will continue to expand in the loop. Although its 0-means is relatively better than sigmoid, it also has the characteristics of slow convergence.

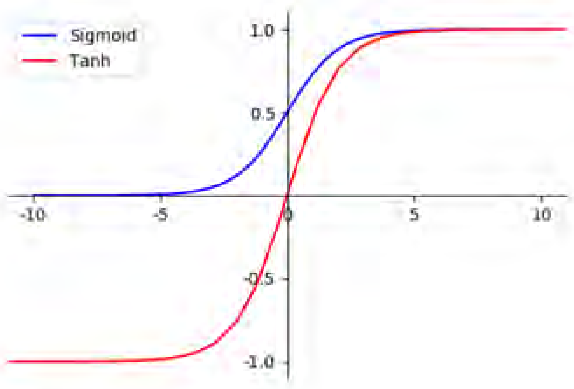


Fig. 2 Sigmoid, Tanh function image

ReLU (Rectified Linear Unit) has the following form:

$$\phi(x) = \max(0, x) \tag{7}$$

Using Relu as the activation function can save a lot of calculations in the whole process and has the characteristics of faster convergence; the output is 0 when the input signal is <0, the output is equal to the input when the input is >0, and the characteristic band of the left output is The sparseness of the network can be mitigated by reducing the interdependence of parameters.

If there is a large gradient in the left part of the original Eco Relu function, the secondary neuron gradient will be forced to 0, and the neurons in the subsequent process cannot be activated, which will result in the loss of data diversification. In order to avoid a situation where the input is less than zero, the non-zero slope is given to all negative values, and the improved Relu function is Leaky-Relu:

$$y_i = \begin{cases} x_i & if\ x_i \geq 0 \\ \dfrac{x_i}{a_i} & if\ x_i \leq 0, \end{cases} \tag{8}$$

Where a_i is a fixed parameter of (1, +∞), which can be randomly generated from the Gaussian distribution, where α=0.01 is selected.

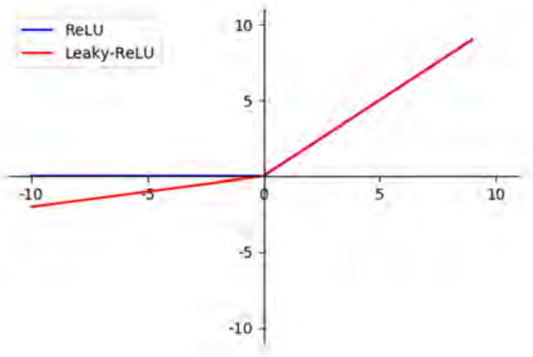


Fig. 3 Relu, Leaky-Relu function image

3.2 Adam Optimization Function.

The Adam (Adaptive Moment Estimation) algorithm is proposed on the advantages of the comprehensive adaptive gradient algorithm (AdaGrad) and the root mean square propagation (RMSProp) algorithm. Different from the traditional stochastic gradient descent algorithm, the Adam algorithm does not use a single learning rate to update the all weights. Instead, the first-order moment estimation and the second-order moment estimation of the gradient are used to set independent adaptive learning rates for different parameters. After the offset correction, the learning rate is fixed within a certain range, and the parameters are relatively stable. It is essentially an RMSprop with a momentum term.

$$
\begin{aligned}
&m_t = \mu * m_{t-1} + (1-\mu) * g_t \\
&n_t = v * n_{t-1} + (1-v) * (g_t)^2 \\
&\hat{m}_t = \frac{m_t}{1-\mu^t} \\
&\hat{n}_t = \frac{n_t}{1-v^t} \\
&\Delta\theta_t = -\frac{\hat{m}_t}{\sqrt{\hat{n}_t} + \varepsilon} * \eta
\end{aligned}
\tag{9}
$$

Where m_t, n_t are the first moment estimate and the second moment estimate of the gradient, respectively, and can also be regarded as an estimate of the expected $\mathrm{E}[g]_t, \mathrm{E}[g^2]_t$; $\hat{m}_t, \hat{n}_t$ is the amount of correction for m_t, n_t, which can be approximated as an unbiased estimate of the desired. It can be seen that the direct estimation of the moment of the gradient has no additional requirements on the memory, and can be dynamically adjusted according to the gradient, and $-\frac{\hat{m}_t}{\sqrt{\hat{n}_t} + \varepsilon} * \eta$ forms a dynamic constraint on the learning rate, and has a clear range.

The parameters used in this experiment were tested repeatedly:

μ(exponential decay rate of first moment estimation)=0.9;

v(the attenuation rate of the second moment estimate) = 0.999;

ϵ (Control zero parameter) = 1e-8.

The main implementation code is as follows:

```
first_momment=beta1*first_momment+(1-beta1)*dx# First moment estimate
second_momment=beta2*second_momment+(1-beta1)*dx*dx# Second moment estimation
first_unbias=first_momment/(1-beta1**t)# Correction
second_unbias=second_momment/(1-beta2**t)
x-=self.yita*first_unbias/(np.sqrt(second_unbias)+1e-7)# Parameter update
```

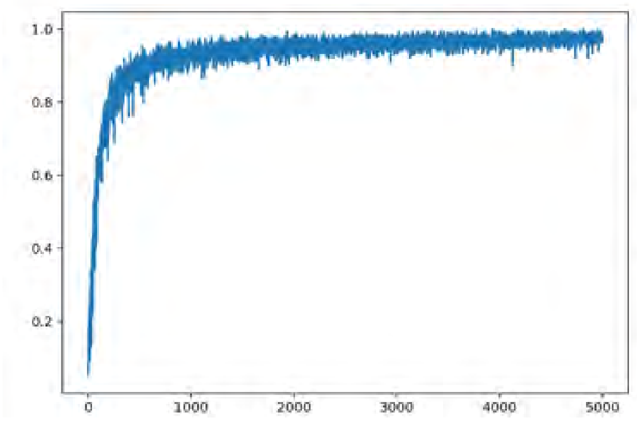
Adam has the following advantages: the algorithm reduces the need for memory while achieving efficient and simple calculation; the gradient transformation does not affect the parameter update; the initialization of the learning rate can limit the update range of the step; adaptive update learning The rate is more efficient in annealing.

3.3 Increase the Number of Convolution Layer Features.

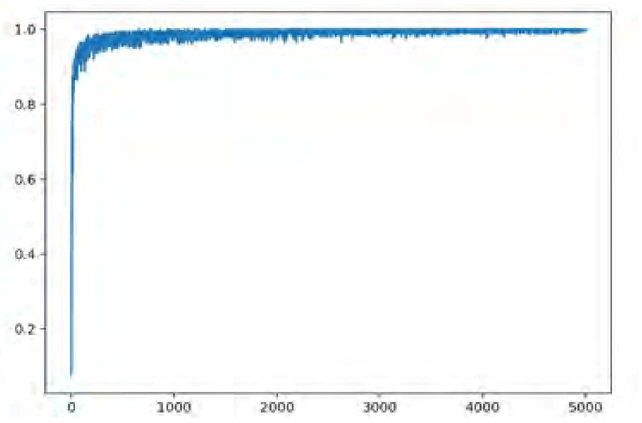
The experiment changed the number of hidden layer neurons in the traditional LeNet5 network, changed the C1 layer to a convolutional layer with 32 channels, and changed the C3 layer to a convolutional layer of 64 channels, improving the training efficiency by reasonably increasing the number of neurons in the hidden layer.

4. Experimental Results and Analysis

The experiment improved the network structure based on the LeNet5 algorithm, and adopted the Leaky-Relu activation function to reverse the optimization of the Adam algorithm. The handwritten digital MNIST database was used as the input data to control different variables for multiple experiments, and the classification accuracy was tested, the result is shown as below:



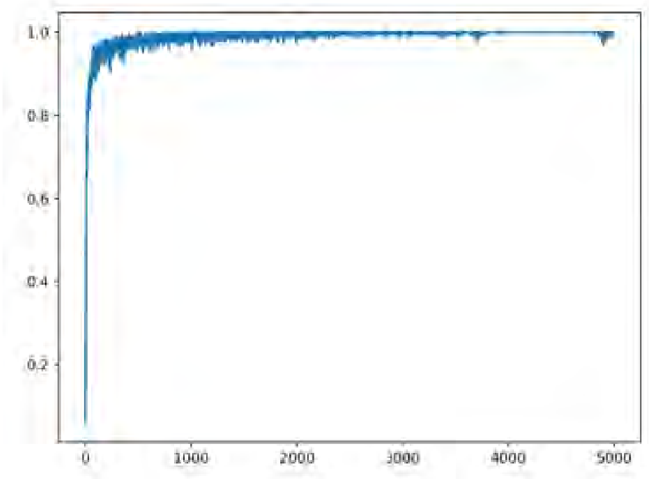
(a)

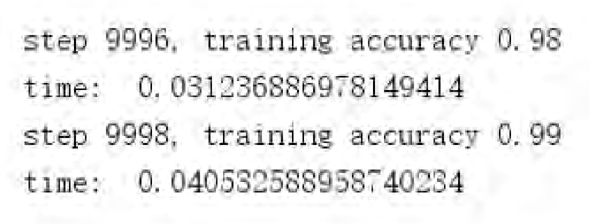


(b)

Figure 4.a Original LeNet5 network convergence rate and accuracy

Figure 4.b Convergence rate and accuracy after using improved activation and optimization functions





```
step 9996,  training accuracy 0.98
time:   0.031236886978149414
step 9998,  training accuracy 0.99
time:   0.040532588958740234
```

(c)

Figure 4.c Convergence rate and accuracy after optimized LeNet5

It can be seen from the experimental results: (1) The improved LeNet5 learning network has a faster convergence rate under the same number of iterations; (2) the classification accuracy rate of the MNIST data set is close to 99%, and with the number of iterations increased, the later test results are more stable. Therefore, it can be proved that the optimization algorithm proposed in this paper can effectively improve the recognition rate of MNIST compared with the original LeNet5 network.

5. Summary

The recognition problem of MNIST handwritten fonts not only has positive practical significance, but also is a representative problem of neural network pattern recognition research. In this paper, the traditional algorithm is improved based on LeNet5 network, and experimental research is carried out to prove that this algorithm has better performance. The experimental results have certain reference value for the traditional handwritten font recognition.

References

[1]. Kussul E, Baidyk T. Improved method of handwritten digit recognition tested on MNIST database[J]. Image & Vision Computing, 2004, 22(12):971-981.

[2]. Deng L. The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web][J]. IEEE Signal Processing Magazine, 2012, 29(6):141-142.

[3]. Cuesta-Infante A, García F J, Pantrigo J J, et al. Pedestrian detection with LeNet-like convolutional networks[J]. Neural Computing & Applications, 2017(3):1-7.

[4]. Yong L I, Lin X Z, Jiang M Y. Facial Expression Recognition with Cross-connect LeNet-5 Network[J]. Acta Automatica Sinica, 2018.

[5]. Zhang X, Zou Y, Shi W. Dilated convolution neural network with LeakyReLU for environmental sound classification[C]// International Conference on Digital Signal Processing. IEEE, 2017:1-5.

[6]. Fan E G. Extended Tanh-function Method and its Applications to Nonlinear Equations[J]. Physics Letters A, 2000, 277(4):212-218.

[7]. Y Lecun, C Cortes. The mnist database of handwrittendigits[J]. Intelligenza Artificiale A.A,2012-2013.

[8]. Li Deng. The MNIST database of handwritten digit images for machine learning research[J]. IEEE Signal Processing Magazine,2012:141-142.