CNN-Based Recognition of Handwritten Digits in MNIST Database

Huimin Wu

Research School of Computer Science, The Australia National University, Canberra
{Huimin Wu}@u6342493@anu.edu.au

Abstract. For humans, identifying numbers or items in a picture is extremely simple, but how do you train the machine to recognize these different things in images? Convolutional Neural Networks (CNN) can solve this problem. In this report, a convolutional neural network has been trained by identifying pictures in MNIST handwritten digital database to predict exactly what the numbers in the picture are. Obviously, human beings can perceive that there is a hierarchy or conceptual structure in the image, but the machine does not, for example the trained neural network is inconvenient to deal with special changes in a position of numbers in digital pictures. Exactly put, no matter what the environment of the image (image background) is, it is unchallenging for human beings to judge whether there is such a figure in the image and it is unnecessary to repeat the learning training.

1 Introduction

In recent years, the development of neural networks has been extremely rapid in the field of pattern recognition system. We use a common pattern recognition technique in this article and use convolution to improve this technique. Then we will introduce the entire project from the data sets used, the neural network we built, and the CNN improvement methods (LeNet-5) used. The neural network I wrote before cannot recognize images without repeat training. It reckons that the ‘number’ appears at different places in a picture is not the same number. The neural network cannot understand the concept of whether an object appears in a different position in the picture is different objects or the same objects. That means the neural network must re-learn to identify various objects in every possible position. How to let neural network understand the concept of ‘translation invariance’, that means ‘0’ is ‘0’, no matter where it appears in the images. In this report, I used a convoluted solution to achieve the goal.

1.1 The MNIST Database

The MNIST database of handwritten digits contains 60,000 training examples and 10,000 testing examples, which are 28 * 28 images. All of digits have already been size-normalized and preprocessed and formatted (LeCun et al., 1998). The four files provided on the website are used in the training and testing for neural networks. In the process of loading the data set can be directly called from the MNIST database, but due to the requirements of the assignment, I downloaded these image files from the website that provides the data set. Because downloading browsers may unzip these image collection files without your attention, this operation may cause the downloaded files to be larger than previously mentioned. Thus, if you need to see some problems with the original image set or data set, you can view the original site of the data set via the link provided in the reference section of the paper. Due to the use of Python's own data set, simplifying the section on data preprocessing in the code. The images are all centered in 28 * 28 field.

1.2 Convolutional Neural Networks

Due to the selection of the data set, we decompose the picture into 28*28 blocks of the same size. According to the original trained neural network, we input a complete picture into the neural network. But for CNN, the pixel block is directly input this time. The same neural network weight will be used for every small tile. If any small tile has any abnormality, we think the tile is interested. In this neural network, there is no order in which small tiles are disturbed, and the results are still saved in the order of input. Then we will get a sequence. The part where the picture is stored is interesting. Since the array is generally large, we will first down sample it to reduce the size of the array. Find the max value in each grid square in our array. Finally, the column will be inputted into the Fully Connected Network and the neural network will determine if the picture matches.

1.3 LeNet-5 of CNN

We know that for a given image, given a convolutional kernel, the convolution is the weighted summation of the pixels based on the convolution window. The convolutional neural network is a special multilayer neural network that is trained using a back-

propagation algorithm. This model is designed to recognize patterns directly from pixel images and can be used to identify extreme changing patterns, such as handwritten characters. LeNet-5 is a convolutional network used in this article.

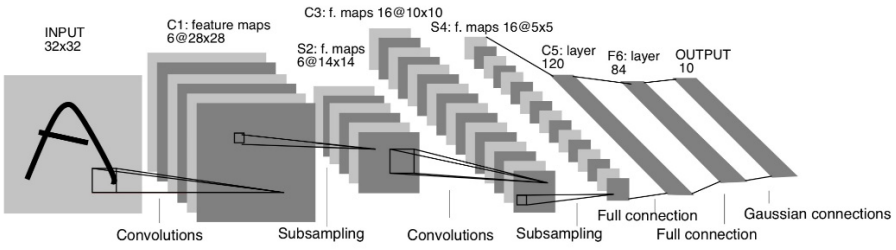
2 Method

In the previous section we have already mentioned that datasets are imported directly. Here we explain how this neural network is defined in the code. This script uses a custom built neural network. I created a hidden layer of neural network. Input layer has 28 * 28 neurons, representing the size of images. Hidden layer has 100 neurons using Sigmoid as activation function. Output layer has 10 neurons, representing the classes of handwriting images.

After the model is built, a random gradient descent training is performed according to the Stochastic Gradient Descent (SGD) training algorithm. Use SGD training algorithm to adjust the weight of the connection between neurons so that the loss reaches a minimum value or stops after a set number of epochs.

2.1 LeNet-5

In this paper, I mainly use a classic structure of CNN, LeNet-5, to identify handwritten patterns. About this article can be found in the reference section, and the following image is also from this article, followed by my explanation of the method (LeCun et al., 1998).



Input: 32*32 handwritten digital pictures, and handwritten numbers contain 0 to 9.

Layer C1: The paper author selected six feature convolution kernels, the size is 5*5, and the size of the feature map is 32-5+1 = 28, so the number of neurons is 6*28*28 = 784.

Layer S2: This is a downsampling layer, the code is used in max pooling, max pooling size is (2, 2), so we block the layer C1 layer 28*28 image, each block size (2 * 2), and finally Can get 14*14 blocks, there are 6 such pictures. The result can be calculated by the sigmoid function.

Layer C3: This layer is a convolutional layer, the size is 5*5, so the new image size obtained is 14-5+1 = 10. Then we get 16 10*10-pictures by weighted combination. The layer S2 to layer C3 calculation method is:

$$Out = f(P + b),\ P = P1 + P2 + \ldots + P6 = WX$$

b represents the offset top, f is the activation function, and W is the parameter to learn.

Layer S4: This layer is the downsampling layer, the largest pool of C3 pictures, the size of (2 * 2), and finally get 16 pictures of 5 * 5 size, this time the number of neurons is 400.

Layer C5: Convolution layer. Get 120 neurons, and finally deal with a multi-layer sensor (full-connection neural network).

Layer F6: Fully connected with C5 layer. Pass the calculated dot to the sigmoid function to create a state of unit i.

3 Results and Discussion

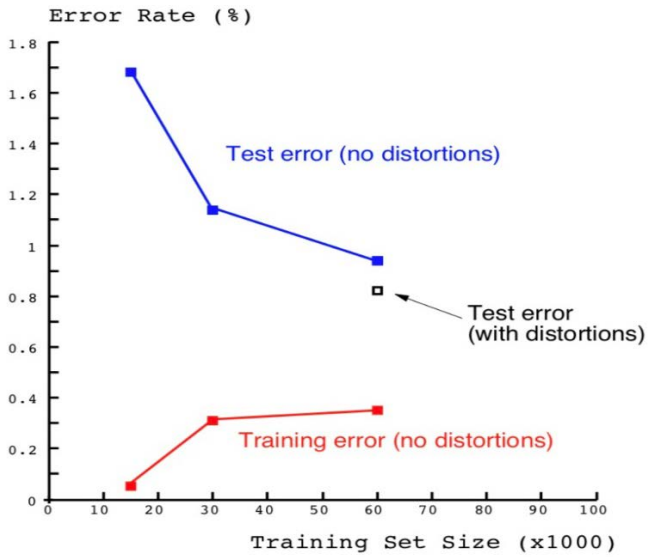
According to the model given above, the final accuracy is stable at more than 90%. In fact, for many tasks, the model is prone to over-training. If over-training occurs during training, the error caused by training will decrease with the increase of training times. The following figure shows the accuracy of my model when the epoch is 100 times.

```
current epoch: 99
Epoch [100/100] Loss: 0.0676  Accuracy: 100.00 %
Epoch [100/100] Loss: 0.1598  Accuracy: 95.00 %
Epoch [100/100] Loss: 0.0744  Accuracy: 100.00 %
Epoch [100/100] Loss: 0.1356  Accuracy: 96.67 %
Epoch [100/100] Loss: 0.2126  Accuracy: 91.67 %
Epoch [100/100] Loss: 0.1401  Accuracy: 96.67 %
Epoch [100/100] Loss: 0.2838  Accuracy: 90.00 %
Epoch [100/100] Loss: 0.1862  Accuracy: 91.67 %
Epoch [100/100] Loss: 0.1351  Accuracy: 96.67 %
Epoch [100/100] Loss: 0.2925  Accuracy: 91.67 %
```

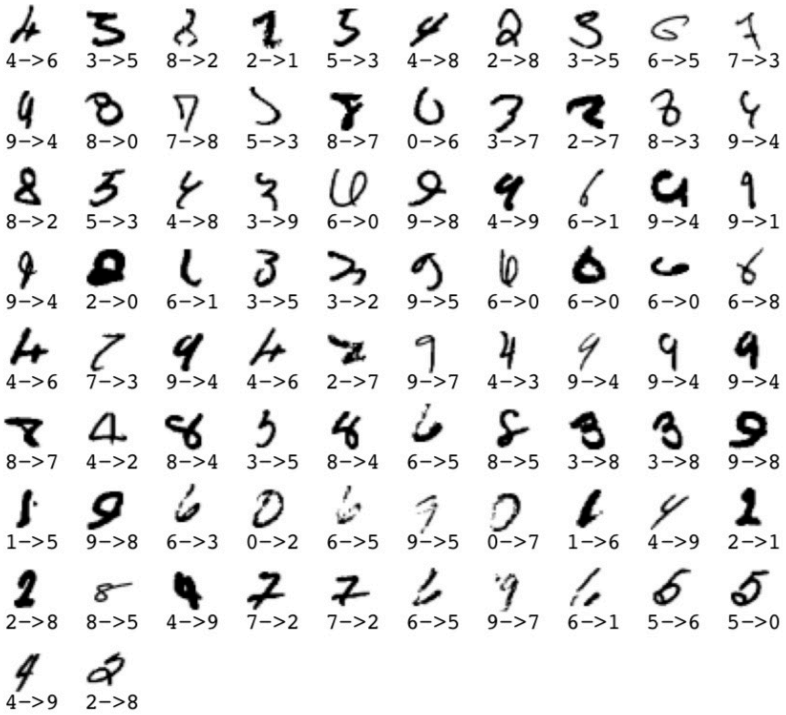
```
/anaconda3/lib/python3.6/site-packages/ipykernel_la
has been deprecated. Change the call to include dir
```

```
Testing Accuracy: 94.00 %
```

Training set size also affects the accuracy, and accuracy increases as the amount of data increases. As shown in the figure below, the more data, the more data in the training set, the smaller the impact of training error and test error, and ultimately the accuracy can be improved. We can find the larger training set can develop the performance of LeNet-5. After improving this problem, our accuracy can reach 94%. If we increase the number of training, this accuracy will be improved to some extent.



But even if you increase the training set, you will find that there is no way to achieve 100% accuracy. I attributed this part to the fact that some hand-written digits are too illegible. For this problem, we can understand that even if people recognize symbols and things on some pictures, they will encounter some patterns that are difficult to judge their specific meanings. The figure below shows 82 misclassified test cases.



We can find these test cases that show failed classifications, and there are some humans who can hardly determine the number of test cases. Although some can be judged, some of the numbers have not been identified because of limitations in the input of standard data images. But I think most patterns can be solved by increasing the training set and increasing the standard number pattern. There are also missing pixels caused by, for example, image compression, or the classification errors caused by image sharpness problems may be difficult to solve by the two solutions I propose.

By comparing with the results in the paper, we have almost achieved the main research methods mentioned in most of the articles, the accuracy of the prediction is maintained

at more than 90%, and data fluctuation due to overfitting will not occur.

4 Conclusion and Future Work

This report is a completely new study for me. I chose the most famous handwritten digit recognition in the CNN model, which is also the earliest prototype of the convolution model. The datasets and reference papers were all from LeCun et al. This model is also the most classic model for image information processing. It also encountered some problems during the construction of LeNet-5 neural network. But those high-level interfaces for building neural networks in Python gave me a lot of convenience and came up with the training results mentioned in the references. But no doubt there is still much room for improvement in my identification of neural networks.

The digits patterns used in this training are all black and white images, so this model may be difficult to deal with common color patterns. The training set of color images may lead to an increase in the degree of ambiguity in the image, which in turn affects the accuracy of the prediction. Moreover, in this paper, I only studied the pictures in the MNIST data set. In actual problems, the specification and definition of the pictures are undoubtedly important factors affecting the accuracy of the model. Therefore, how to preprocess is very important, but in order to simplify the research process, I omitted this step. And in real life, most of the pictures we touch are very complicated and colorful. Pictures are often background. The digits that you want to train and learn may only occupy a small part of the complete picture. How to deal with the size of these pictures and find out which part of the model you want to identify is also a problem. In addition, the standard item pattern is difficult to be standardized. In the paper we only discuss simple numbers, but if we need to identify images of some animals, we need to consider a three-dimensional object from different angles. The difference between the flat images shown. In the future research, the neural network model also needs to be considered in combination with actual conditions. We can further study to improve the accuracy of CNN by improving the issues I mentioned above.

5 Reference

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

LeCun, Y., Cortes, C. & Burges, C. J. C. (1998) *The MNIST database of handwritten digits* [Data files]. Retrieved from: http://yann.lecun.com/exdb/mnist/