Bio-inspired Analysis of Semeion Handwritten Digit Data

Songzeng Fan

Research School of Computer Science, Australian National University

u6013724@anu.edu.au

Abstract.

This paper constructs a convolutional neural network to classify semeion handwritten digit where the data set is from UCI. In order to train neural network to achieve desired results, I use back propagation algorithm here to adjust the parameter of each learnable filter. I also used bimodal distribution removal to drop noises in the data set. After constructing the neural network, I use cross entropy to evaluate the performance. After training, the testing accuracy of classification is around 95%. Comparisons with several other neural network performances which also use this data set, showing that my neural network has comparable performance in terms of classification.

Keywords: Convolutional Neural Network, Semeion Handwritten Digit Data, Bimodal Distribution Removal, Back Propagation Algorithm, Cross Entropy

1 Introduction

Making the machine identify the handwriting has become an increasingly important requirement. It is a significant way of human-computer interaction. A standard discriminative method to digit recognition is to train a classifier to output recognition classes based on the input image [2]. Current approaches to object recognition make essential use of machine learning methods [4]. Neural networks are a powerful technology for classification of visual inputs. The next important practice is that convolutional neural networks are better suited for visual document tasks than fully connected networks [6]. Therefore, in this paper, I choose convolutional neural network as my network architecture.

This semeion handwritten digit data set is created by Tactile Srl, Brescia, Italy and donated in 1994 to Semeion Research Center of Sciences of Communication, Rome, Italy for machine learning research. In this dataset, 1593 handwritten digits from around 80 persons were scanned [8]. Here we use bio-inspired method to build a neural network to identify these handwritten digits. After training the neuron network, we can get the simulation outcome to determine what is the number in the input image. The dataset is from UCI Machine Learning Repository and the data set called Semeion Handwritten Digit Data Set. I am trying to solve a classification problem and this data set is easy to process as a good input form for the neural networks. The reason why I choose this data set here is because recognizing the handwritten numbers has many applications today.

In this paper, I am constructing a feedforward convolutional neural network to classify input handwritten digit images. The prediction works by input image matrix to the neural network, use back propagation algorithm to adjust the parameter of each learnable filter. Then, I pick up some data from raw data to test training result. I use cross entropy to evaluate my neural network.

As Slade and Gedeon mentioned in *BIMODAL DISTRIBUTION REMOVAL*: it requires a large amount of data to train the neural network with low bias and low variance [7]. The data set used in this paper also has low variance. In this case, the accuracy of neural network training result can hardly be further improved. In order to avoid overfitting, I also cannot train a lot of times to improve accuracy rate. In this paper, I use bimodal distribution removal which is introduced by Slade and Gedeon to detect outlier. This method addresses the shortcoming of other method and it can detect the outliers which are close to real data. I will explain this method in the next section with my personal understanding.

2 Method

Here, I separate my implementation into four parts: data pre-process and input, convolutional neural network implements, back propagation, bimodal distribution removal and evaluation method.

2.1 Data Pre-process and Input

Firstly, I analyze the data set. This data set consists of 1593 handwritten digits, stretched in a rectangular box 16x16 in a gray scale of 256 values. Then each pixel of each image was scaled into a Boolean (1/0) value using a fixed threshold.

There is no category for the data which means I don't need apply method such as one hot to convert data format, which guarantees the data of this data set will not become sparse. The whole dataset has 266 columns. Among them, 256 columns are the image data stored as the input, and the last 10 columns are the output. However, only one column in the last 10 columns has valid information. Which means, if it is "0" in this image, in the last 10 columns, only the first column has a value of "1", and the other columns are all 0. Therefore, I extract the information from the last 10 columns to the number 0-9, which easily represent the handwritten digit and machine-readable. According to the description of the dataset in UCI, I split the data into two parts: input and target. The first 256 attributes are regarded as the input of neural network and the last attribute is regarded as target which need to be predicted. As the convolutional neural network input, it needs to convert the 256 columns of the input into the input format of 1*16*16.

After processing the data set, the next step is to input data. Here I choose 50% of the data which is 796 instances as training set. In order to fit my neural network structure which will be described in detail in the next section, I modify 796* 1*16*16 number as a four-dimensional input structure. to input in the neural network. Every time the neural network train, it will get all data as input, which greatly reduces the time of training the neural network to the ideal state.

2.2 Neural Network Implements and Back Propagation

After pre-processing the data and determining the input, I start to construct the convolutional neural network. At the beginning, I confirm the number of input neuron, it should equal to the image input size which is equal to 1 * 16 * 16 here. As for the number of output neuron, because of ten types in target, there are ten output neurons. Then I need to determine how many convolutional layers the neural network has, and which structure will get higher test accuracy. In order to simplify the whole structure and reduce experimental variables, I always set stride to 1. For other parameters, padding is relative to kernel size, specifically, padding equals to kernel size minus 1 and divided by 2. To determine other parameter and number of layers, I do some experiments, and the results are shown in the following table_1. (the training times equal to 200. In channel fits input and set to 1, so it is not listed. Out channel represents the number of filters, for these input images, 16 filters is optimal. Optimizer here use Adam, pooling here is max-pooling)

The number of convolutional layers	Kernel size and padding	Training time (second)	Mean accuracy
1	(5, 2)	11.59	87.72%
2	(5, 2)	28.04	95.30%
2	(3, 1)	16.22	93.94%
2	(7, 3)	42.41	93.64%
2	(9, 4)	59.18	93.95%

Table_1: Comparison of some neural network structures

After comparing with the mean accuracy and training time of these neural network structures, I decided to choose 2 convolutional layers and the kernel size is 5 * 5 and padding set to 2. Here we use feedforward structure, so the structure of the neural network is shown in the following figure_1.

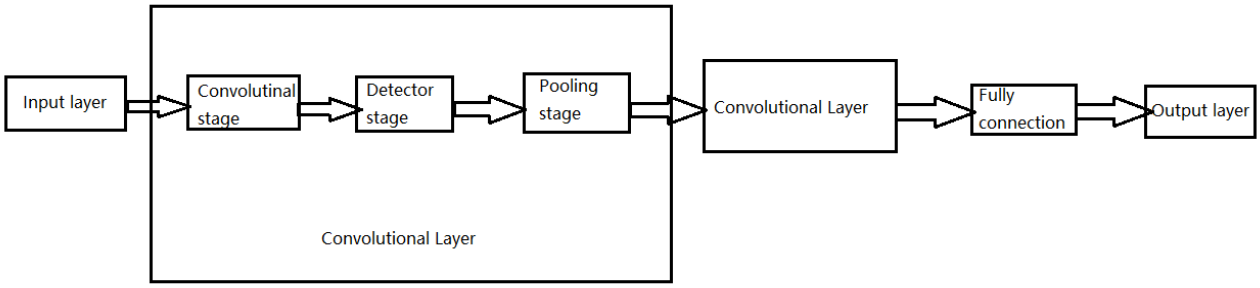

Figure_1: Convolutional Neural Network Structure

After determining the neural network structure, I use back propagation algorithm to backpropagate the error. For the first step, I perform forward pass. Then I use formula $Error_B = Output_B * (1-Output_B) * (Target_B – Output_B)$ to calculate output errors. I calculate hidden layer errors in the next step. Use $W_{AB} = W_{AB} + \eta * Output_A * Error_B$ to change the parameters in filter matrix. The last step is calculating the new output. I use loss.backward() to represent the above steps in the code. The activation function used here is ReLU which is easy to compute and effectively accelerate convergence velocity.

The optimizer I choose here is Adam. I compare Stochastic Gradient Descent (SGD), RMSprop and Adam in my experiment. RMSprop and Adam are all the improved forms of SGD and their mean accuracy are all good. When reduce the learning rate, all of them will not make the neural network overfitting in early. When I reduce the training times, SGD and RMSprop are not performing well in test set and they are also not stable (like figure_2 and figure_3). Although adjusting learning rate to low value and using Adam is a little bit slow in convergence, it is not influenced other noise. The stability of Adam contributes to our follow-up study.

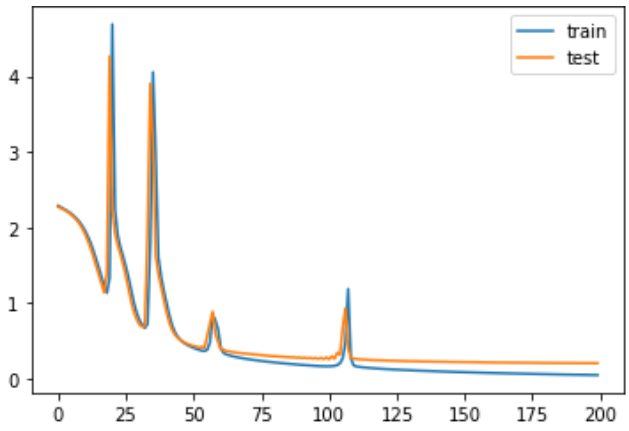

Figure_2: SGD Performance

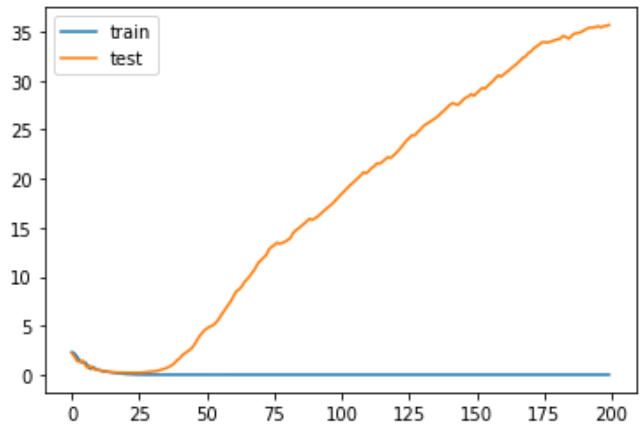

Figure_3: Rprop Performance

After constructing the neural network, I choose cross entropy as my loss function to evaluate my neural network. The formula is $E = -\sum_{j=1}^{T} y_i log_{p_j}$ [3]. I use cross entropy to calculate the loss the neural network. Cross entropy is used to measure the difference between ten probability distributions. The greater the loss value, the greater the difference between the ten distributions, the more unexpected the results of the experiment. Conversely, the smaller the loss value, the more similar the ten distributions, the more consistent with the expectation. Therefore, when the smaller the loss it gets, the better the effectiveness of my neural network training. Thus, the cross entropy can determine whether the training result is fit the expected target.

2.3 Bimodal Distribution Removal

There are several methods to improve the neural network, here I choose bimodal distribution removal (BDR) which is published by P. Slade and T.D. Gedeon in 1993. This method aims to clean up noisy training set from real world data set, it also provides a natural stopping standard to help to terminate training.

As Slade and Gedeon (1993) explained, firstly, I need to analyze the input data. Here I use loss function to evaluate the error of each pattern. The figure_4 is shown as below. The figure illustrates the distribution of the pattern errors as normal, but the variance is large. After several number of training, the variance become low (in the paper low variance means less than 0.1, there is a little bit different for my data set, I will explain later), the error situation is shown in the following figure. Then we calculate the mean error for all the pattern in the training set. According to the mean error, I can pick up patterns which error greater than mean error. For those patterns, I calculate the mean error δ_{ss} and standard deviation σ_{ss}. Finally, we use formula Error >= δ_{ss} + α* σ_{ss} (0 <= α<= 1). Here α is a user-defined parameter to decide how many patterns should be removed each time. Then repeat the above operations until the variance of training set is less than a constant (in the paper it is 0.01)

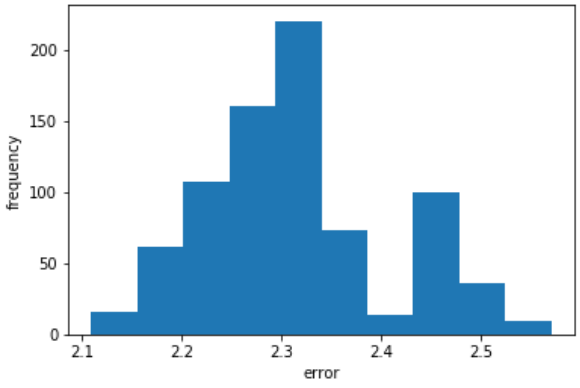

Figure_4: Error distribution at epoch 0

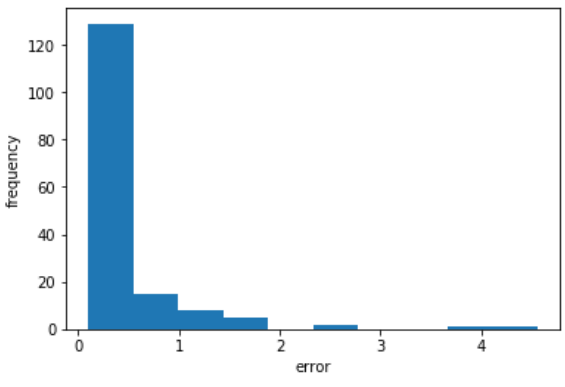

Figure_5: Remain patterns with relatively high error

For my data set, the variance of training set is always very low. In my opinion, that is because the difference between the two outcomes of data in my dataset is relatively large, which leads to a higher accuracy rate at the beginning. This makes the difference between each pattern's error is quite small. In order to solve this problem, I change the condition that BDR start. As the number of train times increases, at a time, the mean accuracy will change greatly. I start BDR at this time and end when the error of training set almost not changing.

2.4 Evaluation Measures

After determining the structure of convolutional neural network and training the neural network, we need to evaluate how the neural network works. The generalized confusion matrix is appropriate for both traditional classification algorithms and sub-pixel area estimation models [5]. The number of columns and rows should be the same and they all need to equal to the number of classification types, which equals to 10 for my dataset. The main diagonal of the confusion matrix represents the number of correctly differentiated quantities. For each column, except the element in the main diagonal, other elements in this column represent the amount of a class that was mistaken for this class. According to the number of the instance which has been correctly classified and misclassified, we can easily calculate the accuracy of classification. The specific example of confusion matrix will present in the next part.

3 Results and Discussion

The result here I use plot chart to perform the change of loss change according to training times. And the accuracy rate can be calculated by confusion matrix. Figure_7 illustrates the loss change without executing BDR. The orange line represents the loss of error set, if the orange line grows, it means the model has overfitting. Because when the neural network becomes overfitting, the error of test set will increase. From figure_7, we find the loss of training and testing decline gently, and the value of loss is quite small, which means the whole convolutional neural network is well trained. After applying BDR to my data set, the loss figure change like figure_8. It is found that BDR has not changed a lot, I think the reason may be next few points:

· The noises in my data set make little effect to the result.

· The optimizer I choose here is Adam, which is good at dealing with unstable target.

· The timing of the start and termination of BDR is not optimal, resulting in the noise being completely eliminated.

Figure_6 shows the confusion matrix for testing. Like the previous part explain, the main diagonal of the confusion matrix represents the number of outcome correctly classified. We can find my neural network can recognize all class well. And the total testing accuracy is 96.37%. Combining this example to explain the confusion matrix in detail. For example, for the first column, we can find there is one instance which should be "6", but it was misrecognized as "0" and an instance which should be "9" that was misrecognized as "0". And for the first row, one "0" instance was misrecognized as "5" and two "8" instances were misrecognized as "0". Other columns and rows are the same as described above.

Testing accuracy of my model is acceptable, the loss change curve is also consistent with expectation.

```
Testing Accuracy: 96.37 %

Confusion matrix for testing:

   90     0     0     0     0     1     0     0     2     0
    0    79     0     0     2     0     0     1     0     0
    0     0    72     0     1     0     1     0     0     0
    0     1     0    71     0     0     0     0     0     0
    0     3     0     0    76     0     1     0     0     0
    0     0     0     1     0    77     0     0     0     1
    1     0     0     0     0     0    80     0     0     0
    0     0     1     1     1     0     0    74     0     1
    0     0     1     0     0     0     1     0    75     1
    1     0     1     2     0     1     0     0     1    76
[torch.FloatTensor of size 10x10]
```

Figure_6: confusion matrix for testing

Figure_7: loss change without executing BDR

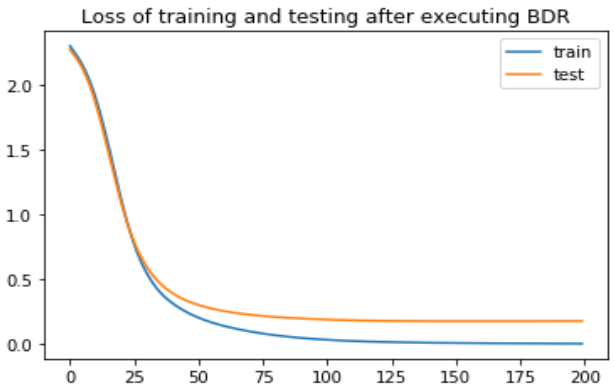

Figure_8: loss change after executing BDR

I choose *MetaNet: The Theory of Independent Judges* by Massimo Buscema at 1998 [1], which use the same data set with my paper. The mean accuracy of their model is exceeding 93.09%. The testing accuracy of my work is slightly higher, but the difference is not great I found a few reasons:

·The method that the paper use is normal artificial neural network. But this dataset refers to picture. Therefore, use deep learning method especially the convolutional neural network is much suitable to deal with this problem.

·The total amount of instance is too small to train a convolutional neural network to the expected state. The dataset which other convolutional neural network use has more than ten thousand even hundred thousand. Big dataset will be suitable for deep learning.

·The data in this data set has low variance, and it do not have a lot of noises. Therefore, if two models both are used some mainstream method, they will get some almost similar results.

·The different type in Buscema's paper have different classification accuracy rates. The difference between the best condition and the worst condition is about 10%. Thus it can be seen that normal neural network is not very stable in processing image input.

During the experiment, I find the variance of test set error increases firstly and decreases secondly. I think this is mainly because in the process of neural network training, the speed of loss decreases varies with the different pattern. Some loss of pattern decreased faster after training, while others were slower. Therefore, the loss gap between different pattern increases, which leads to the increase of the overall variance.

4 Conclusion and Future Work

I constructed a feedforward convolutional neural network to classify handwritten digit. I also used bimodal distribution removal to drop noises in the data set to improve my neural network. The neural network classification accuracy is about 96%.

There are number of direction for future work. The main limitation of my neural network model is not finding a good balance between accuracy rate and overfitting. Especially for high learning rate for some optimizers. For high learning rate, if I increase training epochs, the model will become overfitting and the accuracy rate of the test set will fall. But is I use low learning rate, the training time will increase.

Furthermore, I can study more deeply about how to apply BDR when the error of training set is always less than 0.1 or when the error is always greater than 0.1. I can also study about how to pre-process the data to make another optimizer such as RMSprop stable. I can also try another deep learning neural network architecture, for example, GoogLeNet or ResNet. In my view, combine convolutional neural network and normal neural network together may also help classification.

References

[1]: Buscema, M. (1998). Metanet*: The theory of independent judges. Substance use & misuse, 33(2), 439-461.

[2]: Hinton G E, Dayan P, Revow M. Modeling the manifolds of images of handwritten digits[J]. IEEE transactions on Neural Networks, 1997, 8(1): 65-74.

[3]: Janocha, K., & Czarnecki, W. M. (2017). On loss functions for deep neural networks in classification. arXiv preprint arXiv:1702.05659.

[4]: Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[5]: Lewis, H. G., & Brown, M. (2001). A generalized confusion matrix for assessing area estimates from remotely sensed data. International Journal of Remote Sensing, 22(16), 3223-3235.

[6]: Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003, August). Best practices for convolutional neural networks applied to visual document analysis. In ICDAR (Vol. 3, pp. 958-962).

[7]: Slade, P., & Gedeon, T. D. (1993, June). Bimodal distribution removal. In International Workshop on Artificial Neural Networks (pp. 249-254). Springer, Berlin, Heidelberg.

[8]: Tactile Srl, Brescia, Italy (1994). Semeion Handwritten Digit Data Set. Semeion Research Center of Sciences of Communication, Rome, Italy