Neural Network Reduction in Classification of Handwritten Digits and Application of Deep Learning Using Convolutional Neural Network

Guangnan Zhu [†]

[†]Research School of Computer Science, Australian National University

May 30, 2018

Abstract

Handwritten Recognition is attractive and necessary in the real world. Some research adopt normal classifier to classify the handwritten but the result is not reliable. In this report, we are doing research on recognition and prediction in handwritten digits, using Convolutional Neural Network (CNN) as deep learning network and adopting network reduction method to improve the classification and prediction performance. We use a MINST dataset for training and testing. The techniques help us get the result at 98% accuracy. Also, in this report, we introduce the the methods of reduction which is on the basis of distinctiveness and do analysis and comparison on optimizers to select the most suitable optimizer. And we also discuss the comparison of Cosine Similarity and Euclidean Distance when compute the distinctiveness.

1 Introduction

Handwritten Recognition is an attractive and basic issue in the area of *Image Recognition*. LeCun and Bottou use *Linear Classifier* and achieve 7.6% error rate and 3.5% error rate by using *Non-Linear Classifier*. ([LBBH98]). But for an image dataset, a normal classifier is not suitable for classification because the features of images are quite much and each pixel of the image cannot be simply classify to one or two classes. And a normal classifier is hard to extract the features and hidden features from the image.

Therefore, a neural network is more suitable in the work of recognizing the images. And *Convolutional Neural Network* (*CNN*) is always considered as one of the best method to dispose on images. So in this study, we use *CNN* to build up a deep learning network to do recognition (classification) and prediction on handwritten digits. And we use *MINST* as the dataset which is always used in handwritten digits research.

In addition, determining the exact number of hidden units is always a hard problem in Neural Network. In this study, we adopt the network reduction according to *T.D. Gedeon*'s paper ([TG91]) and do reduction on *CNN* on the basis of the distinctiveness of hidden units to remove some unnecessary hidden units to reduce the time and space complexity of CNN and improve the prediction performance.

To help improving the performance of prediction of *CNN* and *Network Reduction*, in this study, we also compare and analyze the optimizers and choose the most suitable one to apply for the *CNN* model. And we also discuss the difference of Cosine Similarity and Euclidean Distance and which one we should choose to compute distinctiveness.

The outline of the report is listed below. Section 2 declares the problem statement of the report. Section 3 introduce the dataset *MINST* we use. Section 4 introduces *CNN* model and introduces some methods to build up the *CNN* model and introduce how to select the best optimizers. Section 5 will introduce the reduction methods including how to remove units on the basis of distinctiveness and the advantages of reduction methods. Section 6 will show and evaluate the results we achieve and Section 8 and Section 9 list the conclusion and future work of this report.

2 Problem statement

Given a number of information (every pixels) of handwritten digits, after training, our prediction model can classify and the handwritten digits and can predict the handwritten digits according to the information.

Input is every pixels of the picture of handwritten digits, while output is the predicted handwritten digits.

3 Dataset

In this study, the dataset we choose is MINST (Modified National Institute of Standards and Technology database). *MINST* is a large database of handwritten digits that is commonly used for training various image processing systems ([LCB09]). The Figure 1 shows a sample of the handwritten digits. The database is also widely used for training and testing in the field of machine

Figure 1: MINST sample.

learning. This dataset contains 70,000 28*28 images of handwritten digits and each features of the dataset is one of the pixels of the picture. Each *MNIST* digit is labeled with the 10 correct digit class (0, 1, ... 9). The reason why we choose this dataset is:

1. This dataset is a image dataset where every instances is represented by 320 pixels. And there are 10 classes (labels).
2. The total number of instances is 70,000 and we have about 60,000 instances as training set and about 10,000 as testing set.
3. *Convolutional Neural Network* is always considered as a deep learning network for composing on images.

4 Convolutional Neural Network

In machine learning, a *Convolutional Neural Network* (*CNN*) is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery. A *CNN* consists of an input and an output layer, as well as multiple hidden layers. The hidden layers of a *CNN* typically consist of convolution layers, pooling layers, fully connected layers and normalization layers. The structure of *CNN* is shown in Figure 2. In this study, we first build input layer, output layer and two convolutional layers. Second, during each training pass, randomly remove a fraction of neural connections and adds noise to hidden units. And in this study, the activation function is *F.relu()*. Using this *CNN*, we set 320 neurons which represent features of dataset in the input layer and set 10 neurons which represent classes (labels) in the output layer.

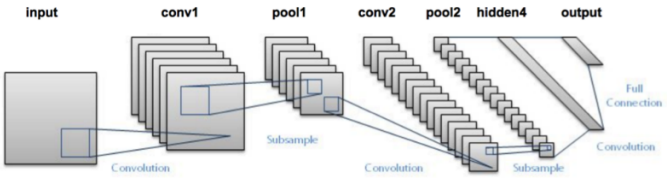

Figure 2: Structure of CNN

And normally, we use the rules put forward by Heaton ([Hea08]) to decide the number of neurons in hidden layer. The optimal number of hidden neurons is in the range between 4 to half of the number of neurons in input layer. And in this study, because we want to adopt network reduction technique, we set the number of hidden neurons related to half of the number of input layers but a little bit larger. We will explain this in the Section 6.1

4.1 Best Optimizer

In this report, we want to compare the optimizers. Different optimizer apply to different cases. Gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. At the same time, every state-of-the-art Deep Learning library contains implementations of various algorithms to optimize gradient descent. These algorithms, however, are often used as black-box optimizers, as practical explanations of their strengths and weaknesses are hard to come by. So in the study, we will compare a set of different optimizers (*SGD, Adagrad, RMSprop and Rprop*) by comparing the predicted result ([KB14]).

5 Reduction Algorithm

In this study, we adopt distinctiveness as the reduction technique. In the first step, we build up a normal *Convolutional Neural Network (CNN)* according to Section 4. And we divide the input as batches to improve the efficiency. Secondly, we train the CNN model using the training set of dataset (Section 3). Then we check the distinctiveness between hidden units and remove the "useless" units. Then we retrain the *CNN* model by using the weight of the original model where we can improve the training efficiency. The procedures are shown in the Figure 6.1.

5.1 Distinctiveness

In this report, we adopt the method for reduction is distinctiveness. The distinctiveness of hidden units is determined from the unit output activation vector over the pattern presentation set. For each hidden unit, we construct a vector of the same dimensionality as the number of patterns in the training set, whose size is $n*m$, each component of the vector corresponding to the output activation of the unit, which is $fc1$ of the net in the code. This vector represents the functionality of the hidden units in input space. And in this model, we remove one or groups of units which are regarded as “useless”. And then we will retrain the network for recovering from deleting.

5.1.1 Remove Useless Vector

In the study, the key technique is pruning the vector. There are some advantages:

Time Complexity The basic idea of reduction is removing the “useless” vector. We set N is the number of the neuron in hidden layer, M is the number of inputs patterns. Back to Neural Network, for a Convolution Neural Network: If the number of neurons of each layer is $n1$, $n2$, $n3$, here $n1$ is M For one epoch, the time complexity should be

$$O(n1 * n2 + n2 * n3).$$

Because the number of input has been set, so $n1$ and $n3$ can be regarded as constant, so the time complexity should be

$$O(N),$$

when N is $n2$. From this, we can clearly know that the time complexity of Neural Network is based on the number of neuron in hidden layer. So removing the "useless" neurons can reduce the running time of Neural Network and improve the efficiency.

Space Complexity Here we set M is the number of inputs, N is the number of hidden neurons, C is the number of output as well as the number of class in target. For space complexity, in the BP network, for hidden layer, the size of the matrix is [M * N] and for output layer is [N * C]. In this case, assume the M and C is constant, So the Space Complexity is still based on N. So removing the "useless" neurons can reduce the space of Neural Network and improve the efficiency.

Better Fitting *T.D. Gedeon* declare that reduction function can shorten the time and reduce the space ([TG91]). In this study, we also discuss another advantages of reduction which is that reduction can solve the over-fitting problem. Figure 3 shows a simple model of Neuron Network. And the output of the Network is

$$y = f(w_1 * x_1 + w_2 * x_2 + \cdots + w_n * x_n),$$

For classification problem, briefly, we want to seek for a line to divide the input into the classes. The output will perform well when the line is divide the input into classes but not too specific, which means that we don't want the line is matching the input. So in the formula, if we add a variable x and it has weight w, then the formula will start to get close to the data and that will be performed badly in test data. For the Neural Network, if we use too much neurons which means that we add too much variable w to the formula, then the over-fitting problem (Figure 4) will occur. So if we use reduction method, we can prevent the over-fitting problem by removing the w from the formula. So in this case, reduction technique can also improve the performance of the Neural Network in a way.

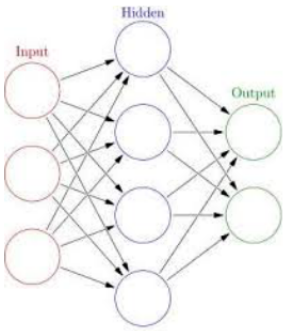

Figure 3: Structure of One Hidden Layer Network

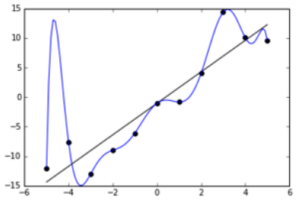

Figure 4: Overfitting Problem

In this study, according to *T.D. Gedeon*'s paper ([TG91]), for distinctiveness problem, useless hidden units are the neurons which is too relevant to each others because the units perform no real function in the final product and may be supportive of significant units. In this model, we want to remove the group of vectors, which is removing the hidden units. The vectors for clone

units would be identical irrespective of the relative magnitudes of their outputs. We introduce the method for calculate the similarity of the vectors.

5.2 Cosine Similarity

Cosine Similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them ([NB10]). And in the interval[0, 2/pie), if the calculating value is bigger, then the two vectors are more relevant. The formula to calculate cosine similarity is

$$cos(\boldsymbol{x}, \boldsymbol{y}) = \frac{\boldsymbol{x} \cdot \boldsymbol{y}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{y}||}.$$

In the study, we convert the cosine value to degree. And to normalize vector angle calculations we extend angular.

Extend Angular Moved the origin point to (0.5,0.5) is a method to extend the angular in [0, pie) to [0, 2pie) and the angle will be easier to judge (Figure 5). So moving the origin point to (0.5, 0.5) is a method to help us check the similarity of the vectors. If obtained angle is smaller than 15°, the two neurons are similar and we need to remove one of them. And we need to add the weight vector of the removed unit to the one remain. If obtained angle is larger than 165°, the two neurons have complementary effects, then both of them need to be removed.

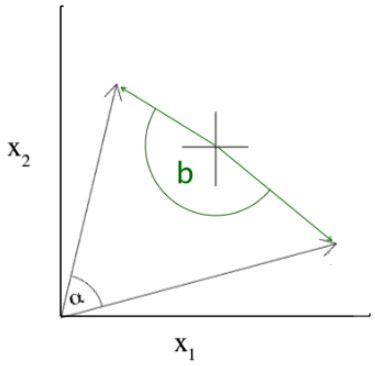

Figure 5: Normalization to Extend Angular

5.3 Euclidean Distance

In mathematics, the Euclidean distance or Euclidean metric is the "ordinary" straight-line distance between two points in Euclidean space. In general, for an n-dimensional space. The distance is

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}.$$

Because the length of the vector is the same, so we can use euclidean distance to calculate how far is between the two vectors in a n-dimensional space(assume n is the length of the vector). If the distance is large, then the two vectors are more irrelevant. If small, then more relevant.

In essence, Cosine similarity is like (squared) Euclidean distance after scaling each vector to unit length. First, according to the formula of Cosine Similarity and Euclidean Distance, we need to normalize the points before using Euclidean yields:

$$Euclidean(\frac{A}{||A||}, \frac{B}{||B||})^2 = 2[1 - CosSim(A, B)],$$

Then we calculate the Euclidean Distance and we can get a value which is similar with Cosine Similarity. Then if the value is bigger than cos(15°) and the direction of two vectors of units are

not opposite, the two units are useful and we need to preserve both of them. Otherwise, we need to remove one of these two units according to the idea of reduction.

Comparison between Cosine Similarity and Euclidean Distance In this case, Cosine Similarity will perform better than Euclidean Distance. The graph shows that Euclidean distance is the absolute distance between the points in the space and is directly based on the locations of the points. Cosine Similarity is calculating the angle of the vector and it shows the difference on the direction instead of location. So for example, assume two points (0, 1) and (1, 0) in the two dimensional space, the Euclidean distance is small but the angle is big. That is to say, euclidean distance shows the difference between every features of the objects while cosine similarity is on the directions. In this case for seeking for the relevant vector in hidden units, we want to classify the input into classes. Instead of seeking for the cluster center, we are using the formula (Figure 6) which can be regarded as a line in the space. Every variable influence the direction of the line and adjust the line directions to classify the inputs data. So in this case, we focus on the direction which means that in the example (0,1) and (1,0), these two vectors are different. Therefore, in our study, we use Cosine Similarity to calculate the relevance of the vectors. ([hm])

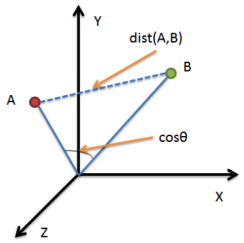

Figure 6: Comparison on Cosine Similarity and Euclidean Distance.

6 Evaluation

6.1 Experiment Setup

In this study, we set 320 neurons in input layer represent the features of dataset and 10 neurons in output layer represent the classes. And as we discuss in Section 4, we set the number of neurons is 165.

Also we set batch size as the default value which is 64 and the test batch size is 1000. It may take more time to compute but there will be less noisy.

We set learning rate as 0.01 and the number of epochs as 10. But in the experiment, we set the epoch is 1 because when epoch is 1, it is easier to find the difference between the 4 optimizers and it is more obvious to find whether reduction method is efficient. In some case, if we set the epoch as the default value which is 10, because of the good training, there is no removed unit in the network. So in this study, the results for comparing the optimizers and reduction methods are shown below when epoch is 1. But the final result is shown when epoch is 10.

And in the study, we will compare a set of the optimizer (*SGD, Adagrad, RMSprop and Rprop*). We train the network with different optimizers and do comparison and analysis.

And we also compare the cosine similarity and euclidean distance. Although we have discuss that Cosine Simialarity is much suitable in solving our problem which is a direction problem in the Subsection 5.3, we still want to compare the performance on this two methods for calculating the distinctiveness.

7 Results

7.1 Best Optimizer Result

The Table 1 shows the average results (test time = 3) of different optimizers that we discuss in Section 4. We use the set of optimizers (*SGD, Adagrad, RMSprop and Rprop*) and demonstrate

the differences of performances. As Table 1 shown, we can find that *Adagrad* performs best and the result is 98% accuracy while *RMSprop* performs worst at 11% accuracy after reduction. SGD performs a little worse than *Adagrad* which is 96% accuracy and *Rprop* is 88% accuracy. And

Table 1: Comparison Of Optimizers

	Testing Accuracy of No Reduction	Testing Accuracy of Reduction	Average Loss of No Reduction	Average Loss of Reduction	No. of Removed Units
Adagrad	97%	98%	0.0942	0.0702	2
SGD	95%	97%	0.1835	0.1105	7
RMSprop	11%	11%	2.3015	2.3092	82
Rprop	88%	88%	0.9867	0.9506	3

the Table 1 also shows that although *RMSprop* has removed 82 hidden neurons, the result didn't improve after reduction and the average loss is high. So we can declare that *RMSprop* is not suitable in this case.

Compare with *SGD*, *Adagrad* and *Rprop*, *SGD* and *Adagrad* all perform better than *Rprop* in not only the result at accuracy but also in testing average loss and reduction efficiency. After using reduction, *Adagrad* and SGD remove some hidden units and the results increase while the result of *Rprop* keep balanced. And the result of *Adagrad* is up to 98% and the average loss is only 0.0702 after removing two hidden units which is better than *SGD* (Accuracy: 97%; Testing Average Loss: 0.1105; Removed Units: 7). So we can declare that *Adagrad* is the best choice in this case.

7.2 Reduction Algorithm Result

As the Table 2 shown, using *Adagrad* as optimizer, the result increase from 97% to 98% and there are two hidden units are removed. And the testing average loss decrease from 0.0942 to 0.0702 in the test set.

Table 2: Comparison Of Reduction and No Reduction

Adagrad	Testing Accuracy of No Reduction	Testing Accuracy of Reduction	Average Loss of No Reduction	Average Loss of Reduction	No. of Removed Units
First Test	97%	98%	0.0941	0.0702	2
Second Test	97%	98%	0.0943	0.0700	2
Third Test	97%	98%	0.0942	0.0704	2

And the loss is plotted in the Figure 7. We can find that the loss decreases from over 2.5 to lower than 0.5. And the loss decrease obviously in the beginning of training and keep stable while the epoch is large. There is no plateau in curve. So the training is reliable.

Table 3 shows the results of using Cosine Similarity and Euclidean Distance. In the table, we can see that Cosine Similarity is a little better than Euclidean Distance. But as we discuss in Subsection 6, Cosine Similarity is suitable for disposing the direction problem like the similarity of two vectors while Euclidean Distance is more suitable in calculating the distance like clustering algorithm.

Table 3: Comparison of Cosine Similarity and Euclidean Distance

	Fist Test	Second Test	Third Test
Cosine Similarity	98.0%	98.2%	97.9%
Euclidean distance	97.8%	98.0%	97.6%

Table 4 shows the different testing running time of no reduction network and reduction network. The table shows clearly that with reduction technique, the network spend less time on

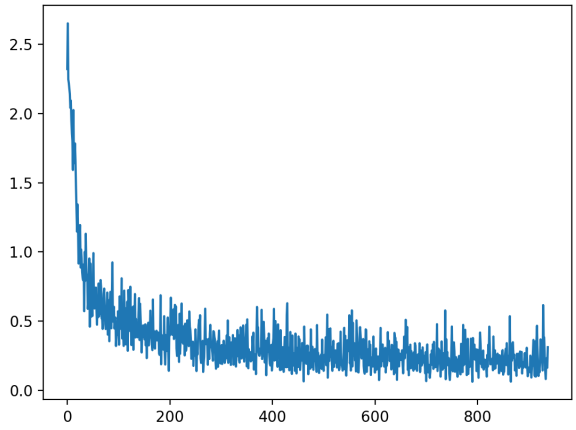

Figure 7: Loss Trends

testing. So the reduction technique can reduce the testing running time significantly as we discuss in Section 5.1.1.

Table 4: Average Time Comparison of Reduction and No Reduction for each Epoch

Testing Time (Seconds)	First Test	Second Test	Third Test
Reduction	1.6583	1.6678	1.6723
No Reduction	1.6164	1.6204	1.6160

And Table 5 shows the result when we set epoch to 10 and we also use *Adagrad* and reduction techniques. As the table shown, the average accuracy is up to 99.1% and the maximum result is 99.3% while the minimum is 98.9%. And the average number of removed units is 0.4 where can be considered as that there are no units need to be removed because the original *CNN* is good enough.

Table 5: Result For Final Test

	First Test	Second Test	Third Test	Forth Test	Fifth Test	Max	Min	Ave
Accuracy	98.9	99.2	99.1	99.3	98.9	99.3	98.9	99.1
No. of Removed Units	1	0	0	0	1	1	0	0.4 $\cong$ 0

From all results above, we can say that reduction technique can help us improve the network by not only reducing the running time and space, but also improve the performance of prediction at accuracy.

7.3 Comparison with a published paper

In this study, we compare the results with LeCun and Botton ([LBBH98]). LeCun and Botton adopt Linear and Non Linear Classifier on MINST data. In their study, each input pixel value contributes to a weighted sum for each output unit and the output unit with the highest sum (including the contribution of a bias constant) indicates the class of the input character. For Linear Classifier, they improve the performance to where the error rate is 7.6% by training each unit of a single-layer network to separate each class from each other class. For Non Linear Classifier, they use baseline Nearest Neighbor Classifier with Euclidean distance measure between input images. They declared that the classifier has the advantage that no training time, and no brain on the

part of the designer, are required. But they also pointed out that the memory requirement and recognition time are large. And the error rate of non-linear classifier is 2.4%.

So compare to their work, this study improve the performance to a lower error rate (0.7% to 1 %). And the result is more stable while the error rate of Linear Classifier is from 7.6% to 12% and Non Linear Classifier is from 2.4% to 5.0%.

8 Conclusion

In this paper, we showed the work on build up the *Convolutional Neural Network (CNN)*. We use *CNN* to work on MINST dataset to dispose and do the prediction on handwritten digits on the basis of pictures of different handwritten digits. As the result shown, the *CNN* model can do a reliable prediction on handwritten digits. And it also proves that *CNN* is useful in composing pictures.

Besides, we have done network reduction on the *CNN* to reduce the time and space complexity and improve the performance of prediction. We adopt network reduction technique on the basis of distinctiveness declared by *T.D. Gedeon* ([TG91]).

With *CNN* and network reduction, we get a result at 98% accuracy in predicting handwritten digits which is better than the one without reduction at 97%. And in the study, we also compare the optimizers and the result show that *Adagrad* is the best choice in this case.

Besides, by comparing with another paper ([LBBH98]) which is also working on prediction in handwritten digits using *MINST* dataset, our result is better in prediction and our result is more stable.

So in this study, our work can classify (recognize) and predict the handwritten digits significantly using *CNN* and network reduction techniques and our work also provide a analyze and comparison on different optimizers. And our work also compare the Cosine Similarity and Euclidean Distance in computing distinctiveness which is the measure of network reduction.

9 Future Work

In the study, we still need to set some parameters like learning rate, epochs and so on. And we need to choose the optimizer by testing the results which may waste time. So we need to focus on a non-parameter *Convolutional Neural Network (CNN)* model and provide a more reliable optimizer. And Qiao, who works on Voice Recognition, declares that although the result at 98% is high but it may occur a big problem in the 2% bad prediction which means that the report is not complete robust ([Qia18]). So a more accurate prediction model should be our future work.

References

[Hea08] Jeff Heaton. *Introduction to neural networks with Java*. Heaton Research, Inc., 2008.

[hm] Anony-Mousse (https://stats.stackexchange.com/users/7828/anony mousse). distance measure of angles between two vectors, taking magnitude into account. Cross Validated. URL:https://stats.stackexchange.com/q/71703 (version: 2013-10-02).

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LCB09] Yann Lecun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 2009. *URL http://yann. lecun. com/exdb/mnist*, 2009.

[NB10] Hieu V Nguyen and Li Bai. Cosine similarity metric learning for face verification. In *Asian conference on computer vision*, pages 709–720. Springer, 2010.

[Qia18] Mingda Qiao. Do outliers ruin collaboration? *arXiv preprint arXiv:1805.04720*, 2018.

[TG91] D. Harris T.D. Gedeon. Network reduction techniques, proceedings international conference on neural networks methodologies and applications. 1:119–126, 1991.