Convolutional Neuron Network with Bimodal Distribution Removal

Xinlu Dong
Research School of Computer Science, Australian National University
u6341945@anu.edu.au

Abstract. Convolutional neural network (CNN) is widely used in image classification. However, in classification of real world image data, noisy is inevitable and including outliers into the training procedure will produce bad results. Thus, outlier detection and deletion are crucial for improving the performance of CNN. Bimodal Distribution Removal (BDR) is an algorithm which can recognize and clean the noisy of the data set and hence improve the classification result. Therefore, in this paper, I build a CNN as a classifier for recognizing the handwritten digits images in MNIST dataset and I use BDR algorithm in the training procedure for outlier cleaning. Compared with the CNN without BDR, the algorithm can improve the average accuracy of the classifier slightly and reduce the training time. Though in terms of accuracy it doesn't perform as stable as a k-Nearest Neighbor (k-NN) classifier proposed by published research paper for the same dataset, it can save more recognition time than the k-NN classifier.

Keywords: Convolutional Neural Network, MNIST, Bimodal Distribution Removal, outlier detection, image classification, handwritten recognition

1 Introduction

Analysis of outliers is a process of checking whether dataset has input errors and contains unreasonable data. It is very dangerous to ignore the existence of outliers because a classifier as a non-parameter estimator may be very sensitive to the noise in the training data. The influence, the emphasis on the occurrence of outliers, and the analysis of their causes often become opportunities for finding problems and improving decision making. Thus, outlier discovery and removal is significant for improving the neural network training result.

Here, I intend to build a classifier to solve the problem of handwritten image recognition and explore how a published well-performed outlier detection method called Bimodal Distribution Removal [1] (BDR) influences the result of the classifier. The main idea of BDR is that the patterns corresponding to the most frequent and second highest values of the loss are identified as outliers and hence are removed from the training set. Firstly, I implement a feed-forward neuron network as classifier for recognizing the handwritten letters, which is a foundation of handwritten image recognition, on the Letter Recognition dataset [2] and I utilize BDR algorithm for outlier detection and deletion during training procedure to see whether it helps with improving the performance of the neuron network classifier. The result shows that BDR can enhance the accuracy of the neuron network by about 1.5%.

Now, I try to build a classifier for handwritten digits recognition, which is another basis for the area of handwritten image recognition. But this time I require the input is real image instead of extracted features like those in the Letter Recognition dataset. Therefore, MNIST dataset [3] is chosen here. It is a database of handwritten digits images, containing 60,000 training set examples and 10,000 testing set examples, big enough for training and testing. All of the examples are labeled so that it can provide correct target in training procedure and guarantee the persuasiveness of calculating accuracy on testing sets. Additionally, the picture pixel in this dataset is small, 28*28 pixels, which means the computational load is relatively low, and a neural network with a small number of layers can also be competent.

As for choosing the classifier, a drawback of feedforward neuron network classifier is that we need to extract features manually for it, however, feature selecting is really difficult, especially for various images – if the number of features is too small, the accuracy of the classifier will be low, which we call under-fitting; if the number of features is too large, it may pay too much attention to a certain feature which leads to misclassification, namely over-fitting, so I choose convolutional neuron network (CNN) replacing normal neuron network. In machine learning, a convolutional neuron network is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery [4]. Making use of convolutional layers instead of densely connected neurons, it avoids complex pre-processing of the image (i.e. extracting features artificially) and receives original image as input directly. For outlier cleaning, I still use BDR algorithm during the CNN training process.

Thus, the aim of this paper is to build a CNN classifier with BDR algorithm to recognize which number is on the images in MNIST dataset and compare the result with the result of the original CNN without BDR to see whether the algorithm helps with optimizing the CNN performance. Additionally, the result is also compared with the result of a published research paper "Fast k-Nearest Neighbor Classification Using Cluster-Based Trees" [5] on the same dataset. This algorithm focuses on speeding up k-NN search to reduce the classifying time.

Furthermore, there are also a lot of published researches proposing various classification methods on MNIST dataset. Linear Classifiers [3] can reach accuracy of 92.4%; 2-layer Neuron network with 800 hidden unites and Cross-Entropy Loss [6] has 98.4% correct recognition; and Convolutional net LeNet-4 with K-NN instead of last layer [3] manages to get accuracy of 98.9%.

The methods of the implement CNN and BDR are detailed in Section 2. In Section 3, I will report and analyze the experimental results and make comparison as mentioned above. Finally, a conclusion and the potential future work are given in Section 4.

2 Method

In this section, I will introduce how the CNN is built and how BDR is implemented. The CNN classifier takes the handwritten digits in MNIST dataset as input and outputs the number with the largest similarity to the original image. The network is trained with back-propagation for updating the weights and BDR algorithm for detecting and removing outliers. The result will be evaluated in terms of accuracy and classifying time on testing set and the time consumed by the training procedure.

2.1 Split the training and testing set

The training set and testing set of MNIST consist of handwritten digits from 250 different people, of which 50% are high school students and 50% are American Census Bureau employees. The black and white handwritten images are normalized to fit into a 28*28 pixel bounding box and anti-aliased, which introduced grayscale levels [7]. That is, each picture is composed of 28*28 pixels and each pixel is represented by a grayscale value. Since MNIST dataset has already divided the data into training sets (60,000 examples) and testing sets (10,000 examples) and also split each pattern into images (inputs) and labels (targets), the training and testing set I use here is the same as what the dataset defines.

2.2 Define the convolutional neuron network

The CNN takes a 28*28 pixel, greyscale, input image from the MNIST dataset, fed through several layers one by one and finally gives an output vector, which contains the log probability that the input was one of the number 0 to 9. Figure 1 [8] shows what the CNN I build here looks like.

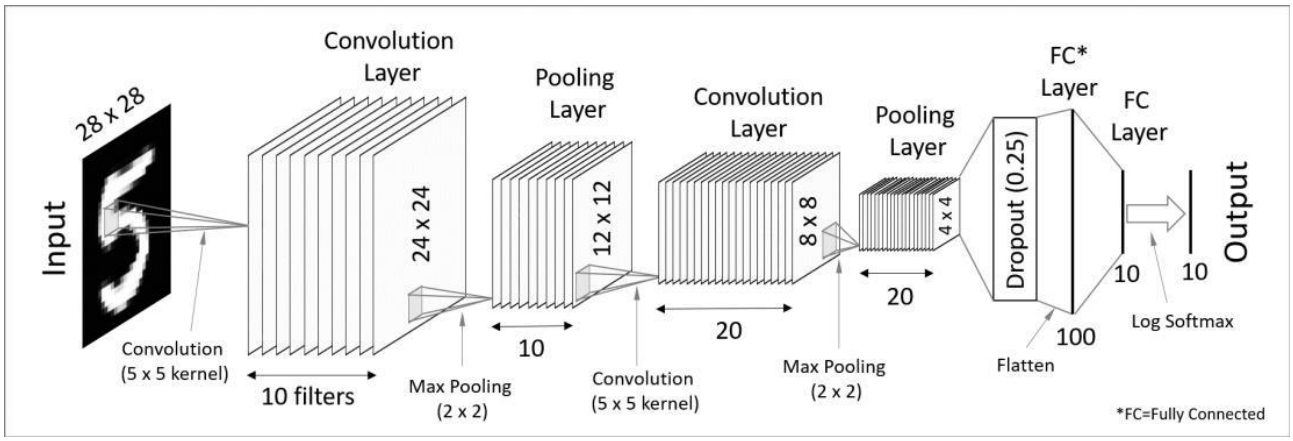

Fig. 1. The structure of the CNN built in this paper.

The CNN contains two convolution layers, two pooling layers, one dropout layer and two fully connected layers. With 10 filters and a 5*5 kernel, the first convolution layer can extract 10 sets of different features (called feature maps) from the original image. Then the 10 feature maps are passed to a max-pooling layer over a 2*2 window. Its main function is to provide strong robustness (for example, max-pooling is to take the maximum value in a small area, if other values in this area slightly change, or the image is slightly shifted, the result after pooling remains unchanged), and reduce the number of parameters and hence prevent the occurrence of over-fitting. The output of the pooling layer is passed to Relu activation function. The definition of Relu function is:

$$f(x) = max(0,x) \quad (1)$$

If the input is less than 0, it will output 0; otherwise, it will output the same as the input. The superiority of Relu function is that it is very easy to calculate which will speed up the classification. It can also reduce the likelihood of gradient vanishing since when the input is positive, the derivative of this function is always 1. Then another convolution and max-pooling operation are implemented similarly, but this time the number of feature maps increases to 20 to explore more features and make the classification more accurate. After these, I use dropout to avoid over-fitting, that is, during each training pass, randomly remove 25% of neural connections. This operation forces a neuron to work with

other randomly selected neurons to weaken the joint adaptability between neurons and enhance the generalization ability. Finally, it comes to the fully connected layers. As the last pooling layer has 4*4*20 neurons which means it produces 320 outputs in total and dropout layer does not modify the number of neurons (it just removes a fraction of neural connections tentatively), the number of input neurons of the first fully connected layers should be 320 and I set the number of output neurons to 50. Then the second fully connected layer receives these 50 outputs. Since the expected output of the whole network is the probability that the input was one of the number 0 to 9, it should have 10 output neurons (the indexes are from 0 to 9 and the value is the probability to be that number) and use SoftMax function as activation function:

$$S(f_{y_i}) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \tag{2}$$

where we have:

- f is a vector of the inputs to the output layer (if we have 10 output units, then there are 10 elements in f)
- i, j indexes the output units (so $j = 1,2,3 \ldots$ *the number of outputs*)
- y_i is the scores computed from the forward propagation of the network

The output of it is equivalent to a categorical probability distribution, it tells us the probability that any of the classes are true. This property of SoftMax function just meets the expectancy. Figure 2 [9] gives an example of how the SoftMax function works.

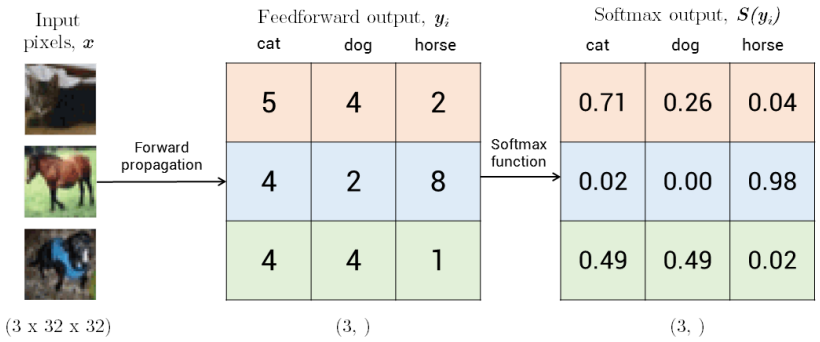

Fig. 2. An example of the SoftMax function: we feed the network three images of cat, horse and dog. The network assigns the first image a confidence of 0.71 that it is a cat, 0.26 that it is a dog, and 0.04 that it is a horse. The same goes for the other samples above.

2.3 Train the CNN classifier

Optimizer. When updating the weights there are several options can be used. I choose the simplest (and rather effective) rule called Stochastic Gradient Descent (SGD). It is calculated as follows:

$$weight = weight - learning\_rate * gradient \tag{3}$$

When using SGD, we can set the learning rate (how drastically the weights should be updated) as well as other parameters, such as momentum (to what extent keep the original update direction). I also try to use Adam and RMSprop with same learning rate and momentum (if it is required) as optimizer to train the same training set and test on the same testing set ten times respectively. Computing the average training time and average accuracy on testing set, Adam reaches about 97% accuracy requiring 688,848 milliseconds for training and RMSprop only has 10.5% accuracy and needs 471.48 milliseconds, whereas SGD gets 97.2% accuracy requiring 586.908 milliseconds. This indicate that SGD performs the best in comprehensive consideration of time consuming and accuracy.

Loss function. CrossEntropy loss and related Negative Log Likelihood (NLL) are suitable for classification problem. Due to the following reason, I choose NLL as the loss function for the classifier. Since the SoftMax activation function used in the CNN model can be interpreted as the probability of a certain input belonging to one of the output classes, and this probability is between 0 and 1, when taking the log of that value, we find that the value increases (and is negative), which is the opposite of what we want, so we simply negate the answer, hence the NLL. The formula is following:

$$L(\mathbf{y}) = -\log(\mathbf{y}) \tag{4}$$

where y is what we obtain from the SoftMax function. Figure 3 [9] gives an example of how NLL works.

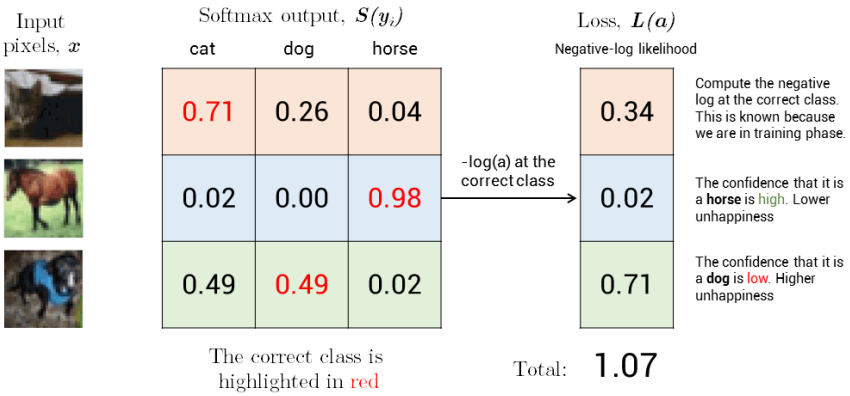

Fig. 3. An example of NLL function: when computing the loss, we can then see that higher confidence at the correct class leads to lower loss and vice-versa.

The whole training procedure can be described as: Firstly, feed the CNN model with the data and get the forward-propagation output of the CNN. Using the output, target and the loss function, compute the loss value. Next, calculate the gradients by backward pass of the network. Then with the gradients, the optimizer can update the weights of the network.

2.4 Implement Bimodal Distribution Removal (BDR)

The main idea of BDR is that the patterns corresponding to the most frequent and second highest values of the loss are identified as outliers, and these patterns are removed from the training set. In the early training epoch, the loss value of the patterns is almost evenly distributed. With the advancement of the learning process, the accuracy of neural networks is getting higher and higher. In addition to certain loss values, the frequency of occurrence of other loss values is greatly reduced. At this time, the distribution of the loss value is similar to the bimodal distribution, so the patterns corresponding to the loss value with the higher frequency have a high possibility of being outliers.

The training procedure begins with the whole training set initially. When the normalized variance of loss value over the training set (call it v_{ts}) is below 0.1, which means the most of patterns are learnt by the network well, the outlier removal should begin. Calculate the mean loss m_{ts} over the current training set and due to the dominance of the low loss peak, m_{ts} will be very low, but greater than nearly all loss in the low error peak (due to the presence of the high loss peak), so the patterns whose loss value is greater than m_{ts} identified as outliers and should be isolated from training set. Then, calculate the mean loss m_{ss} and standard deviation v_{ss} of this subset. The dominance of the outliers in the subset will skew the distribution towards the outliers so m_{ss} will be heavily influenced by the outliers and hence will be relatively high. Thus, the outliers in subset will appear- all patterns from the training set with

$$loss \geqslant m_{ss} + \alpha v_{ss} \qquad where\ 0 \leqslant \alpha \leqslant 1 \tag{5}$$

will be identified as outliers and removed from the training set.

Should BDR be continued indefinitely, eventually all the patterns would be removed from the training set. Removal of the outliers causes the high error peak shrink, resulting in a lower m_{ts} and a very much lower v_{ts}. It is v_{ts} that can be used as a halting condition for training because a very low v_{ts} indicates the training result is steady enough and there is almost no outliers existing. In the experimental code, the α value in (3) is set to 0.5 and the removal is repeated every 64 batches until v_{ts} is smaller than 0.01 (it is a typical figure recommended in the research paper).

2.5 Evaluate the performance of CNN

The performance of the CNN classifier is evaluated in terms of training time required on training set, classifying time and classification accuracy on the testing set. Checking these properties is the most straightforward method to see the efficiency of the classifier. To compute the total training time, at the beginning of the training, record the current time and after the training ends, record the current time. The difference value between the two times is the training time. Similarly, the classifying time can be obtained in the same way in testing.

To see the accuracy of the classifier, the testing dataset which has no overlapping with the training dataset is fed into the classifier and at this time we do not provide target for it and cancel the back-propagation. Then calculate the number of successful recognized patterns (i.e. the index of the max log-probability is equal to the target) divide the total number of patterns and this is the accuracy of this test.

3 Result and Discussion

The comparison is based on the evaluation methods. In the first part, I will compare the training time on training set, classifying time and accuracy on testing set between the original CNN and the CNN with BDR and try to find out the reason of the difference or similarity of them. In the second part, I will compare the classifying time and accuracy between the CNN with BDR and the classifier proposed in a published research.

Comparison of the original CNN and the CNN with BDR. I train the two networks on the same MNIST training set and test them on the same MNIST testing set ten times respectively. The average training time required by the original CNN is 607.739 ms and by CNN with BDR is 423.354 ms. Figure 3 shows the training time in each training epoch with the two networks.

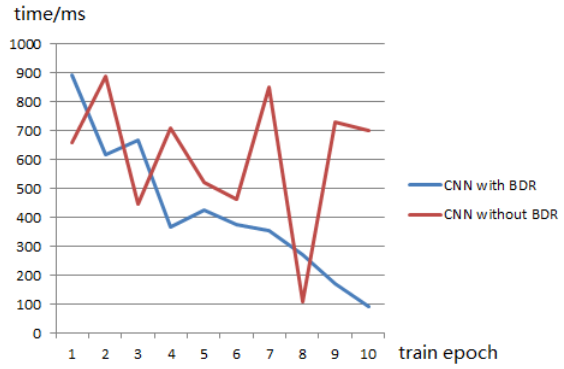

Fig. 4. Training time required for 10 train epoch of the two CNN networks.

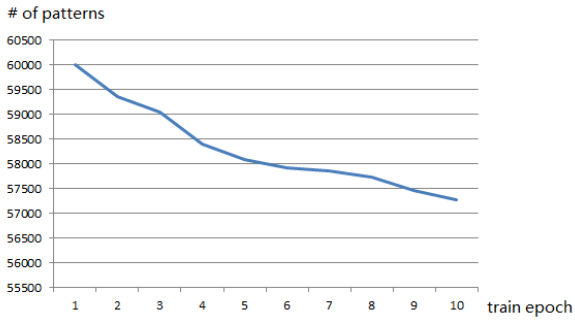

Fig.5. The number of patterns passed into the CNN with BDR in each train epoch.

From the above figures we can see BDR can reduce the training time for CNN. Focusing on CNN with BDR, with the increasing of train epoch, the training time presents a downward trend. The reason is that in each epoch, BDR will detect some outliers and remove them, then in the next epoch, the number of patterns passed to the CNN is decreased (see Figure 4), which means there is less input data, therefore the training time required by the next is reduced. This can also explain why the average training time used by CNN with BDR is less than the original CNN.

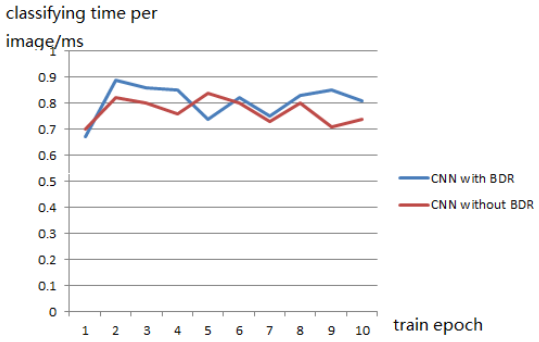

Fig.6. The classifying time for each image in testing set.

The average classifying time per image of the original CNN is 0.77 ms and that of the CNN with BDR is 0.80 ms. From Figure 6 we can see the time required by the two networks is close, stable between 0.7 ms and 0.9 ms. This is because BDR is only implemented in the training procedure thus the classifying in testing is not influenced at all.

Table 1. Comparison of the accuracy over ten tests

Test epoch	CNN without BDR	CNN with BDR
1	0.94	0.94
2	0.96	0.96
3	0.97	0.97

4	0.97	0.97
5	0.97	0.98
6	0.98	0.98
7	0.98	0.98
8	0.98	0.98
9	0.98	0.98
10	0.98	0.98

Over the ten tests, the average accuracy of the original CNN comes to 96.9% while the CNN with BDR reaches 97.0%. The accuracy obtained in each test epoch is almost equal (see Table 1). This indicates that BDR only helps improve the performance of the CNN very slightly. To explore why this outlier removal algorithm has such little effect on the CNN, I visualize the loss distribution of the training procedure (Figure 7).

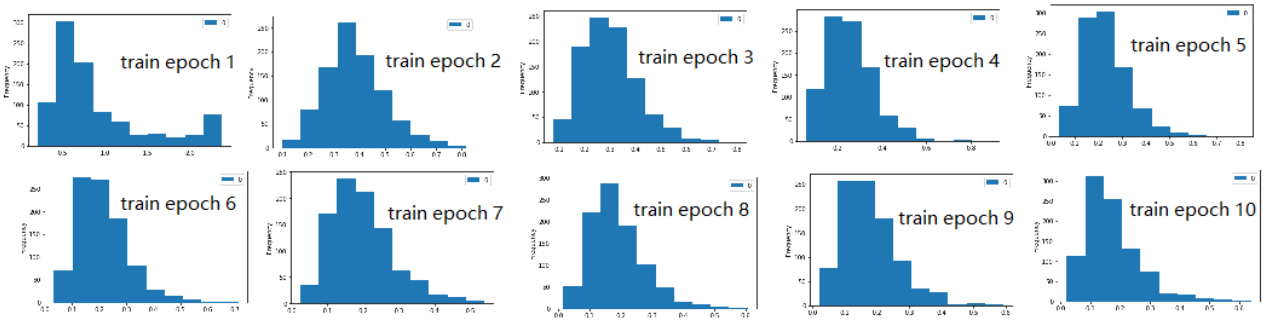

Fig.7. The loss distribution of each train epoch

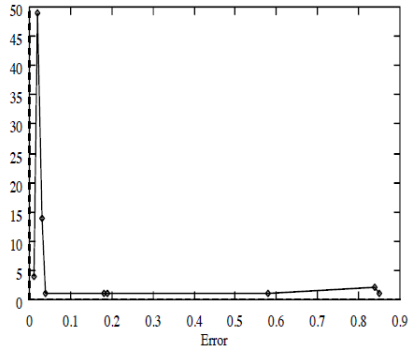

Fig.8. The loss distribution in the example of "Bimodal Distribution Removal" paper

As mentioned in the proposed "Bimodal Distribution Removal" paper, the loss distribution (Figure 8) is almost bimodal with the low loss peak containing patterns which the network has learnt well, and the high loss peak containing the outliers. Thus, BDR algorithm can be used to locate the outliers and drop the outliers in next training epoch. But the loss distribution in MNIST dataset (Figure 7) seems not like bimodal. Except the first epoch, the second peak does not appear. Since there is not a bimodal distribution, the abnormal values identified by the program may not be true outliers. As the dataset used here is a human-selected dataset and most of data it contains is meaningful (all of the data is real handwritten digits and can be recognized by human), hence the dataset is pretty clean. The methods of outlier removal such as BDR are more suitable for noisy datasets as it can accurately detect the outliers and remove them from the training set.

Comparison of the CNN with BDR and a k-Nearest Neighbor classifier. As mentioned in the introduction section, Bin Zhang and Sargur N. Srihari proposed a classifier in "Fast k-Nearest Neighbor Classification Using Cluster-Based Trees" [5] on MNIST dataset. They created a cluster-based tree algorithm to accelerate k-nearest neighbor (k-NN) classification by introducing class-conditional clustering and establishing two decision levels for early decision making. This algorithm focused on speeding up k-NN search to reduce the classifying time. Since the paper just present the accuracy and the recognition time (classifying time) per numeral, I compare these two aspects. All time units are in milliseconds.

Table 2. The result of the "Cluster-Based Trees" k-NN classifier

	Recognition Performance							
	accuracy=97.25%		accuracy=97.70%		accuracy=97.86%		accuracy=98.00%	
	computation	time	computation	time	computation	time	computation	time
Cluster	1115	0.506	1919	1.066	2329	1.484	3840	4.010

Table 3. The accuracy and classifying time of CNN with BDR

Test epoch	classifying time	accuracy
1	0.67	94%
2	0.89	96%
3	0.86	97%
4	0.85	97%
5	0.74	97%
6	0.82	98%
7	0.75	98%
8	0.83	98%
9	0.85	98%
10	0.81	98%

From Table 2 and Table 3 we can see the "Cluster-Based Trees" k-NN classifier has a more stable accuracy because the difference between the highest accuracy and the lowest one is only 0.75% whereas that of my classifier is 4%. Both of the classifiers have the highest accuracy of 98%. As for classifying time, though the research obtains the least time, with the increasing of the accuracy, the time required rises dramatically, while the time of my classifier is stable at about 0.8. Overall, from my perspective, the CNN classifier with BDR is more efficient than the k-NN classifier as reaching the same highest accuracy, the CNN classifier spends less time per image and for the whole dataset containing ten thousands images, it will save a lot of time.

4 Conclusion and Future Work

In this paper, I explained why CNN is chosen for recognizing images and detailed how to build a CNN for a real world dataset MNIST. I also implemented an outlier detection algorithm "Bimodal Distribution Removal" on the CNN and tried to evaluate the performance of the classifier by comparing it with the original CNN and with a published classifier. The result shows that BDR can slightly improve the accuracy and accelerate training time of CNN on MNIST dataset. Furthermore, a CNN classifier with BDR can provide relatively high efficiency.

Compared to the approaches that improve neural networks such as pruning, BDR algorithm is much easier to manipulate because it actually makes changes to external data sets. It can improve the accuracy of CNN to some extent and obviously save the training time. However, the BDR algorithm may not be suitable for all data sets. It is more suitable for noisy data sets and it does not have much effect on clean datasets. This manifests that we need to be more flexible in determining the distribution of loss values and formulating corresponding outlier removal schemes. The future work may involve adding BDR algorithm into other classifiers such as decision tree and k-Nearest Neighbor to see how BDR works with different classification methods.

References

1. Slade, P., & Gedeon, T. D. "Bimodal Distribution Removal". In *International Workshop on Artificial Neural Networks.* Springer, Berlin, Heidelberg. pp. 249-254. June 1993.
2. P. W. Frey and D. J. Slate. "Letter recognition using Holland-style adaptive classifiers". Machine Learning. vol. 6. no. 2. pp. 161_182. 1991.
3. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proc. IEEE*, vol. 81, no. 11, pp. 2278-2324, Nov. 1998.
4. "Convolutional neural network", *En.wikipedia.org*, 2018. [Online]. Available: https://en.wikipedia.org/wiki/Convolutional_neural_network. [Accessed: 28- May- 2018].
5. B. Zhang and S. N. Srihari, "Fast k-nearest neighbor classification using cluster-based trees," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 4, pp. 525–528, Apr. 2004.
6. P.Y. Simard, D. Steinkraus, and J.C. Platt. "Best practices for convolutional neural networks applied to visual document analysis". In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, vol. 2, pp. 958–962, 2003.
7. "MNIST database", *En.wikipedia.org*, 2018. [Online]. Available: https://en.wikipedia.org/wiki/MNIST_database. [Accessed: 28- May- 2018].
8. "Getting started with PyTorch for Deep Learning (Part 3: Neural Network basics)", *Code to Light*, 2018. [Online]. Available: https://codetolight.wordpress.com/2017/11/29/getting-started-with-pytorch-for-deep-learning-part-3-neural-network-basics. [Accessed: 28- May- 2018].
9. L. MIRANDA, "Understanding softmax and the negative log-likelihood", *Lj Miranda*, 2018. [Online]. Available: https://ljvmiranda921.github.io/notebook/2017/08/13/softmax-and-the-negative-log-likelihood. [Accessed: 28- May- 2018].