THE IMPLEMENTATION OF BIMODAL DISTRIBUTION REMOVAL ON CONVOLUTIONAL NEURAL NETWORKS AND APPLICATION ON SEMEION HANDWRITTEN DIGIT DATA SET

Zimin Wan

Research School of Computer Science, Australian National University
U6013849@anu.edu.au

Abstract. Convolutional Neural Networks (CNNs) is a category of Neural Networks that have proven very effective in areas such as image recognition and classification, especially handwritten digit recognition. While dealing with real-world data set, noisy is inevitable, which may produce incorrect classification results. Thus, outlier detection and removal are integral parts of training networks. In this paper, we implement the bimodal distribution removal (BDR) method on Neural Networks and also extend BDR to Convolutional Neural Networks for better performance and test the models on the Semeion Handwritten Digit Data Set. Our experimental results suggest that the BDR method works very well to remove the noises of the training set and halt the training automatically, as well as improve classification accuracy. The highest accuracy of CNN model with BDR reaches 97.32%. The classification results are compared with the results of other researchers' work on the same data set by using Slope Detail feature, and the performance of the implementation of CNN with BDR method is not worse than the performance of Slope Detail.

Keywords: Bimodal Distribution Removal, Deep Learning, Convolutional Neural Networks, Handwritten Digit Recognition

1 Introduction

The handwritten digit recognition problem is a significant part of machine learning and computer vision applications. It is the problem of identifying to which digit a new instance represented, on the basis of a training set of data containing instance whose represented digit is known. Digit recognition is widely used to deal with in postal mail sorting, bank check processing, financial statements and other fields, and the accuracy and time of handwritten digit recognition is crucial [6]. Therefore, improving the performance of handwritten digit recognition is still an active topic in pattern classification research.

Many classification techniques have been employed to solve the handwritten digit recognition problem, such as k nearest neighbour (k-NN), support vector machine (SVM) and multilayer perceptron (MLP). Meanwhile, the comparison of the performance of different classifiers has been investigated and increasingly concerned [5, 6]. Among the investigated classifiers, deep learning especially convolutional neural networks is considered as a good technique due to its ability of object recognition in image data [2]. In the meanwhile, the developments in techniques that improve the performance of the neural networks are increasingly attractive, and many algorithms for cleaning up noisy training sets to improve generalisation have been proposed.

In this paper, we focus on the handwritten digit recognition problem by using deep learning (convolutional neural networks) and compare the performance with the neural networks with one hidden layer (NN). Then we implement a published well-performed method proposed by Slade and Gedeon (1993) [9] to clean up noise on a real-world data set, test whether it can optimise the classifiers to improve the accuracy of classification and reduce the training time. The results of the experiment are compared with results published in another research paper by Hafiz and Bhat (2014) on the same data set [7].

The dataset used in this paper is obtained from the UCI website, named "Semeion Handwritten Digit Data Set" [8], which is a classic handwriting digit recognition data set, with each digit image has been converted to 256 attributes representing gray scale for 256 pixels. The dataset contains 1593 handwritten digit instances from around 80 persons. Each person wrote all the digits from 0 to 9 on a paper twice, with the first time trying to write each digit accurately and the second time in a fast way which lacks accuracy. Thus, some of the digits may be unrecognizable or confused, which can be regarded as a kind of noises. Therefore, it is an appropriate data set to test the performance of BDR.

The methods of the implementation are presented in Section 2. In Section 3 of this paper, we report the experimental results to compare the performance of different methods and compare results with other work, meanwhile, we discussed potential reasons of the experimental results. Finally, a conclusion and the potential future work are given in Section 4.

2 Method

In this paper, the neural networks model is established and trained to solve the handwritten digit recognition problem, and the work is extended to convolutional neural networks, then bimodal distribution removal method is implemented to remove outliers and improve the generalisation of the networks.

In this section, we will discuss the main methods we used to solve the classification problem on the Semeion Handwritten Digit data set, including pre-process the data set, details of the implementation of the neural networks and convolutional neural networks, the method to evaluate the networks and the bimodal distribution removal.

2.1 Pre-process dataset

Semeion handwritten digit data set consists of 1593 instances and 256 attributes of each. Each instance originally scanned with a resolution of 256 gray scale. Each pixel of each original scanned image was first stretched and scaled between 0 and 1 (setting to 0 every pixel whose value of the grey scale was less or equals to the value 127 and setting to 1 each pixel whose original value in the grey scale was over 127). Then each binary image was scaled again into a 16×16 square box (the final 256 binary attributes). Thus, we can regard each instance as a handwritten digit image with size 16×16. Besides, the 10-column binary numbers at the end of each instance represent the digit corresponding to 256 attributes, the index (starts at 0) of "1" indicates its represented digit. e.g. 0 0 1 0 0 0 0 0 0 0 represents 2 is the correct digit.

Split Row Data. The way we used the dataset was to split data into two parts randomly, 80% for training and 20% for testing. The training set is used for the customised networks training as well as verifying and optimising, and the testing set is to test the networks.

Input and Output Coding. Data set containing values for 256 attributes used to classify 1593 handwritten digit images into the corresponding number. The input of the NN is the 256 attributes and the input of the CNN is a is reshaped into a $1593 \times 1 \times 16 \times 16$ four-dimensional array which is to be convolved. The value of the attributes is binary, so there is no need to be normalised. The 10-column outputs are discarded, and a new column is added at the end to indicate the digit (range from 0 to 9) corresponding to the attribute according to the original 10-column variables. Moreover, there is no missing attribute value in this data set.

2.2 Implementation of Neural Networks

A Neural Networks with one hidden layer is implemented by using PyTorch to solve the handwritten digit recognition problem initially. We implement a typical training procedure for the neural networks with back propagation.

There are 256 neurons in the input layer, representing the features of image gray scale, 300 neurons in the hidden layer, and 10 neurons in the output layer representing the digit 0 to 9. Sigmoid is chosen as the activation function.

$$f(x) = sigmoid(x) = \frac{1}{1+e^{-x}} \quad (1)$$

Sigmoid function is one of the most common used in Neural Networks, it exists values between 0 to 1. Therefore, it is suitable used for models that we have to predict the probability as an output.

2.3 Implementation of Convolutional Neural Networks

Then the work is extended by using the deep learning method, and a convolutional neural networks is implemented by using PyTorch. The implemented CNN takes a 16×16 pixel, greyscale, input image, fed through several layers, one after the other, and finally gives an output vector.

The CNN consists of an input and an output layer, as well as multiple hidden layers. Different from neural networks, CNNs make use of convolutional layers (including convolution stage, detector stage and pooling stage) and fully connected layers instead of normal densely connected neurons, and convolutional layers perform convolutions rather than general matrix multiplication. The implemented CNN model is shown in Fig. 1.

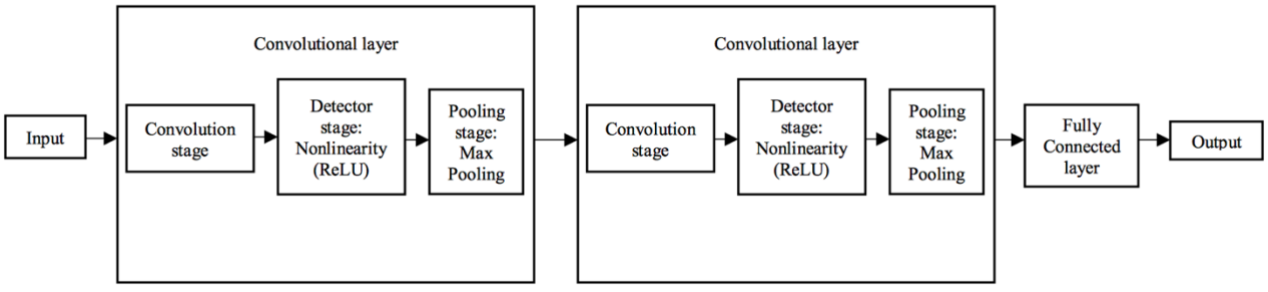

Fig. 1. CNN structure.

The convolution emulates the response of an individual neuron to visual stimuli. A single convolutional layer consists of filters that each play the role of a feature detector. During training, a CNN learns appropriate parameters for these filters. Similar to NN, the output from the convolutional layer is transformed by applying a non-linearity. Pooling stage can downsample feature map into a condensed version and assists in controlling overfitting.

Two convolutional layers are implemented in this research. Within the first convolutional layer, the convolution stage transfers the $1 \times 16 \times 16$ input into 16 filters, i.e. the output size is $16 \times 16 \times 16$. The kernel size (i.e. local receptive field) of the first convolution is 5×5 to detect the features. After that the activation function ReLU is used, and the output is downsampled by using max pooling. The output size of the first convolutional layer is $16 \times 8 \times 8$. The second convolutional layer is similar to the first one but with kernel size 3×3 to detect features better as the output from the

first layer is 8×8, it is not suitable to use a 5×5 kernel on the second convolution. And it transfers the result into $32 \times 4 \times 4$. Then the output is flattened into one-dimensional vector and access to fully connected layer to generate the final output.

Instead of Sigmoid, rectified linear units (ReLU) is used as activation function. It is the most used activation function in recent deep learning especially CNN.

$$f(x) = max(x, 0) \tag{2}$$

A rectified linear unit has output 0 if the input is less than 0, and raw output otherwise. That is, if the input is greater than 0, the output is equal to the input. The machinery of ReLUs is more like a real neuron in human body.

2.4 Backpropagation

The loss function and the optimization for the networks are introduced as follows.

Loss Function. The loss function takes the (output, target) pare of inputs and computes a value that estimates how far away the output is from the target. We choose the cross-entropy loss here to measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label.

In mathematics, we can calculate a separate loss for each class label per observation and sum up the results.

$$-\sum_{c=1}^{N} y_{o,c} \log(p_{o,c}) \tag{3}$$

where we have the following.

- N - number of classes
- y - binary indicator (0 or 1) if class label c is the correct classification for observation o
- p - predicted probability observation o is of class c

Optimization. To backpropagate the error, the whole computational graph is differentiated with respect to the loss, so that the weights can be updated to minimise the loss function. The networks will be trained with Adam as an optimiser. Adam is different to the classical stochastic gradient descent (SGD), it is an extension of SGD algorithm which is widely used in deep learning applications recently, especially tasks such as computer vision and natural language processing. Adam as combining the advantages of Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp), which are two extensions of SGD. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients, which is more effective than SGD [4]. It shows that Adam has better performance than other optimisers on deep learning.

2.5 Evaluation of Model

To evaluate how well the networks performs on different digits, a confusion matrix, which is an intuitive method is implemented.

Each row of the matrix represents the instances for the actual digit while each column represents the instances in a predicted digit. It makes it easy to show if the neural network is confusing different digits (i.e. commonly mislabelling one digit as another). An example of the confusion matrix is shown as below.

```
Confusion matrix for testing:

 35     0     0     0     1     0     0     0     1     0
  0    30     0     0     0     0     0     1     0     0
  0     1    35     0     0     0     0     2     1     0
  0     0     0    29     0     1     0     0     0     2
  0     0     0     0    29     0     0     1     0     1
  0     0     0     1     1    25     0     0     0     0
  1     0     0     0     0     1    33     0     0     0
  0     0     0     0     0     0     0    29     0     2
  0     1     1     0     0     0     0     0    29     0
  0     0     0     0     0     0     0     0     0    36
[torch.FloatTensor of size 10x10]
```

Fig. 2. Confusion matrix.

The 10×10 confusion matrix indicates that the performance of the networks on testing set. The number on the diagonal from left to right represents the number of correctly predicted digit from 0 to 9. For example, the number "35" at third column and third row indicates that the networks correctly identified 35 digits "2", the number "1" at third column ninth row indicate that the networks incorrectly identifying one digit "8" as digit "2", and the number "2" at eighth column third row indicate that the networks incorrectly identifying two digits "2" as digit "7".

Besides, the testing accuracy of the model and the training time are also computed to evaluate the performance of the classifier. Testing accuracy is the ratio between the number of correctly identified digits and the total number of digits on testing set, which quantitatively illustrates the performance of the model.

2.6 Implementation of Bimodal Distribution Removal

While training the networks, in order to detect and remove the noise in the data set to improve the performance of the networks, bimodal distribution removal is implemented for cleaning up noisy training sets to improve generalisation.

Before bimodal distribution removal is proposed, a number of outlier detection methods have been used to denoise, such as Least Median Squares, Least Trimmed Squares [3]. However, weaknesses accompany these methods such as poor performance is shown in real world data set.

Bimodal distribution removal (BDR) is proposed by Slade and Gedeon in 1993 [9], it is a statistically based method which performs well on real world data to denoise training sets. BDR can address all the weaknesses of the outlier detection and removal methods before. In addition, it provides a natural stopping criterion to terminate training, which optimize the training process to prevent overfitting.

To implement bimodal distribution removal, the variance of the error distribution is calculated. As the networks begins to learn the majority of the patterns, the variance drops sharply. Once the variance drops below 0.1, two error peaks have formed, and the removal of outliers can begin. All errors of each pattern at one epoch are collected, then the patterns with error greater than mean error are chosen as a subset. The mean δ_{ss} and standard deviation σ_{ss} of the subset is calculated, and each pattern in the subset with

$$error > \delta_{ss} + \alpha\sigma_{ss} \qquad where 0 \leq \alpha \leq 1 \tag{4}$$

is removed from dataset. α is the coefficient to determine how many patterns to remove, the smaller the α, the more patterns will be removed.

The removal repeats every tens of epochs to enable the networks have time to learn the features of each new training set. The NN implement removal every 50 epochs and the CNN implement removal every 10 epochs as the training of CNN is faster. The high error peak shrinks to a very small constant due to the removal of the outliers from the training set and the variance of the training set will gradually reduce, thus variance of all error below 0.01 can be regarded as a halting condition for the training.

3 Results and Discussion

3.1 Neural Networks and Convolutional Neural Networks

Table 1 illustrates the comparison of the performance between Neural Networks and Convolutional Neural Networks, including the average testing accuracy and the training time. Standard deviations are given in parentheses.

Table 1. Neural Networks and Convolutional Neural Networks

Methods	Testing Accuracy (%)	Training Time (s)
NN	91.505 (1.35)	4.748 (0.33)
CNN	95.368 (0.88)	37.2 (1.21)

As mentioned above, the implemented neural networks have 256 neurons in the input layer, 300 neurons in the hidden layer, and 10 neurons in the output layer. The learning rate is set to be 0.001 and it will be trained 500 epochs to obtain the best testing effects, with accuracy about 91.505%. The implemented convolutional neural networks have two convolutional layers to process image and first convolutional layer with the kernel size 5×5 and second with kernel size 3×3, and both of convolutional layer with stride step 1, which increases overlap between units in convolution layer. The average accuracy of CNN is about 95.368%. The training and testing loss of both NN and CNN are shown as Fig. 3.

Fig. 3. Historical loss of training and testing set of NN (left) and CNN (right).

According to the comparison of NN and CNN, the testing accuracy of CNN is much better than NN, despite the slow training time. Because when we are training Neural Networks to process the image, we need to discard the original shape of image and flatten it as a vector before we can feed it as input to the NN's first fully connected layer. It is an important issue especially when the inputs are the information for each pixel of the image which have natural spatial correlation along the horizontal and vertical axes. However, Convolutional Neural Networks can address this problem by using a more structured weight representation. It does not flatten the image and do a simple matrix-matrix multiplication, instead, it employs convolutional layers that each performs a 2-D convolution on the input image. The convolution layer can extract specific features of the digit image to achieve the effect of classification. Thus, convolutional neural networks is more appropriate and effective than the neural networks to recognize handwritten digits.

3.2 Bimodal Distribution Removal on Networks

The Table 2 illustrates the comparison of the models with and without using BDR, including the average testing accuracy and the training time. Standard deviations are given in parentheses.

Table 2. Networks with and without BDR

Methods	Testing Accuracy (%)	Training Time (s)
NN	91.505 (1.35)	4.748 (0.33)
NN + BDR	92.006 (1.02)	1.872 (0.17)
CNN	95.368 (0.88)	37.2 (1.21)
CNN + BDR	96.843 (1.06)	19.3 (1.01)

As we can see from above comparison, once the bimodal distribution removal applied, the training became efficient. For the neural networks, the training will usually halt at around epoch 150, and the accuracy of testing is around 92.006%, which is slightly higher than NN without using BDR. For the convolutional neural networks, the training will usually halt at epoch 100, the average testing accuracy achieve 96.843% which is also higher than CNN without using BDR. And the best accuracy of CNN with BDR can reach 97.32%. The loss of training and testing of both NN and CNN with BDR implemented is shown as Fig. 4.

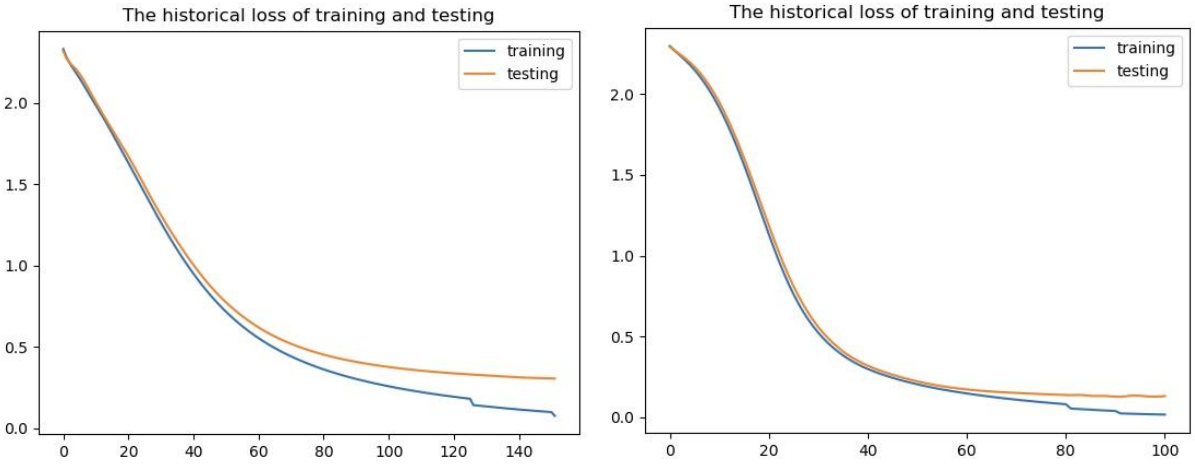

Fig. 4. Historical loss of training and testing set of NN (left) and CNN (right) using BDR.

As we can see from the loss, there are continuous gradient descents at the end at the end of the epochs, which indicates that BDR is implemented, thereby reducing the loss of the training suddenly.

Besides, the average training time of the neural networks with BDR (1.872 seconds) is much faster than the neural networks with normal backpropagation (4.748 seconds). And the average training time of the convolutional neural networks with BDR (19.3) is also faster than without using BDR. It is because that the training epochs of BDR is much less than the normal networks.

Before the BDR implement, the variance of the training set first rise and then gradually decline with the training, which is possible. For instance, the networks may classify every pattern into wrong class initially so that the variance is fairly low, after training, the networks predicts some patterns correctly and other patterns wrong which can lead to a higher variance. And the variance drops again while the accuracy is getting higher with training. Once BDR is applied, the variance shows a gradient decrease with each removal of patterns, as the purpose of BDR method is to reduce the variance of the training set. The variance of training set of CNN and CNN with BDR is shown as Fig. 5.

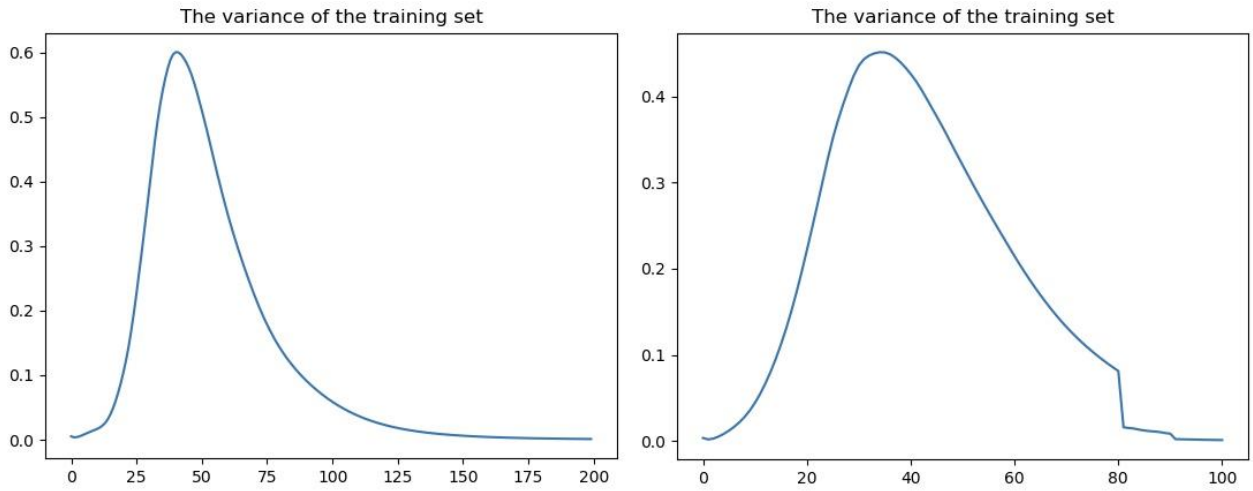

Fig. 5. The variance of training set (left: CNN, right: CNN with BDR).

While training the networks with BDR implemented, the error distribution at different epochs of CNN is shown as Fig. 6. As the handwritten digit problem have 10 classes, the error initially can distribute between these classes. The errors which distribute bimodal or multimodal gradually to one modal, and almost all errors gradually distribute at an increasingly lower level eventually.

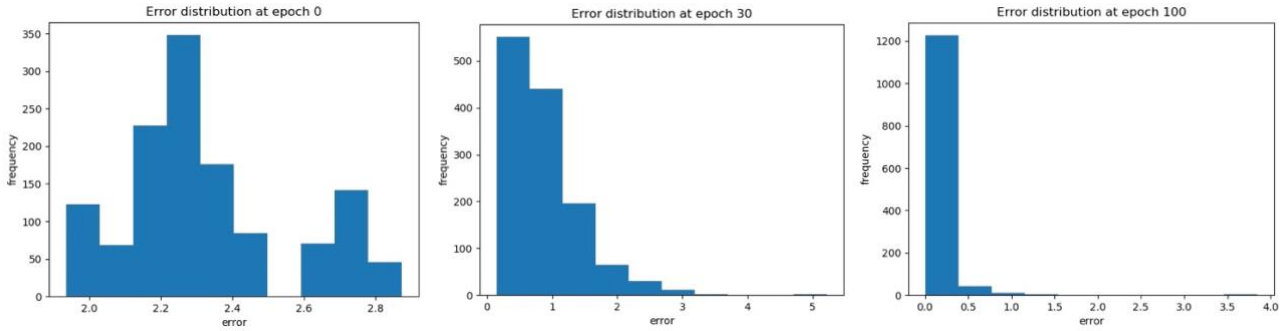

Fig. 6. Error distribution.

As we have presented above, the performance of the BDR is better than which of the normal networks, for both NN and CNN. BDR will present an even better performance if the data set is a bit noisier, as the BDR method can indeed detect the noise and remove them during the training. Besides, the training time of the networks with BDR is much faster. However, the testing accuracy is a little bit unstable, it is because that the convergence of the training is fast as we use Adam being the optimiser. The training of the networks became fast while losing little bit stability.

3.3 Comparison on CNN with BDR and Slope Detail features

Then we try to compare the performance of the implemented CNN model with others works on the same data set. Hafiz and Bhat have proposed one new feature called Slope Detail (SD) combined with commonly used features to recognize handwritten digits on the Semeion Handwritten Digit Data Set [7]. The object of their work is using the Slope Detail features to improve the accuracy of commonly used algorithms such as K-Nearest Neighbour (k-NN) and Support Vector Machine (SVM).

The comparison details are shown in Table 3.

Table 3. Accuracy Comparison of Different Methods.

Methods	Testing Accuracy (%)
NN	91.51
NN+BDR	92.01
CNN	95.37
CNN+BDR	96.84
SVM	95.80
SVM+SD	97.80
k-NN	94.20
k-NN+SD	96.80

The accuracy on the testing set of their implemented SVM and k-NN only with the commonly used features are 95.80% and 94.20%, respectively, which is not better than CNN (95.37%). And the accuracy of CNN with BDR (96.84%) is much better than the normal SVM and k-NN. The better performance may be because the CNN can better extract the features of handwritten digit images as it uses convolution and have removed the side effects of noise.

However, the accuracy of their implemented SVM and k-NN with using SD combined with other commonly used features are high, which are 97.8% and 96.8%, respectively. The accuracy of SVM with SD is higher than the implemented CNN with BDR. It indicates that Slope Detail is indeed a good feature while combined with other commonly used features to recognize handwritten digits and have an impressive accuracy on the Semeion Handwritten Digit Data Set.

4 Conclusion and Future Work

In this paper, we have presented how to implement bimodal distribution removal method while constructing the neural network and convolutional neural networks on the Semeion Handwritten Digit Data Set. It indicates that the BDR method can improve generalisation by removing noise and speed up training by reducing the number of patterns and the training epochs. The accuracy of the networks with BDR is higher than the normal networks, and the BDR also provides a natural stopping criterion to terminate training and prevent overfitting. However, it shows that the performance of the BDR is not as good as the performance of the Slope Detail features using SVM. Thus, it is crucial to choose the appropriate method for different real-world data sets.

The well performance of BDR on a real-world data set provides a new possibility for people to implement classification if the data set is noisy which requires to detect and remove the outliers. However, there are still some inadequacies in the implemented networks. The performance of the networks can be evaluated using cross-entropy, to ensure get the most appropriate parameters. Besides, there still some other classification methods for handwritten digit recognition and some other methods to improve the performance of networks, such as network reduction techniques [1], which can be compared with the current method, or combined with multiple methods, to analyse and determine which methods as well as under what circumstances have better performance on the real-world data set.

References

1. Gedeon, T. D., & Harris, D. (1991). Network reduction techniques. *In Proceedings International Conference on Neural Networks Methodologies and Applications* (Vol. 1, pp. 119-126).
2. Ghosh, M. M. A., & Maghari, A. Y. (2017, October). A Comparative Study on Handwriting Digit Recognition Using Neural Networks. *In Promising Electronic Technologies (ICPET), 2017 International Conference* on (pp. 77-81). IEEE.
3. Joines, J. A., & White, M. W. (1992, June). Improved generalization using robust cost functions. *In Neural Networks, 1992. IJCNN., International Joint Conference* on (Vol. 3, pp. 911-918). IEEE.
4. Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv*:1412.6980.
5. LeOu11, Y., Jackal, L., Bottou, L., Brunet, A., Cortes, C., Denker, J., ... & Simaxd, P. COMPARISON OF LEARNING ALGORITHMS FOR HANDWRITTEN DIGIT RECOGNITION.
6. Liu, C., Nakashima, K., Sako, H., & Fujisawa, H. (2003). Handwritten digit recognition: benchmarking of state-of-the-art techniques. *Pattern Recognition*, 36(10), 2271-2285. doi: 10.1016/s0031-3203(03)00085-2.
7. M.Hafiz, A., & M. Bhat, G. (2014). Handwritten Digit Recognition using Slope Detail Features. *International Journal Of Computer Applications*, 93(5), 14-19. doi: 10.5120/16210-5512.
8. Semeion Research Center of Sciences of Communication, via Sersale 117, 00128 Rome, Italy Tattile Via Gaetano Donizetti, 1-3-5,25030 Mairano (Brescia), Italy.
9. Slade, P., & Gedeon, T. D. (1993, June). Bimodal distribution removal. *In International Workshop on Artificial Neural Networks* (pp. 249-254). Springer, Berlin, Heidelberg.