Implementation and improvement of different neural networks on handwriting digit recognition

Yang Yu

Computer Science and Information Technology Building Australian National University Canberra ACT Australia

Abstract. This report implemented three versions of neural networks, including simple three-layer network, an auto-associative network with share weight, and a convolutional network to classify handwritten digits. The second version of the implementation identify and reduced the noisiest patterns from the training dataset and produced a better result than the first version. The best result was produced by the third version of the implementation, the convolutional neural network. However, both second and third version was not being able to increase the performance of the neural network to the desired level, which may cause by the insignificance of the noisy data and the compression of the original dataset for auto-associative network and convolutional network respectively.

Keywords: Neural Network; Backpropagation; Share Weight; Auto-associative Network; Overfitting.

1 Introduction

1.1 Dataset Selection

Digit recognition is one of the most common data set for machine learning. The dataset used in this report is the optical recognition of handwritten digits data set (Alpaydin & Kaynak, 1998) which contain 64 dimensions and 10 classes from 0 to 9. This dataset was used to experiment linear programming boosting (Demiriz, Bennett, & Shawe-Taylor, 2002) and new algorithm of approximate maximal margin classification (Gentile, 2001).

There are three reasons to choose the optical recognition of handwritten digits dataset. Firstly, it contains 5620 instances, which can provide relatively enough training data to train the network. Secondly, it has been pre-processed and generalized into matrixes of 8x8, which only provide 64 attributes for every instance, so there is no need to process images before training and the training speed will increase because of the reduction of the input size. The final reason is that this dataset has been used in many papers, so there will be enough comparison for the result.

1.2 Problem and Modelling

The main focus of this report is to determine whether the auto-associate network with share weight method or the convolutional neural network can perform a better result on classification dataset.

Demiriz et al.'s paper shown there is a huge gap in accuracy between with noise and without noise. Auto-associative network and shared weights are normally used to compress images (Gedeon, Catalan, & Jin, 1997), but it can also extract patterns from the data. Hence using the auto-associative network with share weight to pre-learn the data set while generalizing the data's patterns. And then calculate the loss of every training patterns via the pre-learned auto-associative network, which might distinguish the noisiest data and remove them to increase the performance of the neural network.

The convolutional neural network is an important deep learning network that demonstrated a great capability for image classification problems (Yu, Jia, & Xu, 2017). In this report, the convolutional neural network was being used to classify the handwritten digits, which are compressed images.

This report has implemented an auto-associative network with three concrete layers from classification network and three mirror layers to train on the input data with 64 dimensions. After certain epochs of training, the network will remove the mirror layers and concentrate on the training of concrete layers with 64 dimensions of inputs and 10 dimensions of outputs. This report also implemented a convolutional neural network as the comparison with the auto-associated network.

1.3 Analyse methods

There are five methods were used in this report to evaluate the performance of the networks. The first one is the training accuracy. After every epoch of the training, the analyse program will calculate the accuracy of the network perform on the training dataset, which can examine the training result in real-time. However, only perform the accuracy analyse the training set is not enough to show the actual capacity of the network, because the network may be overfitting by the training and only recognize the training instead of generalizing the patterns of the data (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Hence, besides calculate accuracy on training data, the report also analysed the accuracy of testing data. The third analyse method is the loss. This method is especially important the regression network because it's the only way to show how much the regression network has learned. The fourth method is confusion matrix. This confusion matrix can easily show how much data was being misclassified. The last method is the training speed. This method is combined the first, second and the third method to calculate the increased speed of each of these figures.

2 Method

There are three versions of the network was implemented. The first version is built as a baseline to compare with the performance of the second and third version of the code. The second version of the code is based on the first version with the noisy data removal that distinguished by the auto-associated network with share weight. The third version is a simple convolutional neural network with one convolutional layer and one pooling layer.

2.1 Data processing

The first step is processing the data. Firstly, load the data via pandas. And then we only need to transfer the data into numeric because the data was already being processed into numbers and there are no characters in the data. Secondly, the pandas dataframe will be transferred into an array, which will be divided into x_array and y_array. Finally, the x_array and y_array will be wrapped by tensors and variables become the data that feed into the network. And the testing data have the same process.

2.2 The first version of implementation: simple three-layer network

The first version of the network is a simple three-layer neural network with 64 input neurons, 20 hidden neurons, and 10 output neurons. There are three different optimizers was being used in the first version neural network, Stochastic Gradient Descent (SGD), Resilient Propagation (Rprop) (Riedmiller & Braun, 1993), which is an adaptive learning algorithm, and Adam algorithm (Kingma & Ba, 2015). And the loss function used in this network is the cross-entropy loss function, which is usually used in classification network. The accuracy of both testing and training was being defined as two functions, each of the functions will be called during every epoch of the training and storing the accuracy into two separate lists.

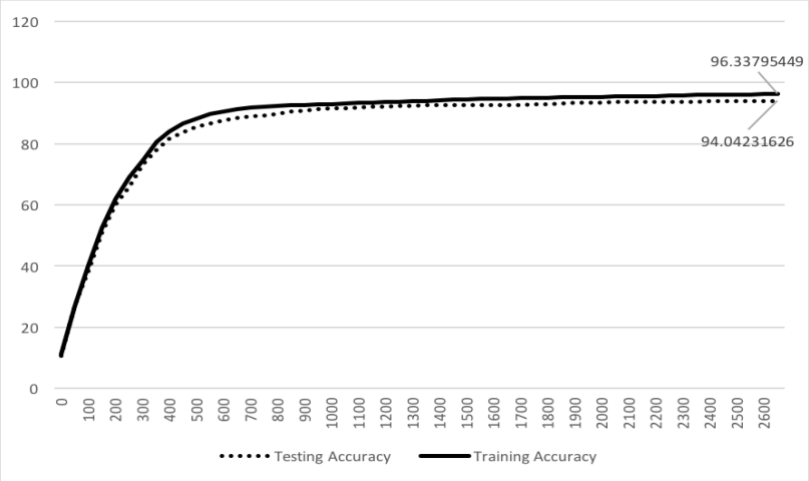

Chart 2.1 Version 1 accuracies with SGD as optimizer

The chart 2.1 has shown the accuracy on both testing and training set of the first layer. After around 600 of epochs, the increased speed on both accuracies start to drop, and both accuracies have reached over 90%. After 2600 of epochs, the training accuracy reached 96.23% and the testing accuracy reached 93.92%. And the testing confusion matrix in table 2.2 shows that the neural network can classify the data in a reasonable range, without very extreme errors.

372	0	1	0	2	0	0	0	0	1
0	368	4	1	0	0	1	1	8	6
0	0	375	2	0	0	1	0	1	1
0	0	1	378	0	5	0	1	1	3
0	5	0	0	365	0	7	1	5	4
1	0	3	1	0	363	0	0	1	7
0	3	0	0	1	0	373	0	0	0
0	2	1	1	4	0	0	377	0	2
2	16	0	1	6	5	1	0	347	2
2	8	0	5	8	1	0	9	3	346

Table 2.2 Version 1 testing confusion matrix

The Rprop optimizer has significantly boosted the speed of the training as shown in the chart 2.3. With the Rprop optimizer, the accuracy has reached the first version's level with only 50 epochs. However, the training accuracy has reached 99.9% and the testing accuracy remains basically the same as the first version's accuracy. The big gap between the accuracies indicates there is an overfitting problem. Adding more training data will be an efficient way to address the overfitting issue. However, there is no more data to train, so in order to solve the overfitting problem, other methods must be adopted.

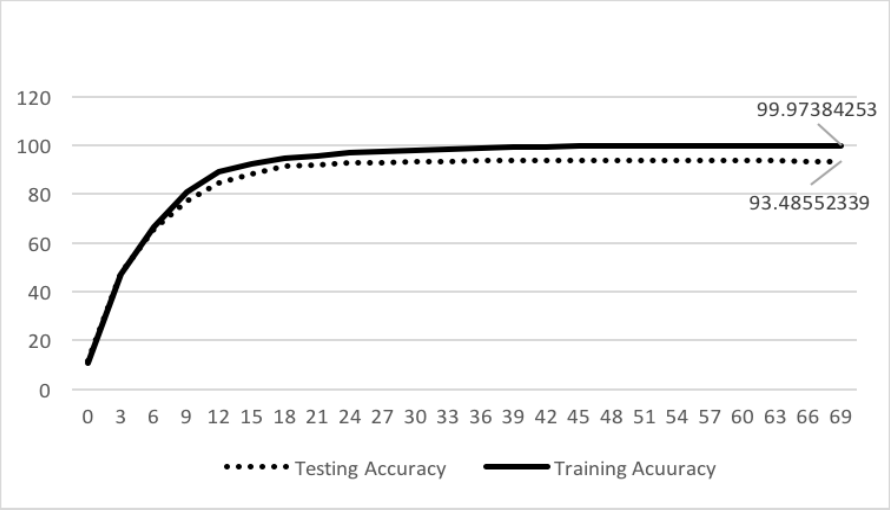

Chart 2.3 Version 1 accuracies with Rprop optimizer

The Rprop indeed boosted the speed of the training, however, the testing accuracy is basically the same as SGD. Adam optimizer on the other hand, not only boosted the speed of the training, it also slightly increased the accuracy on the testing dataset. The chart 2.4 shows the training and testing accuracies when using Adam as the optimizer.

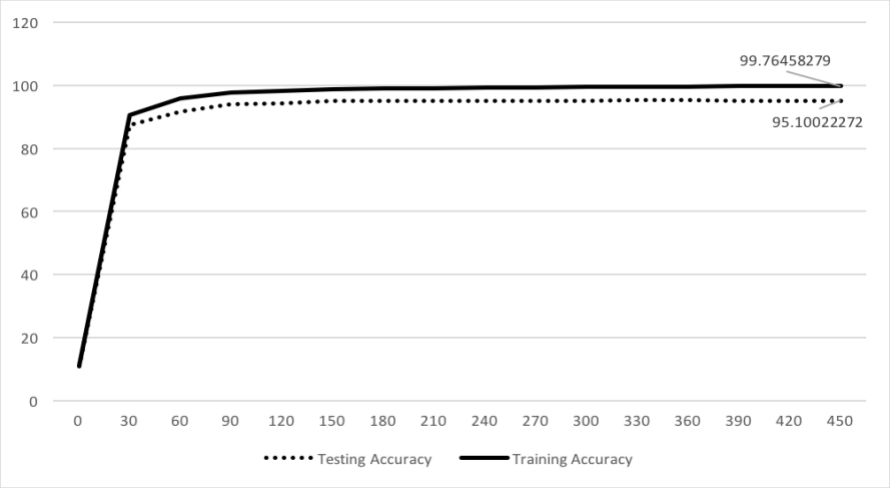

Chart 2.4 Version 1 accuracies with Adam optimizer

2.3 The second version of implementation: auto-associative network with share weight

The auto-associative network with share weight has been used to compress images (Gedeon, Catalan, & Jin, 1997). An assumption of distinguishing and remove the noisy patterns via the auto-associative network will increase the testing accuracy and mitigate the overfitting problem have been made. To verify the assumption, the report implemented an auto-associated neural network with share weight to training with the pattern as both input and output.

A new three-layer auto-associative network with share weight was being implemented. The first layer is the input layer, with 64 input neurons as same as the first and the second version. The second layer is the hidden layer, it contains 10 neutrals. The third layer is the mirror input layer with 64 output neurons. The diagram 2.5 indicated how the network was structured.

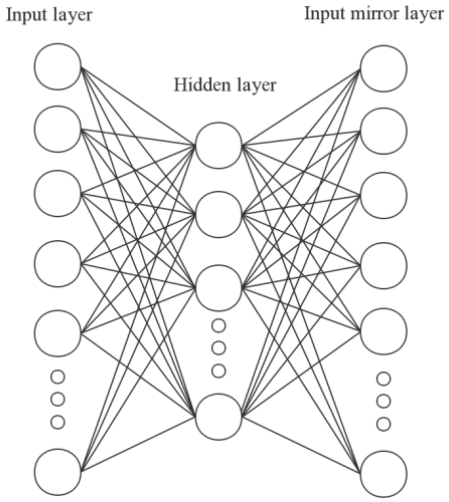

Diagram 2.5 Version 3 network

The training process has divided into two stages. The first stage is to treat the full three-layer network as a regression network and feed the input data into the input layer and use the MSE loss function to calculate the loss. In this stage, the network will extract the pattern of the data. After the first stage, the network will run through all the patterns and produce a list of loss for every pattern. The list of losses will be sorted, and these noisiest data (with the highest loss) will be deleted from the training dataset. The remaining data will be used to train the simple three-layer neural network same as the first version with Adam optimizer.

In the first stage, the report has tried to use different optimizers to reduce the loss. And the chart 2.6 showed the loss reduction process.

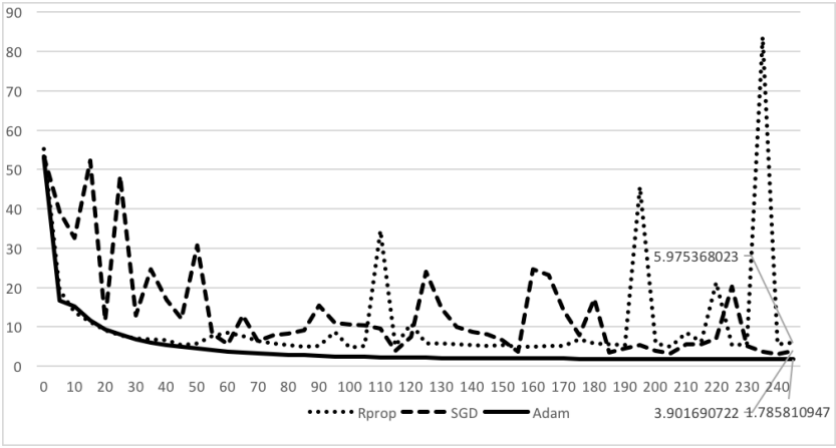

Chart 2.6 Version 2 first stage losses with share weight

It can be seen when using the SGD and Rprop as the optimizer, the loss is significantly unstable, and the numbers at the valley are around 2.90 and 5.97 respectively. The Adam optimizer, on the other hand, can produce much stabler losses and the numbers are lower than both SGD and Rprop with only around 1.78. Therefore, the best choice for the first stage is Adam optimizer.

To implement the share weight in PyTorch, after defined every layer in the network model, extract the weight between input layer and hidden layer and call its "t" function to get the tensors of the weights. After that, wrap the tensors with Parameters object and assign them to the mirror input layers as weight. This operation will connect the weight of mirror layer and the weight of concrete layer. This is the implementation of the weight sharing in PyTorch.

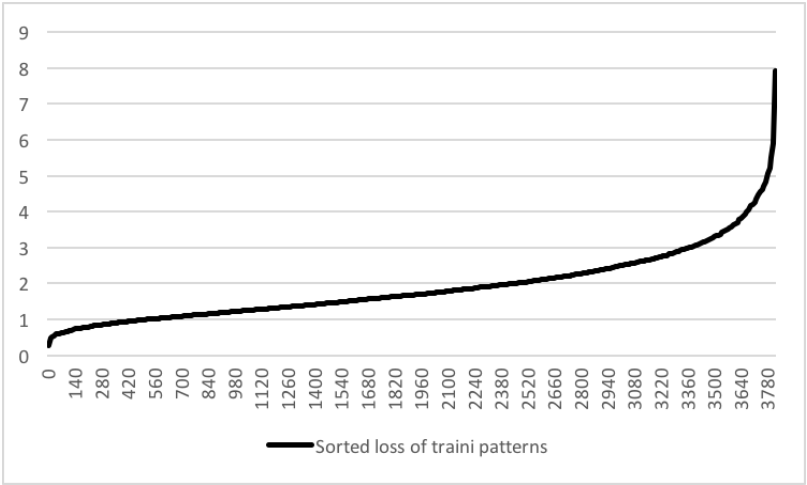

Chart 2.7 Version 2 first stage losses with share weight

In the first stage, when the loss drops to under 2, which is basically the bottom of the loss valley, the program will start to calculate the loss of every training patterns and the result is showed in the Chart 2.7. It is clear that there are small range of data that are significantly noisier than other data. For example, from around row number of 3500, the loss of patterns has a considerable increase, and this set of patterns is the part that will be removed.

In the second stage, the simple three-layer neural network with Adam optimizer that same as the first version will be trained with the cleaned training dataset. The training and testing accuracies of the second stage are displayed in the Chart 2.8.

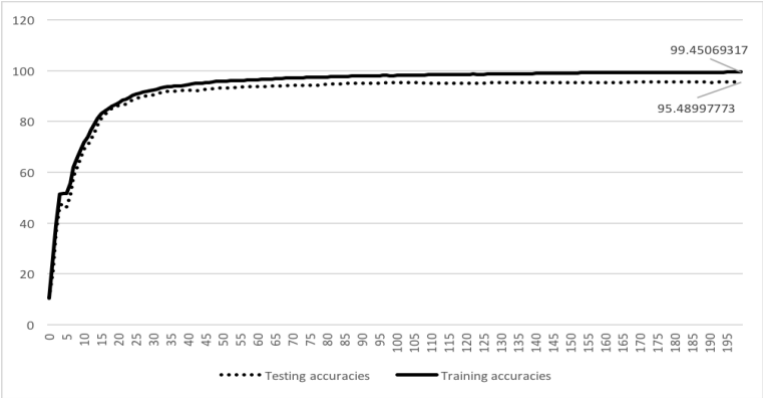

Chart 2.8 Version 2 second stage accuracies on cleaned training data

As the data have shown, while the training accuracies have no visual changes, the testing accuracies have increased around 0.3. This means the auto-associative network indeed identified noisy data, and through remove these data, the performance of the network has increased a little bit.

2.3 The third version of implementation: convolutional neural network

The convolutional neural network as a type of deep learning network are commonly used to work with pictures. The dataset that this report has been working on is a picture problem, hence introduce CNN to train and classify on the dataset might have a better result than the previous networks.

The implementation of the third version of the network is a one layer convolutional neural network with one convolutional layer, and one max-pooling layer, which is an important pooling algorithm for convolutional neural network (Nagi et al., 2011). The network has filters of size 5, patting of length 2, and stride of length 1, it also uses the ReLU as the activation function. After training with 1000 epochs, the testing accuracy has a significant increase compared with the first two versions of networks. As the result shown in chart 2.9, the best testing accuracy can reach almost 96%, which is the best result that this report achieved.

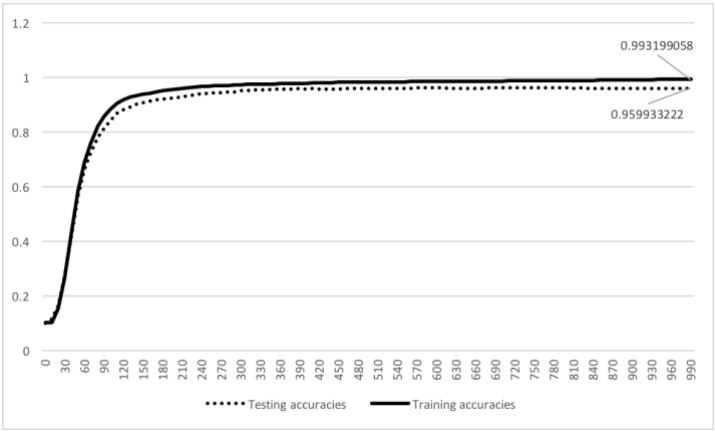

Chart 2.9 Version 2 second stage accuracies on cleaned training data

2.4 Methods of running the code

The code of the implementations can run via python2.7 environment, which requires install packages including numpy, pandas, and torch. To run the first and second version of implementation, only need to put the data folder which contain optdigits.tra and optdigits.tes file within the same directory and run the command “python first-version.py” and “python third-version.py”. The second version contain two files, the “python second-version-share-weight.py” need to be run first and generate a txt file that contain the indexes of the patterns that need to be removed, and then run the “second-version-simple-three-layer.py” file to train the neural network.

3 Result and Discussion

This report has implemented three version of the code from the basic backpropagation classification network to auto-associative network with share weight. And in the process, every version of the code has produced its accuracies on both testing and training dataset which are the most evaluation methods. To evaluate the work of this report, some comparisons with other published paper has been made. Demiriz, Bennett, and Shawe-Taylor(2002) have used the same dataset to examine the linear programming boosting by column generation. And they have achieved the testing accuracy of 95.10% with AdaBoost (Demiriz, Bennett, & Shawe-Taylor, 2002). In this report, the best testing accuracy achieved is 95.99%, which is higher than the result of the published paper.

The second and the third version of the implementation both has improved the performance of the network in certain ways. The second version of the implementation of removal noisy data via auto-associative network and share weight method has slightly increased the testing accuracy from 95.10% in the first version to 95.49%. The second version of the code has slightly reduced the gap between the training accuracy and testing accuracy from 4.66 to 3.97. This means the auto-associative network indeed extracted the feature of the data and reduced some noisy data, in the meanwhile addressed a little bit of the overfitting problem. The third version of the implementation, the convolutional neural network has produced the best result with the testing accuracy of around 96%, and the gap between training accuracy and testing accuracy have dropped to 3.32. This means the convolutional neural network has the ability of address overfitting problem in this dataset.

4 Conclusion and Future Work

This report has experimented with multiple methods to boost the network's performance. Under the structure of the simple back propagation network and limited data numbers, the best way to increase testing accuracy is to change the optimizer. To some extent, the auto-associative network with share weight can extract the data pattern and identify these noisy data that can be removed to increase the performance of the simple three-layer network. And the result has shown that the testing accuracies have indeed increased after the removal of these noisy data. The best result of this report is from the third version of the implementation, the convolutional neural network, however, the increase of the performance is still limited. Even with the best result, the gap between testing accuracy and training accuracy is still clear, which means even with CNN, the overfitting problem still exists.

There are two guesses about why the auto-associated network and convolutional neural network did not perform as good as assumed. Firstly, the data noise might not the main cause of the overfitting problem, therefore, even erase these noisy patterns, the performance still not improve too much. Secondly, the reason why the convolutional neural network did not produce the desired result is that the dataset was being pre-processed, every pattern was being compressed from 32*32 bitmaps into 8*8 data, which might have lost some valuable information that can significantly increase the performance of the convolutional neural network because convolutional neural network can extract information during the training process.

For future work, it is necessary to consider new ways of using the ability of auto-associated networks to identify noisy data in the dataset, which can be quite useful in multiple scenarios including identify abnormal data of human health and do not limit to training neural networks. And the convolutional neural network may need more raw data to train instead of compressed data, which might easier for simple neural networks to train, but not for deep learning networks including the convolutional neural network. Hence the future work can systematically research the performance of convolutional neural network on compressed and non-compressed images.

References:

Kingma, D. P., & Ba, J. (2015). Adam: a method for stochastic optimization. *ICLR*.

Alpaydin, E., & Kaynak, C. (1998, July 1). *Dataset for sensorless drive diagnosis data set*. Retrieved April 19, 2018, from UCI machine learning repository: https://archive.ics.uci.edu/ml/datasets/Dataset+for+Sensorless+Drive+Diagnosis

Demiriz, A., Bennett, K. P., & Shawe-Taylor, J. (2002, January). Linear programming boosting via column generation. *Machine Learning*, *46*(1-3), 225–254.

Gedeon, T. D., Catalan, J. A., & Jin, J. (1997). Image compression using shared weights and bidirectional networks. *2nd International ICSC Symposium on Soft Computing (SOCO'97)*, (pp. 374-381).

Gentile, C. (2001). A new approximate maximal margin classification algorithm. *Journal of Machine Learning Research, 2*, 213-242.

Nagi, J., Ducatelle, F., Di Caro, G. A., Cireşan, D., Meier, U., Giusti, A., . . . Gambardella, L. M. (2011). Max-pooling convolutional neural networks for vision-based hand gesture recognition. *2011 IEEE International Conference on Signal and Image Processing Applications* (pp. 342-347). Kuala Lumpur: IEEE.

Riedmiller, M., & Braun, H. (1993). A direct adaptive method for faster backpropagation learning: the Rprop algorithm. In *IEEE Int. Conf. on Neural Networks* (pp. 586–591).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*, 1929 - 1958.

Yu, S., Jia, S., & Xu, C. (2017). Convolutional neural networks for hyperspectral image classification. *Neurocomputing, 219*, 88-98.