Network Reduction for Fully-connected Layer and Convolution Layer

Longfei Zhao
College of Engineering & Computer Science
Australian National University
Canberra ACT 0200
u5976992@anu.edu.au

Abstract. Gedeon's paper [1] has shown that without retraining building a vector for each hidden unit and calculating their similarities is a way to prune artificial neural network. In this paper, there are some deep details and improvement about this reduction method in an artificial neural network. Furthermore, this reduction method is extended to work with a convolution layer in a convolutional neural network. The Optical Recognition of Handwritten Digits Data Set [2] is used to train with a two-layer artificial neural network and a convolutional neural network and then use reduction method on both of them. As a result, we find that a proper angle threshold will prune your neural network a lot without hurting accuracy. Finally, I give some my thought about advantages and disadvantages of this method and how to improve it in future.

Keywords: network reduction techniques, neural network pruning, convolutional neural network

1 Introduction

Gedeon's paper gives us some guidance and intuition about how to prune redundant units for a fully-connected layer in an artificial neural network. However, there are some details still uncertain or questionable for me. Therefore, I will use this paper to go through technical details, improve it with some my own ideas and finally extend it to work with a convolution layer in a simple convolutional neural network. In this paper, we will focus on how to remove redundant units without hurting accuracy and how much do angle thresholds affect the result. Improving the performance of neural network is not consideration, since reduction method maybe solve some overfitting problem but there is no evidence shown that it could be better than some popular overfitting solutions, such as regularization, drop-out. The whole work here is to give us an intuition about what reduction method do and its' potential usage.

2 Data set

This data set is extracted normalized bitmaps of handwritten digits. It is from a total of 43 people, 30 of them contributed to the training set and different 13 to the test set. There are 3823 instances in train set and 1797 instance in the test set. As shown in Table 1, The distribution of training set and testing set are generally same and uniform. For each sample, the original image is a 32x32 bitmap, which is divided into non-overlapping 4x4 blocks. Therefore, the number of features will be 64 and each feature is an integer in the range from 0 to 16. The output class is an integer in the range from 0 to 9. Some simples are shown in Figure 1.

Table 1. Class distribution of this data set

Class	Number of samples in train set	Number of samples in test set
0	376	177
1	389	182
2	380	177
3	389	183
4	387	181
5	376	182
6	377	181
7	387	179
8	380	174
9	382	190

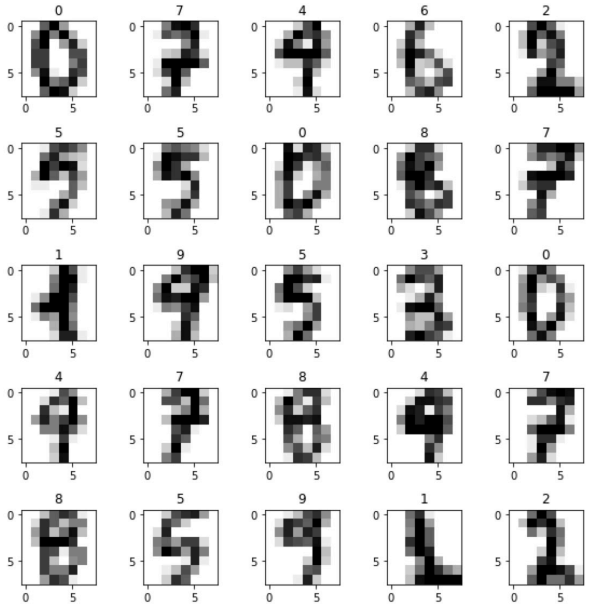

Fig. 1. First 25 samples in the data set.

2.1 Reasons for choosing this data set

There are mainly two reasons why I choose this data set. Firstly, handwritten recognition is a classic classification problem for machine learning. It's easy to get a great well-trained model with a simple neural network structure. Since we should focus on pruning a well-trained overfitting neural network, choosing hard task is not a good idea if we don't even know how to train it well or overfitting. Secondly, I didn't choose some famous data set, such as MNIST or CIFAR-10 because both of the number of instances and the number of features in this data set is much smaller than them. Therefore, it's easy to train it with different parameters in feasible time based on my computing power. After all, how to use this method in different types of layer and how does angle threshold affect accuracy is the main point in this paper.

3 Model design

In order to use this reduction method in both fully-connected layer and convolution layer, there are two neural networks. One is a simple two-layer artificial neural network, which is for the fully-connected layer. Another is a simple convolutional neural network, which is for the convolution layer.

3.1 Neural network structure

For the two-layer artificial neural network, there is just one hidden layer. The number of input feature is 64 and the number of output class is 10. Hidden layer size will be an important hyper-parameter for later implement. The basic idea is from the original paper [1]. There are some details that are missing or changed.

1. I use Tanh function for activation function instead of the Sigmoid function in the original paper. The main reason is that the output's range of Tanh function is from -1 to 1, therefore we don't need to normalize the result. Otherwise, According to Deep Learning Specialization [3], Tanh is always better than Sigmoid.
2. I use Adam method [4] for optimization since it should converge faster.
3. Cost function is CrossEntropyLoss.
4. There is a new function hidden_layer() I build for getting the result of hidden layer units. It is for the reduction method.

For the convolutional neural network, first thing we need to do is to reshape the feature from (64, 1) to (8, 8). Then there is a convolution layer. Filters are set by 5 * 5, stride is 1 and no padding. The number of filter will be an important hyper-parameter for later implement. After the convolution layer, there is just one fully-connected layer which directly connect to final output. Train details are basically same. Activation function is also Tanh function.

3.2 Performance of the neural network

For the two-layer artificial neural network, 100 seems to be a good choice for the number of epochs. I choose 50, 100, 150, 200 for the number of hidden units. Those number will be used in later too. As shown in Figure 2, after 100 epochs the trend is basically flat. The final results are shown in Table 2. As we can see, more hidden units, more quick this neural network converges.

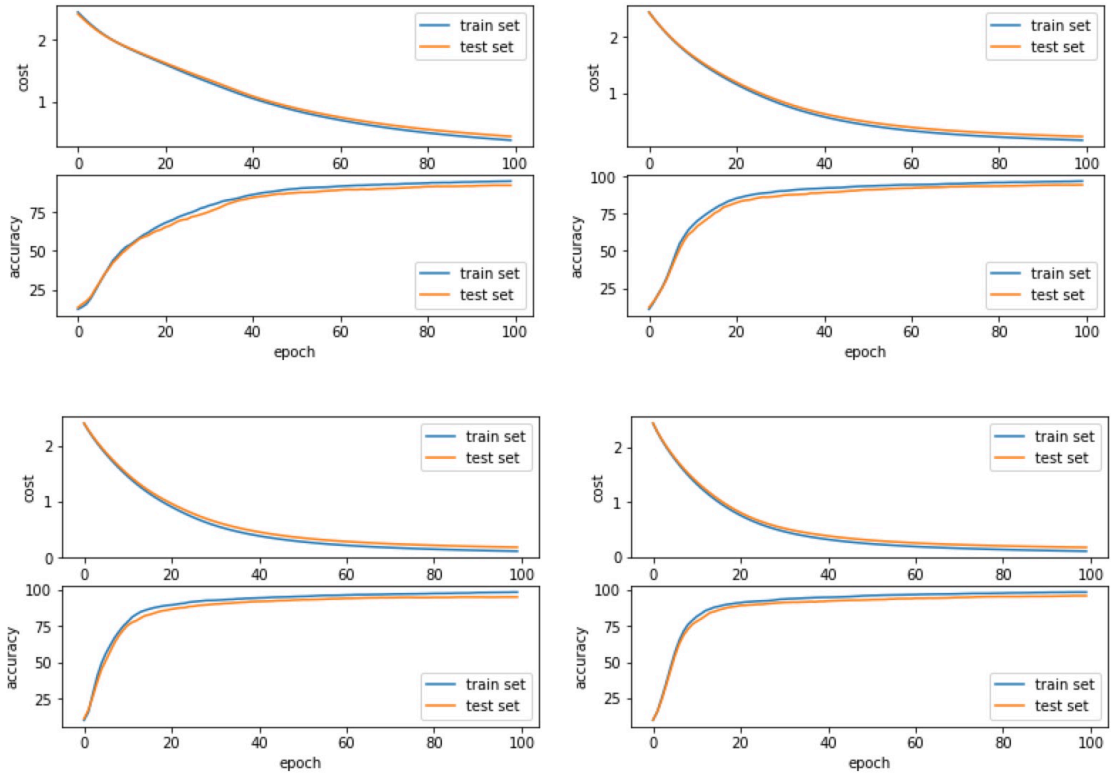

Fig. 2. For the two-layer artificial neural network, the change trend of cost and accuracy in train set and test set. There are four different number of hidden units, which are 50, 100, 150 and 200 according to the order.

Table 2. Final accuracy of train set and test set for different hidden units

	50 hidden units	**100 hidden units**	**150 hidden units**	**200 hidden units**
Train set accuracy	95.1086%	97.1227%	98.2998%	98.3521%
Test set accuracy	92.4318%	94.6021%	94.9917%	95.8264%

For the convolutional neural network, 50 seems to be a good choice for the number of epochs. I choose 50, 100, 150, 200 for the number of filter. Those number will be used in later too. As shown in Figure 3, after 50 epochs the trend is basically flat. The final results are shown in Table 3.

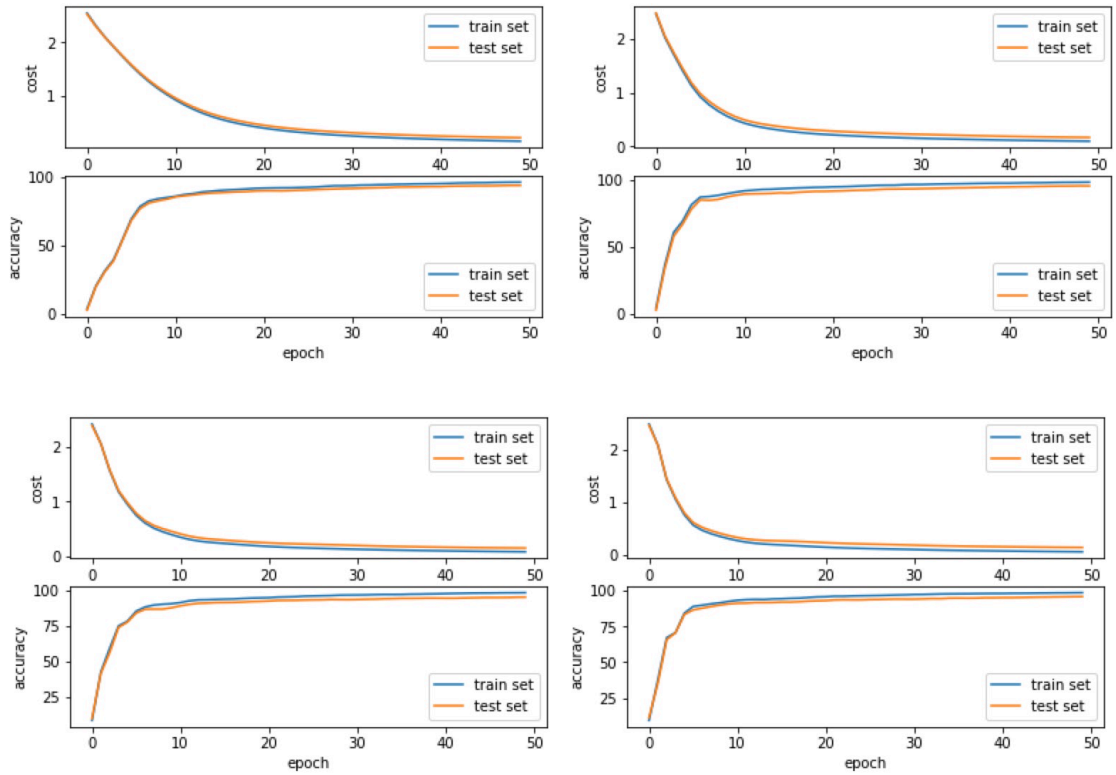

Fig. 3. For the convolutional neural network, the change trend of cost and accuracy in train set and test set. There are four different number of filter, which are 50, 100, 150 and 200 according to the order.

Table 3. Final accuracy of train set and test set for different filters

	50 filters	**100 filters**	**150 filters**	**200 filters**
Train set accuracy	96.3641%	97.9597%	98.5613%	98.5090%
Test set accuracy	93.9900%	95.1586%	95.3812%	95.7707%

I think both of those two results are good enough to be treated as a well-trained neural network. We could use those neural networks to implement the reduction method. Besides, the best result of Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition [5] which used multiple classifiers is 95% recognition and 5% rejection. We can see even simple neural network have a strong learning ability. Deep learning has a great advantage in those tasks comparing some traditional machine learning techniques.

4 Reduction method

For the fully-connected layer, since it is used by the original paper, we could use the basic idea from the original paper. There are some details that I change and workflow as follow list.

1. Instead of use 15 and 165 as threshold in the original paper, I set a parameter as threshold. Therefore, we just need to compare angle with this parameter and 180 minus this parameter.
2. At first, I tried to find which units' output are close to 0, which means the vector's length close to 0 and then remove those redundant units. However, through my observation, their vectors' length always very similar. It's hard to define the threshold for this step. Besides, the original paper doesn't define how is close to 0. Therefore, I just skip this step. This part of code is commented.
3. For convenience, I just set all redundant units' weights to 0 instead of truly removing those redundant units.
4. Firstly, for each unit I build a vector which contains this unit's output for all train example.
5. Secondly, I calculate the angle of any two units just like the original paper.
6. If the angle is smaller than the threshold, that means those two units have similar structure or behavior from input layer to hidden layer. Therefore, we don't need to worry about weights from input layer to hidden layer. We should focus on weight from hidden layer to output weight, since we don't know how those units affect output layer. We need to remove one of the units and add its weight which is from hidden layer to output layer to another one.

7. If the angle is larger than 180 minus the threshold, instead of removing both of units as the original paper, I choose to remove one of the units and minus its weight which is from hidden layer to output layer to another one. I think this method makes more sense since those units should have a different effect for output. Notice that we also focus on the weights that from hidden layer to output layer.
8. Finally, the number of units removed is counted and there is a new weight matrix.

For the convolution layer, it's much tricky. The first thing we should notice is that in convolution layer weights are combined by filters. Therefore, we should consider each filter as integral whole. We need to find a way to remove redundant filters not units. Secondly, for each filter, the output of single sample is matrix not just a real number. Therefore, we cannot use all samples for single filter to directly get a vector to represent a filter just like we did in the fully-connected layer. In fact, if we use all samples to a single filter, we will get a 3d matrix. Then we need to flatten this matrix to vector. Thirdly, for convolution layer we also need to worry about bias which we don't need in fully-connected layer. The other things are basically same as the fully-connected layer.

5 Results and Discussion

For the fully-connected layer, I used the well-trained neural networks before, which are 50, 100, 150, and 200 hidden units. I use them with all angle thresholds from 0 to 90 to get the number of removed units and new accuracy on test set. The result is shown in Figure 4. There are some interesting things. Firstly, from 0 to about 40 degrees, as we can see, the accuracy basically won't decrease and the number of removed units grows very slow. However, from 40 degrees to about 60 degrees, we can easily see that the number of removed units grows very fast and the accuracy starts to decrease slowly. After about 60 degrees, growth of the number of removed units start to become slow and the accuracy starts to decrease very fast.

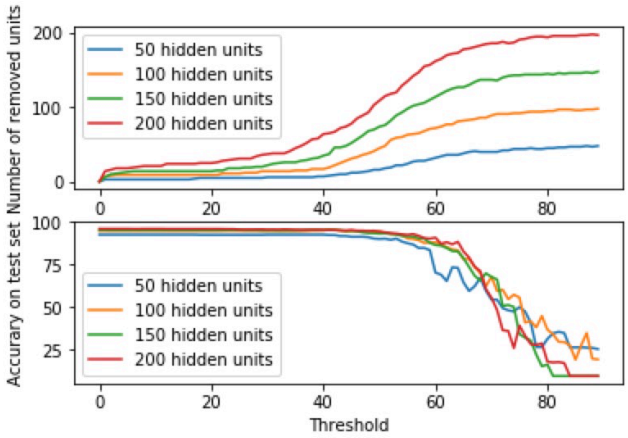

Fig. 4. For the two-layer neural network, the change trend of number of removed units and accuracy on test set through threshold (angle). There are four different number of hidden units, which are 50, 100, 150 and 200.

It's not hard to draw a conclusion that if we want to remove some redundant units without hurting the accuracy, about 40 degrees seems to be a very good choice. However, the number of removed units won't be large. If we want to remove redundant units as many as possible with acceptable accuracy, some degrees range from 40 to 60 seem to be a good choice. Also, I think that more hidden units it has, later its' accuracy starts to decrease fast. Therefore, this proper degree should be related to the hidden units you have.

For the convolution layer, I also used the well-trained neural networks before, which are 50, 100, 150, and 200 filters. As same as fully-connected layer, I use them with all angle thresholds from 0 to 90 to get the number of removed units and new accuracy on test set. The result is shown in Figure 5. The result is quite remarkable. Since, from 0 to about 60 degrees, the accuracy basically remains the same while number of removed filter is quite large. After about 60 degrees, the number of removed filter grows slowly and the accuracy decrease fast.

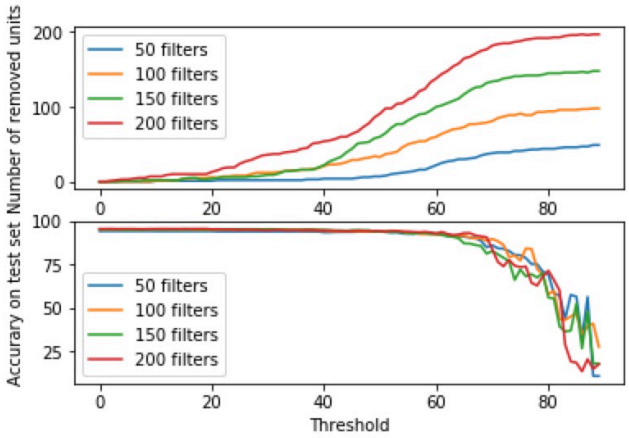

Fig. 5. For the convolutional neural network, the change trend of number of removed units and accuracy on test set through threshold (angle). There are four different number of filters, which are 50, 100, 150 and 200.

In conclusion, for convolutional neural network, 60 degree seems to be a great choice since it won't hurt accuracy and it can remove many redundant filters. Notice that I think in here 50 filters is already quite overfitting since it's just a toy problem. That is why its' performance so good. However, it still can give us some intuition about how many you over-design your neural network. In general, more filters in your neural network, more percentage of filters it can remove. As we can see in Figure 6, the percentage of removed filter is basically ordered by their filters quantity.

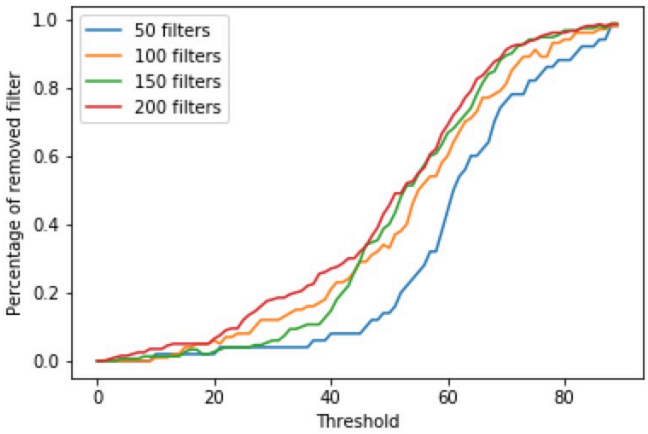

Fig. 5. Same as Figure 5, just change the number of removed units to percentage of removed units.

6 Conclusion and Future Work

The benefit of this method is that we can use it to prune some units and get a smaller neural network without retraining. It can give us intuition about how to prune some of the redundant units. However, there are, I think, still some disadvantages and problems need to figure out.

Firstly, since the hidden layers' behavior is much more complex and hard to be predicted, the whole thing seems like lack of mathematical deduction. It is more like followed some basic idea. For example, we don't consider the distribution of input and the number of samples. However, obviously the units' behavior and threshold should be closely related to those things. Secondly, the threshold is more like a hyper-parameter, that means we need to figure out a good threshold for certain task. We don't know if the conclusion we got in this problem could be used in other applications. This step will cost extra time. Besides, it just could give us some intuition or baseline about pruning neural network not the minimum neural network. Finally, it's also related to which activation function we choose. For example, in this paper I choose Tanh function instead of sigmoid function. I think different activation function maybe also give us a different result.

For improving this method, I have some thought. Firstly, for fully-connected layer, instead of using hidden layer's output, I think maybe using output layer's result makes more sense. For each unit, we can compare their contribution for output. If some of them are similar, which means angle is close to 0, we can combine them together and if some of them are opposite which means angle is close to 180 we can somehow remove them together. Notice that in this situation since we have two layers' weight, how to combine them or remove them together is still a problem. Secondly, I think that using

train set and test set together to prune redundant units maybe make more sense because we cannot assume the distribution of train set and test set are same. Therefore, just using the train set to do this step may cause some bias.

For future work, I think that to figure out the conclusion that we got is general or just for this special problem is very important. If there is some general conclusion, it could be a fixed step when we train a model to give us some intuition about the network's redundancy. Its' time cost is cheap and maybe could work a lot. Otherwise, I think how to use this method in a large network and other types of networks are very attractive.

References

1. Gedeon, T., Harris, D.: Network reduction techniques. In: Proceedings International Conference on Neural Networks Methodologies and Applications, pp. 119-126 (1991)
2. Alpaydin, E., Kaynak, C.: Optical recognition of handwritten digits data set. Department of Computer Engineering, Bogazici University, 80815 Istanbul Turkey (1994). http://archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits
3. Ng, A., Katanforoosh, K., Mourri, Y.B.: Deep learning specialization. deeplearning.ai (2018). https://www.coursera.org/specializations/deep-learning
4. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: Proceedings International Conference on Learning Representations (ICLR). arXiv preprint arXiv:1412.6980, pp 1-13 (2015)
5. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Transactions on Systems, Man, and Cybernetics 22, 418-435 (1992)