Neural Network Reduction: Practical Application of Neural Network Topology Pruning Techniques

Jayden Caelli[1]

[1] Research School of Computer Science,
Australian National University
u556455@anu.edu.au

Abstract. In this paper a convolutional neural network + feed forward neural network is trained to classify hand written digits. Using this method the accuracy of the network model was determined to be 98.47%. Network pruning techniques were investigated to reduce the network model. It was found the threshold angles used for the comparison of neurons could be used to determine the number of neuron's removed, whilst being a trade off between network size and accuracy.

For feed-forward neural networks trained using back-propagation the number of hidden units required can be difficult to determine prior to training [2]. Whilst the number of input neurons and outputs can be determined easily, by the number of input features and type of task being used, the number of hidden neurons cannot be straightforward determined.

If the number of hidden neurons used is too small, then the network may fail to learn, however if the number of hidden neurons is too high the network may take a long time to train [2]. This is one of the disadvantages of back propagation, being that it can take a long time to train a network[2].

Using the minimal number of neurons needed for good performance and efficiency for a network model has a few advantages. One of these being the network model can be made more efficient during actual use, and the minimal number of neurons can be used as guide for solving similar problems [2].

1.2 Network Pruning

Network pruning is the process of removing neurons in a trained network, with the intention to reduce the size of the network [2]. The pruning of neurons is based on the output of each neuron from the training patterns [2]. The removal of neurons is usually done in a two step process. Firstly any neuron which has outputs which are close to zero on all training patterns is removed [2].

Secondly the outputs are further inspected to find properties of each neuron that mean it should be removed. One of these properties that is used is termed *distinctiveness*[2].

The *distinctiveness* of a hidden neuron is the angle between another hidden neurons unit output. Specifically it is defined as the angle between two hidden neurons, where each hidden neuron's vector is the output from each training pattern. That is for each neuron, construct a vector with the dimensions equal to the number of training patterns, and where each value is equal to the output of the neuron for that training pattern. Using these vectors the angle between each pair of hidden neurons can be calculated and compared.[2]
The table below shows an example of 16 training patterns, with 6 hidden neurons [2].

			Hidden Neuron					
Pattern	1	2	3	4	5	6	Result	Target
0	0.330	1.000	0.910	0.582	0.593	0.381	0.000	0.000
1	0.091	0.994	0.229	0.101	0.834	0.839	0.000	0.000
2	0.098	0.994	0.348	0.130	0.874	0.868	0.000	0.000
3	0.022	0.488	0.026	0.012	0.960	0.982	0.025	0.000
4	0.094	0.994	0.325	0.128	0.853	0.849	0.000	0.000
5	0.021	0.489	0.024	0.012	0.952	0.979	0.025	0.000
6	0.023	0.488	0.025	0.016	0.965	0.984	0.025	0.000
7	0.005	0.006	0.001	0.001	0.990	0.998	0.805	1.000
8	0.117	0.994	0.339	0.132	0.875	0.870	0.000	0.000
9	0.026	0.488	0.025	0.012	0.960	0.983	0.025	0.000
10	0.029	0.488	0.026	0.016	0.971	0.986	0.025	0.000
11	0.006	0.006	0.001	0.001	0.991	0.998	0.805	1.000
12	0.027	0.489	0.24	0.016	0.965	0.984	0.025	0.000

13	0.006	0.006	0.001	0.001	0.990	0.998	0.805	1.000
14	0.006	0.006	0.001	0.002	0.992	0.998	0.805	1.000
15	0.001	0.000	0.000	0.000	0.998	1.000	0.816	0.000

If the angle between two neurons is less then approximately 15 degrees, these neurons can be seen as highly similar to each other, and therefore one of them can be removed. The weight vector of the neuron removed is added to the weight vector of the other neuron. [2]

If the angle between two neurons is greater then or equal to 165 degrees then the two neurons can be viewed as being complimentary to each other, and therefore both neurons can be removed.[2]

Below is a table showing the angles calculated between the 6 hidden neuron's output vectors from the table above.

Pair of units	Vector Angle
1 2	81.8
1 3	25,2
1 4	8,4
1 5	176.5
1 6	169.9
2 3	63.0
2 4	77.8
2 5	100.8
2 6	103.7
3 4	18.0
3 5	157.7
3 6	163.7
4 5	173.7
4 6	177.5
5 6	7.3

Since the sigmoid activation function is used, the output values for each neuron is limited to between 0 and 1. Calculating the angle between these vectors would therefore be limited to the range 0-90 degrees. To overcome this the vectors are first calculated in respect to the origin (0.5,0.5) to obtain angles between 0-180 degrees.[2]

Using this method to remove neurons should allow the network to reduce its size without majorly affecting performance, and requiring only minor retraining [2].

2 Example Study

To use the network reduction technique highlighted above the *MNIST Hand Written Digit Data Set* [1] was used. This data set contains images of hand written digits, with the aim to correctly classify the digit that is being represented.

2.1 Data

The data set contains 70,000 samples of hand written digits, with 60,000 being used as training data and 10,000 used for testing. Each sample is a preprocessed sample taken from the original *NIST Hand Written Digit Data Set*, in which each sample is contained in a 28 x 28 pixel image [1]. Each pixel in the image has a value between 0 and 255.

2.2 Data Pre-processing

Each pixel within each sample was preprocessed to be within the range of 0-1, using min max scaling, with a maximum of 255 and minimum of 0.

2.3 Convulotional Neural Network Structure

The convolution neural network structure used resembles closely to the LeNet-5 structure. In this architecture there exists 7 layers, not counting the input. The first layer is a convolutional layer which outputs 6 x 28 x 28 feature maps, taking as input a 1 x 28 x 28 hand written digit. This is then passed to the sub-sampling (max-pool) layer to create 6 x 14 x 14 feature maps. This is then passed to the next convolutional layer which outputs 16 x 10 x 10 feature maps. Again this is passed to another sub-sampling (max-pool) layer to create 16 x 5 x 5 feature maps. This is then passed to a fully connected layer with 120 hidden neurons. The next layer is a fully connected layer with 84 hidden neurons. This is then passed to the output layer with 10 neurons. Below is a figure showing a similar LeNet-5 setup used for hand written digit recognition.

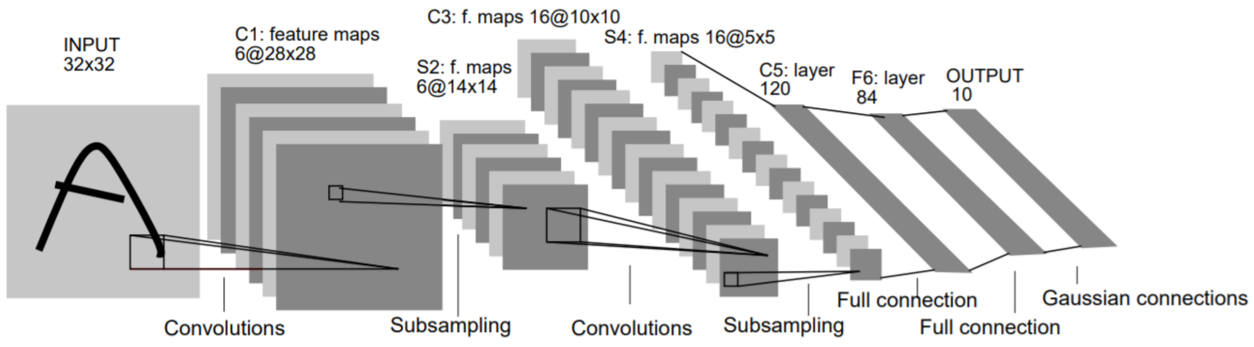

Figure 1: LeNet-5 CNN For Hand Written Digit Recognition [4]

In the implementation of the LeNet-5 structure there exists a fully connected layer with 400 hidden neurons, sitting between the last sub-sampling (max-pool) layer and the first fully connected (120 hidden neuron's) layer. This was needed as during implementation it was not possible for 400 features (16 x 5 x 5) to be passed to a fully connected layer with 120 neurons.

For the last hidden fully connected layer (84 hidden neurons) it was decided to increase this to 110 neurons, as the aim of this paper is to view the effectiveness of neuron pruning, and 110 was found to be the maximum number of neurons that could be used without sacrificing accuracy.

The sigmoid activation function was used for each fully connected layer. The soft max activation function was applied to the output layer (10 neurons) in order to determine the prediction made for the given input (being the maximum probability from the 10 neurons).

2.3 Experiments

To test the accuracy of the network model and compare to other results, the hold out method was used, where the neural network is trained on the 60,000 training samples, and the accuracy is then calculated based on the 10,000 testing samples. Each experiment was run 10 times, and the values averaged.

The network training was performed using mini-batch (50 samples per batch) ADAM gradient descent. A learning rate of 0.001 was used. The training phase repeated until a predefined halting condition was meet. The accuracy was determined to be on average 98.47%. Below is the confusion matrix for one of the trained networks on the test set.

Digit	0	1	2	3	4	5	6	7	8	9
0	973	0	0	0	3	0	2	2	0	0
1	0	1131	1	0	0	1	1	0	1	0
2	1	1	1018	1	3	0	0	6	2	0
3	0	0	2	996	0	5	0	3	2	2
4	0	0	0	0	975	0	0	1	0	6
5	2	0	0	9	0	874	2	0	2	3
6	5	2	0	0	6	3	941	0	1	0
7	0	2	5	0	0	0	0	1015	1	5
8	5	1	2	3	4	3	1	4	948	3
9	0	4	0	2	8	3	0	10	6	976

2.4 Neural Network Pruning

Initially an attempt was made to apply neural network pruning to the model on the last fully connected layer (110 Neurons). However this experiments failed to prune any neurons. It was found that the approximate maximum and minimum angles were 150 and 40 respectively. Based on this experiments were run to see changing the thresholds of the minimum and maximum angles required for the removal of neurons would led to an increasing reduction in the neural network, with an expected reduction in accuracy. To perform this initial thresholds of 40 (for neurons to be deemed identical) and 150 (for neurons to be deemed complimentary) were set. These thresholds were then increased or decreased by 15 degrees (decreased for complimentary and increased for identical).

The neural network pruning was applied to a fully trained neural network (deemed by the early stopping criteria). After each neuron or pair of neuron's was removed the network was trained for one epoch on the entire training set.

Minimum Degree Threshold (Similarity)	Maximum Degree Threshold (Complimentary	Number of Neurons Before Pruning	Accuracy Before Pruning	Minimum Number of Neurons After Pruning	Maximum Number of Neurons After Pruning	Accuracy After Pruning
15	165	110	98.46%	110	110	98.46%
40	165	110	98.48%	108	110	98.47%
40	150	110	98.48%	104	107	98.52%
40	135	110	98.43%	94	100	98.45%
55	135	110	98.35%	51	64	97.9%
70	135	110	98.42%	14	22	92.36%

2.4 Analysis of Results

Firstly the accuracy of the network without pruning is compared. The average accuracy of the network was seen to be 98.47%. This is compared to the results in [3], which achieved an accuracy of 99.746%. In this paper a variation on a convultional neural network layer is defined and used, called relaxed convolutional layer's. This combined with a more advanced technique for modifying the learning rate could in part explain the difference in accuracy[3].

The neural network pruning experiments show that an increase in the threshold's lead to more neurons being pruned, which is expected. The accuracy of the neural network can be seen to sometimes have a minor increase with the removal of hidden neurons (using threshold's 40 – 150 and 40-135), which may be caused by the additional training of the network when a neuron is removed. It is noted this may be a sign that the network model was not fully trained.

It can be seen a significant increase in the threshold's was required to achieve any significant pruning (using 55 – 135), which is unexpected. Since it is already known the optimal number of hidden neurons is 84 in this layer, it was expected that adding additional neurons (which should be redundant) would lead to a network structure that is at least close in range to 84. However it was found that using these thresholds did lead to relatively successfully pruning (a minimum reduction of 42% (110 to 64), with only a minor loss in accuracy (98.35% to 97.9%)).

It can be seen that exceeding this threshold (55 – 135) led to a significant reduction in the number of neurons (minimum reduction of 80%) whilst leading to a significant reduction in accuracy (98.42% to 92.36%).

3 Conclusion and Future work

A convolutional neural network capable of classifying a hand written digit within the MNIST Hand Written Digit Data Set was constructed. It was then shown that using the network pruning property *distinctiveness* could be used to prune this trained neural network. It was shown that modifying the threshold's used to determine when neuron's should be removed could be used to increasing prune the network, with a trade off in accuracy.

Further work could include the use of further neural network pruning properties, such as *badness* [2]. This could be used to remove fully connected layer's within a neural network. Experiments could also be done to prune the convulotional layers within the neural network. In this way pruning would be done to the filter's within a convolutional layer, using an algorithm termed *try and learn* [5], which similarly can be used to prune the convolutional layer, leading to a trade off between network topology and accuracy.

Other future work could include seeing if the number of hidden neurons left after pruning the network could be used to find the optimal or near optimal number of hidden neurons in a similar or even the same problem.

4 References

[1] MNIST Database, “http://yann.lecun.com/exdb/mnist/”
[2] Gedeon, TD, Harris, D, Network Reduction Techniques, In: Proc. Int. Conf. on Neural Networks Methodologies and Applications, AMSE, San Diego, vol. 2, pp. 25-34, 1991.
[3] Chunpeng, W, Wei, F, Yuan, H, Jun, S, Satoshi, N, Handwritten Character Recognition By Alternatively Trained Relaxation Convolutional Neural Network, In: 14th International Conference on Frontiers in Handwriting Recognition, 2014.
[4]Lechun, Y, Bottou, L, Bengio, Y, Haffner, P, Gradient-Based Learning Applied to Document Recognition, In: Proc. Of The IEEE, 1998.
[5] Huang, Q, Zhou, K, You, S, Neumann, U, Learning to Prune Filters in Convolutional Neural Networks, 2018, “https://arxiv.org/pdf/1801.07365.pdf”