Network Reduction on Neural Network Classifier

Helai Bai

Research School of Computer Science, The Australian National University
u6341820@anu.edu.au

Abstract: In recent years, the application of multilayer neural network has become widespread in the field of computer vision, replacing its traditional (hand-crafted) features and classifiers. While maintaining high efficiency, the neural network models are continuously approaching the accuracy limit and even exceed human beings. However, while the improvement of accuracy that multilayer models have achieved, their depth and size are also growing at a significant fast pace, which leads to both computationally intensive and memory intensive. In this case, it is necessary to develop some network reduction techniques to compress the number of parameters and improve the computational efficiency of those models. In this research, we use deep neural network classifier to make classification and explore a network reduction technique which based on Distinctiveness to reduce the number of parameters. We firstly establish a simple artificial neural network (ANN) on a simple ionospheric dataset and prune it. Then, we will extend the simple ANN to a deep convolutional neural network (CNN) on a complex image classification dataset, MNIST, and examine the performance of this network reduction technique on CNN. The experimental result shows that the ANN maintains its original accuracy after pruning without retrain. And, the proposed CNN maintains the same level of accuracy at about 95% but has 70% reduction in the number of parameters. Finally, we will compare our result with a mature CNN model (Ciregan, Meier & Schmidhuber, 2012).

Keywords: Convolutional Neural Network, MNIST, Network Reduction, Distinctiveness.

1 Introduction

Deep neural network has widespread applications in the field of computer vision. With high performance of computational efficiency, deep neural networks have achieved a great success by approaching the accuracy limit and even exceed human beings. One typical application is using CNN on classification task of distinguishing handwritten digits, and the MNIST dataset is the most canonical carrier to test the performance of classifiers. In this case, we will apply CNN on MNIST dataset in order to get more comparable results.

However, while the DNN models widely apply in many practical areas, their depth and size are also growing at a significant fast pace, making them difficult to deploy on embedded systems with limited hardware resources. Even if transmitting over the network, the high bandwidth consumption is daunting for many users. On the other hand, large-scale models also pose enormous challenges to equipment power consumption and operating speed. In this case, developing smaller models is crucial for further application of CNN models, especially for the products with limited hardware resources and need updates frequently. In this case, many network reduction techniques have been developed, and many of them have shown a satisfiable ability in practice.

In this research, we apply a network reduction technique that reduces the number of parameters of the network by pruning neurons with the Distinctiveness (Gedeon & Harris, 1991) criterion. Firstly, we examine this technique by pruning on a simple ANN in order to prove its feasibility. Then, we will apply this technique to a CNN model, as a classifier to distinguish handwritten digits in MNIST dataset. As it is crucial about the network size, we first consider the architecture and hyper-parameters of the models, such as the number of neurons in hidden layers of ANN and the activation and pooling functions of CNN, the number and order of convolutional layers and neurons in the fully connected layer. Finally, we discuss the results of the final implementation of our CNN and compare the results to a mature CNN model.

2 Method

2.1 Dataset and Pre-processing

2.1.1 Ionospheric Dataset

The ionospheric dataset used in this study were collected by the Space Physics Group of The Johns Hopkins University Applied Physics Laboratory, which located in Goose Bay, Labrador. In details of the data, there were 17 pulse numbers for the system which were described by two attributes respectively, where all 34 attributes are continuous, and the 35th attribute is the target which represents whether this data sample is suitable for further analysis (“good”) or not (“bad”). Obviously, this is a binary classification task. Generated by radar and collected from the natural world, the original data in this dataset are "dirty", incomplete, redundant and fuzzy, which cannot meet the requirements of this artificial neural network directly. Hence, we need to preprocess the data to be clean, accurate and concise, which is easier to be fed into the network and, therefore, improving the final performance. As this is a binary classification task, we transform the target value “g” and “b” into 0 and 1 respectively.

Standardization: Center to the mean value and component-wise scale to unit variance. After transforming, all attributes in the same dimension have zero mean and unit variance. Its calculation method is that taking eigenvalues minus the mean then divides by the standard deviation.

Normalization: Scale input vectors individually to unit norm. The p-norm of each sample is calculated, and then each element in the sample is divided by the norm, and the result of this process is that the p-norm (l1-norm, l2-norm) of each processed sample is equal to 1. In this study, I used l2-norm.

2.1.2 MNIST Dataset

The MNIST (Mixed National Institute of Standards and Technology database) comes from the National Institute of Standards and Technology (NIST). The training set consists of handwritten digits from 250 different people, of which 50% are from high school students, and 50% are from the Census Bureau staffs. The test set is the same proportion of handwritten digits. It contains 60,000 training images and 10,000 testing images (Kussul & Baidyk, 2004). Images are in the grayscale format with 28x28 pixels of each image and are stored in bytes. Each image has a corresponding label, which is the number corresponding to the image.

Just like what Ciregan, Meier and Schmidhuber did, we normalize the digit width to 8, 12, 16, 20, 24 pixels to create another five datasets, which is like seeing the data from different angles model (Ciregan et al., 2012). By training five CNNs per dataset, it creates a 30 columns hierarchical Multi-Columns CNN architecture. **Figure 1** shows the architecture of this hierarchical model, where the input image can be processed into n blocks. Each column was trained in different ways for per block and their predictions are averaged. Compared to the simple CNN model with only one columnn, this hierarchical model could achieve higher predicting accuracy.

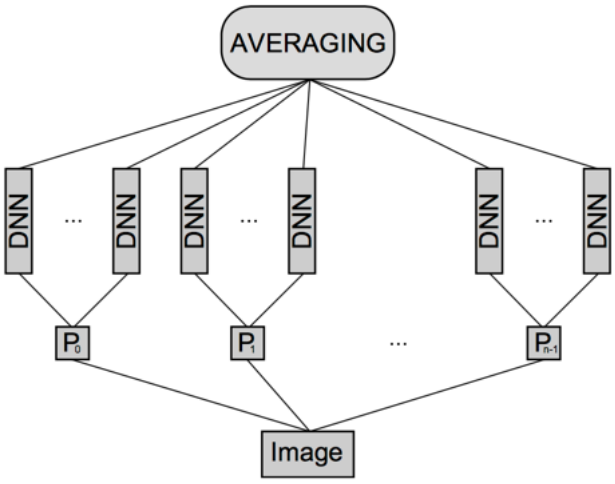

Fig. 1. Model Architecture of MCCNN

However, the experiment result shows that this multi-column CNN needs too much training time, which was extended from minutes to hours. As in this research, our primary aim is just to explore the performance of the pruning technique with the Distinctiveness criterion, thus our final result is a simple CNN with only one column. Besides, for the same reason, we use just 1000 samples (out of 60000) as training set and another 1000 samples (out of 10000) as the test, which is way faster than training on the entire dataset, reducing to less than 1 minute, with only 5 per cent accuracy lower.

2.2 Model Design

2.2.1 ANN Model architecture

As the primary purpose of using ANN is to examine the feasibility of our network reduction technique, we simply establish a two-layer ANN which has only one hidden layer. The neural network is feed-forward and trained by back-propagation. Regarding each hidden neuron in the network as a perceptron, the activation function is Rectified Linear Unit Function, and the output of hidden layer is linear. We applied mini-batch training and stochastic gradient descent to optimize the parameters. Besides, the loss function was generated by cross-entropy. we choose 15 hidden neurons and set up the initial weights of the synapse as random values from -1 to 1 but not all 0.

2.2.2 CNN Model architecture

Initially, we want to follow what Ciregan, Meier and Schmidhuber did to establish a hierarchical multi-column CNN architecture. Normalized the digit width to get six columns, each column, which is a deep CNN that generated from a basic CNN model, was trained in five different ways per block and their predictions are averaged. However, training this complex model is so time-consuming that makes it impractical to running on a single personal laptop with instance feedback. In this case, as our primary purpose of this research is just to examine a network reduction technique, we simply apply this technique on a simple CNN model.

For the simple CNN model, it has two convolutional layers and one fully connected layer. Each convolutional layer consists of one convolutional layer with kernel size 5x5, one activation function and one pooling layer with kernel size 2x2. In particular, we choose ReLU as activation function and max-pooling as pooling method.

2.2.3 Activation Function and Pooling Method in CNN

In this research, we use ReLU as the activation function and max-pooling as the pooling method in the pooling layers. We also have tested the performance of different activation functions (Sigmoid) and pooling methods (Average-Pooling). The result shows that the combination of ReLU and Max-Pooling is the best.

Max-Pooling (1) and Avg-Pooling (2) are defined as:

$$\text{out}(N_i, C_j, h, w) = \max_{m=0}^{kH-1} \max_{n=0}^{kW-1} \text{input}(N_i, C_j, \text{stride}[0] * h + m, \text{stride}[1] * w + n) \tag{1}$$

$$\text{out}(N_i, C_j, h, w) = \frac{1}{(kH * kW)} * \sum_{m=0}^{kH-1} \sum_{n=0}^{kW-1} \text{input}(N_i, C_j, \text{stride}[0] * h + m, \text{stride}[1] * w + n) \tag{2}$$

To analyse this issue more profoundly, we have done a small searching but failed to find any research which gives a satisfying explanation. In this case, we will just try to explain this phenomenon in the result part in an intuitive way.

2.2.4 Loss Function

We use the cross-entropy loss as the loss function, which measures the 'distance' between what the neural network's belief of the output class is, and what the actual target class is, which more clearly describes the distance between the actual model and the ideal model. The cross-entropy is defined as:

$$\text{loss}(x, \text{class}) = -\log \frac{\exp(x[\text{class}])}{\sum_j \exp(x[j])} = -x[\text{class}] + \log(\sum_j exp(x[j])) \tag{3}$$

2.2.5 Optimiser

In this research, we use Adam (adaptive moment estimation) as the optimizer, which is an algorithm for first-order gradient-based optimisation of stochastic objective functions, based on adaptive estimates of lower-order moments (Kingma & Ba, 2014). The Adam algorithm dynamically adjusts the learning rate for each parameter based on the first moment estimation and second-moment estimation of the gradient of the loss function for each parameter. Adam is also based on a gradient descent method, but the learning step size for each iteration parameter has a definite range. It does not result in a large learning step due to a large gradient, and the value of the parameter is relatively stable. It does not require stationary objective, works with sparse gradients, naturally performs a form of step size annealing.

2.2.6 Evaluation

We use K-fold cross-validation in order to evaluate the generalisation ability with different initialisation setting of the model so as to set up the model. The initial dataset is divided into K subsets, a separate subset is retained as the data for the validation model, and the other K-1 sets are used for training. The cross-validation is repeated K times. Each subset is verified once, then average the result of K times.

2.3 Network Reduction

Several studies focus on diminishing the network size and stimulating the computational efficiency. Many studies paid considerable attention to develop the network pruning techniques, introducing the sparsity by prune neurons and channels, which could give approximately 8x reduction in storage requirement (Liu et al., 2017). Many criteria of pruning are developed as well, such as rank-based, relevance, distinctiveness and contribution. Additionally, the technique that quantising the weights and weight sharing also helpful, which remarkably reduce the number of parameters that acquired in convolutional layers, and give 5x reduction (Han et al., 2015). Besides, by applying some encoding techniques like Huffman Coding, the storage size could further shrink another 20% (Han et al., 2015).

A "super-convergence" techniques, based on adjusting learning rates, has an order of magnitude faster than with standard training methods (Smith & Topin, 2018). Besides, the hardware is also an excellent choice for improvement, and an FPGA-based hardware acceleration framework EIE has been developed (Han, Liu, Mao, Pu, Pedram, Horowitz& Dally, 2016) which gives another order of magnitude improvement.

2.3.1 Relevance, Contribution, Sensitivity and Similarity

The property of *relevance* (Mozer & Smolenski, 1989) is estimated by comparing how well the network does with the unit in place, versus the situation if the unit was removed. The property of *contribution* (Sanger, 1989) is the product of the activation of the hidden unit and the weight from the hidden unit to the output unit. The property of *sensitivity* (Karnin, 1990) of the global error function to the removal of a unit can be done by recording the incremental changes to synaptic weights during an epoch of back-propagation. Also, the property of Similarity of hidden units is determined from the unit output activation vector over the pattern presentation set.

Excess units are units with high relevance which can be eliminated into one, or with the low contribution which can be removed, or with low sensitivity or with high.

2.3.2 Distinctiveness

In this study, I applied a pruning technique which was partly introduced in Gedeon and Harris's work, and I used distinctiveness as criteria. In order to calculate the *distinctiveness* between two neurons, we first extract their weight vectors and use these two vectors as variables. We can calculate the distinctiveness of pairs of vectors by calculating the included angle between them in pattern space (Gedeon & Harris, 1991). The angular range is from 0 to 180 degrees. Units with angular less than 15 degrees are considered sufficiently similar, and units with angular more than 165 degrees are considered complementary, which all excess units.

When filtering out the excess units, units with angular less than 15 degrees were eliminated into one which the bias and weights vector of the removed one were added on that of the remaining one. Also, units with angular more than 165 degrees were all removed (Gedeon & Harris, 1991). It must be noted that, these particular angles are not fixed and depends on the structure of actual model.

As this pruning operation aims to reduce the size of the network by merging similar units and removing excess units, besides, keep the performance after reduction, the pruning operation should keep the contribution of each unit as much as possible. Therefore, in details about updating the weights, three parts might need to be considered. Part 1, the weight vectors from prior layer to the target units. Part 2, the bias values of target units. Part 3, the weight vectors from target units to posterior layer.

When using Sigmoid as the activation function, we have to normalise the vectors before calculating angular. As logistic sigmoid squashes all activations between 0 and 1, it has a 'centre' 0.5. Therefore, if we want to scale the range of angular to 0-180 degrees, we can simply subtract 0.5 element-wise from all weights.

2.3.3 Layer

In the ANN model, as hidden layers contribute the most parameters, we will apply the network reduction technique to prune neurons in hidden layers. Additionally, in the CNN model, the number of parameters in the convolution layers of the first few layers is small, but the computational proportion is enormous; while the full-connection layer behind is just the opposite, most CNN networks have this feature. Therefore, we focus on the convolutional layer when performing computational acceleration optimisation and focus on the fully connected layer when performing parameter optimisation and weight clipping.

2.3.4 Algorithm

Firstly, we have to extract the weight vectors of all neurons in the target layers and calculate the included angular between each pair of vectors in each layer. Secondly, based on the Distinctiveness criterion that described above, it is easy to filter out the excess units and divide them into two sets, removing set and eliminating set. Thirdly, eliminating or removing one pair of neurons from what we get above. It is worth mention that elements in the two sets may overlap and we might lose some elements in eliminating set while we are removing elements in removing the set. In this case, we have to deal with eliminating first. Then, calculate the included angular again of the remaining pairs. Finally, repeat this step (removing or eliminating a pair then re-calculate) until no element is filtered out in both sets.

The steps above show what the pruning algorithm do in one processing iteration. We will retrain this pruned model and apply the pruning algorithm again to form another processing iteration. These iterations will run for several times until the whole algorithm converges.

At the specific technical level, if we want to "kill" a neuron from the network, we have to isolate them from other neurons in case affect the training of remaining neurons in the retraining process. That is to say, and we have to stop them taking part in the backward process. There are two ways to "kill" a neuron in the network. One method is to set the "requires_grad" attribute to false of those neurons, which tells the PyTorch to not calculate their gradient during the backward process. Another method is to create a mask matrix that consists of 0s and 1s, which is multiplied by their gradients during the backward process. If one neuron is "killed", then we change the corresponding digit to 0 in the mask matrix.

3 Result and Discussion

Implemented on Python, the network was built basically based on the package PyTorch in this study. We first apply an ANN model in ionosphere dataset to classify "good" or "bad" and then apply a CNN model on MNIST to recognise of handwritten digits. Firstly, by examining the performance of different model architectures and inner structures, we set up the ANN and CNN models with proper initialisation. Then, data from datasets was preprocessed and fed into the neural network. After that, identifying excess units in the target layers by some criteria and eliminating or removing them, the network was adjusted and produced a reduced network at the same accuracy level but with less number of parameters.

3.1 Pruning Neurons in ANN

The ANN model gets 99.65% accuracy on the training set and 96.92% on the test set. Applying the pruning technique which introduced in the previous part, we extracted the parameters in the hidden layer and calculated the included angles between the weight vectors of every two hidden neurons in the same layer, by the distinctiveness as the criterion. In particular, the angles are shown in **Figure 2**.

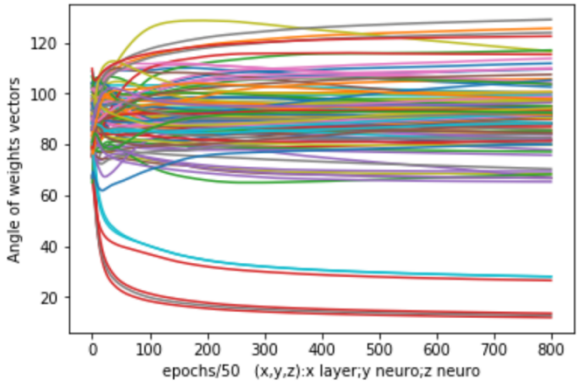

Fig. 2. Included angles of every two weights vectors in the same hidden layer during the whole training.

In this figure, each line represents an included angular of one pair of weight vectors. From the graph, we can find that most lines are gathering together in the range of 60 to 120, but there are a few lines that continue decrease to under 40 degrees. In particular, there are three lines drop to less than 15 degrees and these are the pairs what we intended to filter out.

Filtering out included angles which less than 15 degrees or more than 165 degrees. We have the result that shows in **Table1**.

Table 1. Included angles of weight vectors which less than 15 degrees or more than 165 degrees.

No. layer	No. neuron1	No. neuron2	Angle
1	1	5	11.948781055260131
1	1	9	12.806529576465282
1	5	9	13.543175426417788

In particular, the vector pairs that less than 15 degrees should be eliminated into one unit, by adding the weights of the removed unit to the remained unit, and vector pairs more than 165 degrees should be both removed. Augmenting the weights of neuron No.5 to neuron No.1 on all weight vectors that connect from units in the previous layer to those two units and bias of themselves (like part 1 together with part 2) respectively, the testing accuracy remained 96.92%. Calculating the included angles again, there was still another included angle that filtered out which is "(1,1,9) 11.760503839566276". Pruning again and got the testing accuracy for slightly decreasing to 95.38%. Filtering the included angles one more time, there were no vector pairs that filtered out, which finishes the pruning procedure. The operation described above is designed to automatically.

According to the result above, we can find that during pruning operation, the results of testing accuracy remained stable which fits the discussion we made in previous part. This gives the evidence of the feasibility of our pruning technique.

3.2 Activation Function and Pooling Method in CNN

As it is mentioned above, we use ReLU as the activation function and max-pooling as the pooling method in the pooling layers in this research. And, we also have tested the performance of different combinations of activation functions and pooling methods. The result (**Table 2**) shows that the combination of ReLU and Max-Pooling is the best.

Table 2. Testing accuracy with different combination of activation functions and pooling methods

	ReLU	Sigmoid
Max-pooling	96.70	87.15
Avg-pooling	94.60	90.15

To analyse this issue more profoundly, we here just try to explain this phenomenon in the result part in an intuitive way. The simplest explanation is that, on the one hand, its performance is excellent, on the other hand, many experts use it, so everyone follows this way. However, if we concentrate on the difference between these functions and methods, we will get some straightforward understanding.

Max-Pooling aims to filter. If there is a conforming feature in a certain area, that is, it exists, then this information is retained and delivered. Avg-Pooling aims to extract commonalities. Using Avg-Pooling can enhance the similarities, reduce the difference, and better retain the inner relationship of information. Spatially, Avg-Pooling is linear, and Max-Pooling can be considered nonlinear.

From another perspective, Max-pooling uses the maximum value to represent the original region, while Avg-pooling uses the average value. However, the average value is only a statistically significant value, not the actual value of any single pixel of the original region (or the single point of the output of the previous layer), which easily leads to the alienation of the output result. On the other hand, the correlation between the maximum value and the edge information is higher than the correlation between the mean value and the edge information.

The pooling layer has two main functions. One is to remove redundant information, and the other is to retain feature information of the feature map. In the classification problem, we need to know what objects are in this image, but not much concern about where the objects are located. In this case, it is apparent that max pooling is more suitable than the average pooling. Besides, in areas where the network is relatively deep, features are sparse. If we want to delivery sparse features, choosing the largest value from the region is better than passing the average value.

3.3 Pruning Neurons in CNN

In this research, we use distinctiveness as the criterion to recognise excess units. As we were aiming to examine the performance of the pruning technique with this criterion, we use an ordinary CNN model as the testing model which has two convolutional layers and one fully connected layer. According to the reason that we have discussed in section 2.3.3, our pruning technique will apply to the fully connected layer.

Following the model design in section 2.3.4, we were firstly testing the pruning algorithm's performance in one processing iteration. In this case, we run our algorithm on the CNN model, by setting the excess domain as less than 30 degrees or over 150 degrees, which filtered out 74 pairs in removing set and 329 pairs in eliminating set. Carrying out the pruning technique, it shows that after nearly 400 iterations of pruning and re-calculating, the network maintains its prediction performance at the same accuracy level with only slightly reduced. This could valid the feasibility of our pruning technique. The oscillation of the accuracy along pruning is shown in **Figure 3** and **Figure 4**. After that, we should retrain the outcome network in order to redeem the loss while pruning neurons. The accuracy slightly improves from 93.3% to 93.4%. However, this improvement is not guaranteed in each iteration, but the outcome model that retrained will maintain the same accuracy level comparing to the original one.

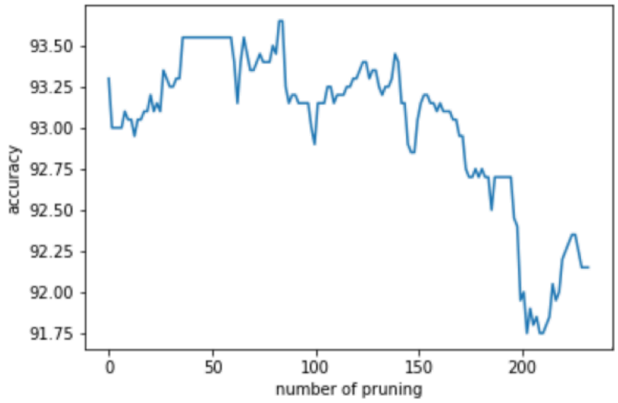

Fig. 3. Predicting accuracy after number of epochs.

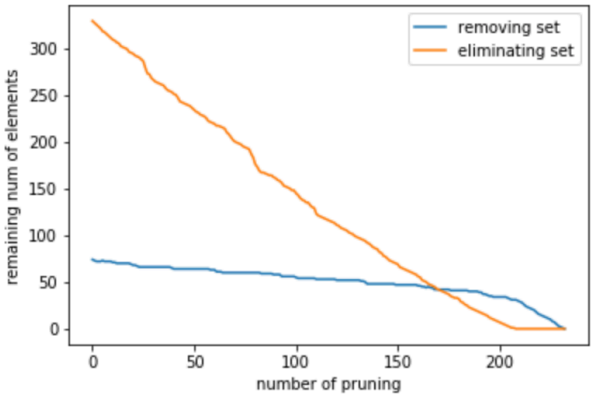

Fig. 4. Numbers of elements in the removing set and eliminating set.

As we have proved the feasibility of our pruning technique, it is time to validate the authenticity of the whole algorithm which retrains the model after pruning. However, applying the whole algorithm on a CNN model will be extremely time-consuming, as the angular of each pair of weight vectors has to be recalculated in each iteration. This is a formidable challenge to most PC devices even if training a small CNN model with only two convolutional layers and one fully connected layer. In our CNN model, there are 784 x 10 connections in the fully connected layer which gives more than 300,000 pairs, and this makes it entirely impractical to test without the help of high-performance computing machine.

Owing to the reason above, we here just test our algorithm in a brute way. For each processing iteration, we will simply remove and eliminate all neurons that filtered out into removing set and eliminating set without recalculating their angular. This could significantly reduce the calculations of included angular and provides instant feedback of the performance. Results are shown in **Figure 5, Figure 6** and **Figure 7**.

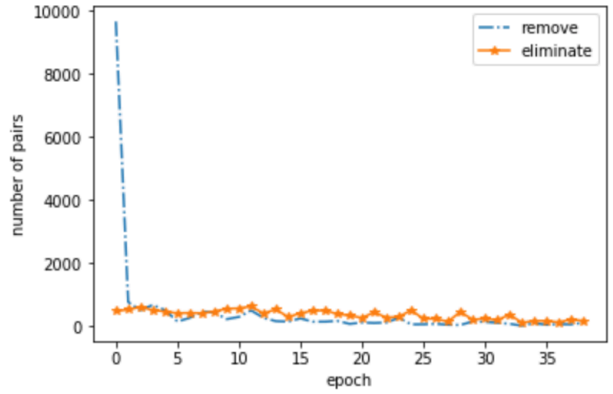

Fig. 5. Numbers of elements in the removing set and eliminating set.

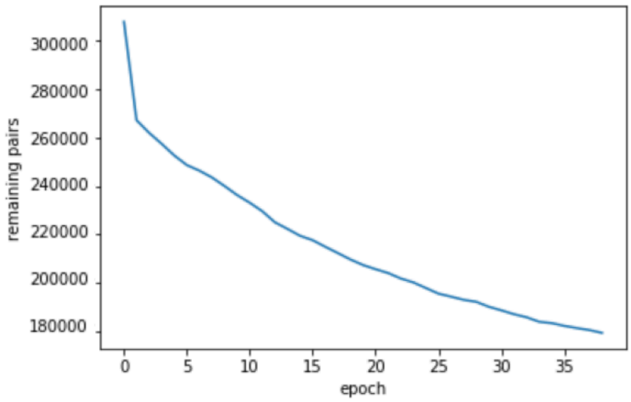

Fig. 6. Numbers of remaining connections in the fully connected layer during the pruning and retraining process.

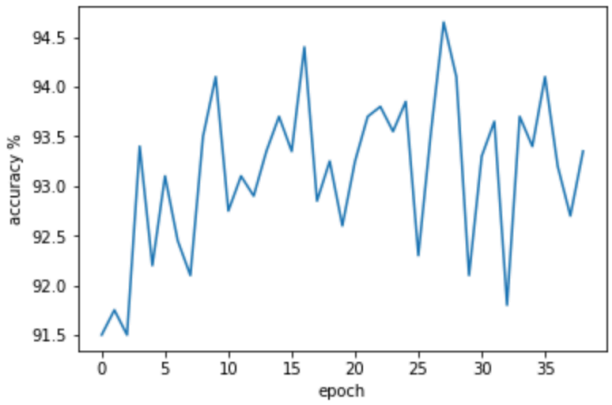

Fig. 7. The oscillation of accuracy during the pruning and retraining process.

Figure 5 illustrates the variation of the remaining number of elements in the removing set and eliminating set respectively. We can find that, initially, there are significant number of elements in removing set, which is owing to the randomly assigned weights in the original network. After a few epochs, the number of elements in both set reduced to a very low level as most excess units have been removed from network. If we proceed this testing, a dramatic dropping will appear when the number of remaining pairs reduces to about 40,000, which represents approximately 70% reduction of number of parameters. **Figure 6** demonstrates the number of remaining pairs in the network, which shows that our algorithm has reduced approximately 20% neurons from the network just in 40 iterations, even the algorithm has not converged. In addition, **Figure 7** indicates that the prediction performance has slightly increased. It must point out that, this result is just under the condition of applying a brute method, which represents that the delicate algorithm will generate a better performance.

3.4 Comparison of Result with Comparative Study

In the research of Ciregan, Meier and Schmidhuber in 2012, they established a multi-column deep neural network (MCDNN) for image classification, which first achieves near-human performance on MNIST handwriting benchmark. By converting the original 20-pixel-width images to 10, 12, 14, 16, 18, 20 pixels and forms six new datasets in order to see the data from different angles model (Ciregan et al., 2012). Training each dataset with five different deep neural networks and averages their output to get the final prediction. Their model was trained 14 hours with all 60,000 training samples and reduced the error rate to only 0.23%.

Comparing to the result of this research, it is no doubt that our result is weaker than Ciregan's. However, our model is much simpler than theirs, and we used only 1,000 samples as the training set to get 96% of accuracy by less than 1 minutes training. Besides, we have proven the network reduction capability of our algorithm on the CNN model.

4 Conclusion and Future Work

In this study, we have firstly applied a simple artificial neural network on an ionosphere dataset to examine the feasibility of the pruning technique that uses distinctiveness as the criterion, which gives a satisfying result to our study. Then, we have applied the deep convolutional neural network technique to solve a classification task on MNIST dataset which aims to recognise the actual number of handwritten digits and devised a simple procedure that automatically identifies and removes the excess units in fully connected layers of the CNN model. We still use distinctiveness as the criterion of recognising excess units, by calculating the included angular of weight vector pairs. Once we filter out the excess units, they are isolated from the network by creating a mask matrix in the backward process. After that, the model will be retrained, and this whole procedure will be carried out several iterations until the algorithm converged.

Our experimental results prove the authenticity of our algorithm. By applying this network reduction technique, we could reduce approximately at least 70% of parameters in the fully connected layer but maintain the same accuracy level comparing to the original.

For future research, we could extend our reduction technique onto the convolutional layers. Although the number of parameters in the convolutional layer is small, it takes the majority of computational proportion. Reducing the number of parameters in the convolutional layers could improve the computational performance of the model. Another potential point of future research is the techniques to compression the size of the model by applying techniques such as encoding methods like Huffman Coding and quantising the weight. For further improving the prediction accuracy, complex models like what Ciregan did are helpful. Some other preprocessing techniques such as outlier deduction could promote the development of these studies.

References

Ciregan, D., Meier, U., & Schmidhuber, J. (2012, June). Multi-column deep neural networks for image classification. In Computer vision and pattern recognition (CVPR), 2012 IEEE conference on (pp. 3642-3649). IEEE.

Gedeon, T.D., & Harris, D. (1991): Network reduction techniques. In Proceedings International Conference on Neural Networks Methodologies and Applications. Vol. 1, pp. 119-126

Han, S., Liu, X., Mao, H., Pu, J., Pedram, A., Horowitz, M. A., & Dally, W. J. (2016, June). EIE: efficient inference engine on compressed deep neural network. In Computer Architecture (ISCA), 2016 ACM/IEEE 43rd Annual International Symposium on (pp. 243-254). IEEE.

Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

Karnin, E. D. (1990). A simple procedure for pruning back-propagation trained neural networks. IEEE transactions on neural networks, 1(2), 239-242.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kussul, E., & Baidyk, T. (2004). Improved method of handwritten digit recognition tested on MNIST database. Image and Vision Computing, 22(12), 971-981.

Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., & Zhang, C. (2017, October). Learning efficient convolutional networks through network slimming. In 2017 IEEE International Conference on Computer Vision (ICCV) (pp. 2755-2763). IEEE.

Mozer, M. C., & Smolensky, P. (1989). Using relevance to reduce network size automatically. Connection Science, 1(1), 3-16.

Smith, L. N., & Topin, N. (2017). Super-Convergence: Very Fast Training of Residual Networks Using Large Learning Rates. arXiv preprint arXiv:1708.07120.

Sanger, D. (1989). Contribution analysis: A technique for assigning responsibilities to hidden units in connectionist networks. Connection Science, 1(2), 115-138.