D06943020 鄧傑方

AVLSI Final Project Report

Network Structure Pruning

I. Background

For the huge neural network, deep compression, such as pruning and quantization, are the key techniques for speed up and energy efficiency. The target of network structure pruning is to remove redundant and duplicate connections, which can improve inference speed and achieve better generalization. Besides, it can also reduce energy required and memory access.

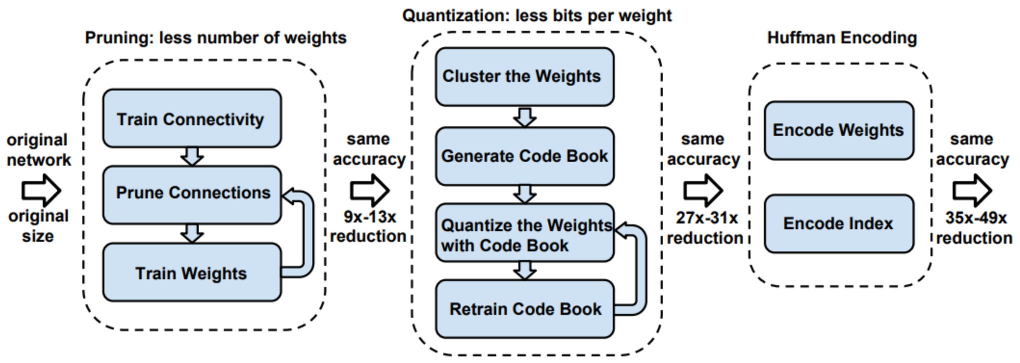


Fig. 1. The overview of deep compression techniques.

II. Paper Survey

■ Optimal Brain Damage [1]

In this paper, they proposed the concept of "saliency", which means the change in the objective function caused by deleting the parameter of neural network. If the saliency is low, the parameter can be removed due to the subtle change for cost function. Therefore, they replaced weight-based method with saliency to decide which parameter can be deleted and got better performance.

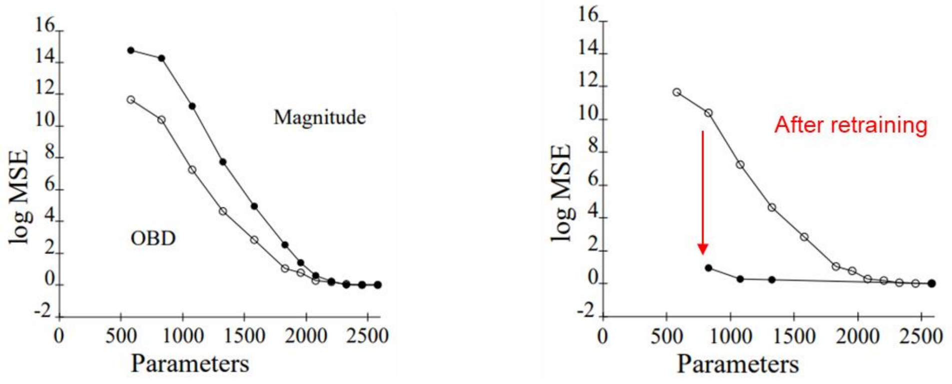


Fig. 2. Simulation result.

■ Second Order Derivatives for Network Pruning: Optimal Brain Surgeon [2]

They found there are several problems arise from simplifying assumptions of OBD which result in choosing the incorrect parameter to delete specially for smaller sized networks. To achieve better performance, they proposed new method to calculate the Hessian matrix recursively to avoid the problems of simplification.

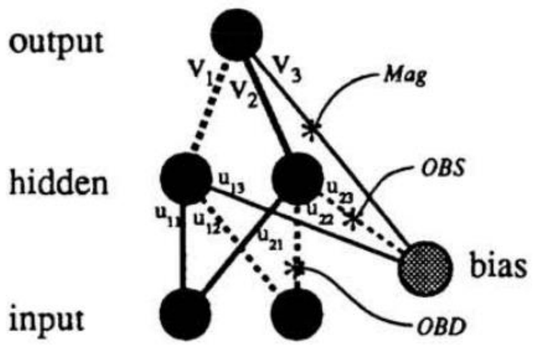
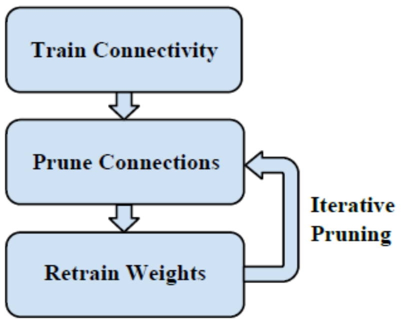


Fig. 3. Parameter removed by different methods.

■ Learning Both Weights and Connections for Efficient Neural Network [3]

In this paper, the connections which are below a threshold will be removed. This method can be divided into three steps -- train connectivity, prune connections and retrain weights.



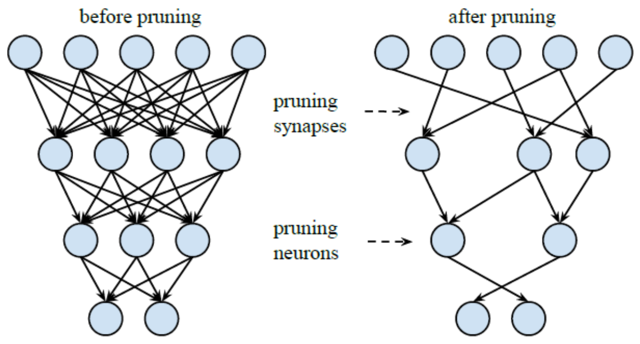


Fig. 4. Three steps process. Fig. 5. The network connections after pruning.

■ Structured Pruning of Deep Convolutional Neural Networks [4]

They thought that most methods are irregular sparsity network connections after pruning which cause extra representation efforts and worse parallel computation. Therefore, they proposed structured pruning with non-zero parameters at well-defined locations and divided into three different level pruning as shown in Fig. 6.

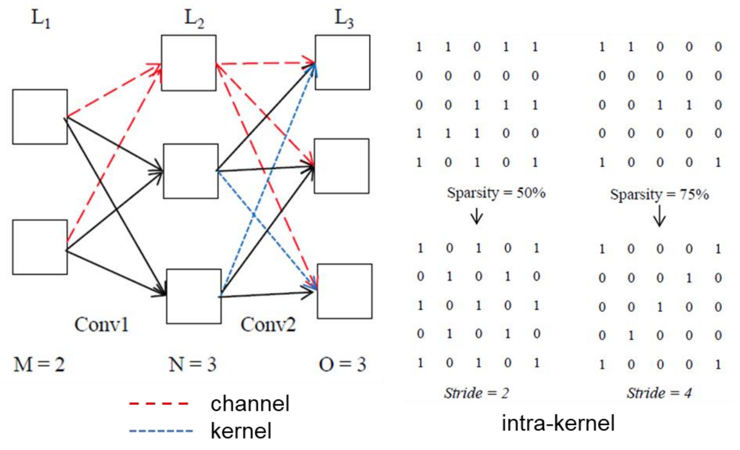


Fig. 6. Three different level of pruning.

■ Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning [5]

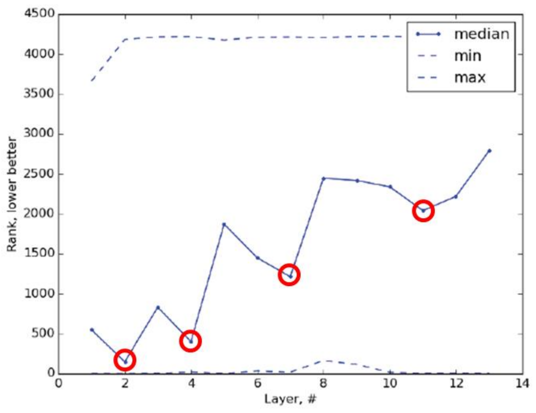
They proposed criteria-based pruning for CNN which means pruning entire feature maps with the optimization functions below:

$$\min_{\mathcal{W}'} |\mathcal{C}(\mathcal{D}|\mathcal{W}') - \mathcal{C}(\mathcal{D}|\mathcal{W})| \ \text{s.t.} \ \|\mathcal{W}'\|_0 \leq B$$

There are different criterias for pruning, such as weight, output feature map, and mutual information. However, they proposed the criteria of Taylor expansion as below:

$$|\Delta\mathcal{C}(h_i)| = |\mathcal{C}(\mathcal{D}, h_i = 0) - \mathcal{C}(\mathcal{D}, h_i)| = \left|\frac{d\mathcal{C}}{dh_i} h_i\right|$$

which means pruning parameters that have flat gradient of the cost function with regard to feature map h_i.



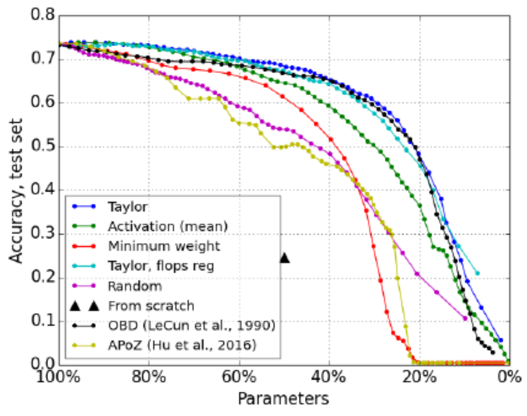


Fig. 7. The importance of different layers.

Fig. 8. The performance of different approaches.

■ Channel-level Acceleration of Deep Face Representations [6]

They proposed two novel methods for compression: one based on eliminating lowly active channels and the other on coupling pruning with repeated use of already computed elements. Besides, pruning of entire channels is an appealing idea, since it leads to direct saving in run time in almost every reasonable architecture.

The comparison between related work is shown below:

	Characteristic	Pruning	Suitable
Y. LeCun [1]	Saliency	Intra-kernel level	small networks
B. Hassibi [2]	Recursively calculate the Hessian matrix	Intra-kernel level	small networks
Han, Song [3]	Absolute value of weight	Intra-kernel level	deep networks
Anwar [4]	Structure pruning	Channel level Kernel level Intra-kernel level	deep networks
P. Molchanov [5]	Taylor expansion	Kernel level	deep networks
Polyak [6]	Prune each layer sequentially	Kernel level	deep networks

III. Database: MNIST

Use MNIST for experiment which is a handwritten digit recognition database. There are 60000 training data and 10000 testing data. Each image is 28x28 resolutions.

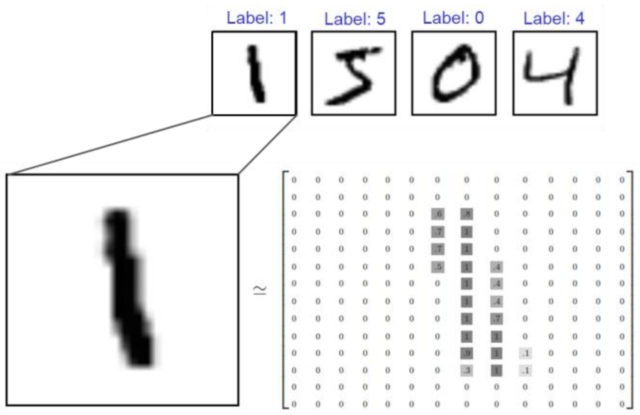
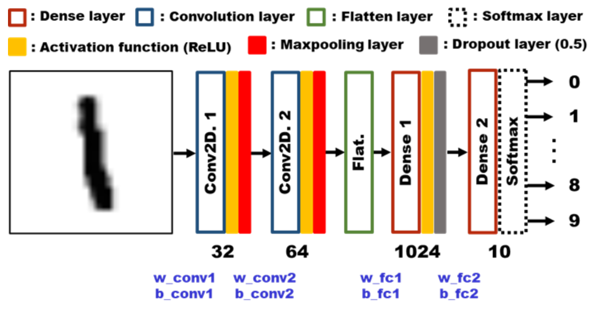


Fig. 9. MNIST database.

IV. Implement

We implement iterative weight-based pruning on TensorFlow which is easier for implementation and has good performance. The model is convolutional neural network with two convolutional layers and two dense layers.



Layer	Parameter
Input	28x28x1
Conv-2D	5x5x32
Maxpooling-2D	2x2x32
Conv-2D	5x5x64
Maxpooling-2D	2x2x64
Dense (Relu)	1024
Output (Softmax)	10

Fig. 10. Our convolutional neural network.

V. Result and Analysis

1. We first compare the pruning method between direct and iterative pruning. Besides, we also want to figure out which network is more important and sensitive to pruning. From Fig. 11, we can find out that iterative pruning is better than direct approaches because it can avoid dramatic degradation of performance via retraining. And the convolutional layer degrades faster than dense layer which means convolutional layer is more sensitive to pruning.
2. The second experiment we want to find out the sensitivity of each layer as shown in Fig. 12. From the results, we can conclude that the first layer of convolutional network and the second layer of dense network are more sensitivity to others.
3. The third experiment we want to evaluate the model size after pruning as shown in Fig. 13. From the results, we can find out that the model weight decreases with the pruning ratio linearly. However, the model size, which use the function of TensorFlow to save model, dramatically increases when pruning ratio is 0.1. This may due to the additionally storage overhead of sparse matrix.
4. The fourth experiment we want to compare the computation time of pruned network to dense network. To our surprise, the computation time of pruned network is higher than dense layer. Therefore, there is no benefit of computation efficiency for pruned network. This may because the sparse connections are not well parallelized for GPU computation.

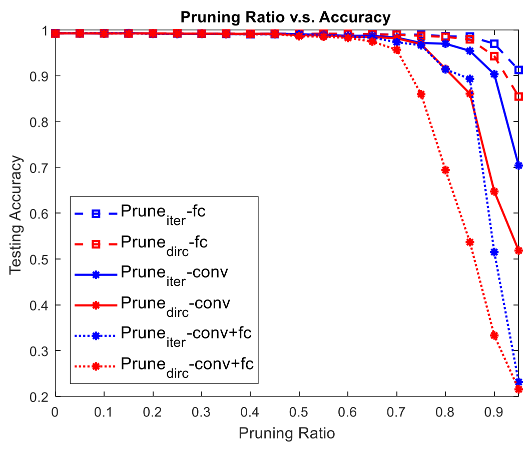


Fig. 11. The pruning ratio to accuracy.

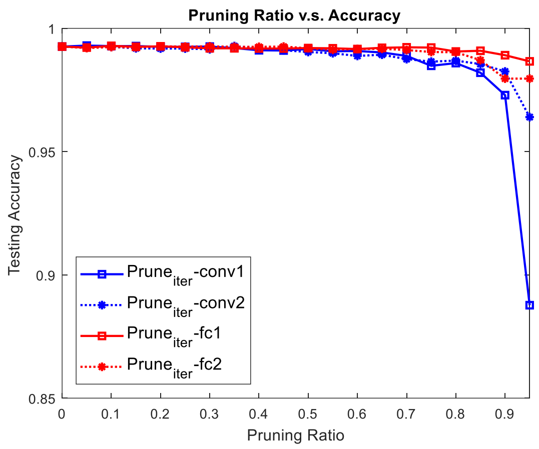


Fig. 12. The sensitivity of each layer.

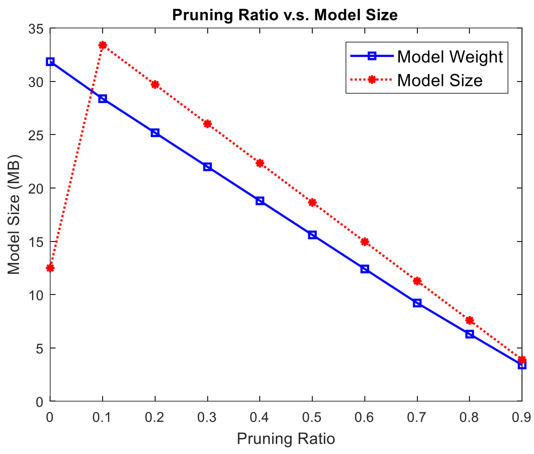


Fig. 13. The pruning ratio to model size.

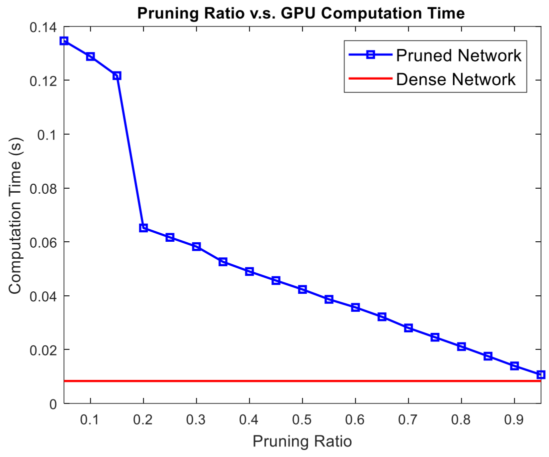


Fig. 14. The pruning ratio to computation time.

VI. Proposed Matrix-based Pruning

Considering to the disadvantages explained above, we want to design a new pruning algorithm, which can maintain original accuracy with less storage overhead for sparse connection and has the advantages of computational efficiency and hardware friendly. The detailed information of our model is illustrated in Fig. 15. And it can be transformed to the format of matrix as shown in Fig. 16. Therefore, our proposed approach is based on removing elements in the matrix as depicted in Fig. 17. With this method, we maintain the form of matrix which can avoid the sparse connection and the additional storage overhead. Besides, the form of matrix is also friendly for GPU computation and hardware implementation.

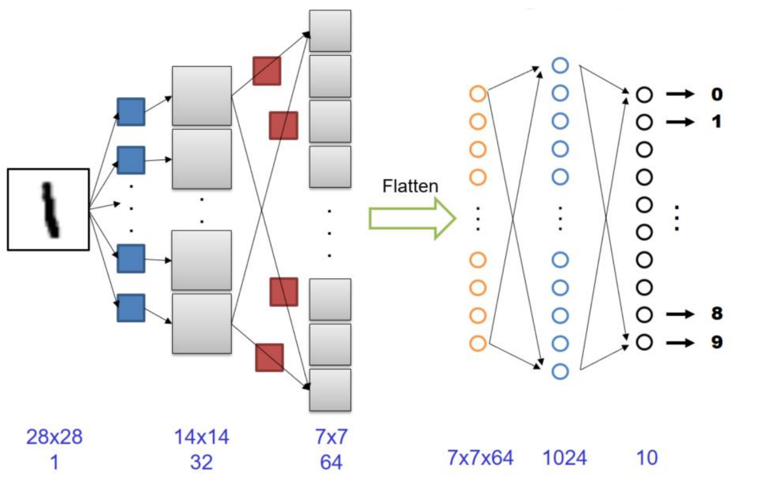


Fig. 15. The detailed information of our convolutional neural network.

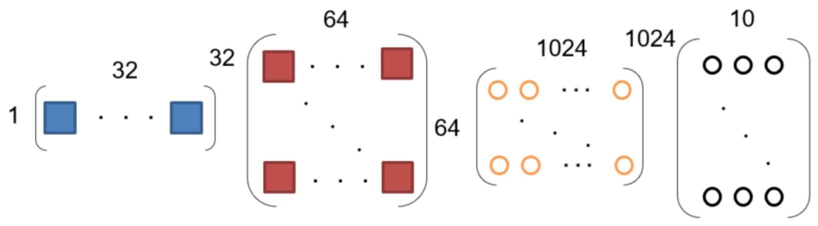


Fig. 16. The matrix form of our network.

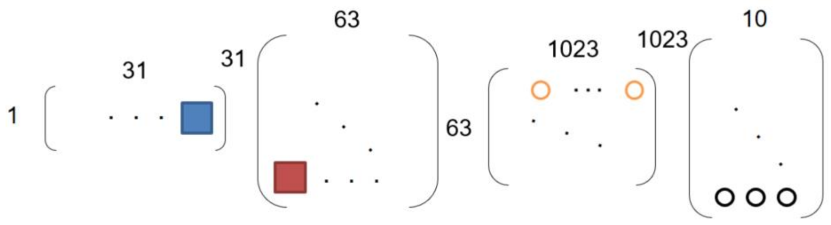


Fig. 17. The result of matrix-based pruning.

Then, we run simulation on proposed matrix-based pruning. From Fig. 18, we can find out that the proposed method has about 2% worse accuracy than original approaches. This may be caused by the more pruned parameters and the matrix-structured pruning. However, the advantages of our proposed approach can be found from Fig. 19 and Fig.20. From Fig. 19, the model weight decreases faster than original. When pruning ratio is 0.8, the proposed method has about 5 times less parameters. Besides, the size of saved model is about 277 times smaller. When applied to larger neural network, the benefit may be more significant. The last experiment is to evaluate the computation time as shown in Fig. 20.

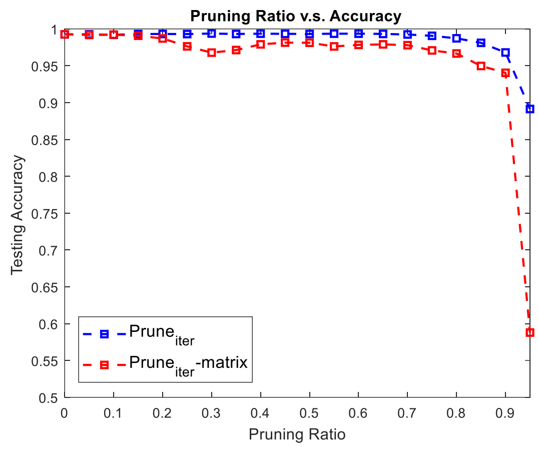


Fig. 18. The pruning ratio to accuracy.

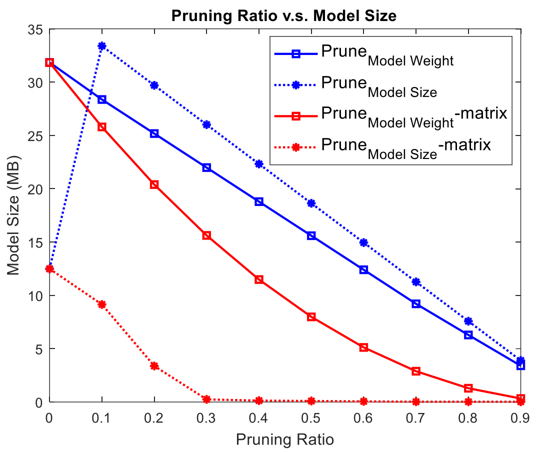


Fig. 19. The pruning ratio to model size.

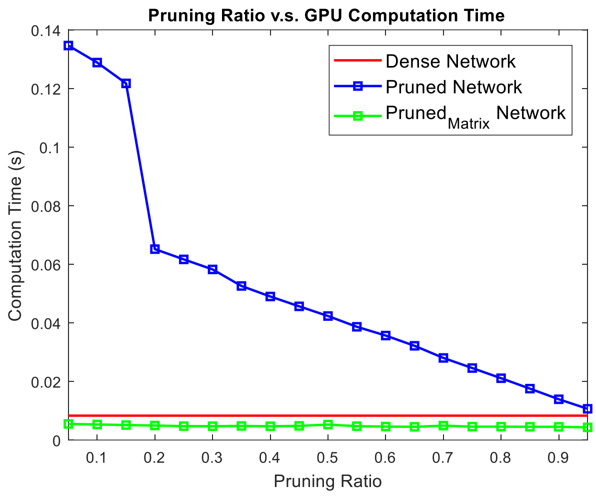


Fig. 20. The pruning ratio to computation time.

VII. Reference

[1] Y. LeCun, J. S. Denker, S. A. Solla, R. E. Howard, and L. D. Jackel, "Optimal brain damage." in NIPs, vol. 89, 1989.

[2] B. Hassibi, D. G. Stork et al., "Second order derivatives for network pruning: Optimal brain surgeon," Advances in neural information processing systems, pp. 164–164, 1993.

[3] Han, Song, et al. "Learning both weights and connections for efficient neural network." *Advances in Neural Information Processing Systems*. 2015.

[4] Anwar, Sajid, Kyuyeon Hwang, and Wonyong Sung. "Structured pruning of deep convolutional neural networks." *ACM Journal on Emerging Technologies in Computing Systems (JETC)* 13.3 (2017): 32.

[5] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz, "Pruning convolutional neural networks for resource efficient transfer learning," *CoRR*, abs/1611.06440, 2016.

[6] Polyak, Adam, and Lior Wolf. “Channel-level acceleration of deep face representations.” IEEE Access 3 (2015): 2163-2175.