Evaluate the Efficiency of Iterative Network Pruning on ResNet Model

Hong Tran (hongtran@wayne.edu)

Advisor: Prof. Dongxiao Zhu

1. INTRODUCTION

The complex neural network is attracting much attention from both researchers as well as practitioners due to its amazing performance. While researchers focusing on growing models deeper and deeper to achieve better state-of-the-art performance; it turns out that these deeper models are facing several challenges: bigger model size, waste more time in the training and inference phase, and high energy consumption due to an expensive computation in deeper models.

With the very fast growth in applying AI in the edge and mobile devices, the model size is one of a big concern. In some circumstances, it might be impossible to deploy these trained models into an edge device (i.e., IoT devices, embedded devices) due to a limitation of the device computation, power, and memory capacity.

To tackle this constraint, many model compression methods were proposed in recent years. Two of these well-known methods are weight pruning and quantization.

In this project, I want to evaluate the efficiency of usinvg the weight pruning and quantization in Deep Compression [1] on a deep neural network (ResNet-18 and ResNet-50) and comparing an experiment result with other methods in the recent year [2], [3].

2. RELATED WORK

A deeper learning models are generally over-parameterized [4] and therefore, these models are significant redundancy in the testing and prediction phase. For example: the ResNet-50 [5] with total 50 convolution layers needs 3.8 billion floating number multiplications operation and 95MB memory storage for its parameters. This redundancy can be helpful during the training phase for optimizing a highly non-convex complex problem which is common in the loss function of the deep learning models.

To make the models more efficient, it needs an enormous joint efforts from many disciplines, including machine learning, optimization, computer architecture. Based on the recent works on network efficiency, we can divide these works into following categorized based on their properties [6].

1. Weight Pruning. This method is to reduce the number of neuron connections as well as unused neurons in the network. This method will transform a dense network to sparse network. There are some issues of using weight pruning. First, the pruning criteria needs to be identified manually. In some application, it is hard to know the best pruning criteria. Second, the pruned network with its sparse weight matrix is reduced in size (by using compressed sparse matrix) but it does not improve the efficiency of training or inference. This issue can be addressed by using some off-the-shelf sparse matrix computation library (e.g., cuSPARSE or MKL SPBLAS library [1]).

2. Weight Sharing/Vector Quantization. This method can be done by reducing the number of bits required to represent the model's weight. There are some techniques are commonly used including k-Means, Product quantization, Residual quantization.

3. Low-rank Approximation. This approach is straightforward for the mode acceleration and compression. However, it requires more computation of decomposition operations and more time for retraining the model to achieve convergence comparing to the original model.

4. Transferred/Compact Convolutional Filters. The use of the transferred convolutional filters to compress convolutional models is suitable for the wide/flat architectures (e.g., VGGNet, AlexNet) but it is not really effective on the thin/deep ones (e.g. ResNet).

5. Knowledge Distillation. This method can make the deep models shallower; however, its performance is less competitive comparing to other type of approaches.

This project targets to conduct an experimental of the Deep Compression 2016 [1] on a deeper neuron network (e.g., ResNet). In general, the scope of this project is as following:

- Understand the concept of the Deep Compression proposal in Song Han 2016 paper [1].
- Implement this concept on Pytorch. The paper's author provides a reference Caffe code for simple Alexnet network, but it is not straight forward to understand. There is one pytorch reference code in GitHub [8], but it does not have the weight pruning for convolution layer, Huffman decode layer and it has several bugs need to be fixed.
- Evaluate this implementation on a deep neural networks model (ResNet) and CIFAR-10 dataset.
- Compare this result with the most recent model compression papers in 2019 ([2] and [3]). The ResNet and CIFAR-10 are well-known in the research community, so we might have much results from other works that we can use to compare with the Deep Compression.

3. TEST RESULT AND DISCUSSION

The Deep Compression was proposed by Song Han 2016 [1]. In this work, the authors suggest a three-stage compression pipeline: pruning, quantization, and Huffman coding (Fig.1).

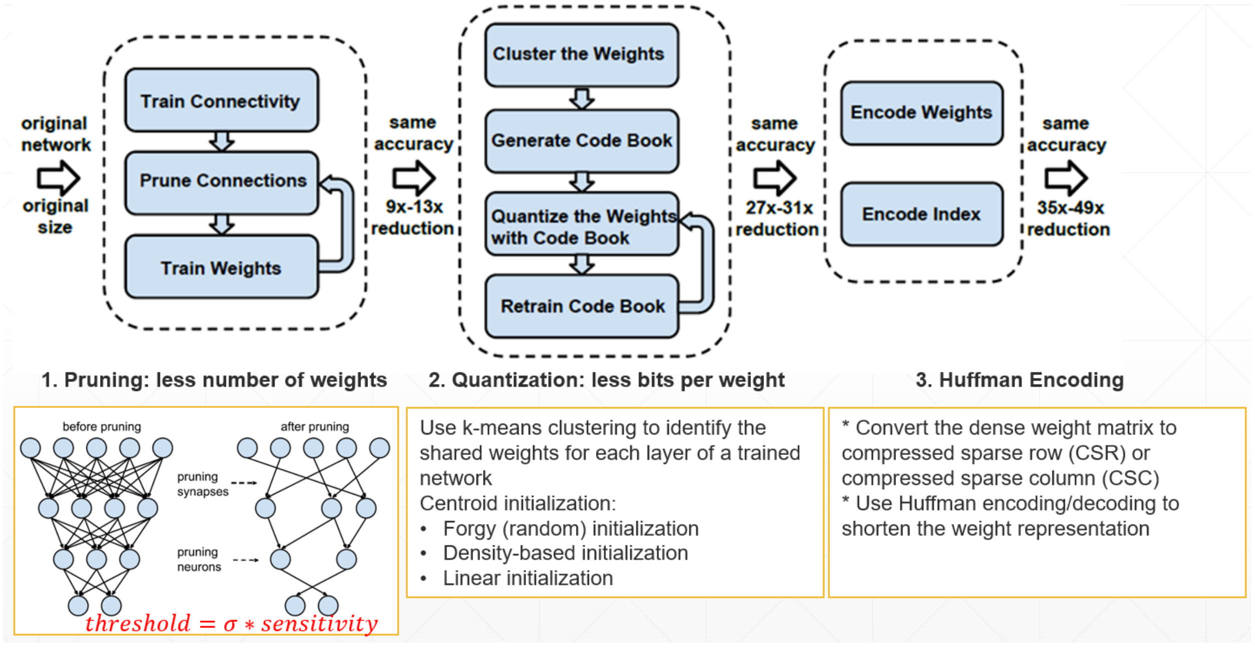
original network
original size
Train Connectivity
Prune Connections
Train Weights
same accuracy
9x-13x reduction
Cluster the Weights
Generate Code Book
Quantize the Weights with Code Book
Retrain Code Book
same accuracy
27x-31x reduction
Encode Weights
Encode Index
same accuracy
35x-49x reduction
1. Pruning: less number of weights
2. Quantization: less bits per weight
3. Huffman Encoding
before pruning
after pruning
pruning synapses
pruning neurons
$threshold = \sigma * sensitivity$
Use k-means clustering to identify the shared weights for each layer of a trained network
Centroid initialization:
• Forgy (random) initialization
• Density-based initialization
• Linear initialization
* Convert the dense weight matrix to compressed sparse row (CSR) or compressed sparse column (CSC)
* Use Huffman encoding/decoding to shorten the weight representation

Figure 1. Deep Compression three-stage pipeline

Step 1: Network Pruning. In this step, we train the model as usual. Once we have the trained model, we start the pruning process by removing all the neuron connections have weight exceed the $threshold = \sigma * sensitivity$. In this experiment, I tune the sensitivity hyperparameter to 0.02 for achieving the best result. We can do this step iteratively until we observe the overfitting. This process actually transforms the model's dense weight matrix to a sparse matrix where all the removed neuron connections will be weighted as zero.

Step 2: Quantization. We can further compress the pruned model by applying quantization. We use the k-Means with linear centroid initialization to quantize the weight matrix of each layer in the model.

Step 3: Convert the model weight matrix to either compressed sparse row (CSR) or compressed sparse column (CSC) format. Finally, we apply Huffman encoding to reduce the number of bits representing the compressed weight matrix.

In this work, I did an experiment on the CIFAR-10 dataset and ResNet-18 and ResNet-50 which have 18 and 50 layers deeper, respectively. The result showing that 94.55% of parameters of the ResNet-50 are pruned (Fig. 2) and the model size finally reduced 57.08x with 2.28% test accuracy reduction as a tradeoff (Table 1). The result is similar to the shallower model ResNet-18.

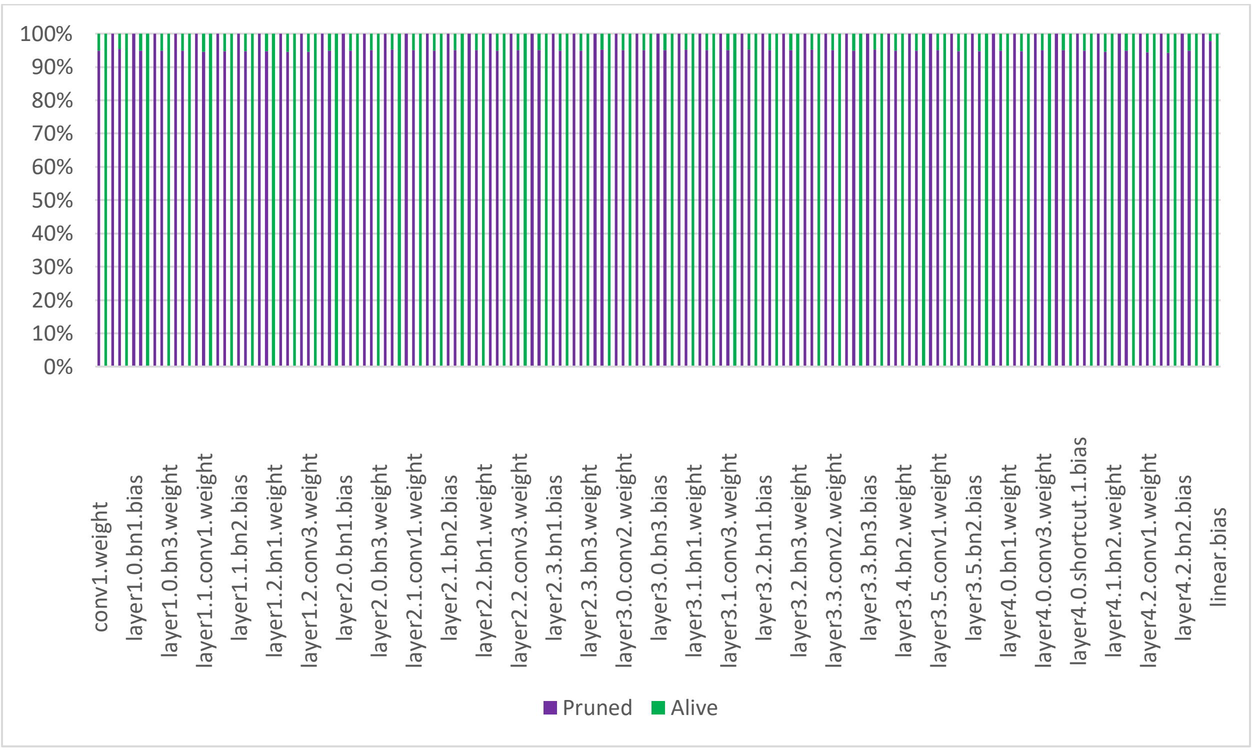

Figure 2. ResNet-50 Weight Pruning

Model	Initial Accuracy	After Pruning				Weight Sharing	Convert to CSR/CSC matrix and Huffman Encoding		Final Result	
		Alive Param	Compression Rate	Accuracy	Accuracy after Retrain	Accuracy	Accuracy	Compression Rate More	Model size	Accuracy
ResNet-18: Total Param #: 11,173,962	87.59% (81 epoches)	563,354	19.83x (94.96% pruned)	10%	89.95%, ↑2.36% (19 epoches)	89.42%, ↓0.53%	89.42%	↓3.52x, ↓71%	↓65.58x, ↓98.47% # Param: ↓94.96%	89.42%, ↑1.83%
ResNet-50: Total Param #: 23,520,842	94.55% (150 epoches)	1,239,011	18.98x (94.73% pruned)	10%	92.6%, ↓1.95% (100 epoches)	92.27%, ↓0.33%	92.27%	↓3.41x, ↓29.35%	↓57.08x, ↓98.25% # Param: ↓94.73%	92.27%, ↓2.28%

Table 1. Deep Compression 2016 Final Result on ResNet-18 and ResNet-50

Comparing to the recent works in 2019 [2] and [3], the Deep Compression 2016 still outperforms the other works with the same test accuracy and achieve a better pruning rate. In the Factorized Convolutional Filters method proposed in the work [2], the CNN-FCF on ResNet-20 and ResNet-56 can prune 68% - 69% parameters with 2.67% - 1.22% decreasing in the test accuracy. In the other work [2] on ResNet-18, by using the Network Purification and Unused Path Removal, it reduces 93.79% parameters and has a 0.92% drop in the accuracy.

We also can compare the Deep Compression's result with the MobileNetV2 2018 [7] (on the same CIFAR-10 dataset). The MobileNetV2 model requires 2.3M parameters to reach the test accuracy of 91.35% [7]. While Deep Compression on ResNet-50 only needs 1.2M parameters to get 92.27% accuracy.

4. CONCLUSION

The Deep Compression 2016 achieves a very good pruned rate and testing accuracy comparing to the recent methods (MobileNetV2 in 2018 and other compression models in 2019). However, the deep compression might have a big inference timing due to some delay in decoding:

- Compressed sparse row (CSR) / Compressed sparse column (CSC) format.
- Huffman encoded weight.

This limitation can be improved by using the hardware accelerator for the sparse matrix calculation (e.g., cuSPARSE CSRMV kernel which is optimized for sparse matrix-vector multiplication on GPU).

One point we can notice in the test result is that the accuracy after compression is slightly increased (1.83%) instead of decreasing. My guess is that the model original accuracy (87.59%) has not reached the optimized point; therefore, the pruning process might remove some unused neurons and make the model a little bit better (as same as we shrink the weight parameter in Lasso regularization).

In future work, we can conduct more experiments to measure and compare the inference timing and the FLOPS complexity between Deep Compression and other models.

BIBLIOGRAPHY

[1] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2015.

[2] T. Li, B. Wu, Y. Yang, Y. Fan, Y. Zhang and W. Liu, "Compressing Convolutional Neural Networks via Factorized Convolutional Filters," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 3972-3981, doi: 10.1109/CVPR.2019.00410.

[3] Ma, Xiaolong & Yuan, Geng & Lin, Sheng & Li, Zhengang & Sun, Hao & Wang, Yetang. (2019). ResNet Can Be Pruned 60x: Introducing Network Purification and Unused Path Removal (P-RM) after Weight Pruning.

[4] Misha Denil, Babak Shakibi, Laurent Dinh, Nando de Freitas, et al. Predicting parameters in deep learning. In Advances in Neural Information Processing Systems, pages 2148–2156, 2013.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," CoRR, vol. abs/1512.03385, 2015.

[6] Choudhary, T., Mishra, V., Goswami, A. et al. A comprehensive survey on model compression and acceleration. Artif Intell Rev 53, 5113–5155 (2020). https://doi.org/10.1007/s10462-020-09816-7

[7] http://dorsalstream.net/2018/04/10/MobilenNetv2/

[8] https://github.com/mightydeveloper/Deep-Compression-PyTorch