Compression of Deep Neural Networks by combining pruning and low rank decomposition

Anonymous Author(s)
Affiliation
Address
`email`

Abstract

Large number of weights in deep neural networks make the models difficult to be deployed in low memory environments such as, mobile phones, IOT edge devices as well as "inferencing as a service" environments on the cloud. Prior work has considered reduction in the size of the models, through compression techniques like weight pruning, filter pruning, etc. or through low-rank decomposition of the convolution layers. In this paper, we demonstrate the use of multiple techniques to achieve not only higher model compression but also reduce the compute resources required during inferencing. We do filter pruning followed by low-rank decomposition using Tucker decomposition for model compression. We show that our approach achieves upto 57% higher model compression when compared to either Tucker Decomposition or Filter pruning alone at similar accuracy for GoogleNet. Also, it reduces the Flops by upto 48% thereby making the inferencing faster.

1 Introduction

Deep neural networks are now being used extensively for a variety of artificial intelligence applications ranging from computer vision [19] to speech recognition [11] and natural language processing [5]. In this paper, we focus particularly on convolutional neural networks (CNNs) which have become ubiquitous in object recognition, image classification, and retrieval (see [17, 8, 10, 29]). As datasets increase in size, networks also increase in complexity, number of layers and parameters in order to absorb the supervision. The increased size of the networks makes it increasingly difficult for the model to be deployed in low memory environments such as, mobile phones, IOT edge devices etc. Recent work has considered reducing the size of networks with limited loss of accuracy in the prediction, so that the model can fit in the memory of low resource systems. For example, in one class of approaches, pruning of the weights of a trained CNN [12] or pruning at the level of feature maps and kernels [2, 24] is done to reduce the model size. Low-bit precision and weight quantization have also been used both to store the CNN parameters as well as for training and inferencing with these models (see half-precision networks [1], XNOR-Net [25], DoReFa-Net [30]), network binarization [6], ternary weight networks [14, 21, 31], vector quantization [9, 22], HashedNets [4] for examples). Recently, there have been some work to transform the convolutional filters to low rank filters using various matrix-factorization and clustering techniques [7, 16, 18, 20, 26] and speeding up computations using FFT [23]. More recently there has been effort to come up with new network architectures to make them more efficient by reducing the model size, working memory and inference time. Networks like SqueezeNet [15] and MobileNet [13] restricts their kernel sizes to 1x1 and 3x3 to reduce the compute and memory requirements and to make inferencing faster.

In this paper, we focus on transfer learning. In such a setting, filter pruning is effective in removing filters that are not relevant for the incremental data. Low rank decomposition techniques on the other hand reduce the dimensions of the weight tensors without losing (much) information. In this paper, we study these complementary techniques and show that by combining these techniques, we achieve an additional 57% model compression when compared to either filter pruning or Tucker Decomposition for popular models like GoogleNet. Also, it reduces the Flops by upto 48% thereby making the inferencing on these networks very fast. The rest of the paper is organized as follows. In Section 2, we describe our methodology of combining filter pruning with tensor decomposition. Our experimental results under different settings are presented in Section 3. Finally, we present our conclusions in Section 4.

2 Methodology

We briefly describe Tucker decomposition and filter pruning approaches from prior work followed by our approach for combining these techniques.

Tucker Decomposition: Tucker decomposition has been widely used as a Low-Rank factorization method for decomposing Convolution and Fully connected layers in CNN [18]. It computes a Higher Order Singular Value Decomposition(HOSVD) of a n-D Tensor along each of it's dimensions/modes. For CNNs, the convolution layer is a 4D Tensor where the first 2 dimensions are the output and input of that layer and the remaining 2 dimensions are the spacial dimensions. The Tucker decomposition of this 4D tensor results in a set of 2D-matrices U along each of the dimensions of the tensor (also called modes) and a core tensor G. A trade-off between space and accuracy can be achieved by varying the ranks of the output core tensor and factor matrices.

$$K_{i,j,s,t} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} \sum_{r_4=1}^{R_4} G_{r_1,r_2,r_3,r_4} \times U^1_{i,r_1} \times U^2_{j,r_2} \times U^3_{s,r_3} \times U^4_{t,r_4} \qquad (1)$$

Filter Pruning: Pruning filters from convolution layers is a standard method of compressing the CNNs [24]. There are several methods of removing filter from CNNs based on their importance. We have followed the techniques suggested in [24] where the filters are removed by minimizing the Taylor series expansion of the error introduced by removing a filter as it yields better results. A threshold parameter provides a tradeoff between space and accuracy by controlling the number of filters to be pruned.

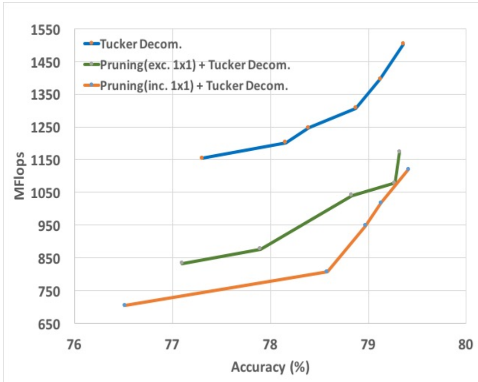
Our Approach: In our proposed method we first perform filter pruning with different pruning percentages (20%, 30%, 40% & 50%). For each of these filter pruned models, we use Tucker decomposition to further reduce the model size and flops required during inferencing. Since the kernel size for CNNs is usually small (of the order of 1x1, 3x3, 5x5 etc.), Tucker decomposition is applied only on mode-1 and mode-2 of the 4-D weight Tensor of a particular layer. Thus, R_3 & R_4 in (1) are equal to s & t respectively and the ranks R_1 & R_2 are determined using Variational Bayesian Matrix Factorization (VBMF) as described in [18]. In order to exploit the trade-off between space and accuracy we vary the threshold parameter of VBMF (varying from 0.8 to 1.4) which determines the low-rank for the approximation. All the implementations were done using Caffe.

Incremental Training: Unlike [18] we perform layer-wise Tucker decomposition in an incremental manner where we decompose one layer at a time and fine-tune entire network for 2 epochs before proceeding to the next layer. This helps the network to regain the accuracy lost due to low rank approximation. After layer-wise decomposition the entire network is fine-tuned for 50 epochs to get to the base accuracy.

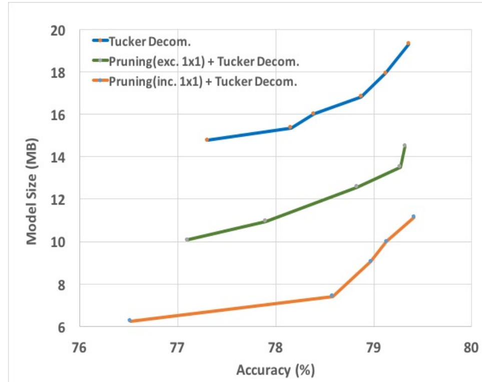
3 Experimental Results

Models Used. We demonstrate our results on state-of-the-art deep neural network GoogleNet [27]. The base model is trained on ImageNet-1K dataset. The datasets used for transfer learning are Food101 [3] and Bird200 [28].

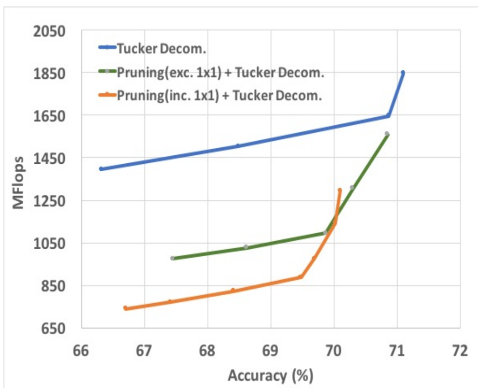
Baseline. Our baseline considers the test accuracy achieved by a model compressed by applying the strategy similar to the one presented in [18], where low rank decomposition of the 3x3 and 5x5 convolution tensors are effected layer by layer employing (i) determining the rank R_3 and R_4 by applying global analytic VBMF on mode-3 matricization and mode-4 matricization of kernel tensors (ii) Tucker decomposition on the tensor (iii) fine-tune the entire network with standard back-propagation. Note that the one-shot whole network decomposition presented in [18] produces worse test accuracy (for the same model size/flops) than the baseline used here, and hence is skipped.



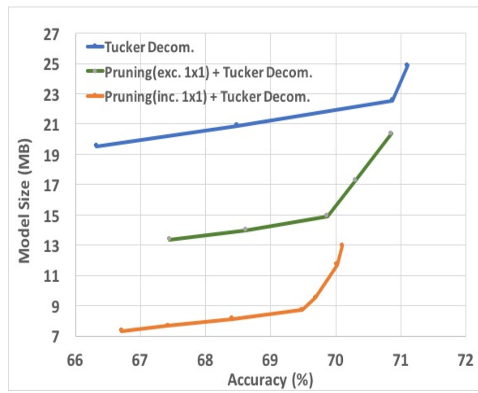
(a) GoogleNet,Food101 (MFlops vs Accuracy)



(b) GoogleNet,Food101 (Modelsize vs Accuracy)



(c) GoogleNet,Bird200 (MFlops vs Accuracy)



(d) GoogleNet,Bird200 (Modelsize vs Accuracy)

Figure 1: Accuracy comparison of combining filter pruning with tensor decomposition over baseline.

Comparison with baseline. We have compared the test accuracy of our approach of combining filter pruning with tensor decomposition with the baseline for GoogleNet for various compression levels. Thus, while the baseline attains a particular model size by employing only Tucker decomposition, the same model size is achieved in our approach by a appropriate combination of filter pruning and tensor decomposition. In an analogous manner, the accuracy of our approach is compared with the baseline for the same computations (FLOPS) of the compressed model.

Figure 1c-1b shows the accuracy gains obtained using our algorithm over the baseline for different compression levels based on model size and computational flops. Since the baseline (tensor decomposition by rank determination through VBMF) does not involve the 1x1 tensors, we show the comparisons for both the scenarios where the 1x1 filters are included (and excluded) in the filter pruning step. We first observe that for each of the compression mechanisms, the drop in accuracy is initially small for some level of compression, but increases drastically as model size or flops decreases. Obviously, pruning the 1x1 filters lead to further reduction in the model size and flops over the scenario when they are not; however even if we keep the 1x1 filters intact, there is significant increase in the test accuracy for the compressed model obtained by combination of filter pruning and tensor decomposition over the model of same size (or flops) obtained just by tensor decomposition.

4 Conclusions and Future Work

We show that our approach of filter pruning followed by low-rank decomposition using Tucker decomposition achieves higher model compression and lower inference complexity when compared to either Tucker Decomposition or Filter pruning alone at similar accuracy for GoogleNet. A future work in this aspect is to incorporate the filter pruning process in the tensor decomposition itself.

References

[1] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse Engel, Linxi Fan, Christopher Fougner, Tony Han, Awni Y. Hannun, Billy Jun, Patrick LeGresley, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Yi Wang, Zhiqian Wang, Chong Wang, Bo Xiao, Dani Yogatama, Jun Zhan, and Zhenyao Zhu. Deep speech 2: End-to-end speech recognition in english and mandarin. *CoRR*, abs/1512.02595, 2015.

[2] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *JETC*, 13(3):32:1–32:18, 2017.

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – mining discriminative components with random forests. In *European Conference on Computer Vision*, 2014.

[4] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 2285–2294, 2015.

[5] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.*, 12:2493–2537, November 2011.

[6] Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016.

[7] Emily Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, pages 1269–1277, Cambridge, MA, USA, 2014. MIT Press.

[8] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, pages 647–655, 2014.

[9] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir D. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115, 2014.

[10] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII*, pages 392–407, 2014.

[11] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[12] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015.

[13] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[14] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *CoRR*, abs/1609.07061, 2016.

[15] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and $<$0.5mb model size. *arXiv:1602.07360*, 2016.

[16] Yani Ioannou, Duncan P. Robertson, Roberto Cipolla, and Antonio Criminisi. Deep roots: Improving CNN efficiency with hierarchical filter groups. In *2017 IEEE Conference on Computer Vision and Pattern Recognition,CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 5977–5986, 2017.

[17] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross B. Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia, MM '14, Orlando, FL, USA, November 03 - 07, 2014*, pages 675–678, 2014.

[18] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *CoRR*, abs/1511.06530, 2015.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, 2012.

[20] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

[21] Fengfu Li and Bin Liu. Ternary weight networks. *CoRR*, abs/1605.04711, 2016.

[22] Zhouhan Lin, Matthieu Courbariaux, Roland Memisevic, and Yoshua Bengio. Neural networks with few multiplications. *CoRR*, abs/1510.03009, 2015.

[23] Michael Mathieu, Mikael Henaff, and Yann LeCun. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2013.

[24] 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.

[25] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*, pages 525–542, 2016.

[26] Aruni RoyChowdhury, Prakhar Sharma, Erik Learned-Miller, and Aruni Roy. Reducing duplicate filters in deep neural networks. In *NIPS workshop on Deep Learning: Bridging Theory and Practice*, 2017.

[27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[28] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset, 2011.

[29] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, pages 818–833, 2014.

[30] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016.

[31] Chenzhuo Zhu, Song Han, Huizi Mao, and William J. Dally. Trained ternary quantization. *CoRR*, abs/1612.01064, 2016.