Hybrid Pruning: Pruning Filters and Weights

Shitikantha Bagh
Department of Electrical Engineering
North Carolina State University
Raleigh,USA
sbagh@ncsu.edu

Efren Martinez-Gomez
Department of Computer Science
North Carolina State University
Raleigh,USA
emartin8@ncsu.edu

Wujie Wen
Department of Computer Science
North Carolina State University
Raleigh,USA
wwen2@ncsu.edu

***Abstract*— In recent years, neural networks have become very powerful and accurate, but they are usually accompanied by an increase in computational power and require large amounts of memory storage. The next stage of improvement is to reduce computational resources and inference cost by pruning the neural network while maintaining accuracy and performance. We decided to approach the pruning of a neural network in two parts. The first part is where we introduce "Pruning Filters for Efficient Convnets"[1], which focuses on pruning filters from CNNs that are not important in the final accuracy. By removing whole filters, computational cost is significantly reduced. After pruning the filters, we retrain the model. This greatly reduces inference and computation cost but does not lead to a drop in accuracy. However, this does not result in sparse connectivity patterns. This leads us to the second half of our approach, which is utilizing the weight connectivity method in "SNIP: Single Shot Network Pruning Based on Connection Sensitivity"[2]. This method introduces a saliency criterion based on connection sensitivity that identifies structurally important connections in the network. It prunes weight connections that are the weakest, leading to an extremely sparse network with relatively the same accuracy. Overall, these two techniques combined lead to a drastic reduction in inference speed and computational cost while maintaining the same accuracy as before the network was pruned. Our method is applicable to various architectures and obtains extremely sparse networks.**

Keywords*— *CNN training, Resource optimization, Pruning methods

I. Introduction

There have been significant advances in various architectural choices in CNNs when dealing with ImageNet (Russakovsky et al. (2015); Krizhevsky et al. (2012); Simonyan & Zisserman (2015); Szegedy et al. (2015a); He et al. (2016)). This has led to networks growing bigger and better but with an increase in the number of parameters and convolution operations. This leads to them being computationally expensive with large amounts of memory required. Although these networks have become more capable and powerful they have significant inference costs. Especially when used with devices that have limited resources. For these applications, in addition to accuracy, computational efficiency and small network sizes are crucial enabling factors (Szegedy et al. (2015b)).

Many pruning techniques have been developed for deep neural networks in machine learning that can reduce the storage and computation costs by model compression (Le Cun et al. (1989); Hassibi & Stork (1993); Srinivas & Babu (2015); Han et al. (2015); Mariet & Sra (2016)). Recently Han et al. (2015; 2016b) report impressive compression rates on AlexNet (Krizhevsky et al. (2012)) and VGGNet (Simonyan & Zisserman (2015)) by pruning weights with small magnitudes and then retraining. Pruning large networks without losing accuracy and decreased inference time is always the goal so they can be applied in real-time applications especially where the resources are limited on the device. In addition, compressed neural networks utilize the model capacity efficiently, and this interpretation can be used to derive better generalization bounds for neural networks (Arora et al. (2018)). However, pruning parameters does not always lead to reducing the computation time since the majority of the parameters removed are from the fully connected layers where the computation cost is low, e.g The fully connected layers of VGG-16 occupy 90% of the total parameters but only contribute less than 1% of the overall floating point operations (FLOP). They also show that convolutional layers can be effectively compressed (Iandola et al. (2016)), but do need more libraries or even specialized hardware (Han et al. (2016a)). This can lead to additional overhead in memory.

CNNs of high capacity usually have significant redundancy among different filters and feature channels. We decided to focus on reducing the computation cost of well-trained CNNs by pruning filters and additionally pruning the weights afterwards. This way we can maximize the reduction of the inference time and computational cost. We use the methods in Pruning Filters For Efficient Convnets[1] to filter prune. Filter pruning is a naturally structured way of pruning that does not introduce sparsity and therefore does not require using sparse libraries or any specialized hardware. We introduce sparsity after we prune the filters and prune the weights based on (SNIP))[2] method that requires no additional libraries or specialized hardware.

We evaluate our method on CIFAR-10 classification datasets with VGG-16 architecture. Despite being simple, our method obtains extremely sparse networks with virtually the same accuracy as the existing baseline.

II. Related Work

The classical methods or early works in network pruning can be categorized into two groups (Reed, 1993):

Sparsity Enforcing Penalties: Methods such as those by Chauvin (1989), Weigend et al. (1991), and Ishikawa (1996) augment the loss function with sparsity enforcing penalty terms (e.g., L0 or L1 norm). Back-propagation penalizes the magnitude of weights during training, and weights below a certain threshold are removed.

Saliency Criterion: Classical saliency criteria include sensitivity of the loss with respect to neurons (Mozer & Smolensky, 1989) or weights (Karnin, 1990) and Hessian of the loss with respect to weights (LeCun et al., 1990; Hassibi et al., 1993). These methods are slow, requiring many iterations of pruning and learning steps.

In recent years, the increased complexity and risk of overfitting in deep neural networks have led to further investigation in network pruning:

Magnitude-Based Approaches: Han et al. (2015) introduced methods using the magnitude of weights as the criterion, achieving extreme sparsity without loss in accuracy. However, drawbacks include reliance on pretraining and expensive prune–retrain cycles.

Hessian-Based Approaches: Hessian-based approaches employ diagonal approximation for computational simplicity but may have drawbacks in terms of overfitting.

Bayesian Methods: Bayesian methods, as applied by Ullrich et al. (2017) and Molchanov et al. (2017a), provide alternatives for network pruning using soft weight sharing and variational inference.

Additional work has been done on reducing computation costs in convolutional layers and removing redundant feature maps:

Computation Cost Reduction: Approaches include approximating convolutional operations, FFT-based convolutions, fast convolution using the Winograd algorithm, quantization, and binarization.

Filter-Level Pruning: Work by Anwar et al. (2015) and Polyak & Wolf (2015) focuses on removing redundant feature maps. Anwar et al. (2015) use particle filtering to select pruning candidates, while Polyak & Wolf (2015) detect less frequently activated feature maps.

Concurrent Work: Lebedev & Lempitsky (2016), Zhou et al. (2016), and Wen et al. (2016) explore training compact CNNs with sparse constraints, leveraging group-sparsity, and introducing structured sparsity regularizers during training. The work presented here focuses on `1-norm for filter-level pruning without introducing extra layer-wise meta-parameters.

The proposed approach in this work may not lead to the best sparsity available compared to other neural networks that have sparsity libraries or customized hardware but it shows good results for its simplicity and adaptability to different architectures and tasks without requiring extensive modification of the pruning procedure.

III. METHODOLOGY

In pursuit of creating computationally efficient and high-performance convolutional neural networks (CNNs), our methodology combines strengths from two prominent pruning techniques: filter-based pruning inspired by the work of Li et al. and connection sensitivity-based pruning as proposed by Lee et al.. Our goal is to synergize these approaches and develop a model that leverages the inherent efficiency of filter pruning and refines its architecture through connection sensitivity.

A. Model Selection and Architecture

Our choice of the VGG16 architecture as the baseline stems from its well-established reputation in the computer vision community. VGG16 strikes a balance between model depth and simplicity, making it an ideal starting point for our exploration. This model selection aligns with our goal of applying pruning techniques to a widely recognized and utilized architecture, allowing for a more comprehensive evaluation of the efficacy of our combined pruning approach.

With the foundation laid in these methodologies, we embark on an intricate journey of iterative filter pruning and connection sensitivity-based fine-tuning, aiming to produce a model that not only conserves computational resources but also excels in capturing essential features for image classification tasks. The subsequent sections detail our step-by-step process, experimental setup, and comprehensive evaluation to shed light on the effectiveness and potential advantages of our novel pruning methodology.

B. Model Pruning

The first phase of our methodology involved the application of filter pruning to enhance model efficiency by identifying and eliminating redundant or less impactful filters that contribute minimally to the overall network performance.

- Filter Pruning: The pruning process involves pruning filters from convolutional layers based on their l-norms, which measure the magnitude of the filter weights. By pruning filters with small 1-norms, it reduces the computation cost and the number of parameters of the convolutional neural networks (CNNs) without hurting the accuracy or introducing sparsity

- Filter Selection Criterion: The sum of the absolute kernel weights of each filter is used as the criterion to select which filters to prune. Filters with smaller kernel weights tend to produce weaker activations and are less important for the output. This criterion is compared with other activation-based criteria and shows that it is effective and data-free.

- Layer Sensitivity Analysis: The sensitivity of each layer to pruning by pruning each layer independently and evaluating the accuracy on the validation set is analyzed. Some layers are more robust or sensitive to pruning than others and this information is used to determine the number of filters to prune for each layer or stage.

- Pruning Across Multiple Layers: There are two strategies to prune filters across multiple layers: independent pruning and greedy pruning. Independent pruning selects the filters to prune at each layer without considering other layers, while greedy pruning accounts for the filters that have been removed in the previous layers. Greedy pruning results in higher accuracy than independent pruning, especially when many filters are pruned.

- Retraining Pruned Networks: After pruning the filters, the pruned networks are retrained to regain the accuracy. There are two strategies to re-train the networks: prune once and retrain, or prune and retrain iteratively. For the layers that are resilient to pruning, the prune once and retrain strategy can be used to save retraining time, while for the layers that are sensitive to pruning or large portions of the networks, the iterative pruning and retraining strategy may yield better results.
- Weights Pruning: The filter-pruned-trained model undergoes weight pruning based on it's sensitivity. The criterion to measure the importance of each connection in a neural network is based on its influence on the loss function. Binary indicator variables are used to represent whether a connection is active or pruned, and then approximate the effect of removing a connection by the derivative of the loss with respect to the indicator variable. This criterion is independent of the weight scale and can be computed efficiently using automatic differentiation. This criterion connection sensitivity and use it to prune redundant connections. Finally, the model is retrained for one last time to regain accuracy.

By strategically removing non-essential filters and less sensitive weights, the resulting model is characterized by reduced memory footprint, computational requirements, and potentially faster inference times—all without compromising its ability to capture and represent essential features in the input data. This methodology aligns with the broader goals of model efficiency, particularly in resource-constrained environments where computational and memory resources are at a premium.

C. *Experimental Setup*

Dataset: For our experiments, we chose the CIFAR-10 dataset due to its diversity and relevance to image classification. The dataset consists of 60,000 32x32 color images across 10 classes, providing a challenging yet representative testbed for our pruned model's generalization capabilities.

Training and Validation: The pruned VGG16 model underwent an extensive training regimen to adapt to the pruned architecture. Careful validation at each step ensured that the model's performance met the desired benchmarks, and any potential issues were promptly addressed.

D. *Evaluation Metrics*

To assess the efficacy of our pruning strategy, we employ two primary evaluation metrics: top-1 accuracy and top-5 accuracy. These metrics offer nuanced insights into the model's performance, providing a comprehensive understanding of its ability to correctly classify the most probable class (top-1) and the top five most probable classes (top-5).

The top-1 accuracy represents the proportion of test samples for which the correct class is the model's most confident prediction. Meanwhile, the top-5 accuracy accounts for the correct classification when considering the top five predictions made by the model. These metrics collectively serve as robust indicators of the pruned model's classification capabilities.

In addition to top-1 and top-5 accuracy, we consider other relevant metrics such as computational time and memory footprint to provide a holistic view of the trade-offs associated with our pruning strategy.

IV. Results

Our baseline model's total size was 63.78 MB with zero sparsity and Top1 and Top 5 accuracy to be 93.65% and 99.34% respectively along with the forward pass duration for inference was recorded at 0.674 sec. After our first step of pruning the filters, the model's total size reduced to 54.74 MB with zero sparsity and Top1 and Top 5 accuracy reduced to 38.15% and 85.16% respectively along with the forward pass duration for inference was recorded at 0.357 sec. The pruned model was retrained to regain the Top1 and Top5 accuracy to 93.70% and 99.33% respectively. The weights of the pretrained model were pruned to introduce 40% sparsity. The model was trained again and the Top1 and Top5 accuracy increased to 99.97 % and 100% respectively and now the forward pass duration for inference was recorded at 0.005 sec.

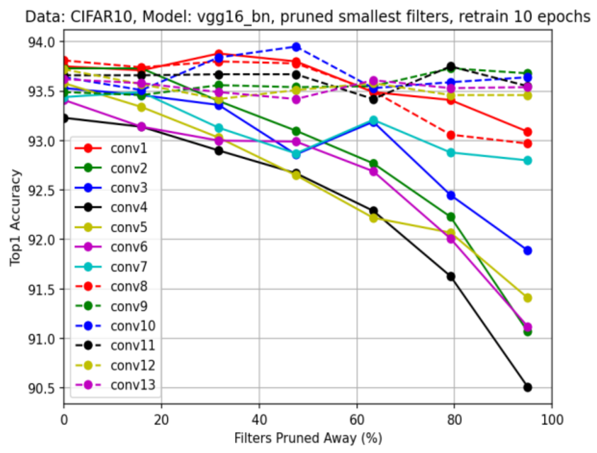

Fig. 1. Top 1 accuracy vs % of Filters pruned away in each layer

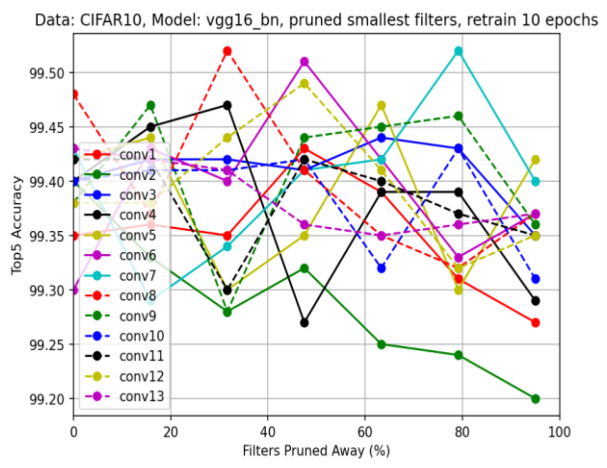

Fig. 2. Top 5 accuracy vs % of Filters pruned away in each layer

V. FUTURE SCOPE

A. *Evaluations on Different Architectures and Datasets*

To broaden the applicability and generalizability of our pruning methodology, future work will involve extensive evaluations on various CNN architectures beyond VGG16. Assessing the performance of the proposed pruning technique on architectures with diverse complexities and structures will provide valuable insights into its adaptability.

Furthermore, exploring the impact of our methodology across a spectrum of datasets will contribute to a comprehensive understanding of its effectiveness. Evaluations on datasets with varying characteristics, sizes, and domains will offer insights into the model's robustness and versatility.

B. *Removal of Unimportant Filters and Layer Weights based on Sensitivity*

A promising avenue for future research involves refining the pruning methodology to incorporate the removal of filters and layer weights based on sensitivity. By extending the sensitivity-based pruning logic to individual filters within layers, the model's architecture can be further optimized for efficiency without compromising accuracy.

This extension will require a nuanced approach to assess the impact of individual filters on the overall model sensitivity, contributing to a more fine-grained pruning process. Evaluations will be conducted to determine the trade-offs and benefits introduced by this enhanced sensitivity-based pruning.

C. *Single Shot Pruning of Weights and Filters*

Building on the iterative nature of our current methodology, future work will explore the possibility of single shot pruning of both weights and filters. This streamlined approach aims to achieve efficient model compression without the need for multiple iterations.

Single-shot pruning will involve a comprehensive analysis of the network's sensitivity and significance of weights and filters in a unified process. Evaluations will focus on understanding the efficacy of this approach in terms of accuracy preservation and computational efficiency, providing valuable insights into the potential advantages of a more streamlined pruning strategy.

By addressing these future research directions, we aim to advance the field of model pruning, refining our methodology for broader applicability and efficiency gains. These proposed avenues represent exciting opportunities to enhance the scalability and effectiveness of the pruning technique outlined in this paper.

VI. CONCLUSION

In conclusion, the application of filter pruning has proven to be a highly effective strategy for achieving significant cost reduction in terms of model size and computational resources. Our methodology, inspired by "Pruning Filters for Efficient Convnets," successfully identified and removed redundant filters from the VGG16 architecture, resulting in a streamlined model with minimal accuracy loss.

Furthermore, the introduction of weight pruning based on sensitivity has emerged as a powerful extension to our pruning framework. This refinement not only contributed to additional reductions in inference time but also demonstrated the adaptability of our methodology to more fine-grained model optimization. By focusing on the significance of individual weights, we achieved a balance between computational efficiency and accuracy preservation.

The combined impact of filter and weight pruning showcases the potential of our methodology for achieving leaner, more efficient convolutional neural networks. The observed cost reduction, both in terms of model size and inference time, positions our approach as a promising solution for resource-constrained environments.

As we move forward, the promising results obtained in this study pave the way for future investigations. Evaluations on diverse architectures, datasets, and the exploration of single shot pruning strategies will contribute to further advancing the field of model compression. In essence, our research not only underscores the efficiency gains achievable through filter and weight pruning but also sets the stage for continued exploration and refinement of these techniques in pursuit of even more resource-efficient deep learning models.

REFERENCES

[1] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, “Pruning Filters for Efficient ConvNets,” *arXiv:1608.08710 [cs]*, Mar. 2017, Available: https://arxiv.org/abs/1608.08710

[2] L. Namhoon, A. Thalaiyasingam, and T. S. Philip H., “SNIP: Single-shot Network Pruning based on Connection Sensitivity,” *arXiv.org*, Oct. 2018, Available: https://arxiv.org/abs/1810.02340

[3] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In NIPS, 2012.

[5] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In ICLR, 2015.

[6] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. In CVPR, 2015a.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In CVPR, 2016.

[8] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. arXiv preprint arXiv:1512.00567, 2015b.

[9] Yann Le Cun, John S Denker, and Sara A Solla. Optimal Brain Damage. In NIPS, 1989.

[10] Babak Hassibi and David G Stork. Second Order Derivatives for Network Pruning: Optimal Brain Surgeon. In NIPS, 1993.

[11] Suraj Srinivas and R Venkatesh Babu. Data-free Parameter Pruning for Deep Neural Networks. In BMVC, 2015.

[12] Song Han, Jeff Pool, John Tran, and William Dally. Learning both Weights and Connections for Efficient Neural Network. In NIPS, 2015.

[13] Zelda Mariet and Suvrit Sra. Diversity Networks. In ICLR, 2016.

[14] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A Horowitz, and William J Dally. EIE: Efficient Inference Engine on Compressed Deep Neural Network. In ISCA, 2016a

[15] Song Han, Huizi Mao, and William J Dally. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. In ICLR, 2016b.

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In NIPS, 2012.

[17] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In ICLR, 2015

[18] Forrest Iandola, Matthew Moskewicz, Khalidand Ashraf, Song Han, William Dally, and Keutzer Kurt. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡ 1MB model size. arXiv preprint arXiv:1602.07360, 2016

[19] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. ICML, 2018.

[20] Russell Reed. Pruning algorithms-a survey. Neural Networks, 1993.

[21] Yves Chauvin. A back-propagation algorithm with optimal use of hidden units. NIPS, 1989

[22] Andreas S Weigend, David E Rumelhart, and Bernardo A Huberman. Generalization by weight elimination with application to forecasting. NIPS, 1991.

[23] Masumi Ishikawa. Structural learning with forgetting. Neural Networks, 1996.

[24] Michael C Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. NIPS, 1989.

[25] Ehud D Karnin. A simple procedure for pruning back-propagation trained neural networks. Neural Networks, 1990.

[26] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. NIPS, 1990.