Pruning at a Glance: A Structured Class-Blind Pruning Technique for Model Compression

Abdullah Salama, Oleksiy Ostapenko, Moin Nabi, Tassilo Klein
SAP Machine Learning Research
{abdullah.salama, oleksiy.ostapenko, m.nabi, tassilo.klein}@sap.com

Abstract

High performance of deep learning models typically comes at cost of considerable model size and computation time. These factors limit applicability for deployment on memory and battery constraint devices such as mobile phones or embedded systems. In this work we propose a novel pruning technique that eliminates entire filters and neurons according to their relative L1-norm as compared to the rest of the network, yielding more compression and decreased redundancy in the parameters. The resulting network is non-sparse, however, much more compact and requires no special infrastructure for its deployment. We prove the viability of our method by achieving 97.4%, 47.8% and 53% compression of LeNet-5, ResNet-56 and ResNet-110 respectively, exceeding state-of-the-art compression results reported on ResNet without losing any performance compared to the baseline. Our approach does not only exhibit good performance, but is also easy to implement on many architectures.

1 Introduction

While deep learning models have become the method of choice for a multitude of applications, their training requires a large number of parameters and extensive computational costs (energy, memory footprint, inference time). This limits their deployment on storage and battery constraint devices, such as mobile phones and embedded systems. To compress deep learning models without loss in accuracy, previous work proposed pruning weights by optimizing network's complexity using second order derivative information [2, 5]. While second order derivative introduces a high computational overhead, [8, 10] explored low rank approximations to reduce the size of the weight tensors.

Another line of work [4, 15], proposed to prune individual layer weights with the lowest absolute value (nonstructural sparsification of layer weights). [3] followed the same strategy while incorporating quantization and Huffman coding to further boost compression. While the aforementioned methods considered every layer independently, [13] proposed to prune the network weights in a class-blind manner, e.g. individual layer weights are pruned according to their magnitude as compared to all weights in the network.

Noteworthy, all approaches that prune weights non-structurally, generally result in high sparsity models that require dedicated hardware and software. Structured pruning alleviates this by removing whole filters or neurons, producing a non-sparse compressed model. In this regard, [12] proposed channel-wise pruning according to the L1-norm of the corresponding filter. [16] learned a compact model based on learning structured sparsity of different parameters. A data-free algorithm was implemented to remove redundant neurons iteratively on fully connected layers in [14]. In [7], connections leading to weak activations were pruned. Finally, [17] pruned neurons by measuring their importance with respect to the penultimate layer. Generally, in structured pruning, each layer is pruned separately, which requires calculation of layer importance before training.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

This work features two key components: a) **Blindness**: all layers are considered simultaneously; blind pruning was first introduced by [13] to prune individual weights; b) **Structured Pruning**: removal of entire filters instead of individual weights. To the best of our knowledge, we are the first to use these two components together to prune filters based on their relative L1-norm compared to the sum of all filters' L1-norms across the network, instead of pruning filters according to their L1-norm within the layer [12], inducing a global importance score for each filter. The contribution of this paper is two-fold: **i)** Proposing a structured class-blind pruning technique to compress the network by removing whole filters and neurons, which results in a compact non-sparse network with the same baseline performance. **ii)** Introducing a visualization of global filter importance to devise the pruning percentage of each layer.

As a result, the proposed approach achieves higher compression gains with higher accuracy compared to the state-of-the-art results reported on ResNet-56 and ResNet-110 on the CIFAR10 dataset [9].

2 Structured class-blind pruning

Consider a network with a convolutional (conv) layer and a fully connected (fc) layer. We denote each filter $Filter_i$, where $i \in [1, F]$, and F is the total number of filters in the conv layer. Each filter is a 3D kernel space consisting of channels, where each channel contains 2D kernel weights. For the fc layer, we denote W_m, a 1-D feature space containing all the weights connected to certain neuron $Neuron_m$, with $m \in [1, N]$ and N denoting the number of neurons. It should be noted that We do not prune the classification layer.

Each pruning iteration in our algorithm is structured as follows:

Algorithm 1 Pruning procedure

1: **for** $i \leftarrow 1$ to F **do** ▷ loop over filters of a conv layer
2: $L1\_conv(i) \leftarrow sum(|Filter_i|)$ ▷ calculate L1-norm of all channels' kernel weights
3: $norm\_conv(i) \leftarrow L1\_conv(i)/size(Filter_i)$ ▷ normalize by filter weights count
4: **for** $m \leftarrow 1$ to N **do** ▷ loop over Neurons of a fc layer
5: $L1\_fc(m) \leftarrow sum(|W_m|)$ ▷ for each Neuron, calculate L1-norm of incoming weights
6: $norm\_fc(m) \leftarrow L1\_fc(m)/size(W_m)$ ▷ normalize by number of weights connected
7: $norms \leftarrow stack(norm\_conv, norm\_fc)$ ▷ stack all normalized norms from all layers
8: $sorted \leftarrow sort(norms)$ ▷ sort ascendingly
9: $threshold \leftarrow perc(sorted, p)$ ▷ threshold based on a percentage p of sorted norms values
10: **for** $i \leftarrow 1$ to F **do**
11: **if** $norm\_conv(i) < threshold$ **then**
12: $prune(Filter_i)$ ▷ remove filter if its normalized norm is less than threshold
13: **for** $m \leftarrow 1$ to N **do**
14: **if** $norm\_fc(m) < threshold$ **then**
15: $prune(Neuron_m)$ ▷ remove neuron if its normalized norm is less than threshold

Importance calculation. Although pre-calculation of filters or layers' sensitivity to be pruned is not needed in our method, it can be visualized as part of the pruning criteria. In our algorithm, blindness implies constructing a hidden importance score, which corresponds to the relative normalized L1-norm. For instance, the relevant importance for a certain filter in a conv layer w.r.t. all other filters in all layers is the ratio between the filter's normalized norm and the sum of all filters' normalized norms across the network.

Normalization. As each layer's filters have different number of kernel weights, we normalize filters' L1-norms by dividing each over the number of kernel weights corresponding to the filter (Line 3 and 6 as indicated in Algorithm 1). Alternatively without normalization, filters with a higher number of kernel weights would have higher probabilities of higher L1-norms, hence lower probability to get pruned.

Retraining process. Pruning without further adaption, results in performance loss. Therefore, in order to regain base performance, it is necessary for the model to be retrained. To this end, we apply an iterative pruning schedule that alternates between pruning and retraining. This is conducted until a maximum compression is reached without losing the base accuracy.

	Method	Error%	Par.%
ResNet-56	Baseline	6.96	
	Li et al. [12]-A	6.90	9.40
	Li et al. [12]-B	6.94	13.70
	NISP[17]	6.99	42.60
	Ours	**6.88**	**47.86**
ResNet-110	Baseline	6.47	
	Li et al. [12]-A	6.45	2.30
	Li et al. [12]-B	6.70	32.40
	NISP[17]	6.65	43.25
	Ours	**6.44**	**53.06**

Table 1: Compression Benchmark Results. Error% percentage for different percentage of parameters pruned (Par.%)

Method	Error%	Par.%	E.Par.%
Baseline	0.80		
Non-Structured	0.77	93.04	86.08
Non-Blind	0.76	89.80	89.80
Ours-Oneshot	0.80	96.06	96.06
Ours	0.75	97.40	**97.40**
Han et al. [4]	0.77	92.00	84.00
Srinivas et al. [15]	0.81	95.84	91.68
Han et al. [3]	**0.74**	**97.45**	-

.

Table 2: Results on LeNet-5. Error% percentage for different percentage of parameters pruned (Par.%); "E.Par%" is the effective pruning percentage after adding the extra indices' storage for non-structured pruning as studied by [1]

3 Experiment

In order to assess the efficacy of the proposed method, we evaluate the performance of our technique on a set of different networks: first, LeNet-5 on MNIST [11]; second, ResNet-56 and ResNet-110 ([6]) on CIFAR-10 [9]. We use identical training settings as [6], after pruning we retrain with learning rate of 0.05.

For ResNet, when a filter is pruned, the corresponding batch-normalization weight and bias applied on that filter are pruned accordingly. After all pruning iterations are finished, a new model with the remaining number of parameters is created.

We report compression results on the existing benchmark [12, 17]. As shown in Table 1, we outperform the state-of-the-art compression results reported by [17] on both ResNet-56 and ResNet-110 with a lower classification error even compared to the baseline.

In Table 2, while using one-shot pruning, the influence of our method's different components; structured pruning and blindness, is analyzed by removing a component each test, resulting in: **i)** Non-Structured - pruning applied on weights separately. **ii)** Non-Blind - every layer is pruned individually. Then, the effect of the pruning strategy on the method with all its components is analyzed by comparing: **i)** Ours-Oneshot - using one-shot pruning and **ii)** Ours - using iterative pruning.

By comparing the previous versions that are using one-shot pruning, our method has less number of parameters compared to the other versions; ("Non-Structured" and "Non-Blind").

Finally, applying pruning iteratively is superior to one-shot pruning. We also show that our method performs better than previously mentioned non-structured weight pruning techniques [4, 15]. Proposed structured class-blind pruning offers comparable performance as [3], without requiring dedicated hardware and software to realize compression.

4 Conclusion

We presented a novel structured pruning method to compress neural networks without losing accuracy. By pruning layers simultaneously instead of looking at each layer individually, our method combines all filters and output features of all layers and prunes them according to a global threshold. We have surpassed state-of-the-art compression results reported on ResNet-56 and ResNet-110 on CIFAR-10 [17], compressing more than 47% and 53% respectively. Also, we showed that only 11K parameters are sufficient to exceed the baseline performance on LeNet-5, compressing more than 97%. To realize the advantages of our method, no customized hardware or libraries are needed. It is worth to say that due to removing whole filters and neurons, the pruning percentage reflects the effective model compression percentage. For the future work, we are dedicated to proving the applicability of our method on several different architectures and datasets. Hence, we plan to experiment on VGG-16, ResNet on ImageNet and/or other comparable architectures.

References

[1] M. D. Collins and P. Kohli. Memory bounded deep convolutional networks. *CoRR*, abs/1412.1442, 2014. URL http://arxiv.org/abs/1412.1442.

[2] Y. L. Cun, J. S. Denker, and S. A. Solla. Advances in neural information processing systems 2. chapter Optimal Brain Damage, pages 598–605. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1-55860-100-7. URL http://dl.acm.org/citation.cfm?id=109230.109298.

[3] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR*, abs/1510.00149, 2015. URL http://arxiv.org/abs/1510.00149.

[4] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015. URL http://arxiv.org/abs/1506.02626.

[5] B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pages 164–171, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc. ISBN 1-55860-274-7. URL http://dl.acm.org/citation.cfm?id=645753.668069.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

[7] H. Hu, R. Peng, Y. Tai, and C. Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *CoRR*, abs/1607.03250, 2016. URL http://arxiv.org/abs/1607.03250.

[8] Y. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *CoRR*, abs/1511.06530, 2015. URL http://arxiv.org/abs/1511.06530.

[9] A. Krizhevsky, V. Nair, and G. Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

[10] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *CoRR*, abs/1412.6553, 2014. URL http://arxiv.org/abs/1412.6553.

[11] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov 1998. ISSN 0018-9219. doi: 10.1109/5.726791.

[12] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2017. URL http://arxiv.org/abs/1608.08710.

[13] A. See, M. Luong, and C. D. Manning. Compression of neural machine translation models via pruning. *CoRR*, abs/1606.09274, 2016. URL http://arxiv.org/abs/1606.09274.

[14] S. Srinivas and R. V. Babu. Data-free parameter pruning for deep neural networks. *CoRR*, abs/1507.06149, 2015. URL http://arxiv.org/abs/1507.06149.

[15] S. Srinivas, A. Subramanya, and R. V. Babu. Training sparse neural networks. *CoRR*, abs/1611.06694, 2016. URL http://arxiv.org/abs/1611.06694.

[16] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. *CoRR*, abs/1608.03665, 2016. URL http://arxiv.org/abs/1608.03665.

[17] R. Yu, A. Li, C. Chen, J. Lai, V. I. Morariu, X. Han, M. Gao, C. Lin, and L. S. Davis. NISP: pruning networks using neuron importance score propagation. *CoRR*, abs/1711.05908, 2018. URL http://arxiv.org/abs/1711.05908.