Hybrid Pruning: Thinner Sparse Networks for Fast Inference on Edge Devices

Xiaofan Xu[*]   **Mi Sun Park**[*]   **Cormac Brick**
Movidius, AIPG, Intel
xu.xiaofan@intel.com   mi.sun.park@intel.com

Abstract

We introduce hybrid pruning which combines both coarse-grained channel and fine-grained weight pruning to reduce model size, computation and power demands with no to little loss in accuracy for enabling modern networks deployment on resource-constrained devices, such as always-on security cameras and drones. Additionally, to effectively perform channel pruning, we propose a *fast sensitivity test* that helps us quickly identify the sensitivity of within and across layers of a network to the output accuracy for target multiplier–accumulators (MACs) or accuracy tolerance. Our experiment shows significantly better results on ResNet50 on ImageNet compared to existing work, even with an additional constraint of channels be hardware-friendly number.

1   Introduction

Recently, with the increasing size of large-scale datasets and high-end GPUs, lots of unprecedented large DNN models were developed. Although these modern networks, such as VGG[13], GoogleNet[14], DenseNet[8] and ResNets[4] show outstanding classification performance on ImageNet (even better than human), they are not easily deployable on resource-constrained inference devices due to large demands for memory, compute and power.

Network Pruning is to remove some redundant weights (or channels) which don't contribute a lot to the output of a network. As a result, it reduces model size and helps preventing over-fitting, and eventually generate sparse (or thinner) model. Weight pruning [3][2] show high compression rate on AlexNet (9 to 17.7×) by pruning redundant weight or additionally allowing splicing of previously pruned weights. Channel-pruning work [9][12][11][7][10] naively prune equal percentage of number of channels based on their own calculation of importance of channels. Although the one goal of identifying importance of channels, remove those that are relatively less important and fine-tune the pruned network is the same for all these works, none of these work has considered the fact that each network has different sensitivity within and across layers to output accuracy. Therefore, evenly prune 50% of channels in all the layers results in significant accuracy drop. To overcome the limitation, we propose a fast sensitive test for identifying corresponding number of channels in each layer for a given accuracy tolerance or target MACs. We also apply weight pruning on channel pruned thinner model to further reduce model size and computation. This hybrid pruning eventually generates thinner sparse model. To the best of our knowledge, this is the first to show that the coexistence of multi granularity of sparsity can help significantly reduce resource demands with no significant loss in accuracy, and gives SOTA result on thinner ResNet50 on ImageNet, with channels be multiple of 8.

2   Hybrid Pruning

In this work, we applied both coarse-grained channel and fine-grained weight pruning on convolutional layers for various types of neural networks. We call this combination hybrid pruning. Figure 1

---

[*]Equal contribution

illustrates the process for one convolutional layer using hybrid pruning resulting in a thinner sparse model.
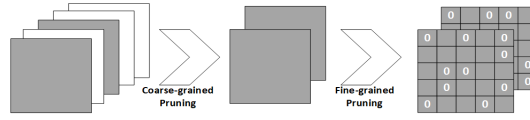

Figure 1: Pictorial view of our proposed hybrid pruning

Sensitivity-aware channel pruning: In general, one big challenge in channel pruning is that if we naively decide the number of channels to prune away based on target sparsity, it will leads to significant loss in accuracy as shown in [9][12][11][7][10] regardless of their filter or channel selection. For ResNet50 on ImageNet, [7] and [11] reported that for 50% compression in parameters gives 1.4 to 1.1 drop @top5 which is significant loss considering this is top5, not even top1. As Algorithm 1 shows, our proposed fast sensitivity test helps us quickly identify the sensitivity of each layer in a network for a given accuracy tolerance. The basic idea is to find the upper bound in the percentage of the number of intact output channels in each layer with potential recoverable loss in accuracy with fine-tuning. This was inspired by the observation that when we damage the network too much by pruning more than acceptance, it becomes more difficult to recover its accuracy. From our experiments, accuracy tolerance of 3 to 5% works well for ResNet50 on ImageNet. Furthermore, since we know number of channels and required MACs in an original network and the relative sensitivity of each layer from the test, we can calculate the final number of channels for target MACs by multiplying the pruning percentage with the original channel number, with an additional constraint of channels be hardware-friendly number. It should be also noted that the sensitivity test itself does not require any training and it only takes 8.86 minutes on 1-CPU Intel Core i7-6850K CPU and 3.38 seconds on GPU GTX-1080-Ti Pascal for conducting this test on ResNet50 on ImageNet. Unlike the existing works AutoML[6] and[1], our method doesn't require any extra training for new models or meta-learning, and it is compatible with any frameworks. It can help users to quickly identify the pruning channels without large GPU resources.

Algorithm 1 Fast sensitivity test for channel pruning

Input: Validation data and a dense model M.
Output: Pruning percentage for each layer in the model M.
1: Threshold accuracy = original dense accuracy - accuracy tolerance (e.g. 3 - 5%)
2: **for** each layer in the model M **do**
3:     sort output channels based on the sum of absolute weight values
4:     **for** sparsity percentage in between 30% - 80% **do**
5:         channel-wise mask is created based on the current sparsity percentage
6:         accuracy = **Forward** the network with $channel \times mask$
7:         **if** accuracy > Threshold accuracy **then**
8:             continue;
9:         **else**
10:             Record the **per-layer** percentage and exit the for loop
11:     **end for**
12: **end for**
Note: Final number of channels per-layer = round((100% - pruning % per-layer) $\times$ original number of channels per-layer), with round(.) to be multiple of 8 or 4.

Statistics-aware weight pruning: After generating thinner dense model from sensitivity-aware channel pruning, we apply statistics-aware weight pruning on the thinner model for further reduction in model size and computation. The basic idea is to compute layer-wise weight threshold based on the current statistical distribution of full dense weights in each layer and mask out those weights that are less than the threshold. To do so, we use layer-wise $mask_l^n$ to represent the binary mask for l^{th} layer at n^{th} iteration, shown in Equation 1. This binary mask is dynamically updated in each iteration of training based on the threshold that is computed based on the mean and standard deviation of the weights in each layer with sparsity level controlling factor σ (σ is the same for all the layers). Since we sparsify weights in forward pass and update full dense weights with the gradients computed with sparse weights during training, it allows previously pruned weights back if it becomes more important ($|W_l^n(i,j)| > t_l^n$), similar to [2]. Our experiment shows that our statistics-aware method works better than sparsifying all the layers with the same sparsity level.

$$mask_l^n(i,j) = \begin{cases} 0 & \text{if } |W_l^n(i,j)| < t_l^n \\ 1 & \text{if } |W_l^n(i,j)| \geq t_l^n \end{cases}, \quad t_l^n = mean(|W_l^n|) + std(|W_l^n|) \times \sigma \qquad (1)$$

3 Experiments

In this section, we show the effectiveness of our proposed fast sensitivity test for channel pruning that can give instant speedup on existing hardware, and moreover demonstrate the coexistence of multi granularity of sparsity can help significantly reduce resource demands for fast inference on edge devices. For fair comparison, similar to previous work [9][11], we do not prune the first Conv, the last FC layer and the last layer in each residual block.

ResNet56 on Cifar10: Table1 shows the result of our hybrid pruning on (bottleneck based) ResNet56. Our hybrid pruning contains the channel pruning shown in Figure2, it should be noted that in Figure2 and 3, the first Conv, the last FC and the last layer in each residual block are omitted for clarity, since these layers are intact and have no reduction in output channels, compared to original network (blue dots). The number of survived channels (red dots) in each layer are predetermined from the sensitivity test with accuracy tolerance of 2% to be conservative in accuracy loss. Furthermore, we added an additional constraint of channels be multiple of 4 for optimal hardware utilization. As a result, most of the layers can be pruned up to 75% with no significant loss in accuracy. It is also observed that those few layers that have increased output channels (compared to previous layer) are more sensitive. After achieving $2.4\times$ instant speedup based on channel pruning, our hybrid pruning gives final $4.5\times$ reduction in parameters (additional $1.8\times$ reduction on thinner ResNet56) with less than 1% accuracy drop. Hybrid pruning is only applied on the convolutional layers to boost the model to 78% sparsity and it can be further pruned by applying weight pruning on the FC layer.
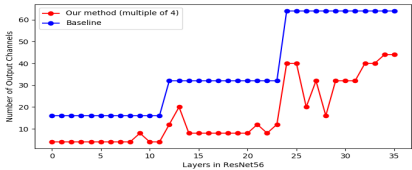

Figure 2: Number of output channels in ResNet56

Table 1: Hybrid pruning on ResNet56 on Cifar10 with comparison to other works

	Top 1 (%) Baseline →Pruned	No. of Params Baseline→Pruned	Sparsity (%) Baseline→Pruned
PF[9]	93.04→91.31	0.73M→-	0→13.7
SFP[5]	93.59→92.26	-	-
CP[7]	92.8→91.8	-	-
Our Channel Pruning	93.27→92.67	0.59M→0.24M	0→59
Our Hybrid Pruning	93.27→**92.48**	0.59M→**0.13M**	0→**78**

ResNet50 on ImageNet: Figure3 shows the output channels of pruned ResNet50 using ImageNet. The experiment was intended for $2\times$MAC reduction based on the sensitivity test with a constraint of channels be multiple of 8. It is observed that, similar to ResNet56, those few layers that have increased output channels (compared to previous layer) are more sensitive. Interestingly, unlike ResNet56, the last few layers that have the most number of channels are also sensitive. From this observation, we can infer that although network structure looks similar, it might be safe to assume that each network has different sensitivity and it's also different across layers within a network, therefore it's important to have a such technique that can help us to quickly identify the sensitivity of a network. As Table 2 shows, our sensitivity based channel pruning method outperformed the existing naive methods with large margin. Even our hybrid ResNet50 gives better accuracy with less number of parameters ($3.7\times$ reduction in parameters). In addition to parameter storage and bandwidth savings, both key for edge devices, our hybrid model can further boost the performance on any hardware with sparse matrix support.
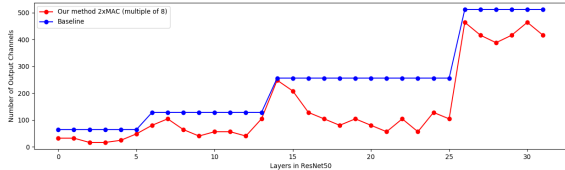

Figure 3: Number of output channels in ResNet50

Table 2: Hybrid Pruning on ResNet50 on ImageNet with comparison to other works

	Top 1 (%) Baseline	Top 1 (%) Channel Pruning	Top 1 (%) Hybrid Pruning	Top 5 (%) Baseline	Top 5 (%) Channel Pruning	Top 5 (%) Hybrid Pruning	Original No. of Params	No. of Params (Coarse-grained Pruning)	No. of Params (Hybrid Pruning)
CP[7]	-	-	-	92.2	90.8	-	-	2× reduction	-
NISP[16]	-	0.21% reduction	-	-	-	-	-	27.12% reduction	-
SPP[15]	-	-	-	91.2	90.4	-	-	2× reduction	-
ThiNet[11]	72.88	71.01	-	91.14	90.02	-	25.5M	12.38M	-
SFP[5]	76.15	74.61	-	92.87	92.06	-	-	30% reduction	-
Ours	76.01	**74.87**	**74.32**	92.93	**92.43**	**92.05**	25.5M	17.2M (32.5% reduction)	**6.9M (72.9% reduction)**

References

[1] T.-W. Chin, C. Zhang, and D. Marculescu. Layer-compensated pruning for resource-constrained convolutional neural networks. *arXiv preprint arXiv:1810.00518*, 2018.

[2] Y. Guo, A. Yao, and Y. Chen. Dynamic network surgery for efficient dnns. In *Advances In Neural Information Processing Systems*, pages 1379–1387, 2016.

[3] S. Han, J. Pool, J. Tran, and W. Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

[6] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018.

[7] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[8] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[9] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016.

[10] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017.

[11] J.-H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. *arXiv preprint arXiv:1707.06342*, 2017.

[12] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient transfer learning. *CoRR*, abs/1611.06440, 2016.

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[15] H. Wang, Q. Zhang, Y. Wang, and H. Hu. Structured probabilistic pruning for convolutional neural network acceleration. *arXiv preprint arXiv:1709.06994*, 2017.

[16] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V. I. Morariu, X. Han, M. Gao, C.-Y. Lin, and L. S. Davis. Nisp: Pruning networks using neuron importance score propagation. *Preprint at https://arxiv.org/abs/1711.05908*, 2017.