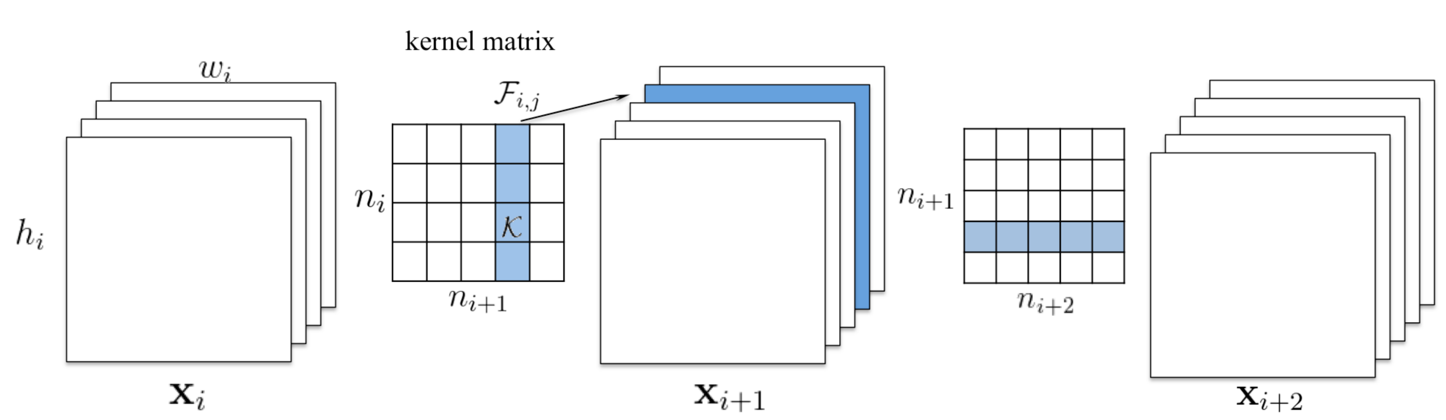
Pruning Filters for Efficient ConvNets

Hao Li[1], Asim Kadav[2], Igor Durdanovic[2], Hanan Samet[1], Hans Peter Graf[2]
University of Maryland[1], NEC Labs America[2]

Motivation

- Low computation cost of CNNs is a crucial factor for mobile applications and cloud services.
- Convolutional layers dominate computation and storage costs in state-of-art CNNs [1].
- Pruning small weights [2] mostly reduces the storage cost of parameters from FC layers and require sparse convolutions.



Pruning filters and their corresponding feature maps is a natural way to reduce the computation and storage cost of conv layers without introducing sparse kernels.

Contributions

✓ Reduce the inference computation cost of CNNs by pruning filters avoiding the need for sparse convolution libraries.

✓ Simple criterion for filter selection, without examining each feature map's importance [3,4].

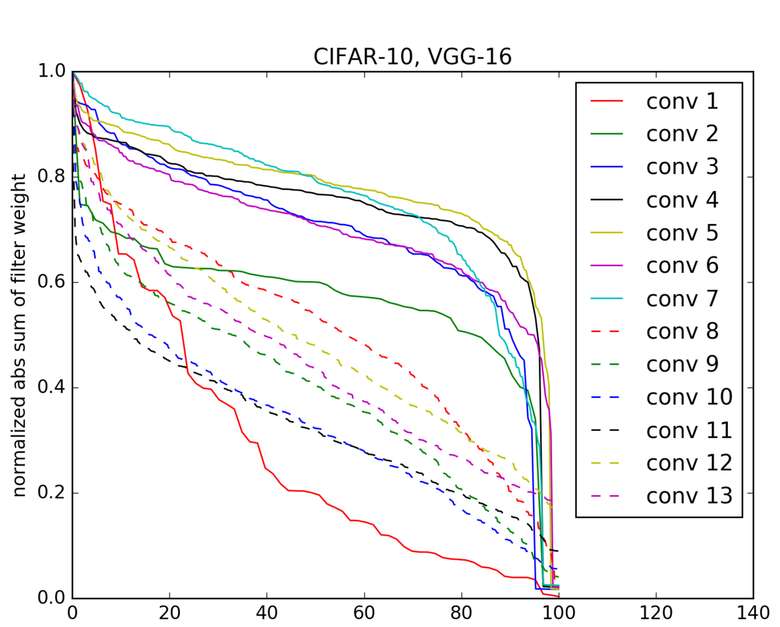
✓ Prune multiple filters together and retrain once, avoiding iterative pruning and retraining.

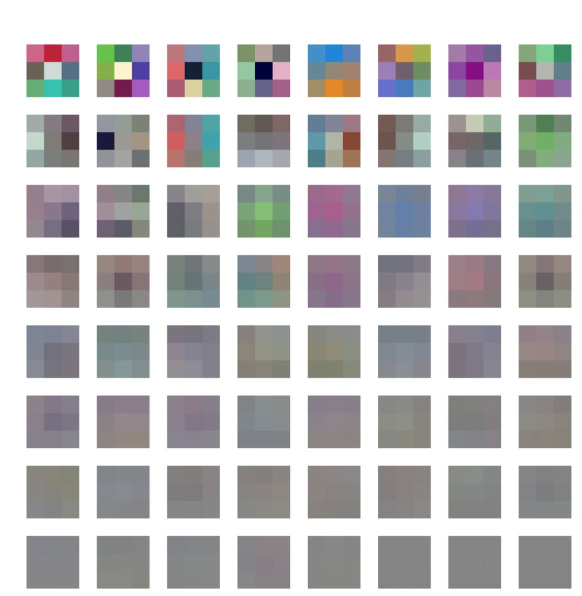
Determine Filters' Importance

➢ For each conv layer, we measure each filter's relative importance by its absolute weight sum $\sum|\mathcal{F}_{i,j}|$, i.e., its ℓ_1-norm. This value also represents the average magnitude of its weights.

➢ Filters with small weights tend to produce feature maps with weak activations.

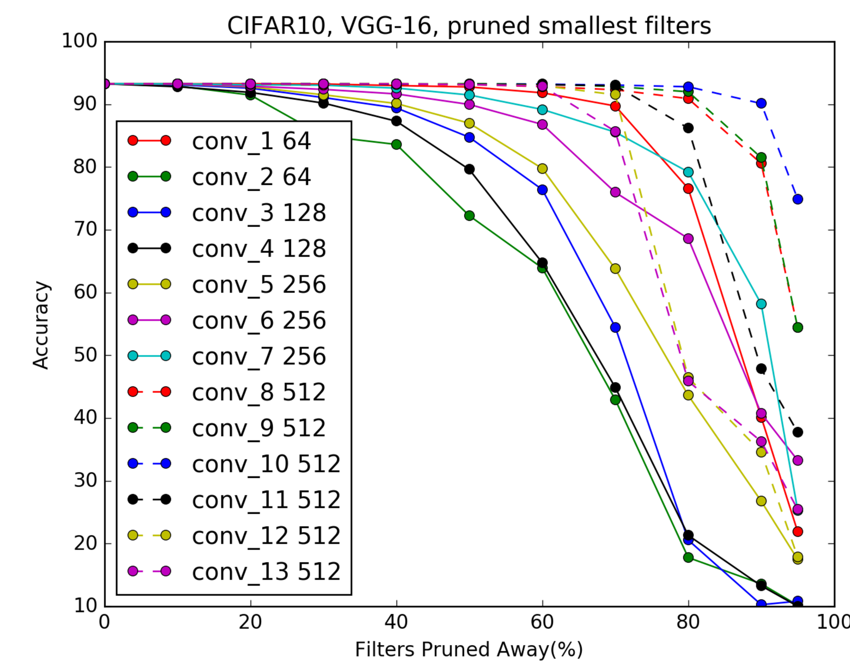
➢ Pruning the *smallest* filters works better in comparison with pruning the same number of *random* or *largest* filters.



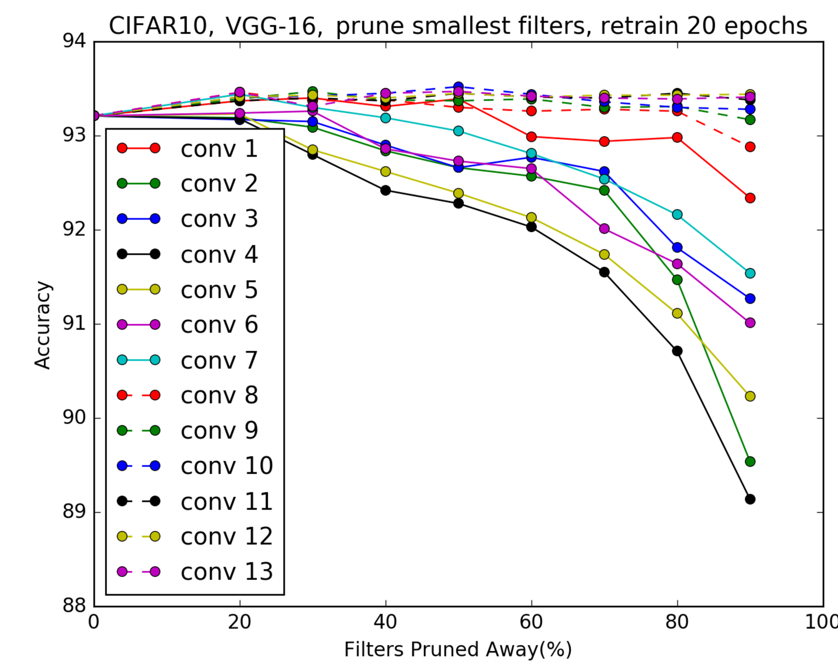


Filters are ranked by their abs weight sum. (Left) The y-axis is the filter weight sum divided by the max value among filters in that layer. (Right) Visualization of filters in the first conv layer of VGG-16 trained on CIFAR-10.

Determine Single Layer's Sensitivity to Pruning



Pruning the smallest filters of single layer

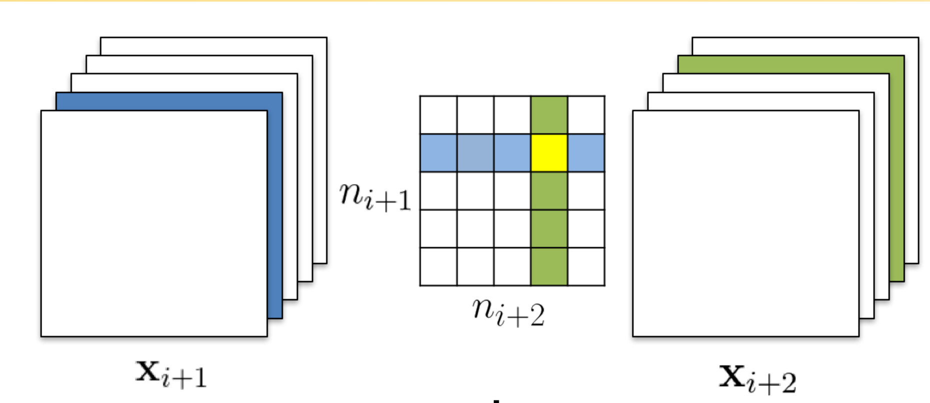


Regain accuracy by retraining

Pruning ratios

- Layers with the same input sizes often have similar sensitivity to pruning. We use the same pruning ratio for these layers to avoid tuning layer-specific meta-parameters.
- For layers that are sensitive to pruning, we use a small pruning rate or completely skip pruning them.

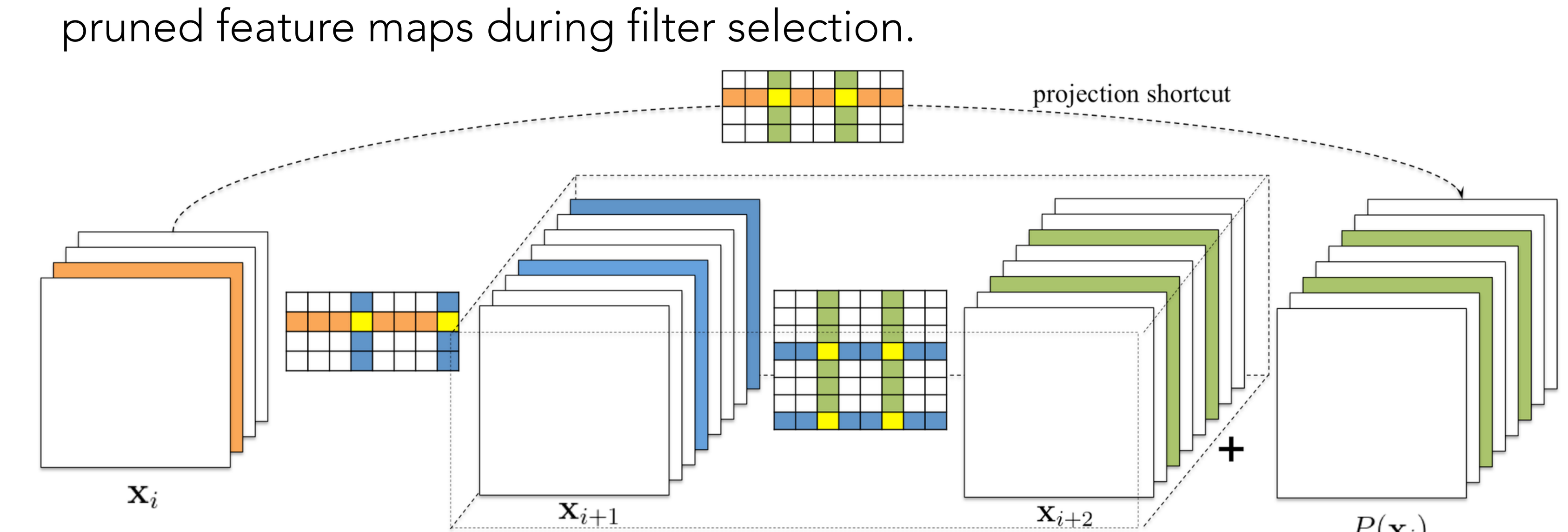
Prune Filters across Multiple Layers



Pruning filters across consecutive layers

✗ *Independent* pruning determines filters to be pruned at one layer independent of other layers.

✓ *Greedy* pruning does not count kernels connected with the previously pruned feature maps during filter selection.



Pruning residual blocks with projection shortcut

- The first layer of the residual block can be pruned without restrictions.
- The filters to be pruned in the second conv layer of the residual blocks is determined by the pruning result of the shortcut projection.

Retrain Pruned Networks

Instead of *iterative* pruning and retraining, we adopt a ***one-shot*** pruning and retraining strategy (~¼ of the original training time).

Experiments

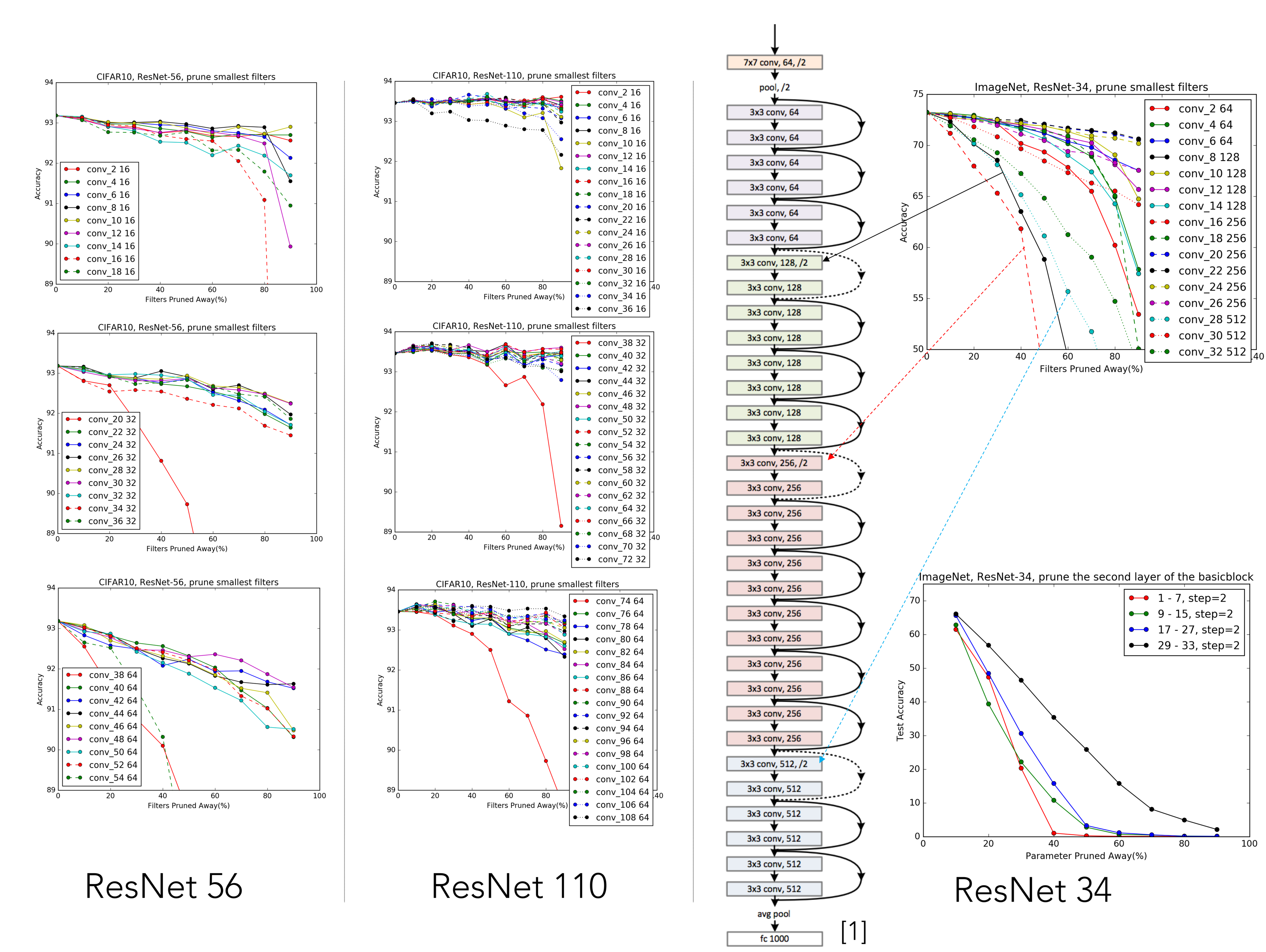
	Model	Error	FLOP	Pruned	Parameters	Pruned
CIFAR-10	VGG-16	6.75	3.13×10^8		1.5×10^7	
	VGG-16-pruned-A	**6.60**	2.06×10^8	34.2%	5.4×10^6	64%
	VGG-16-pruned-A scratch train	6.88				
	ResNet-56	6.96	1.25×10^8		8.5×10^5	
	ResNet-56-pruned-A	6.90	1.12×10^8	10.4%	7.7×10^5	9.4%
	ResNet-56-pruned-B	**6.94**	9.09×10^7	27.6%	7.3×10^5	13.7%
	ResNet-56-pruned-B scratch train	8.69				
	ResNet-110	6.47	2.53×10^8		1.72×10^6	
	ResNet-110-pruned-A	**6.45**	2.13×10^8	15.9%	1.68×10^6	2.3%
	ResNet-110-pruned-B	6.70	1.55×10^8	38.6%	1.16×10^6	32.4%
	ResNet-110-pruned-B scratch train	7.06				
ILSVRC12	ResNet-34	26.77	3.64×10^9		2.16×10^7	
	ResNet-34-pruned-A	27.44	3.08×10^9	15.5%	1.99×10^7	7.6%
	ResNet-34-pruned-B	27.83	2.76×10^9	24.2%	1.93×10^7	10.8%
	ResNet-34-pruned-C	27.52	3.37×10^9	7.5%	2.01×10^7	7.2%

Overall results

- ~30% reduction in FLOPs for VGG-16 (on CIFAR-10) and ResNets without significant loss in accuracy.
- Training a pruned model from scratch performs worse than retraining a pruned model.
- Pruning the first layer of the residual block is more effective.

Sensitivity analysis

- For ResNets, layers that are sensitive to pruning are close to the residual blocks where the number of feature maps changes.



[1] He et al. Deep Residual Learning for Image Recognition. CVPR 2017
[2] Han et al. Learning both Weights and Connections for Efficient Neural Network. NIPS 2015
[3] Polyak and Wolf. Channel-Level Acceleration of Deep Face Representations. IEEE Access, 2015.
[4] Anwar et al. Structured Pruning of Deep Convolutional Neural Networks. arXiv 2015.

5th International Conference on Learning Representations (**ICLR**) 2017, Toulon, France