Network Pruning

Papers

- Song Han, Jeff Pool, John Tran, William J. Dally: *Learning both Weights and Connections for Efficient Neural Network*. **NIPS 2015**

- Jiecao Yu, Andrew Lukefahr, David Palframa, Ganesh Dasika, Reetuparna Das, Scott Mahlke: *Scalpel: Customizing DNN Pruning to the Underlying Hardware Parallelism.* **ISCA 2017**

- Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, William J. Dally: *Exploring the Regularity of Sparse Structure in Convolutional Neural Networks.* **NIPS 2017**

- T.-J. Yang, Y.-H. Chen, V. Sze: *Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning*. **CVPR 2017**.

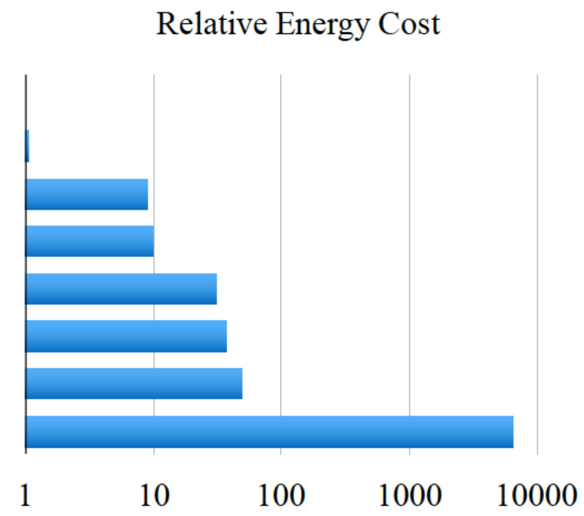
Song Han, Jeff Pool, John Tran, William J. Dally

Learning both Weights and Connections for Efficient Neural Network

NIPS 2015

Smaller models are better in terms of energy, as they reduce DRAM access

Operation	Energy [pJ]	Relative Cost
32 bit int ADD	0.1	1
32 bit float ADD	0.9	9
32 bit Register File	1	10
32 bit int MULT	3.1	31
32 bit float MULT	3.7	37
32 bit SRAM Cache	5	50
32 bit DRAM Memory	**640**	**6400**

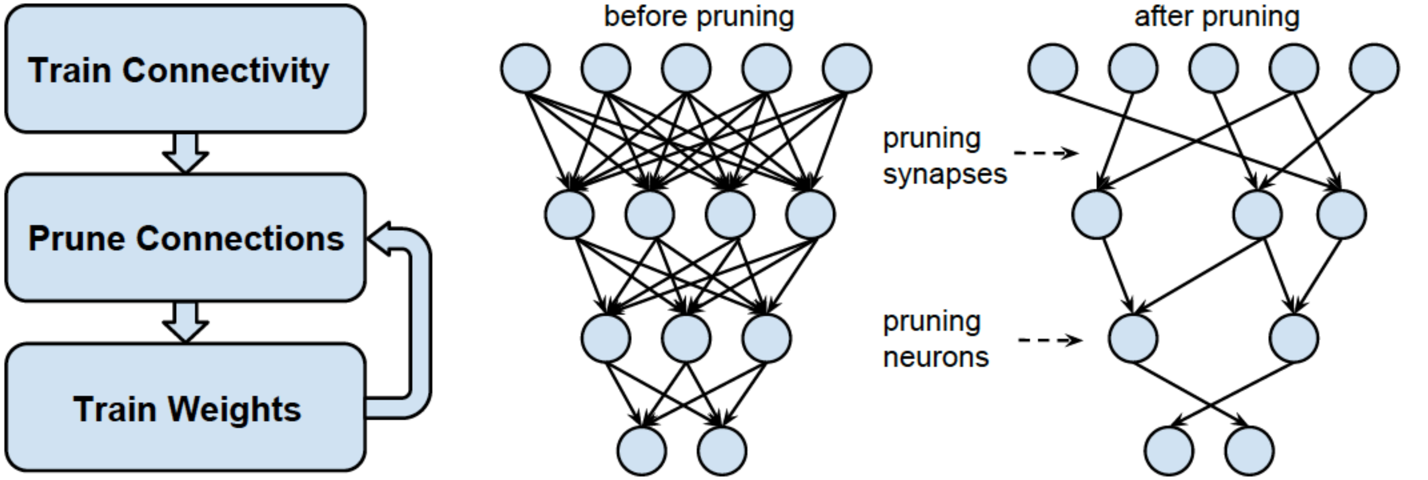


Additional good point about mobile app size (Playstore restrictions, communication costs)

Main intuition in reducing model size: **DNNs have redundancy.** So it is good to identify what connections are important and only retain those, to reduce model size.

Magnitude of weights that a connection gets after training, is taken as a proxy for importance. Connections with lower weights are removed. Removing connections is called “Pruning”.

Magnitude based pruning



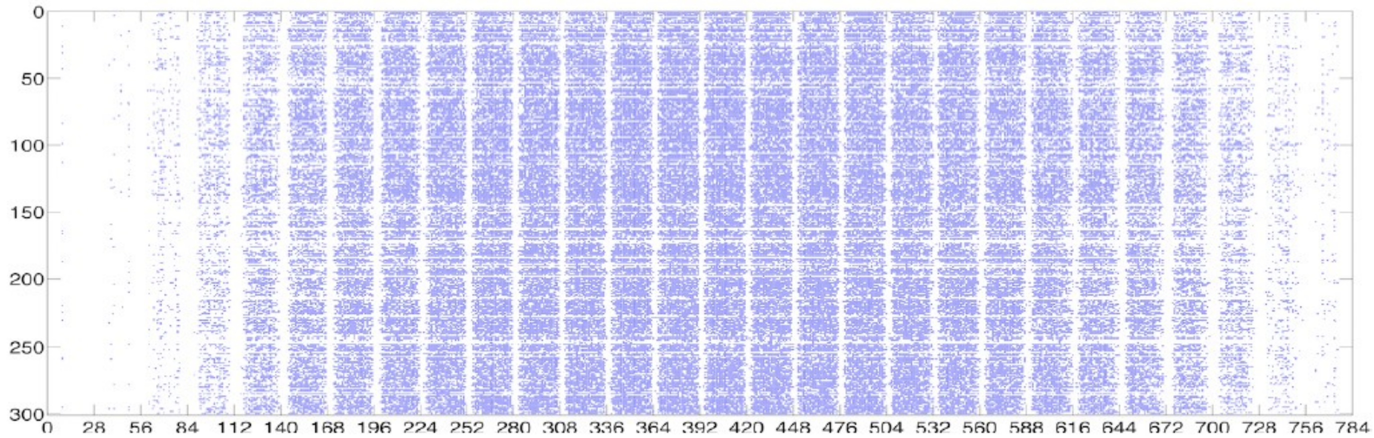
The first step of "Train Connectivity" do not need to run to network convergence. This is inspired by how strong and weak connections are developed in brain.

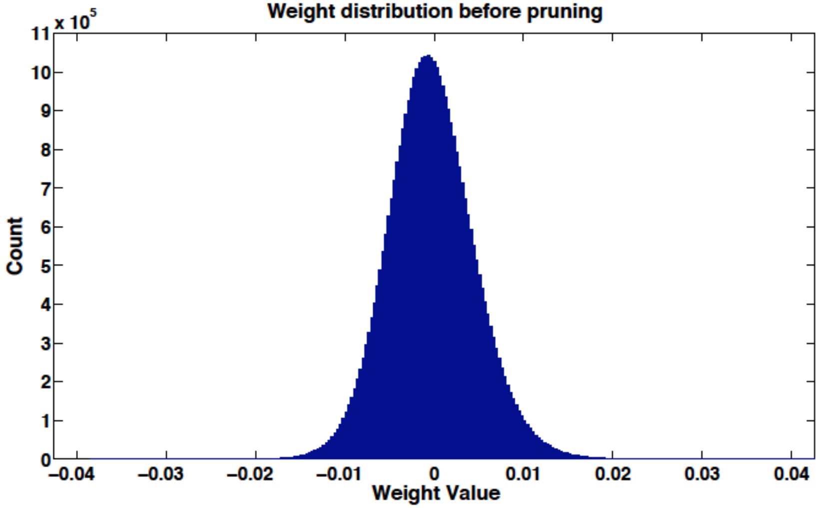
Effect of pruning (implemented in Caffe)

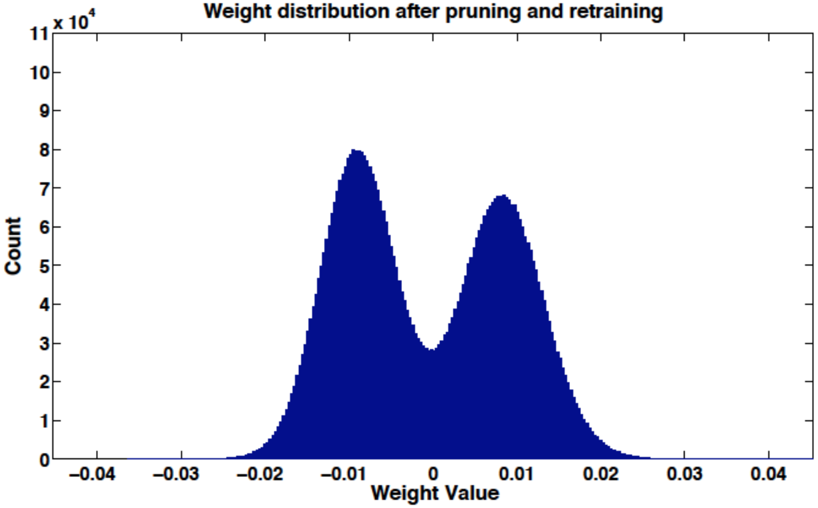
Network	Top-1 Error	Top-5 Error	Parameters	Compression Rate
LeNet-300-100 Ref	1.64%	-	267K	
LeNet-300-100 Pruned	1.59%	-	**22K**	**12**$\times$
LeNet-5 Ref	0.80%	-	431K	
LeNet-5 Pruned	0.77%	-	**36K**	**12**$\times$
AlexNet Ref	42.78%	19.73%	61M	
AlexNet Pruned	42.77%	19.67%	**6.7M**	**9**$\times$
VGG-16 Ref	31.50%	11.32%	138M	
VGG-16 Pruned	31.34%	10.88%	**10.3M**	**13**$\times$

Evidence of this main intuition in reducing model size:
DNNs have redundancy, as accuracy drop is minimal.

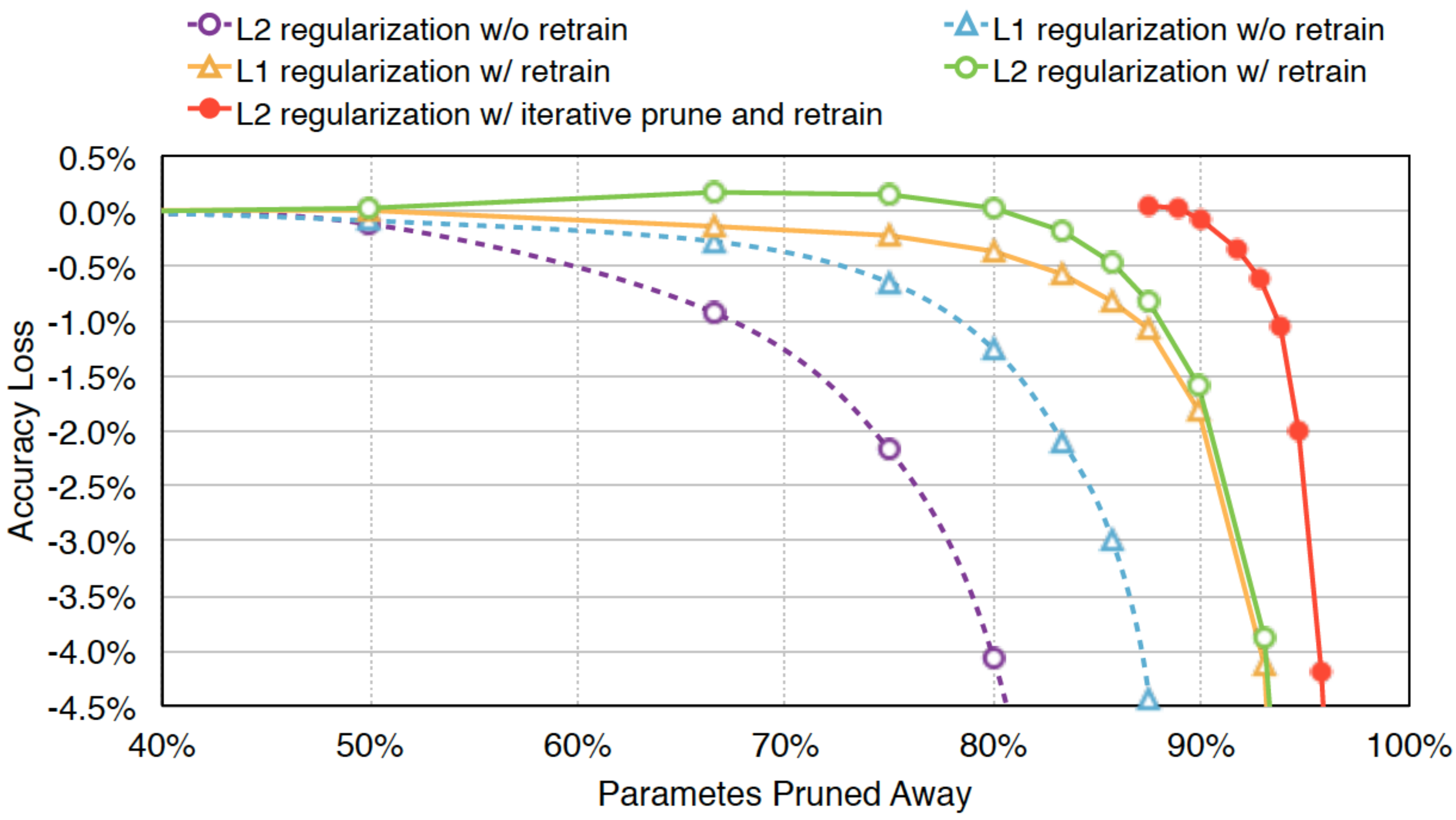
What is retained?







Accuracy-pruning trade-off, effect of retraining and regularization



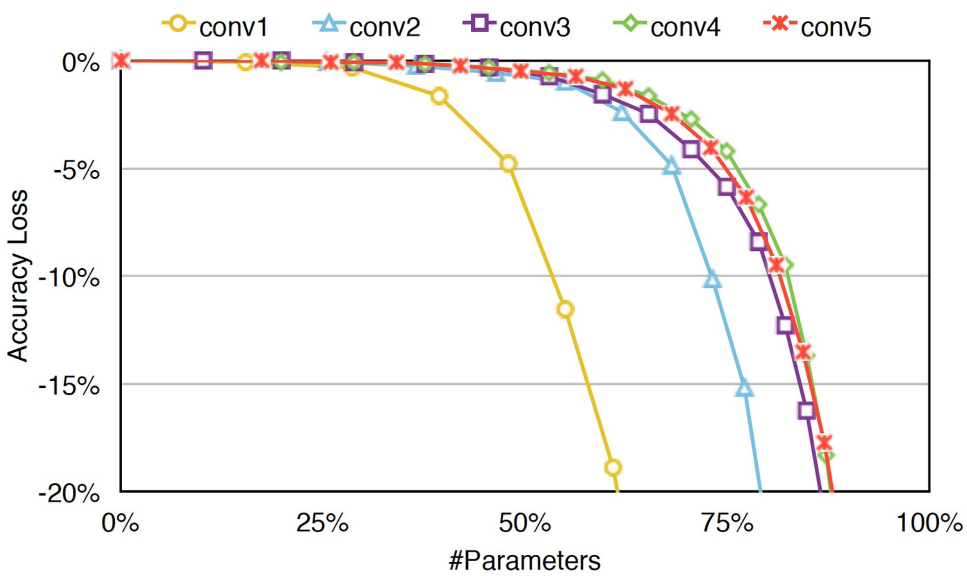
Learning connections, along with weights

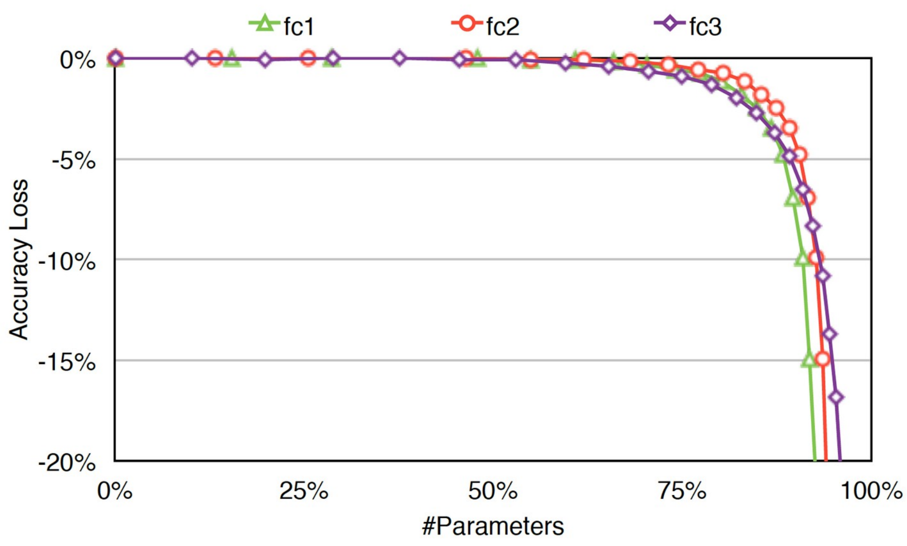
- L2 regularization gives better accuracy for pruned networks
- Reduce dropouts, as some connections are already pruned

$$C_i = N_i N_{i-1} \quad (1) \qquad D_r = D_o \sqrt{\frac{C_{ir}}{C_{io}}} \quad (2)$$

- Start from learned weights of retained connections during retraining, instead of re-initializing them
- Iterative pruning better at minimizing connections than one step aggressive pruning
- Pruning connections followed up by pruning neurons, which retain zero connections

Layer type vs. sensitivity





Comparison with other methods

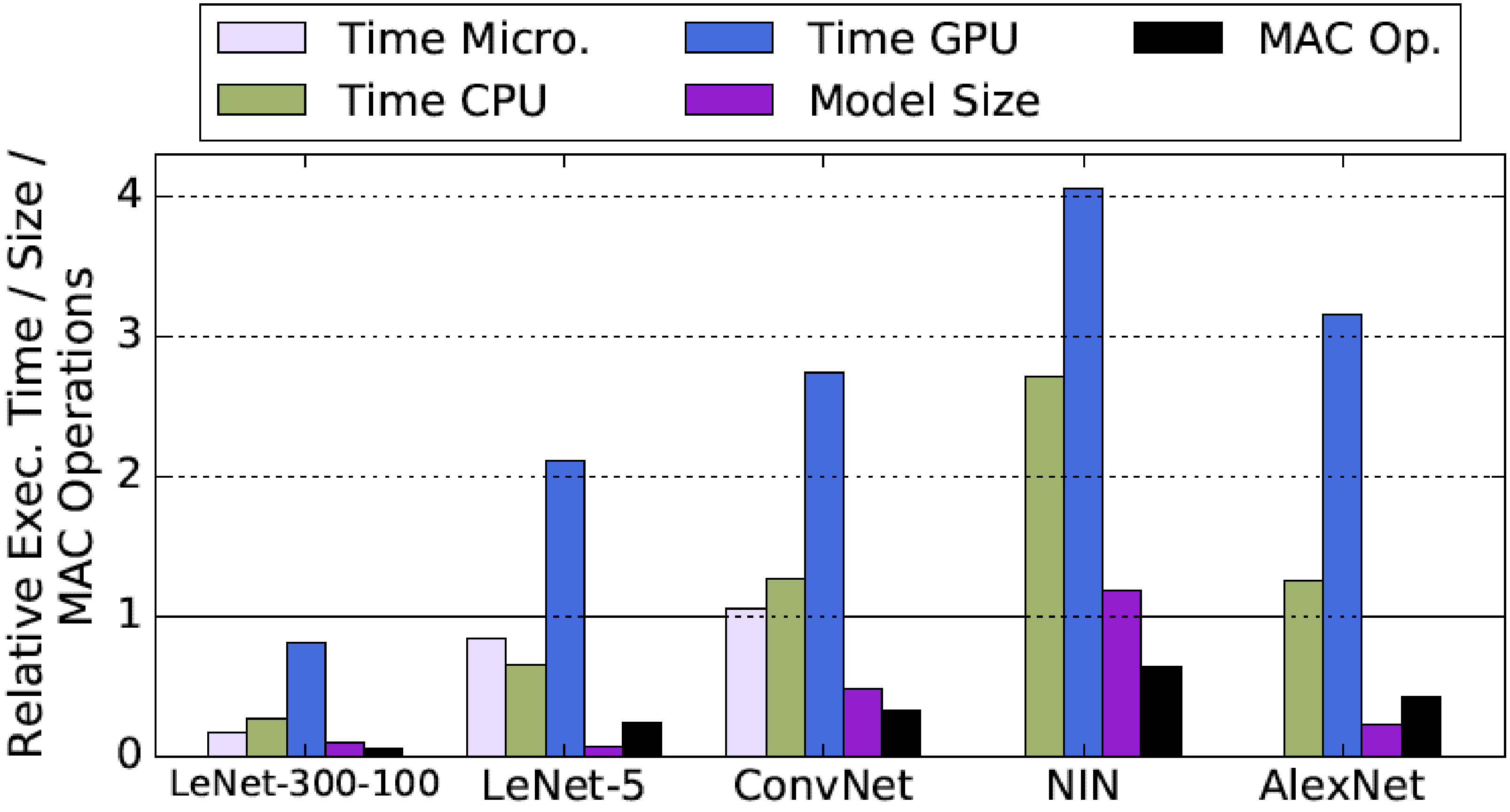
Network	Top-1 Error	Top-5 Error	Parameters	Compression Rate
Baseline Caffemodel [26]	42.78%	19.73%	61.0M	1×
Data-free pruning [28]	44.40%	-	39.6M	1.5×
Fastfood-32-AD [29]	41.93%	-	32.8M	2×
Fastfood-16-AD [29]	42.90%	-	16.4M	3.7×
Collins & Kohli [30]	44.40%	-	15.2M	4×
Naive Cut	47.18%	23.23%	13.8M	4.4×
SVD [12]	44.02%	20.56%	11.9M	5×
Network Pruning	**42.77%**	**19.67%**	**6.7M**	**9×**

Jiecao Yu, Andrew Lukefahr, David Palframa, Ganesh Dasika, Reetuparna Das, Scott Mahlke

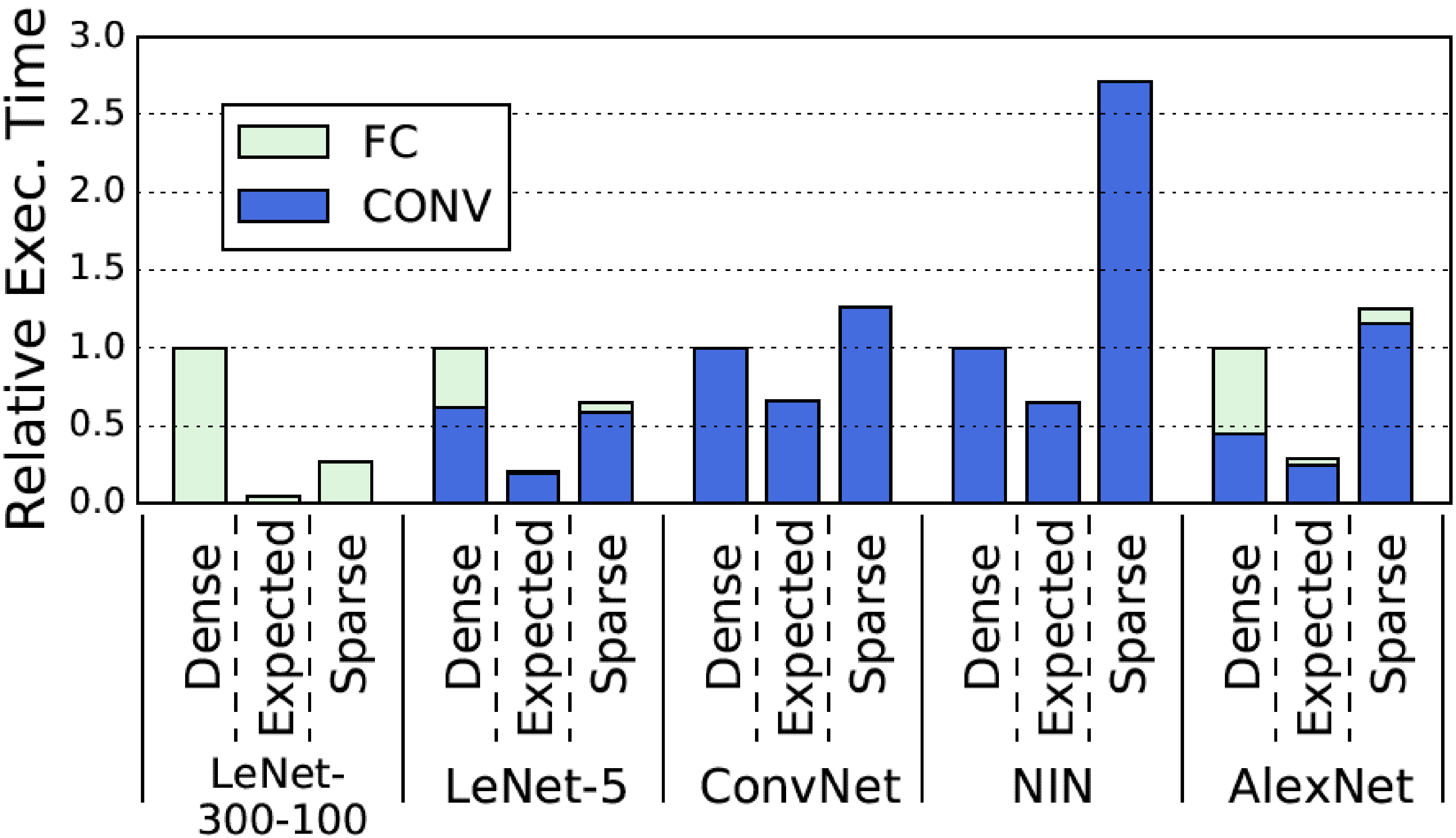
Scalpel: Customizing DNN Pruning to the Underlying Hardware Parallelism

ISCA 2017

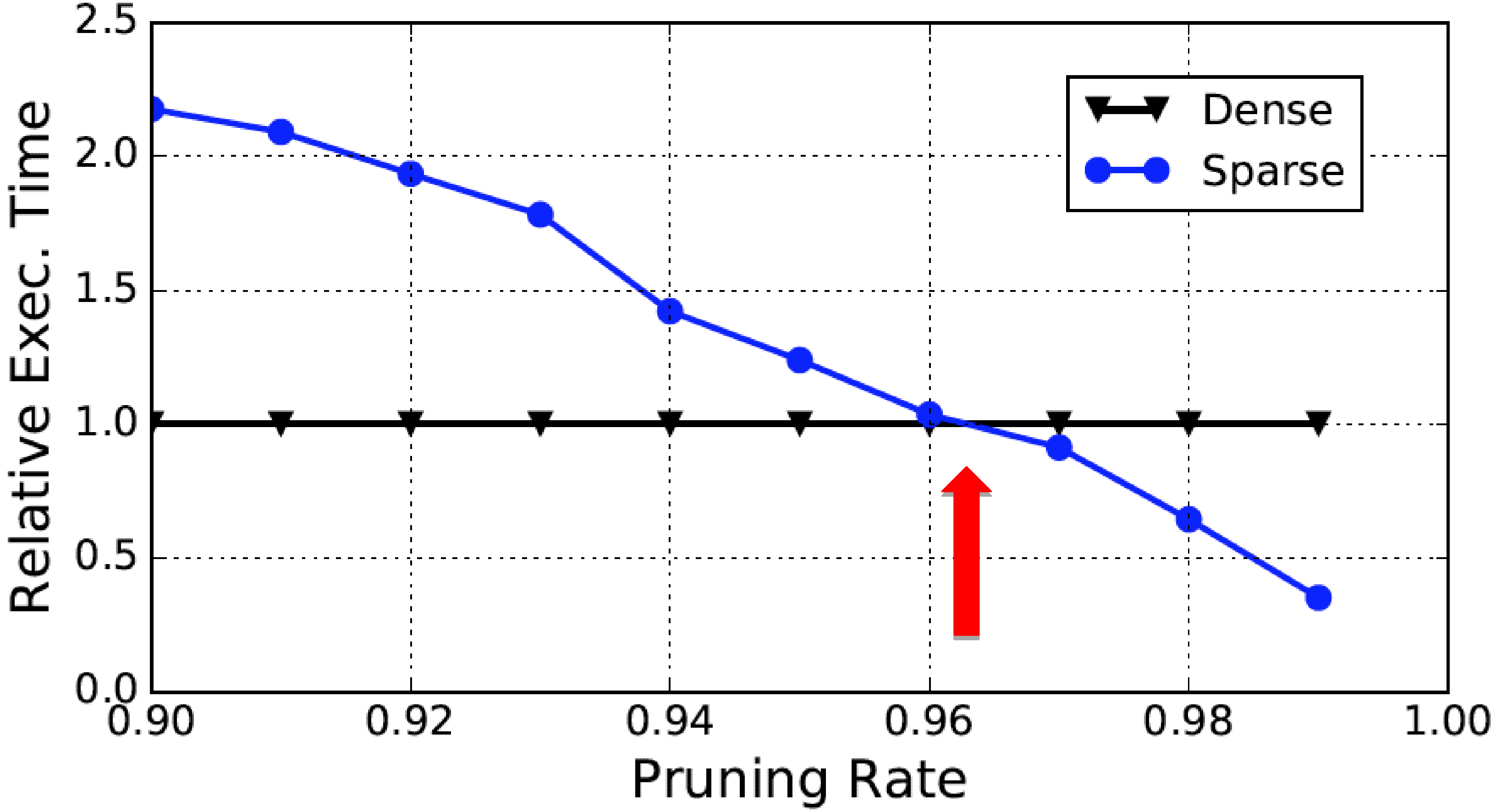
Effect of pruning on latency for existing hardware architecture



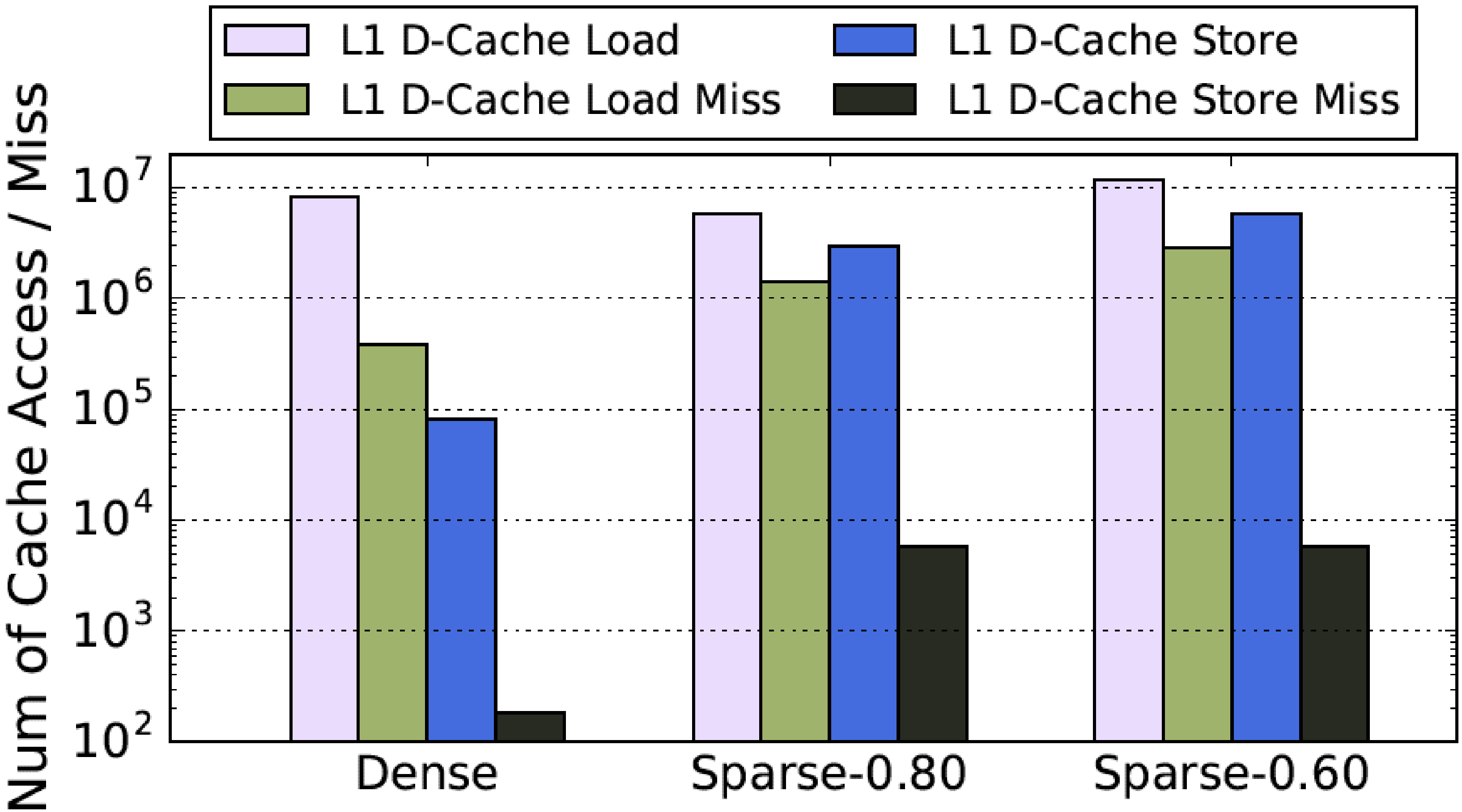
Effect of pruning on latency for existing hardware architecture



To see speedup, insane amount of sparsity is needed -> poor accuracy

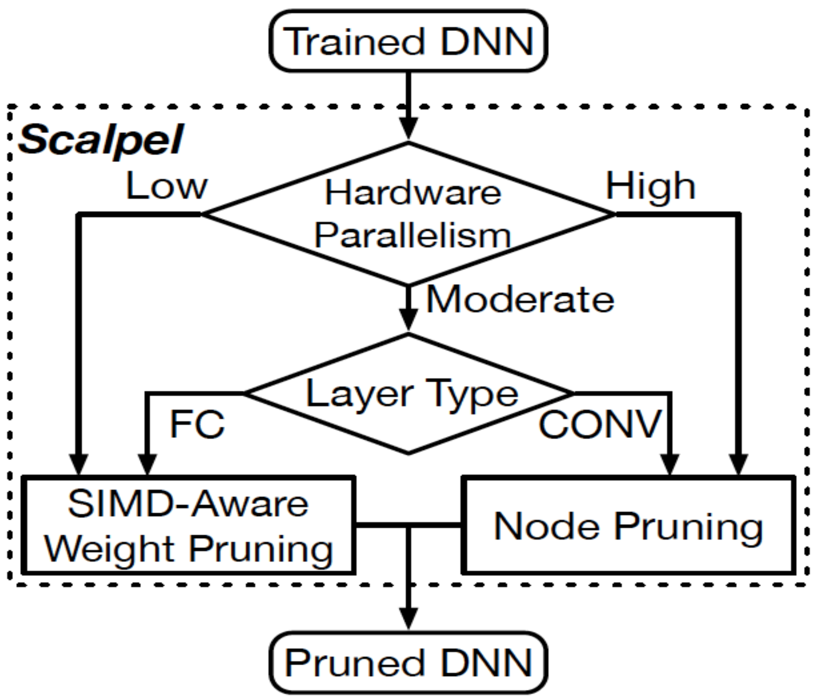


Cache misses, the reason?

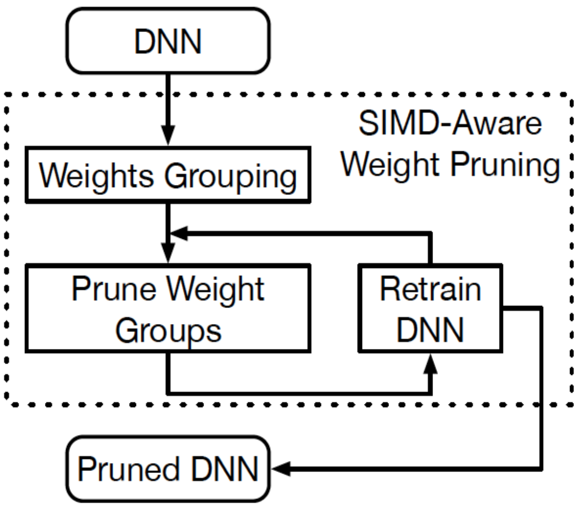


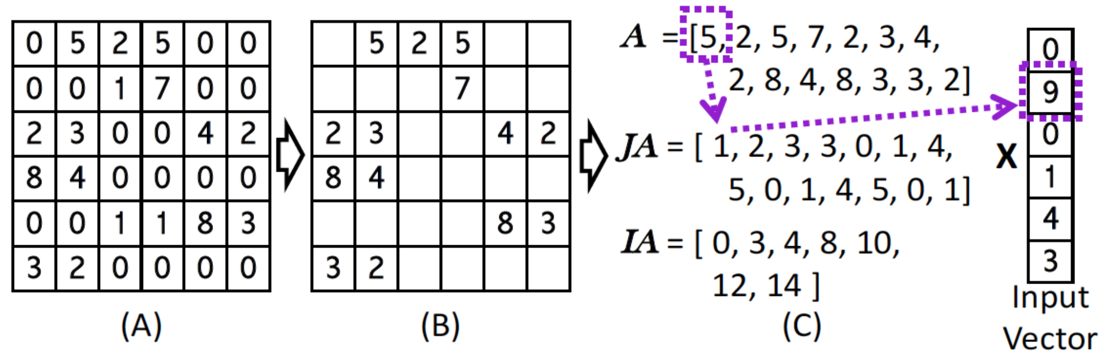
Hardware classes and Scalpel

	Parallelism		
	Low	Moderate	High
Example	Micro-controller	CPU	GPU
Memory Hierarchy	No cache	Deep cache hierarchy	High bandwidth / long latency
Memory Size	~100KB SRAM	~8MB SRAM	2-12GB DRAM

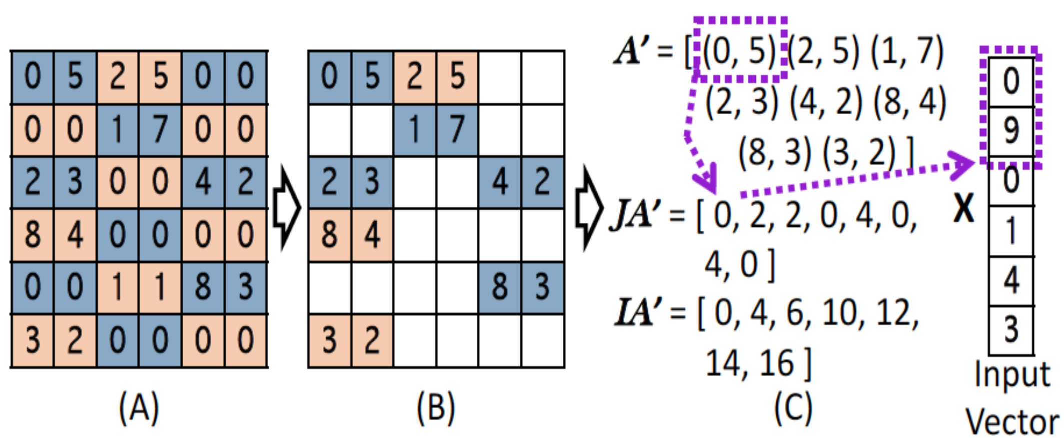


“SIMD aware” weight pruning

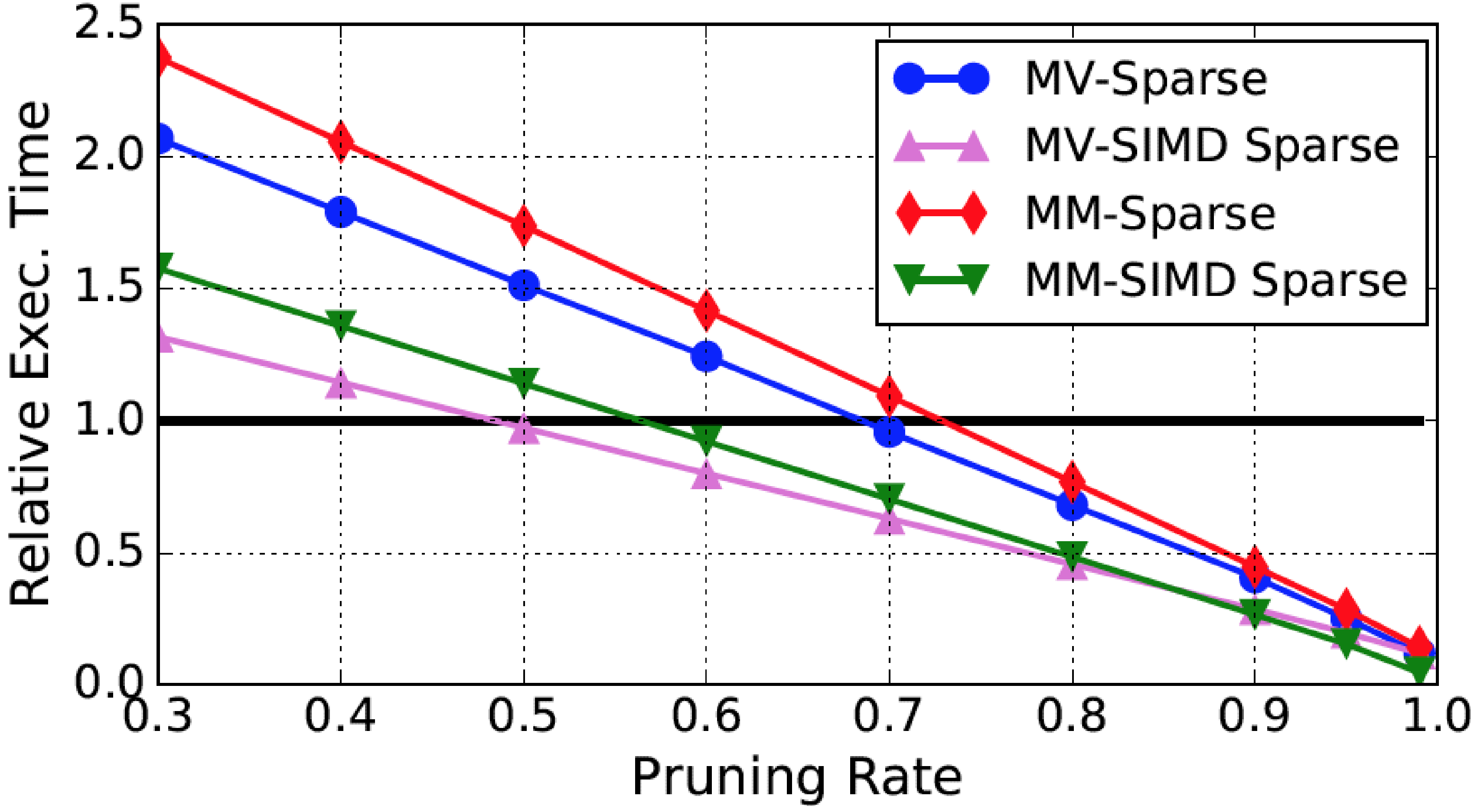




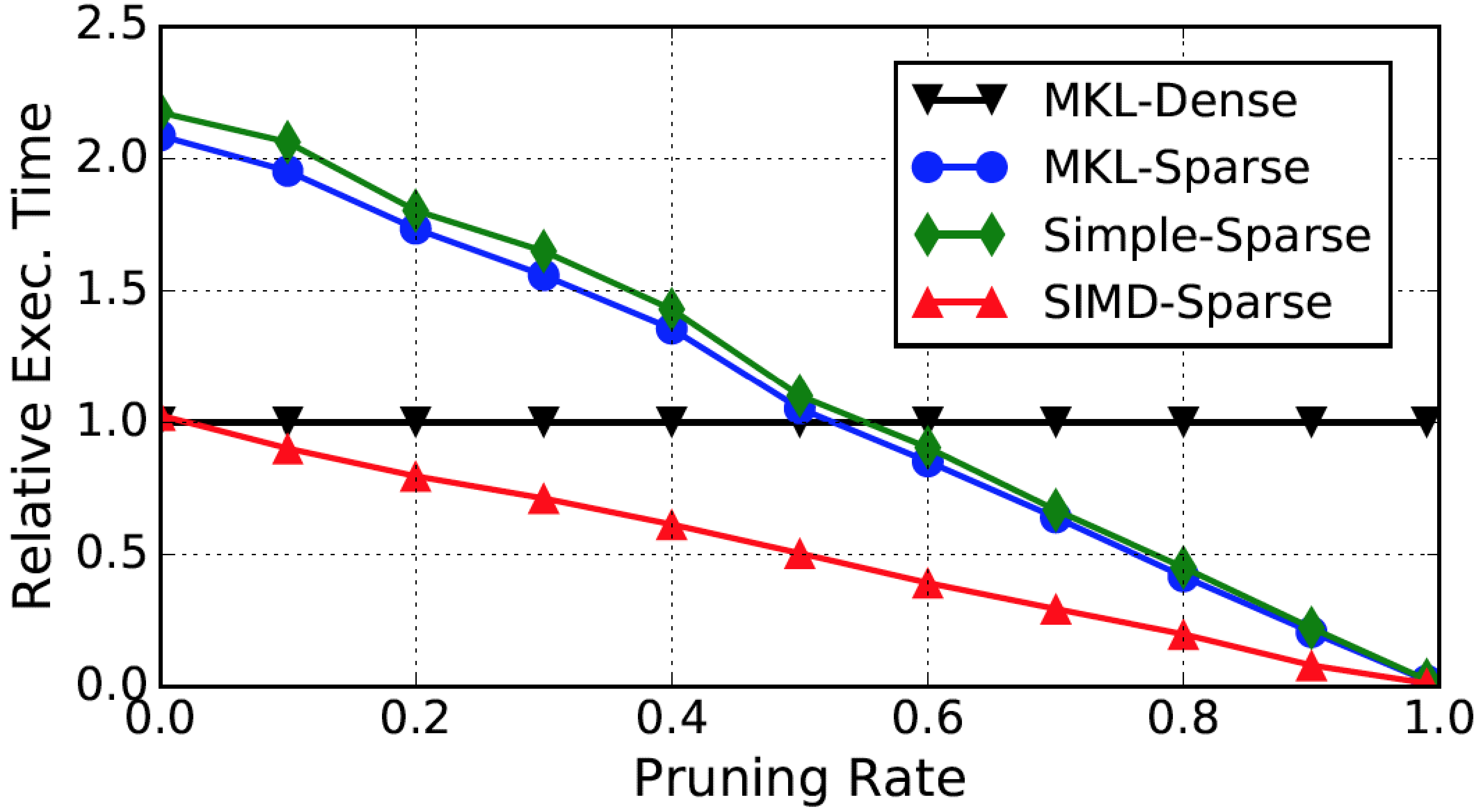
Group size equal to SIMD width = 2 for Cortex M4,
more efficient memory accesses



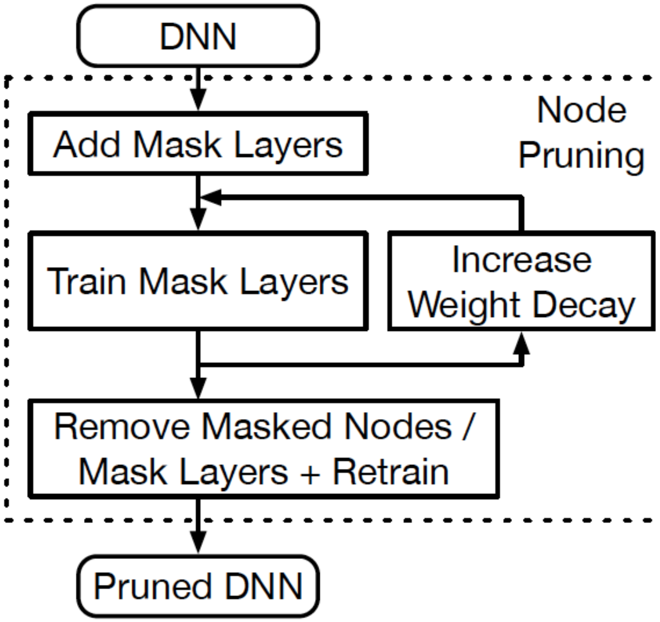
Effect of SIMD awareness on Cortex M4



Effect of SIMD awareness on Intel i7

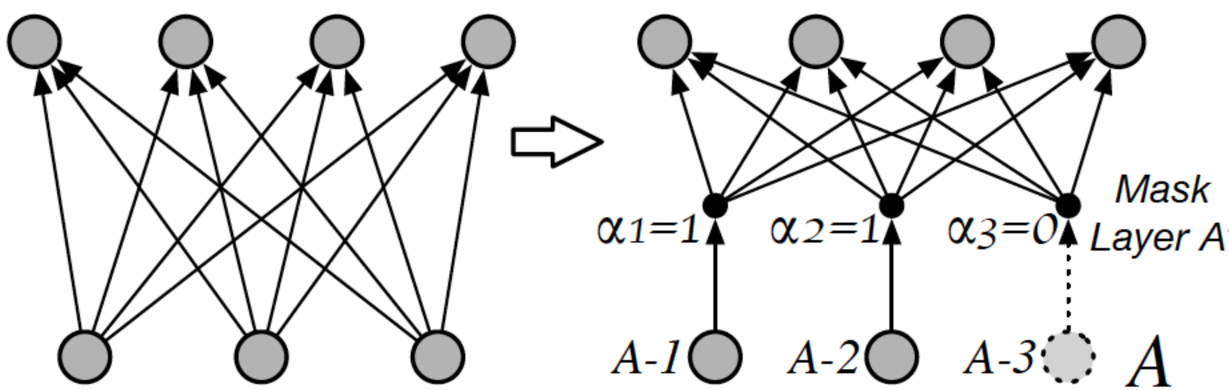


Node pruning, for highly parallel hardware

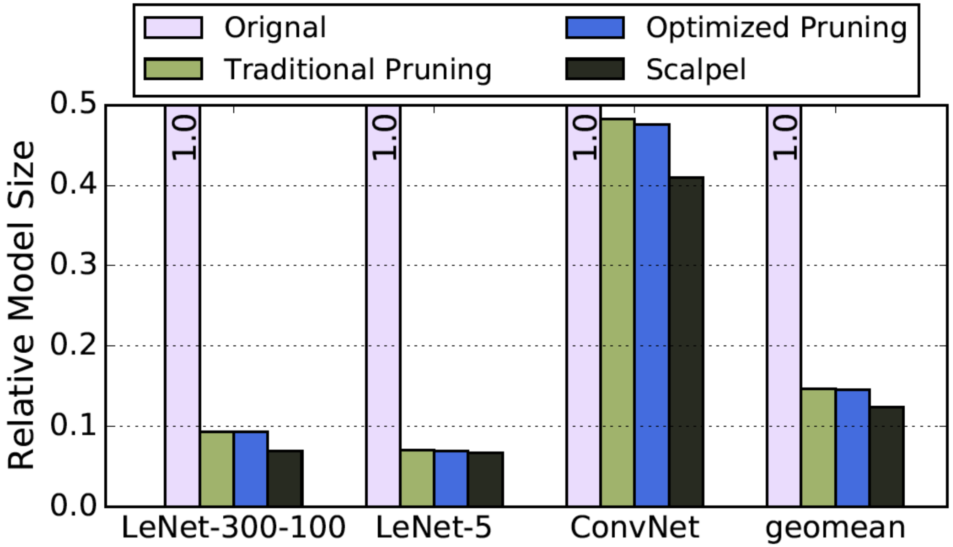


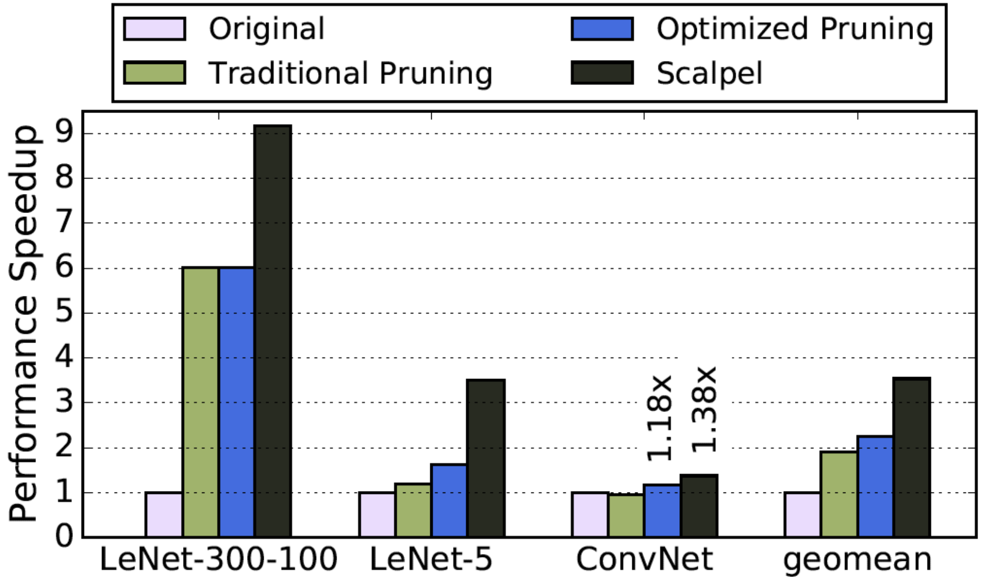
One neuron in FC layer or one feature map in Conv layer is considered a node.

Shrinks the size of each layer, but keeps the **dense** structure.

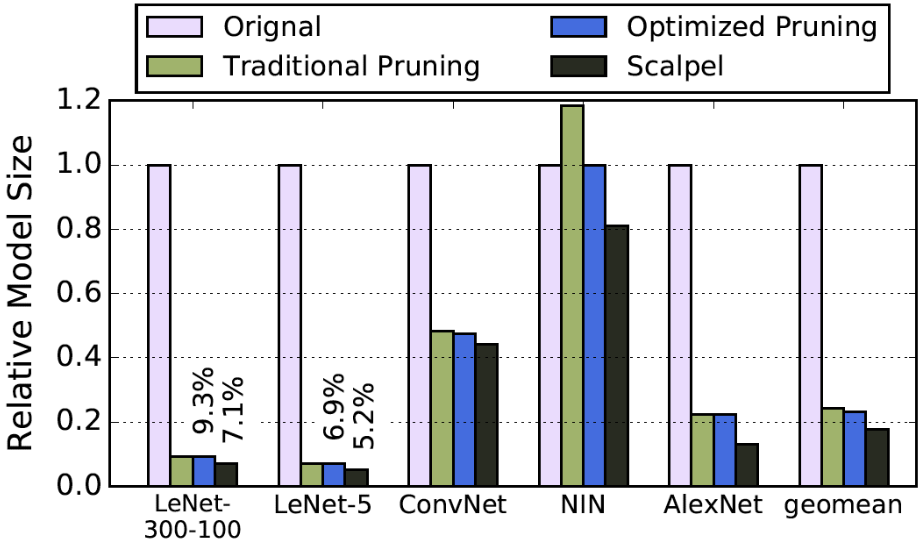


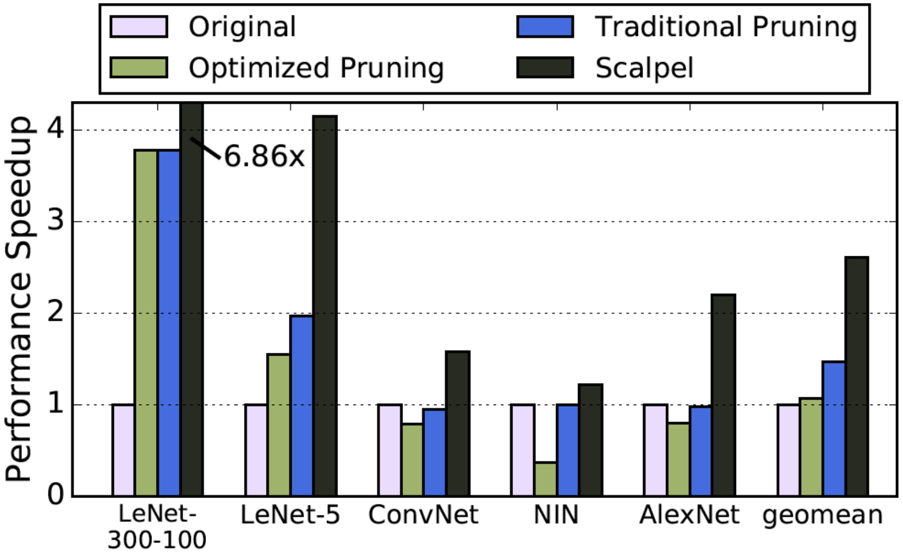
Scalpel results for ARM Cortex M4



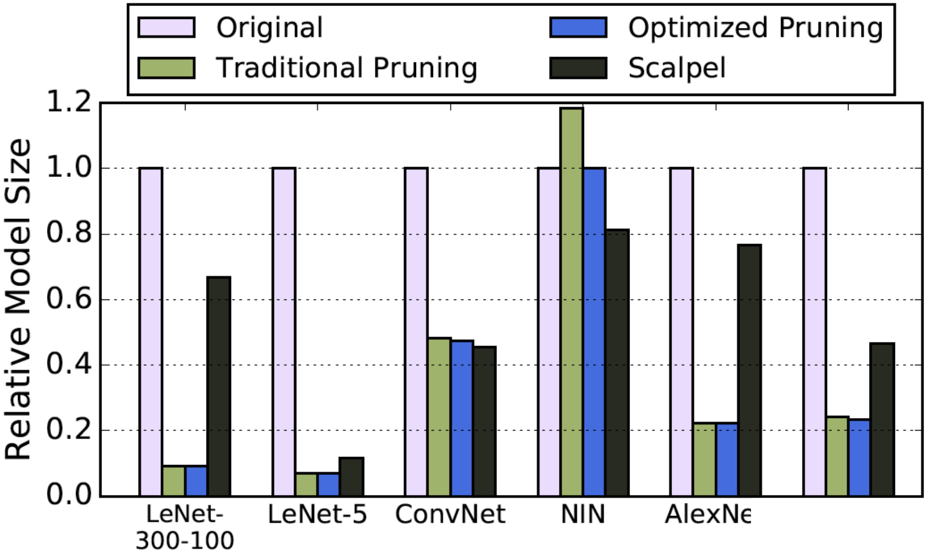


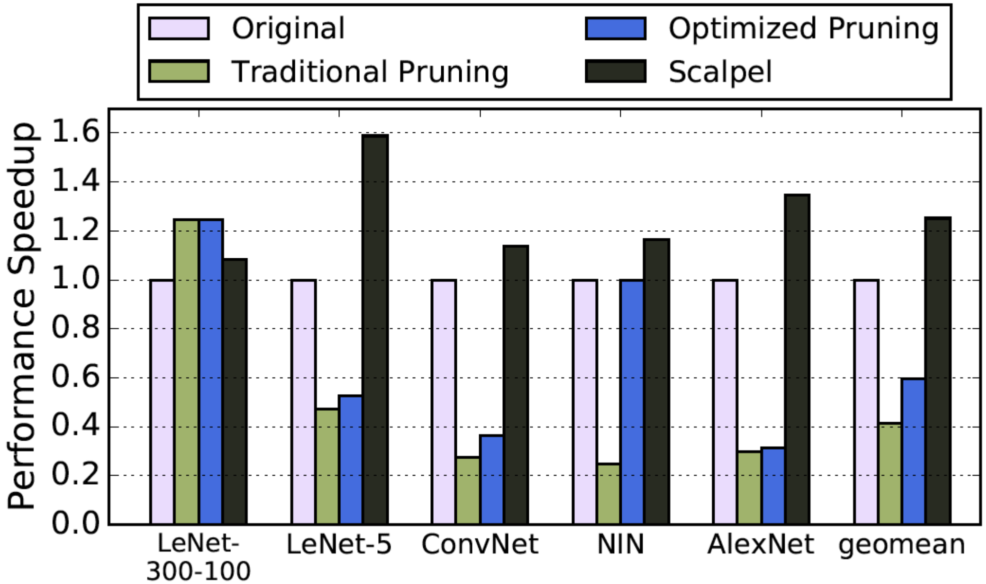
Scalpel results for Intel I7





Scalpel results for Nvidia GTX Titan X



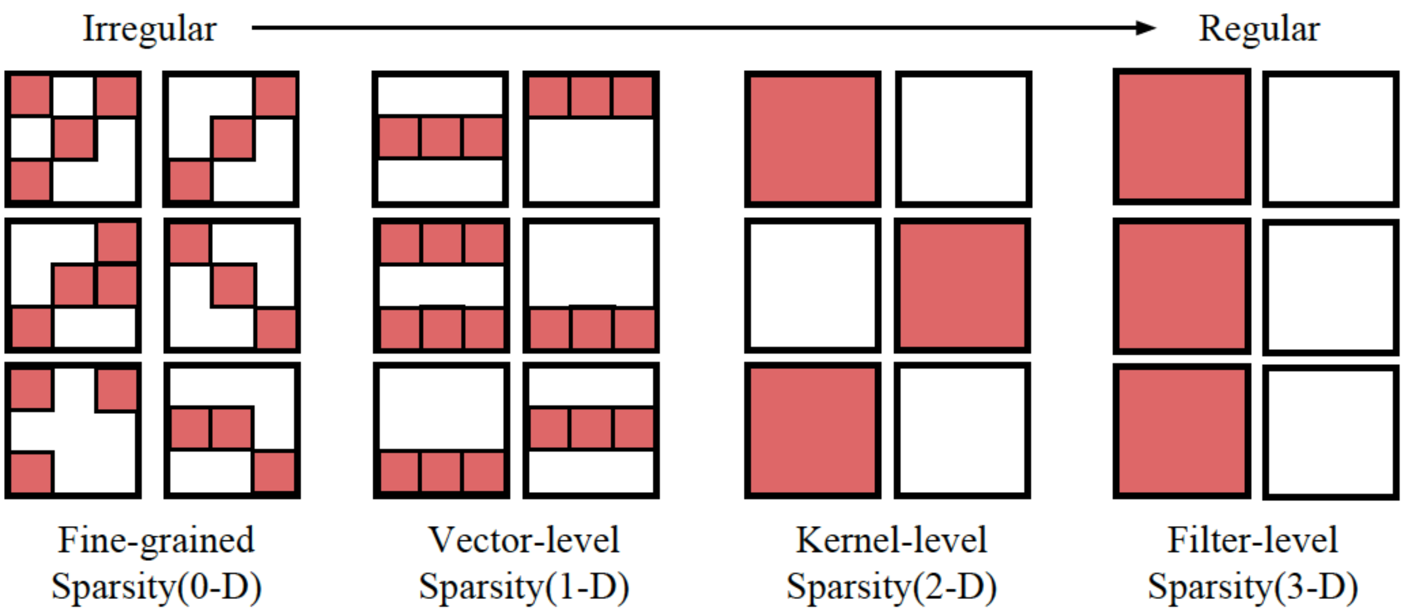


Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, William J. Dally

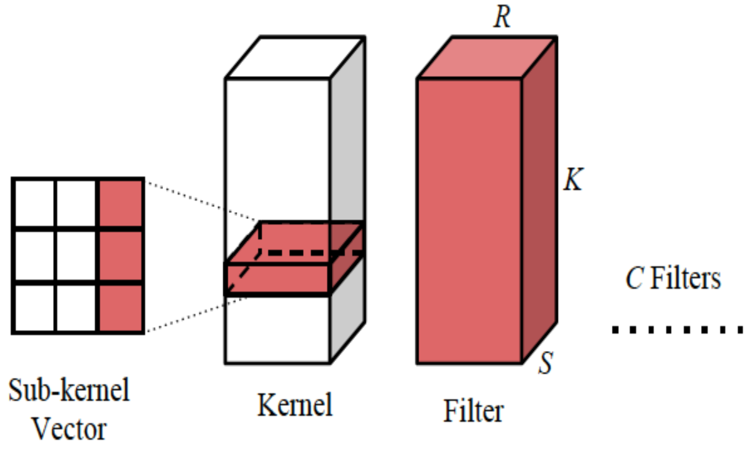
Exploring the Regularity of Sparse Structure in Convolutional Neural Networks.

NIPS 2017

Irregular fine-grained vs. regular coarse grained pruning



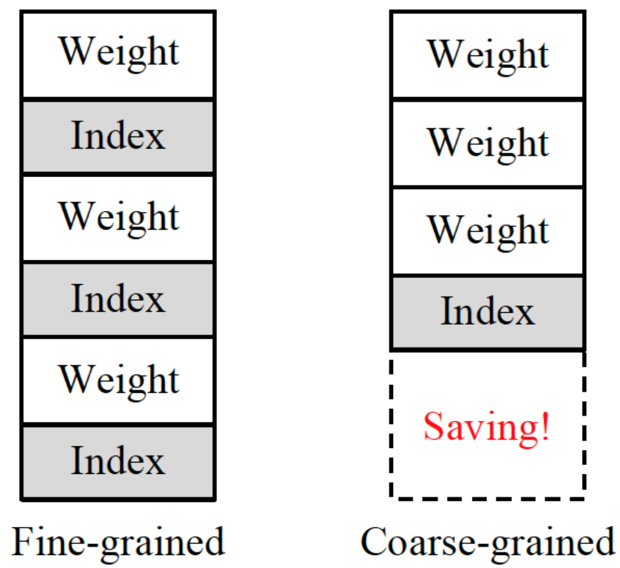
Granularity levels




```
Weights = Array(C, K, R, S)

# Case: Dimension-level granularity
Filter(3-Dim) = Weights[c, :, :, :]
Kernel(2-Dim) = Weights[c, k, :, :]
Vector(1-Dim) = Weights[c, k, r, :]
Fine-grain(0-Dim) = Weights[c, k, r, s]
```

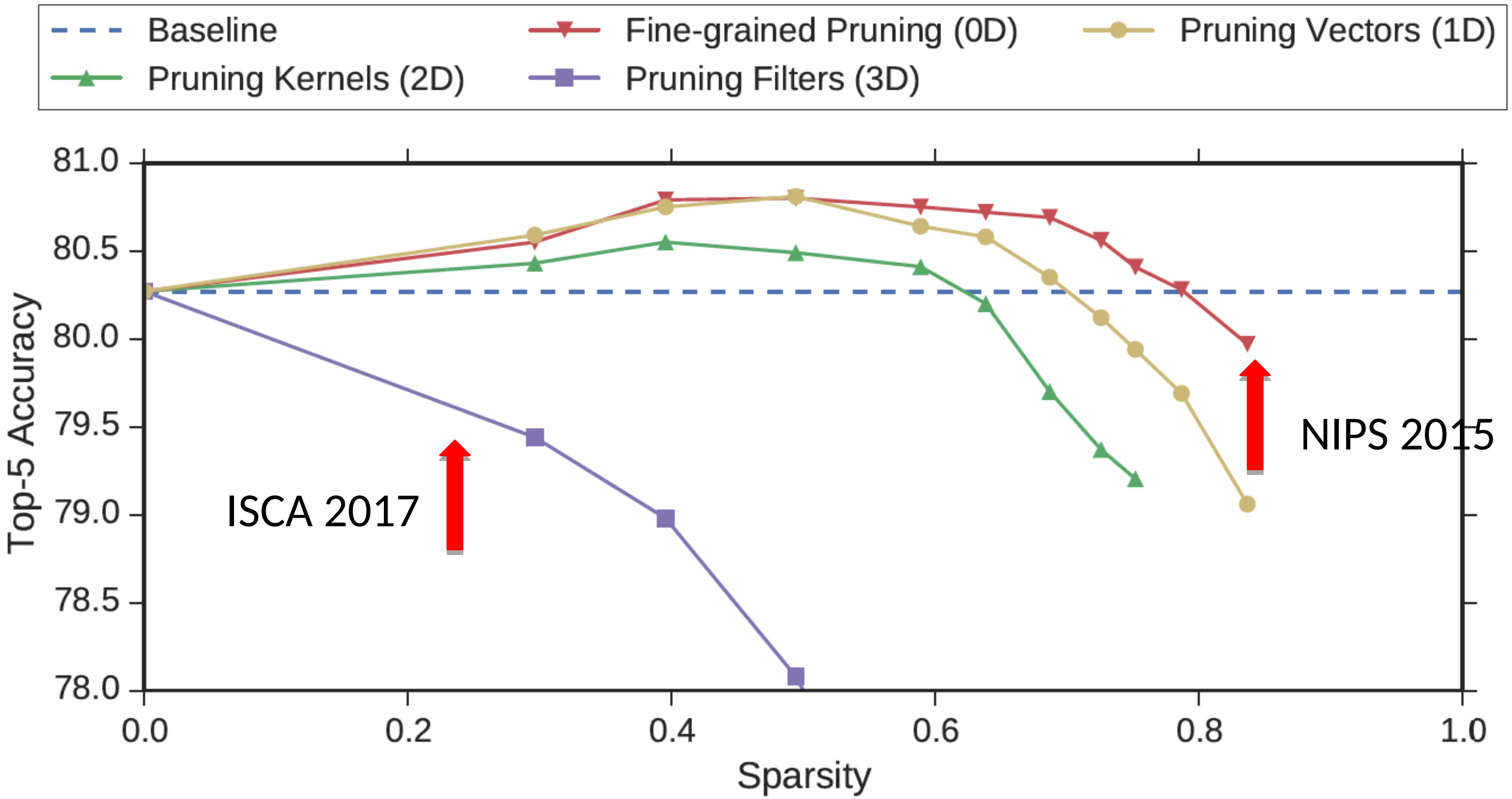
Why coarse grained better suits hardware architecture



Output memory references for VGG-16 (convolutional layers only).

Density	Fine-grained (0-D)	Vector Pruning (1-D)	Relative # of memory references
40.1%	1.77B	1.23B	**69.5%**
33.1%	1.53B	1.03B	**67.2%**
27.5%	1.33B	0.87B	**65.3%**

What granularity is best for accuracy (Alexnet)?

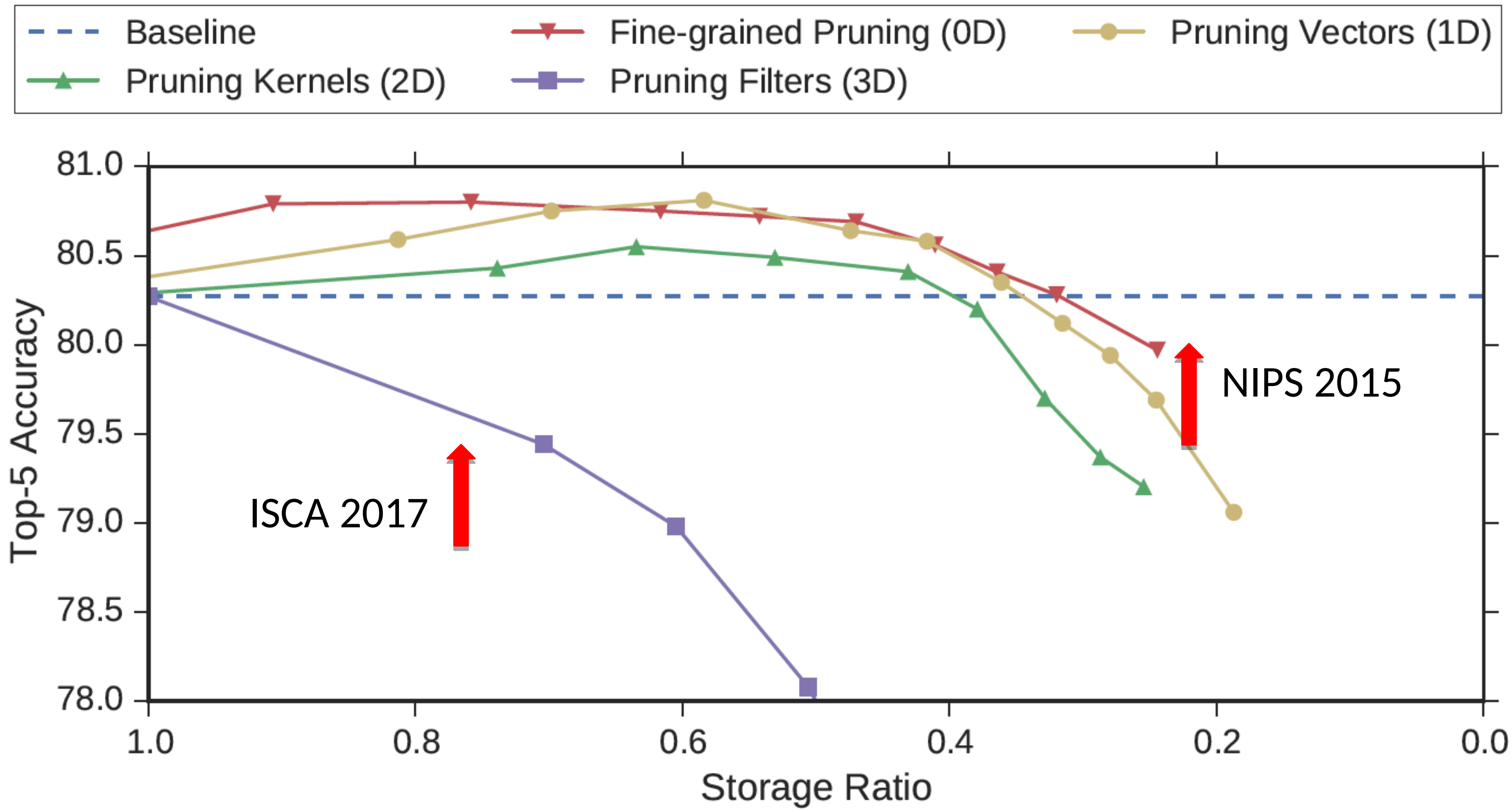


What granularity is best for accuracy (many nets, at a **given sparsity point**)?

Comparison of accuracies with the same density/sparsity.

Model	Density	Granularity	Top-5
AlexNet	24.8%	Kernel Pruning (2-D)	79.20%
		Vector Pruning (1-D)	79.94%
		Fine-grained Pruning (0-D)	**80.41%**
VGG-16	23.5%	Kernel Pruning (2-D)	89.70%
		Vector Pruning (1-D)	90.48%
		Fine-grained Pruning (0-D)	**90.56%**
GoogLeNet	38.4%	Kernel Pruning (2-D)	88.83%
		Vector Pruning (1-D)	89.11%
		Fine-grained Pruning (0-D)	**89.40%**
ResNet-50	40.0%	Kernel Pruning (2-D)	92.07%
		Vector Pruning (1-D)	92.26%
		Fine-grained Pruning (0-D)	**92.34%**
DenseNet-121	30.1%	Kernel Pruning (2-D)	91.56%
		Vector Pruning (1-D)	91.89%
		Fine-grained Pruning (0-D)	**92.21%**

What granularity is best for model size (Alexnet)?



What granularity is best for model size (many nets, at a **given accuracy point**)?

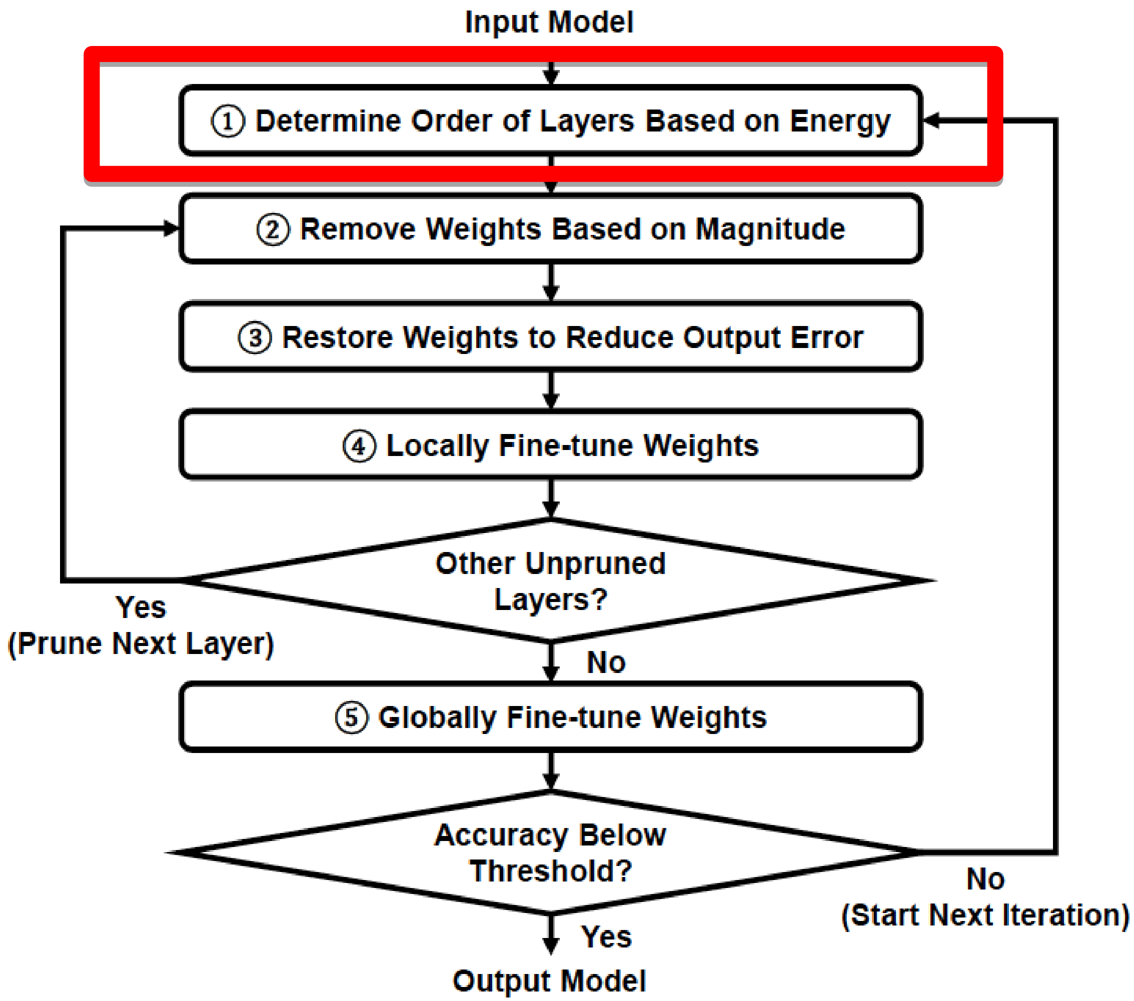
Model	Top-5 Accuracy	Granularity	Density	Storage Ratio
AlexNet	80.3%	Kernel Pruning (2-D)	37.8%	39.7%
		Vector Pruning (1-D)	29.9%	34.5%
		Fine-grained Pruning (0-D)	22.1%	**33.0%**
VGG-16	90.6%	Kernel Pruning (2-D)	44.4%	46.9%
		Vector Pruning (1-D)	30.7%	**35.8%**
		Fine-grained Pruning (0-D)	27.0%	40.6%
GoogLeNet	89.0%	Kernel Pruning (2-D)	43.7%	51.6%
		Vector Pruning (1-D)	36.9%	**47.4%**
		Fine-grained Pruning (0-D)	32.3%	48.5%
ResNet-50	92.3%	Kernel Pruning (2-D)	61.3%	77.0%
		Vector Pruning (1-D)	40.0%	**52.7%**
		Fine-grained Pruning (0-D)	37.1%	55.7%
DenseNet-121	91.9%	Kernel Pruning (2-D)	35.5%	48.9%
		Vector Pruning (1-D)	31.1%	43.8%
		Fine-grained Pruning (0-D)	26.6%	**39.8%**

T.-J. Yang, Y.-H. Chen, V. Sze

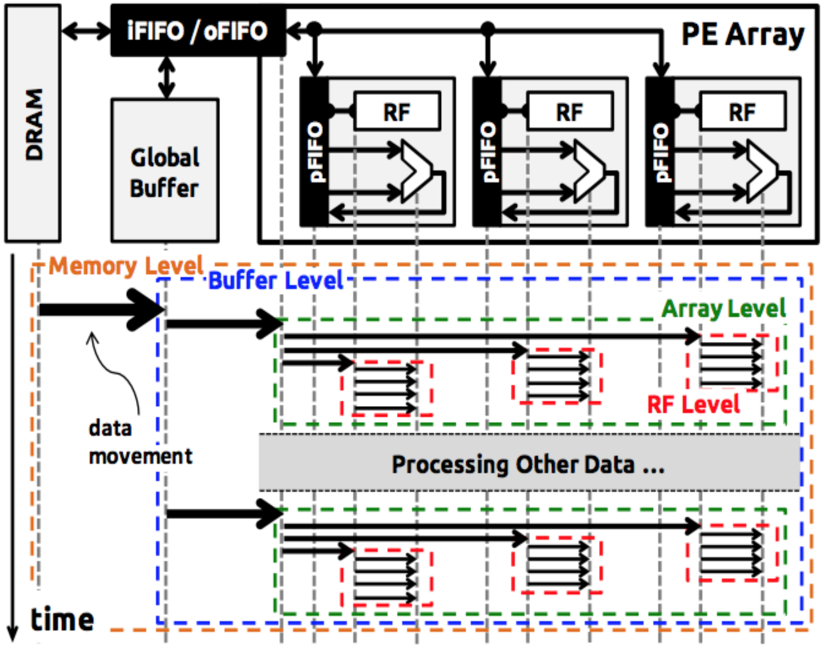
Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning

CVPR 2017

Explicitly consider energy consumption of different DNN layers



How to find energy consumption of layers and order them?



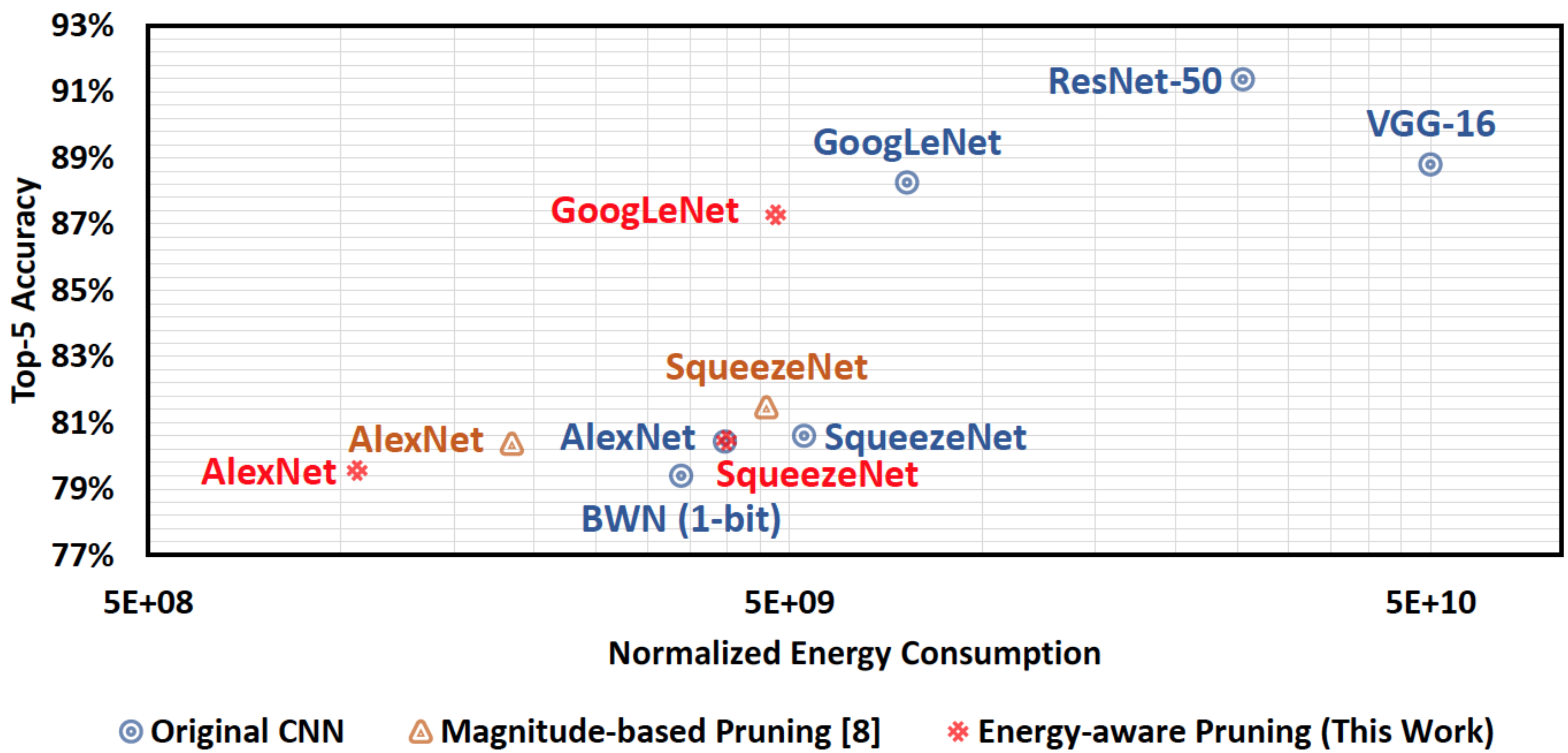
	DRAM	Buffer	Array	RF
Reuses	a = 1	b = 2	c = 3	d = 4

$$a \times EC(\text{DRAM}) + ab \times EC(\text{global buffer}) +$$
$$abc \times EC(\text{array}) + abcd \times EC(\text{RF}),$$

Effect of energy aware pruning, accuracy-latency trade-offs

Model		Top-5 Accuracy	# of Non-zero Weights ($\times 10^6$)		# of Non-skipped MACs ($\times 10^8$)[1]		Normalized Energy ($\times 10^9$)[1,2]	
AlexNet	(Original)	80.43%	60.95	(100%)	3.71	(100%)	3.97	(100%)
AlexNet	([8])	80.37%	6.79	(11%)	1.79	(48%)	1.85	(47%)
AlexNet	(Energy-Aware Pruning)	79.56%	5.73	(9%)	0.56	(15%)	1.06	(27%)
GoogLeNet	(Original)	88.26%	6.99	(100%)	7.41	(100%)	7.63	(100%)
GoogLeNet	(Energy-Aware Pruning)	87.28%	2.37	(34%)	2.16	(29%)	4.76	(62%)
SqueezeNet	(Original)	80.61%	1.24	(100%)	4.51	(100%)	5.28	(100%)
SqueezeNet	([8])	81.47%	0.42	(33%)	3.30	(73%)	4.61	(87%)
SqueezeNet	(Energy-Aware Pruning)	80.47%	0.35	(28%)	1.93	(43%)	3.99	(76%)

Effect of energy aware pruning, accuracy-latency trade-offs



- Deeper CNNs with fewer weights do not necessarily consume less energy than shallower CNNs with more weights -> **Squeeznet vs. Alexnet**

- Reducing number of weights saves more energy than reducing bitwidth of weights ->**BWN vs. pruned Alexnet**

Papers: Summary

- Song Han, Jeff Pool, John Tran, William J. Dally: *Learning both Weights and Connections for Efficient Neural Network*. NIPS 2015

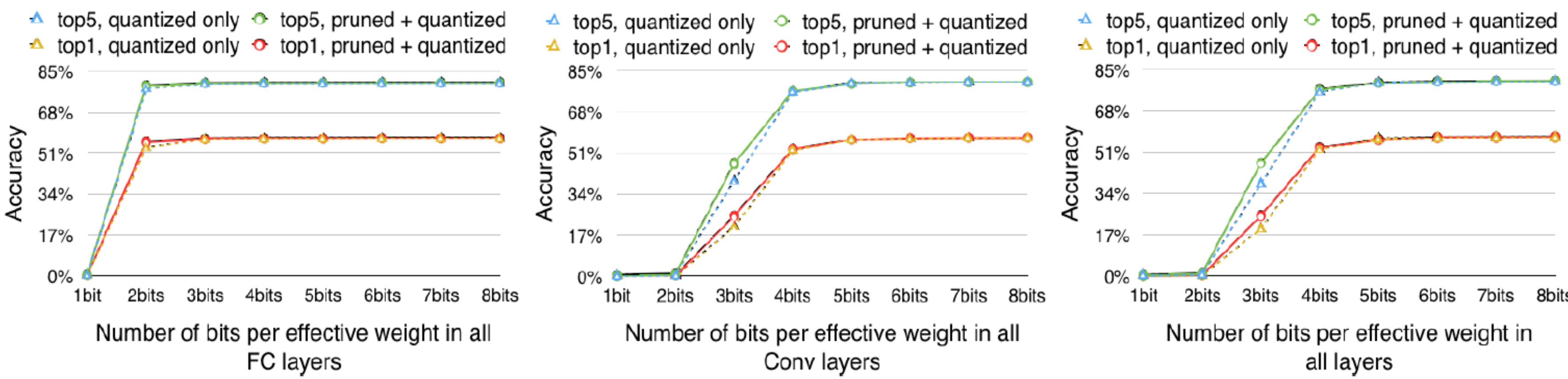
- Jiecao Yu, Andrew Lukefahr, David Palframa, Ganesh Dasika, Reetuparna Das, Scott Mahlke: *Scalpel: Customizing DNN Pruning to the Underlying Hardware Parallelism.* ISCA 2017

- Huizi Mao, Song Han, Jeff Pool, Wenshuo Li, Xingyu Liu, Yu Wang, William J. Dally: *Exploring the Regularity of Sparse Structure in Convolutional Neural Networks.* NIPS 2017

- T.-J. Yang, Y.-H. Chen, V. Sze: *Designing Energy-Efficient Convolutional Neural Networks using Energy-Aware Pruning.* CVPR 2017.

- Song Han, Huizi Mao, William J. Dally: *Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding.* ICLR 2016

Pruning doesn't hurt quantization



Pruning helps quantization (unsupervised)