Exploring the Granularity of Sparsity in Convolutional Neural Networks

Huizi Mao[1], **Song Han**[1], Jeff Pool[2], Wenshuo Li[3], Xingyu Liu[1], Yu Wang[3], William J. Dally[1,2]

Stanford University[1], NVIDIA[2], Tsinghua University[3]

Models are Getting Larger

IMAGE RECOGNITION

16X
Model

8 layers
1.4 GFLOP
~16% Error

152 layers
22.6 GFLOP
~3.5% error

2012
AlexNet

2015
ResNet

SPEECH RECOGNITION

10X
Training Ops

80 GFLOP
7,000 hrs of Data
~8% Error

465 GFLOP
12,000 hrs of Data
~5% Error

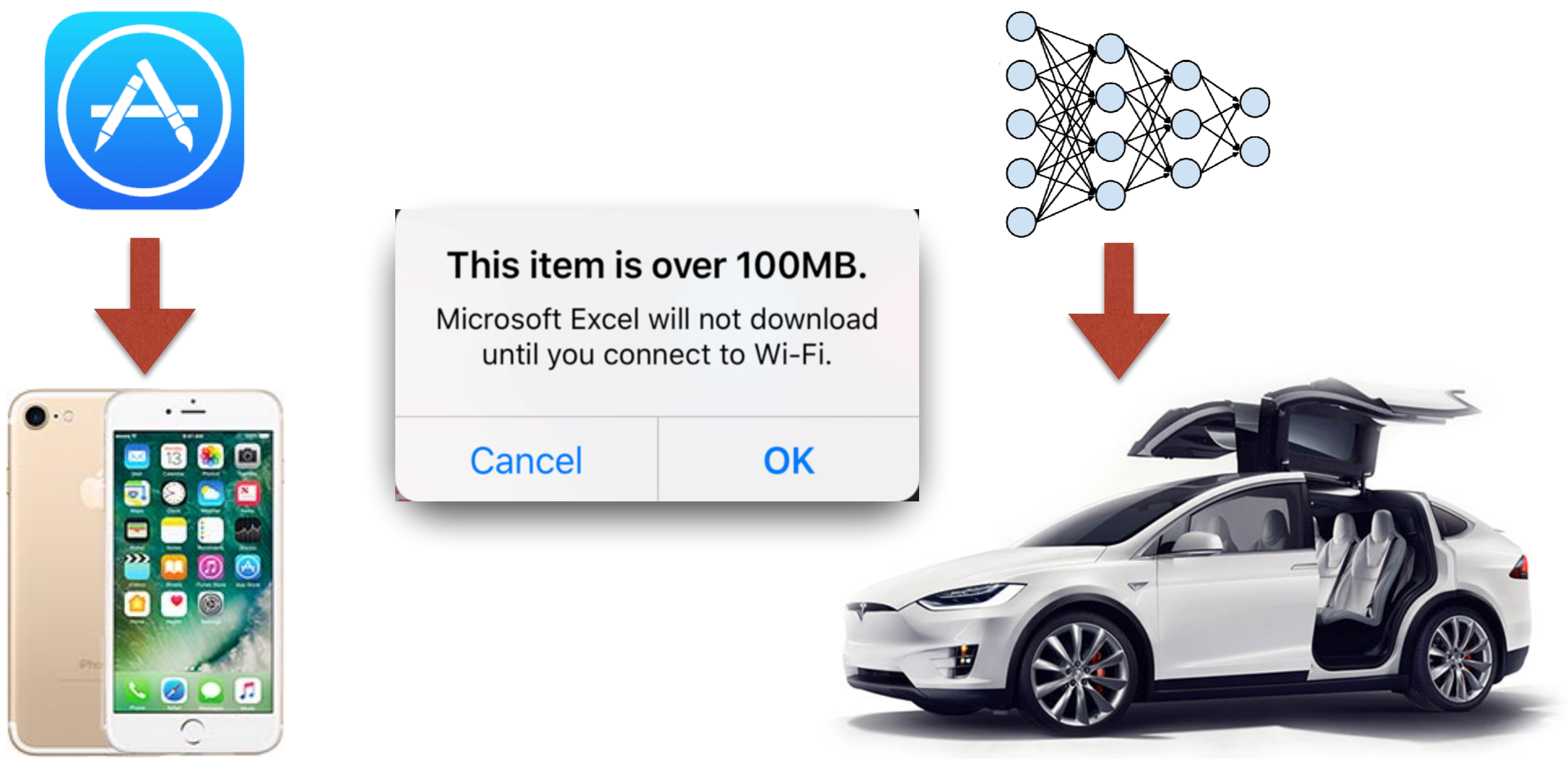
2014
Deep Speech 1

2015
Deep Speech 2

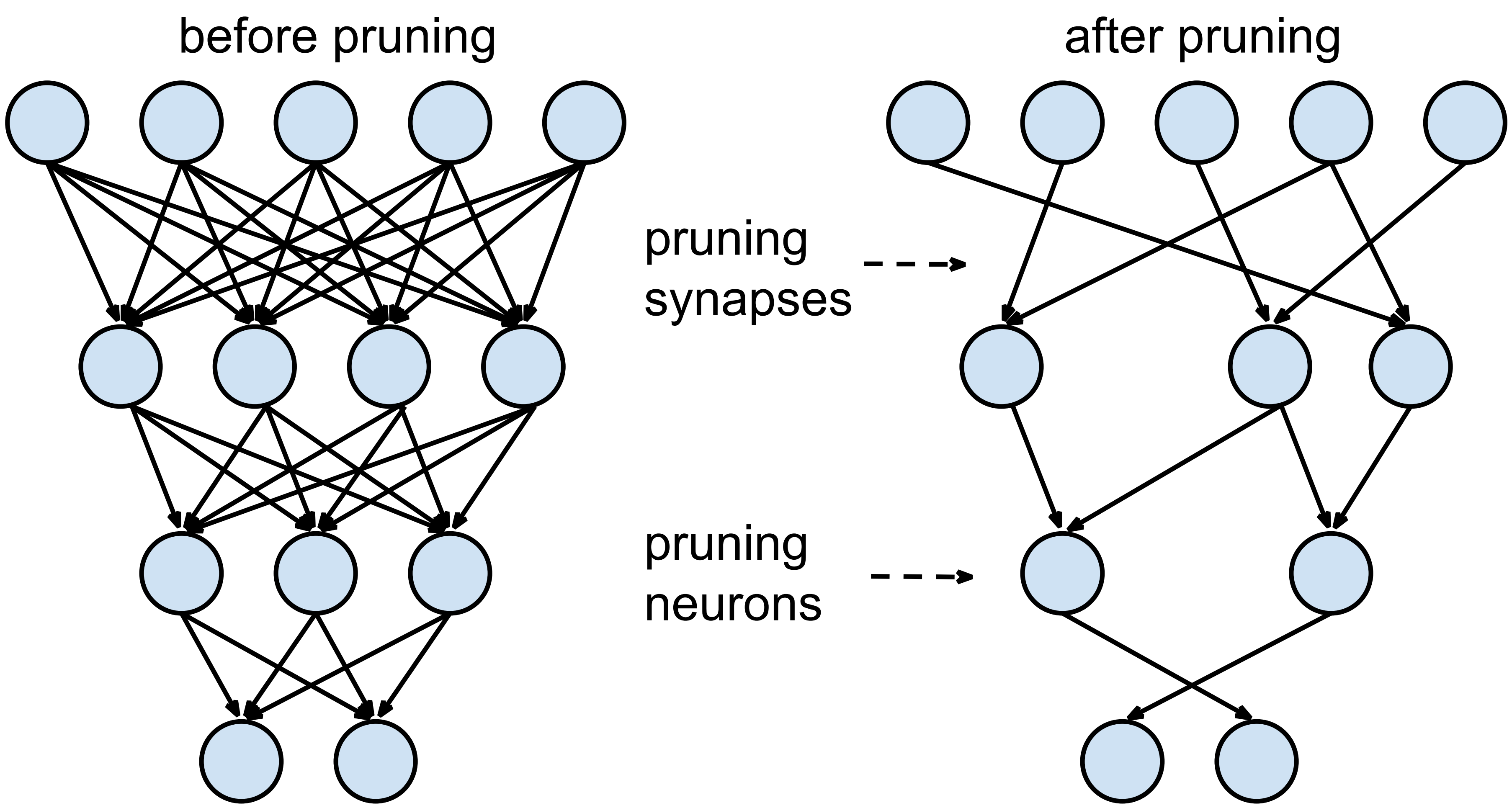
Dally, NIPS'2016 workshop on Efficient Methods for Deep Neural Networks

The Challenge: Model Size

Hard to distribute large models through over-the-air update



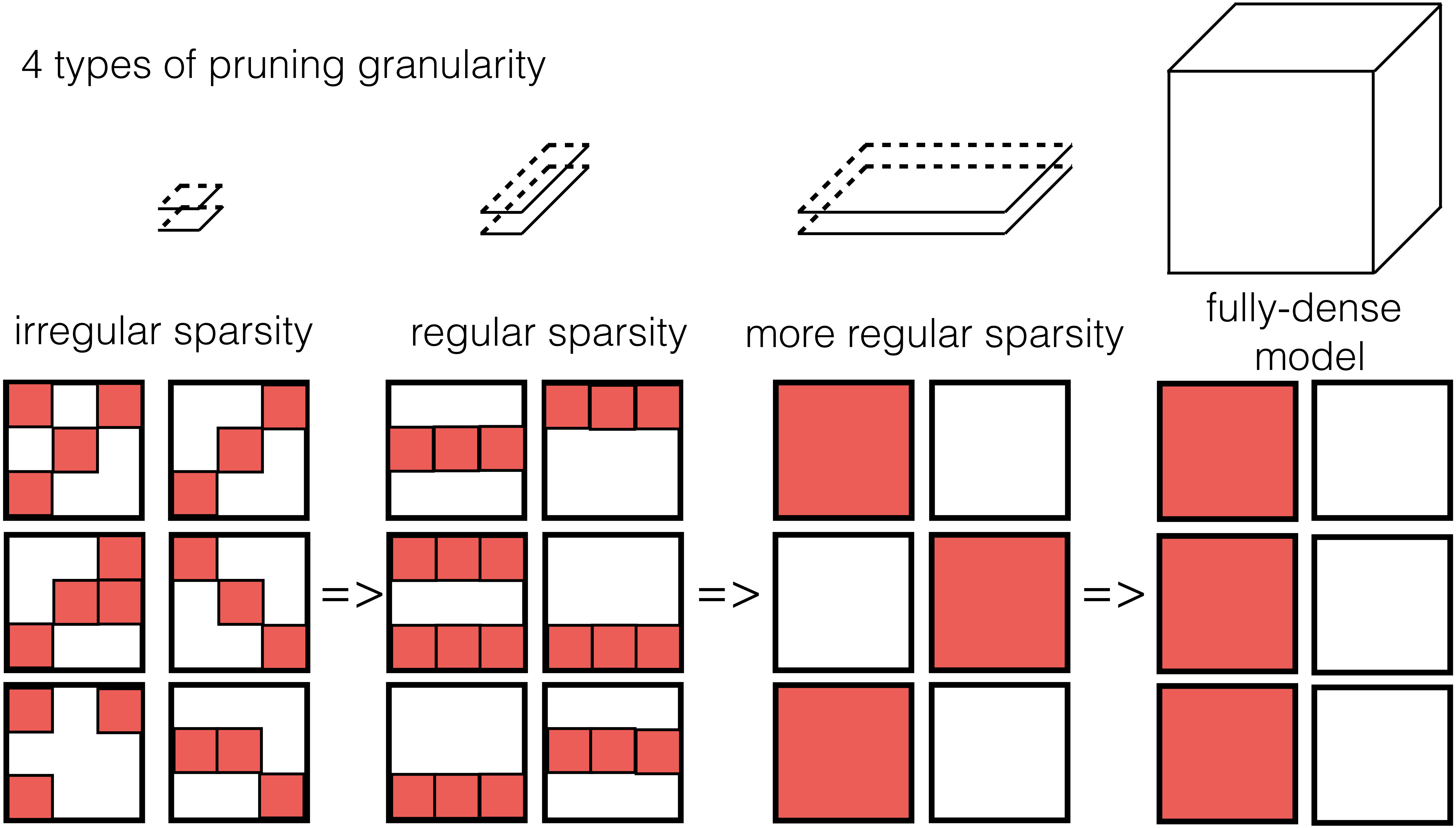
Our Previous Work: Pruning Neural Networks



[Han et al. NIPS'15]

Exploring the Granularity of Sparsity that is Hardware-friendly

4 types of pruning granularity



[Han et al, NIPS'15]

[Molchanov et al, ICLR'17]

Pruning Algorithm

Algorithm 1: Pruning Deep Neural Networks

Initialization: $W^{(0)}$ *with* $W^{(0)} \sim N(0, \Sigma)$.
Hyper-parameter: *threshold, step.*
Output : $W^{(t)}$.

——— *Train Connectivity* ———

while *not converged* **do**
    $W^{(t)} = W^{(t-1)} - \eta^{(t)} \nabla f(W^{(t-1)}; x^{(t-1)});$
    $t = t + 1;$
end

——— *Prune Connections* ———

// initialize the mask by thresholding the weights.
$Mask = \mathbb{1}(|W| > threshold);$
$W = W \cdot Mask;$

——— *Retrain Weights* ———

while *not converged* **do**
    $W^{(t)} = W^{(t-1)} - \eta^{(t)} \nabla f(W^{(t-1)}; x^{(t-1)});$
    $W^{(t)} = W^{(t)} \cdot Mask;$
    $t = t + 1;$
end

——— *Iterative Pruning* ———

$threshold = threshold - step;$
goto *Pruning Connections;*

Coarse Grain Pruning Saves Index

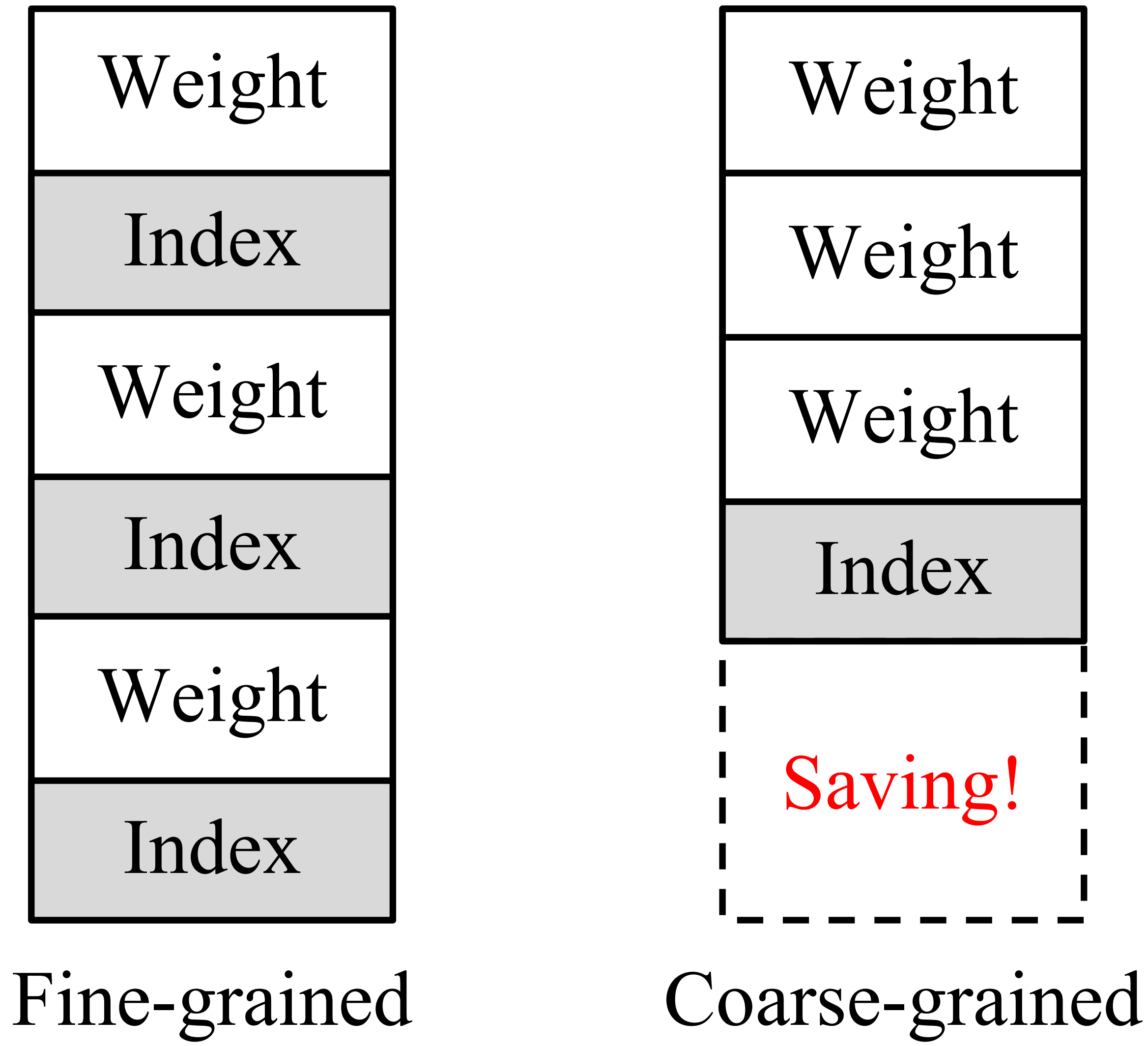
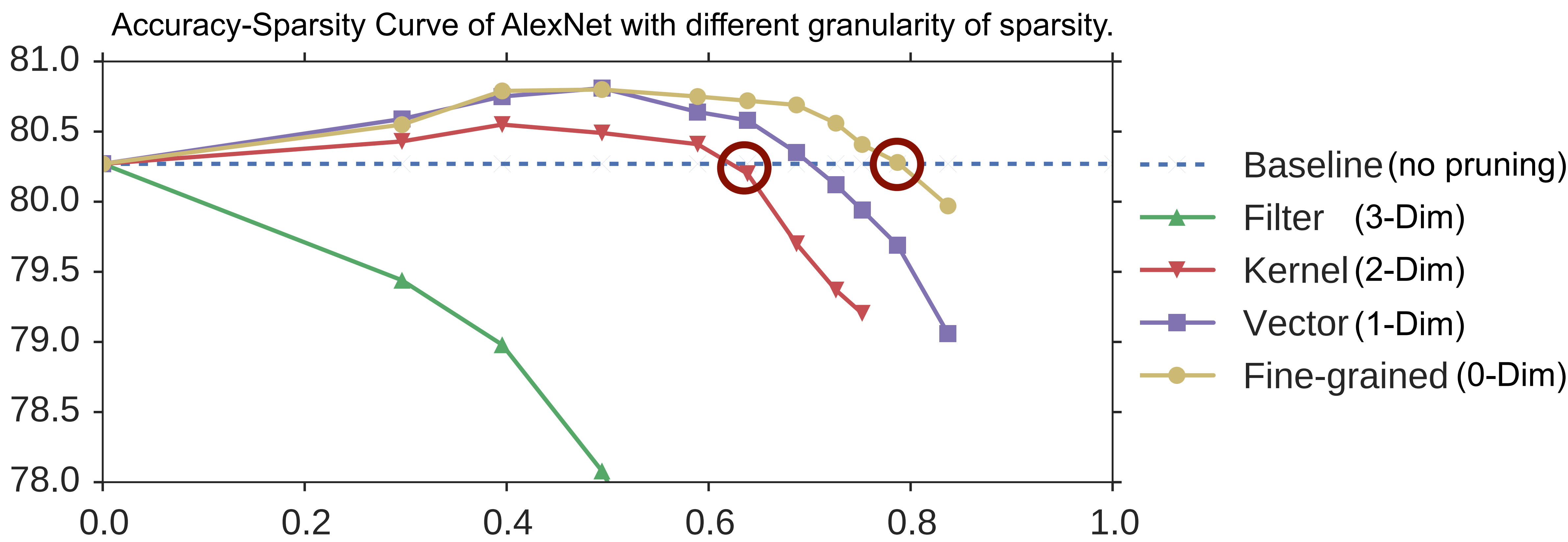


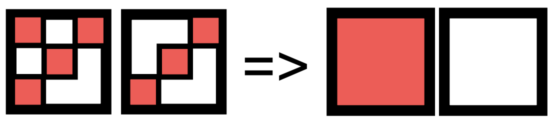
Figure 5: Illustration of index saving.

Accuracy ~ Sparsity ~ Granularity



X-axis: sparsity of convolution layers (percentage of zero weights).
Y-axis: top-5 accuracy on ImageNet validation set.

Fine-grain Pruning: remove 80% weights (no loss of accuracy)
2-D Pruning: remove 65% weights (no loss of accuracy)
=> Although we are able to remove less #weights, we get better regularity, SIMD friendly



Prediction Accuracy Comparison

(under the same density)

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%**

Storage Ratio Comparison

(under the same prediction accuracy)

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%**

Coarse Grain Pruning doesn't Impact Quantization

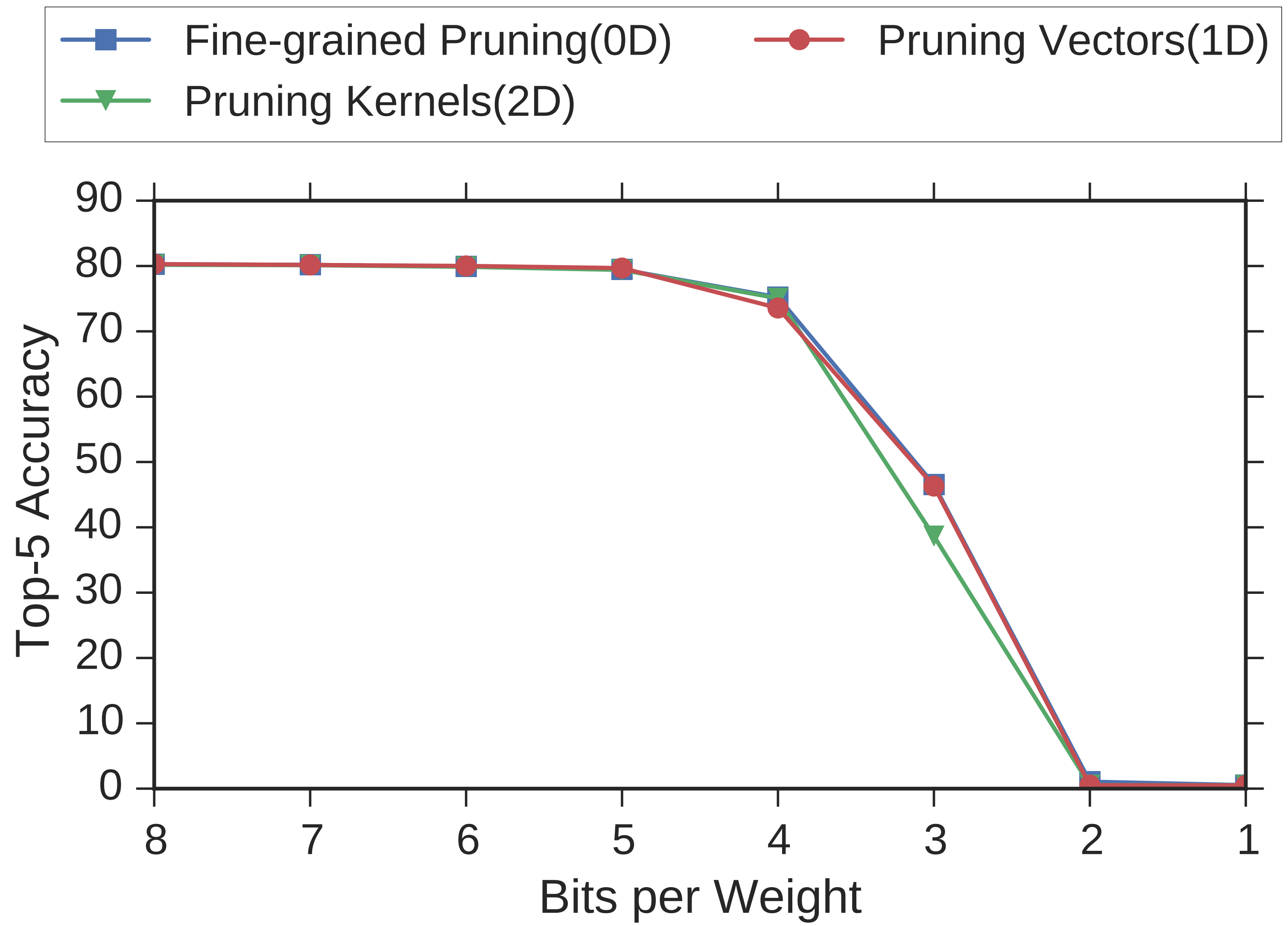
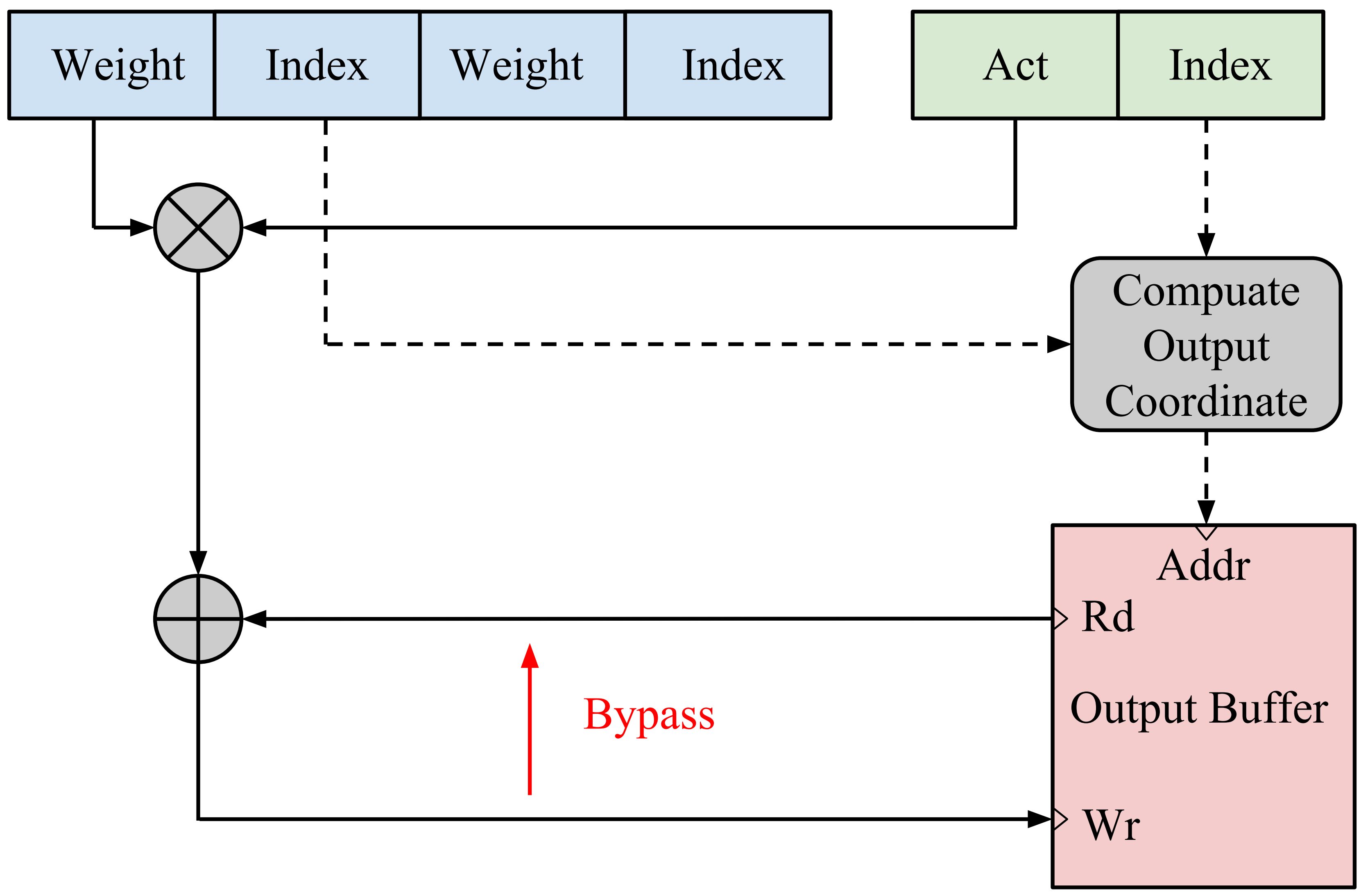


Figure 6: Three curves are almost identical, indicating sparsity structure does not impact quantization.

Coarse Pruning Helps SCNN Architecture



A simplified dataflow of SCNN architecture. With coarse grained pruning, bypass is possible when the same output address is referenced again. Reduce memory access.

Reduce #Memory Reference

Table 3: 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%**

Comparison with Previous Work

Layer	Param.	NIPS'15 [7]	NIPS'16 [8]	Fine-grained Pruning (ours)	Vector Pruning (ours)	Kernel Pruning (ours)
conv1	35K	84%	**54%**	83%	83%	83%
conv2	307K	38%	41%	**26%**	**26%**	**26%**
conv3	885K	35%	28%	**23%**	**23%**	**23%**
conv4	664K	37%	32%	**23%**	**23%**	**23%**
conv5	443K	37%	33%	**23%**	**23%**	**23%**
fc6	38M	9%	**3.7%**	7%	7%	7%
fc7	17M	9%	**6.6%**	7%	7%	7%
fc8	4M	25%	**4.6%**	18%	18%	18%
Total	61M	11%	**5.7%**	8.4%	8.4%	8.4%
FLOPs	1.5B	30%	25.4%	**24.1%**	**24.1%**	**24.1%**
Storage(conv)	2.3MB	55.6%	48.3%	36.4%	28.0%	**25.5%**
Storage(total)	61MB	16.7%	**8.5%**	12.6%	12.3%	12.2%
#Mem Reference	99M	74.4%	71.7%	60.5%	**34.6%**	35.2%
Top-5 Accuracy		80.23%	80.01%	**80.41%**	79.94%	79.20%