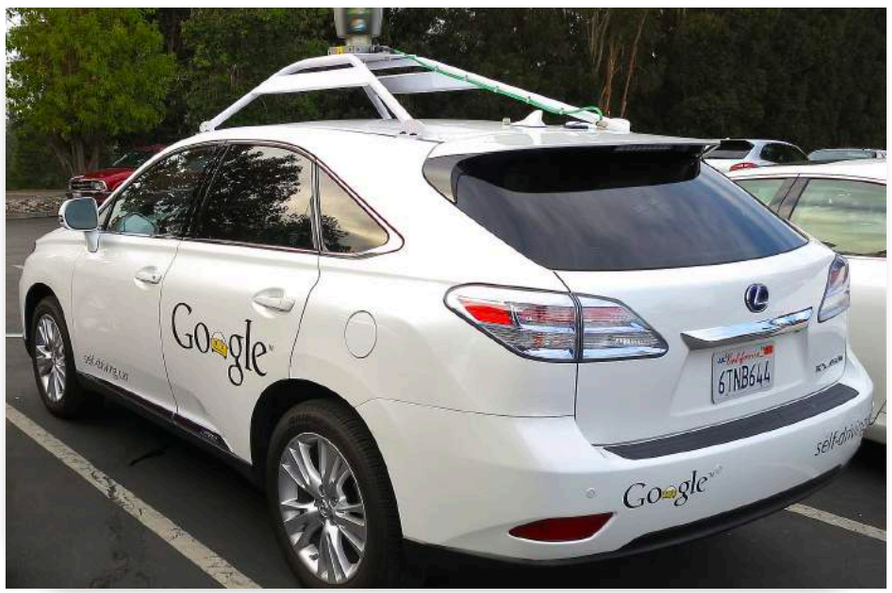
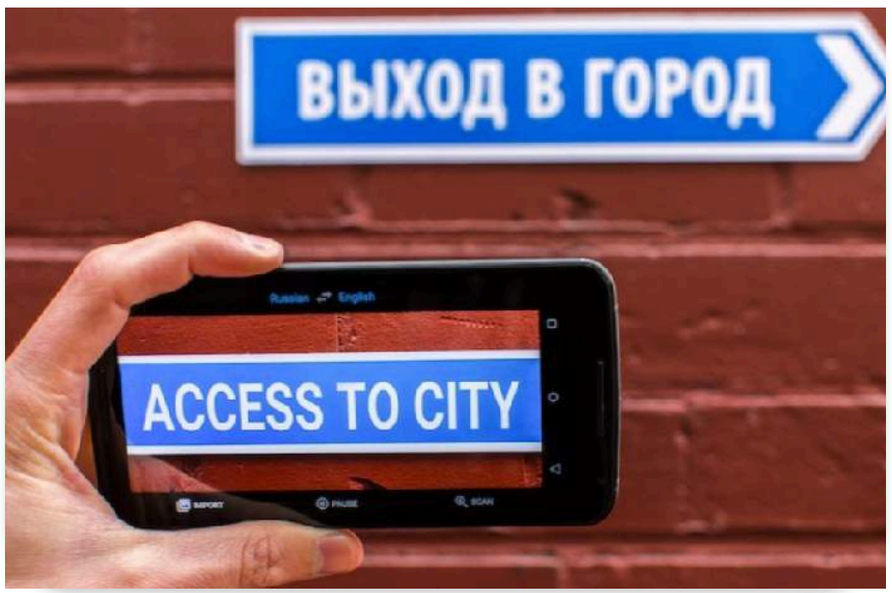
Bandwidth-Efficient Deep Learning
——from Compression to Acceleration

Song Han
Assistant Professor, EECS
Massachusetts Institute of Technology

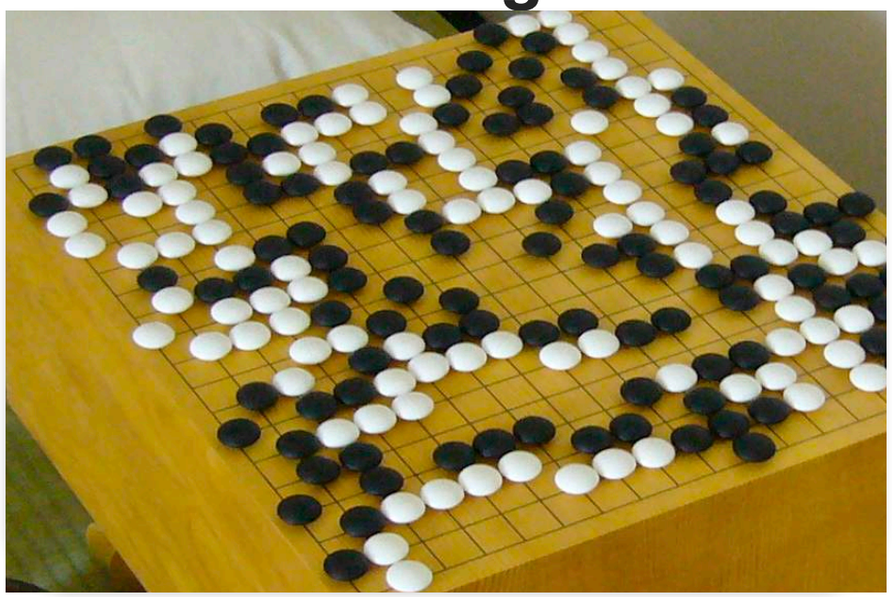
AI is Changing Our Lives



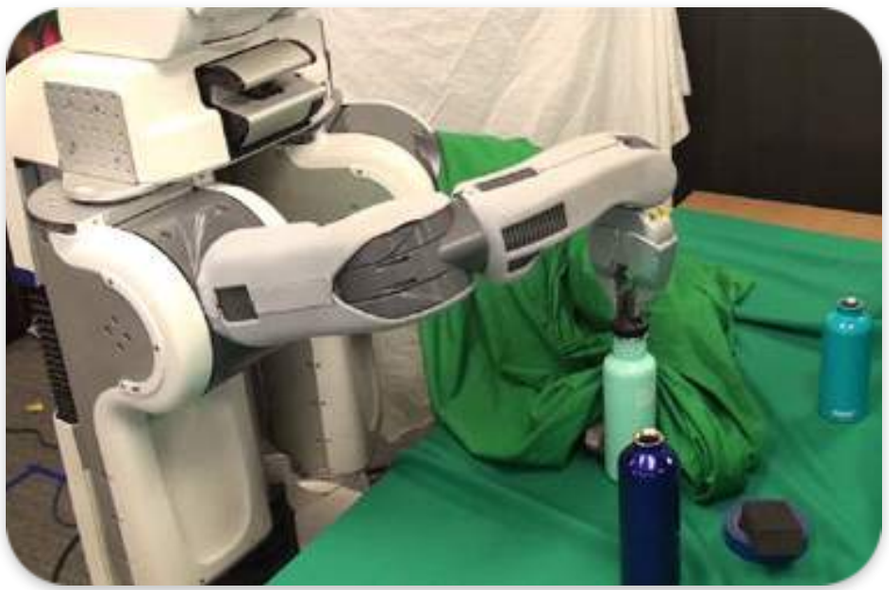
Self-Driving Car



Machine Translation



AlphaGo



Smart Robots

Models are Getting Larger

IMAGE RECOGNITION

16X
Model

8 layers
1.4 GFLOP
~16% Error

2012
AlexNet

152 layers
22.6 GFLOP
~3.5% error

2015
ResNet

SPEECH RECOGNITION

10X
Training Ops

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

2014
Deep Speech 1

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

2015
Deep Speech 2

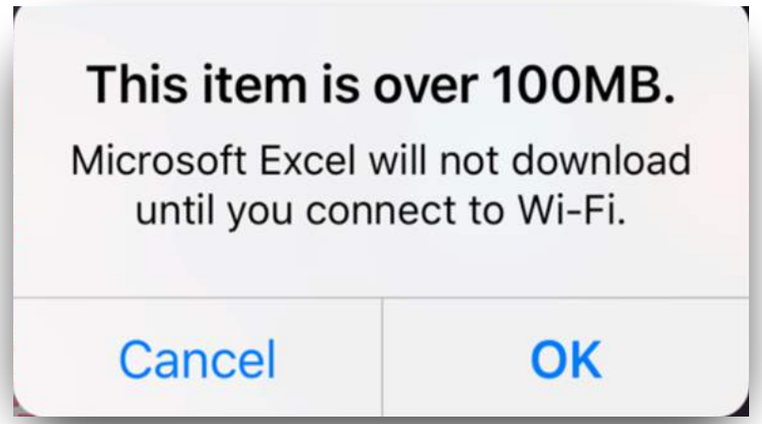
Microsoft

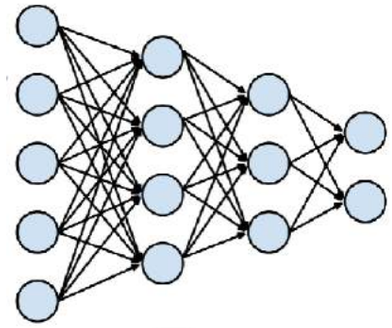
Baidu 百度

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

The first Challenge: Model Size

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





The Second Challenge: Speed

	Error rate	Training time
ResNet18:	10.76%	2.5 days
ResNet50:	7.02%	5 days
ResNet101:	6.21%	1 week
ResNet152:	6.16%	1.5 weeks

Such long training time limits ML researcher's productivity

Training time benchmarked with fb.resnet.torch using four M40 GPUs

The Third Challenge: Energy Efficiency

AlphaGo: 1920 CPUs and 280 GPUs,

$3000 electric bill per game

on mobile: drains battery

on data-center: increases TCO

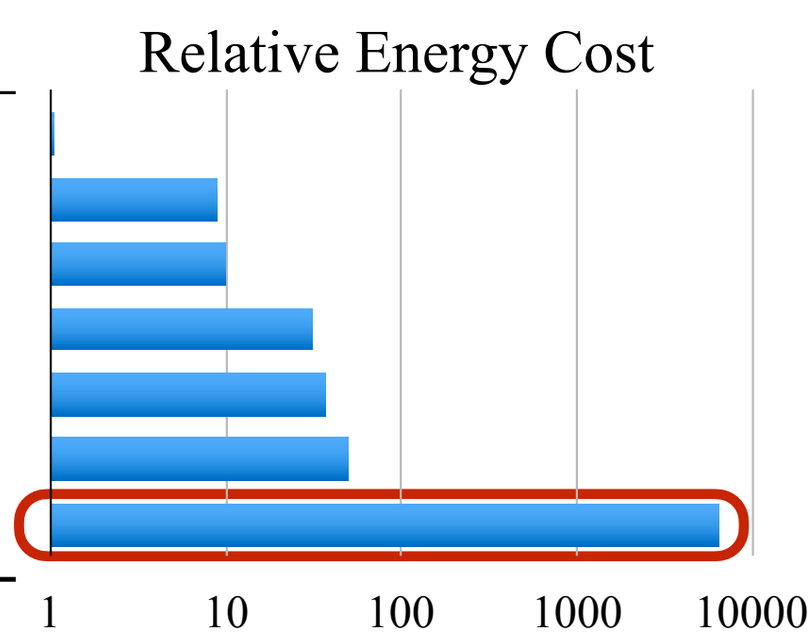
Where is the Energy Consumed?

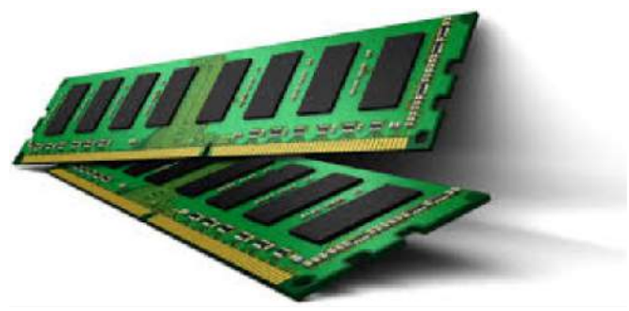
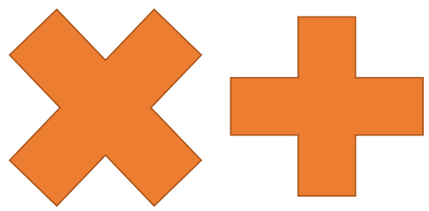
larger model => more memory reference => more energy

Where is the Energy Consumed?

larger model => more memory reference => more energy

Operation	Energy [pJ]
32 bit int ADD	0.1
32 bit float ADD	0.9
32 bit Register File	1
32 bit int MULT	3.1
32 bit float MULT	3.7
32 bit SRAM Cache	5
32 bit DRAM Memory	**640**

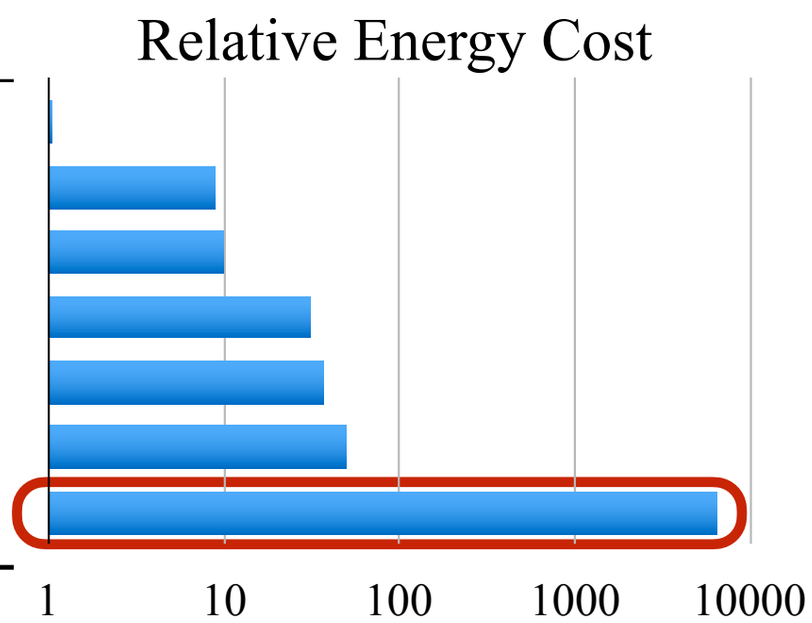


1  **= 1000** 

Where is the Energy Consumed?

larger model => more memory reference => more energy

Operation	Energy [pJ]
32 bit int ADD	0.1
32 bit float ADD	0.9
32 bit Register File	1
32 bit int MULT	3.1
32 bit float MULT	3.7
32 bit SRAM Cache	5
32 bit DRAM Memory	**640**



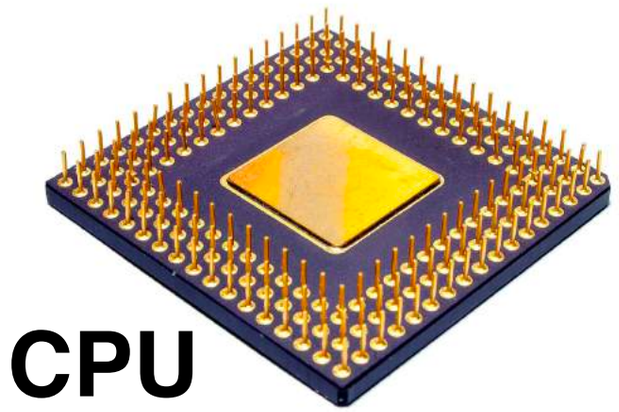
how to make deep learning more efficient?

Improve the Efficiency of Deep Learning by Algorithm-Hardware Co-Design

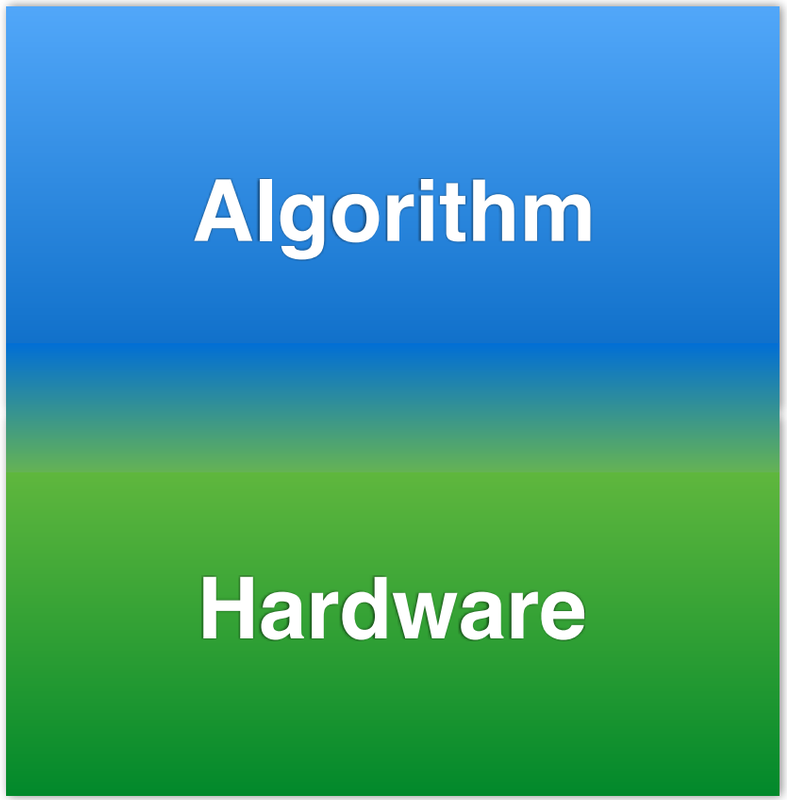
Application as a Black Box

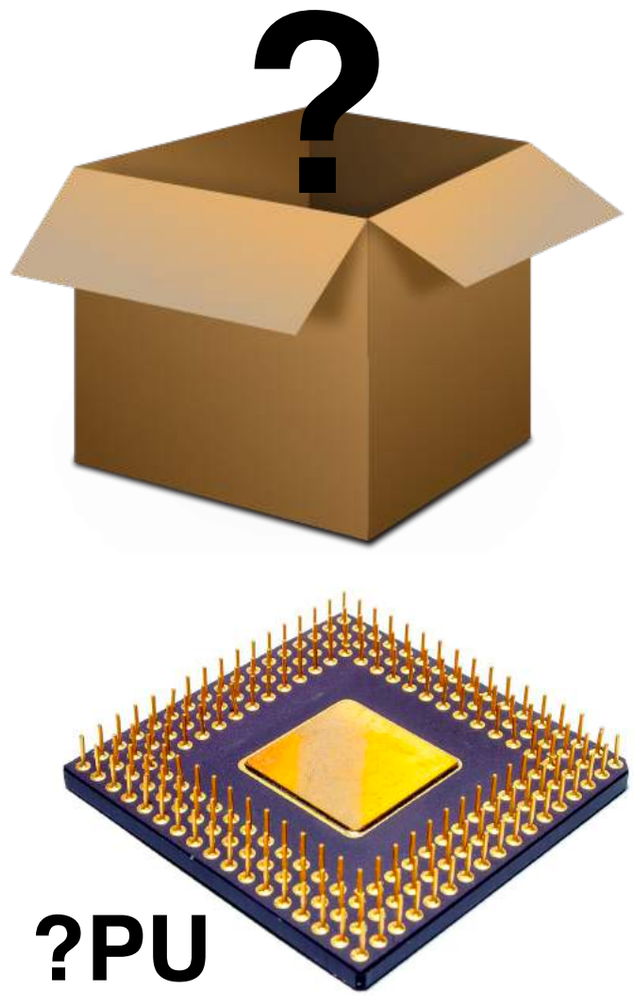
Algorithm

Hardware



Open the Box before Hardware Design

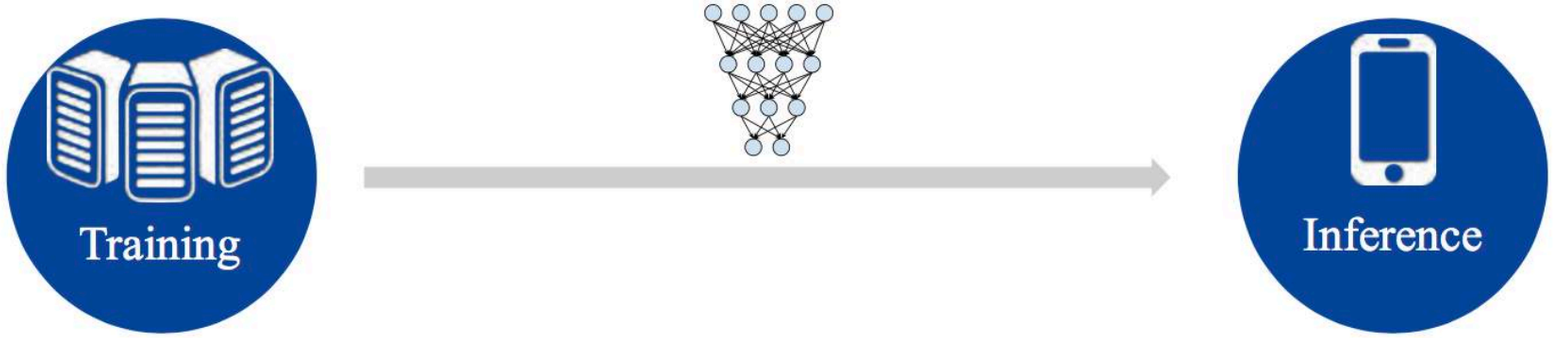




Breaks the boundary between algorithm and hardware

Proposed Paradigm

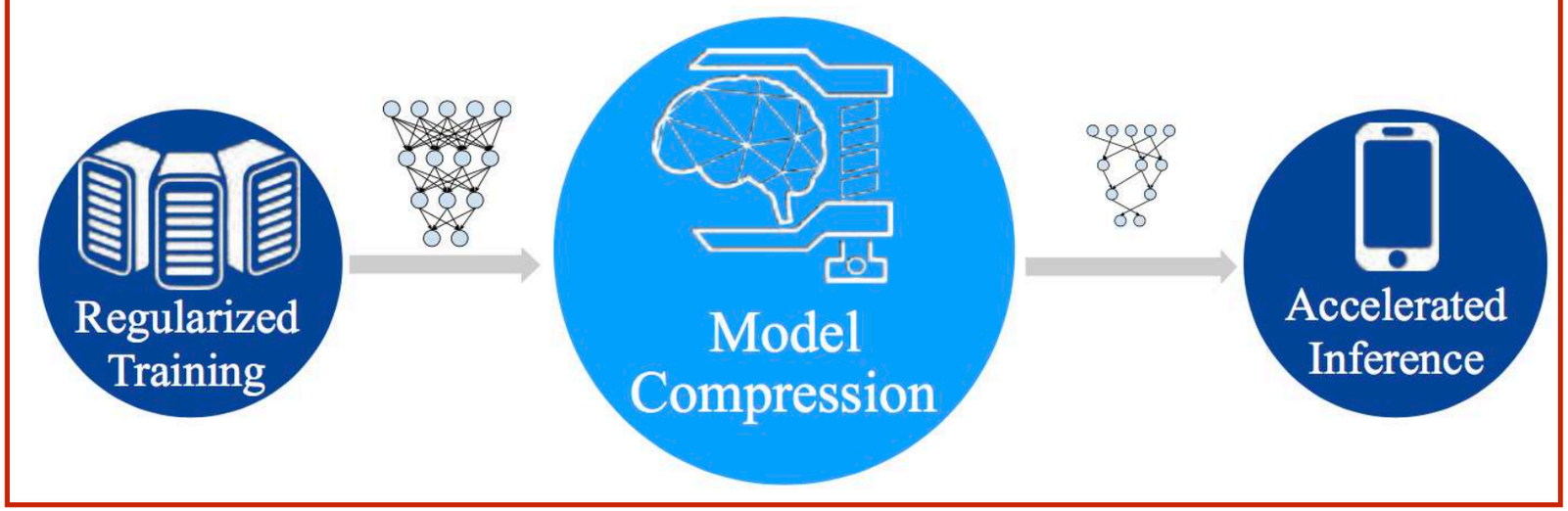
Conventional



Slow

Power-Hungry

Proposed



Fast

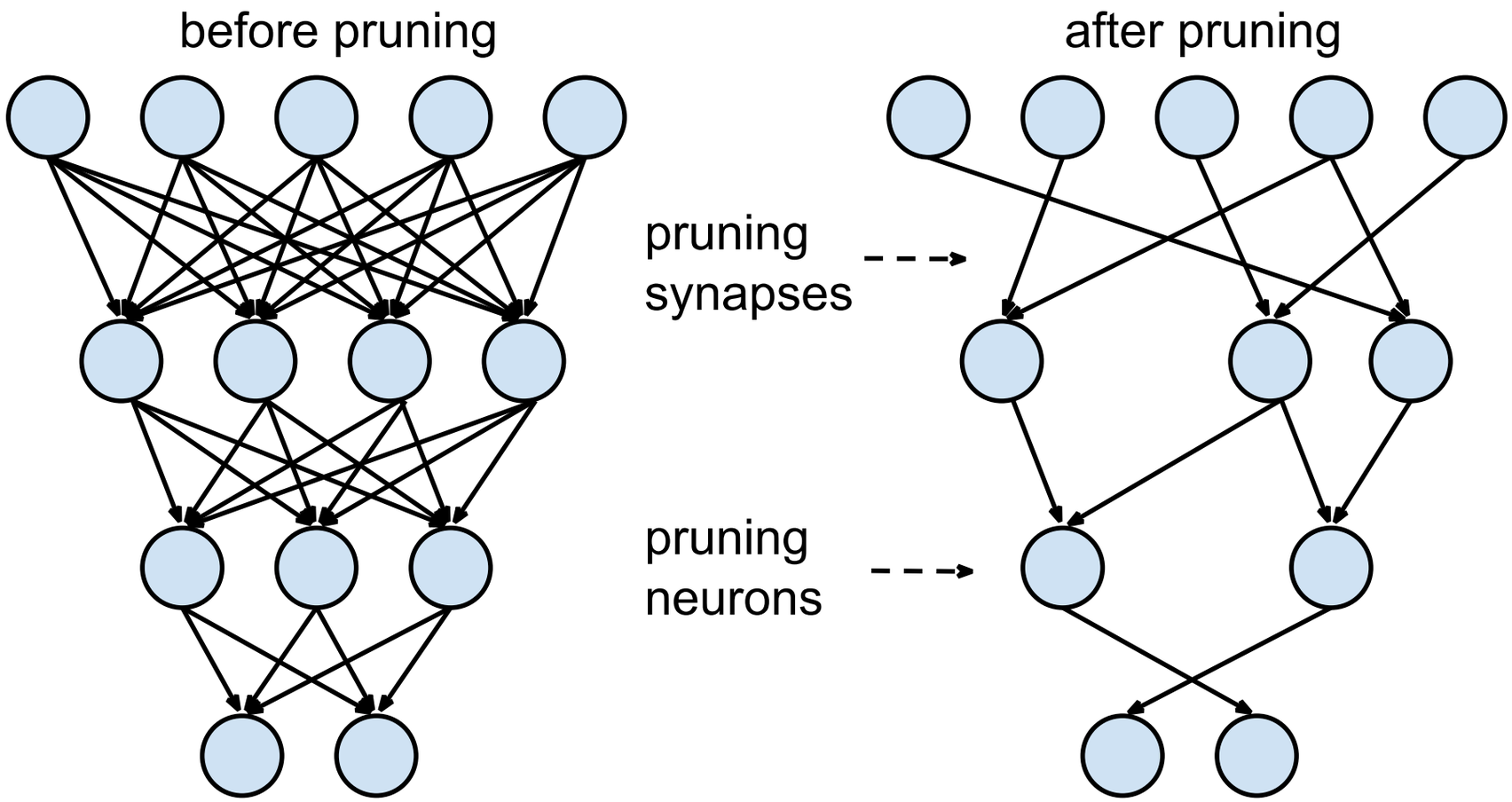
Power-Efficient

How to compress a deep learning model?

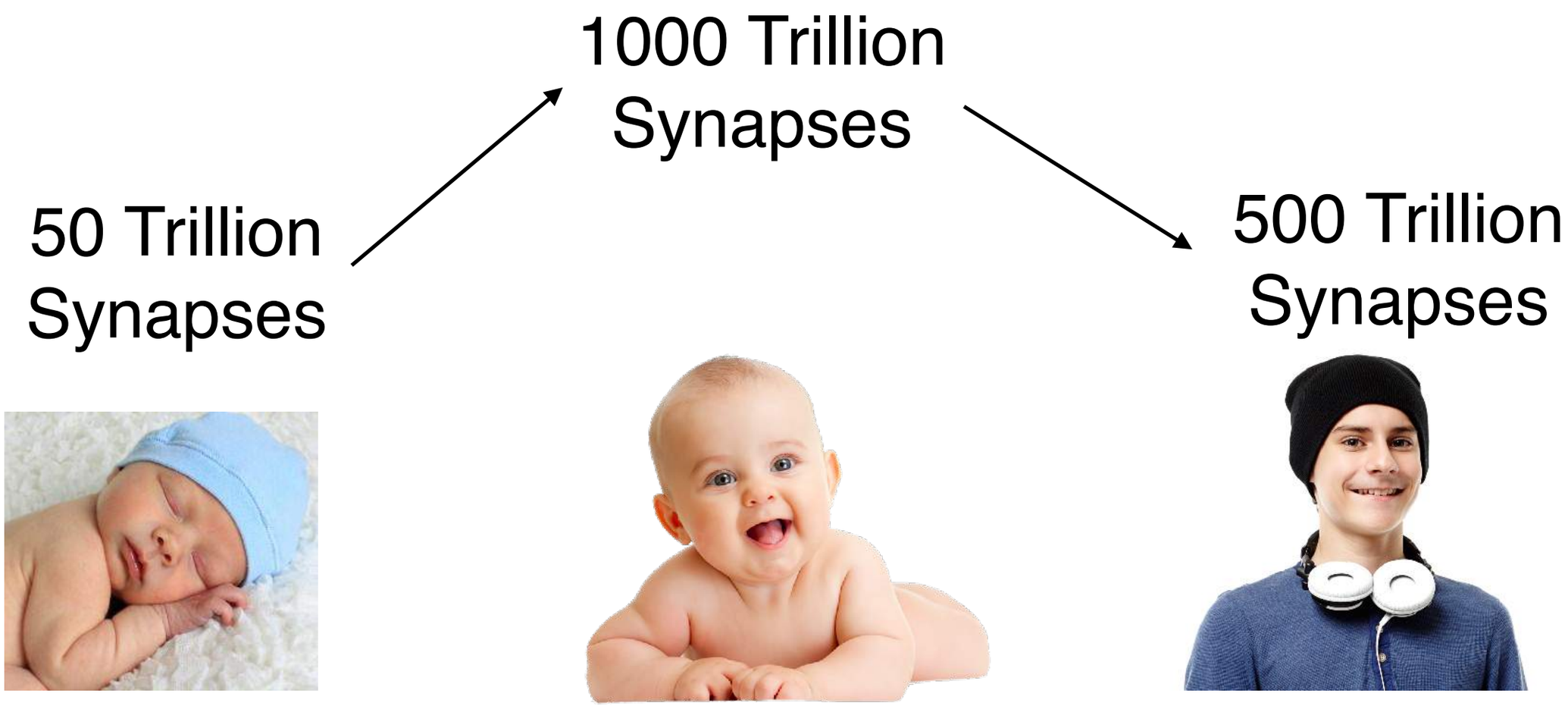
Learning both Weights and Connections for Efficient Neural Networks

Han et al.
NIPS 2015

Pruning Neural Networks



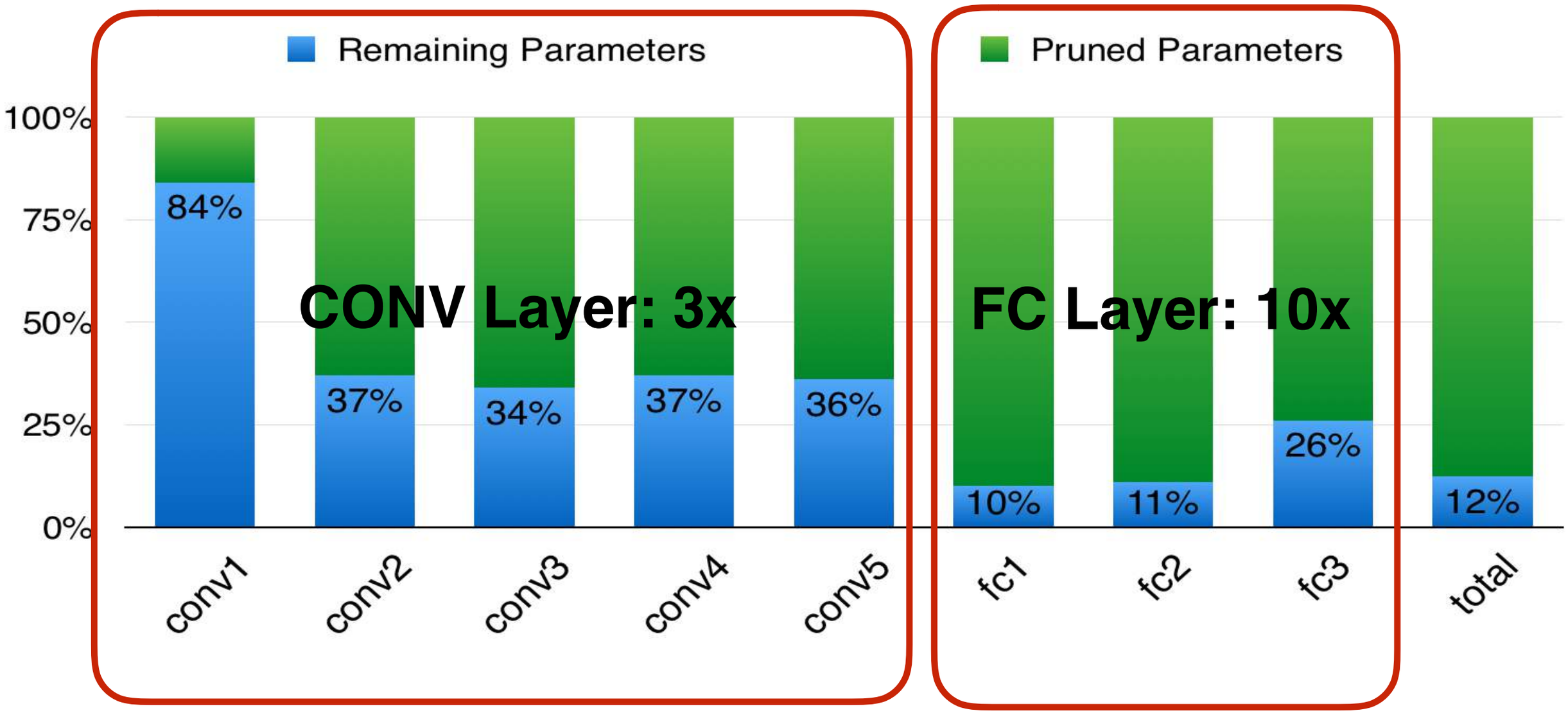
Pruning Happens in Human Brain



Christopher A Walsh. Peter Huttenlocher (1931-2013). Nature, 502(7470):172–172, 2013.

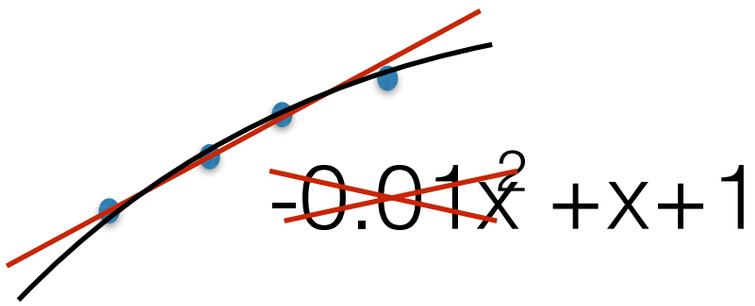
[Han et al. NIPS'15]

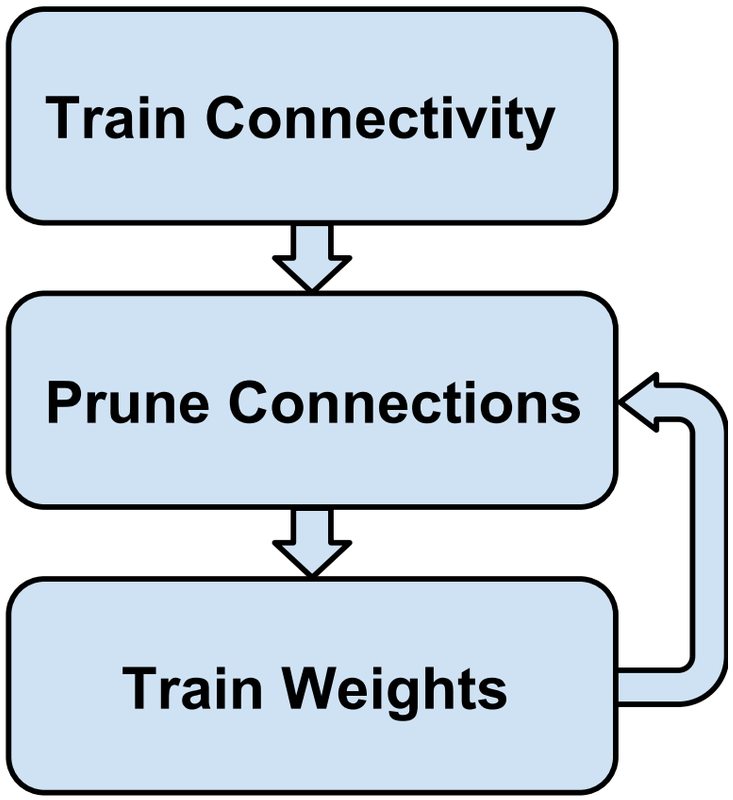
Pruning AlexNet



[Han et al. NIPS'15]

Pruning Neural Networks

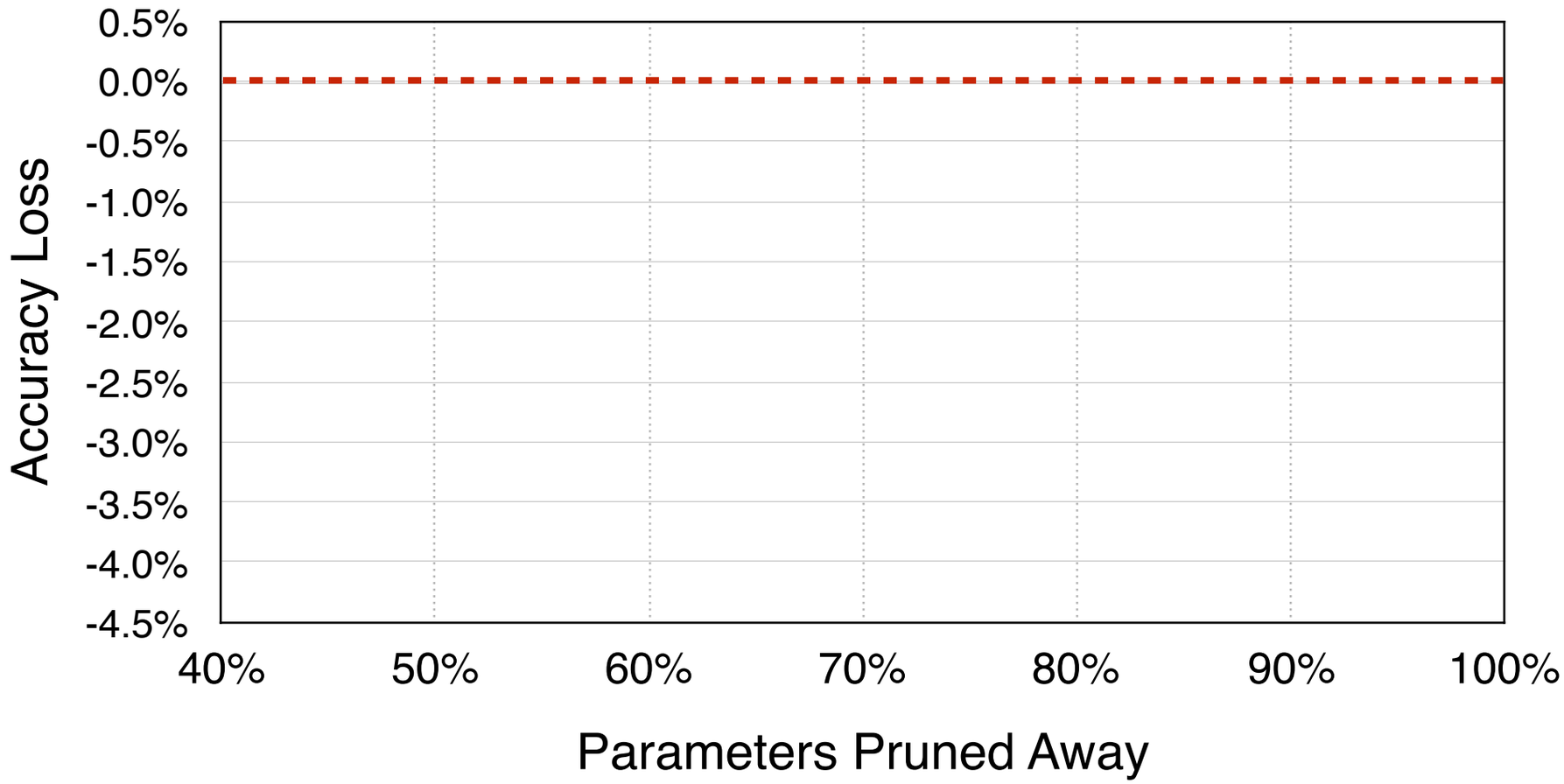




[Han et al. NIPS'15]

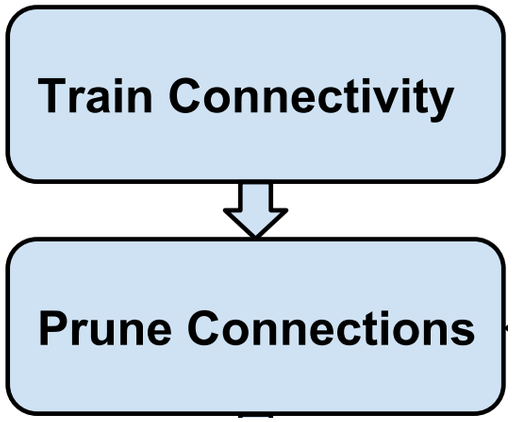
Pruning Neural Networks

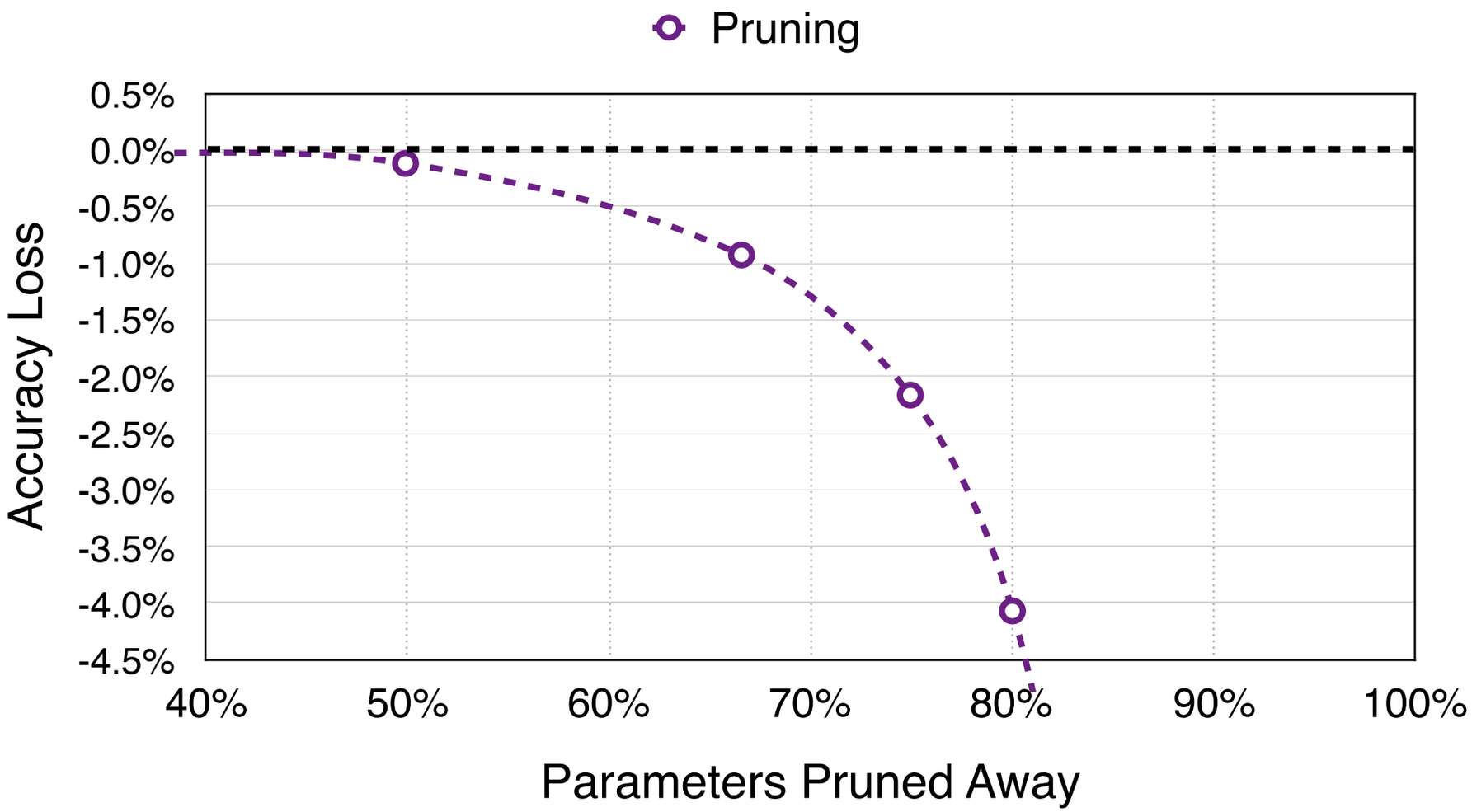
Train Connectivity



[Han et al. NIPS'15]

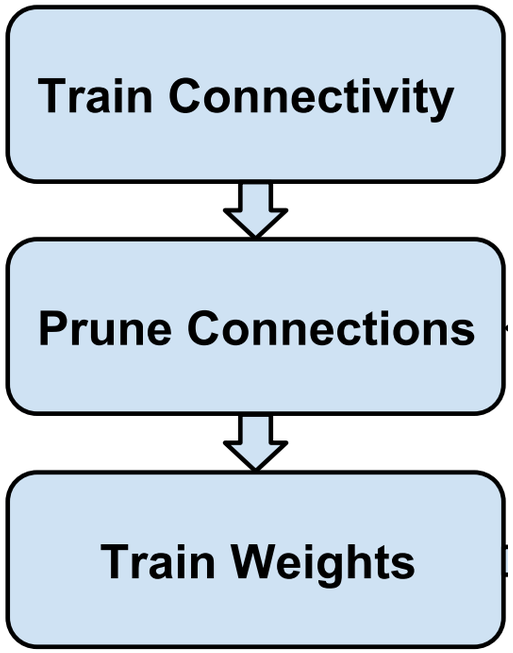
Pruning Neural Networks

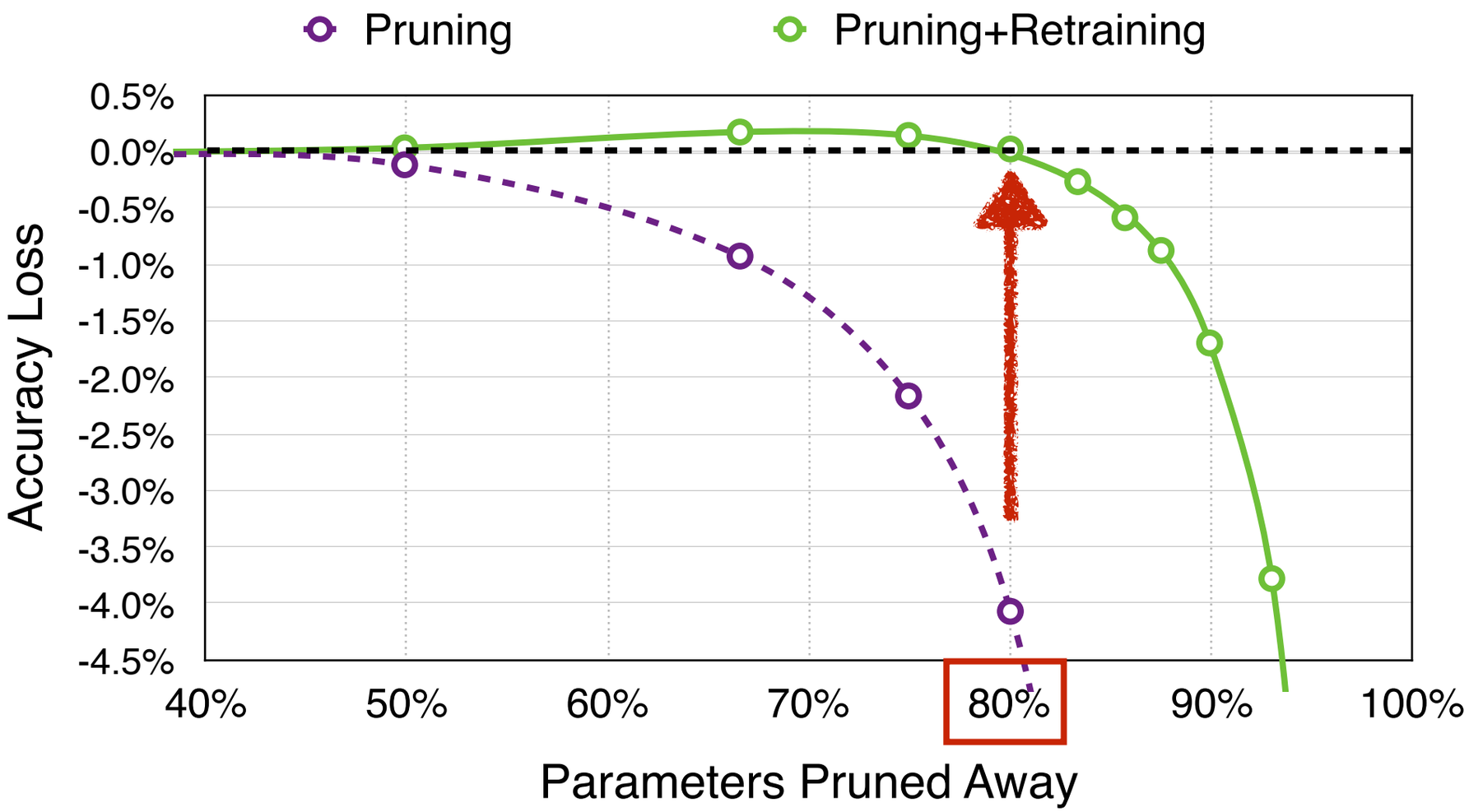




[Han et al. NIPS'15]

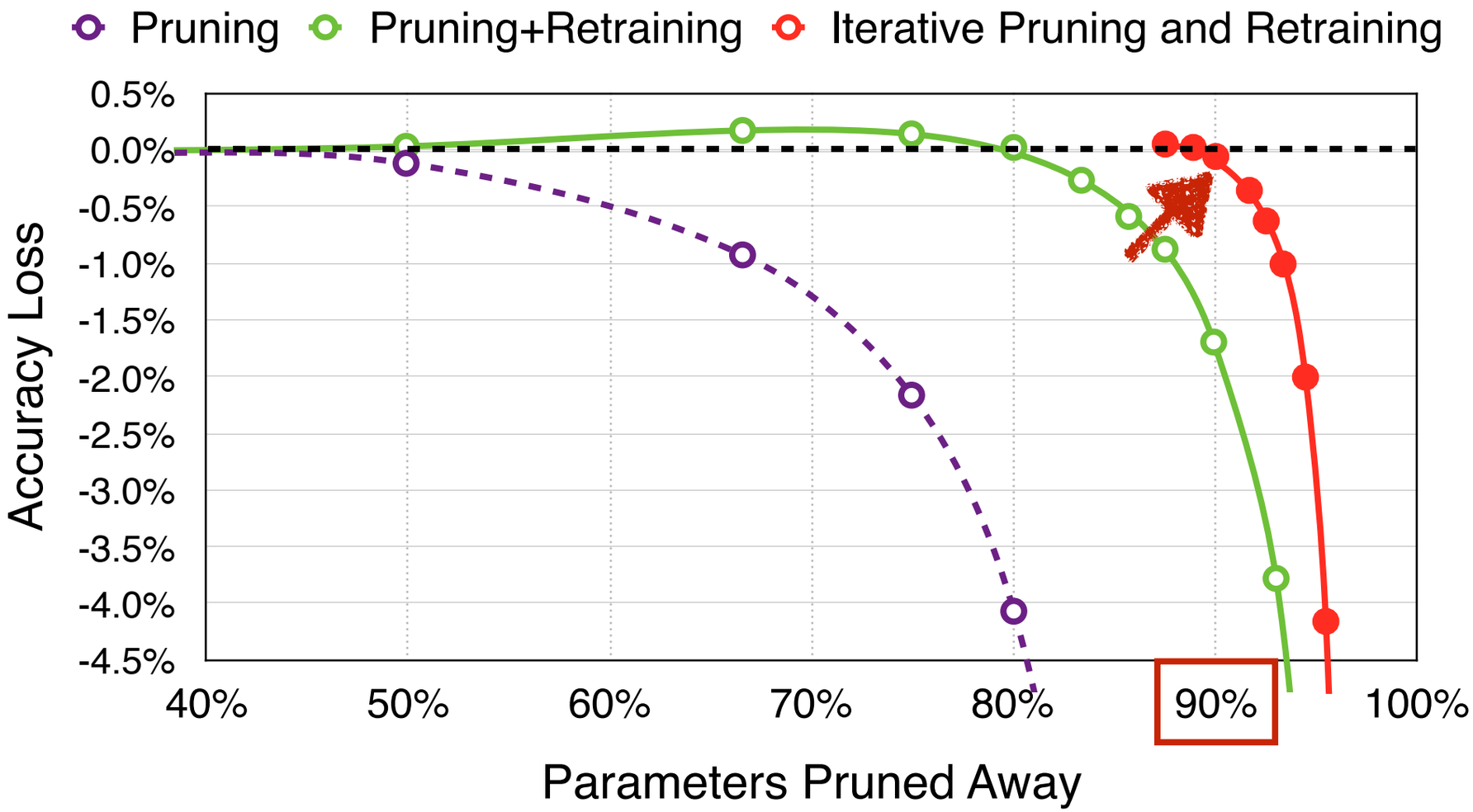
Retrain to Recover Accuracy





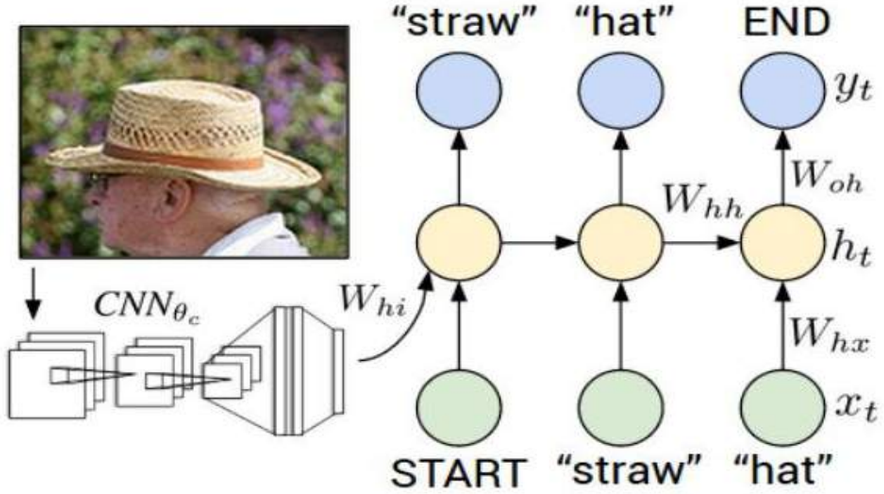
[Han et al. NIPS'15]

Iteratively Retrain to Recover Accuracy

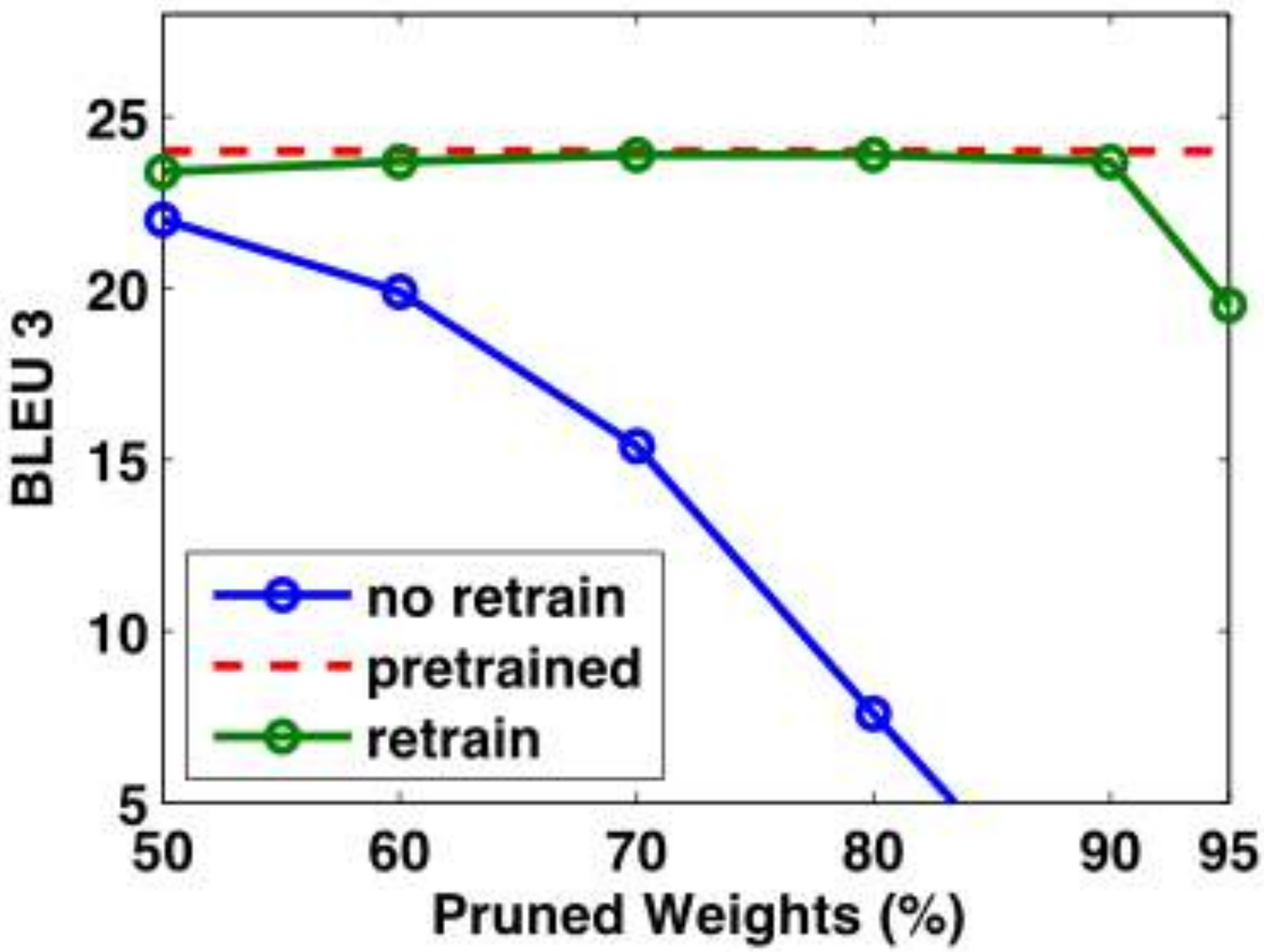


[Han et al. NIPS'15]

Pruning RNN and LSTM



*Karpathy et al "Deep Visual-Semantic Alignments for Generating Image Descriptions"



[Han et al. NIPS'15]

Pruning RNN and LSTM

90%

- **Original**: a basketball player in a white uniform is playing with a ball
- **Pruned 90%**: a basketball player in a white uniform is playing with a basketball

90%

- **Original** : a brown dog is running through a grassy field
- **Pruned 90%**: a brown dog is running through a grassy area

90%

- **Original** : a man is riding a surfboard on a wave
- **Pruned 90%**: a man in a wetsuit is riding a wave on a beach

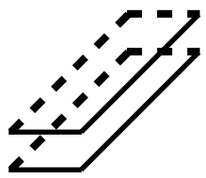
95%

- **Original** : a soccer player in red is running in the field
- **Pruned 95%**: a man in a red shirt and black and white black shirt is running through a field

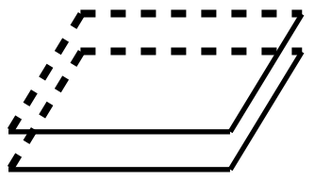
Exploring the Granularity of Sparsity that is Hardware-friendly

4 types of pruning granularity

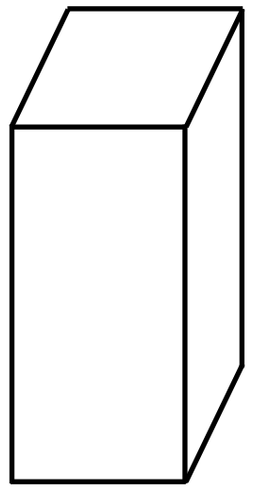
irregular sparsity



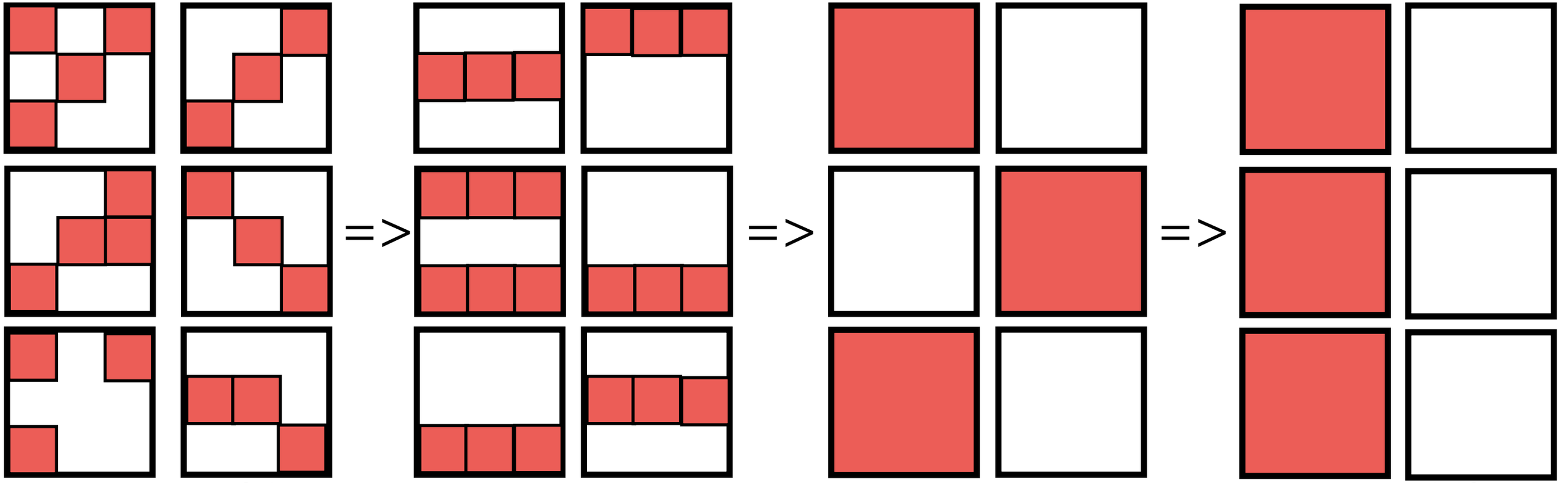
regular sparsity



more regular sparsity



fully-dense model



[Han et al, NIPS'15]

[Molchanov et al, ICLR'17]

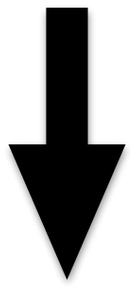
Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding

Han et al.
ICLR 2016
Best Paper

[Han et al. ICLR'16]

Trained Quantization

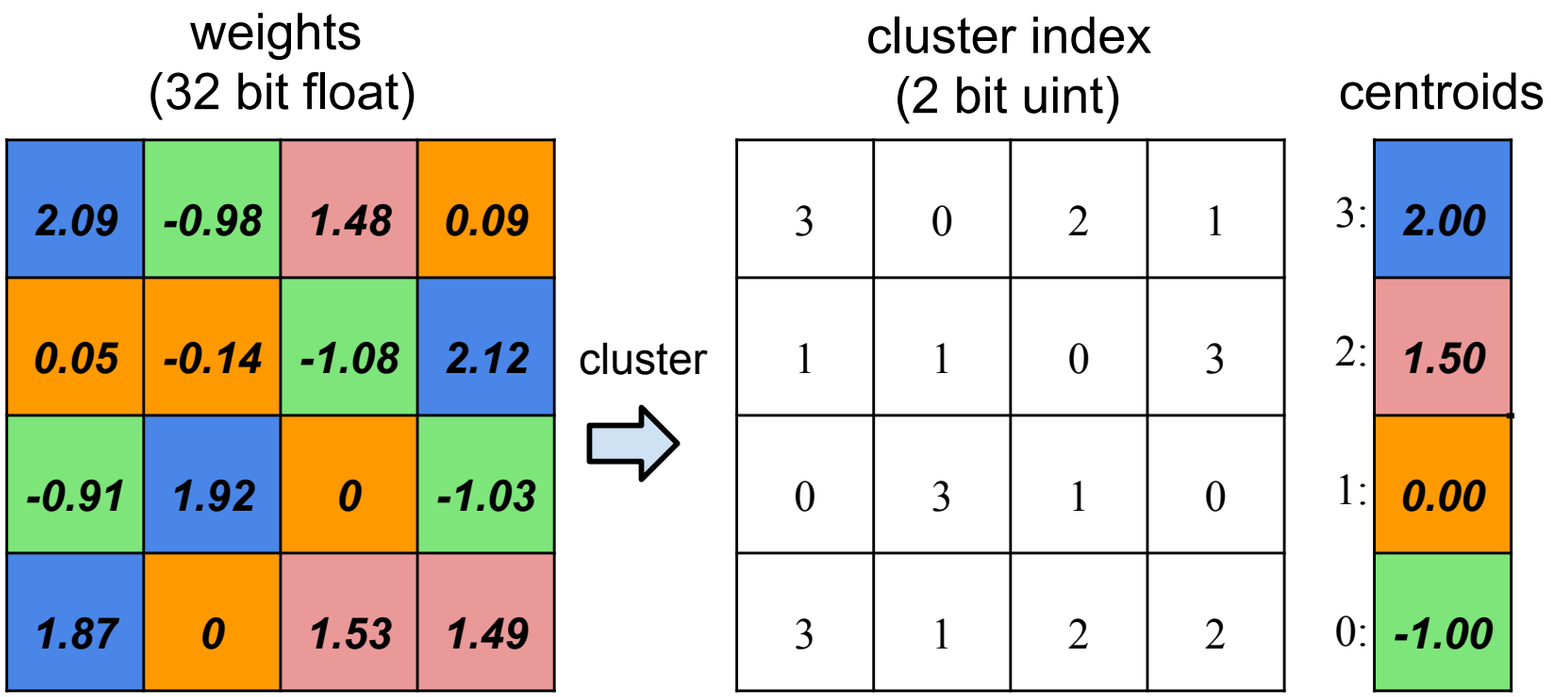
~~2.09, 2.12, 1.92, 1.87~~



2.0

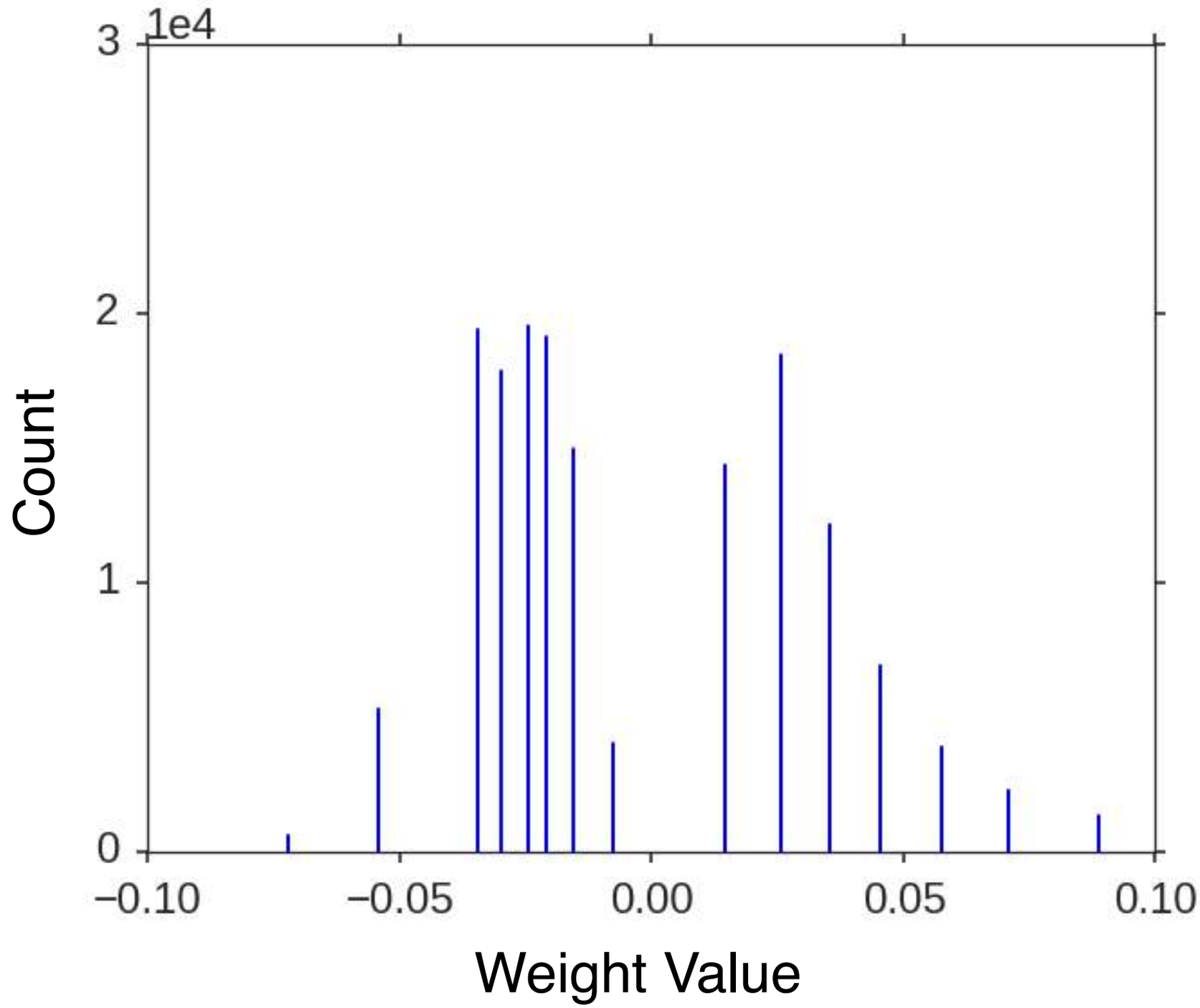
[Han et al. ICLR'16]

Trained Quantization



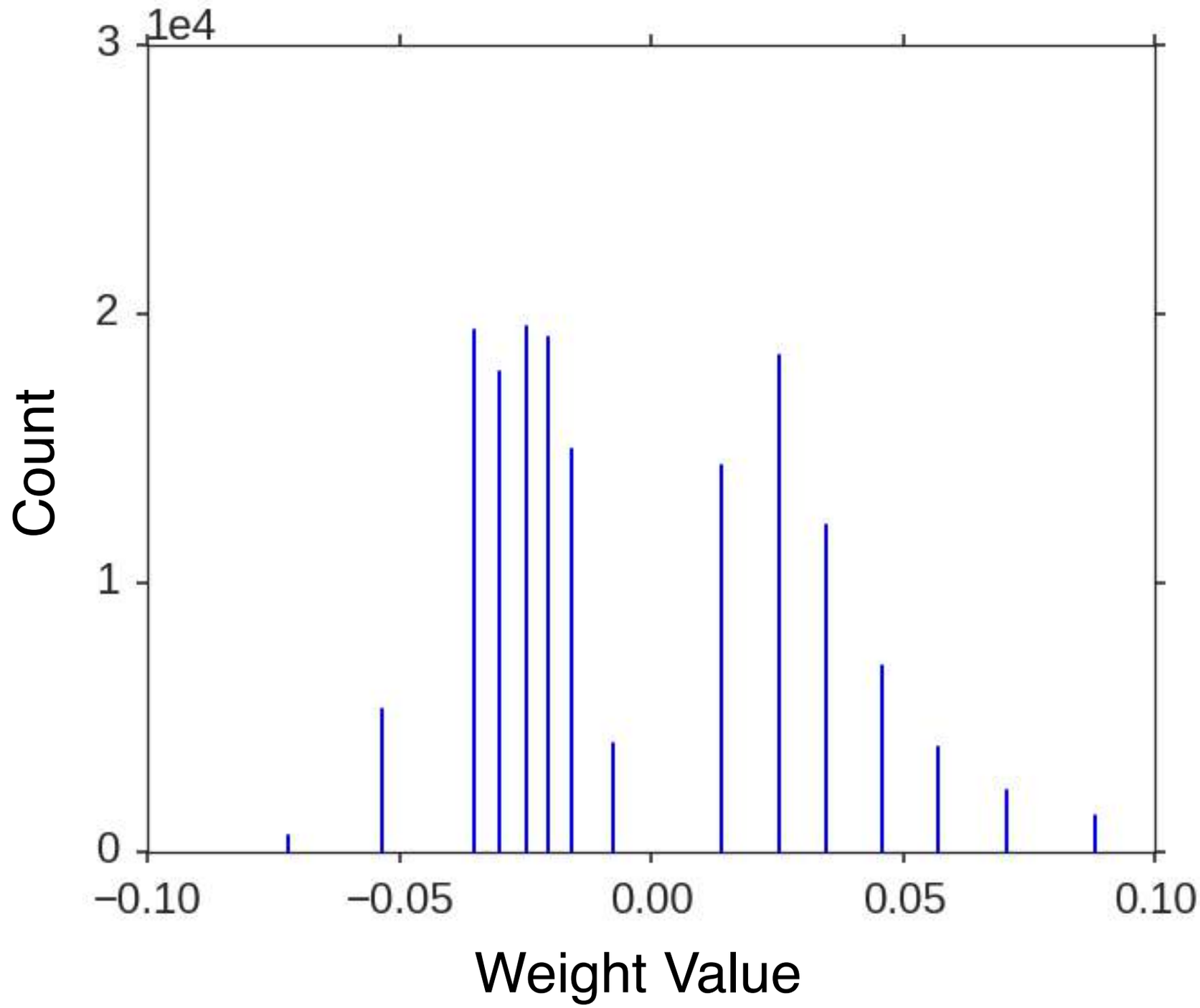
[Han et al. ICLR'16]

After Trained Quantization: Discrete Weight



[Han et al. ICLR'16]

After Trained Quantization: Discrete Weight after Training

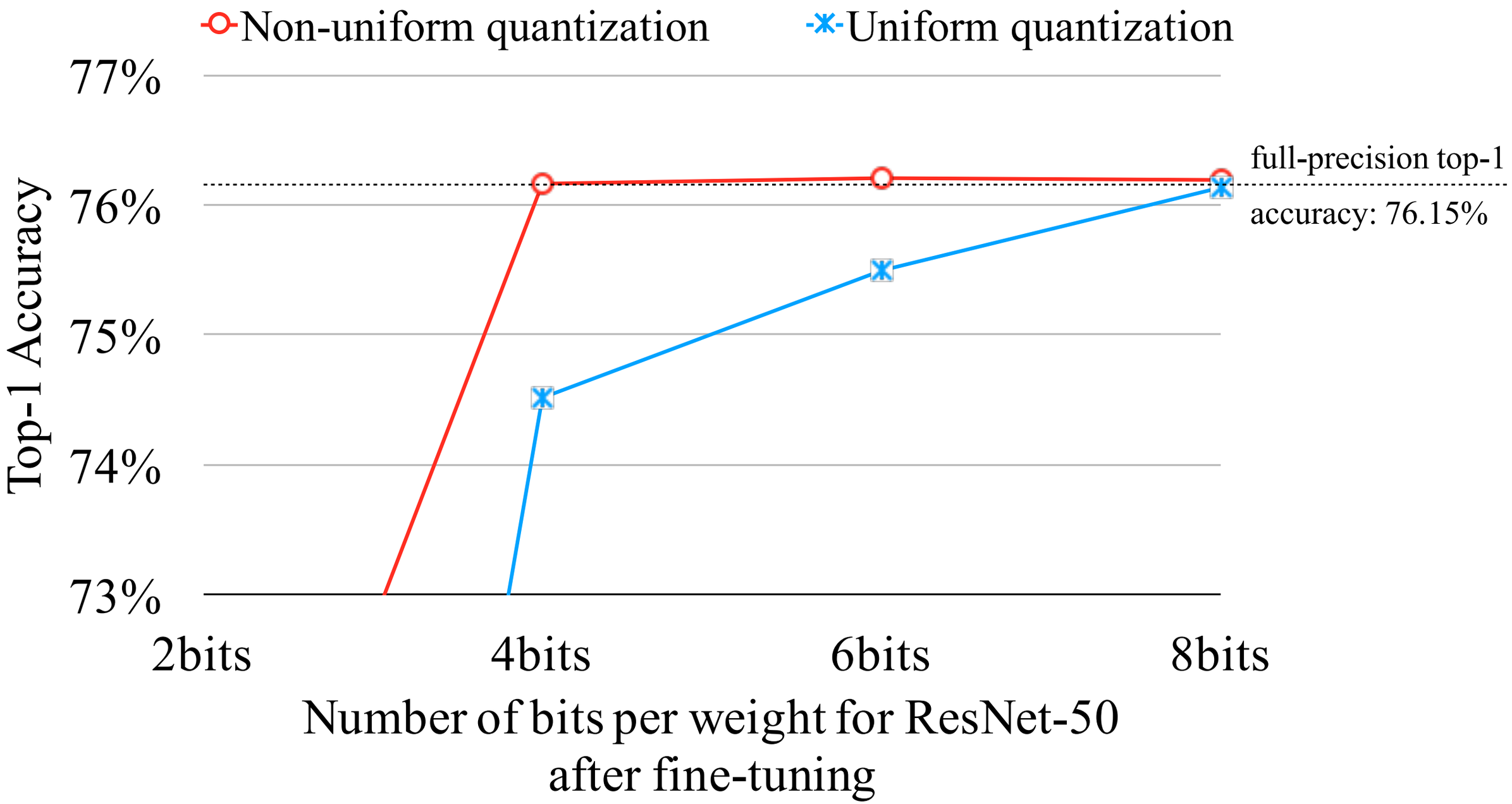


[Han et al. ICLR'16]

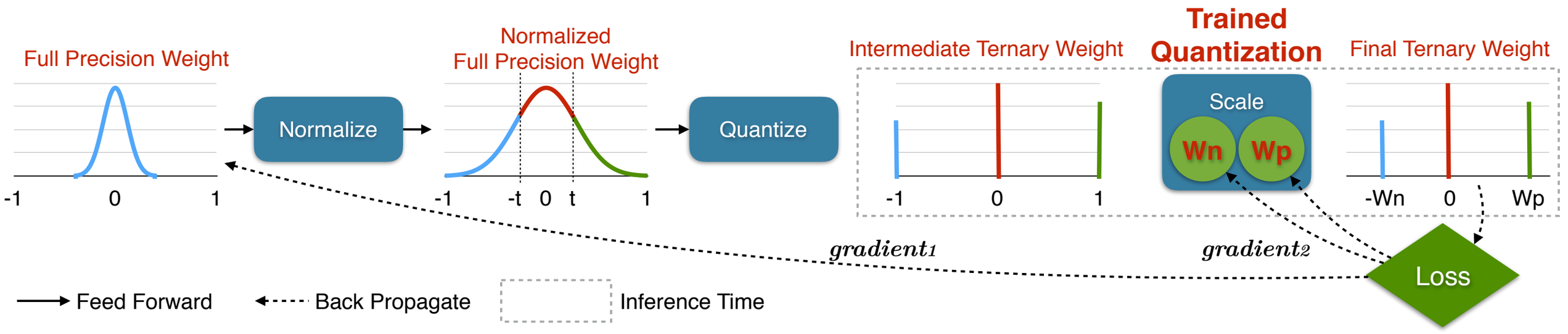
How Many Bits do We Need?

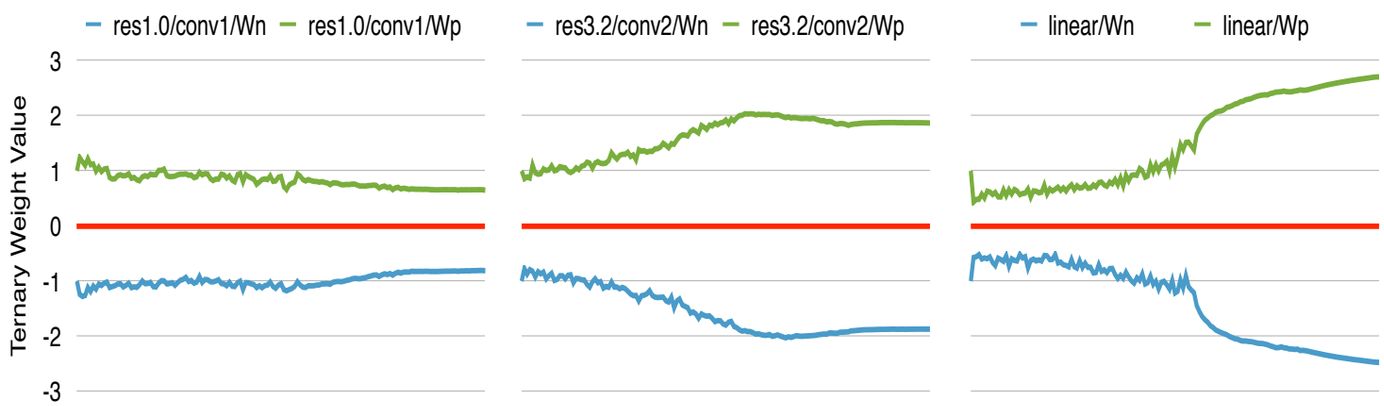
[Han et al. ICLR'16]

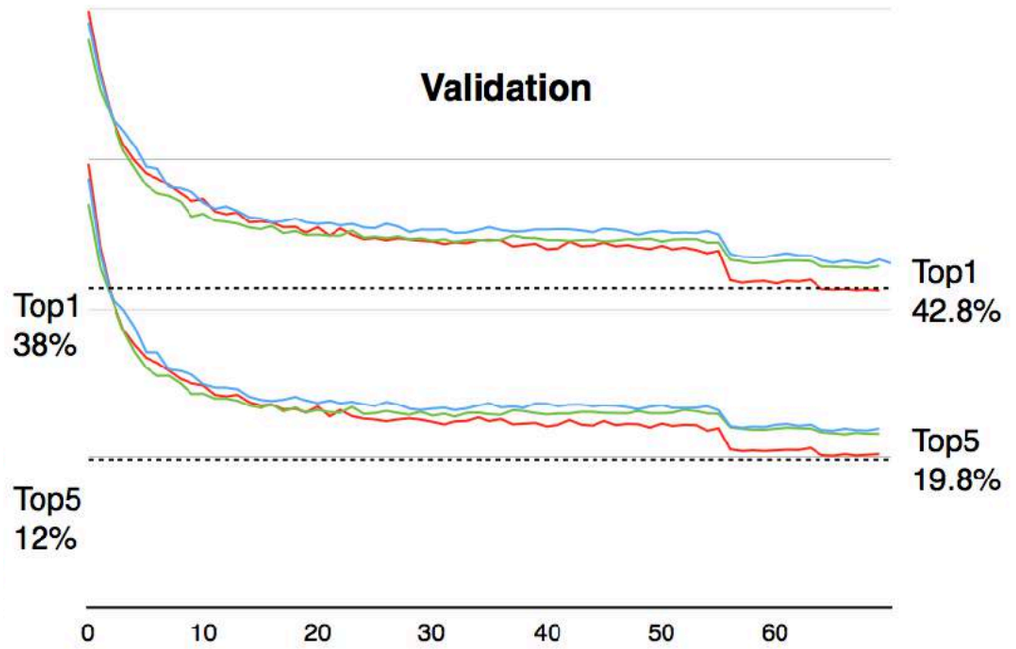
How Many Bits do We Need?



More Aggressive Compression: Ternary Quantization







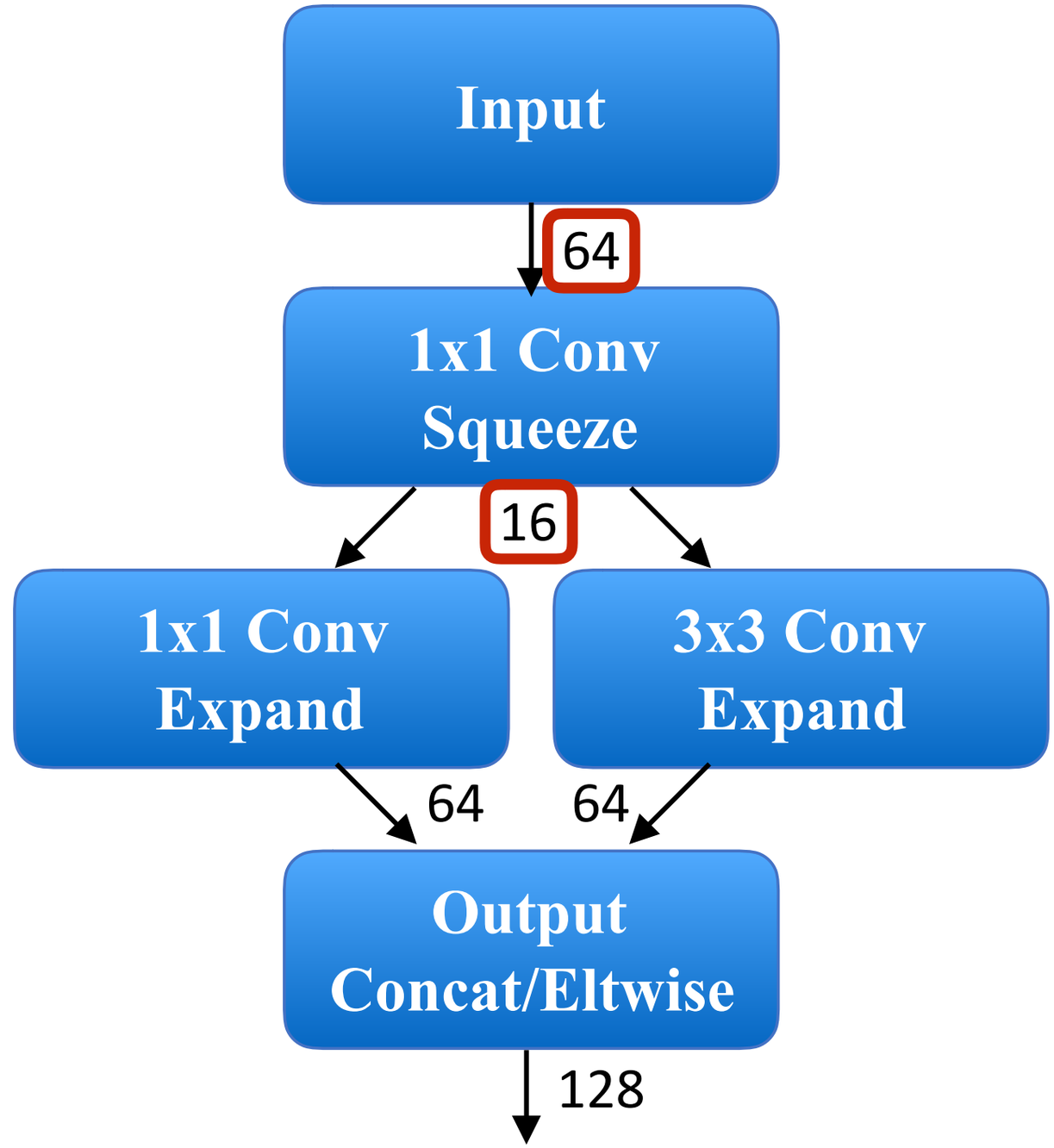
[Han et al. ICLR'16]

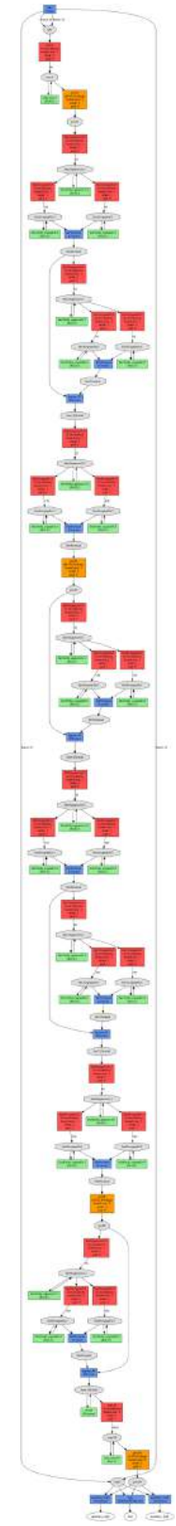
Results: Compression Ratio

Network	Original Size	Compressed Size	Compression Ratio	Original Accuracy	Compressed Accuracy
LeNet-300	1070KB	27KB	**40x**	98.36%	98.42%
LeNet-5	1720KB	44KB	**39x**	99.20%	99.26%
AlexNet	240MB	6.9MB	**35x**	80.27%	80.30%
VGGNet	550MB	11.3MB	**49x**	88.68%	89.09%
Inception-V3	91MB	4.2MB	**22x**	93.56%	93.67%
ResNet-50	97MB	5.8MB	**17x**	92.87%	93.04%

Can we make compact models to begin with?

SqueezeNet



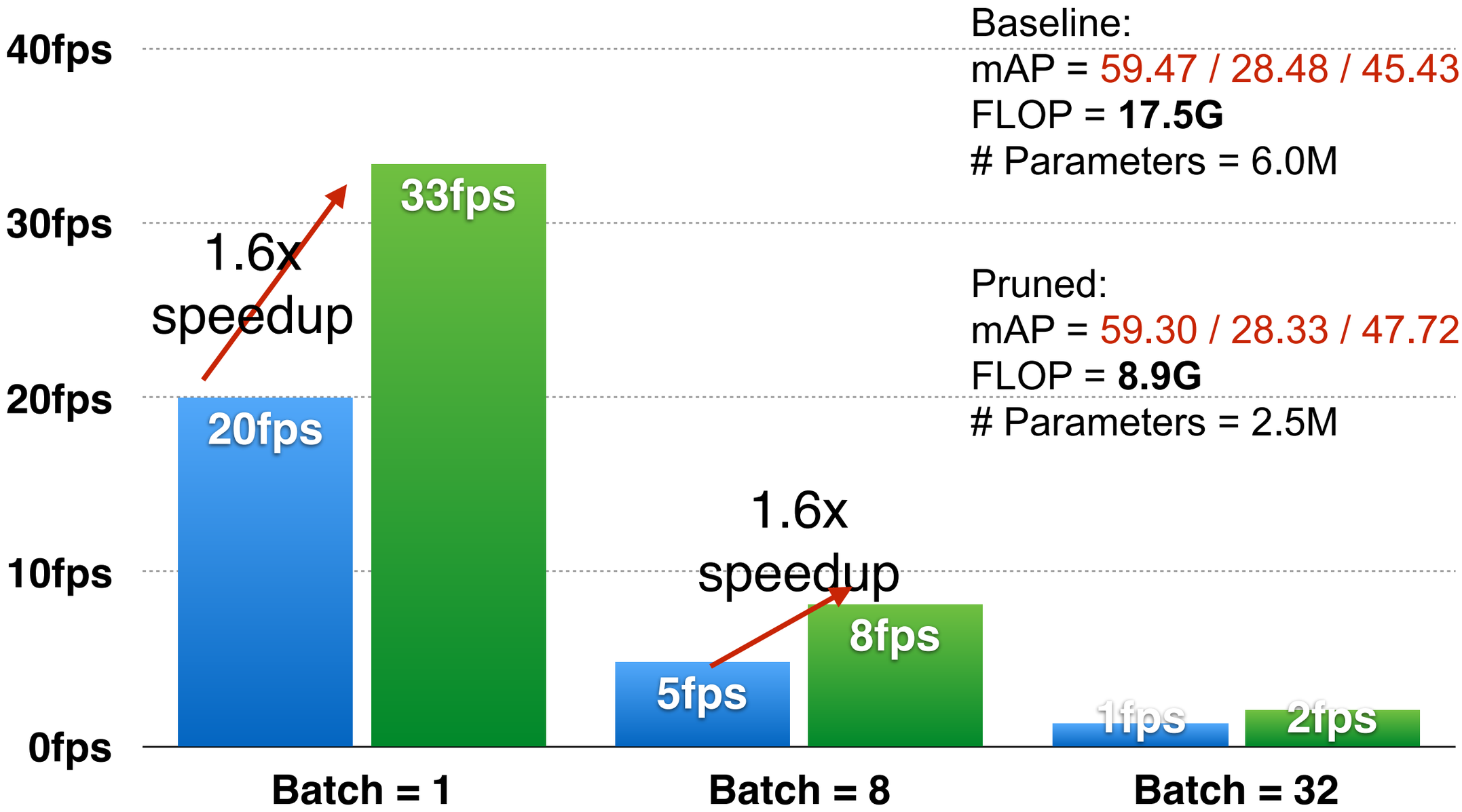


Iandola et al, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size”, arXiv 2016

Compressing SqueezeNet

Network	Approach	Size	Ratio	Top-1 Accuracy	Top-5 Accuracy
AlexNet	-	240MB	**1x**	57.2%	80.3%
AlexNet	SVD	48MB	**5x**	56.0%	79.4%
AlexNet	Deep Compression	6.9MB	**35x**	57.2%	80.3%
SqueezeNet	-	4.8MB	**50x**	57.5%	80.3%
SqueezeNet	Deep Compression	0.47MB	**510x**	57.5%	80.3%

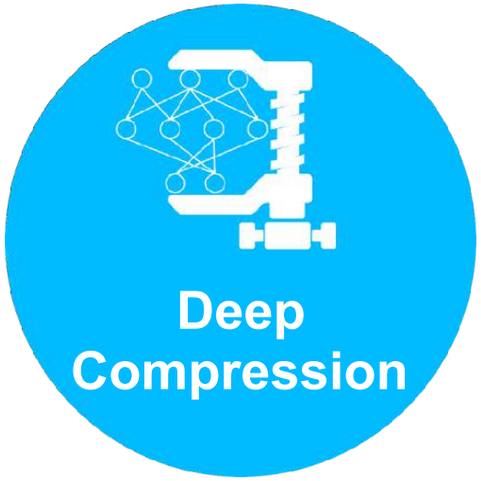
Results: Speedup



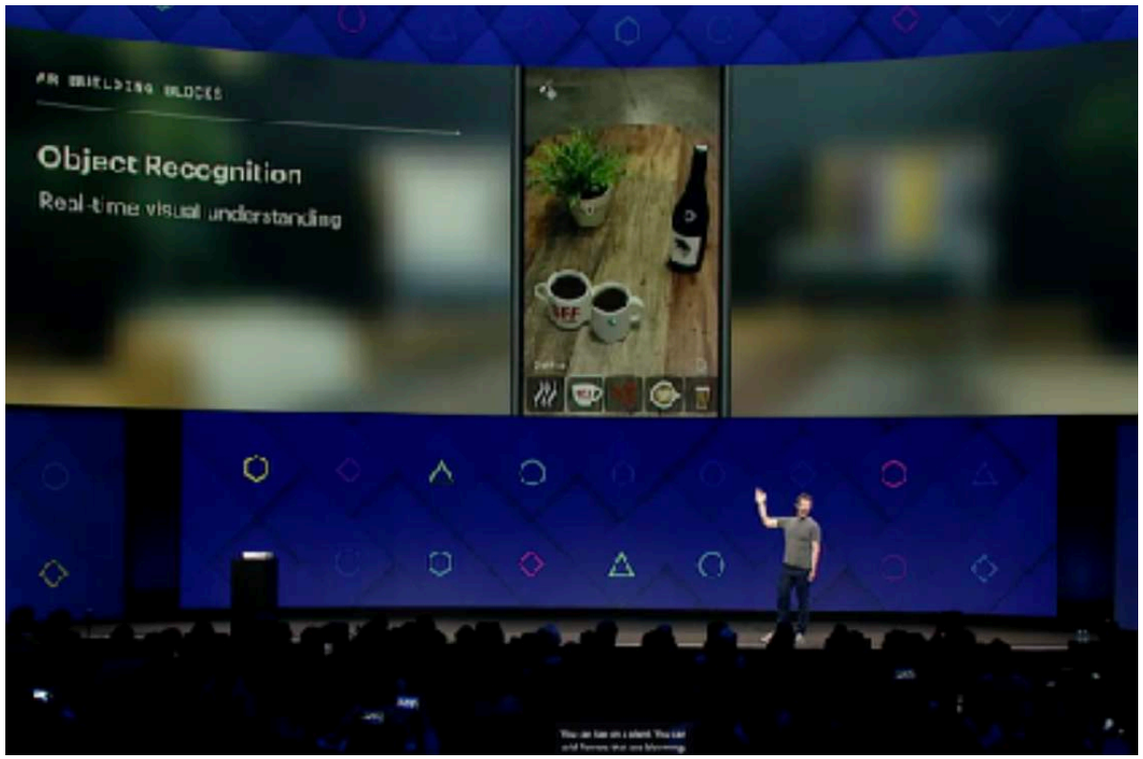
Baseline:
mAP = 59.47 / 28.48 / 45.43
FLOP = **17.5G**
# Parameters = 6.0M

Pruned:
mAP = 59.30 / 28.33 / 47.72
FLOP = **8.9G**
# Parameters = 2.5M

Deep Compression Applied to Industry



A Facebook AR prototype backed by Deep Compression



- 8x model size reduction by Deep Compression
- Run on mobile
- Source: Facebook F8’17

EIE: Efficient Inference Engine on Compressed Deep Neural Network

Han et al.
ISCA 2016

Deep Learning Accelerators

- First Wave: Compute (Neu Flow)
- Second Wave: Memory (Diannao family)
- Third Wave: Algorithm / Hardware Co-Design (EIE)

Google TPU: “This unit is designed for dense matrices. Sparse architectural support was omitted for time-to-deploy reasons. Sparsity will have high priority in future designs”

[Han et al. ISCA'16]

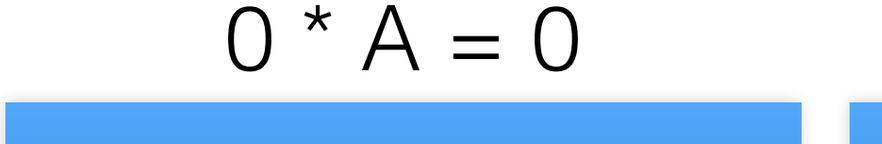
EIE: the First DNN Accelerator for Sparse, Compressed Model

0 * A = 0

Sparse Weight

90% *static* sparsity

10x less computation



5x less memory footprint

W * 0 = 0

Sparse Activation

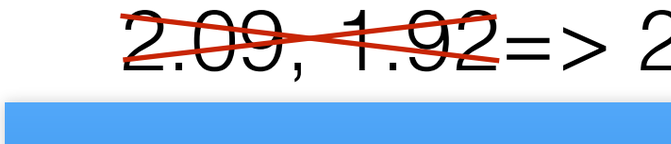
70% *dynamic* sparsity

3x less computation

~~2.09, 1.92~~=> 2

Weight Sharing

4-bit weights



8x less memory footprint

[Han et al. ISCA'16]

EIE: Parallelization on Sparsity

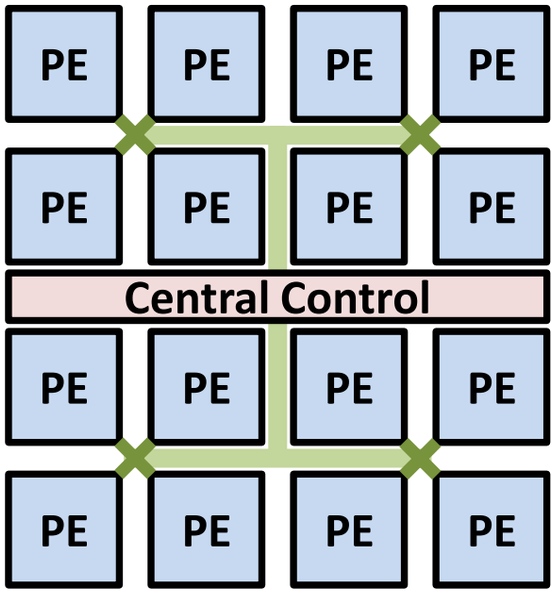
$$\vec{a} \begin{pmatrix} 0 & \boldsymbol{a_1} & 0 & a_3 \end{pmatrix}$$

$$\times$$

$$\begin{pmatrix} w_{0,0} & \boldsymbol{w_{0,1}} & 0 & w_{0,3} \\ 0 & \mathbf{0} & w_{1,2} & 0 \\ 0 & \boldsymbol{w_{2,1}} & 0 & w_{2,3} \\ 0 & \mathbf{0} & 0 & 0 \\ 0 & \mathbf{0} & w_{4,2} & w_{4,3} \\ w_{5,0} & \mathbf{0} & 0 & 0 \\ 0 & \mathbf{0} & 0 & w_{6,3} \\ 0 & \boldsymbol{w_{7,1}} & 0 & 0 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ -b_2 \\ b_3 \\ -b_4 \\ b_5 \\ b_6 \\ -b_7 \end{pmatrix} \overset{ReLU}{\Rightarrow} \begin{matrix} \vec{b} \\ \begin{pmatrix} b_0 \\ b_1 \\ 0 \\ b_3 \\ 0 \\ b_5 \\ b_6 \\ 0 \end{pmatrix} \end{matrix}$$

[Han et al. ISCA'16]

EIE: Parallelization on Sparsity



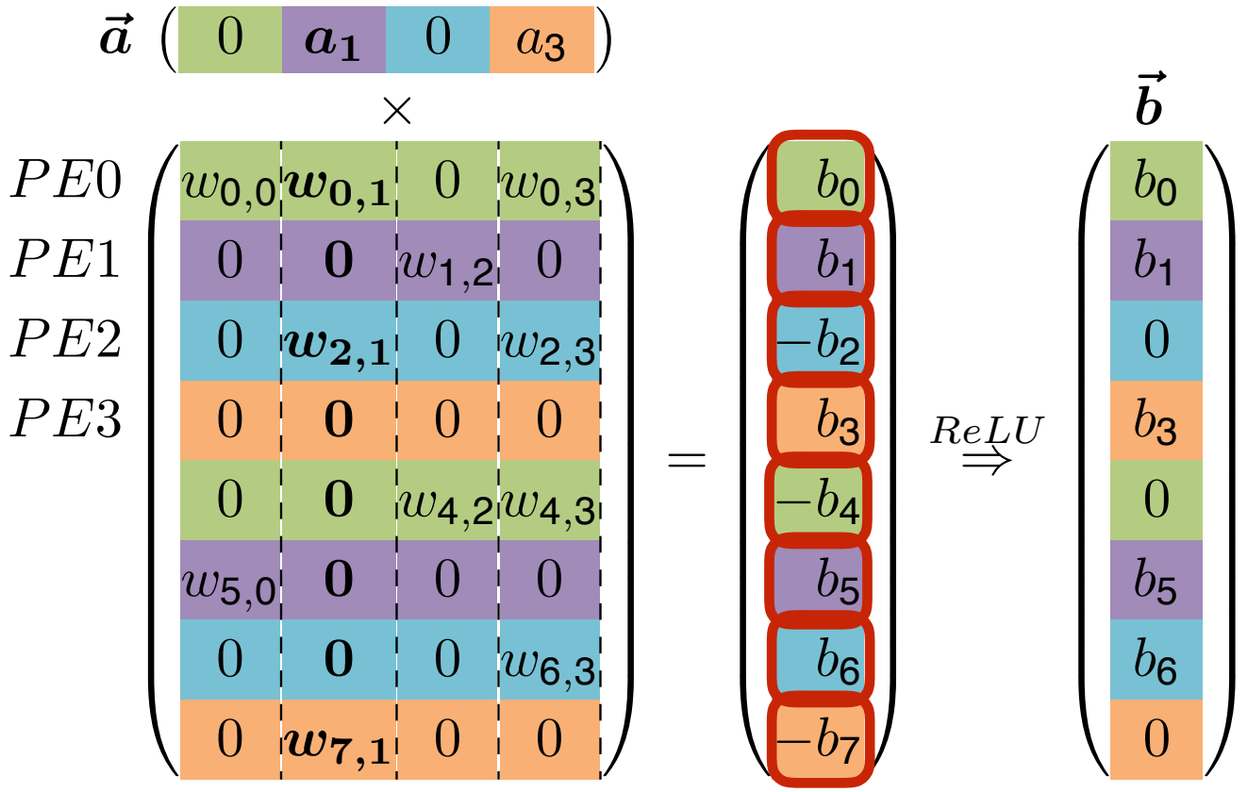
$$\vec{a}\ (\ 0 \quad a_1 \quad 0 \quad a_3\ )$$

$$\times$$

$$\begin{array}{l} PE0 \\ PE1 \\ PE2 \\ PE3 \\ \\ \\ \\ \end{array} \begin{pmatrix} w_{0,0} & \boldsymbol{w_{0,1}} & 0 & w_{0,3} \\ 0 & \mathbf{0} & w_{1,2} & 0 \\ 0 & \boldsymbol{w_{2,1}} & 0 & w_{2,3} \\ 0 & \mathbf{0} & 0 & 0 \\ 0 & \mathbf{0} & w_{4,2} & w_{4,3} \\ w_{5,0} & \mathbf{0} & 0 & 0 \\ 0 & \mathbf{0} & 0 & w_{6,3} \\ 0 & \boldsymbol{w_{7,1}} & 0 & 0 \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ -b_2 \\ b_3 \\ -b_4 \\ b_5 \\ b_6 \\ -b_7 \end{pmatrix} \overset{ReLU}{\Rightarrow} \vec{b} \begin{pmatrix} b_0 \\ b_1 \\ 0 \\ b_3 \\ 0 \\ b_5 \\ b_6 \\ 0 \end{pmatrix}$$

[Han et al. ISCA'16]

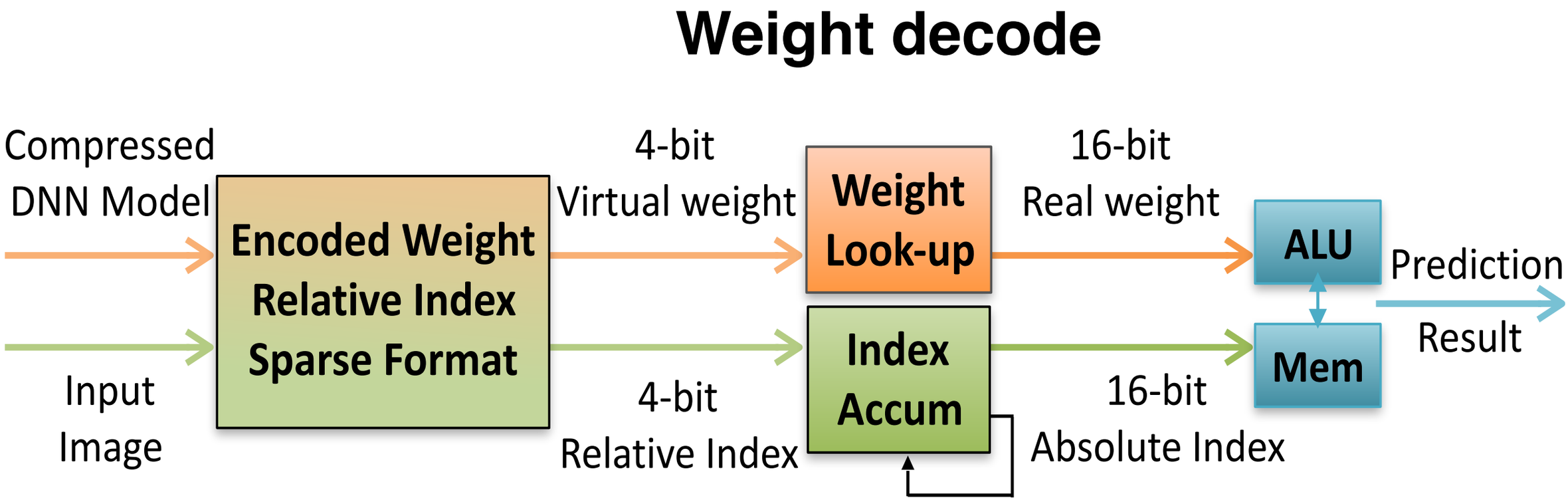
Dataflow



rule of thumb:
0 * A = 0 W * 0 = 0

[Han et al. ISCA'16]

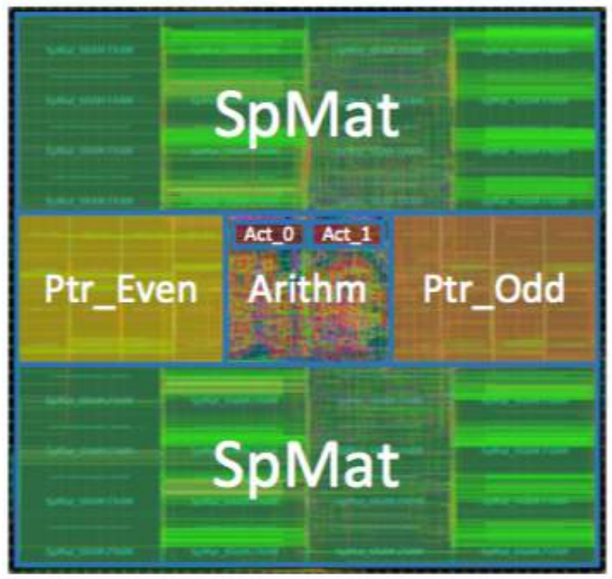
EIE Architecture



rule of thumb: 0 * A = 0 W * 0 = 0 ~~2.09, 1.92~~=> 2

[Han et al. ISCA'16]

Post Layout Result of EIE

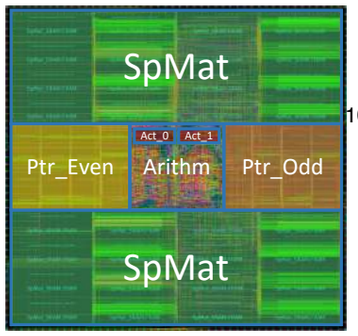


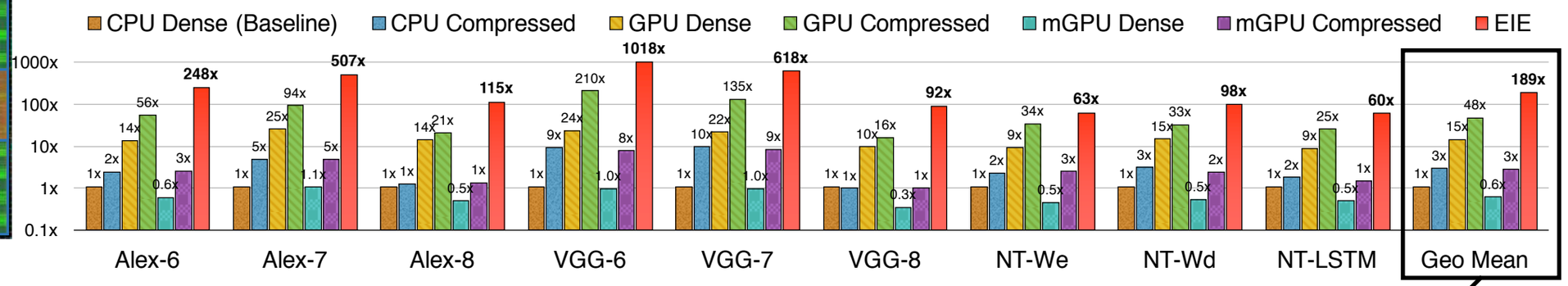
Technology	40 nm
# PEs	64
on-chip SRAM	8 MB
Max Model Size	84 Million
Static Sparsity	10x
Dynamic Sparsity	3x
Quantization	4-bit
ALU Width	16-bit
Area	40.8 mm^2
MxV Throughput	81,967 layers/s
Power	586 mW

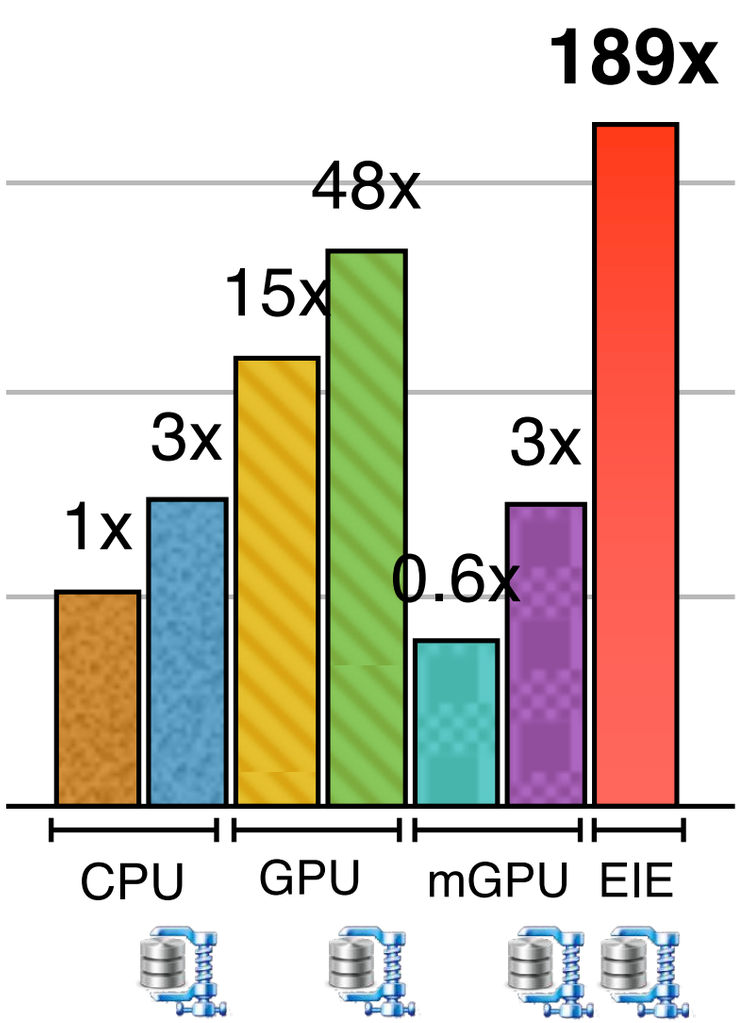
1. Post layout result
2. Throughput measured on AlexNet FC-7

[Han et al. ISCA'16]

Speedup on EIE



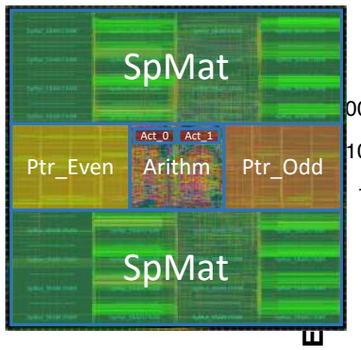


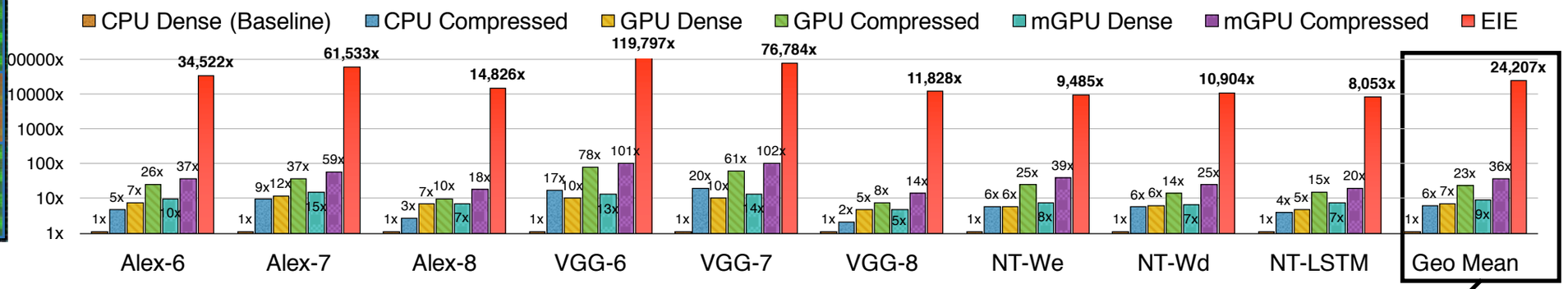


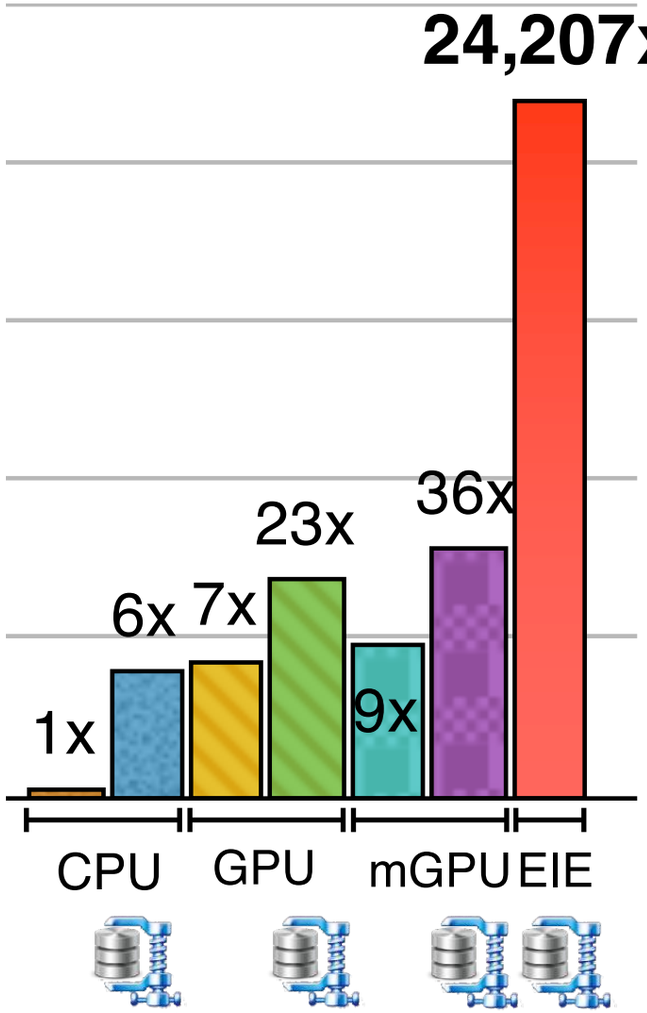
Geo Mean

[Han et al. ISCA'16]

Energy Efficiency on EIE



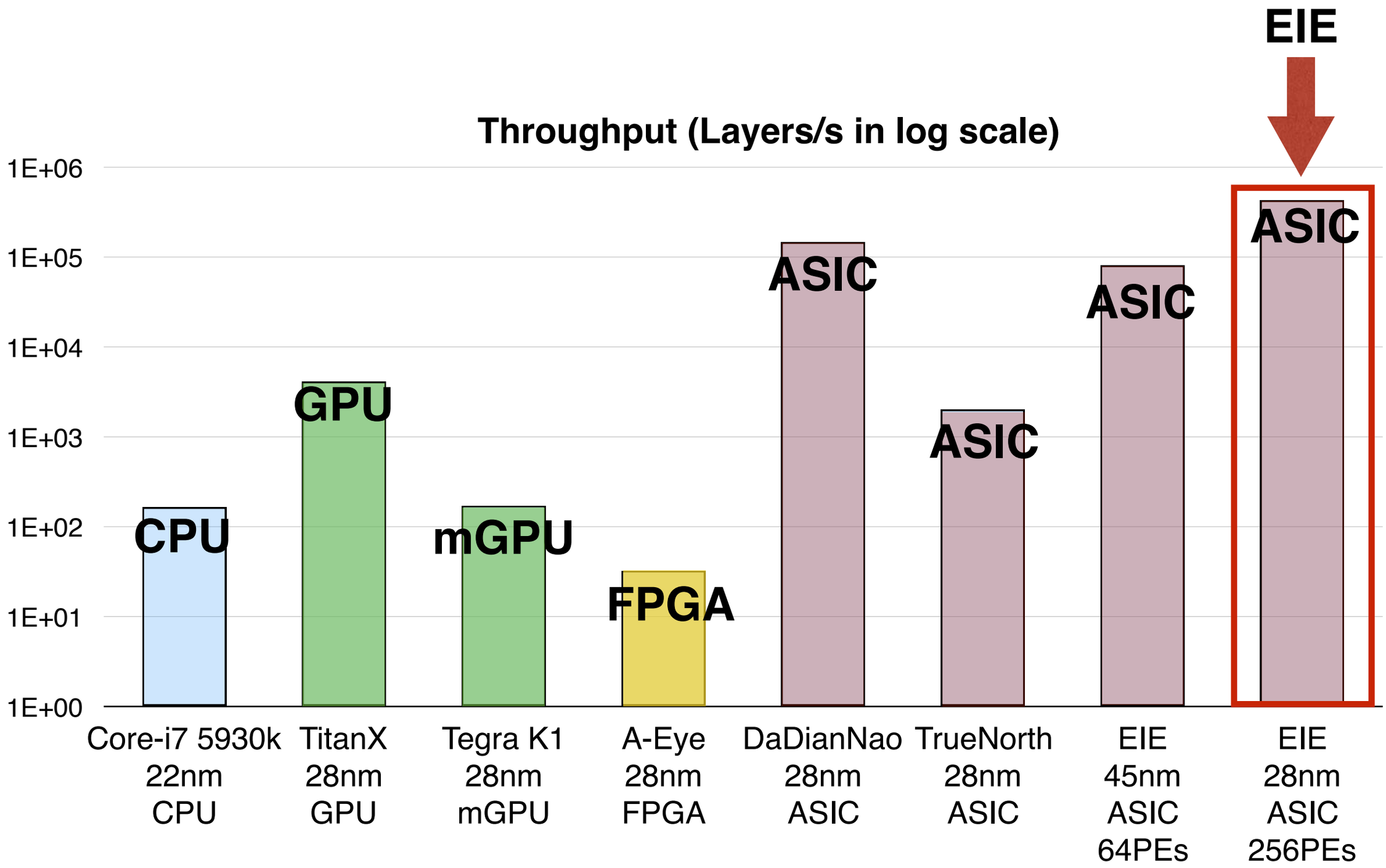




Geo Mean

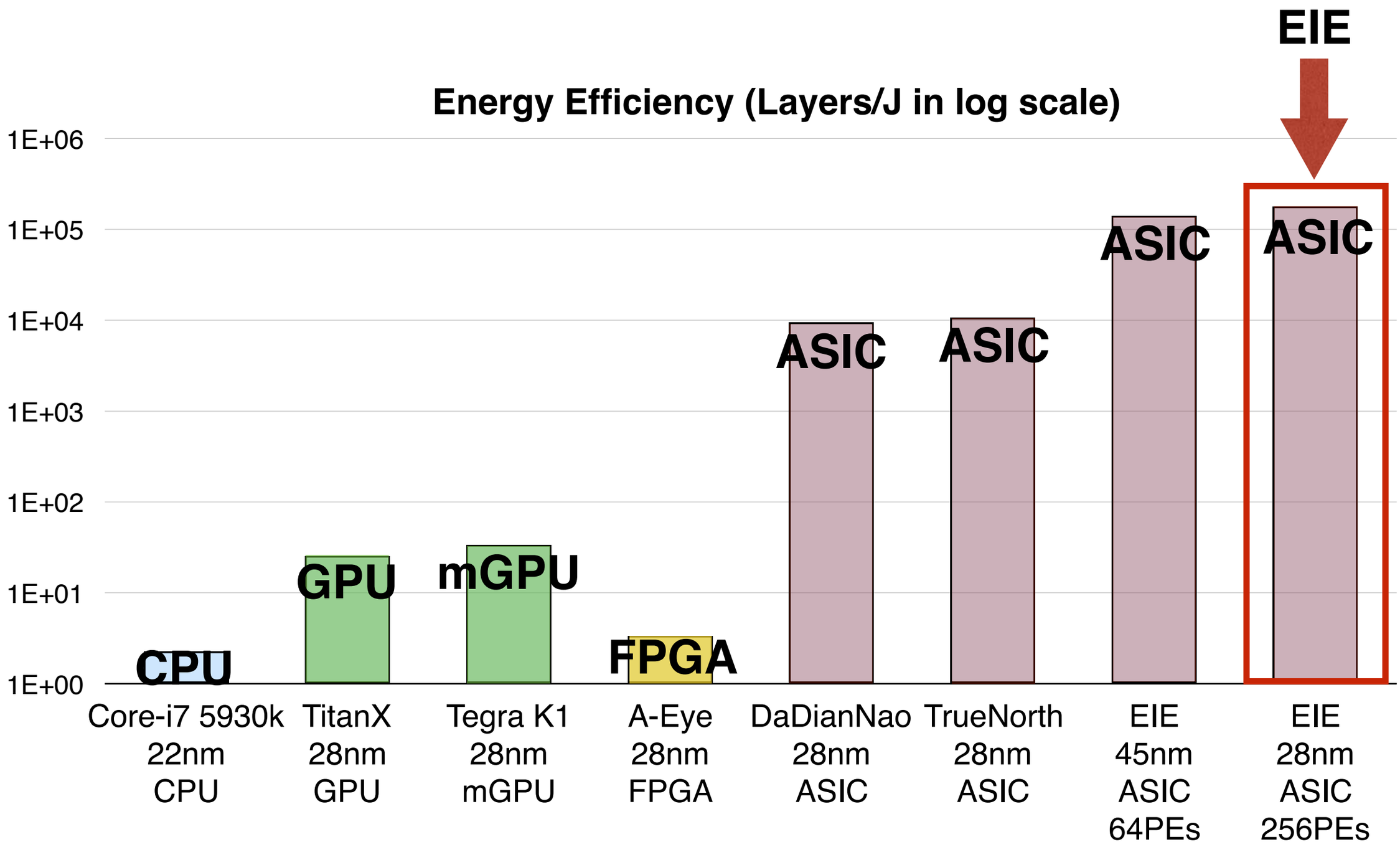
[Han et al. ISCA'16]

Comparison: Throughput



[Han et al. ISCA'16]

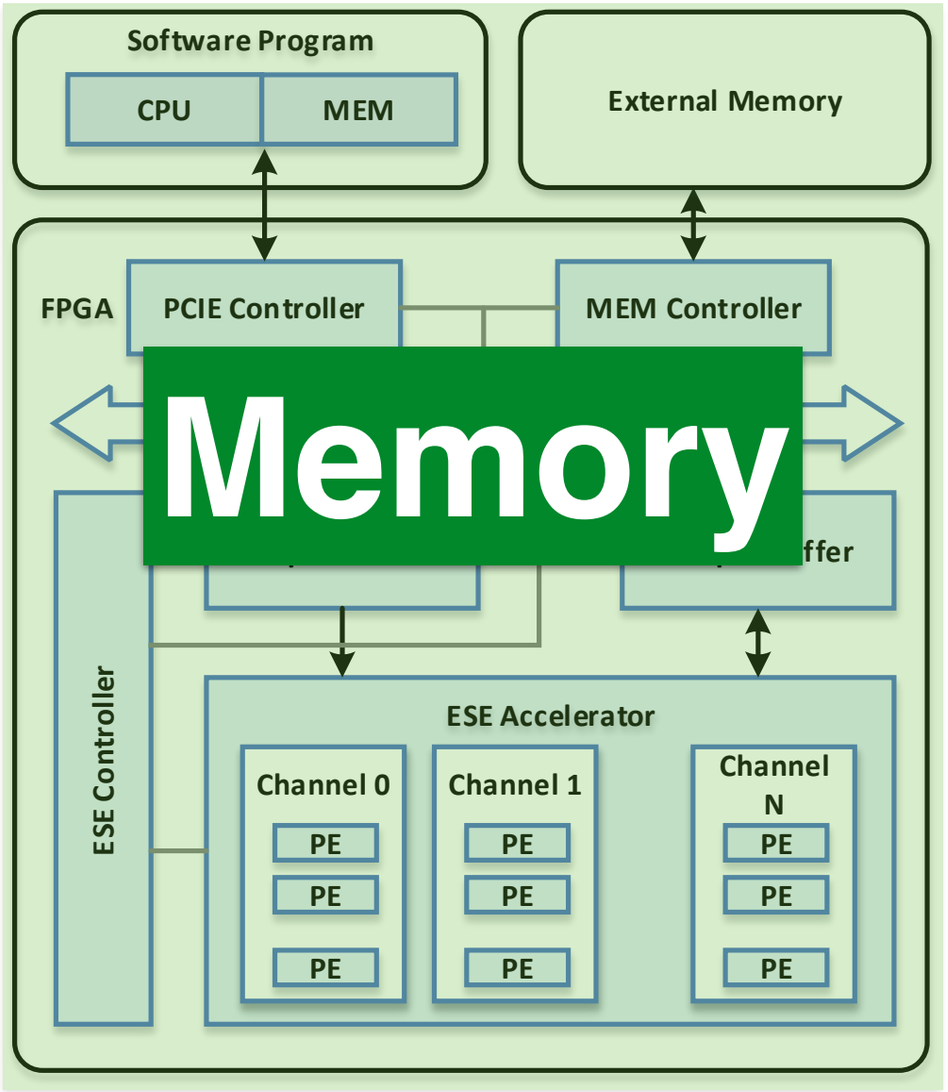
Comparison: Energy Efficiency

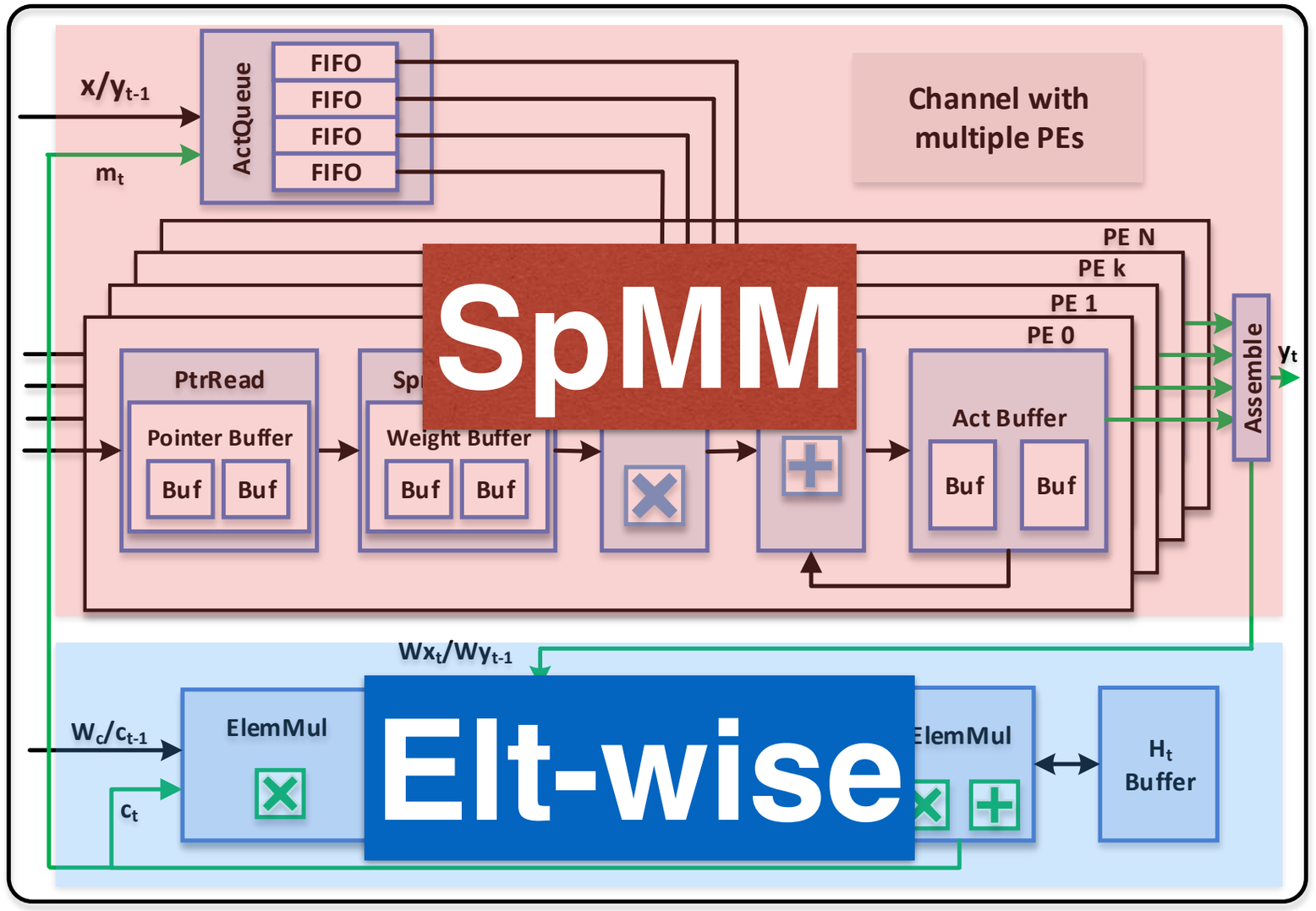


Feedforward => Recurrent neural network?

[Han et al. FPGA'17]

ESE Architecture





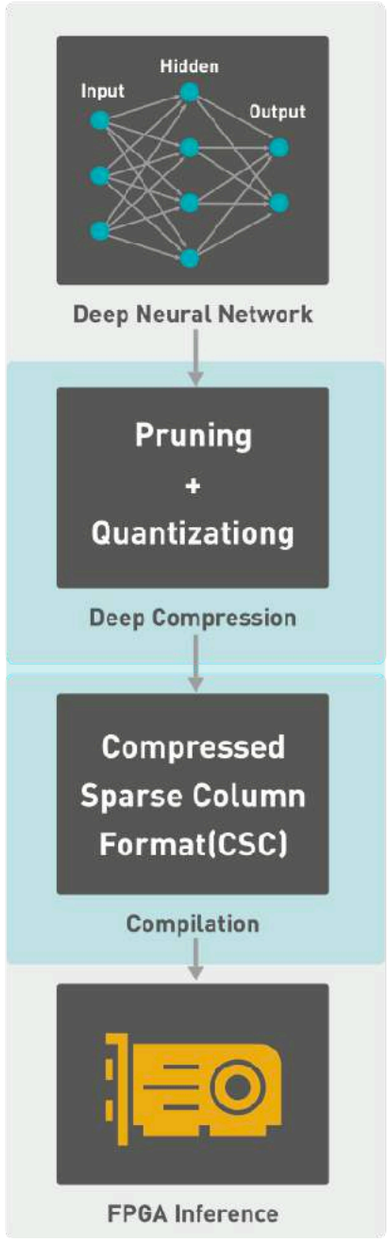
ESE is now on AWS Marketplace with Xilinx VU9P FPGA

Algorithm

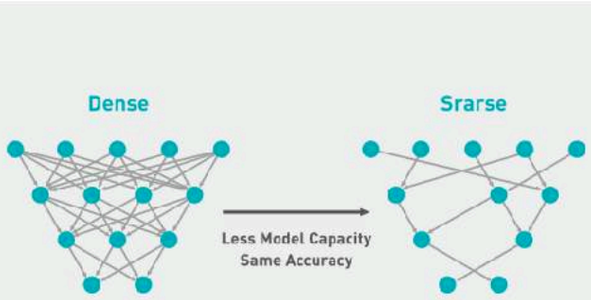
Software

Hardware

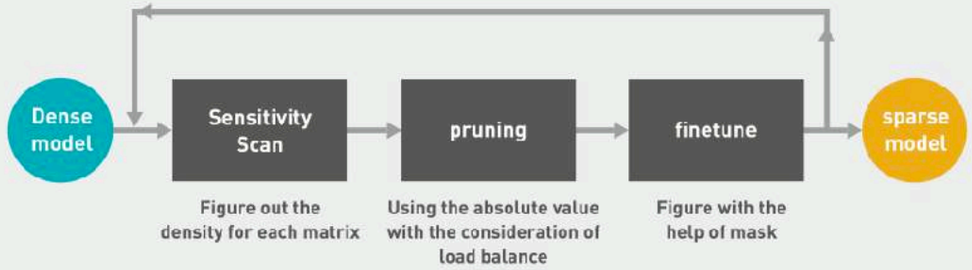
Flowchart

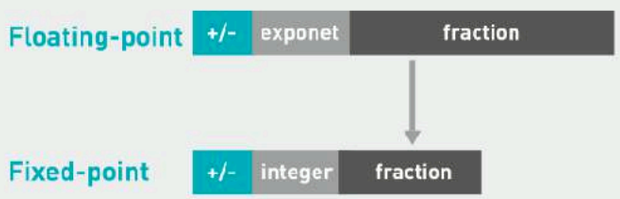


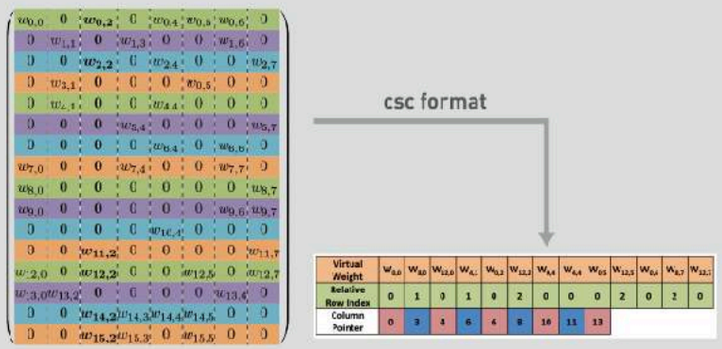
Example

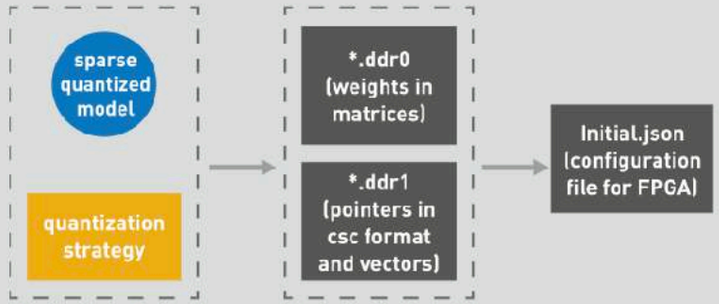


Algorithm Flow

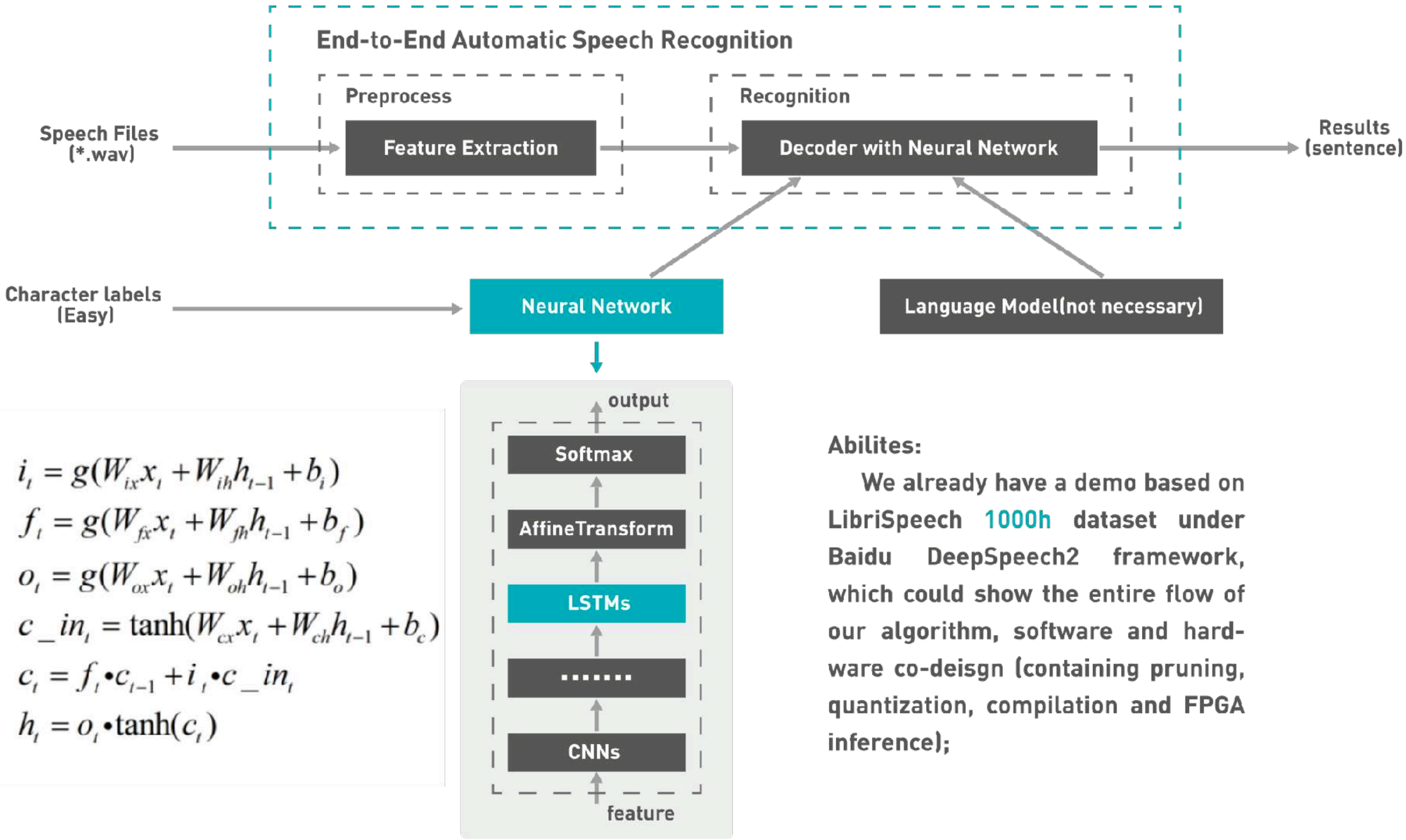








ESE is now on AWS Marketplace with Xilinx VU9P FPGA

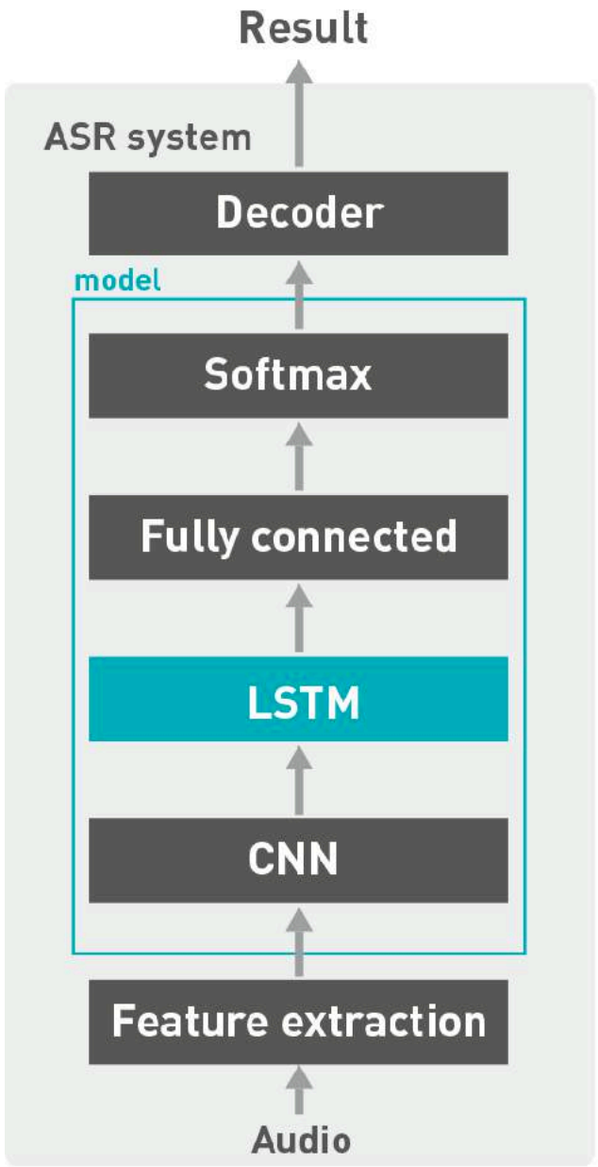


$$
\begin{aligned}
i_t &= g(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \\
f_t &= g(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \\
o_t &= g(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \\
c\_in_t &= \tanh(W_{cx}x_t + W_{ch}h_{t-1} + b_c) \\
c_t &= f_t \bullet c_{t-1} + i_t \bullet c\_in_t \\
h_t &= o_t \bullet \tanh(c_t)
\end{aligned}
$$

Abilites:

We already have a demo based on LibriSpeech 1000h dataset under Baidu DeepSpeech2 framework, which could show the entire flow of our algorithm, software and hard-ware co-deisgn (containing pruning, quantization, compilation and FPGA inference);

ESE is now on AWS Marketplace with Xilinx VU9P FPGA



	Time (ms)	Only CPU on AWS	CPU + FPGA on AWS	CPU + GPU P4 locally (cudnn)
	Decoding	0.09	0.08	0.40
1 layer only used in training	Softmax	0.07	0.08	0.08
1 layer	Fully connected	0.11	0.73	0.12
5 layers Bi-directional	**LSTM**	**118.31**	**16.97**	**38.58**
2 layers with BN and Hardtanh	CNN	14.16		1.21
	Feature extraction	2.78	2.73	1.95
	E2E time	*135.61*	***20.59*** ★	*42.35*

https://aws.amazon.com/marketplace/pp/B079N2J42R

Thank you!

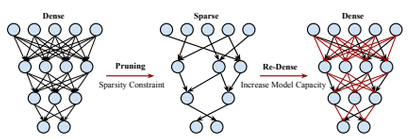
songhan@mit.edu

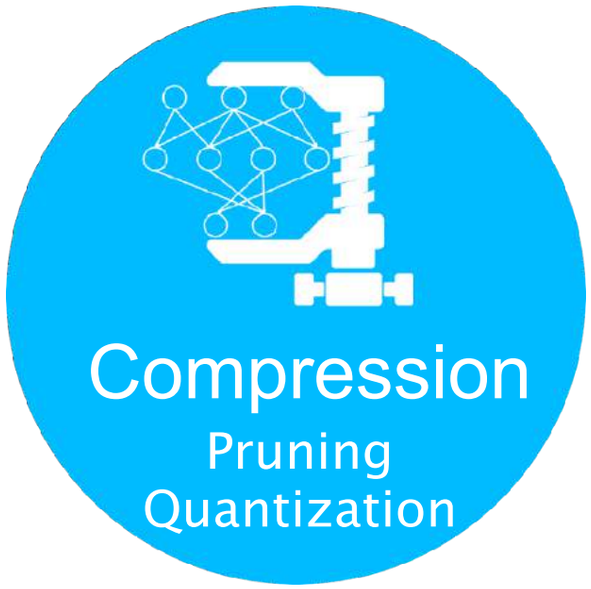
Smart

Fast

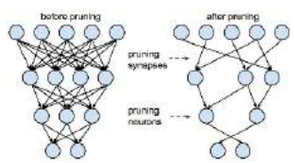
Efficient

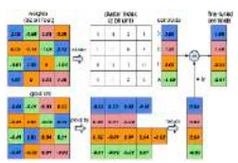
Han et al. ICLR'17





Han et al. NIPS'15
Han et al. ICLR'16
(Best paper award)





Han et al. ISCA'16
Han et al. FPGA'17
(Best paper award)