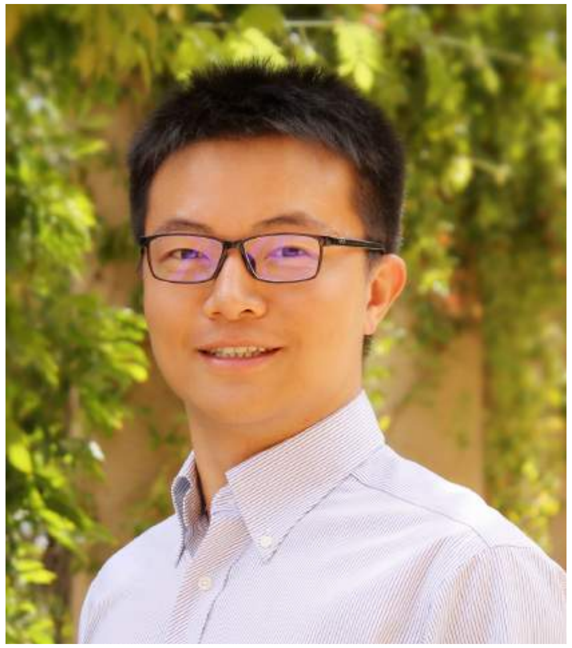
Deep Learning
Tutorial and Recent Trends

Song Han

Stanford University

Feb 22, 2017
FPGA'17, Monterey

Intro



Song Han
PhD Candidate
Stanford

Bill Dally
Chief Scientist
NVIDIA
Professor
Stanford

Deep Learning is Changing Our Lives

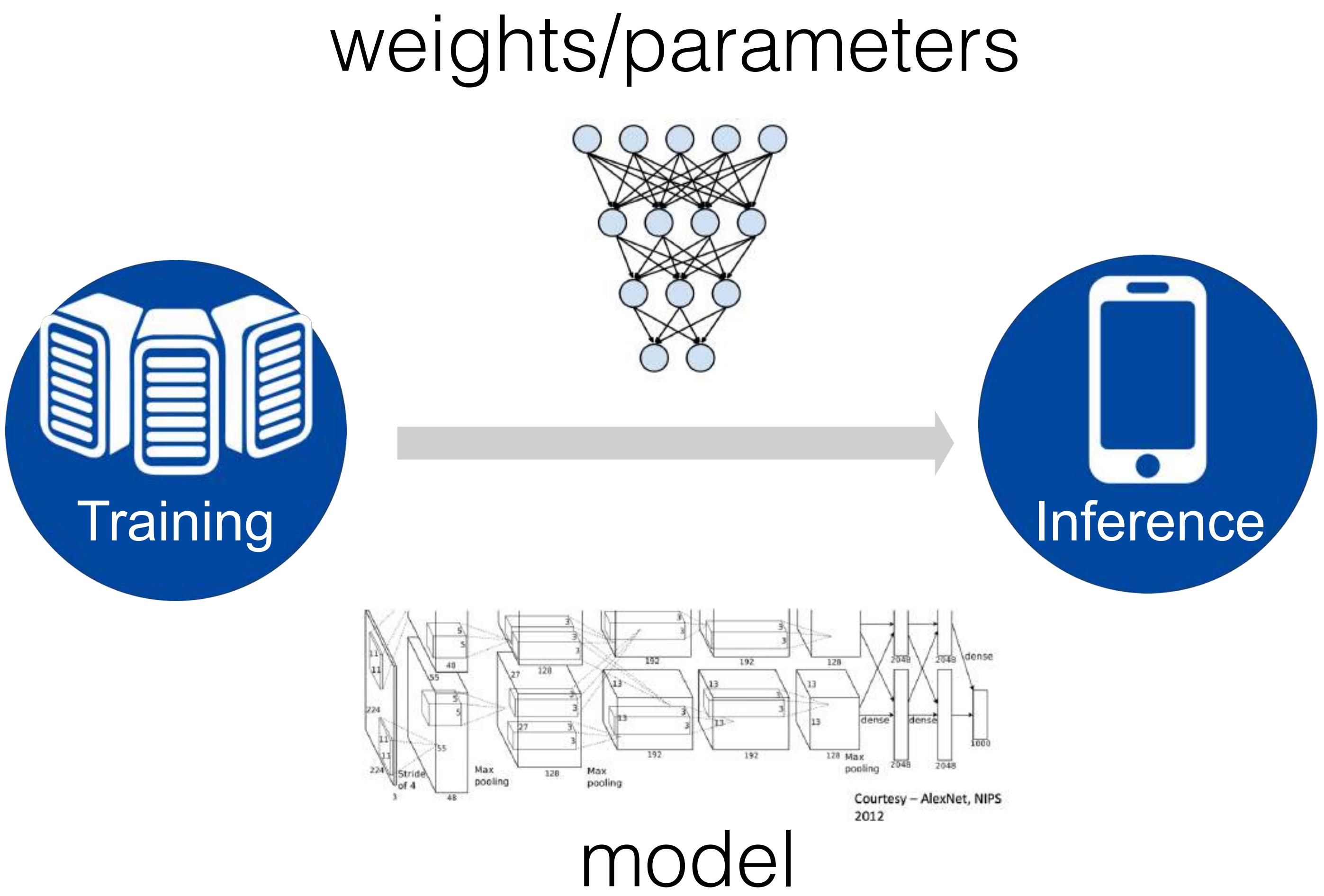
Self-Driving

Machine Translation

Play Go

Artistic Style Transfer

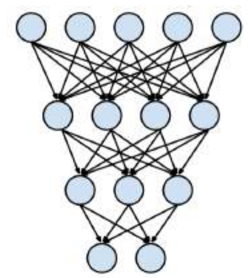
Machine Learning 101: the Setup

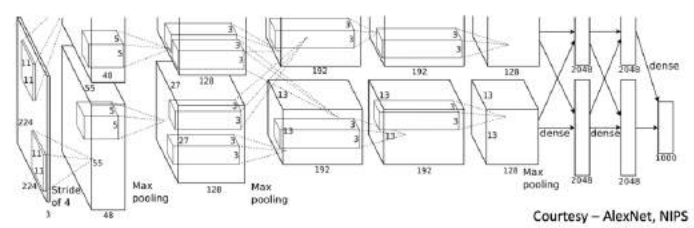


Machine Learning 101: the Setup

training dataset

weights/parameters





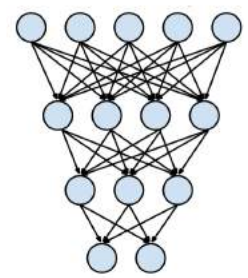
model

training hardware

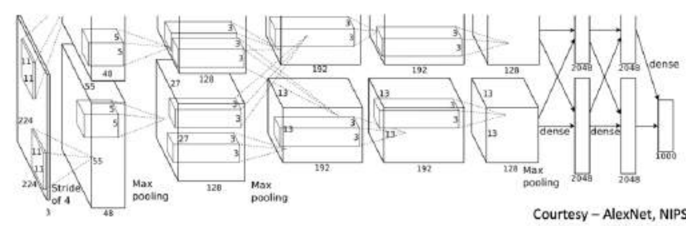
Machine Learning 101: the Setup

training dataset

weights/parameters



test image



model

training hardware

inference hardware

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

Microsoft

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

Baidu 百度

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

The Problem of Deep Learning

Computation Intensive, Memory Intensive, Difficult to Deploy

AlphaGo: 1920 CPUs and 280 GPUs,
$3000 electric bill per game

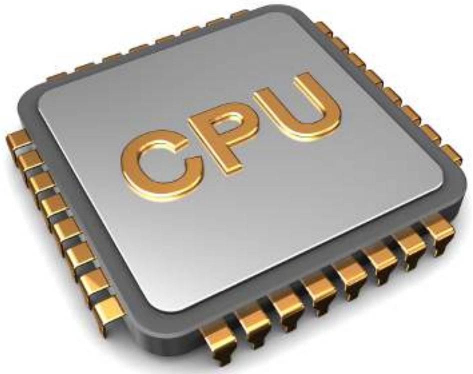
Given the power budget, Moore's law is no longer providing more computation

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

Application as Black Box

Application

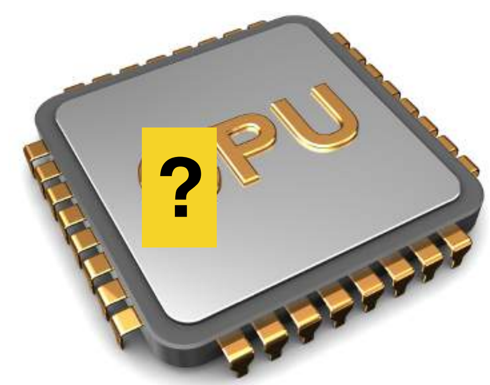
Hardware



Open the Box for HW Design

Application

Hardware



Breaks the boundary between application and architecture

Agenda

Inference

Training

Agenda

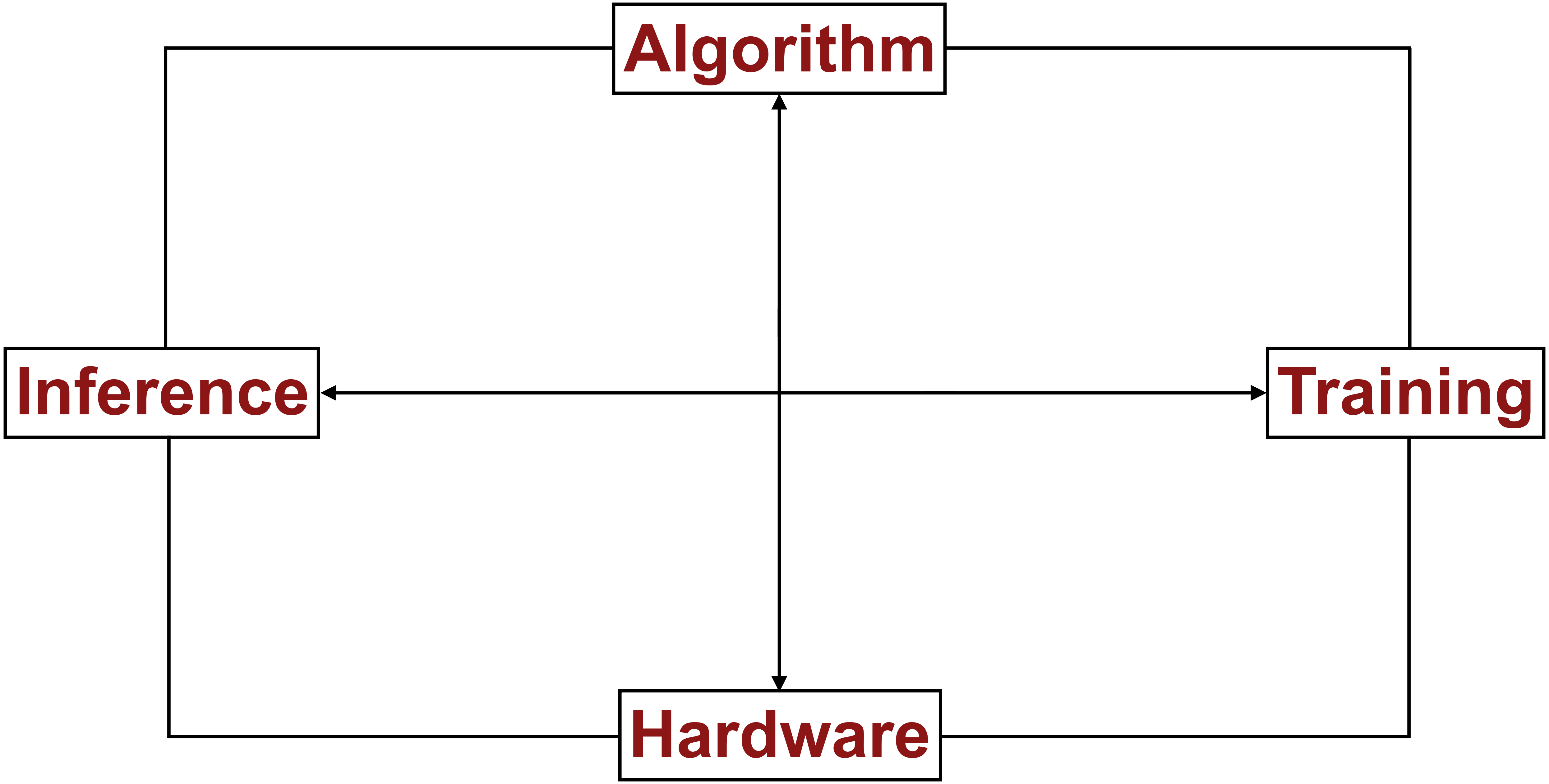
Algorithm

Inference

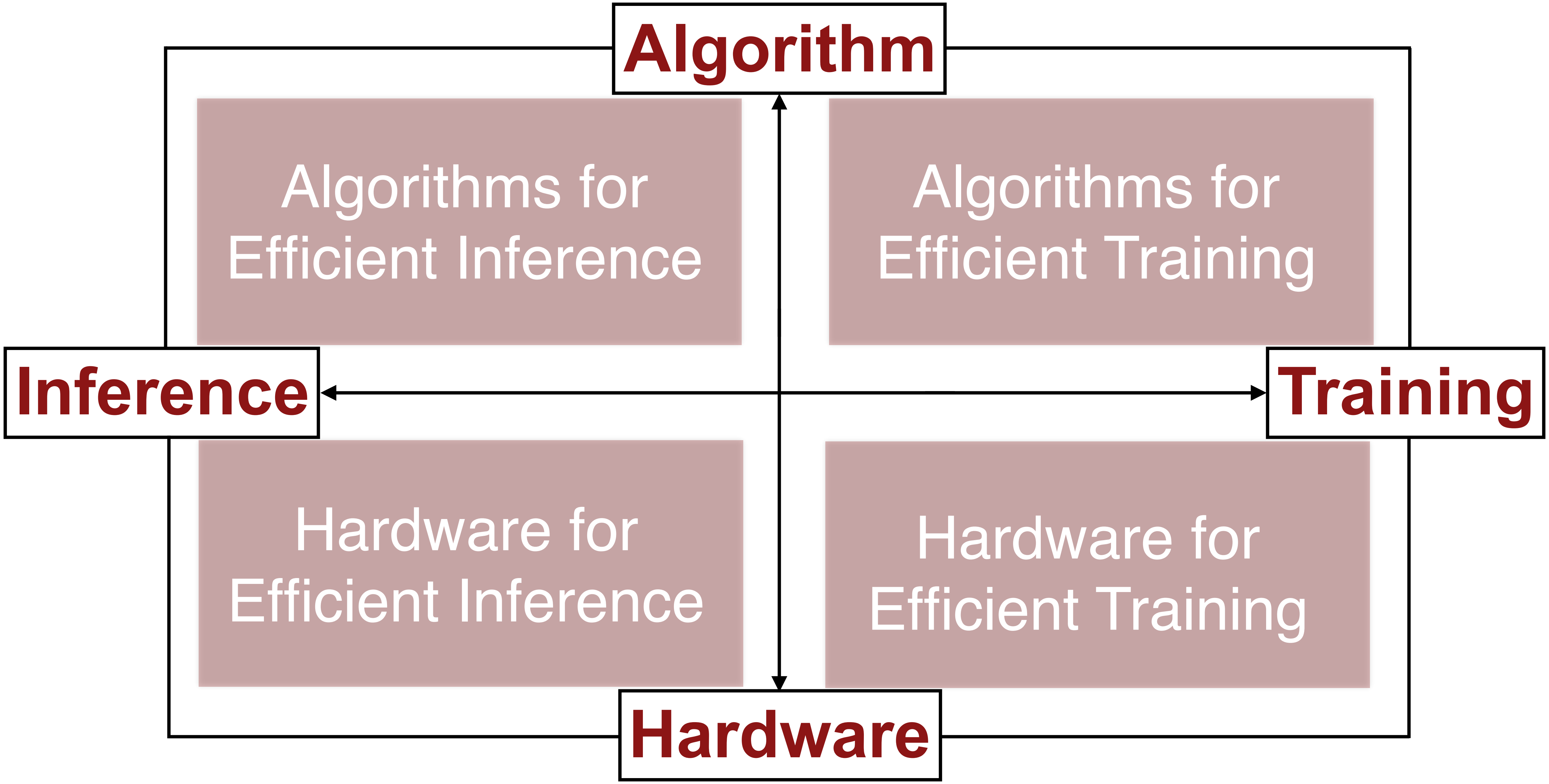
Training

Hardware

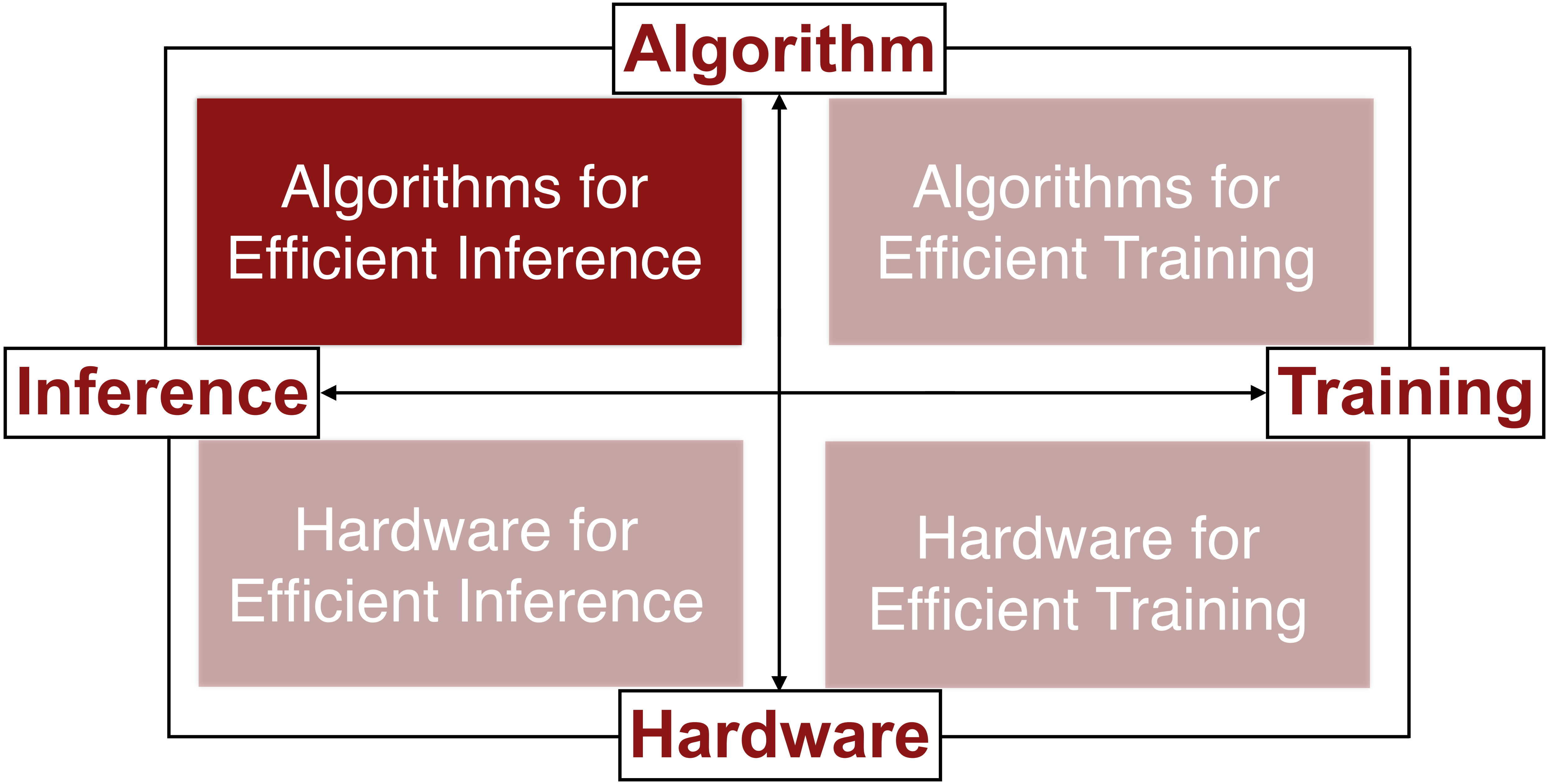
Agenda



Agenda

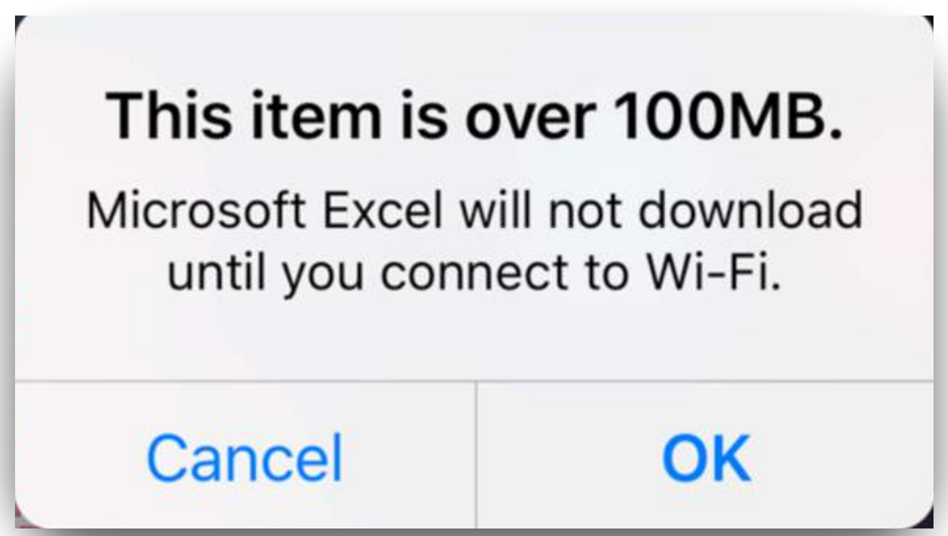

Algorithm
Algorithms for Efficient Inference
Algorithms for Efficient Training
Inference
Training
Hardware for Efficient Inference
Hardware for Efficient Training
Hardware

Agenda


Algorithm
Algorithms for Efficient Inference
Algorithms for Efficient Training
Inference
Training
Hardware for Efficient Inference
Hardware for Efficient Training
Hardware

The Problem: Large DNN Model

App developers suffers from the model size



The Problem: Large DNN Model

Hardware engineer suffers from the model size
larger model => more memory reference => more energy

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**

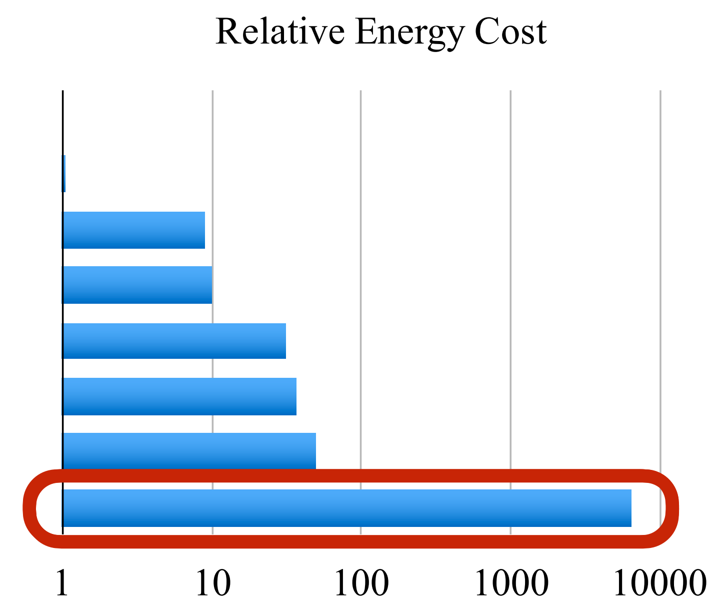
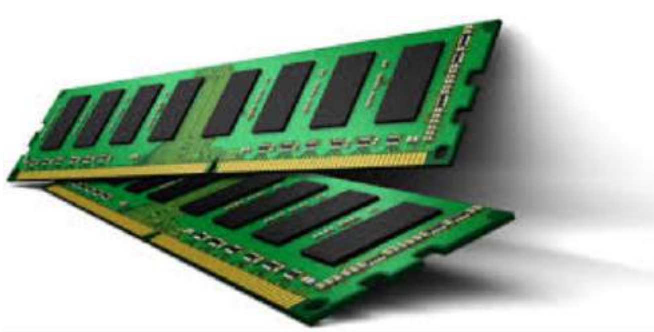
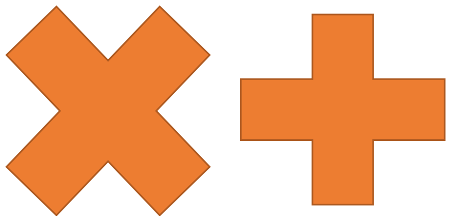


Figure 1: Energy table for 45nm CMOS process. Memory access is 2-3 orders of magnitude more energy expensive than arithmetic operations.

1  **= 100** 

The Problem of Large DNN on Mobile

Hardware engineer suffers from the model size

larger model => more memory reference => more energy

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**

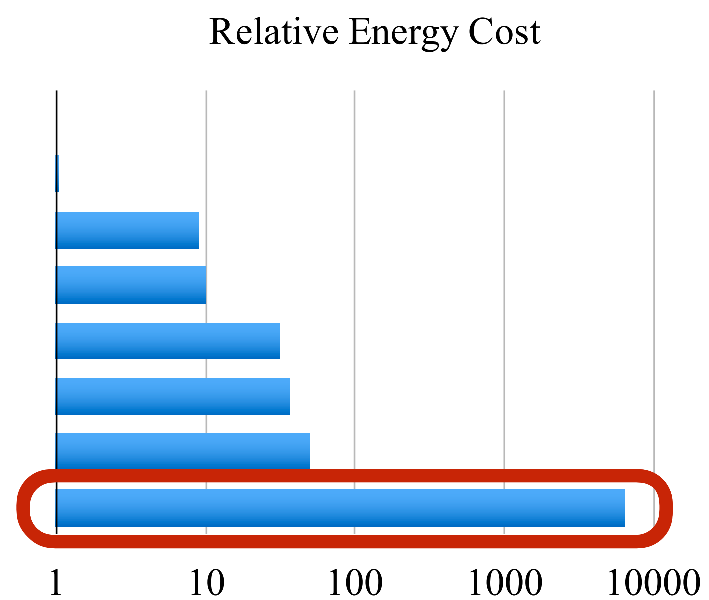


Figure 1: Energy table for 45nm CMOS process. Memory access is 2 orders of magnitude more energy expensive than arithmetic operations.

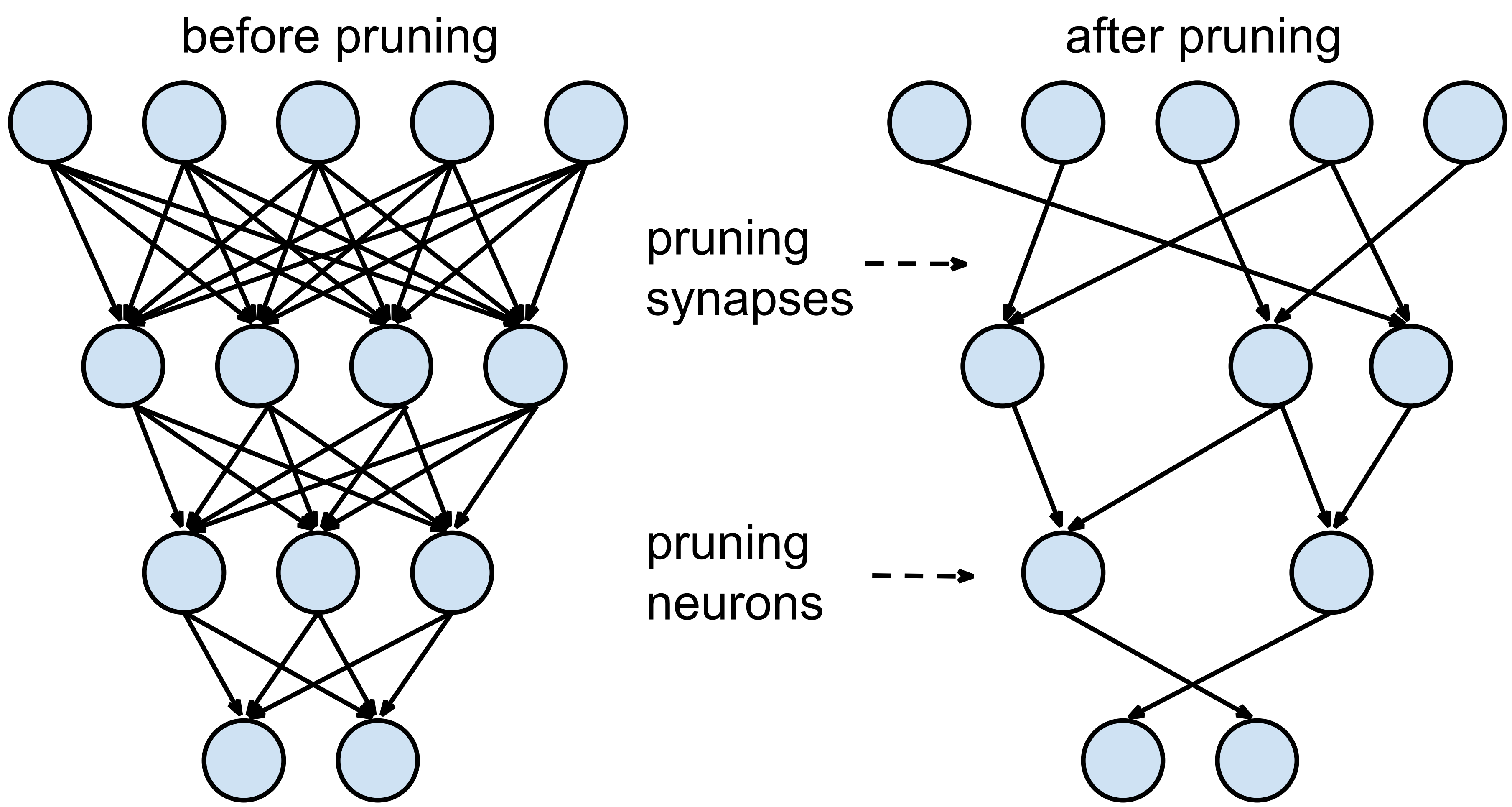
Part 1: Algorithms for Efficient Inference

- 1. Pruning
- 2. Weight Sharing
- 3. Quantization
- 4. Low Rank Approximation
- 5. Binary / Ternary Net
- 6. Winograd Transformation

Part 1: Algorithms for Efficient Inference

- 1. Pruning
- 2. Weight Sharing
- 3. Quantization
- 4. Low Rank Approximation
- 5. Binary / Ternary Net
- 6. Winograd Transformation

Pruning Neural Networks



Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS'15

Synapses in Human Brain

1000 Trillion Synapses

50 Trillion Synapses

500 Trillion Synapses

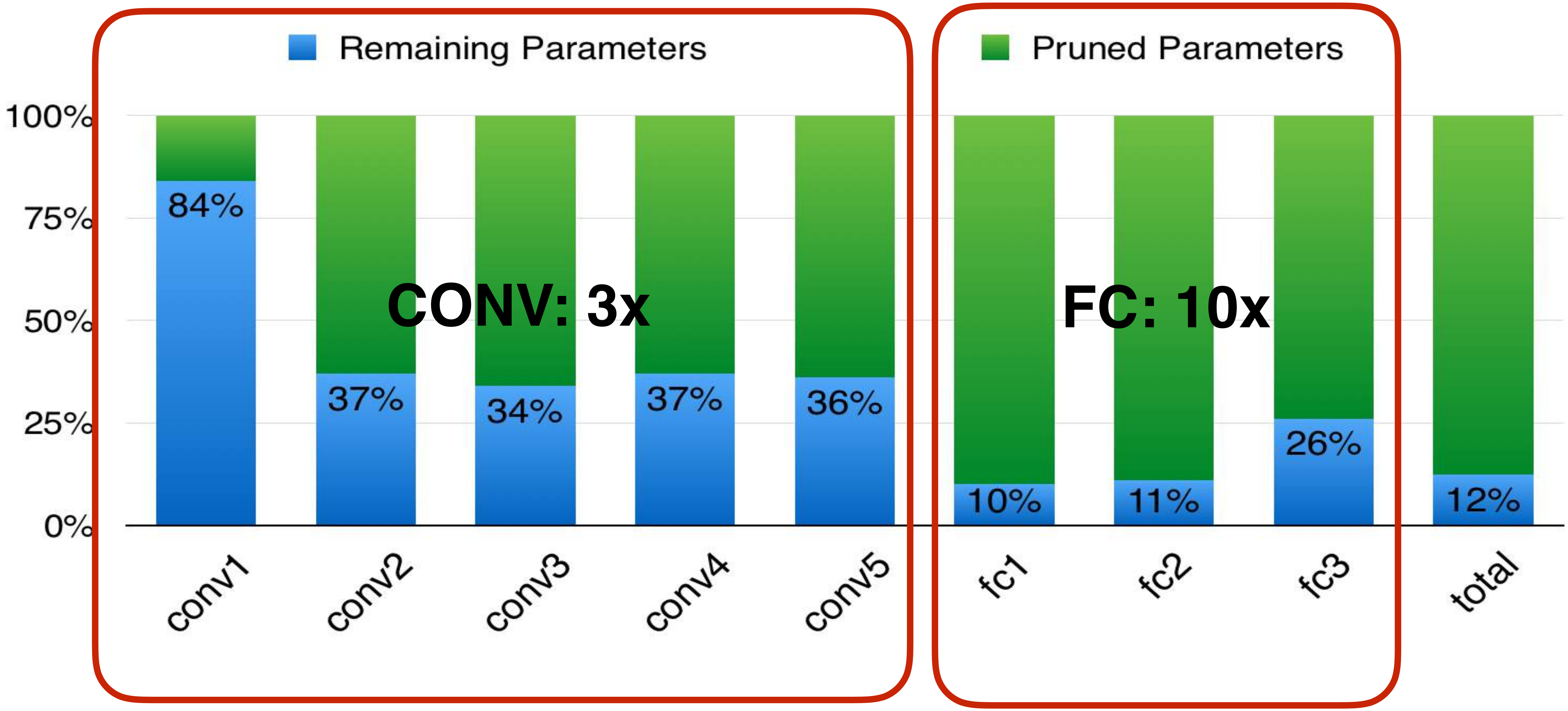
Newborn

1 year old

Adolescent

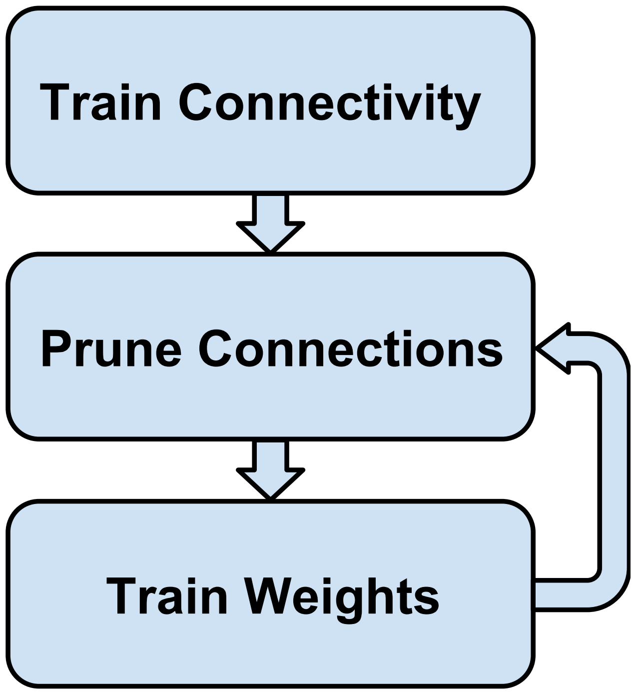
Christopher A Walsh. Peter Huttenlocher (1931-2013). Nature, 502(7470):172–172, 2013.

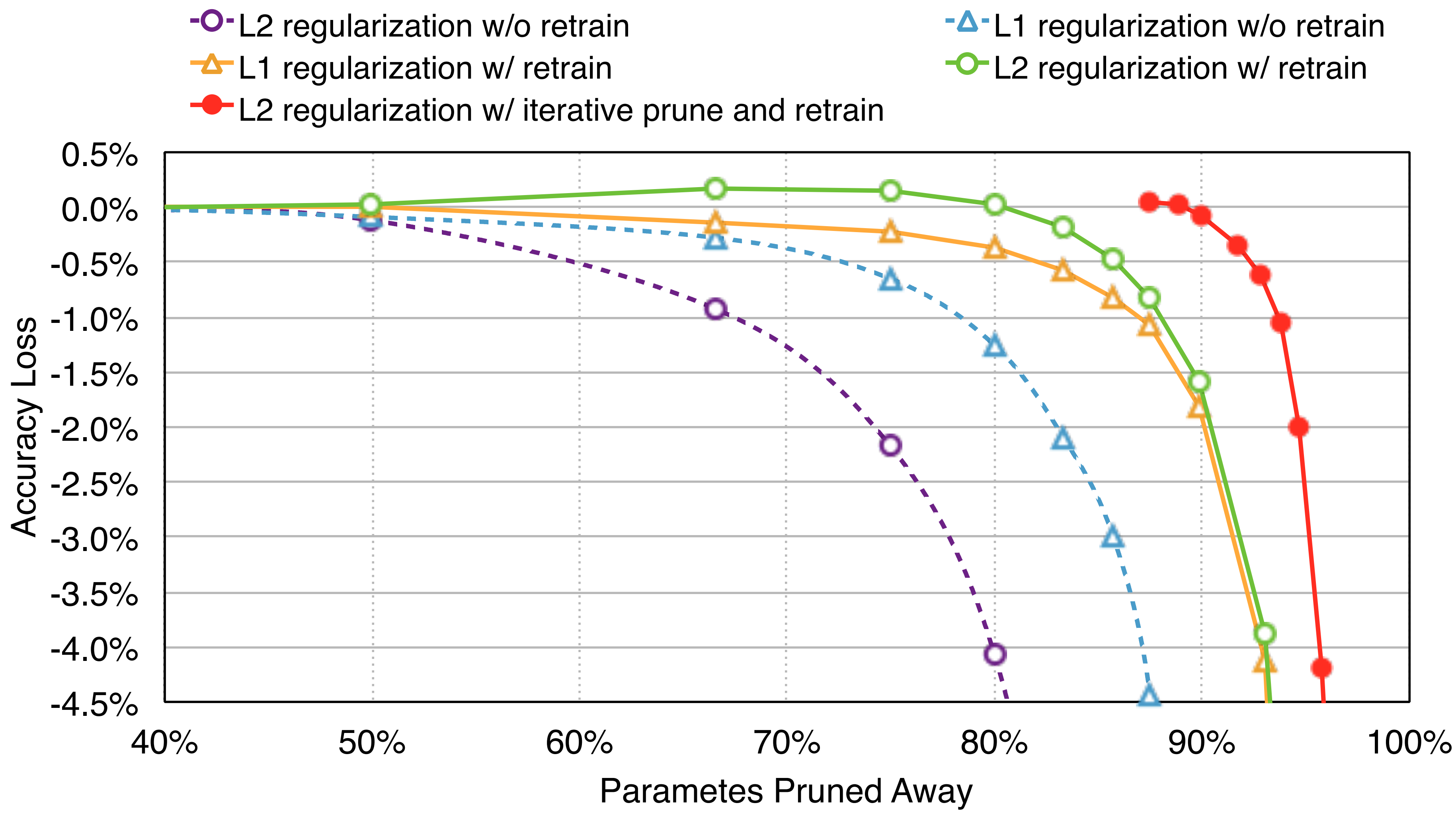
AlexNet



Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS 2015

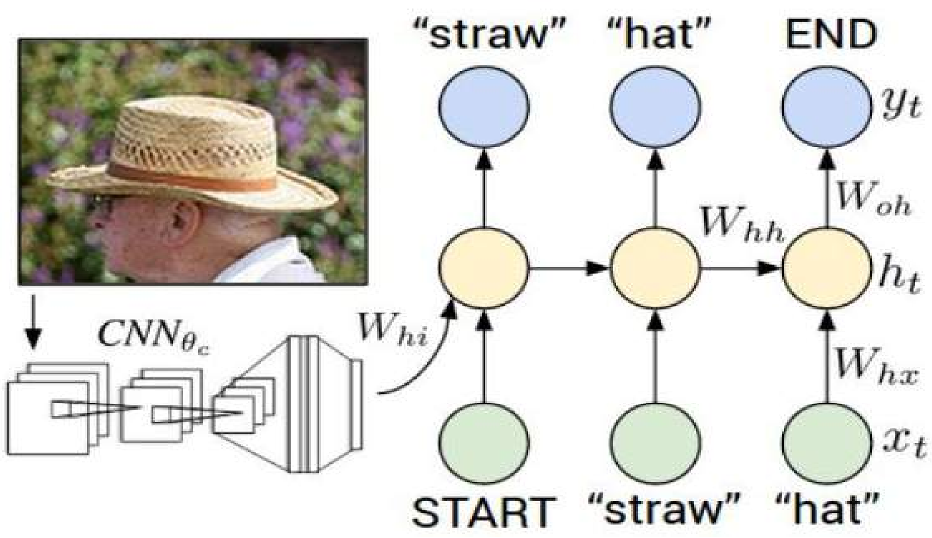
Retrain to Recover Accuracy



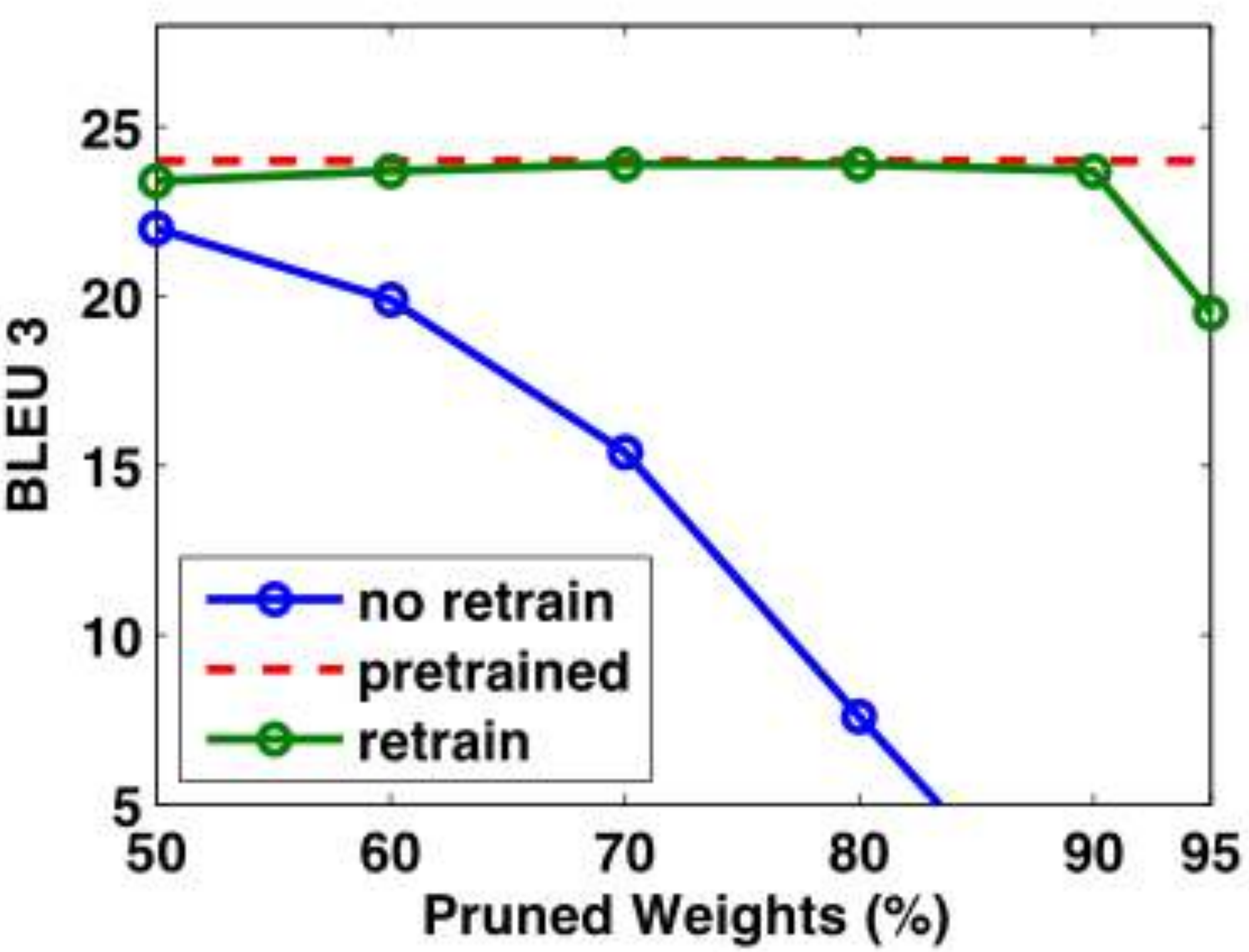


Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS 2015

Pruning RNN and LSTM



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



Pruning NeuralTalk and LSTM

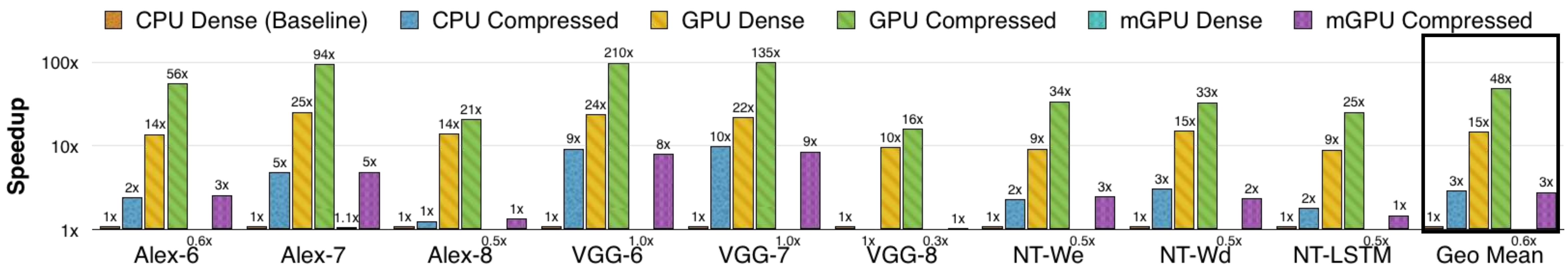
- **Original**: a basketball player in a white uniform is playing with a ball
- **Pruned 90%**: a basketball player in a white uniform is playing with a basketball

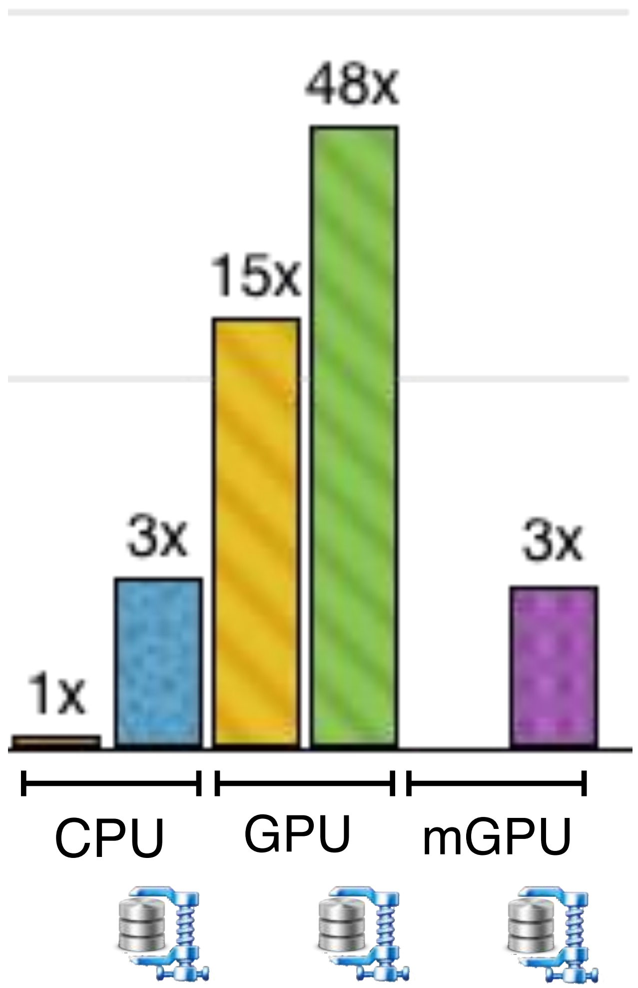
- **Original** : a brown dog is running through a grassy field
- **Pruned 90%**: a brown dog is running through a grassy area

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

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

Speedup for Pruned FC layer



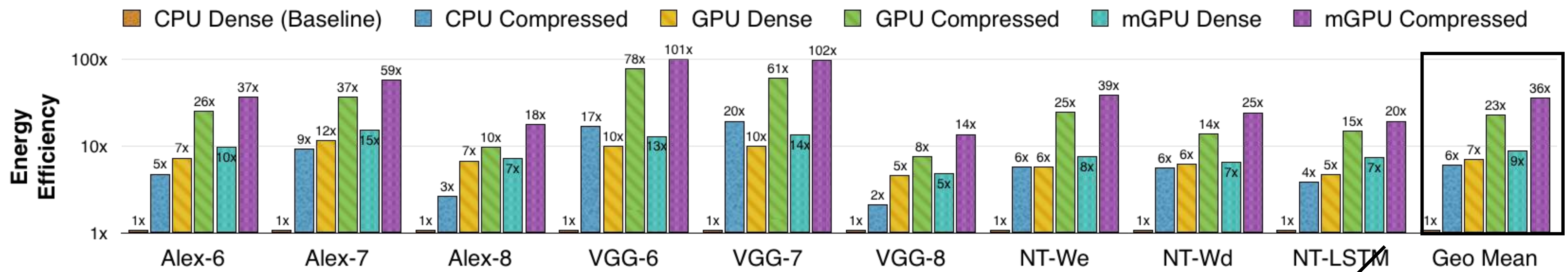


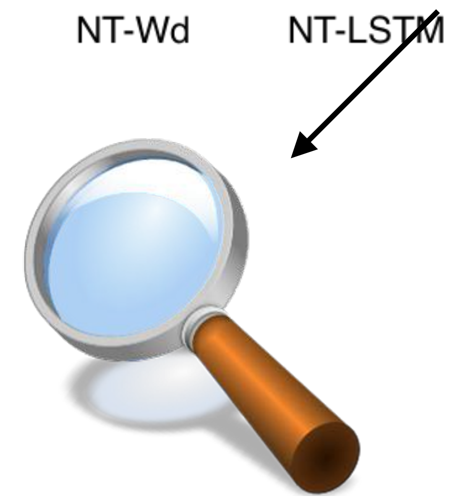
Geo Mean

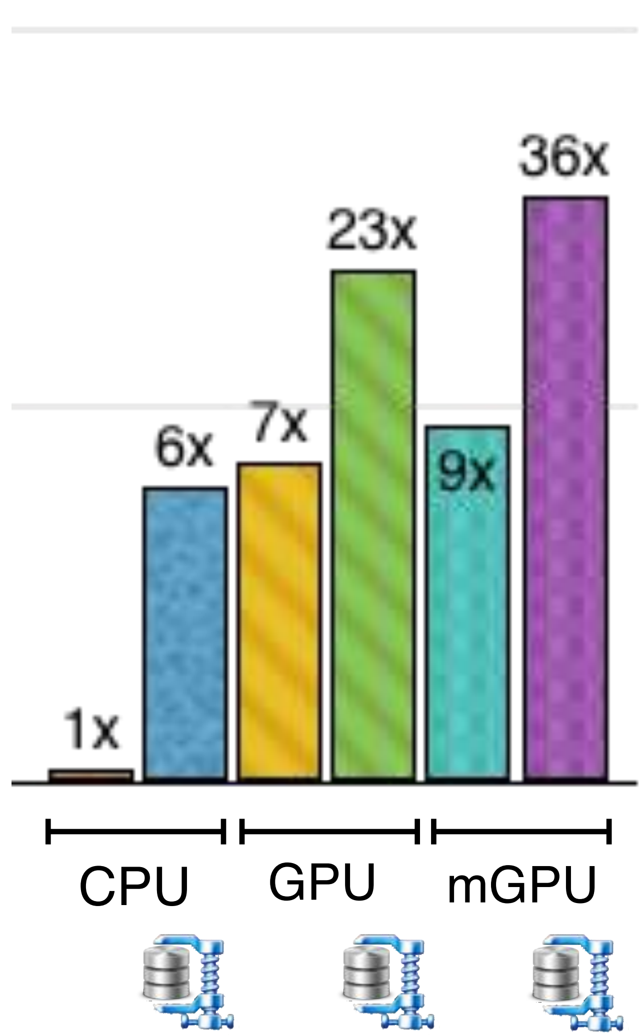
Baseline:

- Intel Core i7 5930K: MKL CBLAS GEMV, MKL SPBLAS CSRMV
- NVIDIA GeForce GTX Titan X: cuBLAS GEMV, cuSPARSE CSRMV
- NVIDIA Tegra K1: cuBLAS GEMV, cuSPARSE CSRMV

Energy Efficiency for Pruned FC layer







Geo Mean

Baseline:

- Intel Core i7 5930K: MKL CBLAS GEMV, MKL SPBLAS CSRMV
- NVIDIA GeForce GTX Titan X: cuBLAS GEMV, cuSPARSE CSRMV
- NVIDIA Tegra K1: cuBLAS GEMV, cuSPARSE CSRMV

Part 1: Algorithms for Efficient Inference

- 1. Pruning
- 2. Weight Sharing
- 3. Quantization
- 4. Low Rank Approximation
- 5. Binary / Ternary Net
- 6. Winograd Transformation

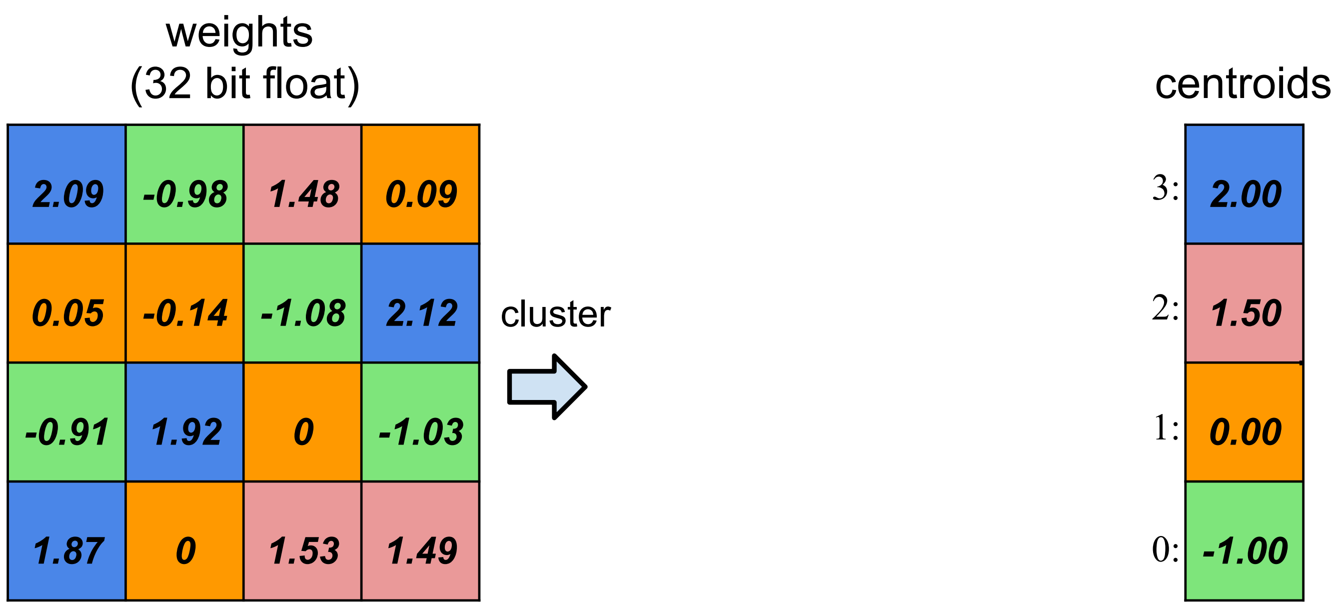
Weight Sharing

weights
(32 bit float)

2.09	-0.98	1.48	0.09
0.05	-0.14	-1.08	2.12
-0.91	1.92	0	-1.03
1.87	0	1.53	1.49

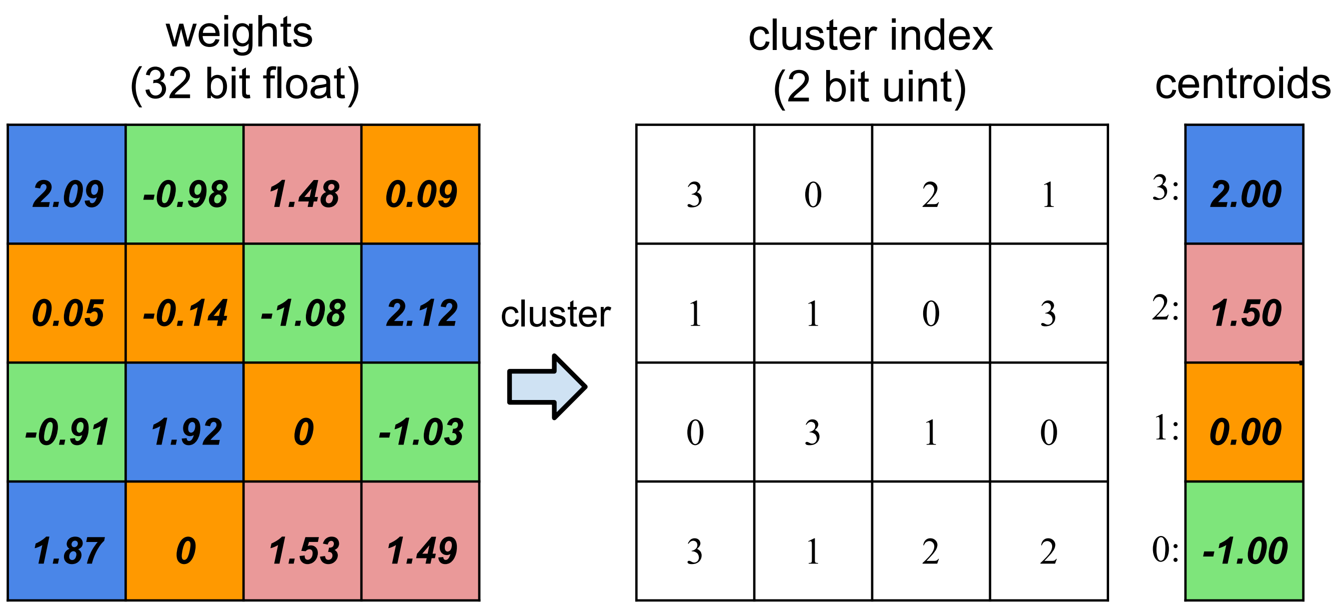
Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

Weight Sharing



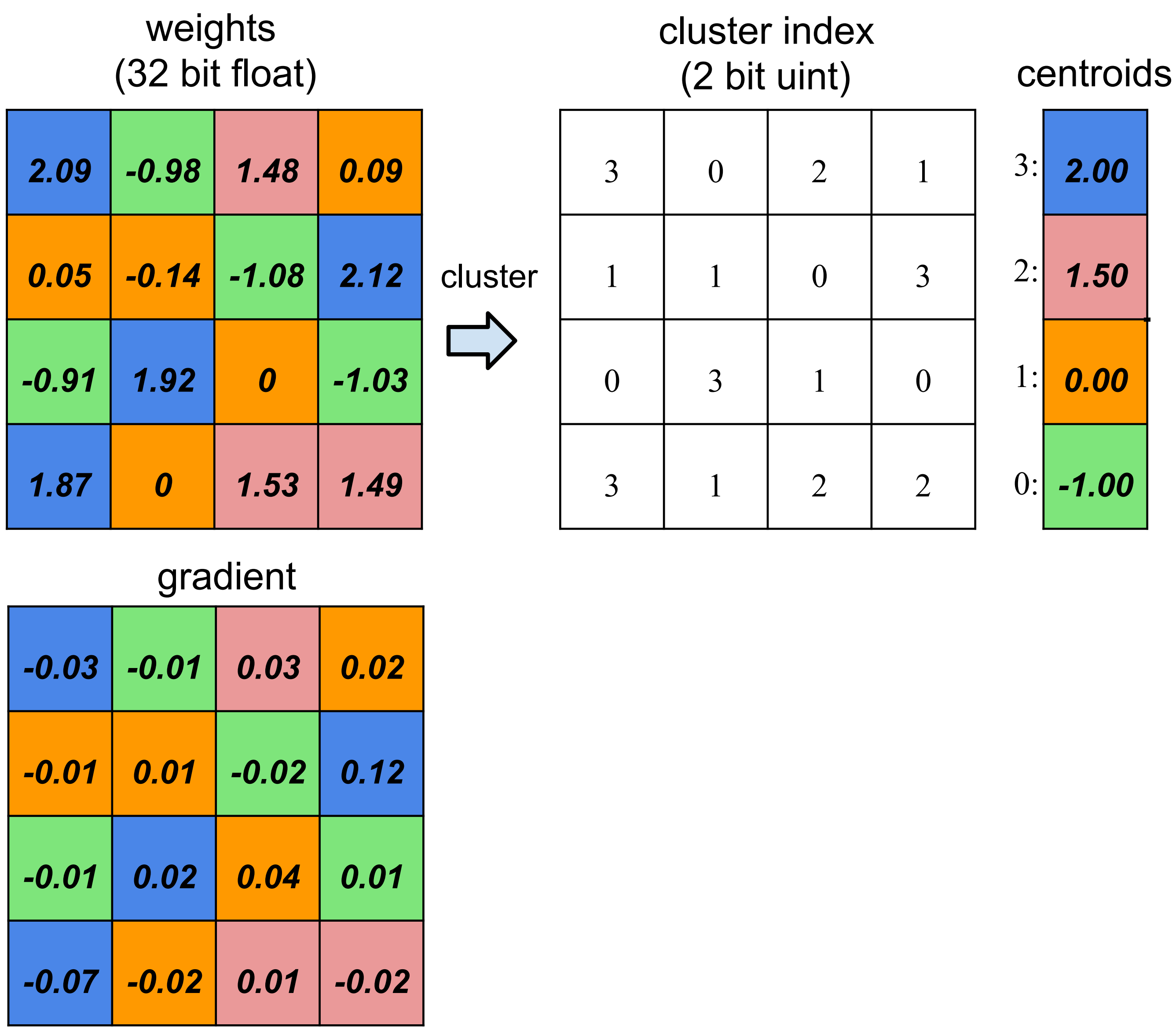
Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

Weight Sharing



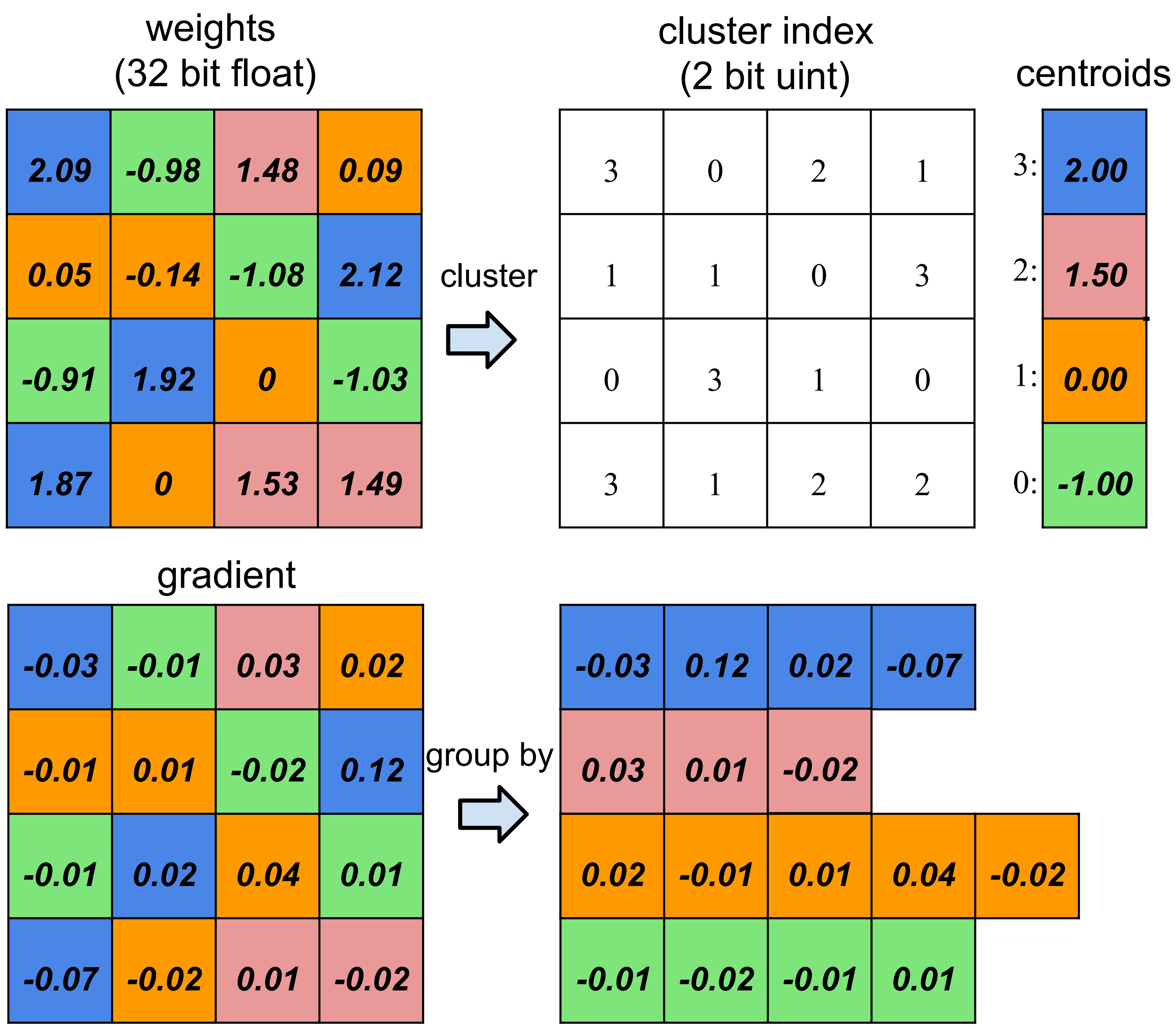
Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

Weight Sharing



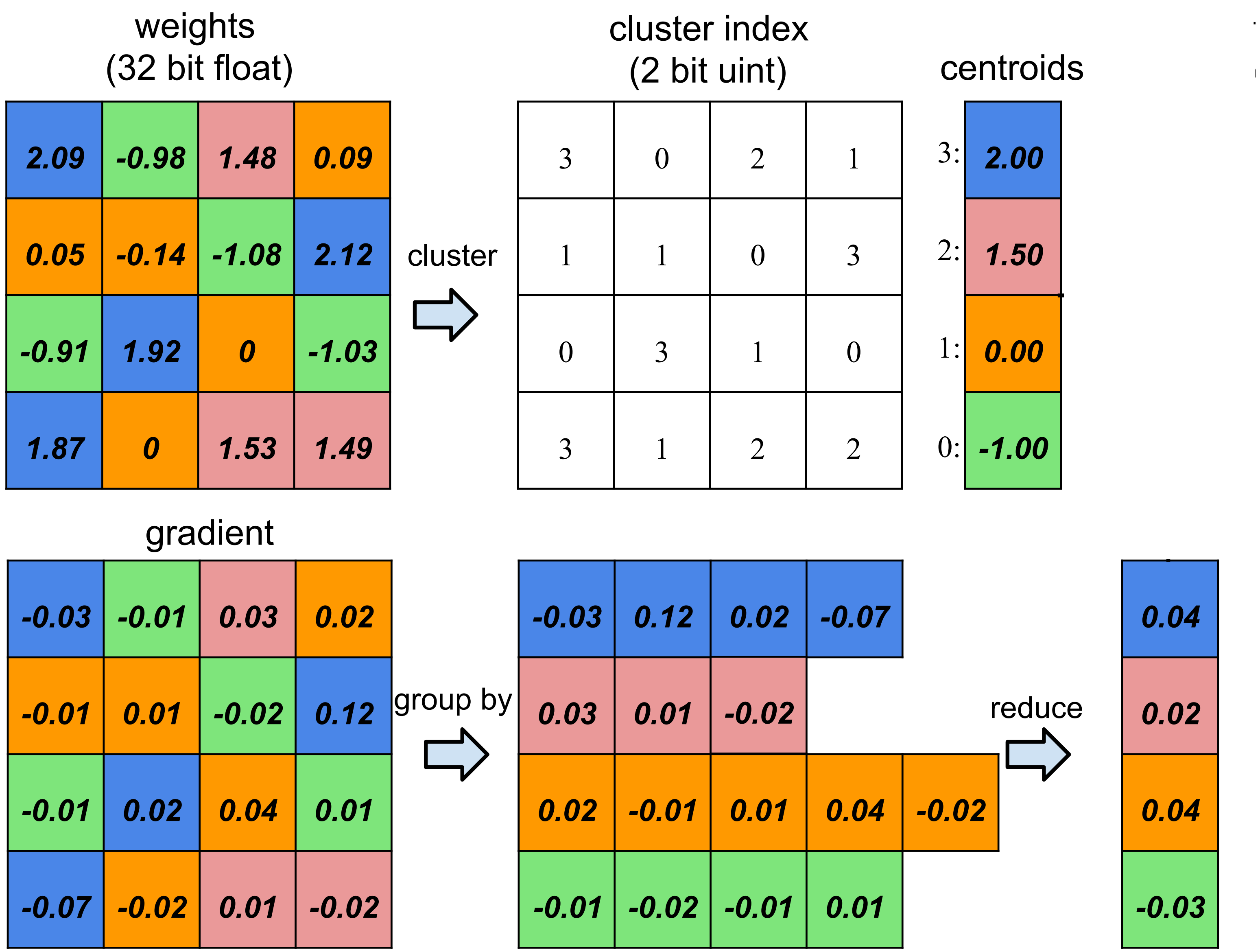
Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

Weight Sharing



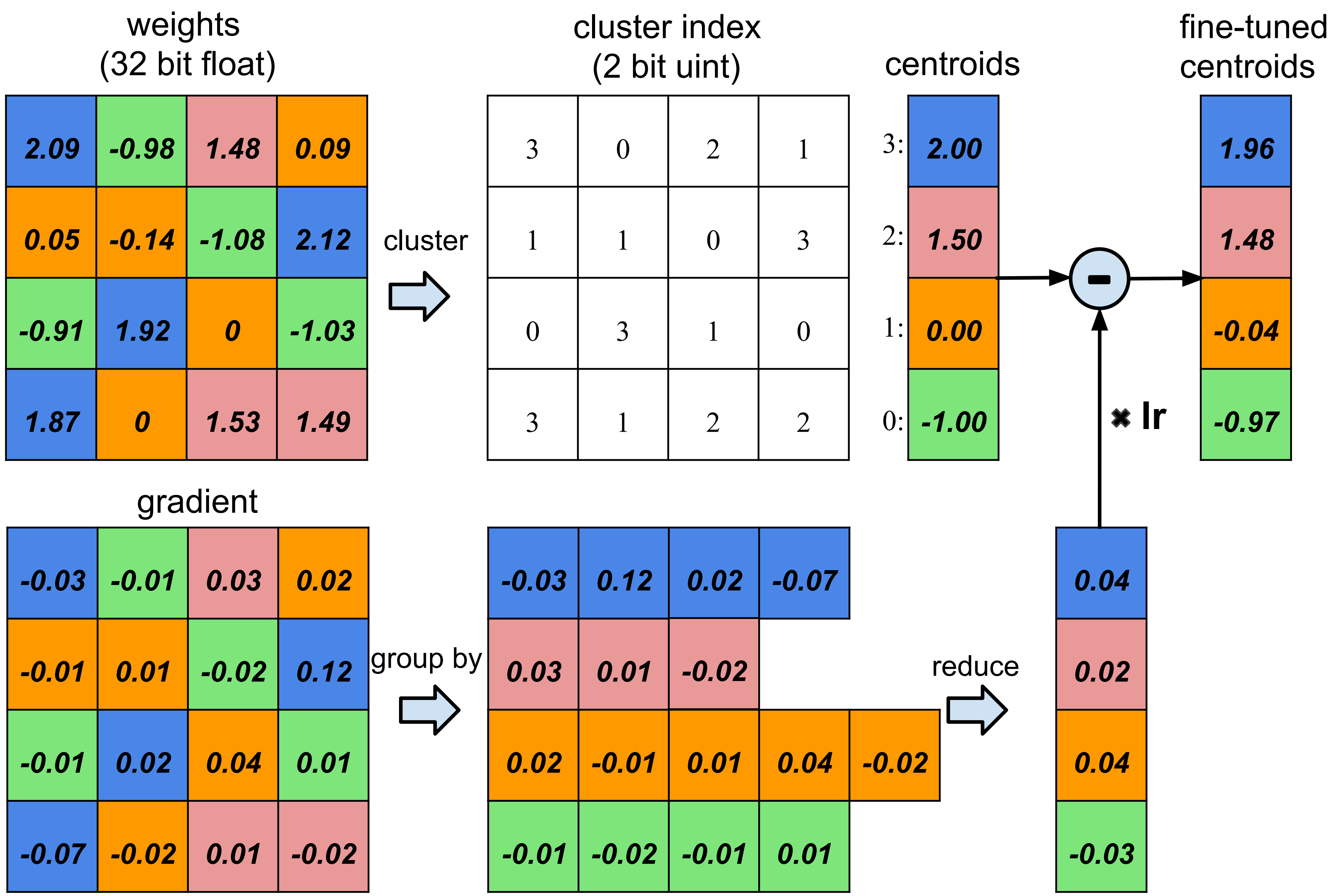
Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

Weight Sharing



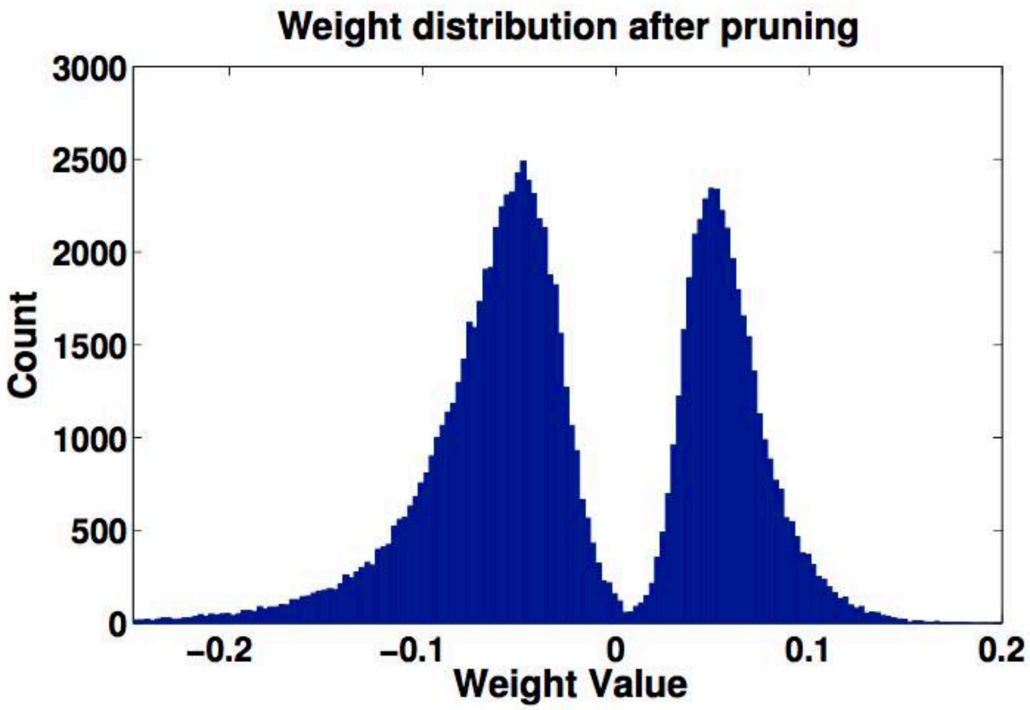
Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

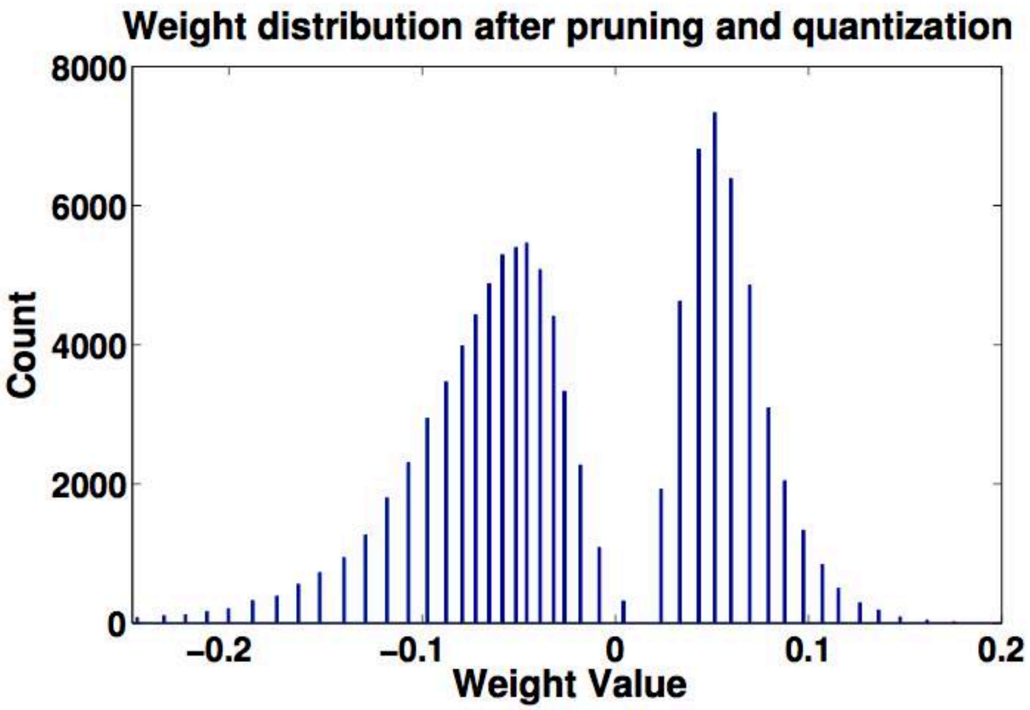
Weight Sharing



Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

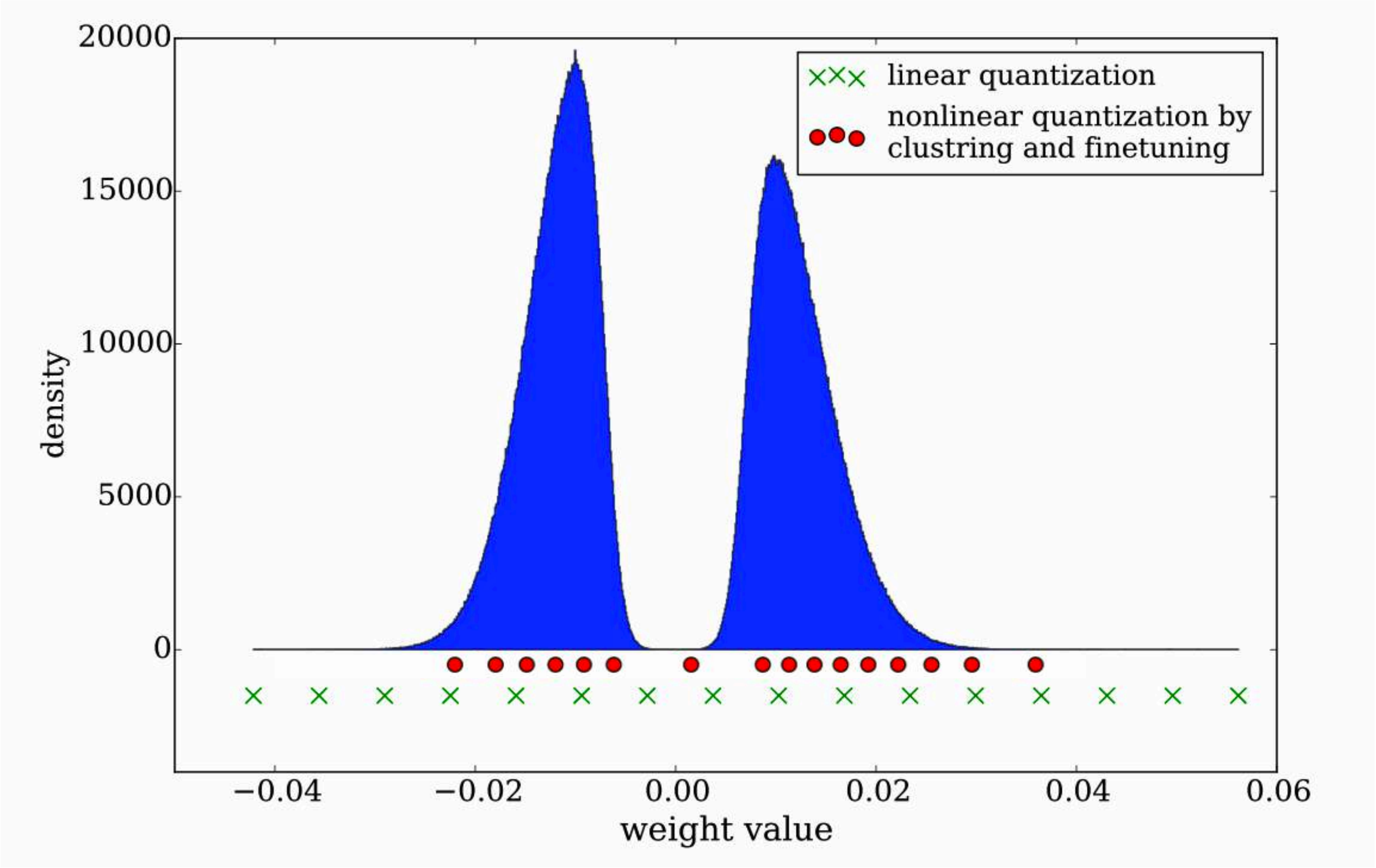
Trained Quantization Changes Weight Distribution





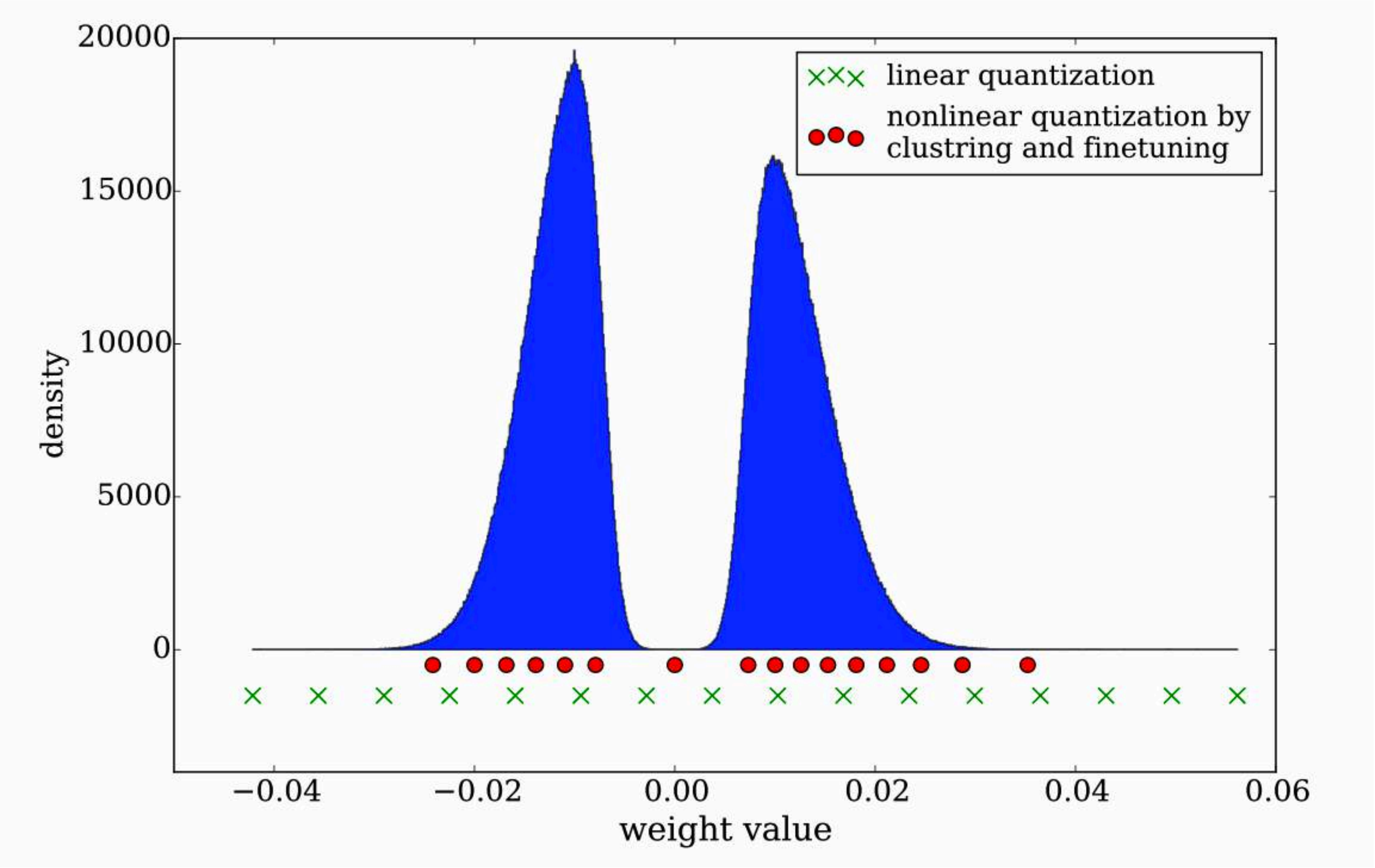
Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

Trained Quantization Changes Weight Distribution



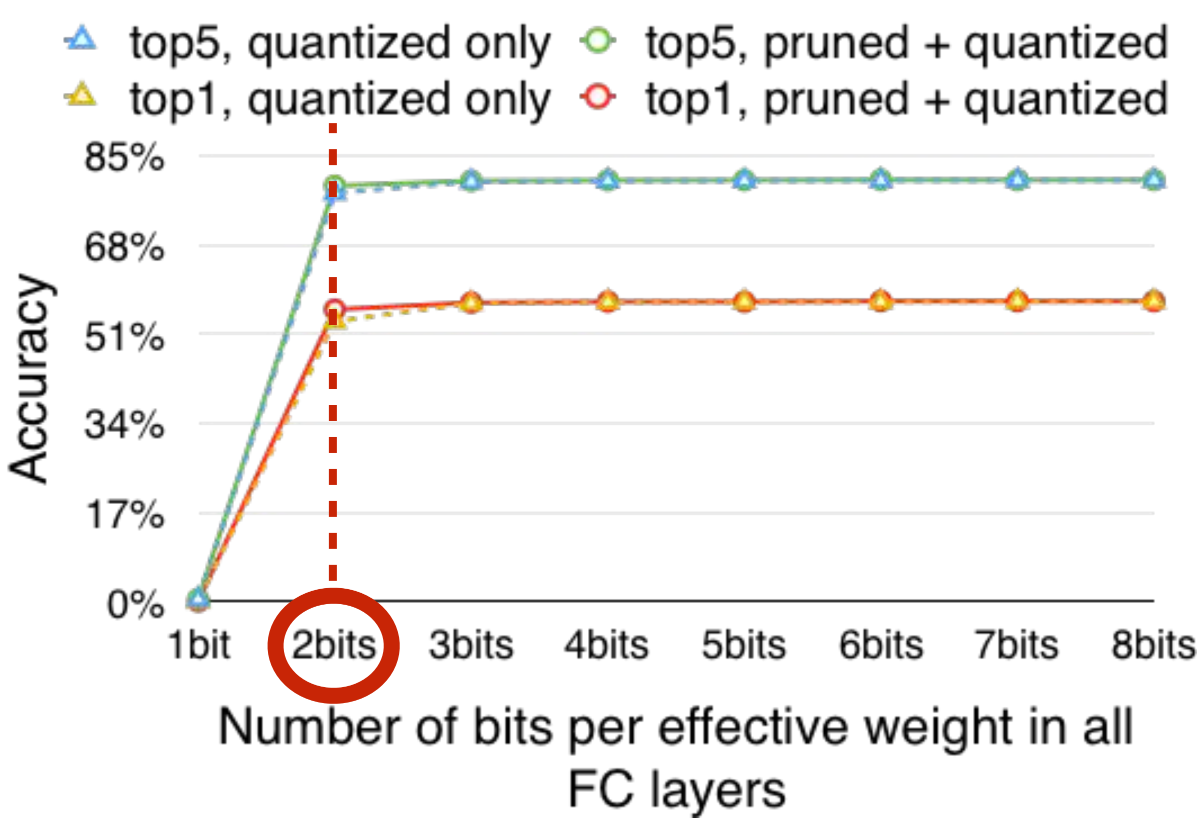
Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

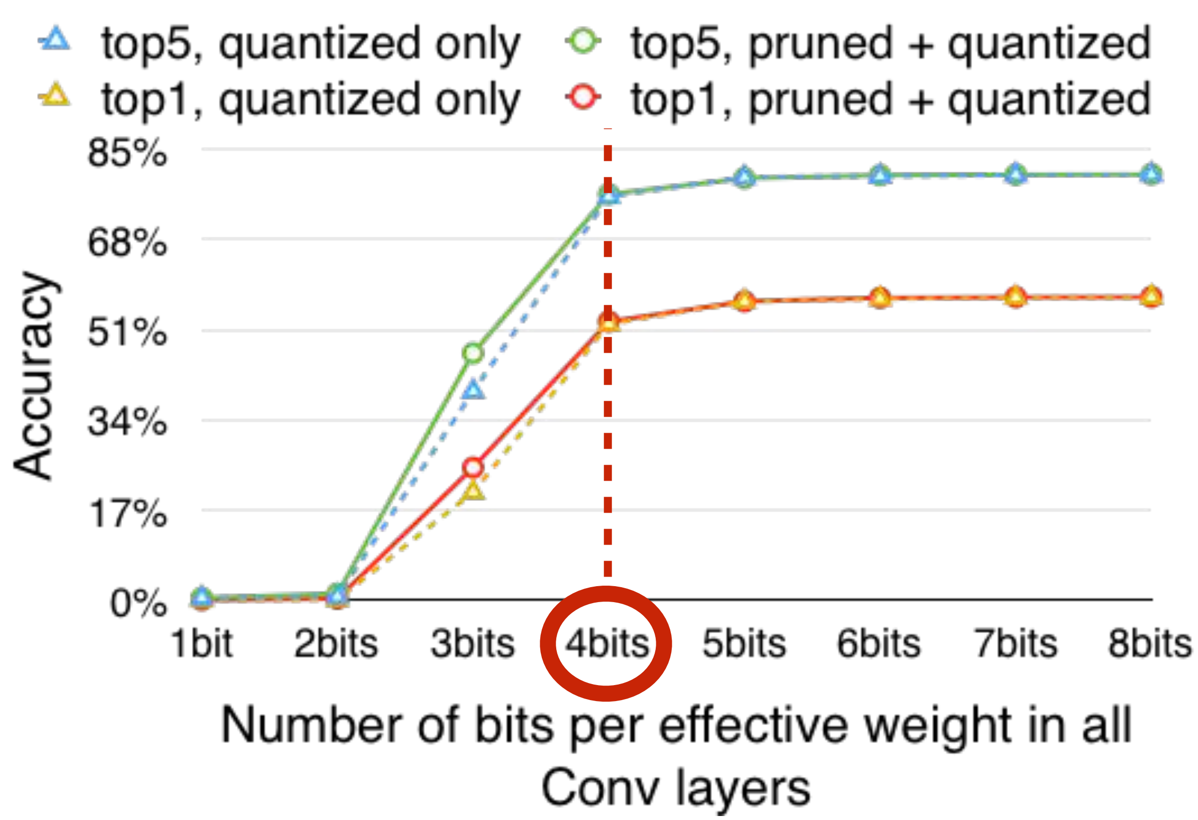
Trained Quantization Changes Weight Distribution



Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

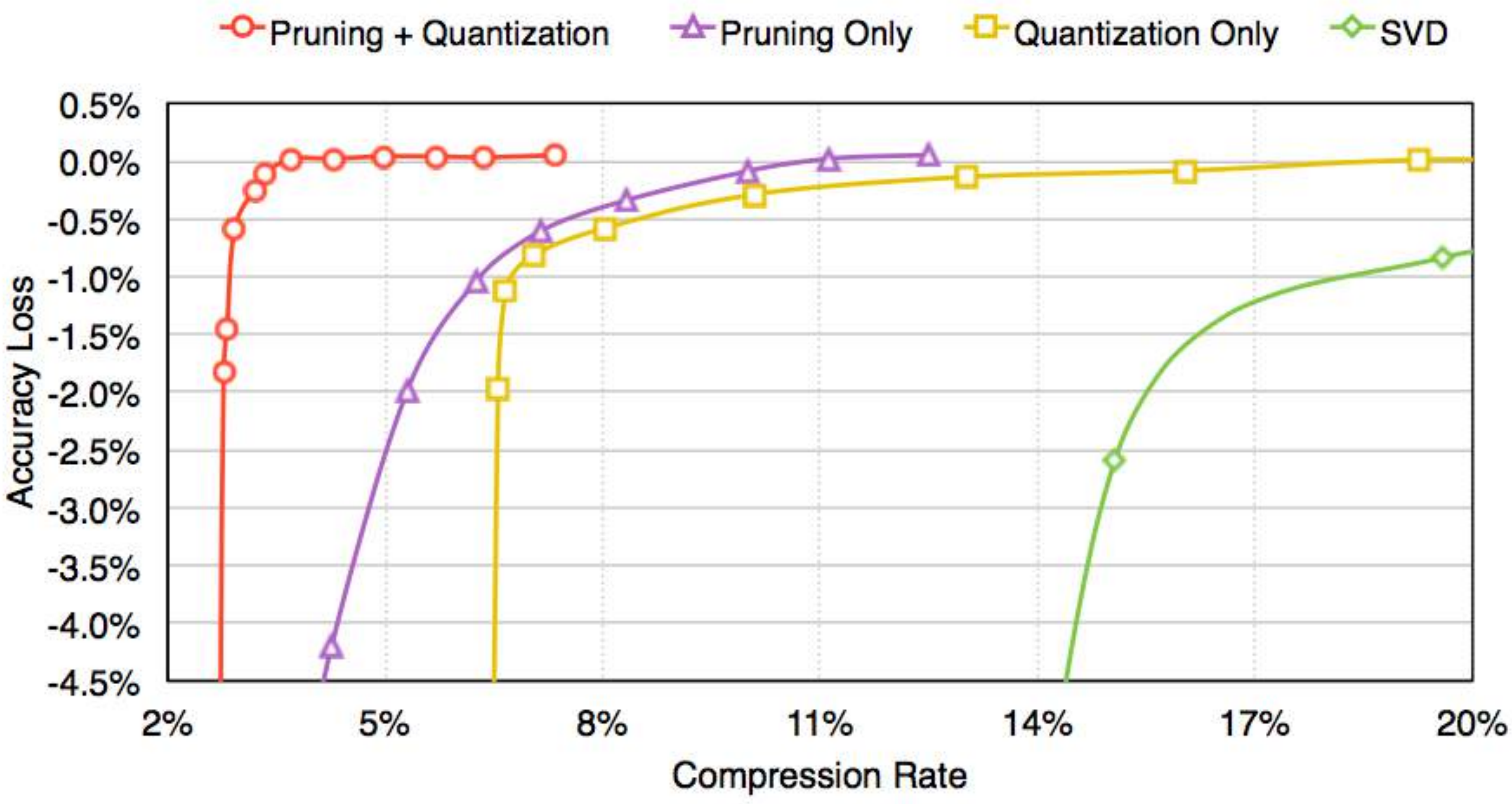
Bits Per Weight





Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

Pruning + Trained Quantization



AlexNet on ImageNet

Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

Results of Deep Compression

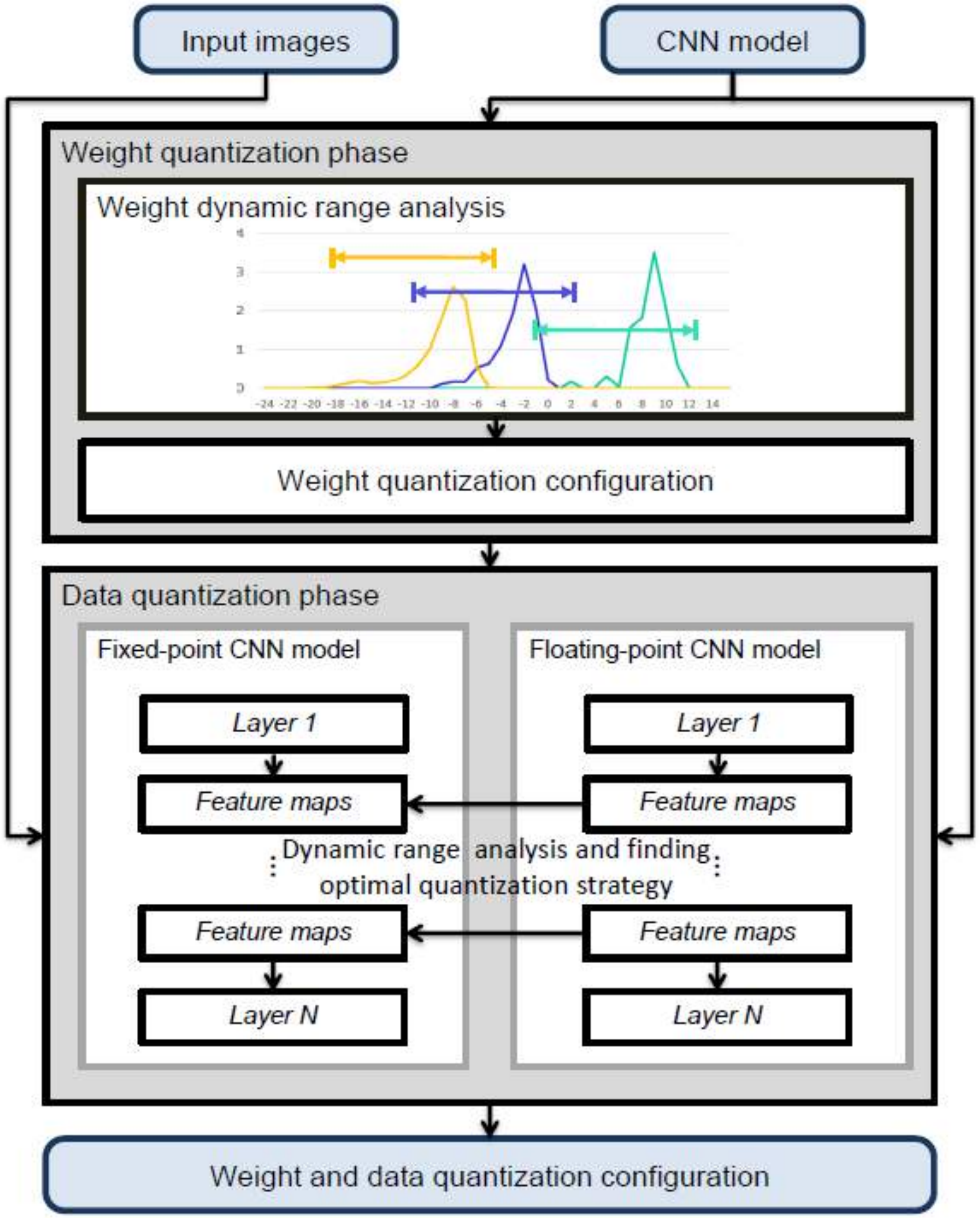
Network	Original Size	Compressed Size	Compression Ratio	Original Accuracy	Compressed Accuracy
LeNet-300	1070KB →	27KB	**40x**	98.36% →	98.42%
LeNet-5	1720KB →	44KB	**39x**	99.20% →	99.26%
AlexNet	240MB →	6.9MB	**35x**	80.27% →	80.30%
VGGNet	550MB →	11.3MB	**49x**	88.68% →	89.09%
GoogleNet	28MB →	2.8MB	**10x**	88.90% →	88.92%
SqueezeNet	4.8MB →	0.47MB	**10x**	80.32% →	80.35%

Han et al. Deep Compression, ICLR 2016 (Best Paper Award)

Part 1: Algorithms for Efficient Inference

- 1. Pruning
- 2. Weight Sharing
- 3. Quantization
- 4. Low Rank Approximation
- 5. Binary / Ternary Net
- 6. Winograd Transformation

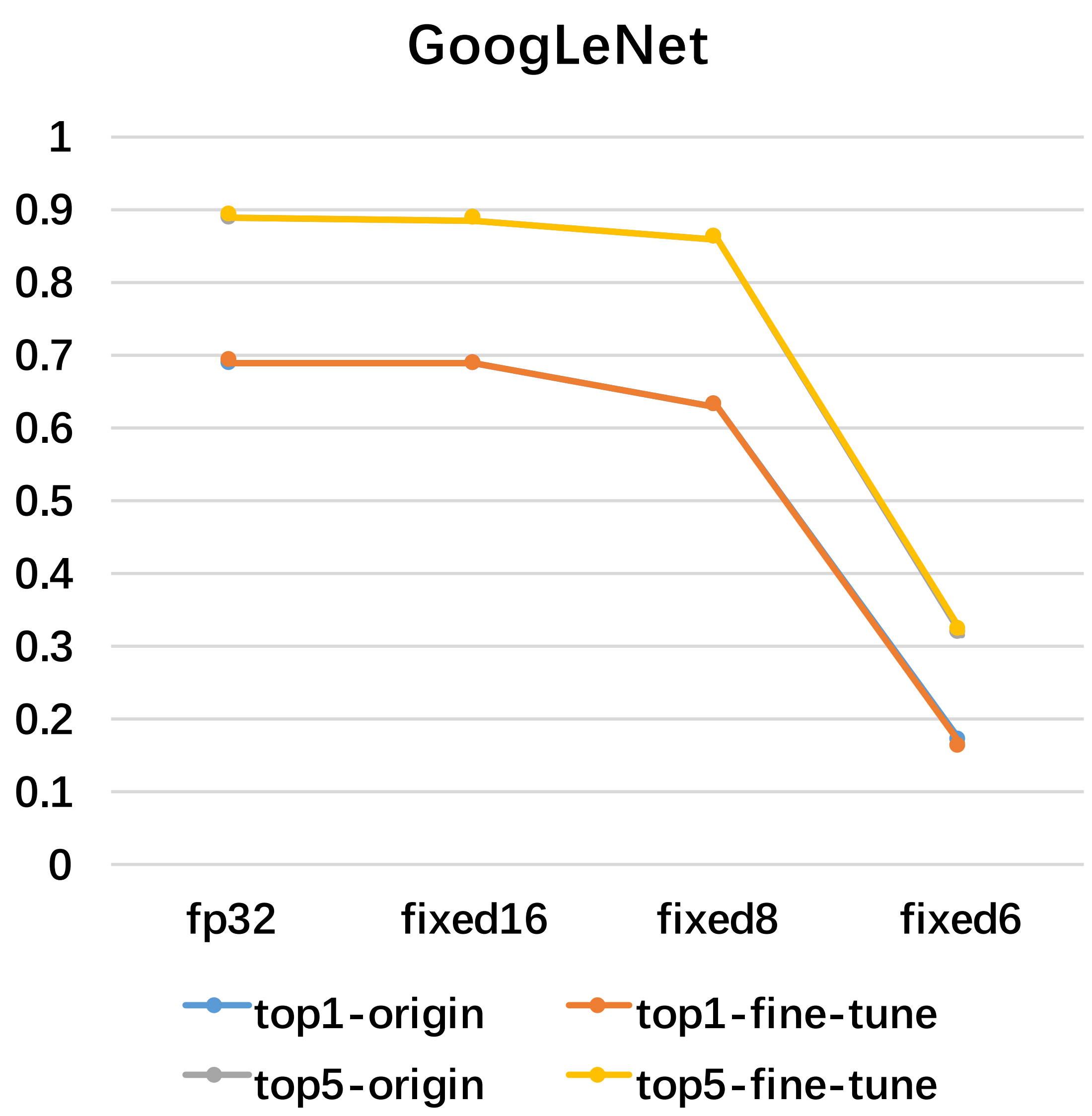
Quantizing the Weight and Activation

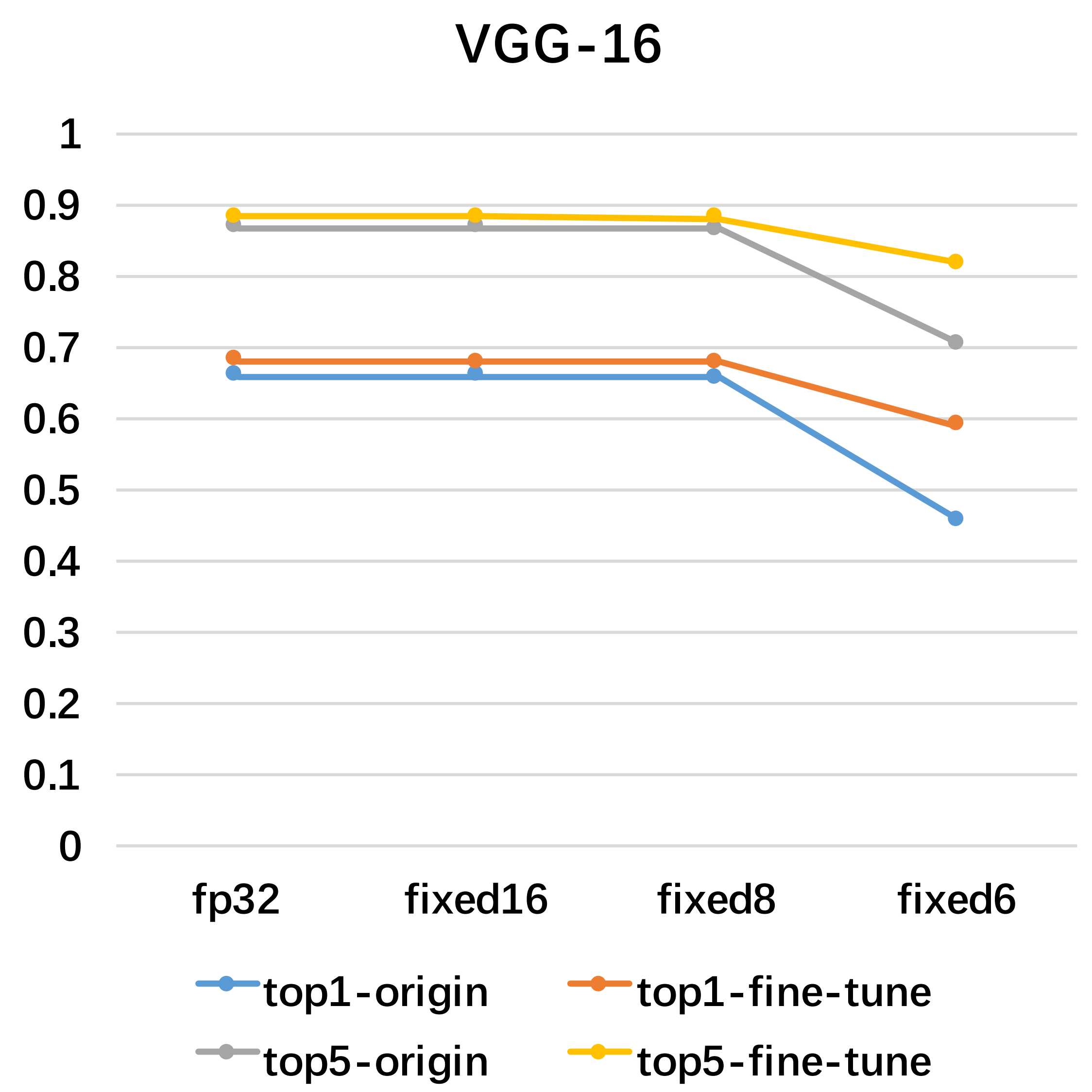


- Train with float
- Quantizing the weight and activation:
  - Gather the statistics for weight and activation
  - Choose proper radix point position
- Fine-tune in float format
- Convert to fixed-point format

Qiu et al. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network, FPGA'16

Quantization Result





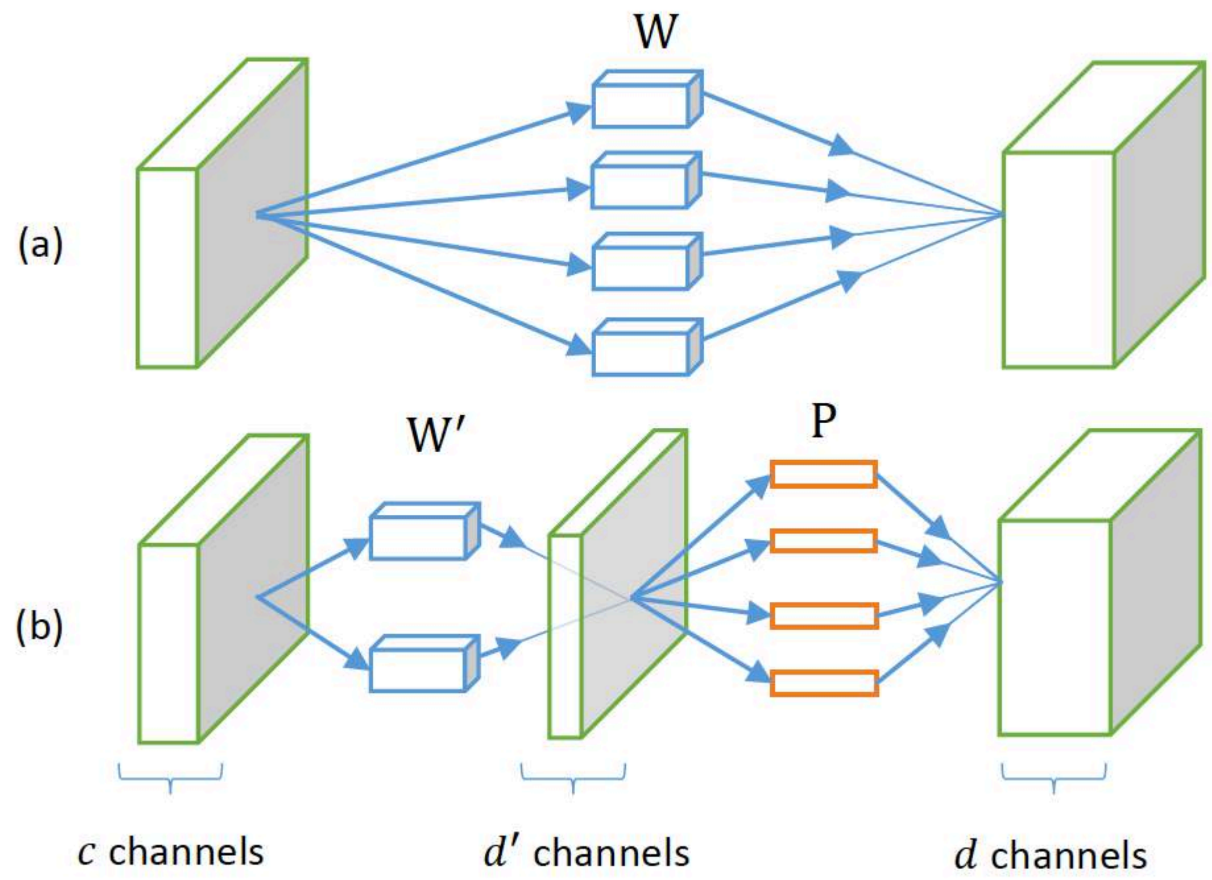
Qiu et al. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network, FPGA'16

Part 1: Algorithms for Efficient Inference

- 1. Pruning
- 2. Weight Sharing
- 3. Quantization
- 4. Low Rank Approximation
- 5. Binary / Ternary Net
- 6. Winograd Transformation

Low Rank Approximation for Conv

- Layer responses lie in a low-rank subspace
- Decompose a convolutional layer with d filters with filter size $k \times k \times c$ to
  - A layer with d' filters ($k \times k \times c$)
  - A layer with d filter ($1 \times 1 \times d'$)



Zhang et al Efficient and Accurate Approximations of Nonlinear Convolutional Networks CVPR'15

Low Rank Approximation for Conv

speedup	rank sel.	Conv1	Conv2	Conv3	Conv4	Conv5	Conv6	Conv7	err. $\uparrow$ %
$2\times$	no	32	110	199	219	219	219	219	1.18
$2\times$	**yes**	32	83	182	211	239	237	253	**0.93**
$2.4\times$	no	32	96	174	191	191	191	191	1.77
$2.4\times$	**yes**	32	74	162	187	207	205	219	**1.35**
$3\times$	no	32	77	139	153	153	153	153	2.56
$3\times$	**yes**	32	62	138	149	166	162	167	**2.34**
$4\times$	no	32	57	104	115	115	115	115	4.32
$4\times$	**yes**	32	50	112	114	122	117	119	**4.20**
$5\times$	no	32	46	83	92	92	92	92	6.53
$5\times$	**yes**	32	41	94	93	98	92	90	**6.47**

Zhang et al Efficient and Accurate Approximations of Nonlinear Convolutional Networks CVPR'15

Low Rank Approximation for FC

Build a mapping from row / column indices of matrix $\boldsymbol{W} = [W(x, y)]$ to vectors $\boldsymbol{i}$ and $\boldsymbol{j}$: $x \leftrightarrow \boldsymbol{i} = (i_1, \ldots, i_d)$ and $y \leftrightarrow \boldsymbol{j} = (j_1, \ldots, j_d)$.

TT-format for matrix $\boldsymbol{W}$:

$$\boldsymbol{W}(i_1, \ldots, i_d; j_1, \ldots, j_d) = \boldsymbol{W}(x(\boldsymbol{i}), y(\boldsymbol{j})) = \underbrace{\boldsymbol{G}_1[i_1, j_1]}_{1\times r} \underbrace{\boldsymbol{G}_2[i_2, j_2]}_{r\times r} \ldots \underbrace{\boldsymbol{G}_d[i_d, j_d]}_{r\times 1}$$

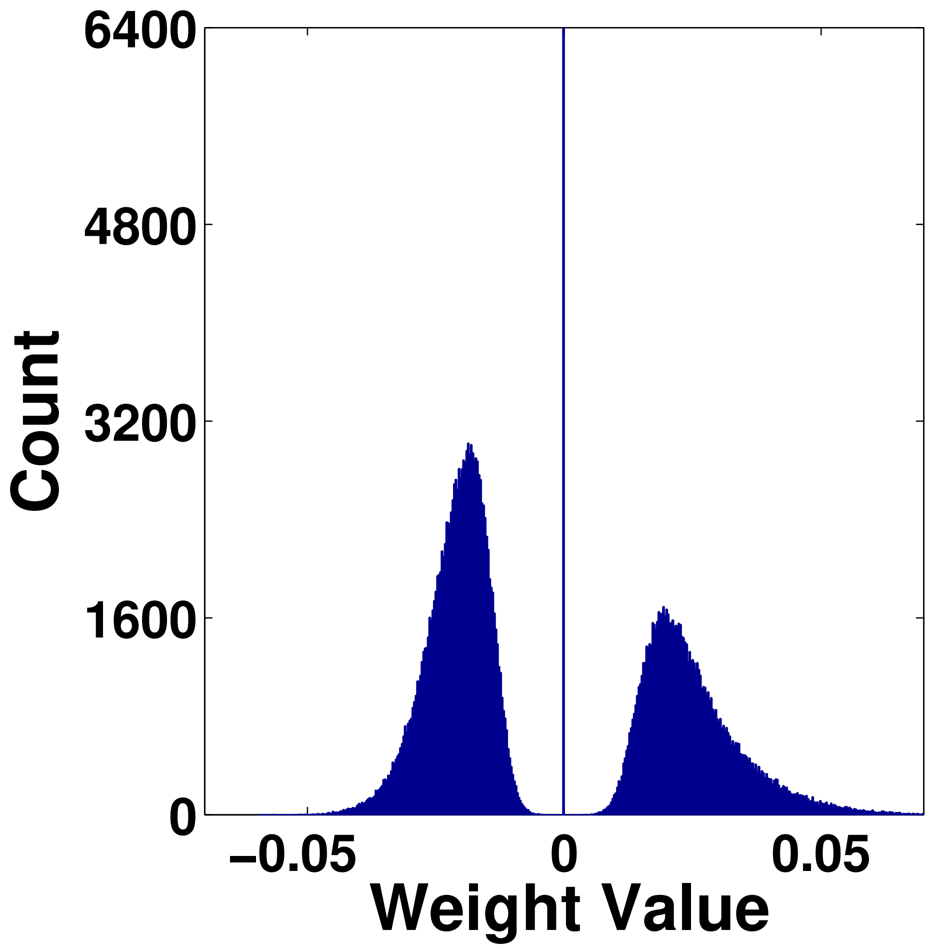
Type	1 im. time (ms)	100 im. time (ms)
CPU fully-connected layer	16.1	97.2
CPU TT-layer	1.2	94.7
GPU fully-connected layer	2.7	33
GPU TT-layer	1.9	12.9

Novikov et al Tensorizing Neural Networks, NIPS'15

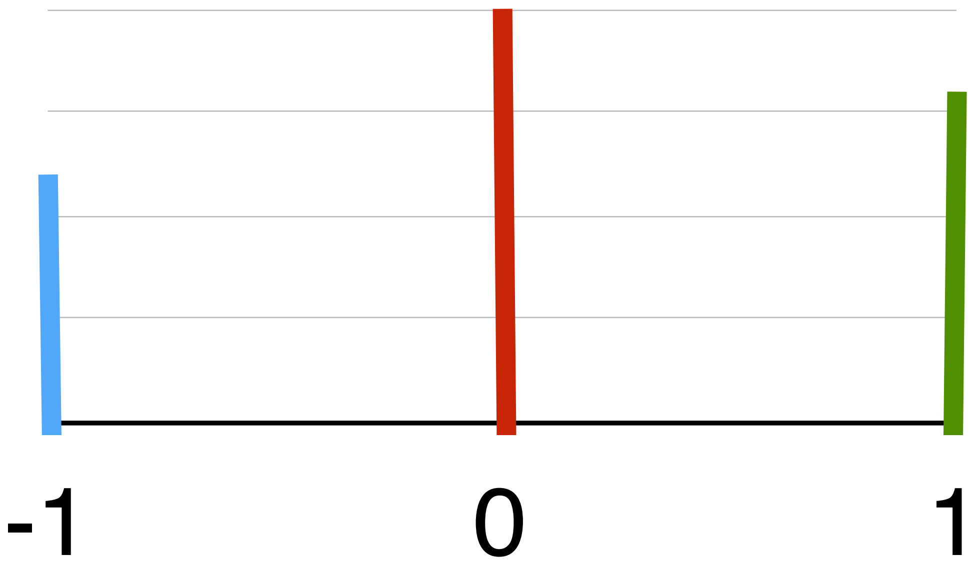
Part 1: Algorithms for Efficient Inference

- 1. Pruning
- 2. Weight Sharing
- 3. Quantization
- 4. Low Rank Approximation
- 5. Binary / Ternary Net
- 6. Winograd Transformation

Binary / Ternary Net: Motivation



=>



Binary-Weight-Network and XNOR-Network

$$\mathbf{X}^\intercal \mathbf{W} \approx \beta \mathbf{H}^\intercal \alpha \mathbf{B}$$

$\mathbf{H}, \mathbf{B} \in \{+1, -1\}^n$ and $\beta, \alpha \in \mathbb{R}^+$

- Binarize both weights and inputs
- Convolution as Binary dot product
- Dot product between implemented by XNOR-Bitcounting operations

$$\alpha^*, \mathbf{B}^*, \beta^*, \mathbf{H}* = \underset{\alpha, \mathbf{B}, \beta, \mathbf{H}}{\operatorname{argmin}} \|\mathbf{X}^\intercal \mathbf{W} - \beta \alpha \mathbf{H}^\intercal \mathbf{B}\|$$

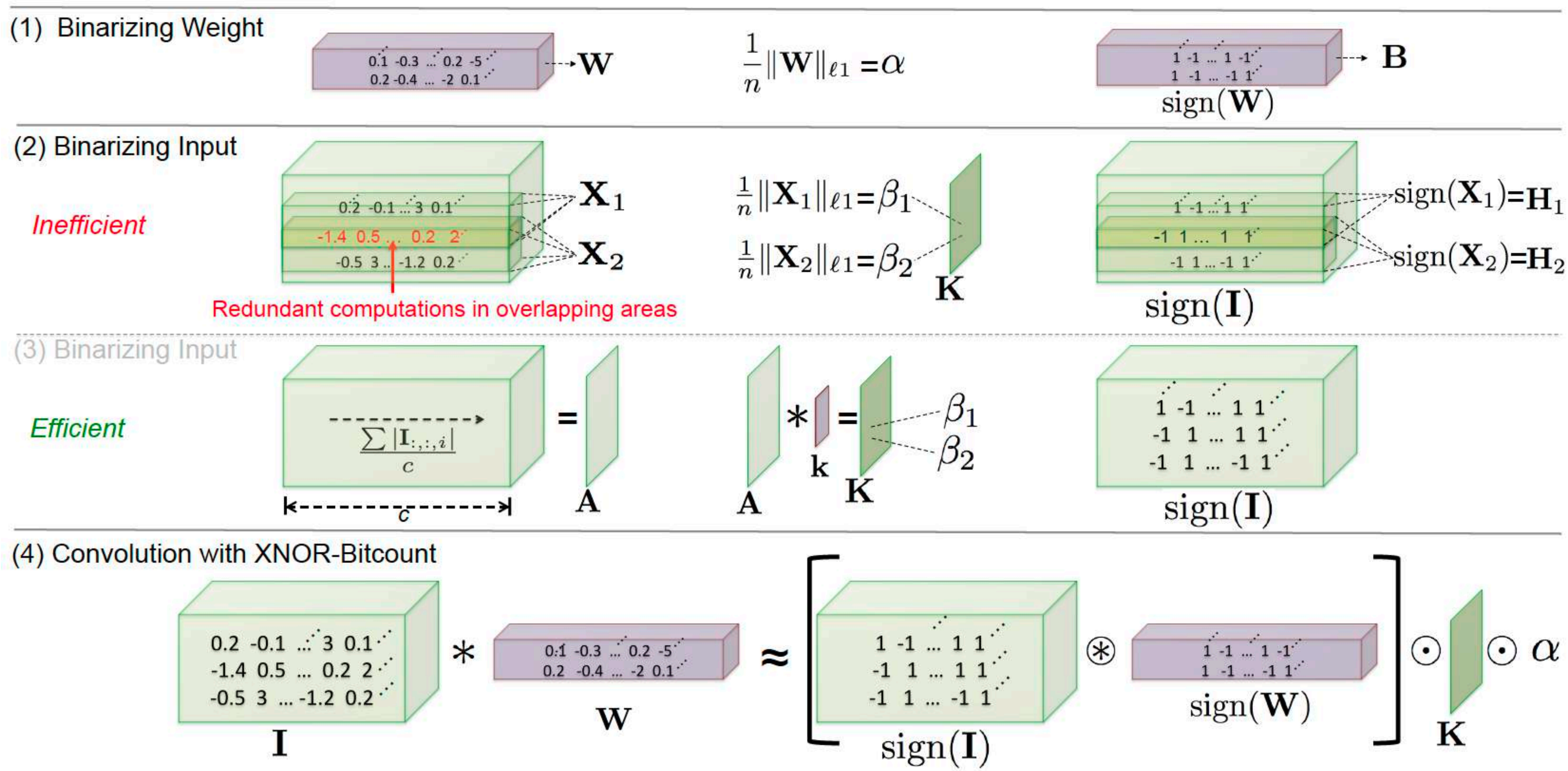
$$\gamma^*, \mathbf{C}^* = \underset{\gamma, \mathbf{C}}{\operatorname{argmin}} \|\mathbf{1}^\intercal \mathbf{Y} - \gamma \mathbf{1}^\intercal \mathbf{C}\|$$

$$\mathbf{C}^* = \operatorname{sign}(\mathbf{Y}) = \operatorname{sign}(\mathbf{X}^\intercal) \operatorname{sign}(\mathbf{W}) = \mathbf{H}^{*\intercal} \mathbf{B}^*$$

$$\gamma^* = \frac{\sum |\mathbf{Y}_i|}{n} = \frac{\sum |\mathbf{X}_i||\mathbf{W}_i|}{n} \approx \left(\frac{1}{n}\|\mathbf{X}\|_{\ell 1}\right)\left(\frac{1}{n}\|\mathbf{W}\|_{\ell 1}\right) = \beta^* \alpha^*$$

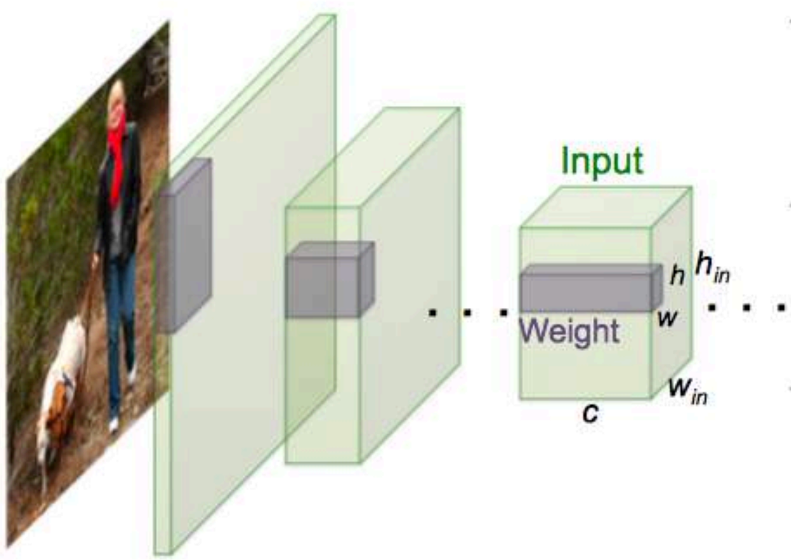
Rastegari et al. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks ECCV 2016

Binary-Weight-Network and XNOR-Network



Rastegari et al. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks ECCV 2016

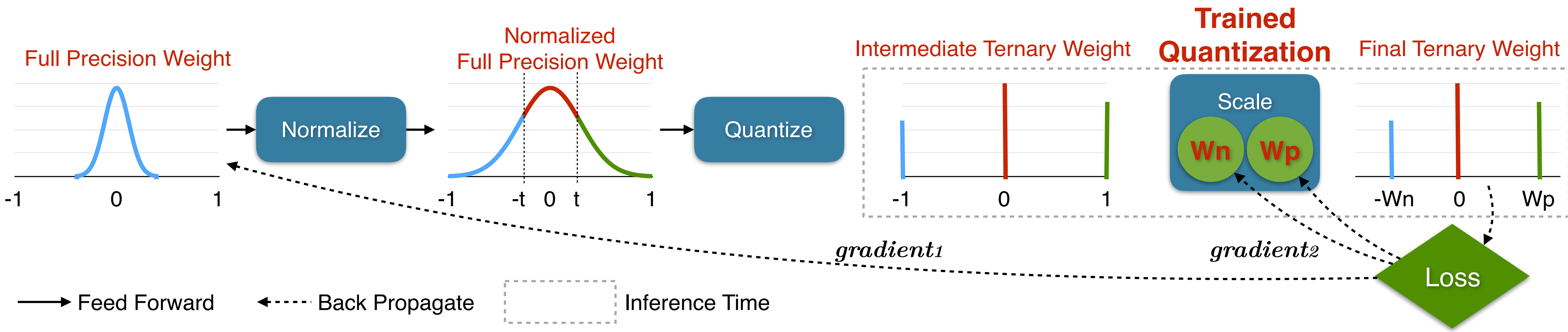
Binary-Weight-Network and XNOR-Network



	Network Variations	Operations used in Convolution	Memory Saving (Inference)	Computation Saving (Inference)	Accuracy on ImageNet (AlexNet)
Standard Convolution	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Real-Value Weights: 0.12 -1.2 ... 0.41 / -0.2 0.5 ... 0.68	+, −, ×	1x	1x	%56.7
Binary Weight	Real-Value Inputs: 0.11 -0.21 ... -0.34 / -0.25 0.61 ... 0.52; Binary Weights: 1 -1 ... 1 / -1 1 ... 1	+, −	~32x	~2x	%56.8
BinaryWeight Binary Input (**XNOR-Net**)	Binary Inputs: 1 -1 ... -1 / -1 1 ... 1; Binary Weights: 1 -1 ... 1 / -1 1 ... 1	XNOR, bitcount	~32x	~58x	%44.2

Rastegari et al. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks ECCV 2016

Trained Ternary Quantization



Zhu, Han, Mao, Dally. Trained Ternary Quantization, ICLR'17

Weight Evolution during Training

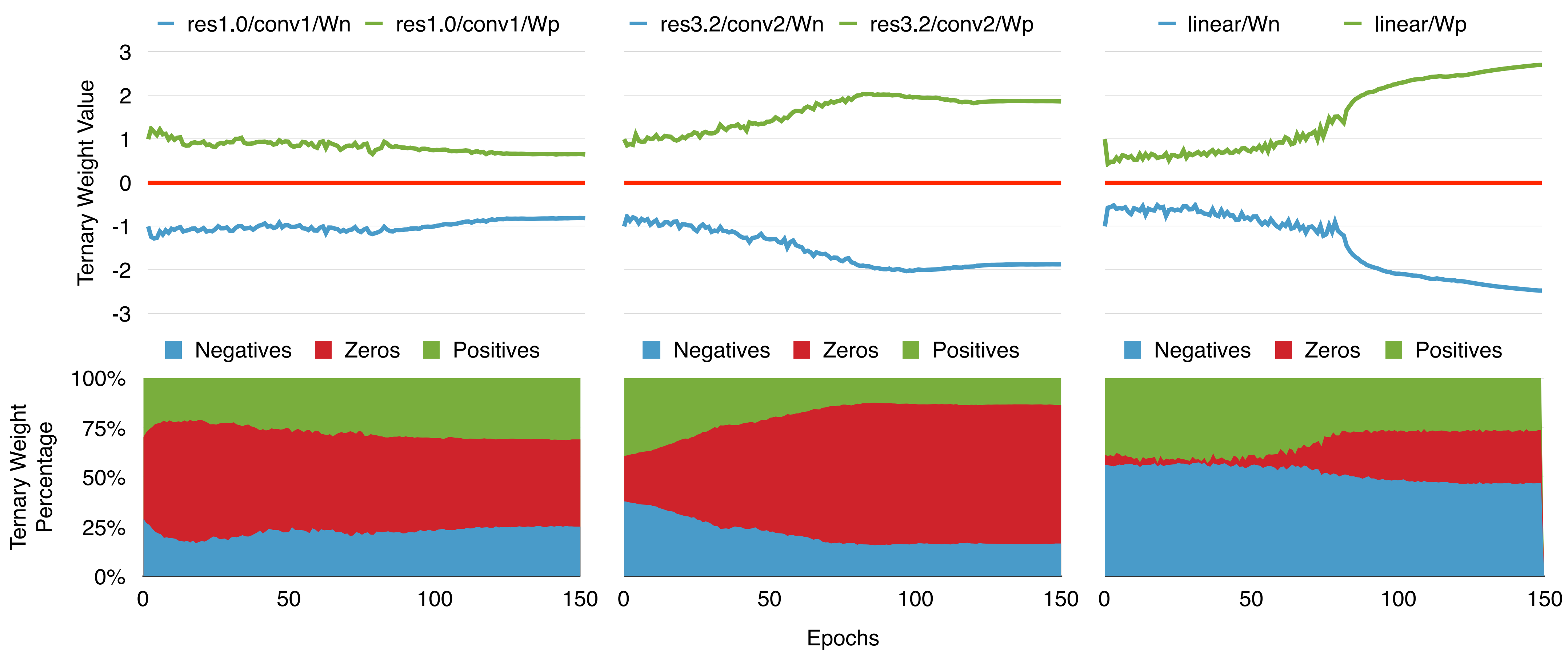
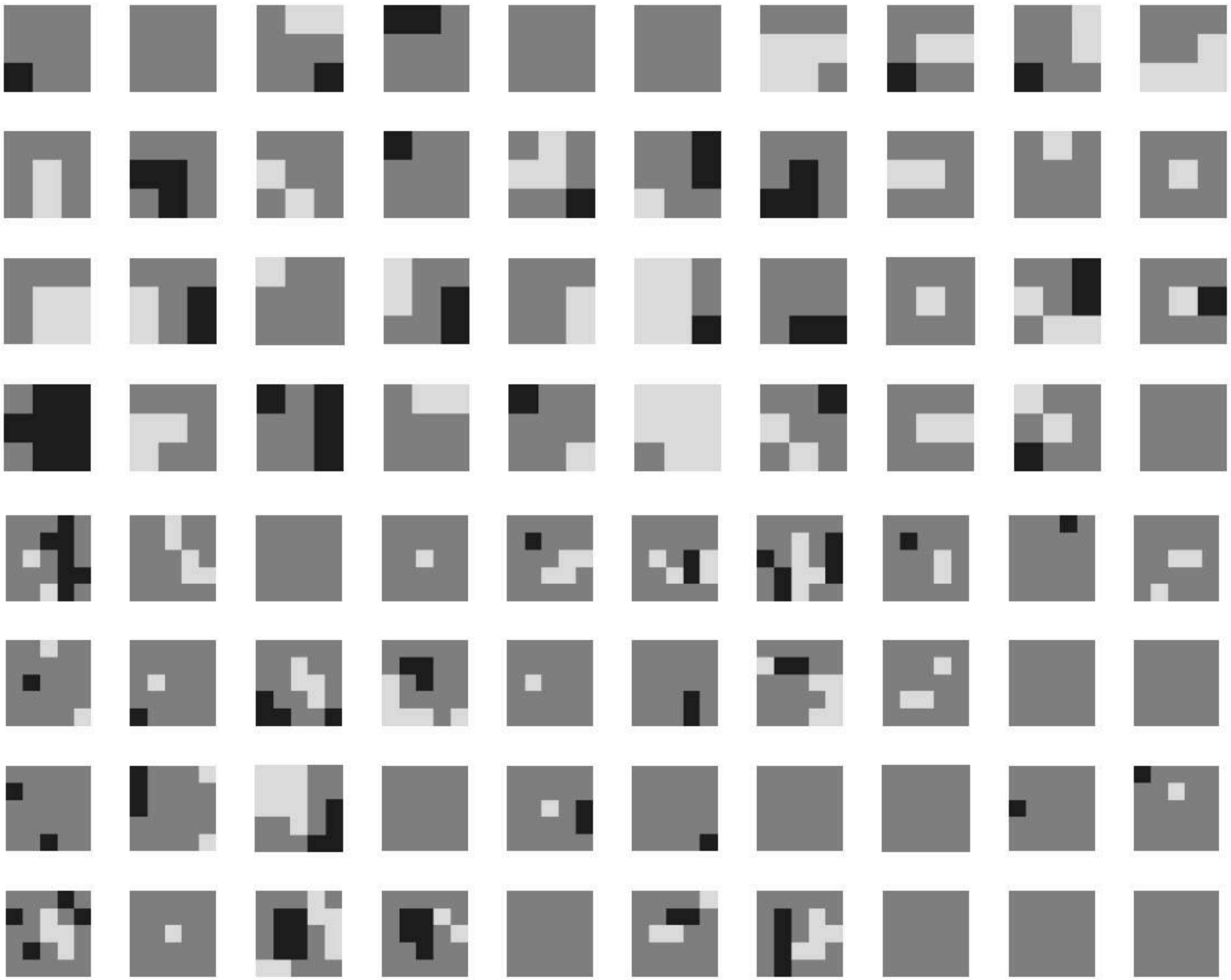


Figure 2: Ternary weights value (above) and distribution (below) with iterations for different layers of ResNet-20 on CIFAR-10.

Visualization of the TTQ Kernels



Zhu, Han, Mao, Dally. Trained Ternary Quantization, ICLR’17

Error Rate on ImageNet

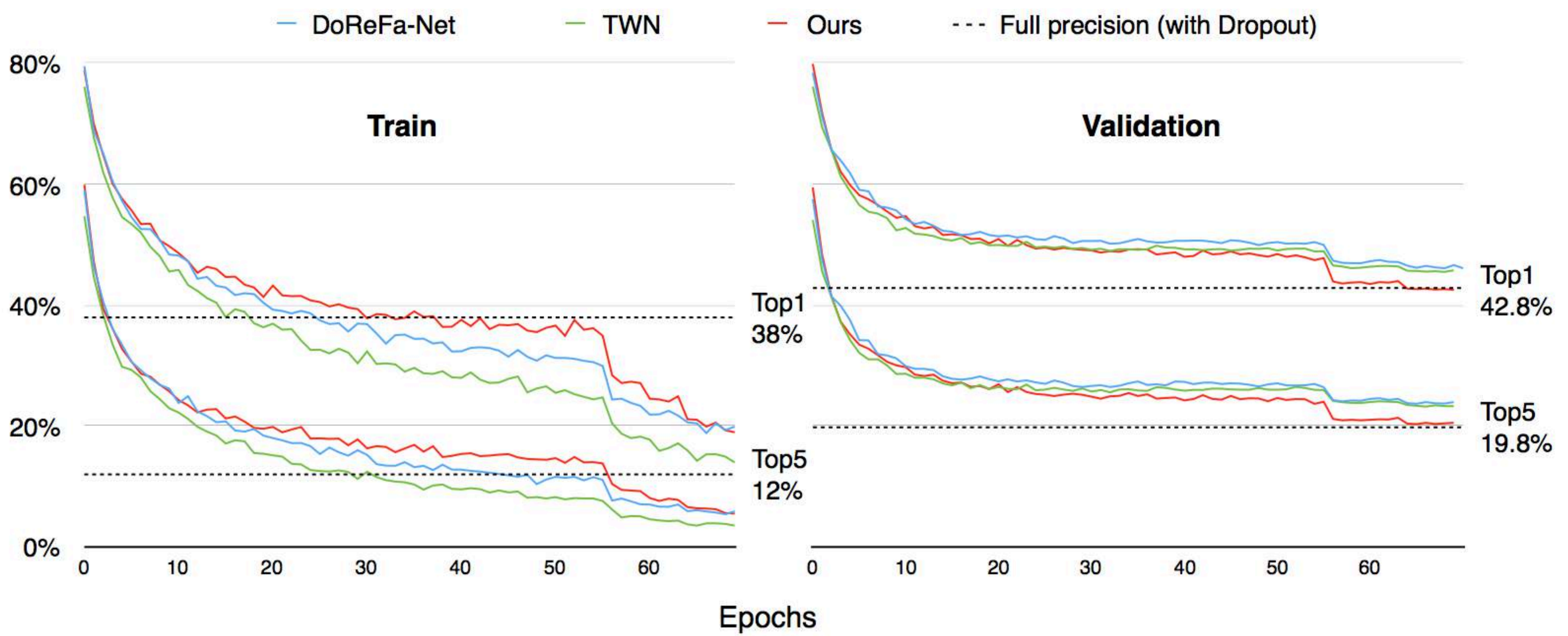


Figure 4: Training and validation accuracy of AlexNet on ImageNet

Zhu, Han, Mao, Dally. Trained Ternary Quantization, ICLR'17

Related Works

Lin, Zhouhan, et al. "Neural networks with few multiplications." *arXiv preprint arXiv:1510.03009* (2015).

Introduce Binary and Ternary Connection.

Use probabilistic quantization method.

Introduce concept of latent weights.

Zhou, Shuchang, et al. "DoReFa-Net: Training Low Bitwidth Convolutional Neural Networks with Low Bitwidth Gradients." *arXiv preprint arXiv:1606.06160* (2016).

Adopt Thresholding quantization method.

Related Works

Li, Fengfu, and Bin Liu. "Ternary Weight Networks." *arXiv preprint arXiv:1605.04711* (2016).

Treat quantization as

$$\alpha^*, \Delta^* = \underset{\alpha \geq 0, \Delta > 0}{\operatorname{argmin}} ||\mathbf{W} - \alpha \mathbf{W}^t||_2^2$$

$$\alpha^*_\Delta = \frac{1}{|\mathbf{I}_\Delta|} \sum_{i \in \mathbf{I}_\Delta} |\mathbf{W}_i|. \qquad \Delta^* = 0.7 \cdot E(|\mathbf{W}|) \approx \frac{0.7}{n} \sum_{i=1}^{n} |\mathbf{W}_i|$$

Back-propagate using identical mapping.

Rastegari, Mohammad, et al. "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks." *arXiv preprint arXiv:1603.05279*(2016).

Same strategy on binary weights.

Part 1: Algorithms for Efficient Inference

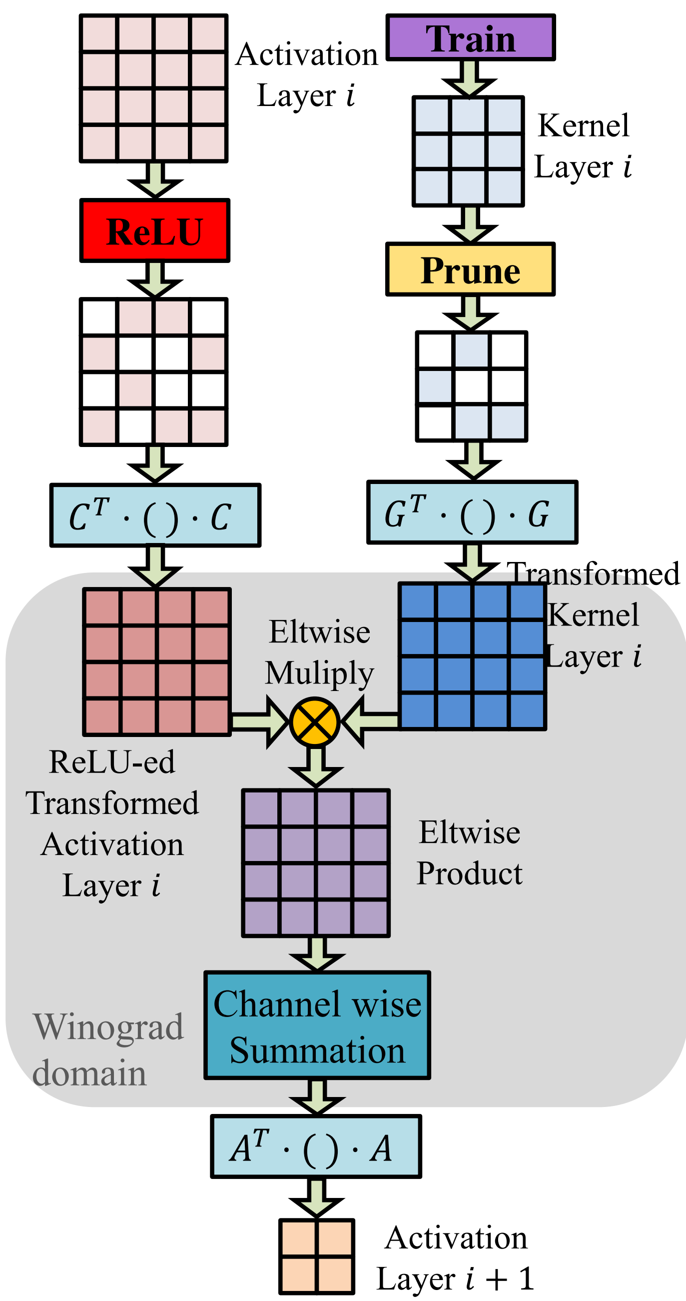
- 1. Pruning
- 2. Weight Sharing
- 3. Quantization
- 4. Low Rank Approximation
- 5. Binary / Ternary Net
- 6. Winograd Transformation

Winograd Convolution

$$
\begin{aligned}
Y_{i,k,\widetilde{x},\widetilde{y}} &= \sum_{c=1}^{C} D_{i,c,\widetilde{x},\widetilde{y}} * G_{k,c} \\
&= \sum_{c=1}^{C} A^T \left[U_{k,c} \odot V_{c,i,\widetilde{x},\widetilde{y}} \right] A \\
&= A^T \left[\sum_{c=1}^{C} U_{k,c} \odot V_{c,i,\widetilde{x},\widetilde{y}} \right] A
\end{aligned}
$$

Winograd. *Arithmetic complexity of computations*, volume 33. Siam, 1980
Lavin & Gray, Fast Algorithms for Convolutional Neural Networks, 2015

Training in the Winograd Domain



(a)

Producing 4 output pixels:

Direct Convolution:

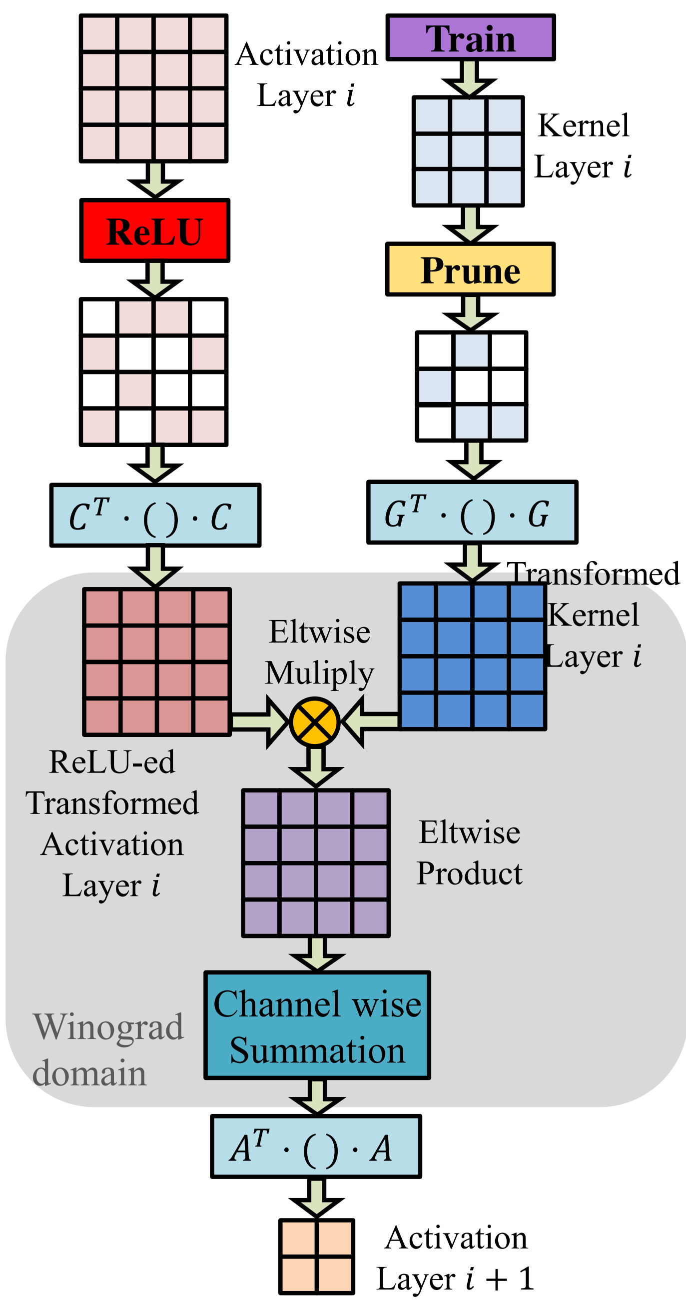
- 4*9=36 multiplications (**1x**)

Winograd convolution:

- 4*4=16 multiplications (**2.25x** less)

Liu et al. "Efficient Sparse-Winograd Convolutional Neural Networks", submitted to ICLR 2017 workshop

Training in the Winograd Domain



(a)

Producing 4 output pixels:

Direct Convolution:

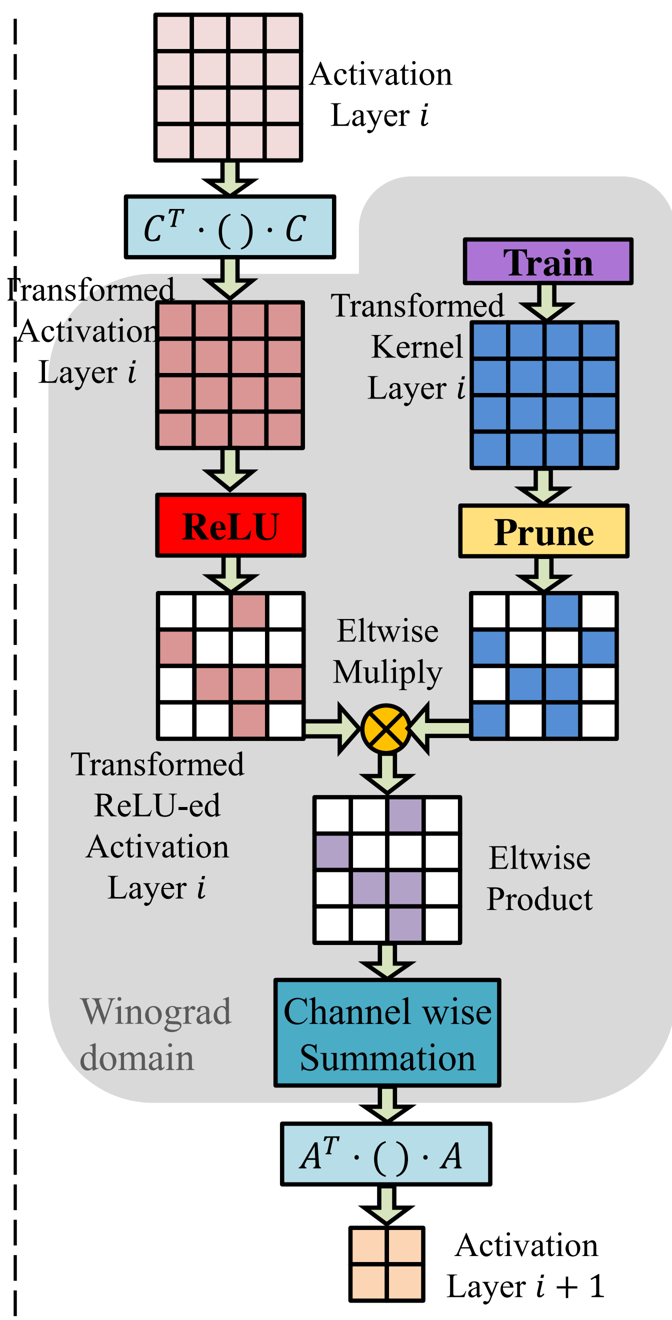
- 4*9=36 multiplications (**1x**)
- sparse weight [NIPS'15] (**3x**)
- sparse activation (relu) (**3x**)
- Overall saving: **9x**

Winograd convolution:

- 4*4=16 multiplications (**2.25x** less)
- dense weight (**1x**)
- dense activation (**1x**)
- Overall saving: **2.25x**

Liu et al. "Efficient Sparse-Winograd Convolutional Neural Networks", submitted to ICLR 2017 workshop

Solution: Fold Relu into Winograd



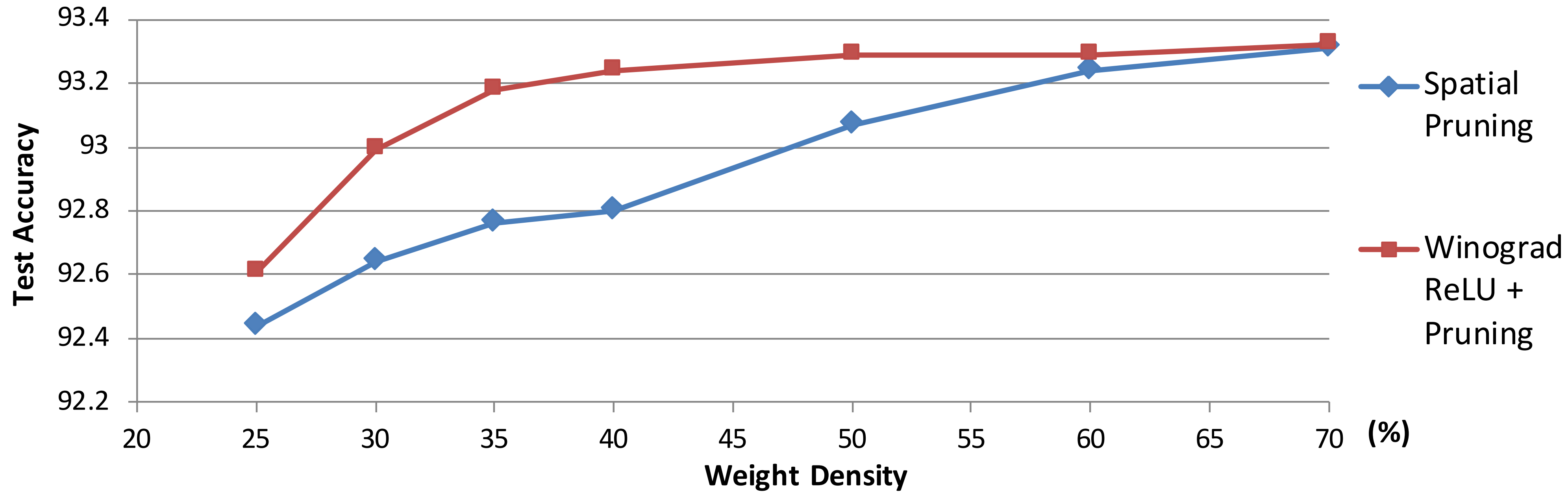
Producing 4 output pixels:

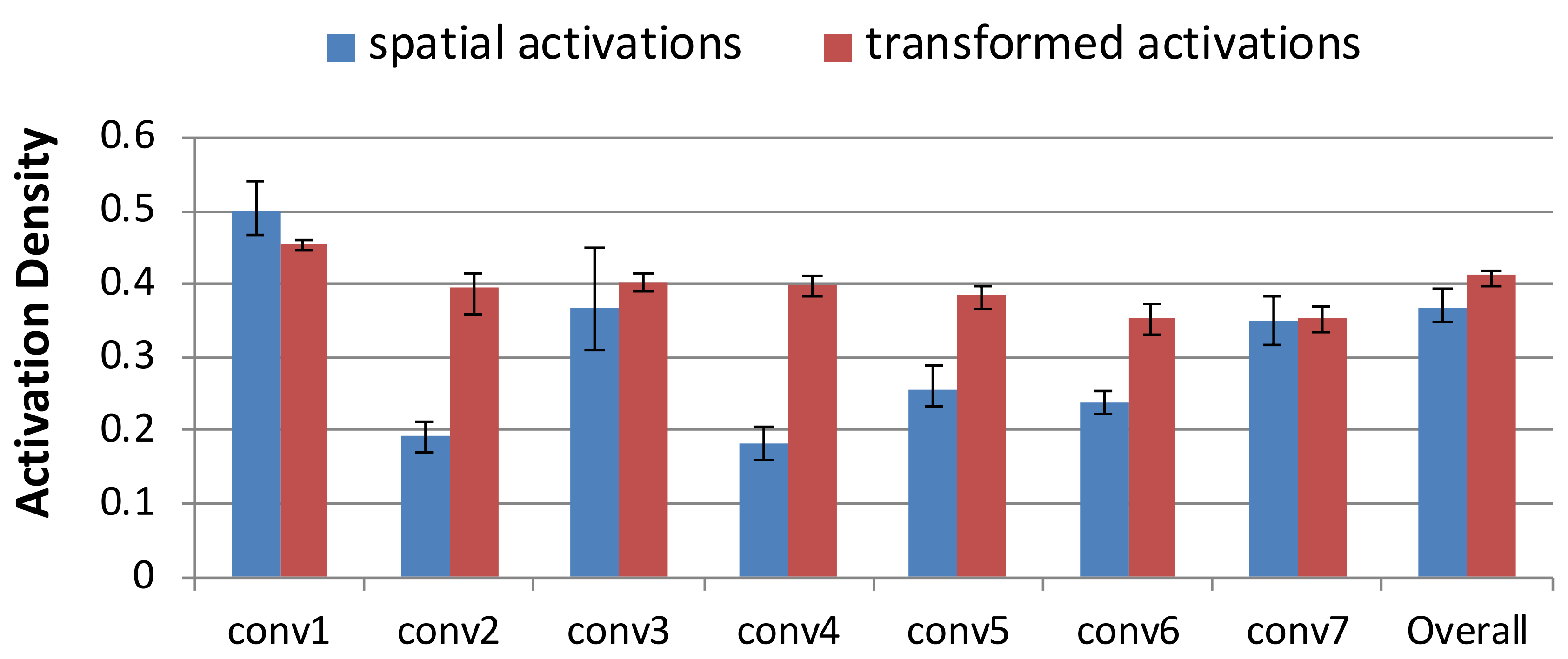
Direct Convolution:
- 4*9=36 multiplications (**1x**)
- sparse weight [NIPS'15] (**3x**)
- sparse activation (relu) (**3x**)
- Overall saving: **9x**

Winograd convolution:
- 4*4=16 multiplications (**2.25x** less)
- sparse weight (**2.5x**)
- dense activation (**2.25x**)
- Overall saving: **12x**

Liu et al. "Efficient Sparse-Winograd Convolutional Neural Networks", submitted to ICLR 2017 workshop

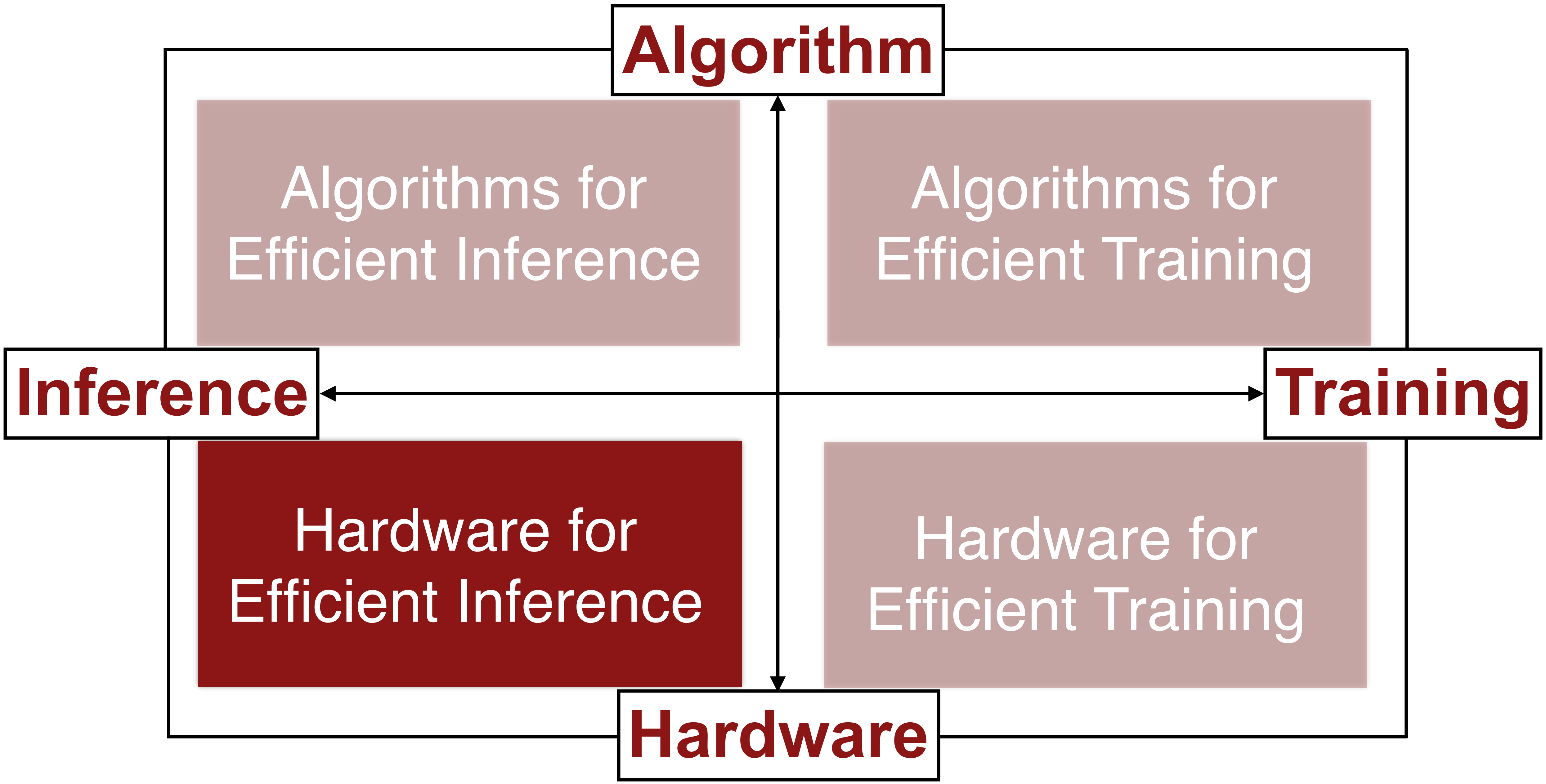
Result





Liu et al. “Efficient Sparse-Winograd Convolutional Neural Networks”, submitted to ICLR 2017 workshop

Agenda


Algorithm
Algorithms for Efficient Inference
Algorithms for Efficient Training
Inference
Training
Hardware for Efficient Inference
Hardware for Efficient Training
Hardware

Diannao (Electric Brain)

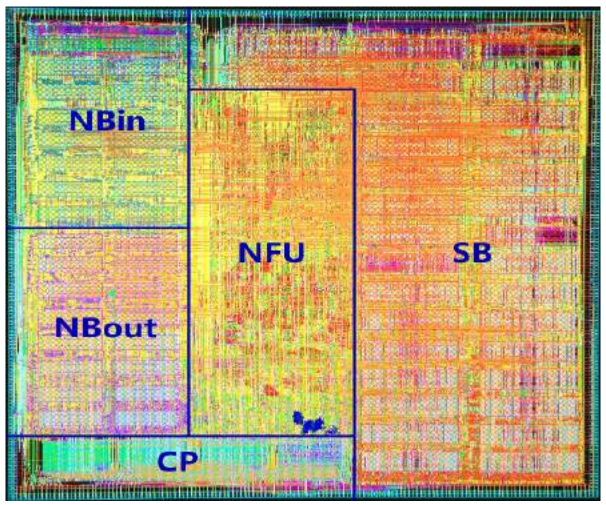


Figure 15. *Layout (65nm).*

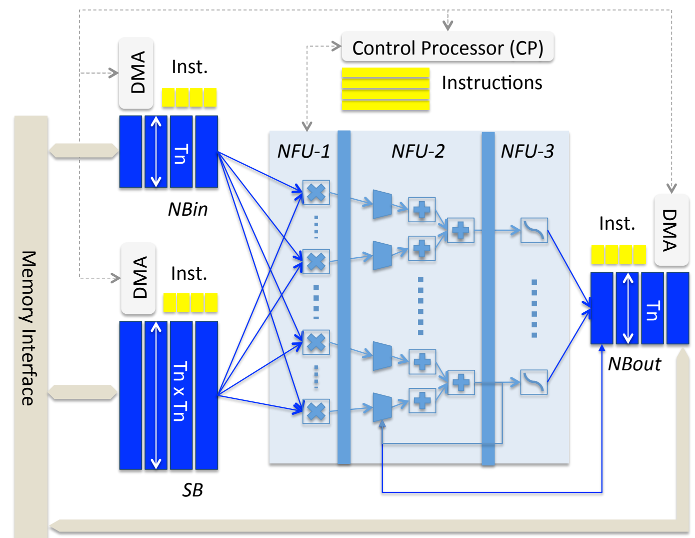


Figure 11. *Accelerator.*

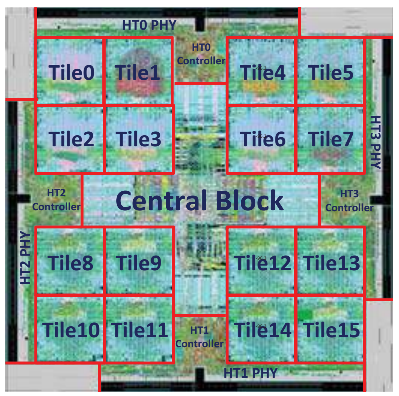
Component or Block	Area in μm^2	(%)	Power in mW	(%)	Critical path in ns
ACCELERATOR	3,023,077		485		1.02
Combinational	608,842	(20.14%)	89	(18.41%)	
Memory	1,158,000	(38.31%)	177	(36.59%)	
Registers	375,882	(12.43%)	86	(17.84%)	
Clock network	68,721	(2.27%)	132	(27.16%)	
Filler cell	811,632	(26.85%)			
SB	1,153,814	(38.17%)	105	(22.65%)	
NBin	427,992	(14.16%)	91	(19.76%)	
NBout	433,906	(14.35%)	92	(19.97%)	
NFU	846,563	(28.00%)	132	(27.22%)	
CP	141,809	(5.69%)	31	(6.39%)	
AXIMUX	9,767	(0.32%)	8	(2.65%)	
Other	9,226	(0.31%)	26	(5.36%)	

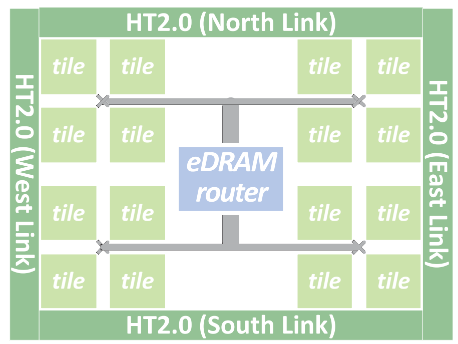
Table 6. *Characteristics of accelerator and breakdown by component type (first 5 lines), and functional block (last 7 lines).*

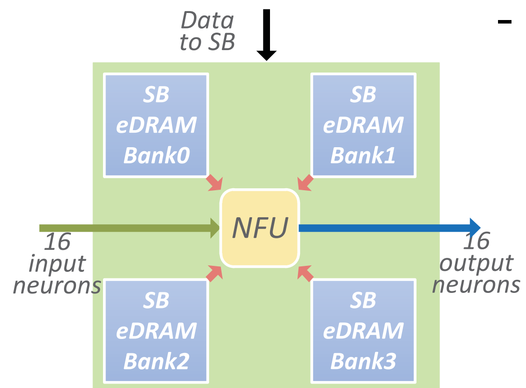
- Diannao improved CNN computation efficiency by using dedicated functional units and memory buffers optimized for the CNN workload.
- Multiplier + adder tree + shifter + non-linear lookup orchestrated by instructions
- Weights in off-chip DRAM
- 452 GOP/s, 3.02 mm^2 and 485 mW

Chen et al. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning, ASPLOS 2014

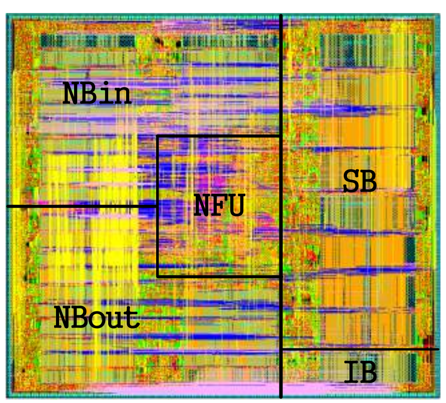
Diannao and Friends

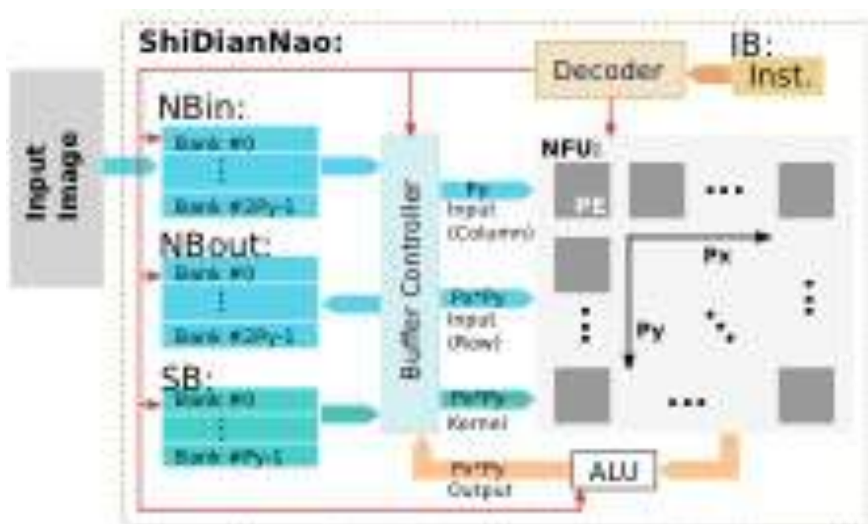






- DaDiannao (Bigger Computer) uses multi-chip and EDRAM to fit larger models. Each chip is 68mm^2 fitting 12 Million parameters, consumes 16W

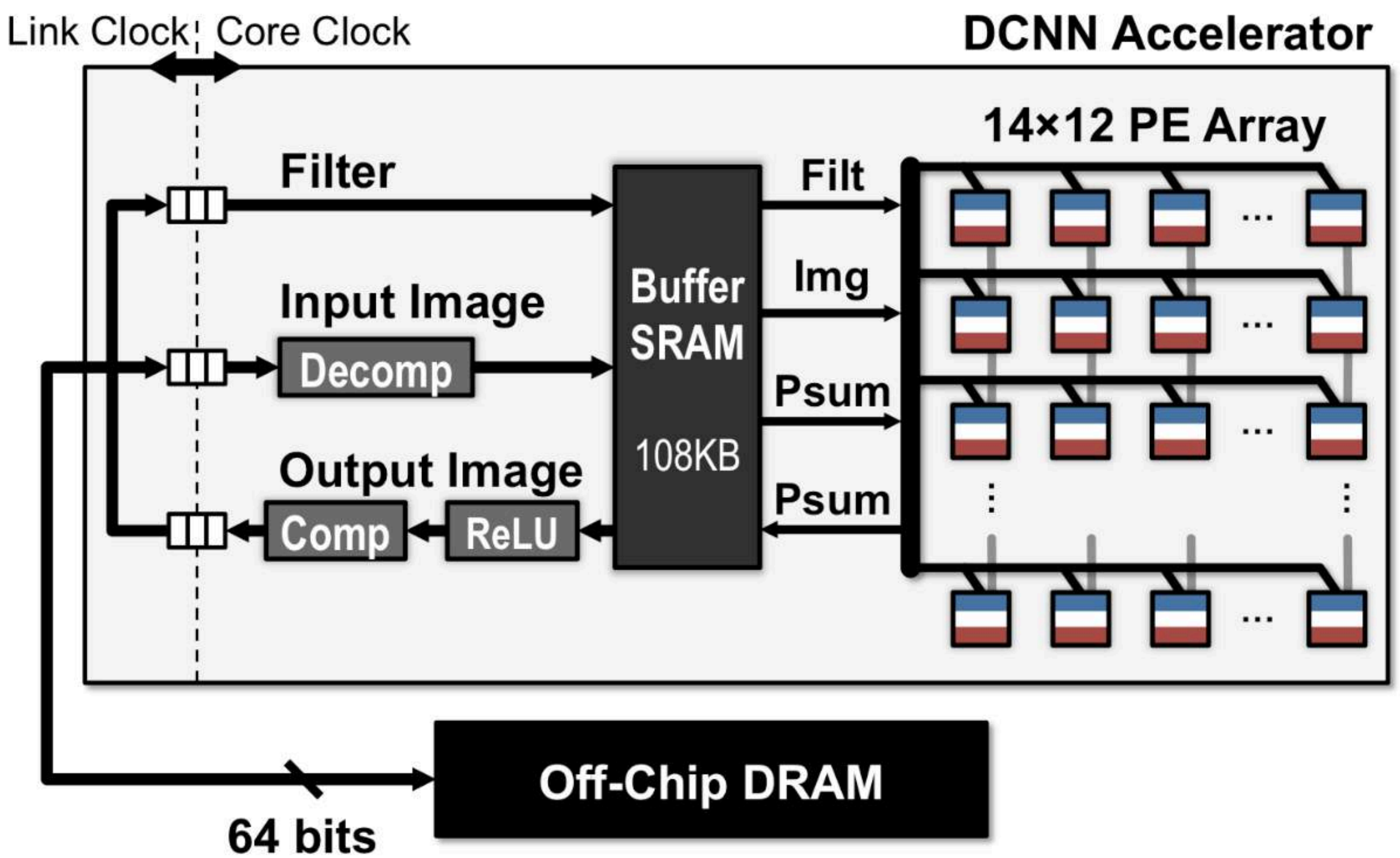




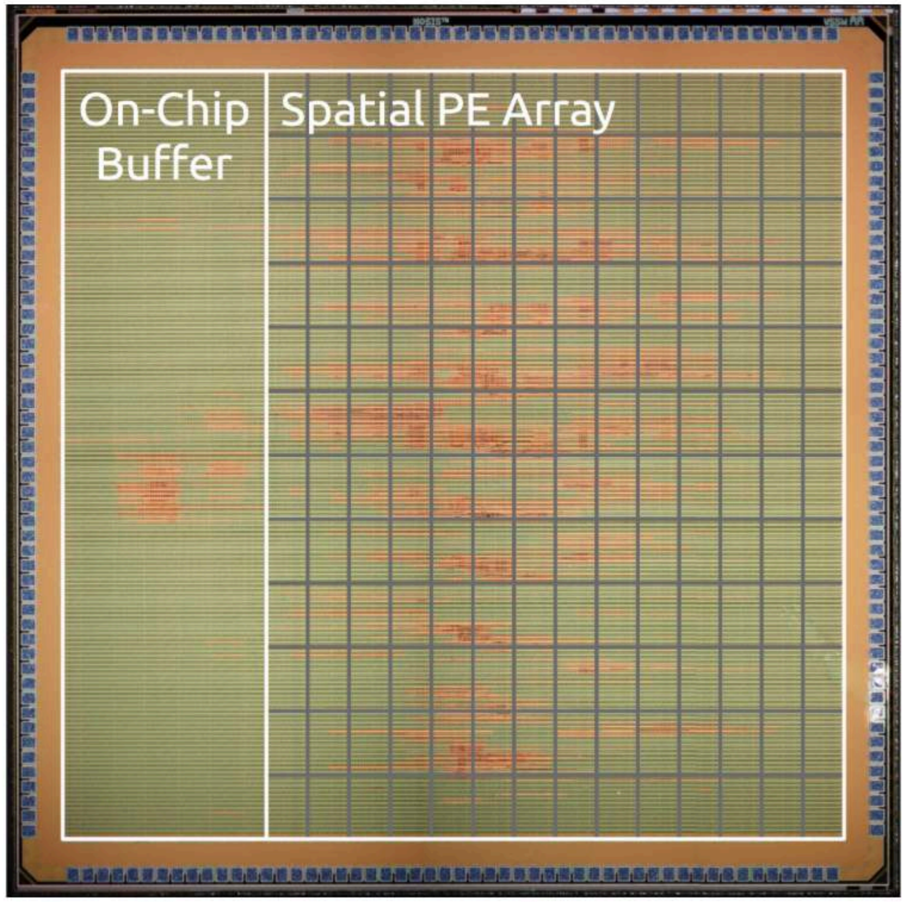
- ShiDiannao (Vision Computer) It can fits small model (up-to 64K parameters) on-chip. It maps the computation on 2D PE array. The chip is 4.86 mm^2 and consumes 320 mW

Chen et al. Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning, ASPLOS 2014

Eyeriss: Reduce Memory Access by Row-Stationary Dataflow



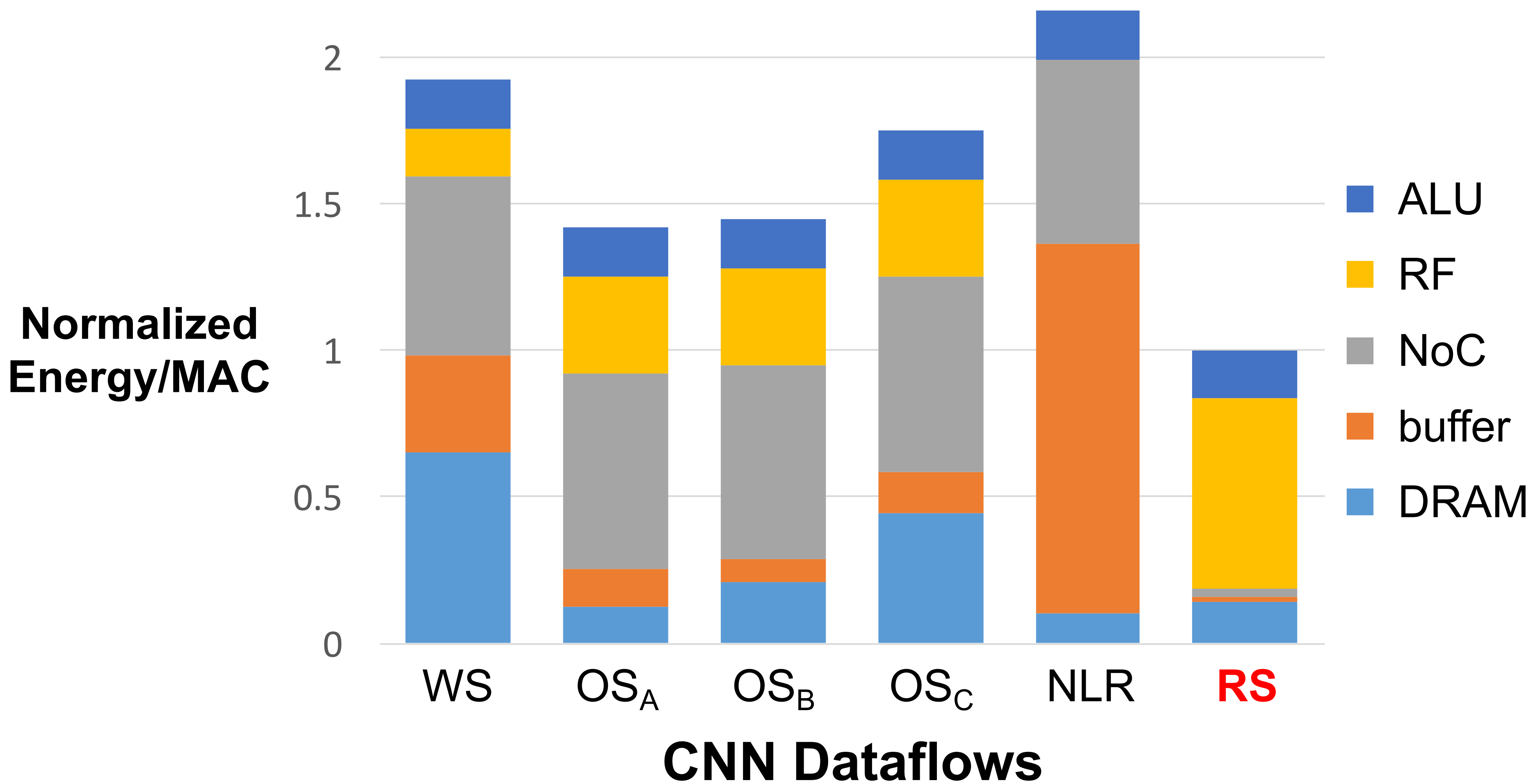
Eyeriss Architecture



Die Photo

Chen et al Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks ISCA 2016

Eyeriss: Reduce Memory Access by Row-Stationary Dataflow



RS uses **1.4× – 2.5× lower** energy than other dataflows

Chen et al Eyeriss: A Spatial Architecture for Energy-Efficient Dataflow for Convolutional Neural Networks ISCA 2016

EIE: Reduce Memory Access by Compression

logically

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

$$\times$$

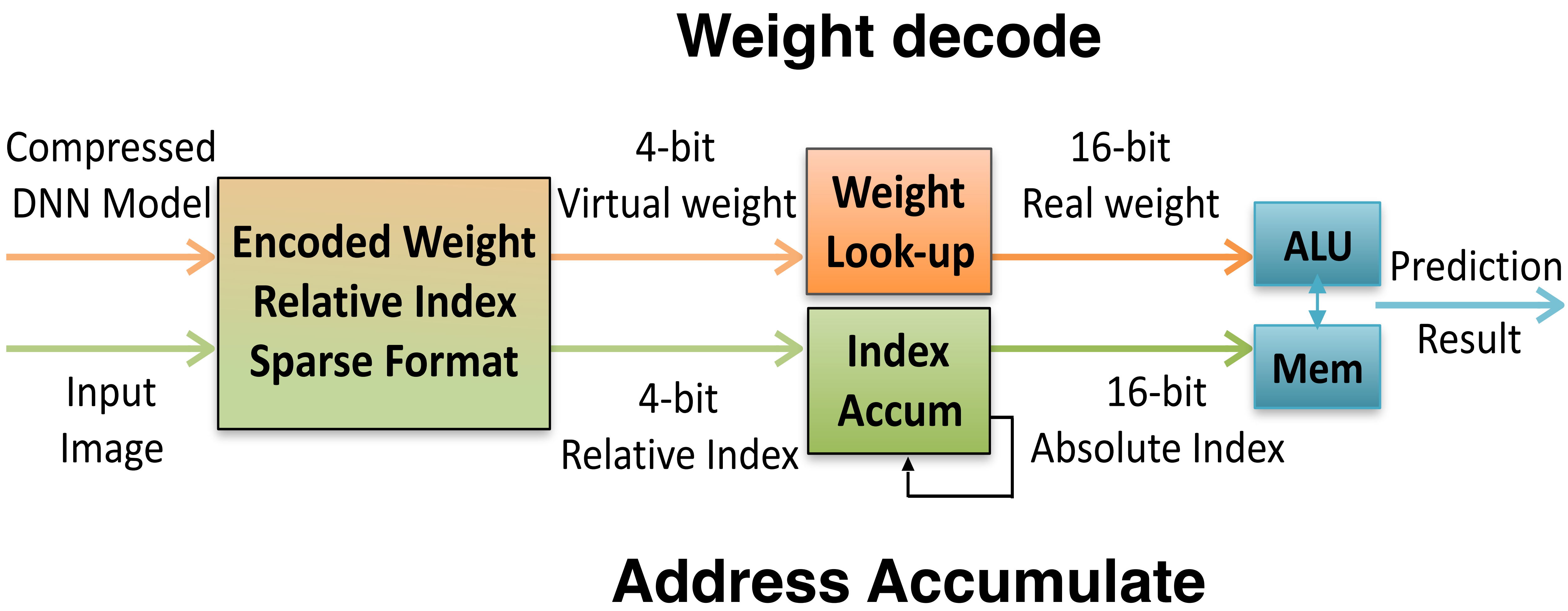
$$\begin{array}{c} PE0 \\ PE1 \\ PE2 \\ PE3 \\ \\ \\ \\ \\ \end{array} \begin{pmatrix} w_{0,0} & \boldsymbol{w_{0,1}} & 0 & w_{0,3} \\ 0 & \mathbf{0} & w_{1,2} & 0 \\ 0 & \boldsymbol{w_{2,1}} & 0 & w_{2,3} \\ 0 & \mathbf{0} & 0 & 0 \\ 0 & \mathbf{0} & w_{4,2} & w_{4,3} \\ w_{5,0} & \mathbf{0} & 0 & 0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

physically

Virtual Weight	$W_{0,0}$	$W_{0,1}$	$W_{4,2}$	$W_{0,3}$	$W_{4,3}$
Relative Index	0	1	2	0	0
Column Pointer	0	1	2	3	

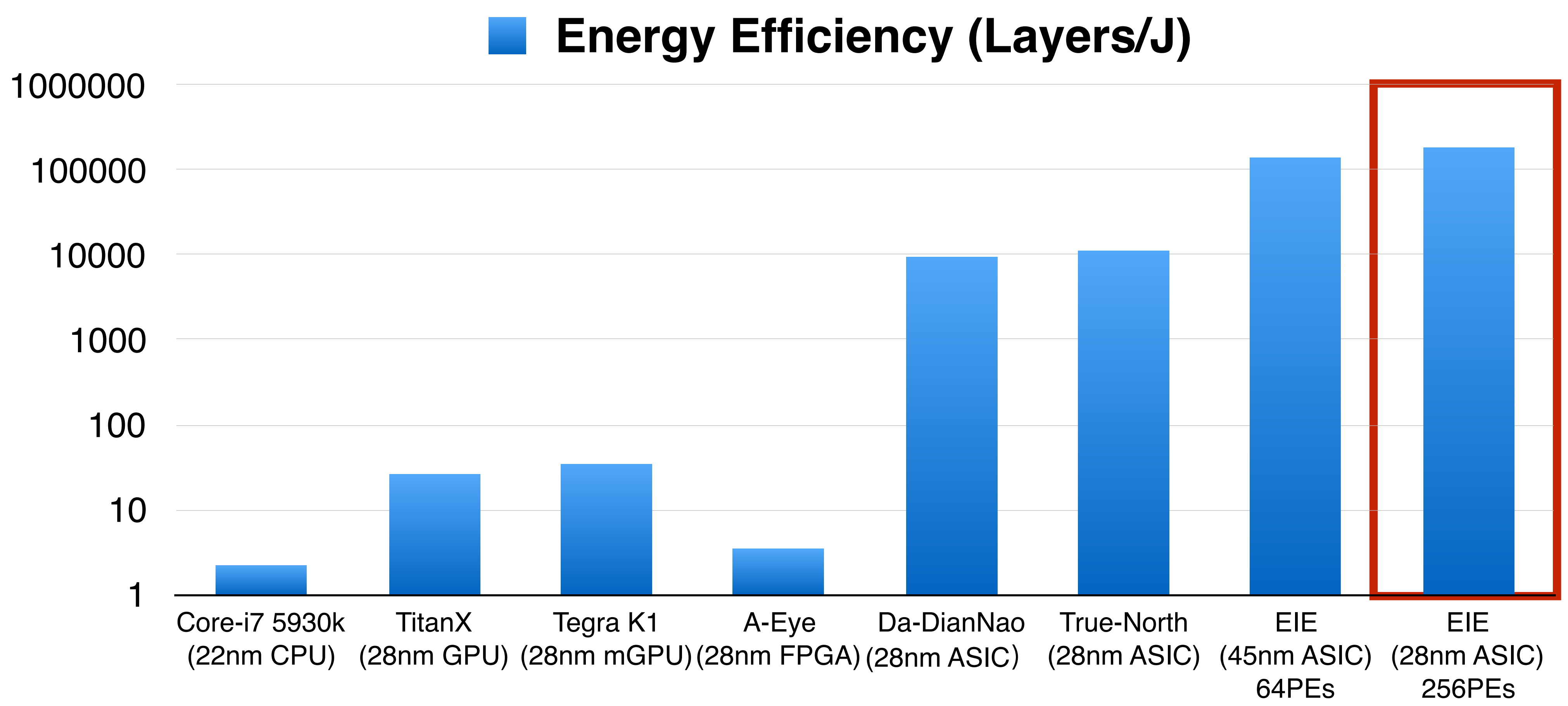
Han et al. "EIE: Efficient Inference Engine on Compressed Deep Neural Network", ISCA 2016, Hotchips 2016

EIE Architecture



Han et al. "EIE: Efficient Inference Engine on Compressed Deep Neural Network", ISCA 2016

Comparison: Energy Efficiency

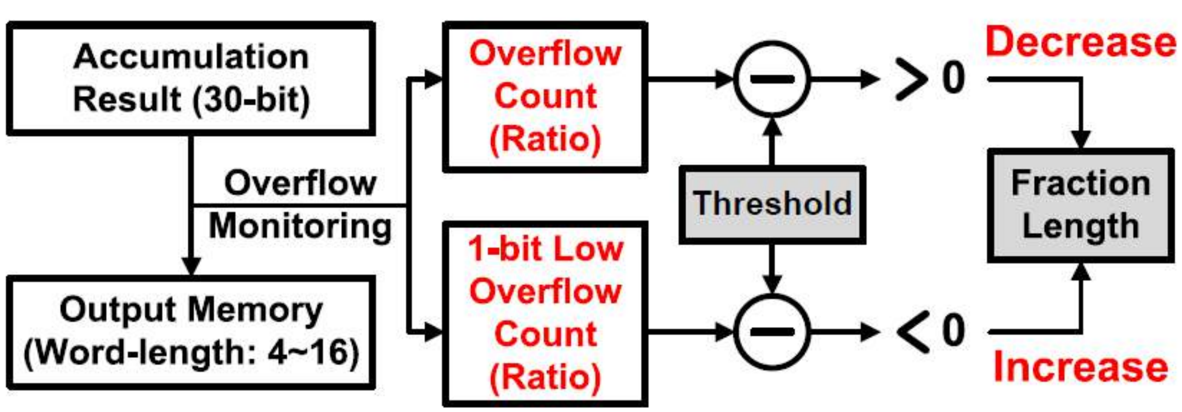


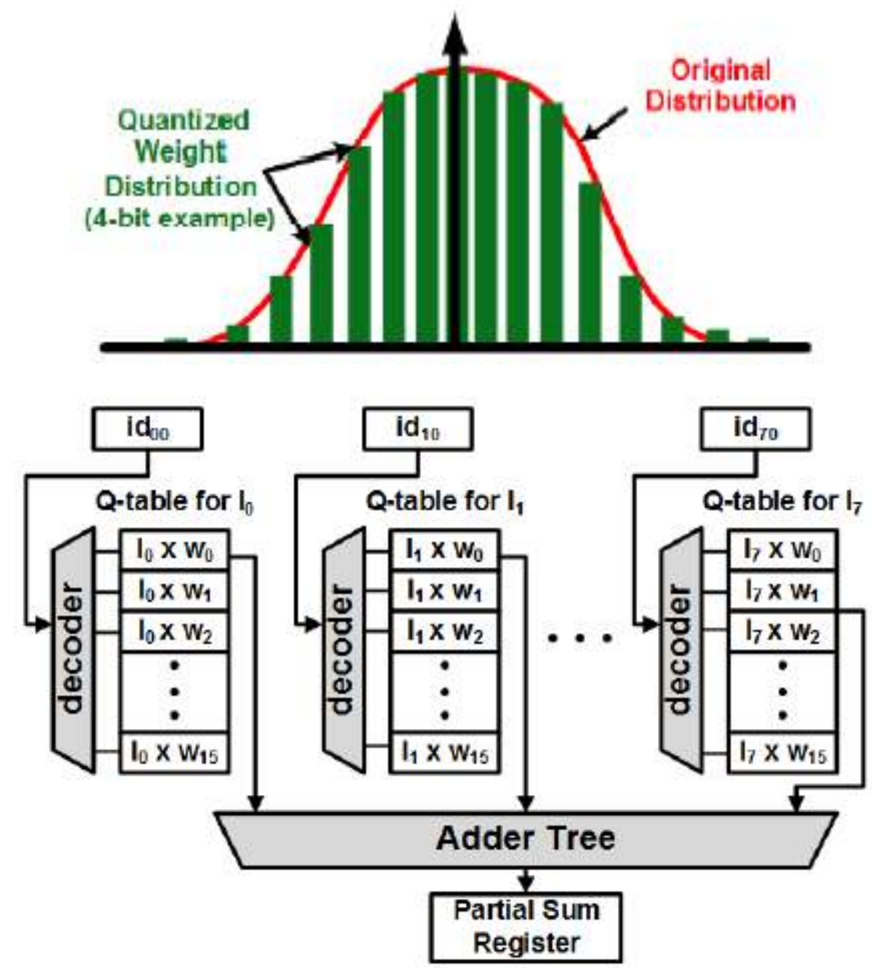
Han et al. “EIE: Efficient Inference Engine on Compressed Deep Neural Network”, ISCA 2016

Dynamic Fixed-Point + Lookup

DNPU: An 8.1TOPS/W Reconfigurable CNN-RNN Processor for General Purpose Deep Neural Networks

- Architecture: Conv accelerator + FC accelerator + RISC controller
- Mixed tiling strategy
- **Dynamic fixed-point with on-line adaptation**
- **Quantization table based multiplication**

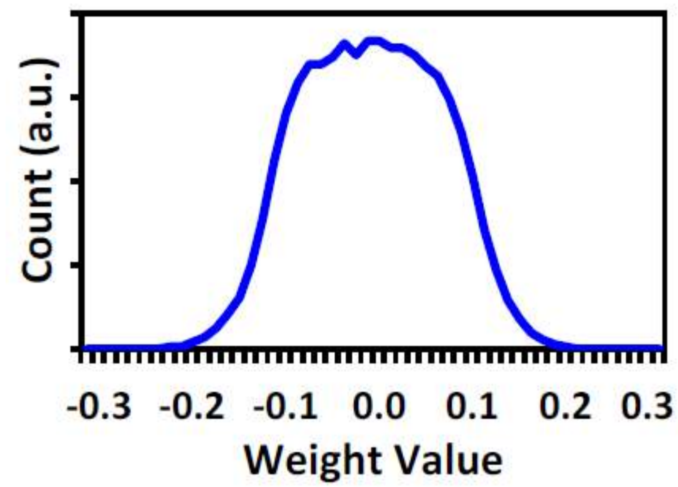


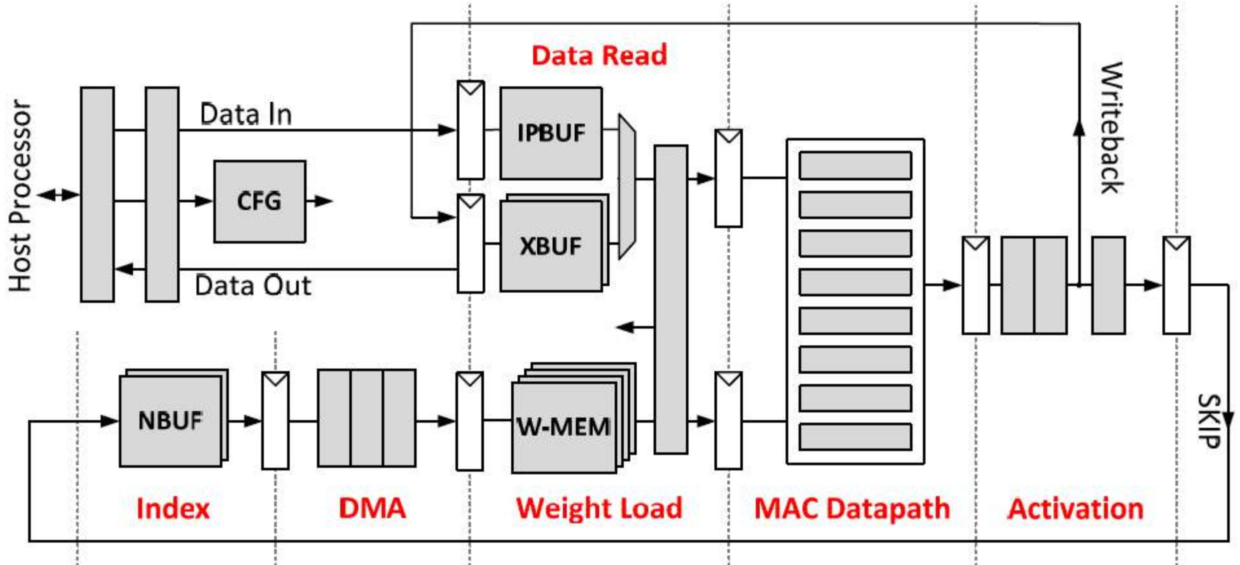


Sparse Activation + Sign-Magnitude Number Format

A 28nm SoC with a 1.2GHz 568nJ/Prediction Sparse Deep-Neural-Network with >0.1 Timing Error Rate Tolerance for IoT Applications

- HW support data sparsity
- Reduce switching using sign-magnitude number format
- Timing violation tolerant





Two's Compliment (TC)

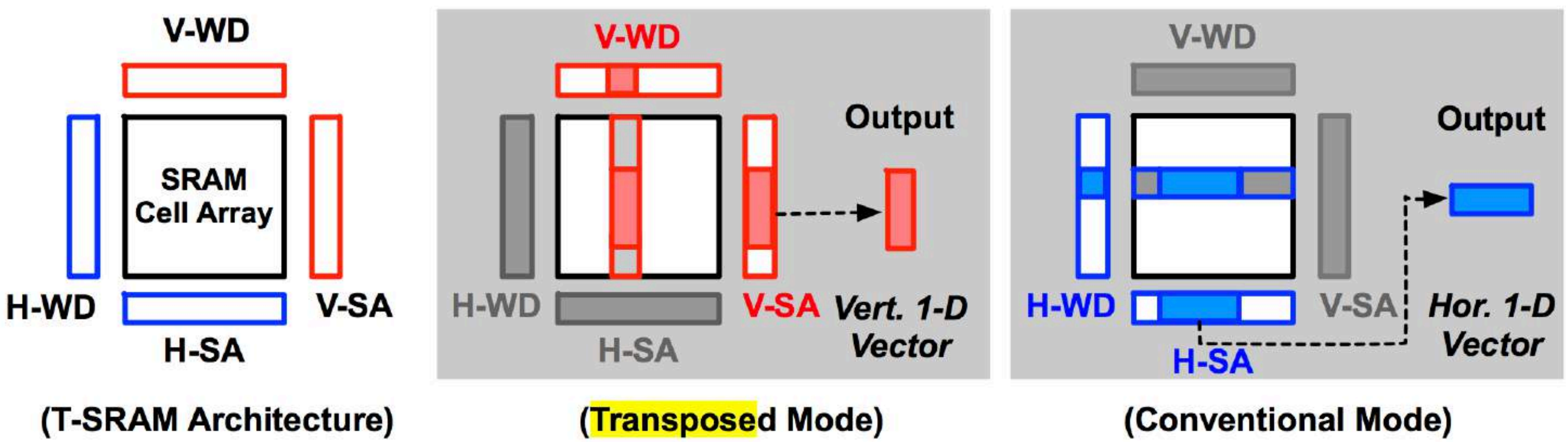
-1 : 11111111
+1 : 00000001

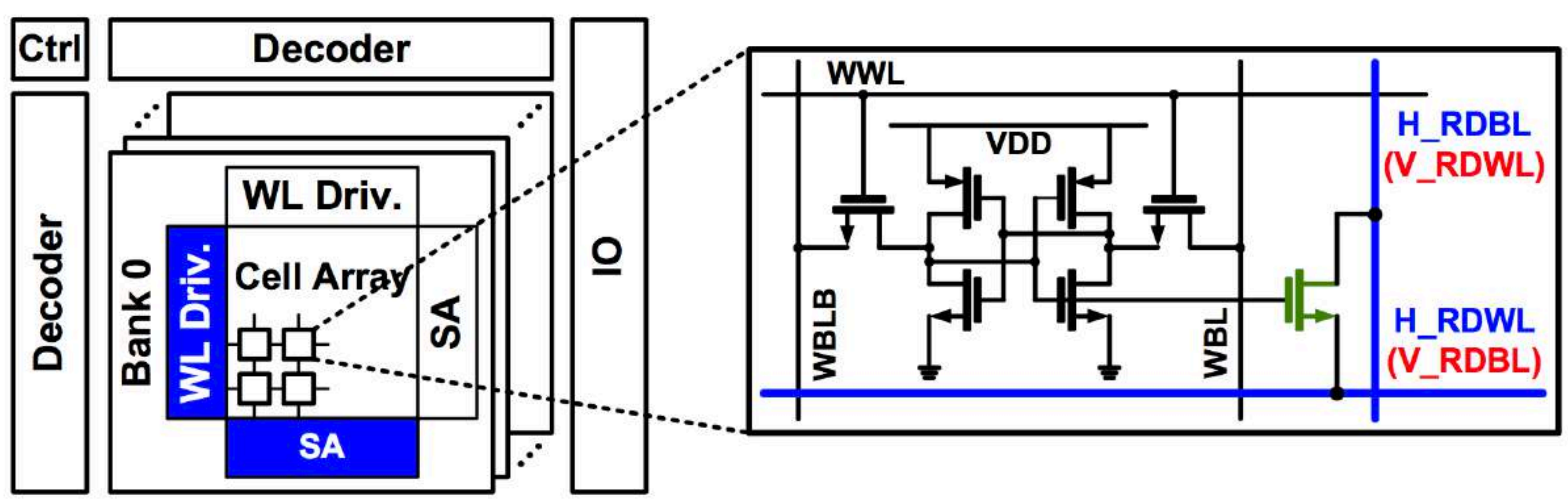
Sign-Magnitude (SM)

-1 : 10000001
+1 : 00000001

Separable Kernel + Transpose-Read SRAM

A 0.62mW Ultra-Low-Power Convolutional- Neural-Network Face-Recognition Processor and a CIS Integrated with Always-On Haar-Like Face Detector





Binary / Ternary NN Accelerator

3:30 - 3:55

Accelerating Binarized Convolutional Neural Networks with Software-Programmable FPGAs

Ritchie Zhao[1], Weinan Song[1], Wentao Zhang[1], Tianwei Xing[2], Jeng-Hau Lin[3], Mani Srivastava[2], Rajesh Gupta[3], Zhiru Zhang[1]
[1]Cornell University, [2]UCLA, [3]UCSD

9:25 - 9:50

FINN: A Framework for Fast, Scalable Binarized Neural Network Inference

Yaman Umuroglu[1,2], Nicholas J. Fraser[1,3], Giulio Gambardella[1], Michaela Blott[1], Philip Leong[3], Magnus Jahre[2], Kees Vissers[1]
[1]Xilinx Research Labs, [2]Norwegian University of Science and Technology, [3]University of Sydney

Agenda

Algorithm

Inference

Training

Hardware

	Inference	Training
Algorithm	Algorithms for Efficient Inference	Algorithms for Efficient Training
Hardware	Hardware for Efficient Inference	Hardware for Efficient Training

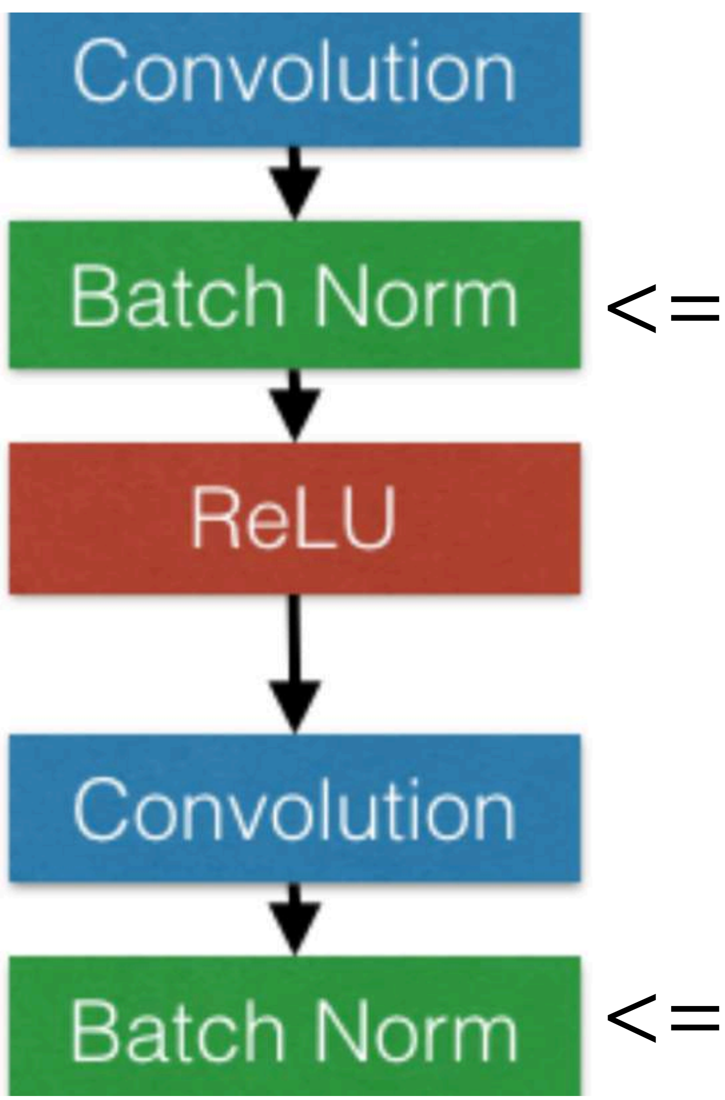
Part 3: Efficient Training Algorithm

- 1. Batch Normalization
- 2. Model Distillation
- 3. DSD: Dense-Sparse-Dense Training

Part 3: Efficient Training Algorithm

- 1. Batch Normalization
- 2. Model Distillation
- 3. DSD: Dense-Sparse-Dense Training

Batch Normalization



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

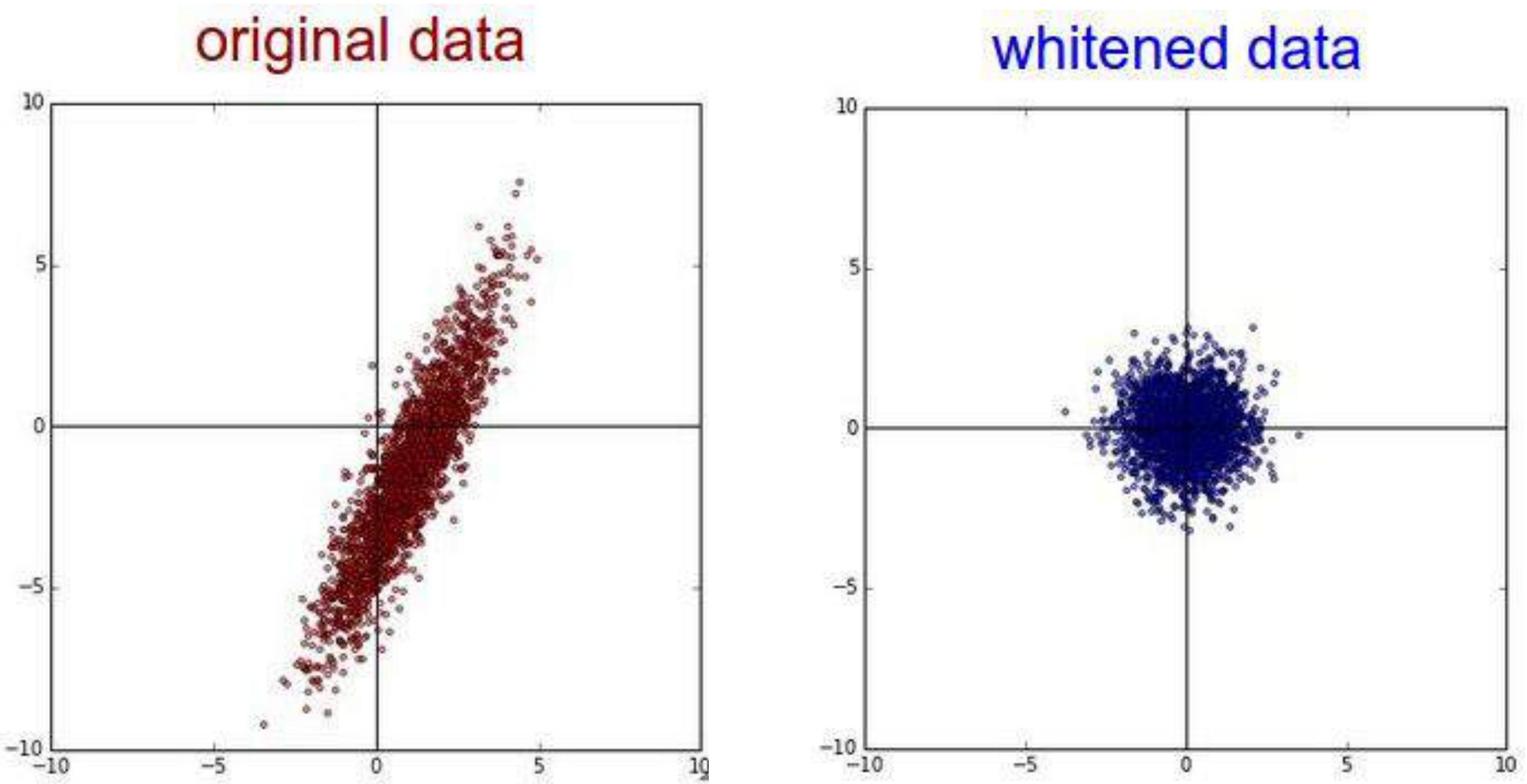
$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

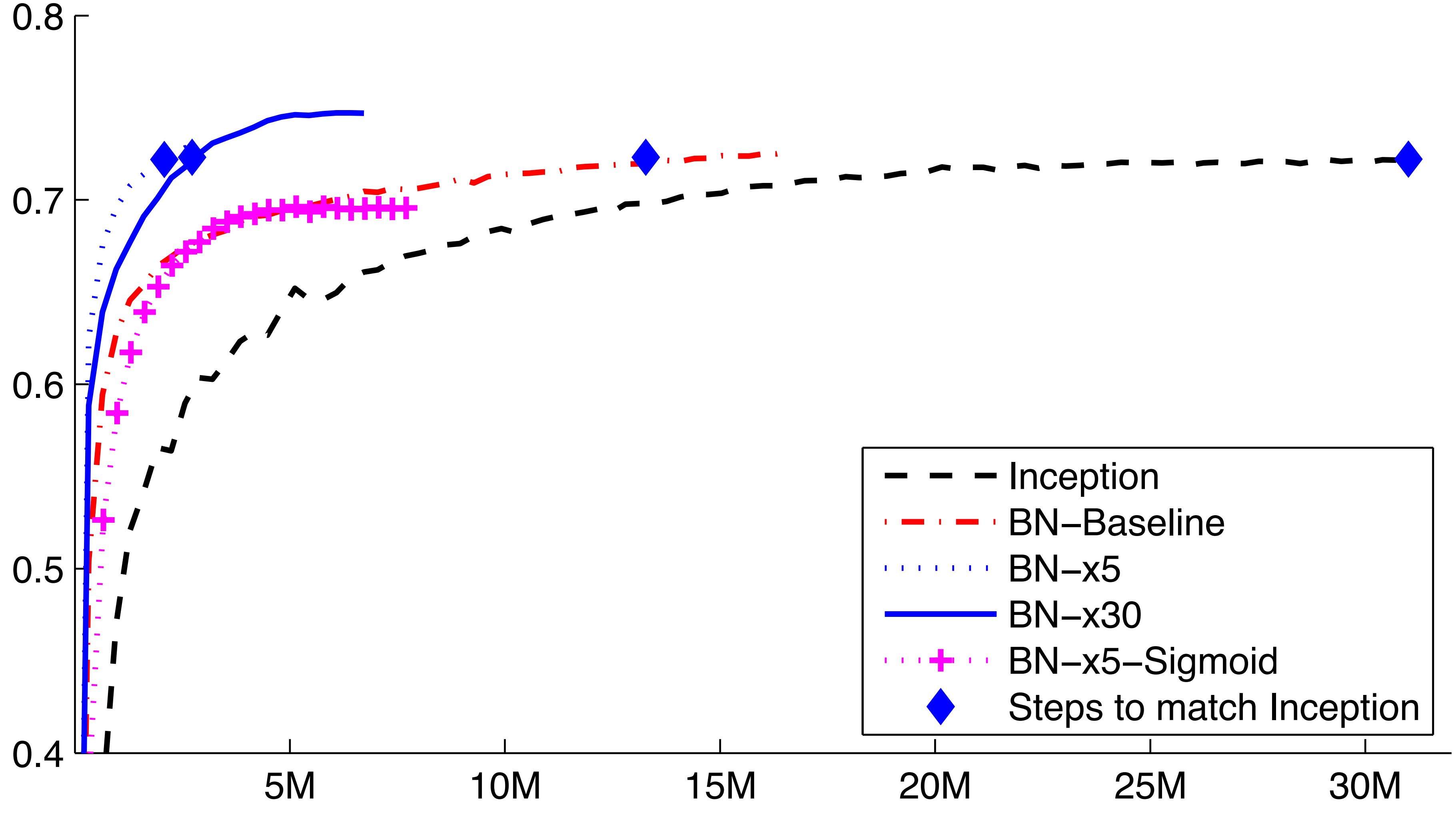
Ioffe et al. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift
http://torch.ch/blog/2016/02/04/resnets.html

Batch Normalization



Fei-Fei Li & Andrej Karpathy & Justin Johnson, Stanford CS231n course slides

Batch Normalization Helps Convergence



Ioffe et al. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Tips Using Batch Normalization

- Increase learning rate.
- Remove Dropout
- Reduce the regularization.
- Accelerate the learning rate decay.
- Remove Local Response Normalization
- Shuffle training examples
- Reduce the photometric distortions.

Ioffe et al. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Part 3: Efficient Training Algorithm

- 1. Batch Normalization
- 2. Model Distillation
- 3. DSD: Dense-Sparse-Dense Training

Softened outputs reveal the dark knowledge

cow	dog	cat	car	
0	1	0	0	original hard targets

cow	dog	cat	car	
10^{-6}	.9	.1	10^{-9}	output of geometric ensemble

cow	dog	cat	car	
.05	.3	.2	.005	softened output of ensemble

Hinton et al. Dark knowledge / Distilling the Knowledge in a Neural Network

Softened outputs reveal the dark knowledge

$$p_i = \frac{\exp\left(\frac{z_i}{T}\right)}{\sum_j \exp\left(\frac{z_j}{T}\right)}$$

- Method: Divide score by a "temperature" to get a much softer distribution

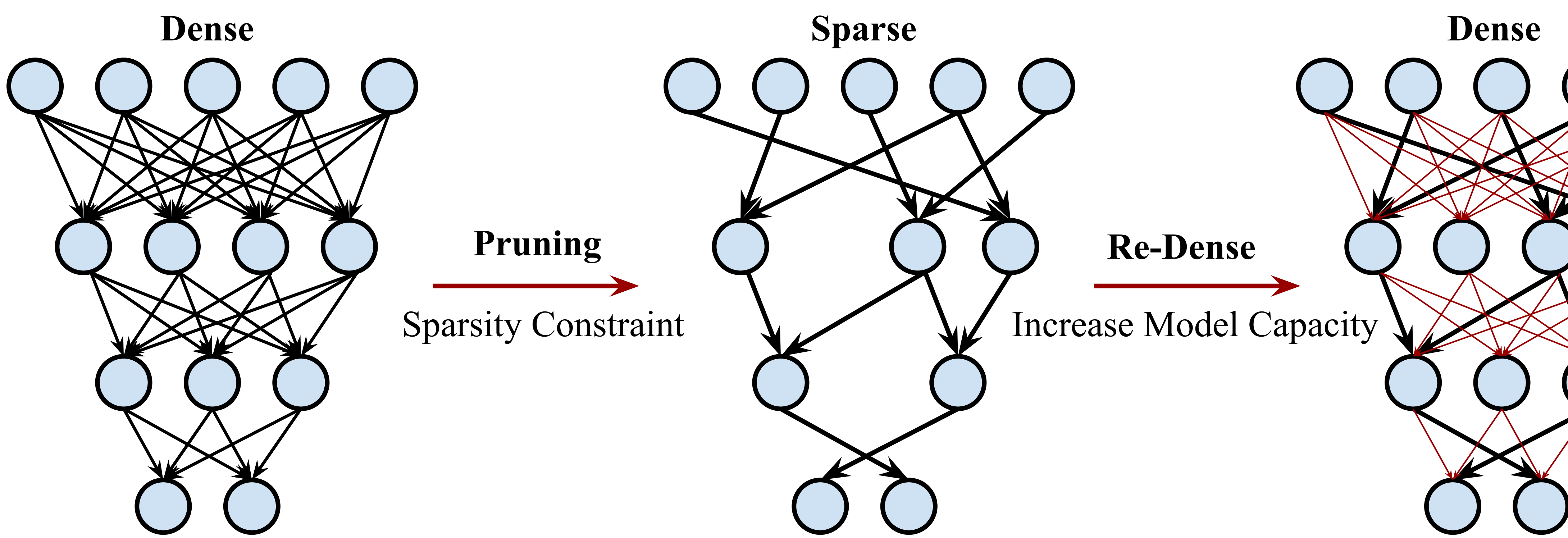
- Result: Start with a trained model that classifies 58.9% of the test frames correctly. The new model converges to 57.0% correct even when it is only trained on 3% of the data

Hinton et al. Dark knowledge / Distilling the Knowledge in a Neural Network

Part 3: Efficient Training Algorithm

- 1. Batch Normalization
- 2. Model Distillation
- 3. DSD: Dense-Sparse-Dense Training

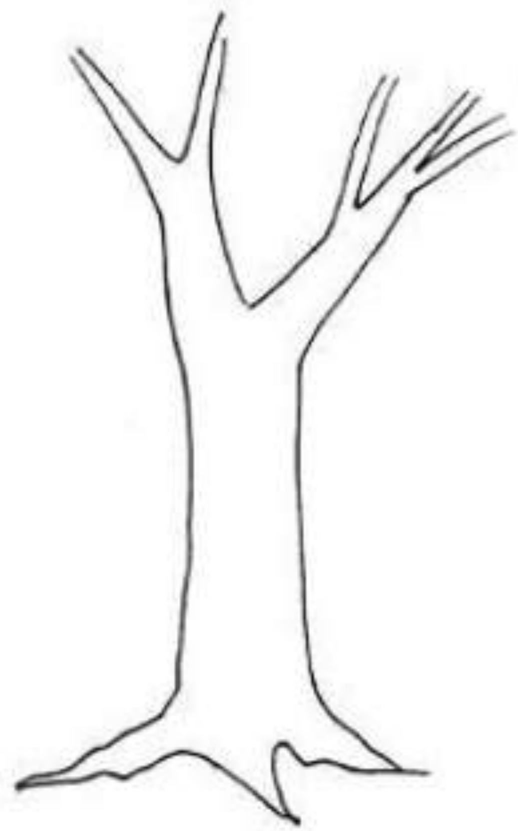
DSD: Dense Sparse Dense Training

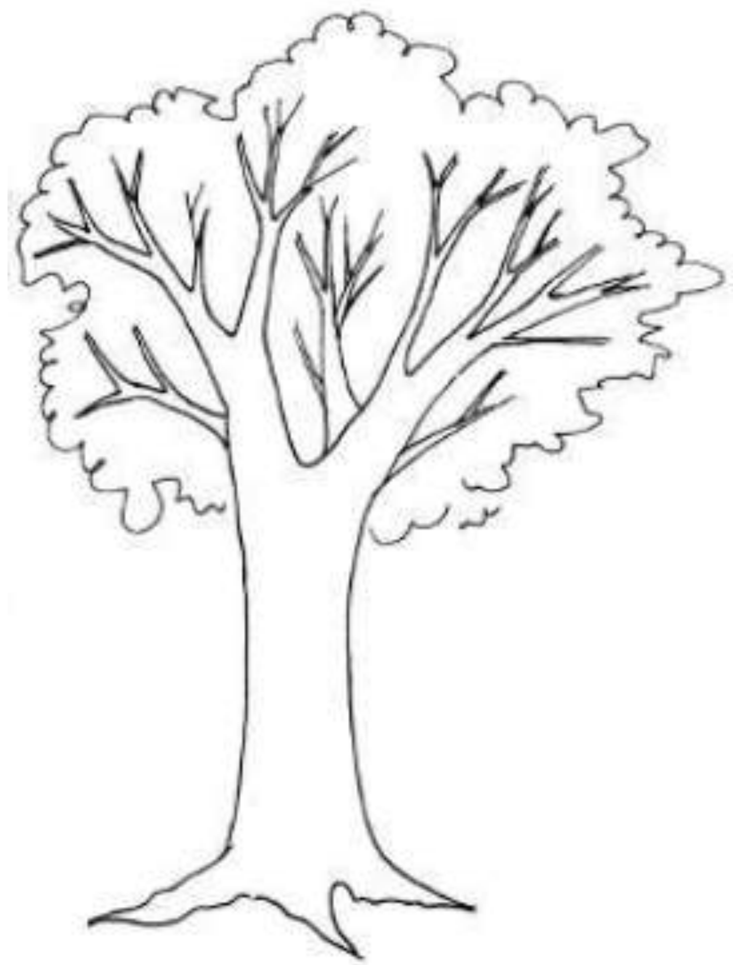


DSD produces same model architecture but can find better optimization solution, arrives at better local minima, and achieves higher prediction accuracy across a wide range of deep neural networks on CNNs / RNNs / LSTMs.

Han et al. “DSD: Dense-Sparse-Dense Training for Deep Neural Networks”, ICLR 2017

DSD: Intuition





learn the trunk first then learn the leaves

Han et al. “DSD: Dense-Sparse-Dense Training for Deep Neural Networks”, ICLR 2017

DSD: Results

Neural Network	Domain	Dataset	Type	Baseline	DSD	Abs. Imp.	Rel. Imp.
GoogLeNet	Vision	ImageNet	CNN	31.1%[1]	**30.0%**	1.1%	3.6%
VGG-16	Vision	ImageNet	CNN	31.5%[1]	**27.2%**	4.3%	13.7%
ResNet-18	Vision	ImageNet	CNN	30.4%[1]	**29.3%**	1.1%	3.7%
ResNet-50	Vision	ImageNet	CNN	24.0%[1]	**23.2%**	0.9%	3.5%

DSD Model Zoo is online: https://songhan.github.io/DSD

Han et al. "DSD: Dense-Sparse-Dense Training for Deep Neural Networks", ICLR 2017

DSD: Results

Neural Network	Domain	Dataset	Type	Baseline	DSD	Abs. Imp.	Rel. Imp.
GoogLeNet	Vision	ImageNet	CNN	31.1%[1]	**30.0%**	1.1%	3.6%
VGG-16	Vision	ImageNet	CNN	31.5%[1]	**27.2%**	4.3%	13.7%
ResNet-18	Vision	ImageNet	CNN	30.4%[1]	**29.3%**	1.1%	3.7%
ResNet-50	Vision	ImageNet	CNN	24.0%[1]	**23.2%**	0.9%	3.5%
NeuralTalk	Caption	Flickr-8K	LSTM	16.8[2]	**18.5**	1.7	10.1%

DSD Model Zoo is online: https://songhan.github.io/DSD

Han et al. "DSD: Dense-Sparse-Dense Training for Deep Neural Networks", ICLR 2017

DSD: Results

Neural Network	Domain	Dataset	Type	Baseline	DSD	Abs. Imp.	Rel. Imp.
GoogLeNet	Vision	ImageNet	CNN	31.1%[1]	**30.0%**	1.1%	3.6%
VGG-16	Vision	ImageNet	CNN	31.5%[1]	**27.2%**	4.3%	13.7%
ResNet-18	Vision	ImageNet	CNN	30.4%[1]	**29.3%**	1.1%	3.7%
ResNet-50	Vision	ImageNet	CNN	24.0%[1]	**23.2%**	0.9%	3.5%
NeuralTalk	Caption	Flickr-8K	LSTM	16.8[2]	**18.5**	1.7	10.1%
DeepSpeech	Speech	WSJ'93	RNN	33.6%[3]	**31.6%**	2.0%	5.8%
DeepSpeech-2	Speech	WSJ'93	RNN	14.5% [3]	**13.4%**	1.1%	7.4%

DSD Model Zoo is online: https://songhan.github.io/DSD

Han et al. "DSD: Dense-Sparse-Dense Training for Deep Neural Networks", ICLR 2017

DSD on Caption Generation

✗ **Baseline**: a boy in a red shirt is climbing a rock wall.

✗ **Sparse**: a young girl is jumping off a tree.

✓ **DSD**: a young girl in a pink shirt is swinging on a swing.

○ **Baseline**: a basketball player in a red uniform is playing with a ball.

○ **Sparse**: a basketball player in a blue uniform is jumping over the goal.

✓ **DSD**: a basketball player in a white uniform is trying to make a shot.

✓ **Baseline**: two dogs are playing together in a field.

✓ **Sparse**: two dogs are playing in a field.

✓ **DSD**: two dogs are playing in the grass.

✗ **Baseline**: a man and a woman are sitting on a bench.

○ **Sparse**: a man is sitting on a bench with his hands in the air.

○ **DSD**: a man is sitting on a bench with his arms folded.

✗ **Baseline**: a person in a red jacket is riding a bike through the woods.

✓ **Sparse**: a car drives through a mud puddle.

✓ **DSD**: a car drives through a forest.

Baseline model: Andrej Karpathy, Neural Talk model zoo.
Han et al. "DSD: Dense-Sparse-Dense Training for Deep Neural Networks", ICLR 2017

DSD on Caption Generation

Baseline: a boy is swimming in a pool.
Sparse: a small black dog is jumping into a pool.
DSD: a black and white dog is swimming in a pool.

Baseline: a group of people are standing in front of a building.
Sparse: a group of people are standing in front of a building.
DSD: a group of people are walking in a park.

Baseline: two girls in bathing suits are playing in the water.
Sparse: two children are playing in the sand.
DSD: two children are playing in the sand.

Baseline: a man in a red shirt and jeans is riding a bicycle down a street.
Sparse: a man in a red shirt and a woman in a wheelchair.
DSD: a man and a woman are riding on a street.

Baseline: a group of people sit on a bench in front of a building.
Sparse: a group of people are standing in front of a building.
DSD: a group of people are standing in a fountain.

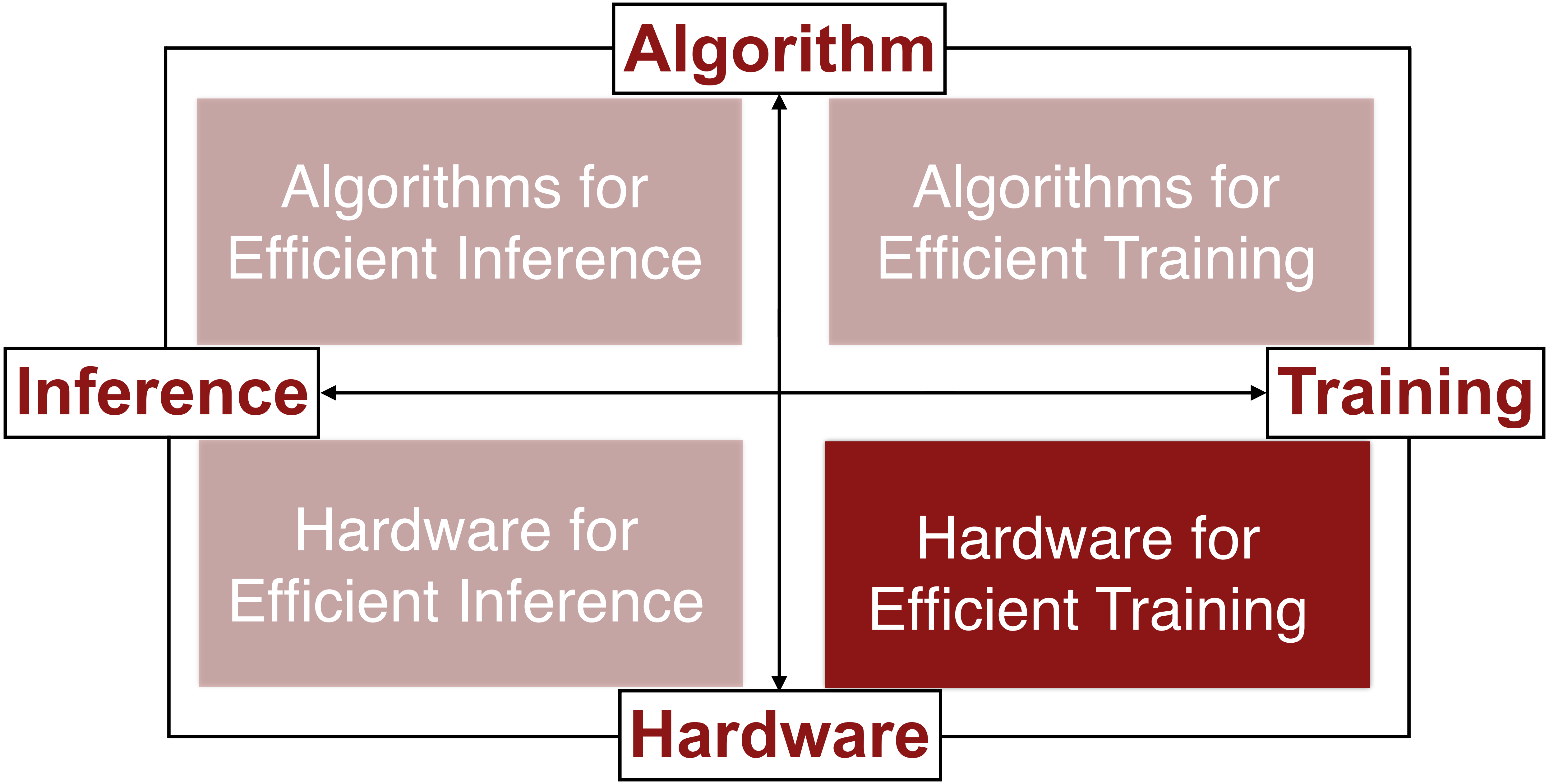
Baseline: a man in a black jacket and a black jacket is smiling.
Sparse: a man and a woman are standing in front of a mountain.
DSD: a man in a black jacket is standing next to a man in a black shirt.

Baseline: a group of football players in red uniforms.
Sparse: a group of football players in a field.
DSD: a group of football players in red and white uniforms.

Baseline: a dog runs through the grass.
Sparse: a dog runs through the grass.
DSD: a white and brown dog is running through the grass.

Baseline model: Andrej Karpathy, Neural Talk model zoo.

Agenda


Algorithm
Algorithms for Efficient Inference
Algorithms for Efficient Training
Inference
Training
Hardware for Efficient Inference
Hardware for Efficient Training
Hardware

CPUs for Training

Intel Knights Landing (2016)

- 7 TFLOPS FP32
- 16GB MCDRAM– 400 GB/s
- 245W TDP
- 29 GFLOPS/W (FP32)
- 14nm process

Knights Mill: next gen Xeon Phi “optimized for deep learning”

Intel announced the addition of new vector instructions for deep learning (AVX512-4VNNIW and AVX512-4FMAPS), October 2016

Slide Source: Sze et al Survey of DNN Hardware, MICRO’16 Tutorial.
Image Source: Intel, Data Source: Next Platform

GPUs for Training

Nvidia PASCAL GP100 (2016)

- 10/20 TFLOPS FP32/FP16
- 16GB HBM – 750 GB/s
- 300W TDP
- 67 GFLOPS/W (FP16)
- 16nm process
- 160GB/s NV Link

Slide Source: Sze et al Survey of DNN Hardware, MICRO'16 Tutorial.
Data Source: NVIDIA

GPU Systems for Training

Nvidia DGX-1 (2016)

- 170 TFLOPS
- 8× Tesla P100, Dual Xeon
- NVLink Hybrid Cube Mesh
- Optimized DL Software
- 7 TB SSD Cache
- Dual 10GbE, Quad IB 100Gb
- 3RU – 3200W

Slide Source: Sze et al Survey of DNN Hardware, MICRO'16 Tutorial.
Data Source: NVIDIA

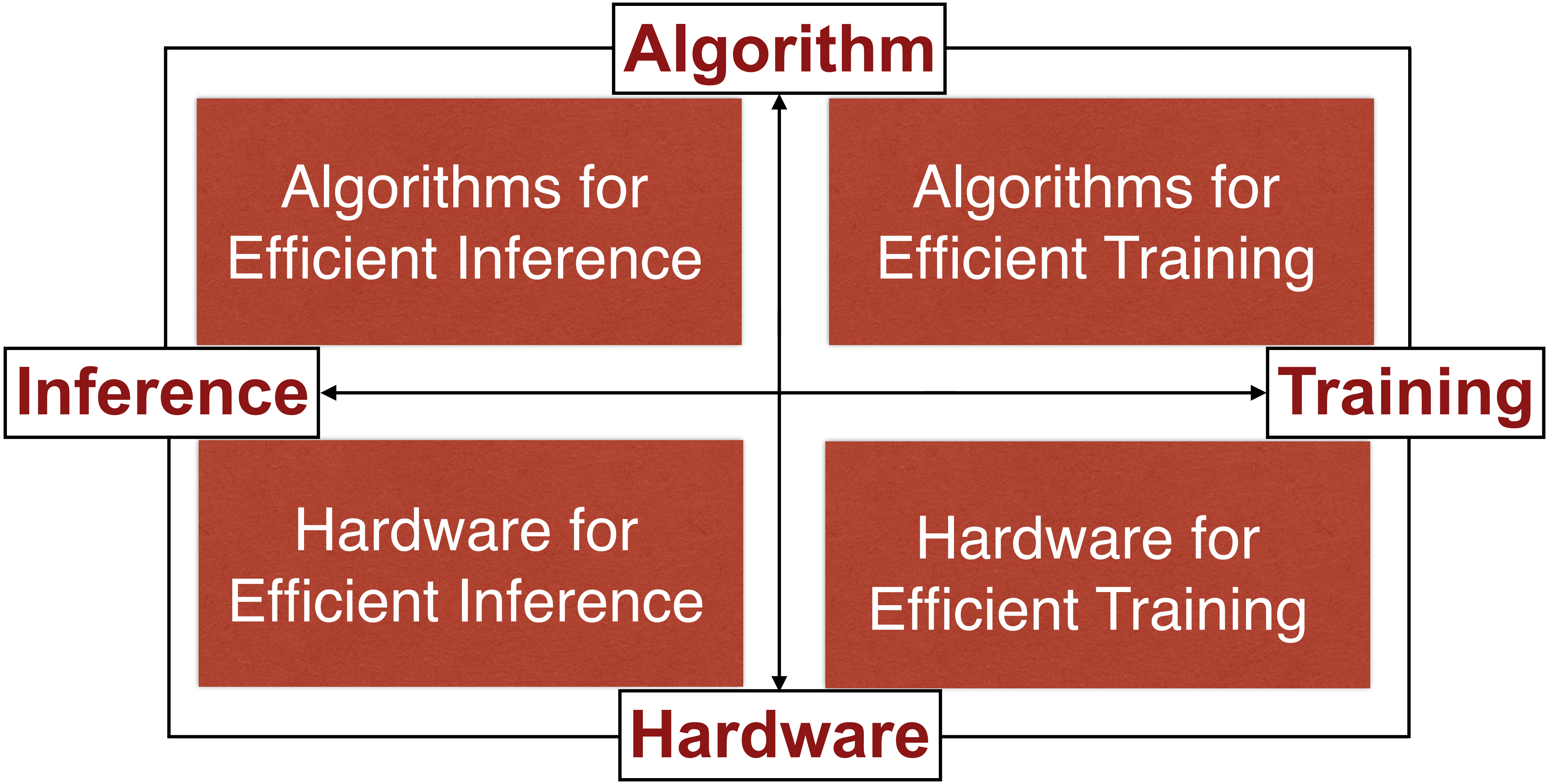
Cloud Systems for Training

Facebook Big Sur

- Open Rack Compliant
- Powered by 8 Tesla M40 GPUs
- 2x Faster Training for Faster Deployment
- 2x Larger Networks for Higher Accuracy

Slide Source: Sze et al Survey of DNN Hardware, MICRO'16 Tutorial.
Data Source: Facebook

Wrap-Up


Algorithm
Algorithms for Efficient Inference
Algorithms for Efficient Training
Inference
Training
Hardware for Efficient Inference
Hardware for Efficient Training
Hardware

Wrap-Up

Algorithm

Inference

1. Pruning
2. Weight Sharing
3. Quantization
4. Low Rank Approximation
5. Binary / Ternary Net
6. Winograd Transformation

Training

1. Batch Normalization
2. Model Distillation
3. DSD Training

Hardware

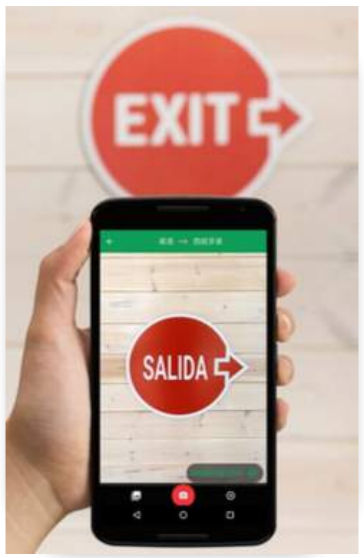
Inference

1. Saves BW by Better Data Flow
2. Saves BW by Compression
3. Replace Mul with Lookup
4. Separable Kernel + Transpose SRAM
5. Binary / Ternary NN Accelerator

Training

1. CPU
2. GPU
3. GPU Cloud

Future: Intelligence on Mobile



Phones

Drones

Robots

Glasses

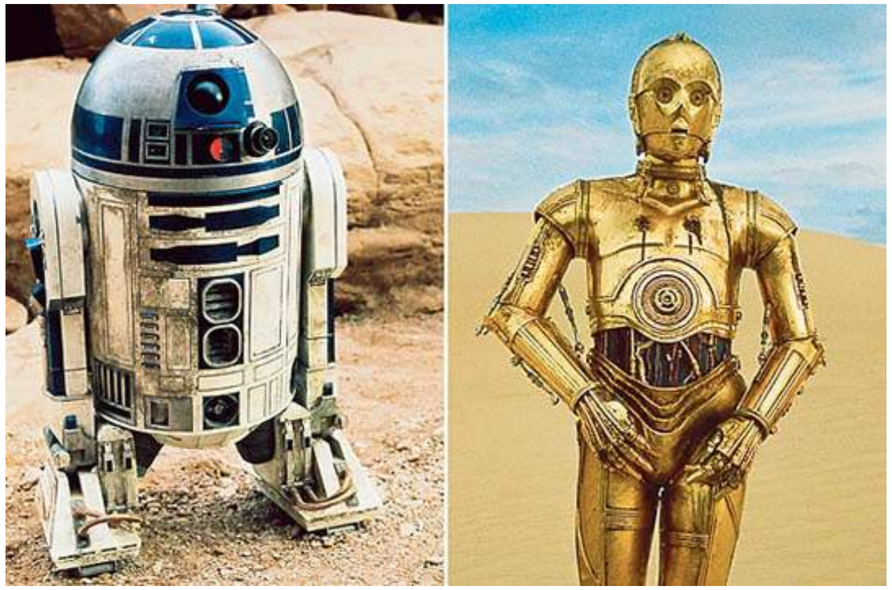
Self Driving Cars

Limited Resource
Battery Constrained
Cooling Constrained

Outlook: the Path for Computation

PC

Mobile-First



AI-First

Computation → Mobile Computation → Brain-Inspired Intelligent Computation

Sundar Pichai, Google IO, 2016

Thank you!

stanford.edu/~songhan