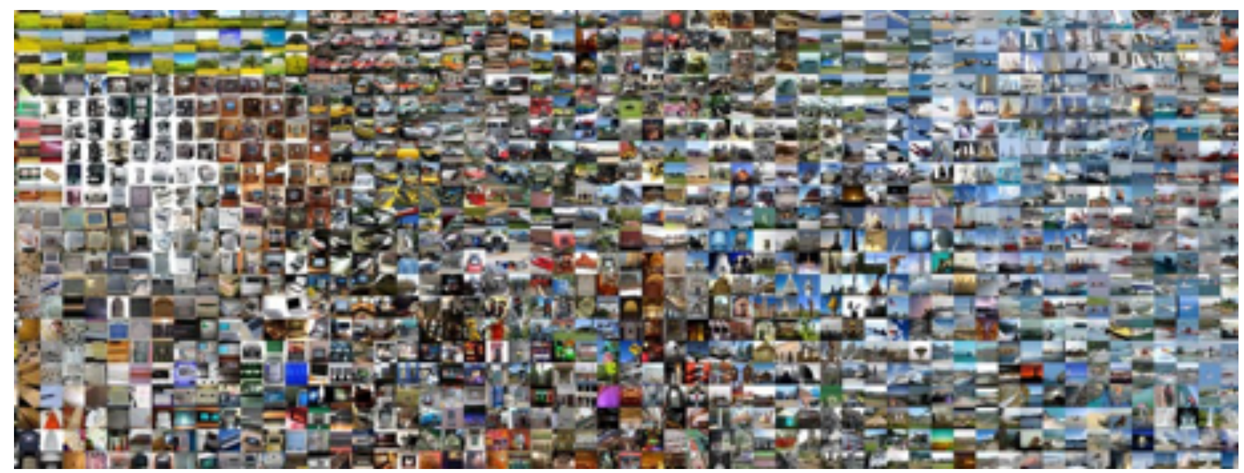
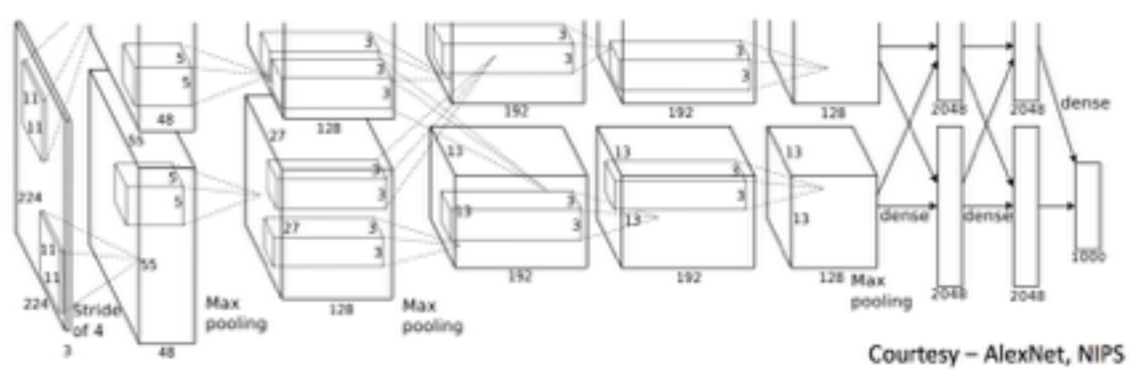
# Efficient Methods for Deep Learning

**Song Han**

**Stanford University**
Sep 2016

# Hardware and Data enable Deep Learning




Courtesy – AlexNet, NIPS 2012

Dally, NIPS'2015 tutorial on High-Performance Hardware for Machine Learning

# The Need for Speed

More data → Bigger Models →
More Need for Compute

But Moore's law is no longer providing more compute…

Dally, NIPS'2015 tutorial on High-Performance Hardware for Machine Learning

# Goal: Improve the Efficiency of Deep Learning

## For Mobile + Cloud

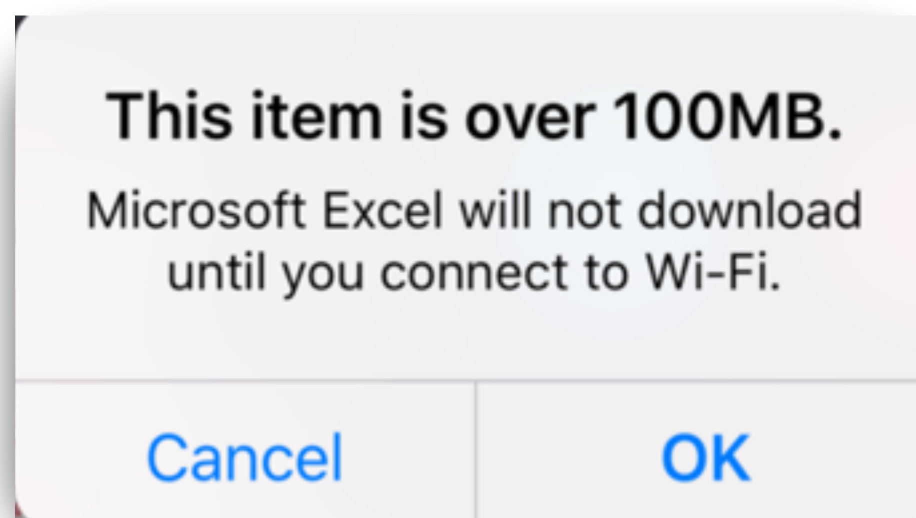
# Embedded Applications: Self-Driving Cars

nVidia Drive PX2

24 Tps/sec @ 20W

# Challenges for Efficient Deep Learning

## Model Size!



# Challenges for Efficient Deep Learning

| 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** |

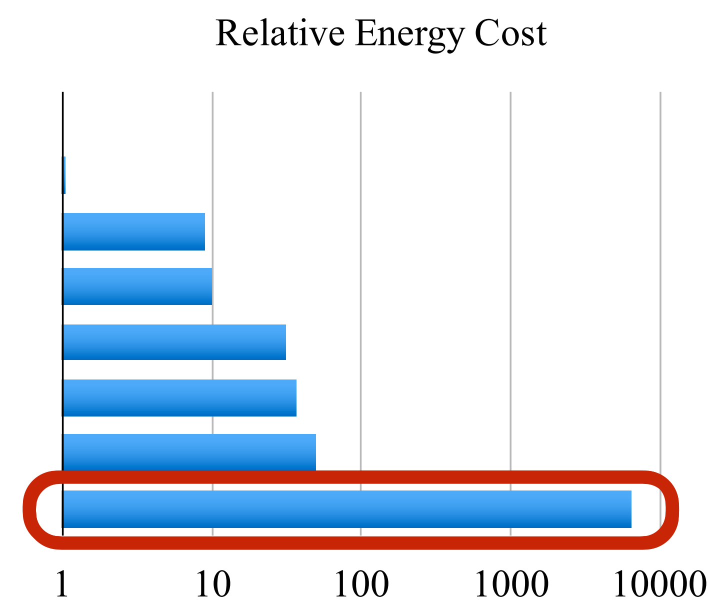
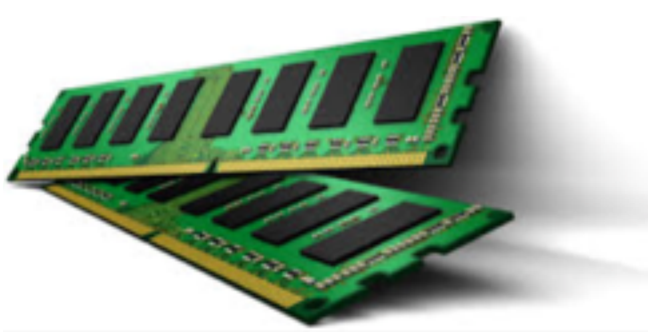



Figure 1: Energy table for 45nm CMOS process. Memory access is 2 orders of magnitude more energy expensive than arithmetic operations.

1  = 100 × +

# Part 1: Deep Compression

Song Han
CVA group, Stanford University

Han et al. “Learning both Weights and Connections for Efficient Neural Networks”, NIPS’15
Han et al. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, ICLR 2016

# Deep Compression

**Problem 1: DNN Model Size too Large**
**Solution 1: Deep Compression**

**Problem 1: DNN Model Size too Large**
**Solution 1: Deep Compression**

# Deep Compression

**Problem 1: DNN Model Size too Large**
**Solution 1: Deep Compression**

**Smaller Size**

90% zeros in weights
4-bit weight

# Deep Compression

**Problem 1: DNN Model Size too Large**
**Solution 1: Deep Compression**

### Smaller Size
90% zeros in weights
4-bit weight

### Accuracy
No loss of accuracy /
Improved accuracy

# Deep Compression

**Problem 1: DNN Model Size too Large**
**Solution 1: Deep Compression**

### Smaller Size
90% zeros in weights
4-bit weight

### Accuracy
No loss of accuracy /
Improved accuracy

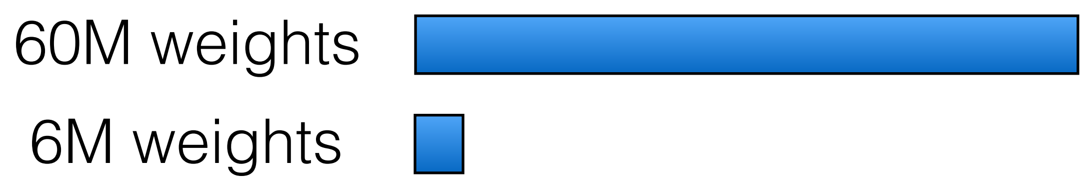
### On-chip
State-of-the-art DNN
fit on-chip SRAM

# Deep Compression Overview

- AlexNet: 35×, 240MB => 6.9MB
- VGG16: 49×, 552MB => 11.3MB
- GoogLeNet: 10x, 28MB => 2.8MB
- SqueezeNet: 10x, 4.8MB => 0.47MB
- No loss of accuracy on ImageNet12
- Weights fits on-chip SRAM cache, taking 120x less energy than DRAM memory

# Deep Compression Pipeline

- **Network Pruning**:
  10x fewer weights



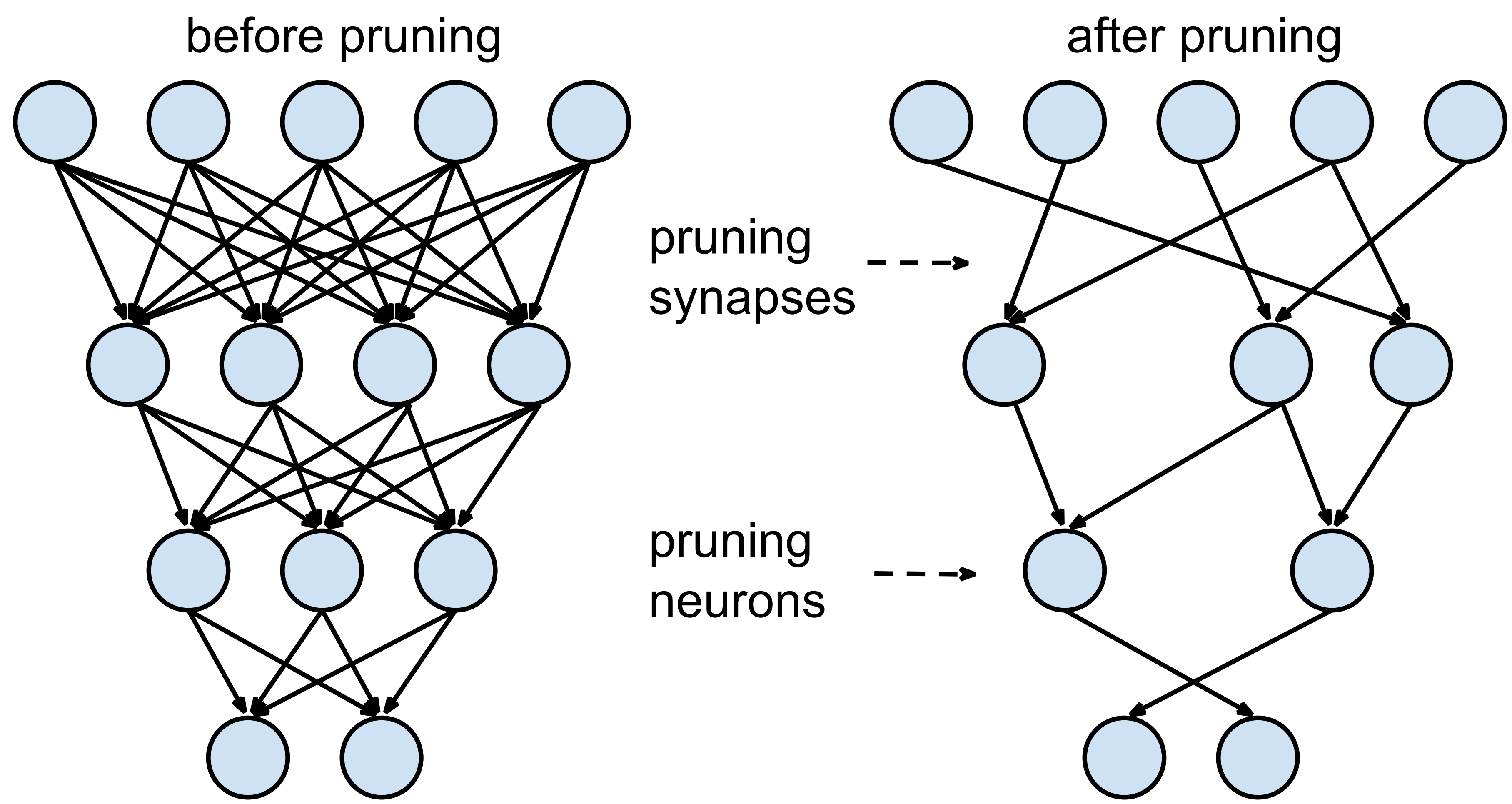

- **Weight Sharing**:
  only 4-bits per remaining weight


- **Huffman Coding**:
  Entropy of the Total Remaining Weights

# Deep Compression Pipeline

- **Network Pruning**:
  Less Number of Weights
- **Weight Sharing**:
  Reduce Storage for Each Remaining Weight
- **Huffman Coding**:
  Entropy of the Total Remaining Weights

# 1. Pruning





[1] LeCun et al. Optimal Brain Damage NIPS'90
[2] Hassibi, et al. Second order derivatives for network pruning: Optimal brain surgeon. NIPS'93
[3] Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS'15

# Pruning: Motivation

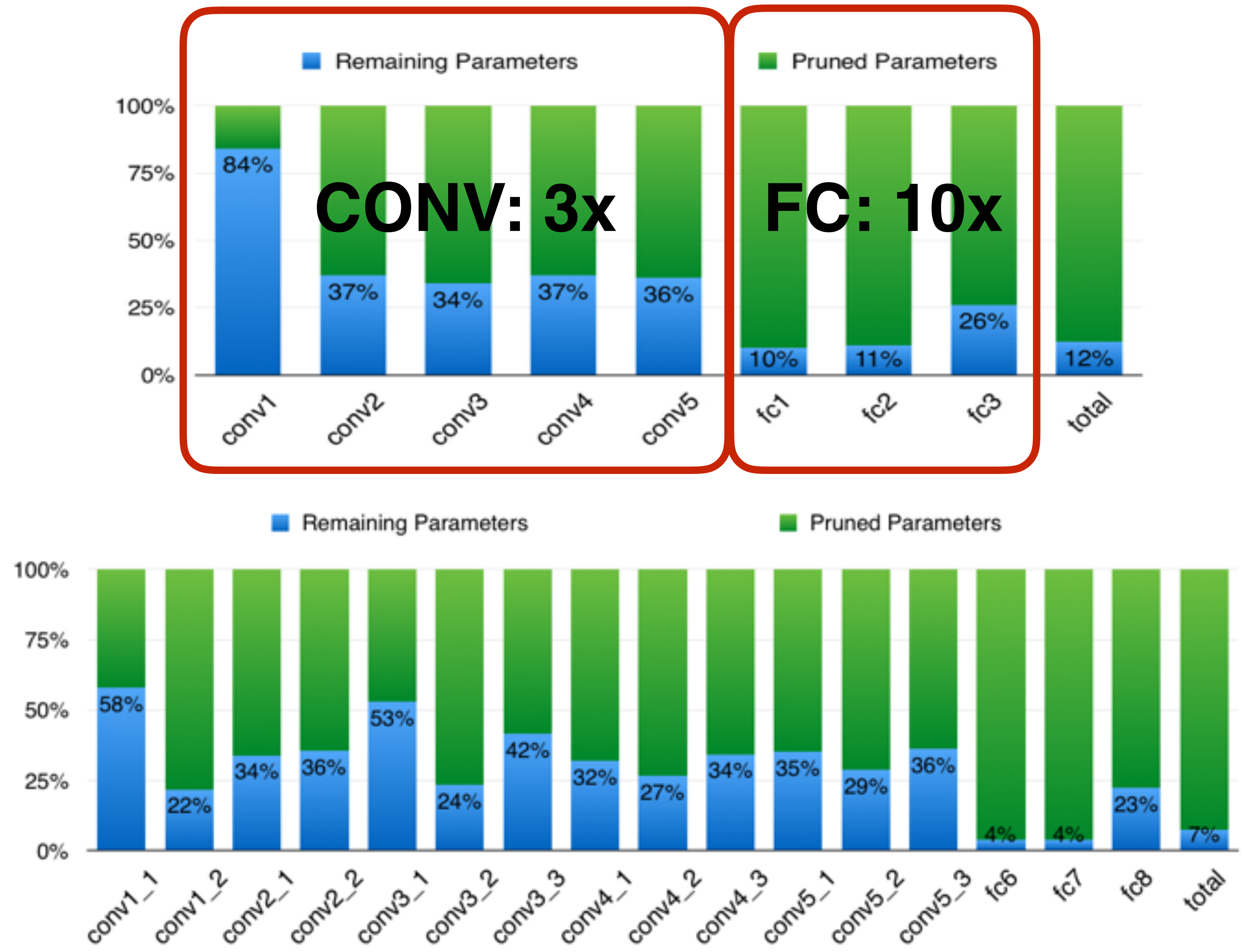
| Age | Number of Connections | Stage |
|---|---|---|
| at birth | 50 Trillion | newly formed |
| 1 year old | 1000 Trillion | peak |
| 10 year old | 500 Trillion | pruned and stabilized |

Table 1: The synapses pruning mechanism in human brain development

- Trillion of synapses are generated in the human brain during the first few months of birth.
- **1 year old**, peaked at **1000 trillion**
- Pruning begins to occur.
- **10 years old**, a child has nearly **500 trillion** synapses
- This 'pruning' mechanism removes redundant connections in the brain.

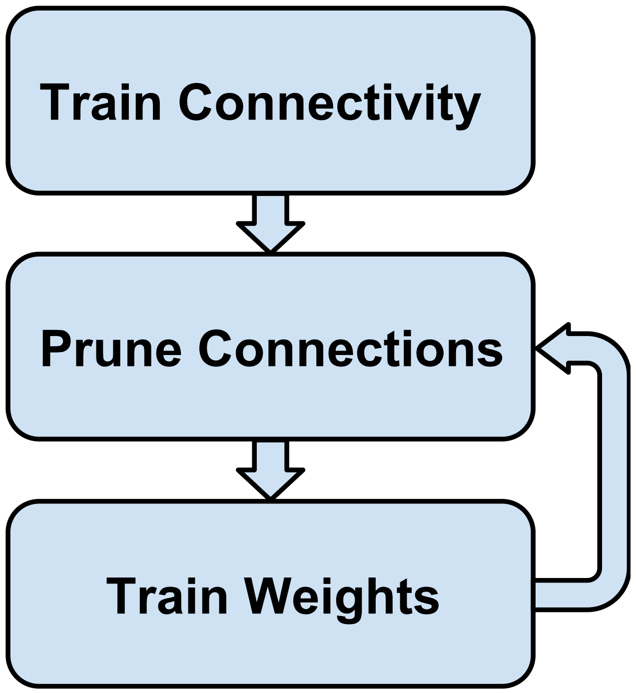
[1] Christopher A Walsh. Peter Huttenlocher (1931-2013). Nature, 502(7470):172–172, 2013.

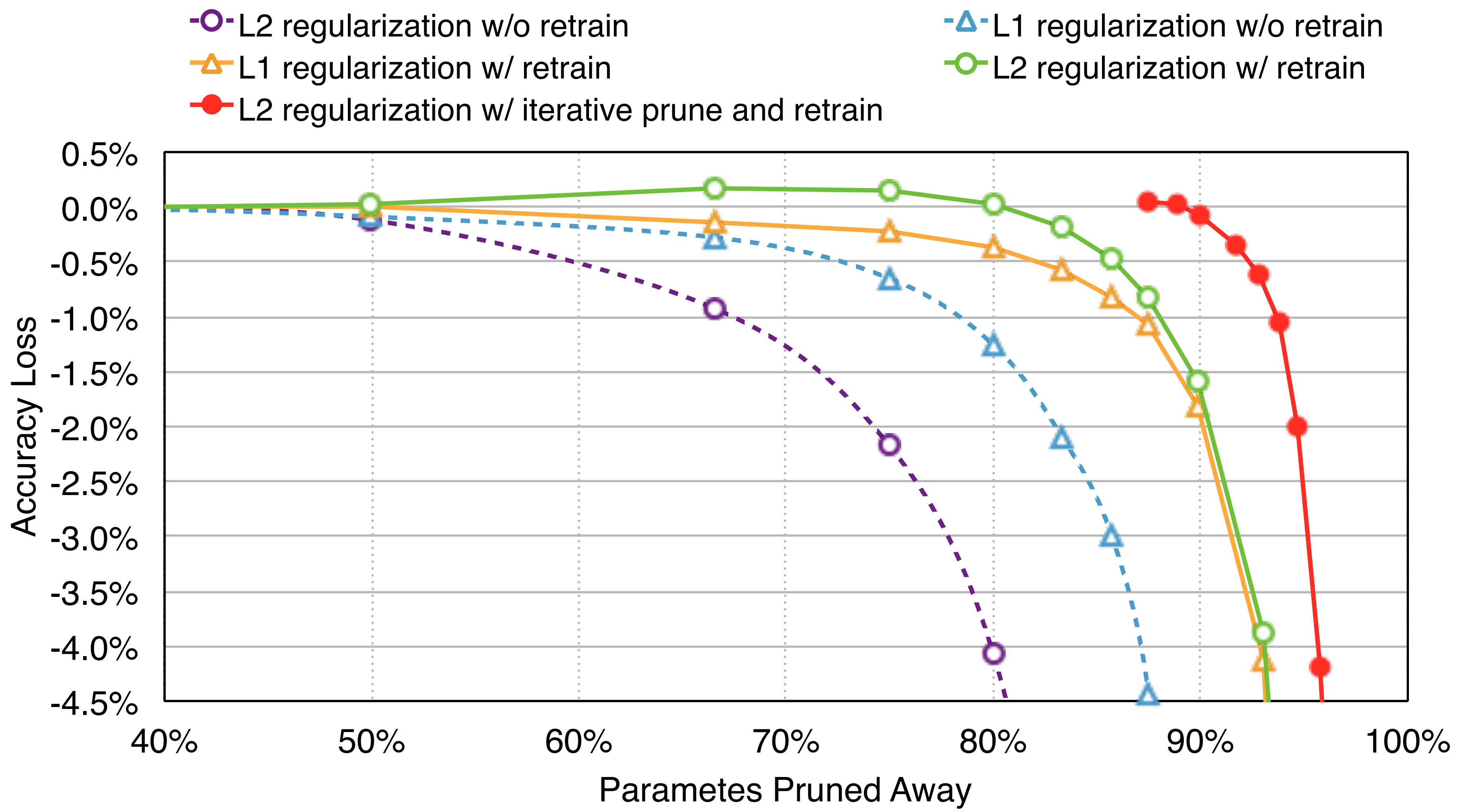
# AlexNet & VGGNet




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

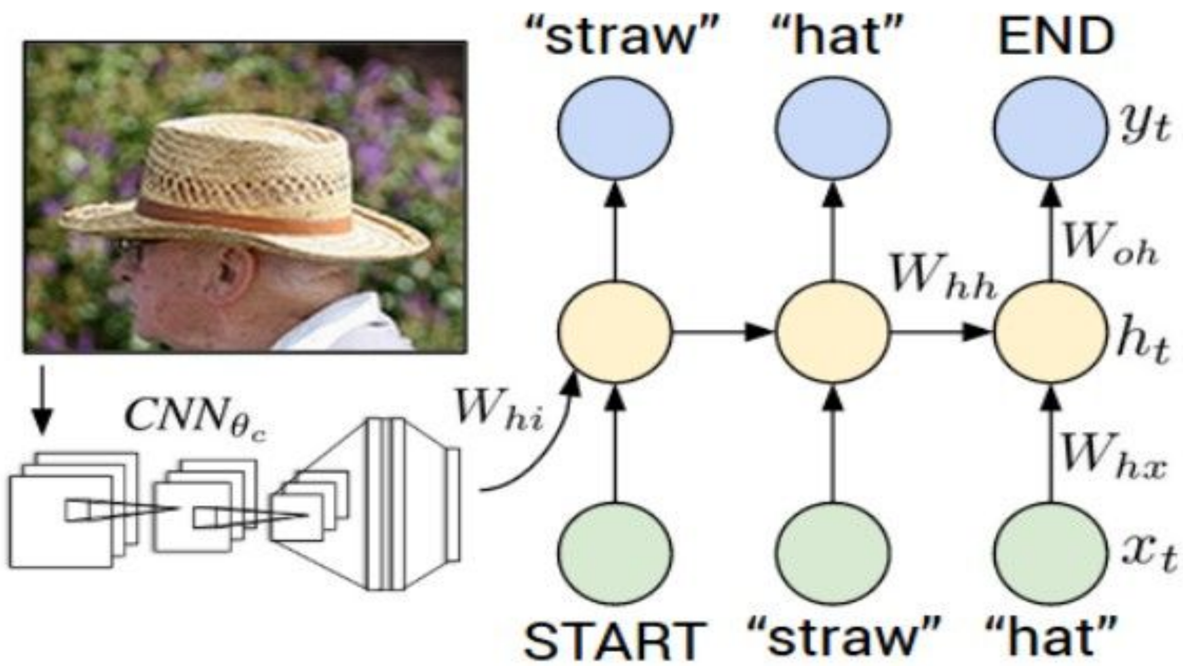
# Pruning: Result

| 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**$\times$ |
| VGG16 Ref | 31.50% | 11.32% | 138M | |
| VGG16 Pruned | 31.34% | 10.88% | **10.3M** | **13**$\times$ |

Table 1: Network pruning can save $9\times$ to $13\times$ parameters with no drop in predictive performance

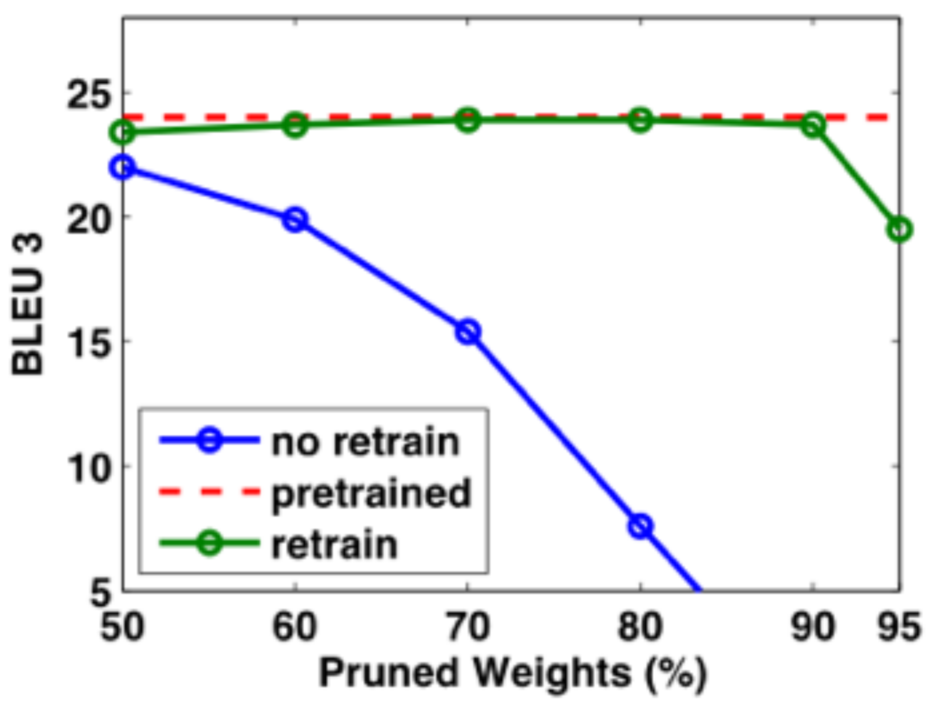
Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS 2015

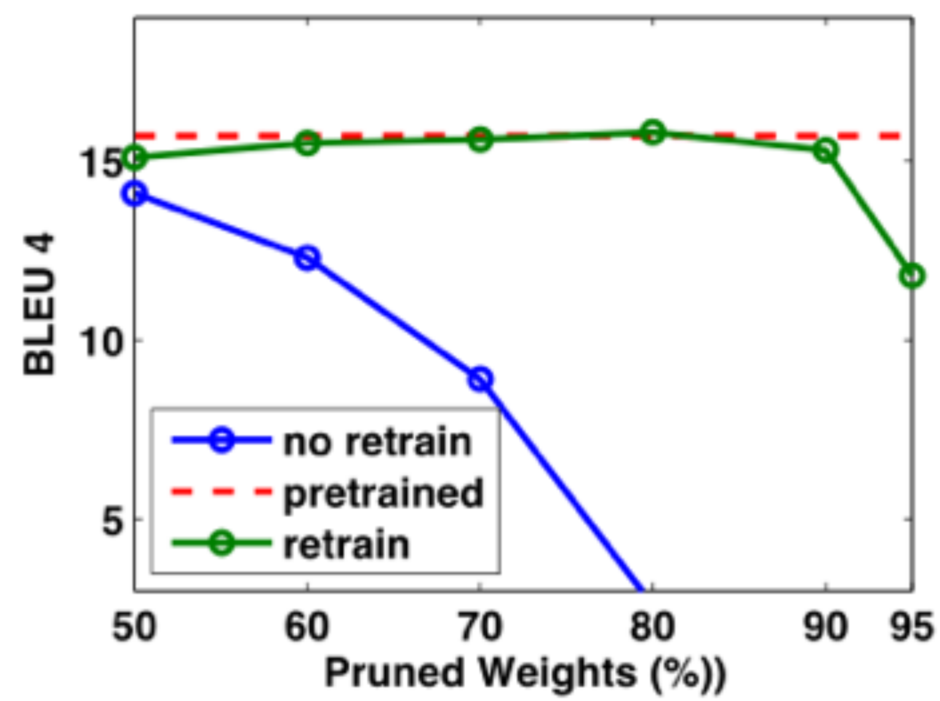
# Pruning RNN and LSTM




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

- Pruning away 90% parameters in NeuralTalk doesn't hurt BLUE score with proper retrain



# 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

# Deep Compression Pipeline

- **Network Pruning**:
  Less Number of Weights
- **Weight Sharing**:
  Reduce Storage for Each Remaining Weight
- **Huffman Coding**:
  Entropy of the Total Remaining Weights

# Weight Sharing: Overview

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: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, ICLR 2016

# Weight Sharing: Overview

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 |

cluster

centroids

| | |
|---|---|
| 3: | 2.00 |
| 2: | 1.50 |
| 1: | 0.00 |
| 0: | -1.00 |

Han et al. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, ICLR 2016

# Weight Sharing: Overview

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 |

cluster

cluster index
(2 bit uint)

| | | | |
|---|---|---|---|
| 3 | 0 | 2 | 1 |
| 1 | 1 | 0 | 3 |
| 0 | 3 | 1 | 0 |
| 3 | 1 | 2 | 2 |

centroids

| | |
|---|---|
| 3: | 2.00 |
| 2: | 1.50 |
| 1: | 0.00 |
| 0: | -1.00 |

Han et al. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, ICLR 2016

# Weight Sharing: Overview

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 |

cluster

cluster index
(2 bit uint)

| | | | |
|---|---|---|---|
| 3 | 0 | 2 | 1 |
| 1 | 1 | 0 | 3 |
| 0 | 3 | 1 | 0 |
| 3 | 1 | 2 | 2 |

centroids

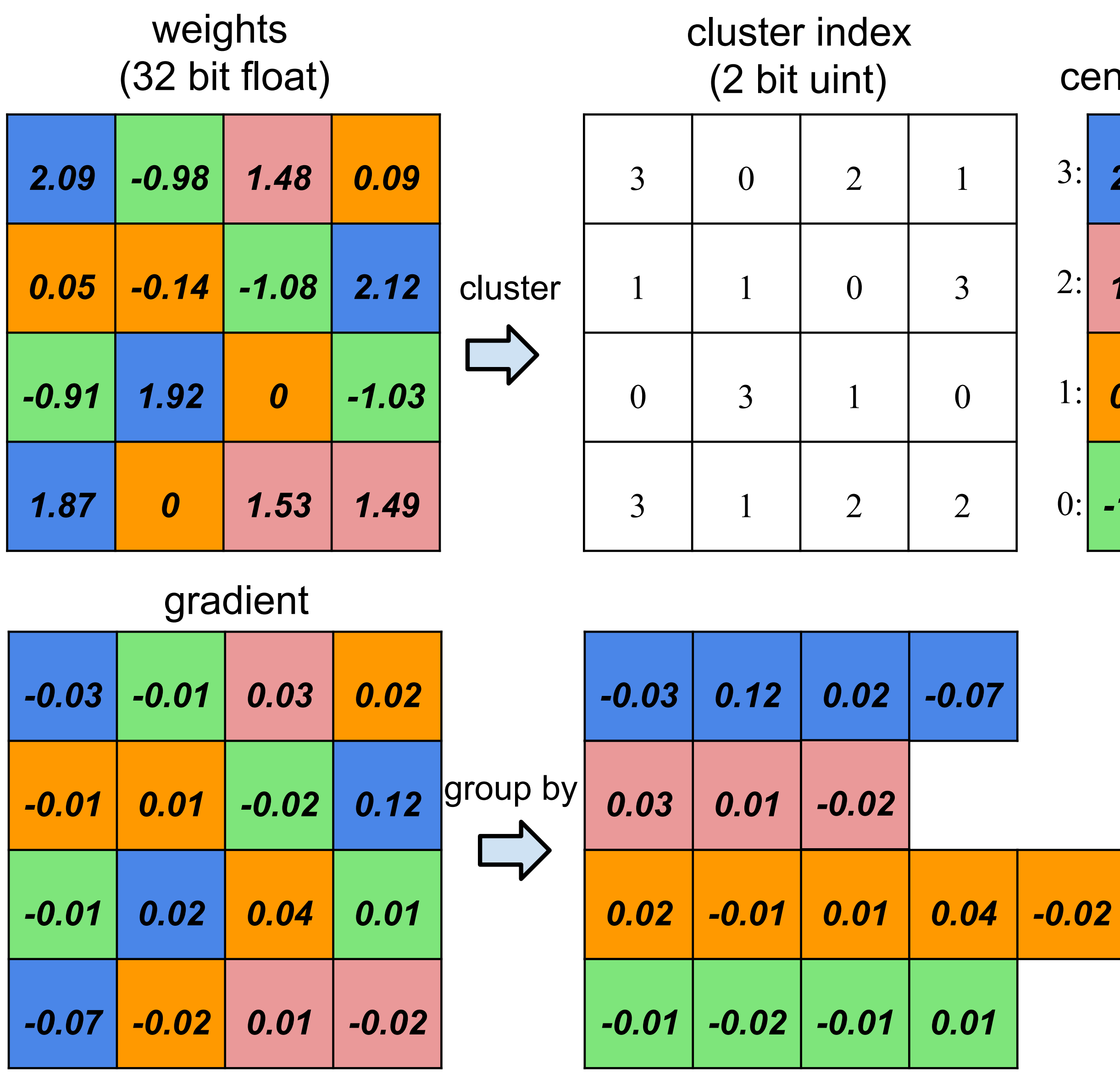
| | |
|---|---|
| 3: | 2.00 |
| 2: | 1.50 |
| 1: | 0.00 |
| 0: | -1.00 |

gradient

| | | | |
|---|---|---|---|
| -0.03 | -0.01 | 0.03 | 0.02 |
| -0.01 | 0.01 | -0.02 | 0.12 |
| -0.01 | 0.02 | 0.04 | 0.01 |
| -0.07 | -0.02 | 0.01 | -0.02 |

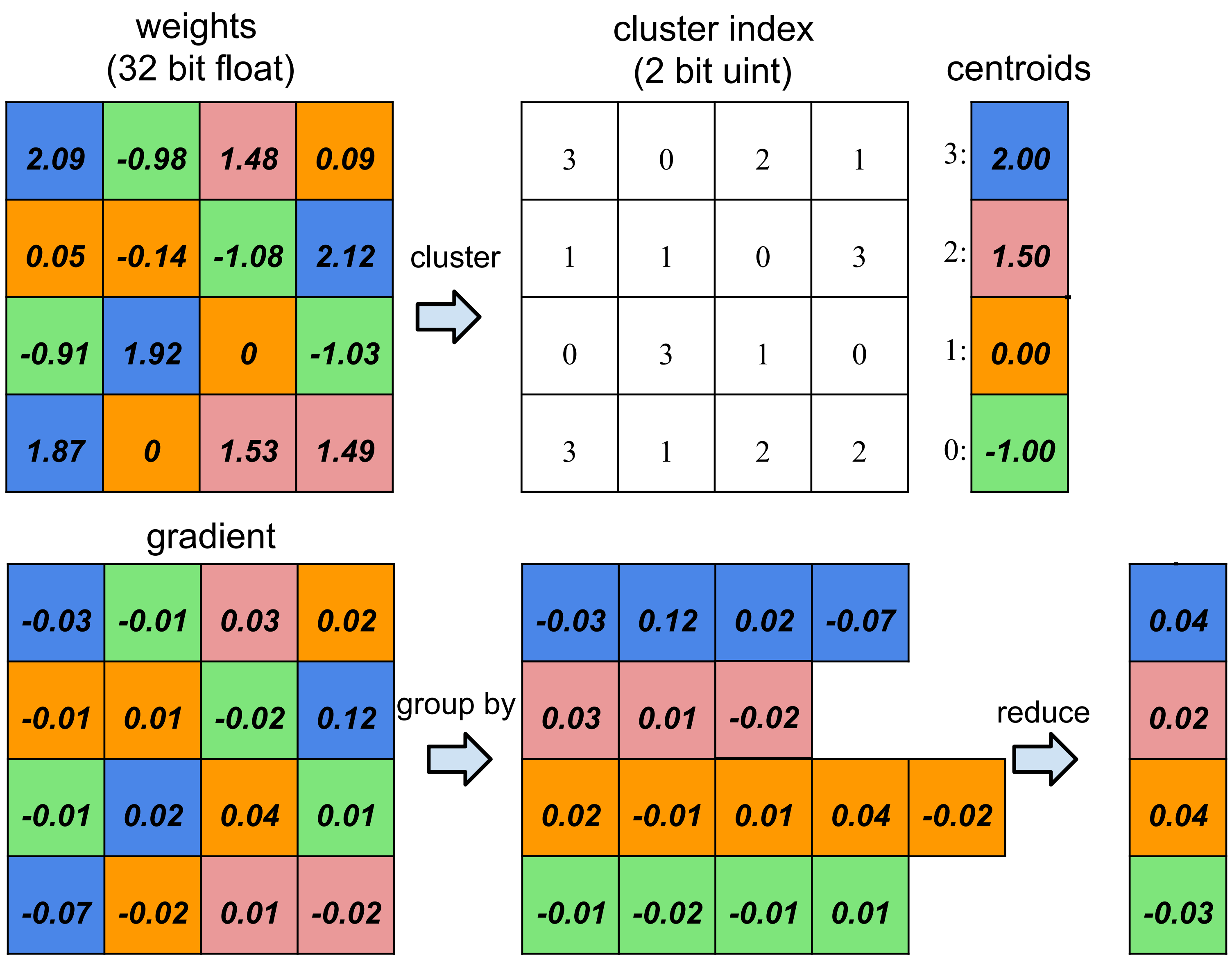
Han et al. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, ICLR 2016

# Weight Sharing: Overview



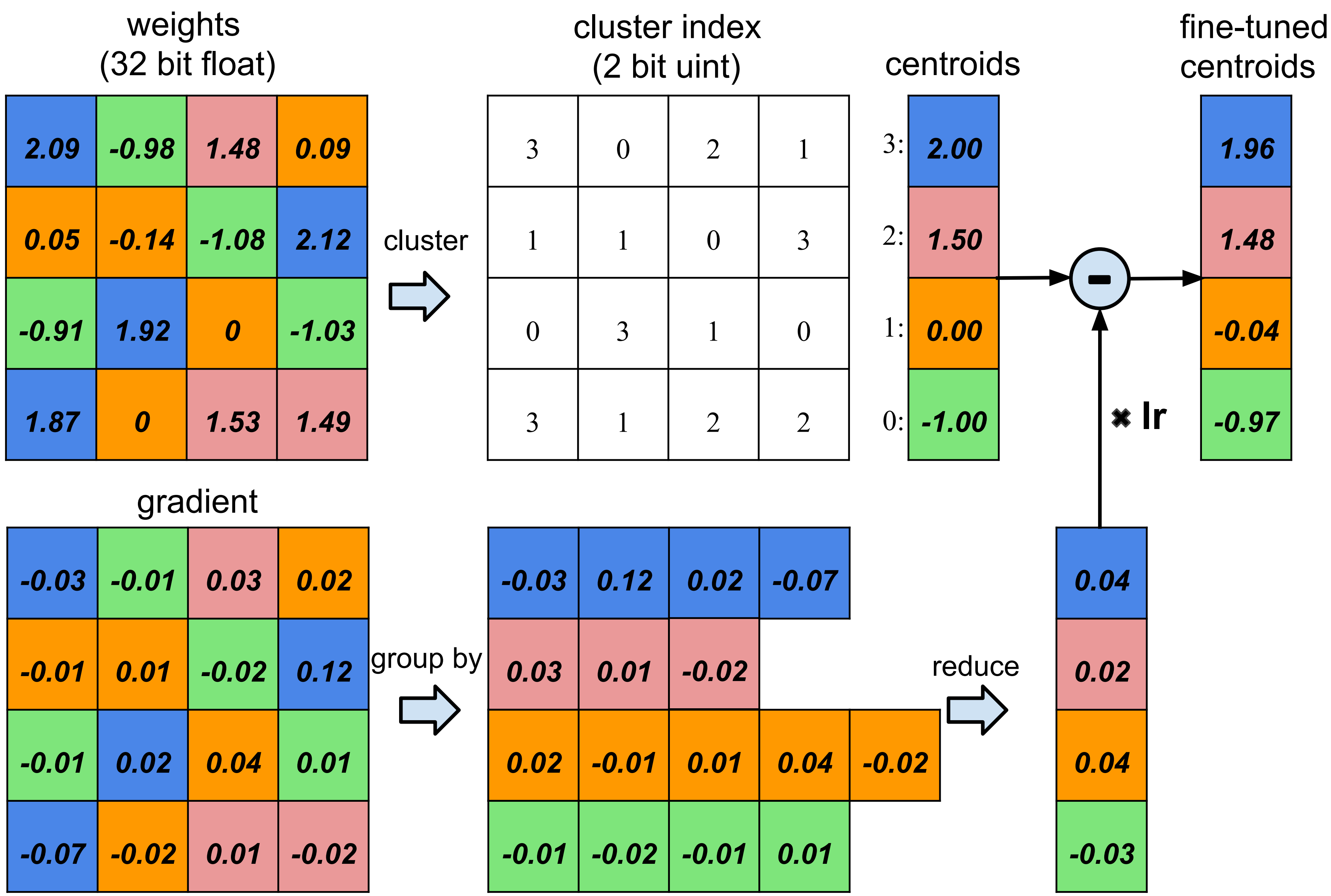

Han et al. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, ICLR 2016

# Weight Sharing: Overview



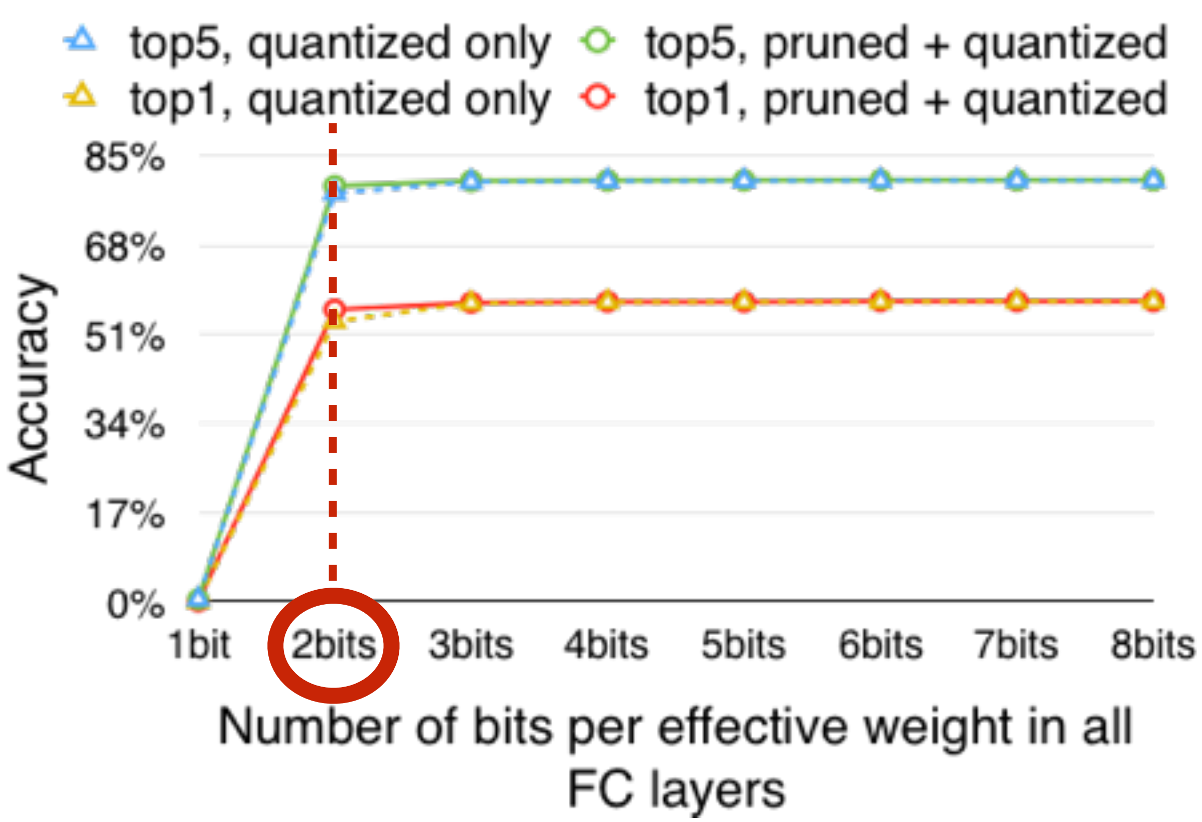

Han et al. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, ICLR 2016

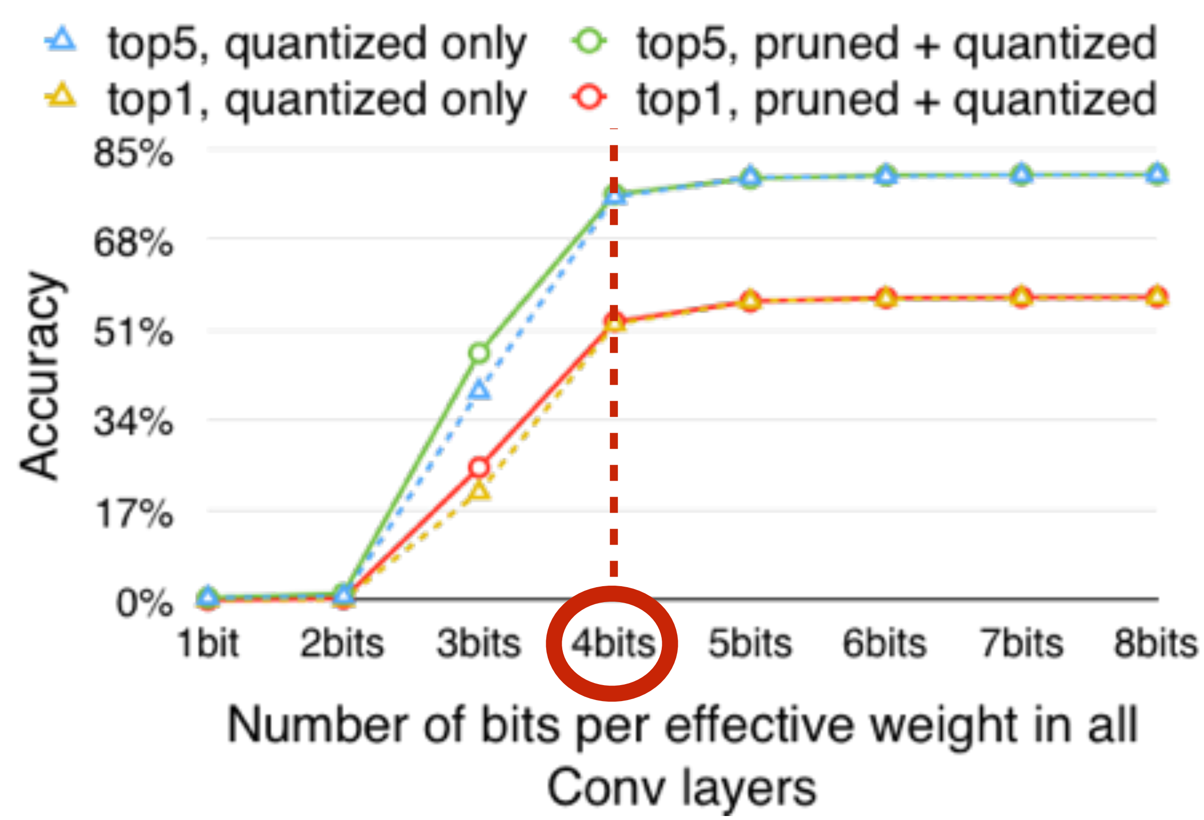
# Weight Sharing: Overview




Han et al. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, ICLR 2016

# Bits Per Weight



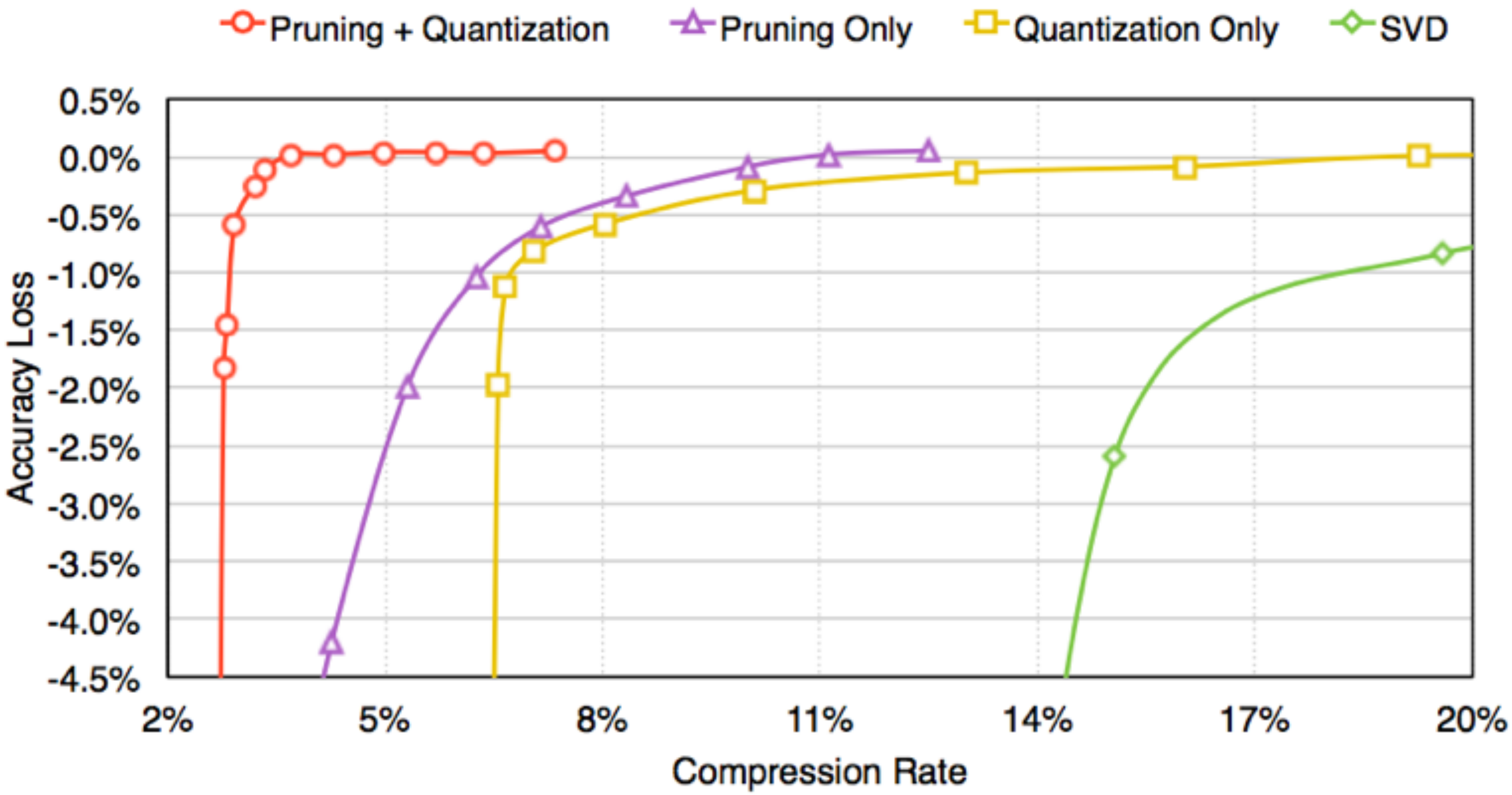

Han et al. "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding", ICLR 2016

# Pruning + Trained Quantization

| #CONV bits / #FC bits | Top-1 Error | Top-5 Error | Top-1 Error Increase | Top-5 Error Increase |
|---|---|---|---|---|
| 32bits / 32bits | 42.78% | 19.73% | - | - |
| 8 bits / 5 bits | 42.78% | 19.70% | 0.00% | -0.03% |
| 8 bits / 4 bits | 42.79% | 19.73% | 0.01% | 0.00% |
| 4 bits / 2 bits | 44.77% | 22.33% | 1.99% | 2.60% |

Han et al. "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding", ICLR 2016

# Pruning + Trained Quantization



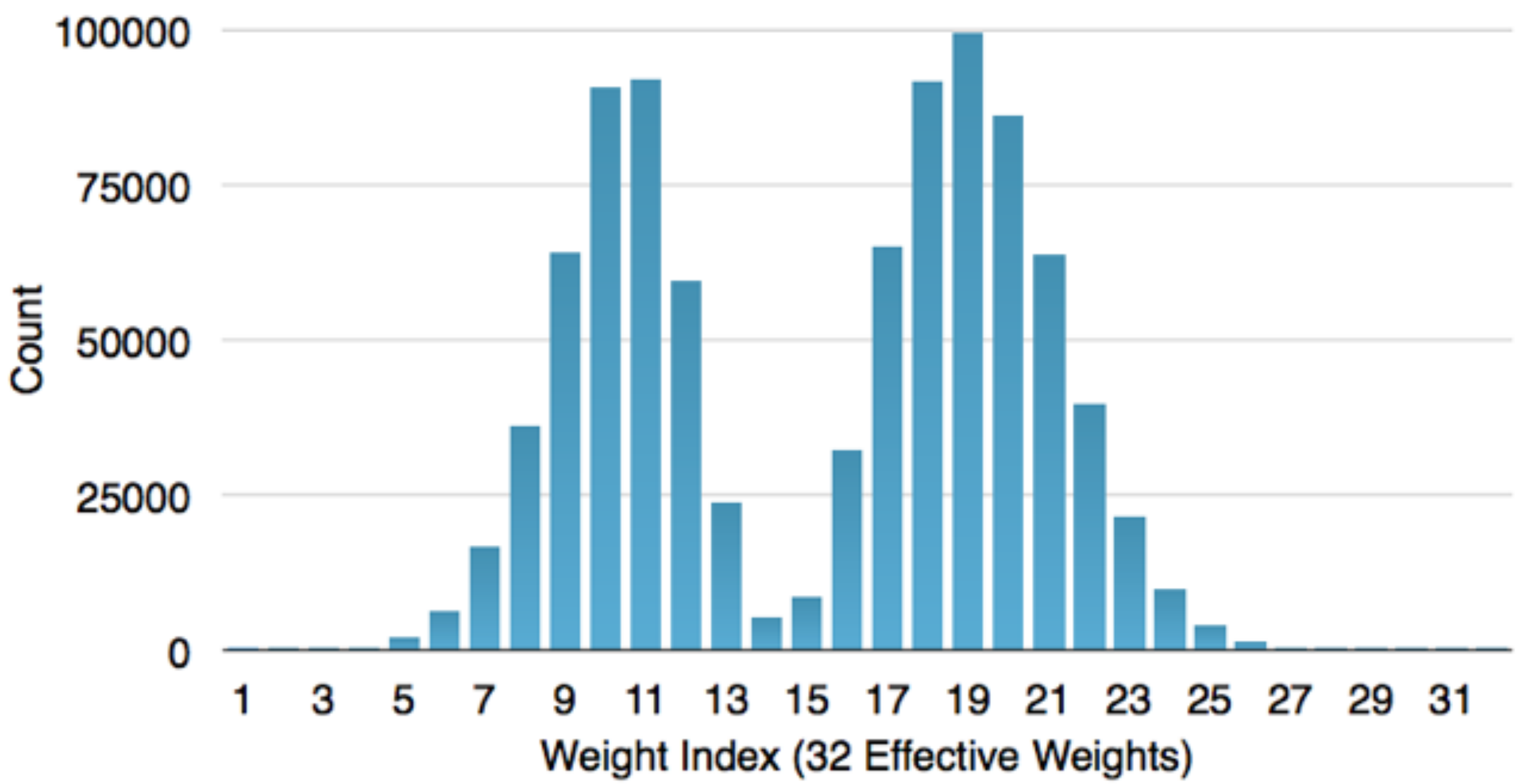

Han et al. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, ICLR 2016

# Deep Compression Pipeline

- **Network Pruning**:
  Less Number of Weights
- **Weight Sharing:**
  **Reduce Storage for Each Remaining Weight**
- **Huffman Coding:**
  **Entropy of the Total Remaining Weights**

Han et al. "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding", ICLR 2016

# Huffman Coding




- Frequent weights: use less bits to represent
- In-frequent weights: use more bits to represent

Han et al. “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, ICLR 2016

# Deep Compression Results

| 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% |

- No loss of accuracy after compression.
- Fits in SRAM cache (120x less energy than DRAM).

# 660KB model, AlexNet-accuracy

https://github.com/songhan/SqueezeNet_compressed

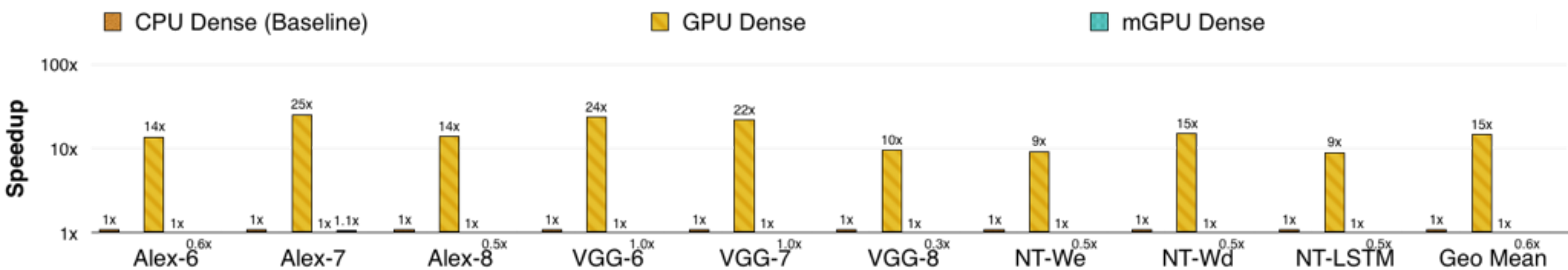
Iandola, Han,et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size” arXiv 2016

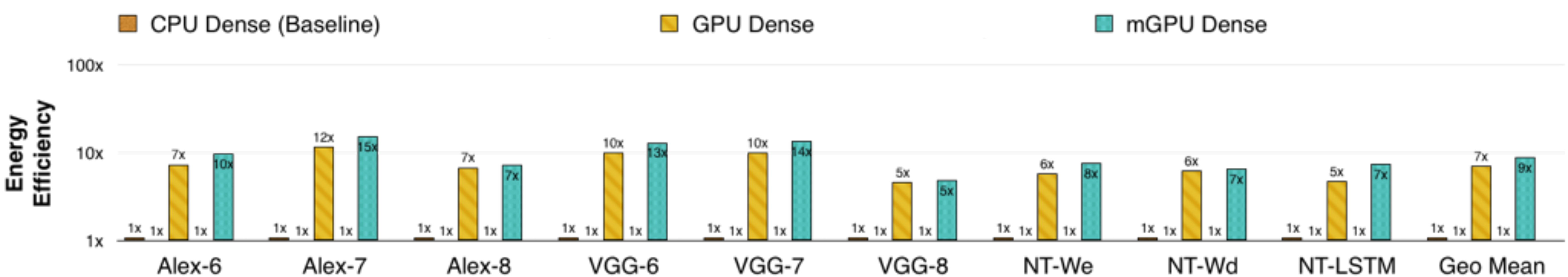
# Conclusion

- **Complex DNNs can be put in mobile applications (<10MB total)**
  - 500MB with-FC network (125M weights) becomes 10MB
  - 10MB all-CONV network (2.5M weights) becomes 1MB
- **Memory bandwidth reduced by 10-50x**
  - Particularly for FC layers in real-time applications with no reuse
- **Faster Prediction**
  - Works well for sparsity level 10%-20%. Ads, Speech…

What happens once DNN size is so small that it fits in SRAM Cache?

# Speedup/Energy Efficiency on CPU/GPU

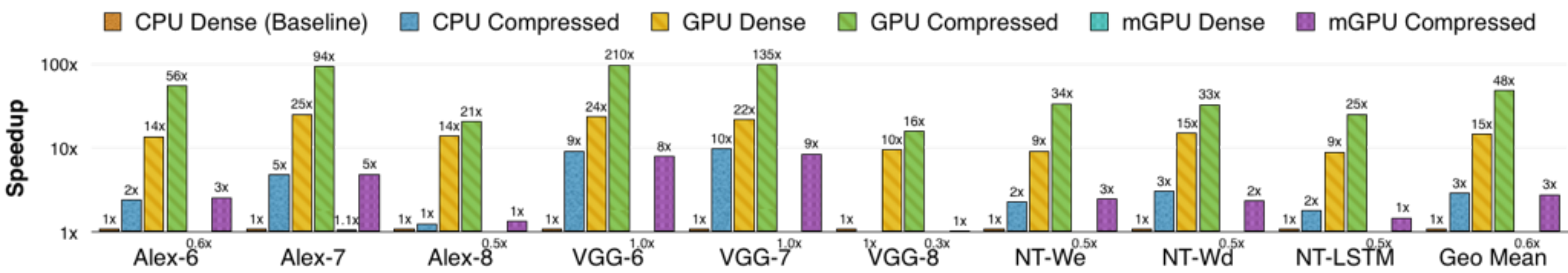


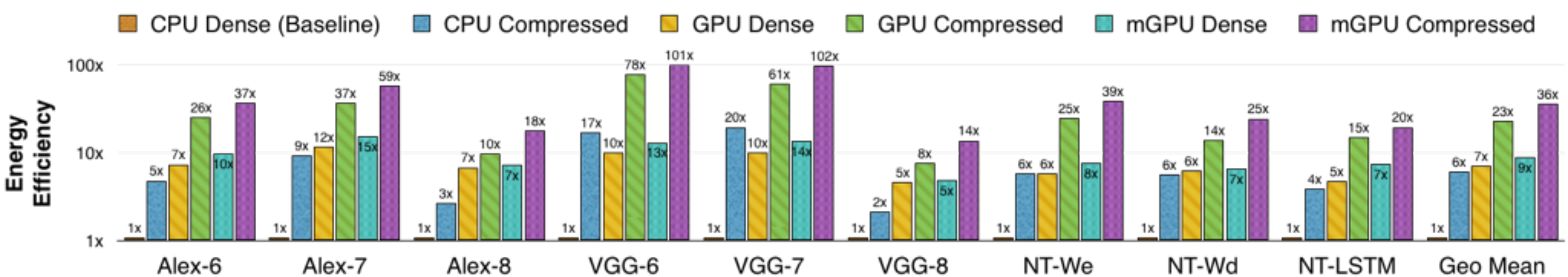




CPU: Core i-7 5930k; GPU: GTX TitanX ; mobile GPU: Tegra K1; All scenarios: batchsize = 1

# Speedup/Energy Efficiency on CPU/GPU

**Facebook is using this to speedup ads click prediction**




CPU: Core i-7 5930k; GPU: GTX TitanX ; mobile GPU: Tegra K1; All scenarios: batchsize = 1

# Part 2: EIE

## Efficient Inference Engine on Compressed Deep Neural Network

Song Han
CVA group, Stanford University

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

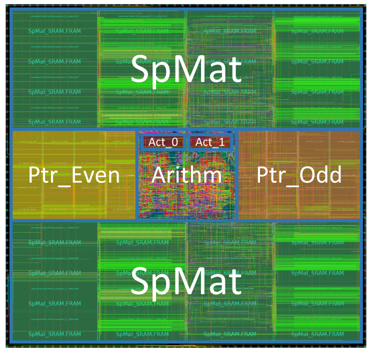
# Problem 2: Faster, Energy Efficient

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

## Problem 2: Faster, Energy Efficient
## Solution 2: EIE accelerator

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

# EIE: First Accelerator for Compressed Sparse Neural Network

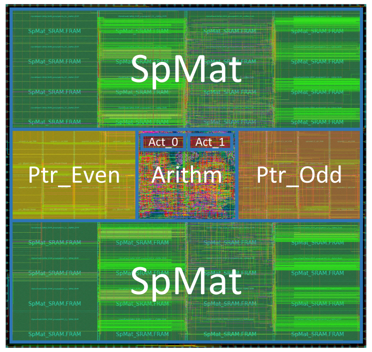



## Problem 2: Faster, Energy Efficient

## Solution 2: EIE accelerator

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

# EIE: First Accelerator for Compressed Sparse Neural Network




## Problem 2: Faster, Energy Efficient

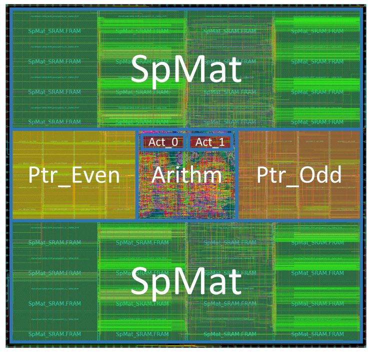
## Solution 2: EIE accelerator

**Sparse Matrix**

90% *static* sparsity
in the weights,
**10x less** computation,
**5x less** memory footprint

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

# EIE: First Accelerator for Compressed Sparse Neural Network




## Problem 2: Faster, Energy Efficient

## Solution 2: EIE accelerator

**Sparse Matrix**

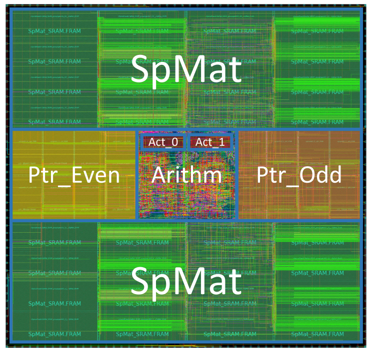
90% *static* sparsity in the weights, **10x less** computation, **5x less** memory footprint

**Sparse Vector**

70% *dynamic* sparsity in the activation **3x less** computation

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

# EIE: First Accelerator for Compressed Sparse Neural Network




## Problem 2: Faster, Energy Efficient

## Solution 2: EIE accelerator

**Sparse Matrix**

90% *static* sparsity
in the weights,
**10x less** computation,
**5x less** memory footprint

**Sparse Vector**

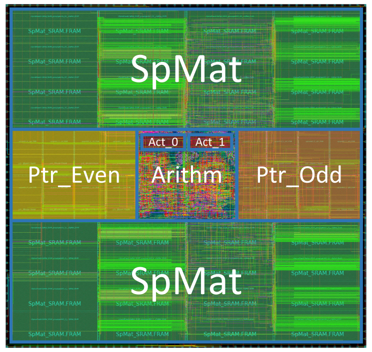
70% *dynamic* sparsity
in the activation
**3x less** computation

**Weight Sharing**

4bits weights
**8x less** memory
footprint

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

# EIE: First Accelerator for Compressed Sparse Neural Network




## Problem 2: Faster, Energy Efficient

## Solution 2: EIE accelerator

### Sparse Matrix

90% *static* sparsity
in the weights,
**10x less** computation,
**5x less** memory footprint

### Sparse Vector

70% *dynamic* sparsity
in the activation
**3x less** computation

### Weight Sharing

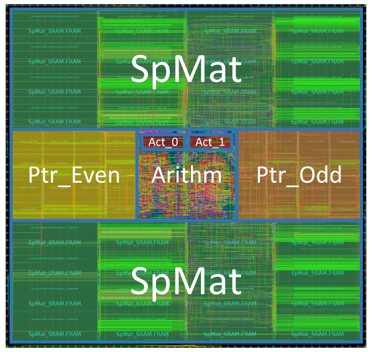
4bits weights
**8x less** memory
footprint

### Fully fits in SRAM

**120x** less energy than DRAM

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

# EIE: First Accelerator for Compressed Sparse Neural Network




## Problem 2: Faster, Energy Efficient

## Solution 2: EIE accelerator

### Sparse Matrix

90% *static* sparsity
in the weights,
**10x less** computation,
**5x less** memory footprint

### Sparse Vector

70% *dynamic* sparsity
in the activation
**3x less** computation

### Weight Sharing

4bits weights
**8x less** memory
footprint

### Fully fits in SRAM

**120x** less energy than DRAM

Savings are **multiplicative**: 5x3x8x120=14,400 theoretical energy improvement.

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

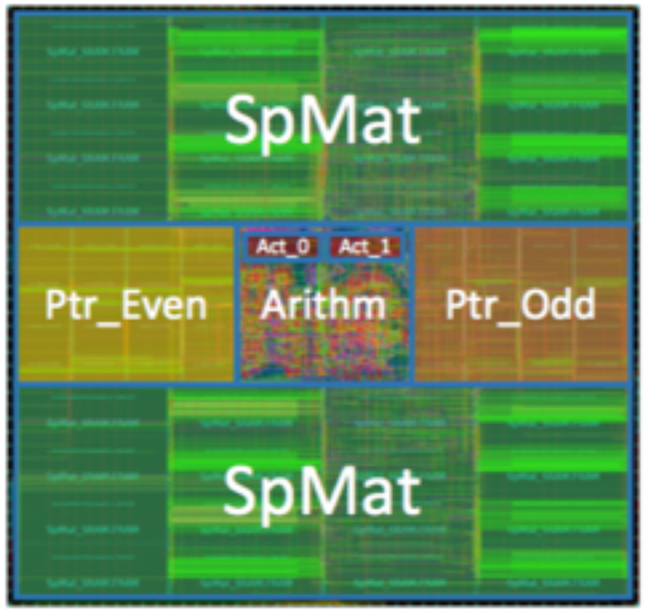
# Benchmark

- CPU: Intel Core-i7 5930k
- GPU: NVIDIA TitanX
- Mobile GPU: NVIDIA Jetson TK1

| Layer | Size | Weight Density | Activation Density | FLOP % | Description |
|---|---|---|---|---|---|
| AlexNet-6 | 4096 × 9216 | 9% | 35.1% | 3% | AlexNet for image classification |
| AlexNet-7 | 4096 × 4096 | 9% | 35.3% | 3% | |
| AlexNet-8 | 1000 × 4096 | 25% | 37.5% | 10% | |
| VGG-6 | 4096 × 25088 | 4% | 18.3% | 1% | VGG-16 for image classification |
| VGG-7 | 4096 × 4096 | 4% | 37.5% | 2% | |
| VGG-8 | 1000 × 4096 | 23% | 41.1% | 9% | |
| NeuralTalk-We | 600 × 4096 | 10% | 100% | 10% | RNN and LSTM for image caption |
| NeuralTalk-Wd | 8791 × 600 | 11% | 100% | 11% | |
| NeuralTalk-LSTM | 2400 × 1201 | 10% | 100% | 11% | |

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

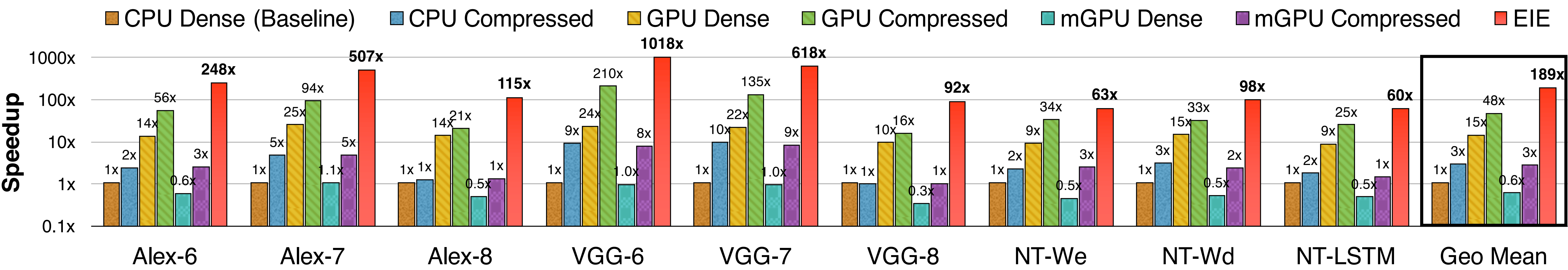
# Result of EIE




| | |
|---|---|
| Technology | 45 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**

# Speedup on EIE



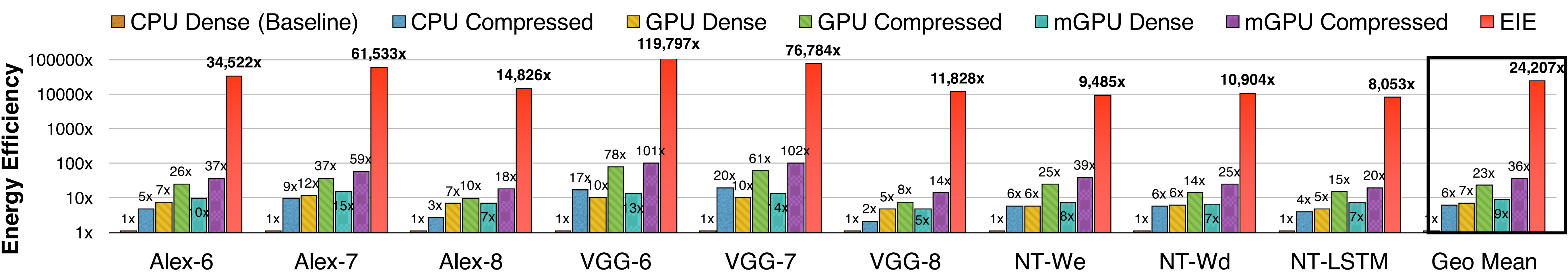

**Compared to CPU and GPU:**

189x and 13x faster

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 on EIE




**Compared to CPU and GPU:**

24,000x and 3,400x more energy efficient

Baseline:

- Intel Core i7 5930K: reported by pcm-power utility
- NVIDIA GeForce GTX Titan X: reported by nvidia-smi utility
- NVIDIA Tegra K1: measured with power-meter, 60% AP+DRAM power

# Where are the savings from?

- <u>Four factors for energy saving:</u>

- **10× *static* weight sparsity;**
  less work to do; less bricks to carry.
- **3× *dynamic* activation sparsity;**
  carry only good bricks; ignore broken bricks.
- **Weight sharing with only 4-bits per weight;**
  lighter bricks to carry.
- **DRAM => SRAM, no need to go off-chip;**
  carry bricks from NY to Stanford => SF to Stanford.

# Conclusion

- EIE: first accelerator for compressed, sparse neural network.
- Compression => Acceleration, no loss accuracy.
- Distributed storage/computation to parallelize/load balance across PEs.
- 13x faster and 3,400x more energy efficient than GPU.
  2.9x faster and 19x more energy efficient than past ASICs.