Deep Compression and EIE:
— Deep Neural Network Model Compression and Efficient Inference Engine

Song Han
CVA group, Stanford University
Apr 7, 2016
Intro about me and my advisor

• Fourth year PhD with Prof. Bill Dally at Stanford.
• Research interest: deep learning model compression and hardware acceleration, to make inference more efficient for deployment.
• Recent work on “Deep Compression” and “EIE: Efficient Inference Engine” covered by TheNextPlatform & O’Reilly & TechEmergence & HackerNews

• Professor at Stanford University and former chairman of CS department, leads the Concurrent VLSI Architecture Group.
• Chief Scientist of NVIDIA.
• Member of the National Academy of Engineering, Fellow of the American Academy of Arts & Sciences, Fellow of the IEEE, Fellow of the ACM and numerous other rewards…
Thanks to my collaborators

- **NVIDIA**: Jeff Pool, John Tran, Bill Dally
- **Stanford**: Xingyu Liu, Jing Pu, Ardavan Pedram, Mark Horowitz, Bill Dally
- **Tsinghua**: Huizi Mao, Song Yao, Yu Wang
- **Berkeley**: Forrest Iandola, Matthew Moskewicz, Khalid Ashraf, Kurt Keutzer

You’ll be interested in his GTC talk: S6417 - FireCaffe
This Talk:

- **Deep Compression**\[^{[1,2]}\]: A Deep Neural Network Model Compression Pipeline.
- **EIE Accelerator**\[^{[3]}\]: Efficient Inference Engine that Accelerates the Compressed Deep Neural Network Model.
- **SqueezeNet++**\[^{[4,5]}\]: ConvNet Architecture Design Space Exploration

---

[1]. Han et al. “Learning both Weights and Connections for Efficient Neural Networks”, NIPS 2015  
[4]. Iandola, Han, et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size” arXiv 2016  
[5]. Yao, Han, et.al, “Hardware-friendly convolutional neural network with even-number filter size” ICLR’16 workshop
Deep Learning: Next Wave of AI

Image Recognition

Speech Recognition

Natural Language Processing
Applications
The Problem:
If Running DNN on Mobile…

App developers suffers from the model size

“At Baidu, our #1 motivation for compressing networks is to bring down the size of the binary file. As a mobile-first company, we frequently update various apps via different app stores. We've very sensitive to the size of our binary files, and a feature that increases the binary size by 100MB will receive much more scrutiny than one that increases it by 10MB.” —Andrew Ng
The Problem:
If Running DNN on Mobile...

Hardware engineer suffers from the model size
(embedded system, limited resource)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy [pJ]</th>
<th>Relative Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>32 bit int ADD</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>32 bit float ADD</td>
<td>0.9</td>
<td>9</td>
</tr>
<tr>
<td>32 bit Register File</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>32 bit int MULT</td>
<td>3.1</td>
<td>31</td>
</tr>
<tr>
<td>32 bit float MULT</td>
<td>3.7</td>
<td>37</td>
</tr>
<tr>
<td>32 bit SRAM Cache</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>32 bit DRAM Memory</td>
<td>640</td>
<td>6400</td>
</tr>
</tbody>
</table>

Relative Energy Cost

Stanford University
The Problem:
If Running DNN on the Cloud…

Network Delay
Power Budget
User Privacy

Intelligent but Inefficient
Deep Compression

Problem 1: Model Size
Solution 1: Deep Compression

- Smaller Size: Compress Mobile App Size by 35x-50x
- Accuracy: no loss of accuracy; improved accuracy
- Speedup: make inference faster
Problem 2: Latency, Power, Energy
Solution 2: ASIC accelerator

- **Offline**
  - No dependency on network connection

- **Real Time**
  - No network delay
  - High frame rate

- **Low Power**
  - High energy efficiency
  - That preserves battery
Part 1: Deep Compression

- AlexNet: 35x, 240MB => 6.9MB => **0.47MB (510x)**
- VGG16: 49x, 552MB => 11.3MB
- With no loss of accuracy on ImageNet12
- Weights fits on-chip SRAM, taking 120x less energy than DRAM

1. Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS 2015
2. Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, ICLR 2016
3. Iandola, Han, et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size” ECCV submission
1. Pruning

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

- 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.

---

Retrain to Recover Accuracy

Han et al. Learning both Weights and Connections for Efficient Neural Networks, NIPS 2015
Pruning: Result on 4 Covnets

<table>
<thead>
<tr>
<th>Network</th>
<th>Top-1 Error</th>
<th>Top-5 Error</th>
<th>Parameters</th>
<th>Compression Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet-300-100 Ref</td>
<td>1.64%</td>
<td>-</td>
<td>267K</td>
<td></td>
</tr>
<tr>
<td>LeNet-300-100 Pruned</td>
<td>1.59%</td>
<td>-</td>
<td>22K</td>
<td>12×</td>
</tr>
<tr>
<td>LeNet-5 Ref</td>
<td>0.80%</td>
<td>-</td>
<td>431K</td>
<td></td>
</tr>
<tr>
<td>LeNet-5 Pruned</td>
<td>0.77%</td>
<td>-</td>
<td>36K</td>
<td>12×</td>
</tr>
<tr>
<td>AlexNet Ref</td>
<td>42.78%</td>
<td>19.73%</td>
<td>61M</td>
<td></td>
</tr>
<tr>
<td>AlexNet Pruned</td>
<td>42.77%</td>
<td>19.67%</td>
<td>6.7M</td>
<td>9×</td>
</tr>
<tr>
<td>VGG16 Ref</td>
<td>31.50%</td>
<td>11.32%</td>
<td>138M</td>
<td></td>
</tr>
<tr>
<td>VGG16 Pruned</td>
<td>31.34%</td>
<td>10.88%</td>
<td>10.3M</td>
<td>13×</td>
</tr>
</tbody>
</table>

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

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

Visualization of the first FC layer’s sparsity pattern of Lenet-300-100. It has a banded structure repeated 28 times, which correspond to the un-pruned parameters in the center of the images, since the digits are written in the center.
Pruning NeuralTalk and LSTM

- Pruning away 90% parameters in NeuralTalk doesn’t hurt BLUE score with proper retraining

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 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

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

Abi See, “CS224N Final Project: Exploiting the Redundancy in Neural Machine Translation”
Pruning Neural Machine Translation

Word Embedding:

- top layer weights
- target embedding weights
- source embedding weights

Dark means zero and redundant, White means non-zero and useful

LSTM:

Source layer 1 weights
Target layer 1 weights
Source layer 2 weights
Target layer 2 weights

Abi See, “CS224N Final Project: Exploiting the Redundancy in Neural Machine Translation”
## Speedup (FC layer)

![Speedup Graph]

- **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

Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, ICLR 2016
Energy Efficiency (FC layer)

- **Intel Core i7 5930K**: CPU socket and DRAM power are reported by pcm-power utility
- **NVIDIA GeForce GTX Titan X**: reported by nvidia-smi utility
- **NVIDIA Tegra K1**: measured the total power consumption with a power-meter, 15% AC to DC conversion loss, 85% regulator efficiency and 15% power consumed by peripheral components => 60% AP+DRAM power

Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, ICLR 2016
2. Weight Sharing (Trained Quantization)

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

Figure 3: Weight sharing by scalar quantization (top) and centroids fine-tuning (bottom)

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

Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, ICLR 2016
Accuracy ~ #Bits on 5 Conv Layers + 3 FC Layers
Weight Sharing: Result

• 16 Million => 2^4=16

• 8/5 bit quantization results in no accuracy loss

• 8/4 bit quantization results in no top-5 accuracy loss, 0.1% top-1 accuracy loss

• 4/2 bit quantization results in -1.99% top-1 accuracy loss, and -2.60% top-5 accuracy loss, not that bad:

Han et al. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, ICLR 2016
Figure 7: Pruning doesn’t hurt quantization. Dashed: quantization on unpruned network. Solid: quantization on pruned network; Accuracy begins to drop at the same number of quantization bits whether or not the network has been pruned. Although pruning made the number of parameters less, quantization still works well, or even better (3 bits case on the left figure) as in the unpruned network.
3. Huffman Coding

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

Huffman code is a type of optimal prefix code that is commonly used for loss-less data compression. It produces a variable-length code table for encoding source symbol. The table is derived from the occurrence probability for each symbol. As in other entropy encoding methods, more common symbols are represented with fewer bits than less common symbols, thus save the total space.

Figure 5: Distribution for weight (Left) and index (Right). The distribution is biased and can be compressed by Huffman encoding.
Deep Compression Result on 4 Convnets

<table>
<thead>
<tr>
<th>Network</th>
<th>Top-1 Error</th>
<th>Top-5 Error</th>
<th>Parameters</th>
<th>Compress Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeNet-300-100 Ref</td>
<td>1.64%</td>
<td>-</td>
<td>1070 KB</td>
<td></td>
</tr>
<tr>
<td>LeNet-300-100 Compressed</td>
<td>1.58%</td>
<td>-</td>
<td>27 KB</td>
<td>40×</td>
</tr>
<tr>
<td>LeNet-5 Ref</td>
<td>0.80%</td>
<td>-</td>
<td>1720 KB</td>
<td></td>
</tr>
<tr>
<td>LeNet-5 Compressed</td>
<td>0.74%</td>
<td>-</td>
<td>44 KB</td>
<td>39×</td>
</tr>
<tr>
<td>AlexNet Ref</td>
<td>42.78%</td>
<td>19.73%</td>
<td>240 MB</td>
<td></td>
</tr>
<tr>
<td>AlexNet Compressed</td>
<td>42.78%</td>
<td>19.70%</td>
<td>6.9 MB</td>
<td>35×</td>
</tr>
<tr>
<td>VGG16 Ref</td>
<td>31.50%</td>
<td>11.32%</td>
<td>552 MB</td>
<td></td>
</tr>
<tr>
<td>VGG16 Compressed</td>
<td>31.17%</td>
<td>10.91%</td>
<td>11.3 MB</td>
<td>49×</td>
</tr>
</tbody>
</table>

Table 1: The compression pipeline can save 35× to 49× parameter storage with no drop in predictive performance

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

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

<table>
<thead>
<tr>
<th>Layer</th>
<th>#Weights (P)</th>
<th>Weights% (P)</th>
<th>Weight bits (P+Q)</th>
<th>Weight bits (+H)</th>
<th>Index bits (P+Q)</th>
<th>Index bits (+H)</th>
<th>Compress rate (P+Q)</th>
<th>Compress rate (P+Q+H)</th>
</tr>
</thead>
<tbody>
<tr>
<td>conv1</td>
<td>35K</td>
<td>84%</td>
<td>8</td>
<td>6.3</td>
<td>4</td>
<td>1.2</td>
<td>32.6%</td>
<td>20.53%</td>
</tr>
<tr>
<td>conv2</td>
<td>307K</td>
<td>38%</td>
<td>8</td>
<td>5.5</td>
<td>4</td>
<td>2.3</td>
<td>14.5%</td>
<td>9.43%</td>
</tr>
<tr>
<td>conv3</td>
<td>885K</td>
<td>35%</td>
<td>8</td>
<td>5.1</td>
<td>4</td>
<td>2.6</td>
<td>13.1%</td>
<td>8.44%</td>
</tr>
<tr>
<td>conv4</td>
<td>663K</td>
<td>37%</td>
<td>8</td>
<td>5.2</td>
<td>4</td>
<td>2.5</td>
<td>14.1%</td>
<td>9.11%</td>
</tr>
<tr>
<td>conv5</td>
<td>442K</td>
<td>37%</td>
<td>8</td>
<td>5.6</td>
<td>4</td>
<td>2.5</td>
<td>14.0%</td>
<td>9.43%</td>
</tr>
<tr>
<td>fc6</td>
<td>38M</td>
<td>9%</td>
<td>5</td>
<td>3.9</td>
<td>4</td>
<td>3.2</td>
<td>3.0%</td>
<td>2.39%</td>
</tr>
<tr>
<td>fc7</td>
<td>17M</td>
<td>9%</td>
<td>5</td>
<td>3.6</td>
<td>4</td>
<td>3.7</td>
<td>3.0%</td>
<td>2.46%</td>
</tr>
<tr>
<td>fc8</td>
<td>4M</td>
<td>25%</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3.2</td>
<td>7.3%</td>
<td>5.85%</td>
</tr>
<tr>
<td>total</td>
<td>61M</td>
<td>11%</td>
<td>5.4</td>
<td>4</td>
<td>4</td>
<td>3.2</td>
<td>3.7%</td>
<td>2.88%</td>
</tr>
</tbody>
</table>

Table 4: Compression Statistics for Alexnet. P: pruning, Q: quantization, H: Huffman Encoding

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

New Network Topology

+ Deep Compression

Iandola, Han, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size
New Network Topology + Deep Compression

Fig 1: SqueezeNet architecture

Fig 2. Deep compression is compatible with even extreme efficient network architecture such as SqueezeNet: It can be pruned 3x, quantized to 6bit w/o loss of accuracy.

Iandola, Han, et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size” arXiv 2016
470KB model, AlexNet-accuracy

<table>
<thead>
<tr>
<th>CNN architecture</th>
<th>Compression Approach</th>
<th>Data Type</th>
<th>Original → Compressed Model Size</th>
<th>Reduction in Model Size vs. AlexNet</th>
<th>Top-1 ImageNet Accuracy</th>
<th>Top-5 ImageNet Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>None (baseline)</td>
<td>32 bit</td>
<td>240MB</td>
<td>1x</td>
<td>57.2%</td>
<td>80.3%</td>
</tr>
<tr>
<td>AlexNet</td>
<td>SVD [3]</td>
<td>32 bit</td>
<td>240MB → 48MB</td>
<td>5x</td>
<td>56.0%</td>
<td>79.4%</td>
</tr>
<tr>
<td>AlexNet</td>
<td>Network Pruning [4]</td>
<td>32 bit</td>
<td>240MB → 27MB</td>
<td>9x</td>
<td>57.2%</td>
<td>80.3%</td>
</tr>
<tr>
<td>AlexNet</td>
<td>Deep Compression [5]</td>
<td>5-8 bit</td>
<td>240MB → 6.9MB</td>
<td>35x</td>
<td>57.2%</td>
<td>80.3%</td>
</tr>
<tr>
<td>SqueezeNet (ours)</td>
<td>None</td>
<td>32 bit</td>
<td>4.8MB</td>
<td>50x</td>
<td>57.5%</td>
<td>80.3%</td>
</tr>
<tr>
<td>SqueezeNet (ours)</td>
<td>Deep Compression</td>
<td>8 bit</td>
<td>4.8MB → 0.66MB</td>
<td>363x</td>
<td>57.5%</td>
<td>80.3%</td>
</tr>
<tr>
<td>SqueezeNet (ours)</td>
<td>Deep Compression</td>
<td>6 bit</td>
<td>4.8MB → 0.47MB</td>
<td>510x</td>
<td>57.5%</td>
<td>80.3%</td>
</tr>
</tbody>
</table>

Iandola, Han, et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size” arXiv 2016
470KB model, AlexNet-accuracy

https://github.com/songhan/SqueezeNet_compressed
Smaller DNN means…

(1) Less communication across servers during distributed training.
(2) Easier to download from App Store.
(3) Less bandwidth to update model to an autonomous car.
(4) Easier to deploy on embedded hardware with limited memory.
A Model Compression Tool for App Developers

deepcompression.net is under construction

• **Easy Version (done):**
  ✓ No training needed
  ✓ Fast (3 minutes)
  ✗ 5x - 10x compression rate
  ✗ 1% loss of accuracy

• **Advanced Version (todo):**
  ✓ 35x - 50x compression rate
  ✓ no loss of accuracy
  ✗ Training is needed
  ✗ Slow
DeepCompressor.net

Job Management

Job Configuration

Job Name
No more than 30 characters

Job Type
- SVD for Conv Layer
- SVD for FC Layer
- Pruning
- Quantization
- No-model Pruning and Quantization

Current Files

- uploaded_prototxt
  - example_svd_job_outp
  - trainval_prototxt

- uploaded_caffemodel
  - a.caffemodel
  - small_0.8117_0.5886_1
  - small_0.8315_0.5923_1

- generated_prototxt
  - wrongoutput_prototxt
  - vgg16_svd_fc6_512.prot
  - vgg16_svd_fc6_512.prot

- generated_caffemodel
  - vgg16_svd_fc6_512.ca
  - 1
  - ff
  - 33
DeepCompression.net

Provides a trial account for GTC attendees:

• Username: deepcompression
• Password: songhan

welcome your feedback!
Conclusion

- We have presented a method to compress neural networks without affecting accuracy by finding the right connections and quantizing the weights.
- Pruning the unimportant connections => quantizing the network and enforce weight sharing => apply Huffman encoding.
- We highlight our experiments on ImageNet, and reduced the weight storage by 35×, VGG16 by 49×, without loss of accuracy.
- Now weights can fit in cache
Part2: SqueezeNet++

——CNN Design Space Exploration

Song Han

CVA group, Stanford University

Apr 7, 2016
## Motivation: How to choose so many architectural dimensions?

<table>
<thead>
<tr>
<th>layer name/type</th>
<th>output size</th>
<th>filter size / stride (if not a fire layer)</th>
<th>depth</th>
<th>( s_{1x1} ) (#1x1 squeeze)</th>
<th>( e_{1x1} ) (#1x1 expand)</th>
<th>( e_{3x3} ) (#3x3 expand)</th>
<th>( s_{1x1} ) sparsity</th>
<th>( e_{1x1} ) sparsity</th>
<th>( e_{3x3} ) sparsity</th>
<th># bits</th>
<th>#parameter before pruning</th>
<th>#parameter after pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>input image</td>
<td>224x224x3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>conv1</td>
<td>111x111x96</td>
<td>7x7/2 (x96)</td>
<td>1</td>
<td>100% (7x7)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6bit</td>
<td>14,208</td>
<td>14,208</td>
</tr>
<tr>
<td>maxpool1</td>
<td>55x55x96</td>
<td>3x3/2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fire2</td>
<td>55x55x128</td>
<td></td>
<td>2</td>
<td>16</td>
<td>64</td>
<td>64</td>
<td>100%</td>
<td>100%</td>
<td>33%</td>
<td>6bit</td>
<td>11,920</td>
<td>5,746</td>
</tr>
<tr>
<td>fire3</td>
<td>55x55x128</td>
<td></td>
<td>2</td>
<td>16</td>
<td>64</td>
<td>64</td>
<td>100%</td>
<td>100%</td>
<td>33%</td>
<td>6bit</td>
<td>12,432</td>
<td>6,258</td>
</tr>
<tr>
<td>fire4</td>
<td>55x55x256</td>
<td></td>
<td>2</td>
<td>32</td>
<td>128</td>
<td>128</td>
<td>100%</td>
<td>100%</td>
<td>33%</td>
<td>6bit</td>
<td>45,344</td>
<td>20,646</td>
</tr>
<tr>
<td>maxpool4</td>
<td>27x27x256</td>
<td>3x3/2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fire5</td>
<td>27x27x256</td>
<td></td>
<td>2</td>
<td>32</td>
<td>128</td>
<td>128</td>
<td>100%</td>
<td>100%</td>
<td>33%</td>
<td>6bit</td>
<td>49,440</td>
<td>24,742</td>
</tr>
<tr>
<td>fire6</td>
<td>27x27x384</td>
<td></td>
<td>2</td>
<td>48</td>
<td>192</td>
<td>192</td>
<td>100%</td>
<td>50%</td>
<td>33%</td>
<td>6bit</td>
<td>104,880</td>
<td>44,700</td>
</tr>
<tr>
<td>fire7</td>
<td>27x27x384</td>
<td></td>
<td>2</td>
<td>48</td>
<td>192</td>
<td>192</td>
<td>100%</td>
<td>50%</td>
<td>33%</td>
<td>6bit</td>
<td>111,024</td>
<td>46,236</td>
</tr>
<tr>
<td>fire8</td>
<td>27x27x512</td>
<td></td>
<td>2</td>
<td>64</td>
<td>256</td>
<td>256</td>
<td>100%</td>
<td>50%</td>
<td>33%</td>
<td>6bit</td>
<td>188,992</td>
<td>77,581</td>
</tr>
<tr>
<td>maxpool8</td>
<td>13x13x512</td>
<td>3x3/2</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fire9</td>
<td>13x13x512</td>
<td></td>
<td>2</td>
<td>64</td>
<td>256</td>
<td>256</td>
<td>50%</td>
<td>100%</td>
<td>30%</td>
<td>6bit</td>
<td>197,184</td>
<td>77,581</td>
</tr>
<tr>
<td>conv10</td>
<td>13x13x1000</td>
<td>1x1/1 (x1000)</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20% (3x3)</td>
<td></td>
<td>6bit</td>
<td>513,000</td>
<td>103,400</td>
</tr>
<tr>
<td>avgpool10</td>
<td>1x1x1000</td>
<td>13x13/1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 1. SqueezeNet architectural dimensions.
Micro-Architecture DSX

Use sensitive analysis

- Micro-Architecture: how to size the layers: 64? 128? 256?
- Sensitivity: Pruning 50% weights for a single layer and measure the accuracy.
- => Sensitivity analysis helps sizing the number of parameters in a layer.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Top-1 Accuracy</th>
<th>Top-5 Accuracy</th>
<th>Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SqueezeNet</td>
<td>57.5%</td>
<td>80.3%</td>
<td>4.8MB</td>
</tr>
<tr>
<td>SqueezeNet++</td>
<td>59.5%</td>
<td>81.5%</td>
<td>7.1MB</td>
</tr>
</tbody>
</table>

Iandola, Han, et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size”, submitted to ECCV
“SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size”, submitted to ECCV
Table 3. SqueezeNet accuracy and model size using different macroarchitecture

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Top-1 Accuracy</th>
<th>Top-5 Accuracy</th>
<th>Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla SqueezeNet</td>
<td>57.5%</td>
<td>80.3%</td>
<td>4.8MB</td>
</tr>
<tr>
<td>SqueezeNet + Simple Bypass</td>
<td><strong>60.4%</strong></td>
<td><strong>82.5%</strong></td>
<td>4.8MB</td>
</tr>
<tr>
<td>SqueezeNet + Complex Bypass</td>
<td>58.8%</td>
<td>82.0%</td>
<td>7.7MB</td>
</tr>
</tbody>
</table>

Iandola, Han, et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size”, submitted to ECCV
**DSD Training (Dense-Sparse-Dense) improves accuracy**

![Diagram of DSD Training]

**Table 5.** Improving accuracy with dense→sparse→dense (DSD) training.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Top-1 Accuracy</th>
<th>Top-5 Accuracy</th>
<th>Model Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>SqueezeNet</td>
<td>57.5%</td>
<td>80.3%</td>
<td>4.8MB</td>
</tr>
<tr>
<td>SqueezeNet (DSD)</td>
<td><strong>61.8%</strong></td>
<td><strong>83.5%</strong></td>
<td><strong>4.8MB</strong></td>
</tr>
</tbody>
</table>

- dense→sparse→dense (DSD) training yielded 4.3% higher accuracy.

- Sparsity is a form of regularization. Once the network arrives at a local minimum given the sparsity constraint, relaxing the constraint gives the network more freedom to escape the saddle point and arrive at a higher-accuracy local minimum.

- Regularizing models by intermittently pruning parameters throughout training would be an interesting area of future work.
Design Space Exploration Conclusion

- SqueezeNet++: sizing the layers with sensitivity analysis
- Use even kernel
- SqueezeNet +simple + complex bypass layer
- DSD training: improves accuracy by 4.3%
Part 3:
EIE: Efficient Inference Engine on Compressed Deep Neural Network

Song Han
CVA group, Stanford University
Apr 7, 2016

ASIC Accelerator on Compressed DNN

- **sparse, indirectly indexed, weight shared** MxV accelerator.

- **Offline**
  No dependency on network connection

- **Real Time**
  No network delay
  high frame rate

- **Low Power**
  High energy efficiency
  that preserves battery

Distribute Storage and Processing

Figure 2: Matrix $W$ and vectors $a$ and $b$ are interleaved over 4 PEs. Elements of the same color are stored in the same PE.

Evaluation


2. RTL in Verilog, verified its output result with the golden model in Modelsim.

3. Synthesized EIE using the Synopsys Design Compiler (DC) under the TSMC 45nm GP standard VT library with worst case PVT corner.

4. Placed and routed the PE using the Synopsys IC compiler (ICC). We used Cacti to get SRAM area and energy numbers.

5. Annotated the toggle rate from the RTL simulation to the gate-level netlist, which was dumped to switching activity interchange format (SAIF), and estimated the power using Prime-Time PX.

Layout of an EIE PE

Figure 7: Layout of one PE in EIE under TSMC 45nm process.

Table 2: The implementation results of one PE in EIE and the breakdown by component type (line 3-7), by module (line 8-13). The critical path of EIE is $1.15\,\text{ns}$.
Baseline and Benchmark

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

Table 3: Benchmark from state-of-the-art DNN models

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size</th>
<th>Weight%</th>
<th>Act%</th>
<th>FLOP%</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alex-6</td>
<td>9216, 4096</td>
<td>9%</td>
<td>35.1%</td>
<td>3%</td>
<td>Compressed AlexNet [1] for large scale image classification</td>
</tr>
<tr>
<td>Alex-7</td>
<td>4096, 4096</td>
<td>9%</td>
<td>35.3%</td>
<td>3%</td>
<td></td>
</tr>
<tr>
<td>Alex-8</td>
<td>4096, 1000</td>
<td>25%</td>
<td>37.5%</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>VGG-6</td>
<td>25088, 4096</td>
<td>4%</td>
<td>37.5%</td>
<td>2%</td>
<td>Compressed VGG-16 [3] for large scale image classification</td>
</tr>
<tr>
<td>VGG-7</td>
<td>4096, 4096</td>
<td>9%</td>
<td>37.5%</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>VGG-8</td>
<td>4096, 1000</td>
<td>23%</td>
<td>41.1%</td>
<td>9%</td>
<td>Compressed NeuralTalk [7] with RNN and LSTM for automatic image captioning</td>
</tr>
<tr>
<td>NT-We</td>
<td>4096, 600</td>
<td>10%</td>
<td>100%</td>
<td>10%</td>
<td></td>
</tr>
<tr>
<td>NT-Wd</td>
<td>600, 8791</td>
<td>11%</td>
<td>100%</td>
<td>11%</td>
<td></td>
</tr>
<tr>
<td>NTLSTM</td>
<td>1201, 2400</td>
<td>10%</td>
<td>100%</td>
<td>11%</td>
<td></td>
</tr>
</tbody>
</table>

Result: Speedup / Energy Efficiency

## Comparison with other Platforms

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform Type</td>
<td>CPU</td>
<td>GPU</td>
<td>mGPU</td>
<td>FPGA</td>
<td>ASIC</td>
<td>ASIC</td>
<td>ASIC</td>
</tr>
<tr>
<td>Technology</td>
<td>22nm</td>
<td>28nm</td>
<td>28nm</td>
<td>28nm</td>
<td>28nm</td>
<td>28nm</td>
<td>45nm</td>
</tr>
<tr>
<td>Clock (MHz)</td>
<td>3500</td>
<td>1075</td>
<td>852</td>
<td>150</td>
<td>606</td>
<td>Async</td>
<td>800</td>
</tr>
<tr>
<td>Memory type</td>
<td>DRAM</td>
<td>DRAM</td>
<td>DRAM</td>
<td>DRAM</td>
<td>eDRAM</td>
<td>SRAM</td>
<td>SRAM</td>
</tr>
<tr>
<td>Max DNN model size</td>
<td>&lt;16G</td>
<td>&lt;3G</td>
<td>&lt;500M</td>
<td>&lt;500M</td>
<td>18M</td>
<td>256M</td>
<td>84M</td>
</tr>
<tr>
<td>Quantization Strategy</td>
<td>32-bit float</td>
<td>32-bit float</td>
<td>32-bit float</td>
<td>16-bit fixed</td>
<td>16-bit fixed</td>
<td>1-bit fixed</td>
<td>4-bit fixed</td>
</tr>
<tr>
<td>Area (mm²)</td>
<td>356</td>
<td>601</td>
<td>-</td>
<td>-</td>
<td>67.7</td>
<td>430</td>
<td>40.8</td>
</tr>
<tr>
<td>M×V Throughput (Frames/s)</td>
<td>162</td>
<td>4,115</td>
<td>173</td>
<td>33</td>
<td>36,900</td>
<td>1,191</td>
<td>82,000</td>
</tr>
<tr>
<td>Power(W)</td>
<td>140</td>
<td>250</td>
<td>8.0</td>
<td>9.63</td>
<td>15.97</td>
<td>0.23</td>
<td>0.59</td>
</tr>
<tr>
<td>Energy Efficiency (Frames/s/W)</td>
<td>1.16</td>
<td>20.5</td>
<td>21.6</td>
<td>3.4</td>
<td>2,310</td>
<td>5,178</td>
<td>138,983</td>
</tr>
</tbody>
</table>
Where are the savings from?

• **Three factors for energy saving:**

  • **Matrix is compressed by 35×;**
    less work to do; less bricks to carry

  • **DRAM => SRAM, no need to go off-chip: 120×;**
    carry bricks from Stanford to Berkeley => Stanford to Palo Alto

  • **Sparse activation: 3×;**
    lighter bricks to carry

Load Balancing and Scalability

Figure 8: Load efficiency improves as FIFO size increases. When the size is larger than eight, the marginal gain quickly diminishes. So we choose FIFO depth to be eight.

Figure 11: System scalability. The average efficiency of single PE finally decreases as the number of PEs increases. On some very sparse layers, having more PEs initially increases the efficiency a bit.

media coverage
EMERGENT CHIP VASTLY ACCELERATES DEEP NEURAL NETWORKS

December 8, 2015       Nicole Hemsoth

Stanford University PhD candidate, Song Han, who works under advisor and networking pioneer, Dr. Bill Dally, responded in a most soft-spoken and thoughtful way to the question of whether the coupled software and hardware architecture he developed might change the world.

Compressed representations in the age of big data

Emerging trends in intelligent mobile applications and distributed computing

By Ben Lorica, January 21, 2016

When developing intelligent, real-time applications, one often has access to a data platform that can wade through and unlock patterns in massive data sets. The back-end infrastructure for such applications often relies on distributed, fault-tolerant, scaleout technologies designed to handle large data sets. But, there are situations when compressed representations are useful and even necessary. The rise of mobile computing and sensors (IoT) will lead to devices and software that push computation from the cloud toward the edge. In addition, in-memory computation tends to be much faster, and thus, many popular (distributed) systems operate on data sets that can be cached.

To drive home this point, let me highlight two recent examples that illustrate the importance of efficient compressed representations: one from mobile computing, the other from a popular distributed computing framework.

Deep neural networks and intelligent mobile applications

In a recent presentation, Song Han, of the Concurrent VLSI Architecture (CVA) group at Stanford University, outlined an initiative to help optimize deep neural networks for mobile devices. Deep learning has produced impressive results across a range of applications in computer vision, speech, and machine translation. Meanwhile the growing popularity of mobile computing platforms means many mobile applications will need to have capabilities in these areas. The challenge is that deep learning

Like a “Limitless” Pill for Deep Neural Networks

Song Han may have done just that. Once an intern at Google, now a PhD candidate at Stanford University, Han is currently researching under the advisory of networking pioneer Dr Bill Dally. Under Dally’s guidance, Han is working to develop a small chip called EIE which is intended to increase the role of static random access memory (SRAM) while scaling down networks to more manageable and efficient sizes in a technique called deep compression – i.e. EIE is sort of like a “Limitless” pill for deep neural networks.

http://techemergence.com/a-limitless-pill-for-deep-neural-networks/
Hacker News

1. ▲ iOS 9.3 Preview (apple.com)
   88 points by jmduke 45 minutes ago | 58 comments

2. ▲ Why NSA Surveillance Scares Me (bentilly.blogspot.com)
   131 points by btilly 3 hours ago | 34 comments

3. ▲ The Latest Battle Over When and Where Kids Can Walk to School (citylab.com)
   27 points by jcater 1 hour ago | 4 comments

4. ▲ Open Guide to Equity Compensation (github.com)
   297 points by zalzal 5 hours ago | 52 comments

5. ▲ Emergent Chip Vastly Accelerates Deep Neural Networks (nextplatform.com)
   49 points by dharma1 2 hours ago | 7 comments

6. ▲ IBM ported Go to s390x mainframes (github.com)
   157 points by pythonist 4 hours ago | 49 comments

7. ▲ A new way police are surveilling: Calculating threat ‘score’ (washingtonpost.com)
   40 points by danso 2 hours ago | 26 comments

8. ▲ Google is Forcing Routebuilder to Shut Down (medium.com)
   275 points by jerry2 6 hours ago | 87 comments

9. ▲ David Bowie Has Died (hollywoodreporter.com)
   1394 points by hccampus 12 hours ago | 256 comments

10. ▲ Summon Your Tesla from Your Phone (teslamotors.com)
    24 points by nbaksalar 1 hour ago | 4 comments

https://news.ycombinator.com/item?id=10881683
Conclusion

• We present EIE, an energy-efficient engine optimized to operate on compressed deep neural networks.

• By leveraging sparsity in both the activations and the weights, EIE reduces the energy needed to compute a typical FC layer by 3,000×.

• With wrapper logic on top of EIE, 1x1 convolution and 3x3 convolution is possible.
Hardware for Deep Learning

PC                             Mobile                             Intelligent Mobile

Computation                      Mobile Computation                      Intelligent Mobile Computation
Recap

Model Compression
[1]. Han et al. “Learning both Weights and Connections for Efficient Neural Networks”, NIPS 2015

Hardware Acceleration

CNN Architecture Design Space Exploration
[4]. Iandola, Han, et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size” arXiv’16
[5]. Yao, Han, et.al, “Hardware-friendly convolutional neural network with even-number filter size” ICLR 2016 workshop
Thank you!

songhan@stanford.edu

[1]. Han et al. “Learning both Weights and Connections for Efficient Neural Networks”, NIPS 2015
[4]. Iandola, Han, et al. “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size” ECCV’16 submission
[5]. Yao, Han, et.al, “Hardware-friendly convolutional neural network with even-number filter size” ICLR’16 workshop