Building Energy-Efficient Accelerators for Deep Learning

Vivienne Sze & Yu-Hsin Chen
Eyeriss Project

Yu-Hsin Chen
PhD Candidate
MIT

Vivienne Sze
Principal Investigator
MIT

Joel Emer
Principal Investigator
MIT

Senior Distinguished Research Scientist
NVIDIA

Tushar Krishna
Postdoc
MIT

(currently)
Assistant Professor
Georgia Tech.
Future of Deep Learning

**Recognition**

**Self-Driving Cars**

**AI**

---

**Computation**

- **Deep CNN**
  - 15k – 300k OP/Px

- **Hand-Crafted Method***
  - 0.1k – 0.5k OP/Px

---

**Accuracy**

---

**DCNN Accelerator** is Crucial

- **High Throughput** for Real-time Processing
- **Sub-watt Power/Energy** Consumption

---

* Deformable Part Model
Deep CNN Explained

Modern Deep CNN: 5 – 100+ Layers

CONV Layer → Low-Level Features → CONV Layer → High-Level Features → FC Layers → Classes

Convolution → Activation → Norm. → Pooling
Convolution is the Most Important

Modern Deep CNN: 5 – 100+ Layers

CONV Layer \(\rightarrow\) Low-Level Features \(\rightarrow\) CONV Layer \(\rightarrow\) High-Level Features \(\rightarrow\) FC Layers \(\rightarrow\) Classes

Convolution

Takes 90% – 99% of Computation and Runtime
Convolution in CNN

Input Image

Output Image

Filter

R

H

E
Convolution in CNN

- **Input Image**
- **Output Image**
- **Filter**
  - dot product
  - partial sum accumulation
Convolution in CNN

Many Input Channels

Input Image

Output Image
Convolution in CNN

Many Filters

Input Image

Output Image

Many Output Channels
Convolution in CNN

Filters

Many Input Images

Many Output Images

Many Input Images

Many Output Images
### AlexNet\(^1\) Convolutional Layer Configurations

<table>
<thead>
<tr>
<th>Layer</th>
<th>Filter Size (R)</th>
<th># Filters (M)</th>
<th># Channels (C)</th>
<th>Stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11x11</td>
<td>96</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5x5</td>
<td>256</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3x3</td>
<td>384</td>
<td>256</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3x3</td>
<td>384</td>
<td>192</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3x3</td>
<td>256</td>
<td>192</td>
<td>1</td>
</tr>
</tbody>
</table>

Layer 1: 105M MACs*
Layer 2: 224M MACs*
Layer 3: 150M MACs*

1. Krizhevsky, NIPS 2012

* per frame. MAC = Multiply and Accumulate
## Large Sizes with Varying Shapes

### AlexNet Convolutional Layer Configurations

<table>
<thead>
<tr>
<th>Layer</th>
<th>Filter Size (R)</th>
<th># Filters (M)</th>
<th># Channels (C)</th>
<th>Stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11x11</td>
<td>96</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5x5</td>
<td>256</td>
<td>48</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3x3</td>
<td>384</td>
<td>256</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3x3</td>
<td>384</td>
<td>192</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>3x3</td>
<td>256</td>
<td>192</td>
<td>1</td>
</tr>
</tbody>
</table>

- A large amount of **computation** in each layer
- A large amount of **data accesses to memory**
- **Adaptive processing** required for different shapes

---

1. Krizhevsky, NIPS 2012
Properties We Can Leverage

- Operations exhibit high parallelism → high throughput possible
- Data reuse opportunities → exploit low-cost memory
Architecture
Highly Parallel Compute Paradigms

Temporal Architecture (SIMD/SIMT)

Memory Hierarchy
Register File

Control

Spatial Architecture (Dataflow Processing)
Highly Parallel Compute Paradigms

Temporal Architecture (SIMD/SIMT)

Spatial Architecture (Dataflow Processing)
Highly Parallel Compute Paradigms

Temporal Architecture (SIMD/SIMT)

- Memory Hierarchy
- Register File
- Control

Spatial Architecture (Dataflow Processing)

- Memory Hierarchy
- Processing Engine (PE)
Highly Parallel Compute Paradigms

Temporal Architecture (SIMD/SIMT)

Adaptive Configuration with autonomous local control

Spatial Architecture (Dataflow Processing)

Memory Hierarchy
Highly Parallel Compute Paradigms

Temporal Architecture (SIMD/SIMT)

Adaptive Configuration with autonomous local control

Efficient Data Reuse thru. distributed local storage

Spatial Architecture (Dataflow Processing)

Memory Hierarchy
Highly Parallel Compute Paradigms

Temporal Architecture (SIMD/SIMT)

Adaptive Configuration with autonomous local control

Efficient Data Reuse thru. distributed local storage

Natural Dataflow Mapping in-place data consumption

Spatial Architecture (Dataflow Processing)

Memory Hierarchy
How to Map the Dataflow?

CNN Convolution

Spatial Architecture
(Dataflow Processing)

Memory Hierarchy
Dataflow Mapping
Moving Data is Expensive

Data Movement Energy Cost

- Off-Chip DRAM
- Buffer
- PE
- RF
- ALU

<table>
<thead>
<tr>
<th>DRAM</th>
<th>ALU</th>
<th>Buffer</th>
<th>PE</th>
<th>ALU</th>
<th>PE</th>
<th>ALU</th>
<th>PE</th>
<th>ALU</th>
<th>PE</th>
<th>ALU</th>
<th>ALU</th>
</tr>
</thead>
</table>

- 200×
- 6×
- 2×
- 1×
- 1× (Reference)
Moving Data is Expensive

- Reuse input data (Filter/Image) in local memories
- Keep partial sum accumulation local too

Data Movement Energy Cost:
- DRAM → ALU: 200x
- Buffer → ALU: 6x
- PE → ALU: 2x
- RF → ALU: 1x
- ALU → ALU: 1x (Reference)
Energy-Efficient CNN Dataflow

Processing **9 MACs** within the same PE
Energy-Efficient CNN Dataflow

Processing **9 MACs** within the same PE

- **Weight Stationary**: Max filter weight reuse
  
  # Data Touches: \(1 + 9 + 9 = 19\)
Energy-Efficient CNN Dataflow

Filter \( \times \) Input Image = Output Image

Processing 9 MACs within the same PE

- **Weight Stationary**: \# Data Touches = 19
- **Output Stationary**: Max partial sum accumulation
  \# Data Touches: \( 9 + 9 + 1 = 19 \)
Energy-Efficient CNN Dataflow

Processing 9 MACs within the same PE

• Weight Stationary: \# Data Touches = 19
• Output Stationary: \# Data Touches = 19
• Row Stationary: balance reuse of all data types
  \# Data Touches: $3 + 5 + 3 = 11$
### CNN Dataflows Comparison

<table>
<thead>
<tr>
<th>CNN Dataflows</th>
<th>Norm. Energy*</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Local Reuse</td>
<td>2.1×</td>
</tr>
<tr>
<td>Weight Stationary</td>
<td>1.9×</td>
</tr>
<tr>
<td>Output Stationary</td>
<td>1.5×</td>
</tr>
<tr>
<td>Row Stationary</td>
<td>1×</td>
</tr>
</tbody>
</table>

* 256 PEs with the same total memory area

---

Exploit Data Statistics
Zero Compression Saves DRAM BW

Apply Activation (ReLU) on Filtered Image Data

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>-1</td>
<td>-3</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>-5</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>-2</td>
<td>6</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

ReLU

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

DRAM Access (MB)

Uncompressed Filters + Images

Compressed Filters + Images

AlexNet Conv Layer

1.2x 1.4x 1.7x 1.8x 1.9x
Zero Data Processing Gating

- Skip PE local memory access
- Skip MAC computation
- Save PE processing power by 45%

No R/W

No Switching

Register File

Zero Buff

Enable
Eyeriss Accelerator
Eyeriss DCNN Accelerator System

Link Clock, Core Clock

Eyeriss Accelerator

12×14 PE Array

Filter

Input Image

Decomp

Global Buffer

108KB

Comp

ReLU

Output Image

Off-Chip DRAM

64 bits

Filt

Img

Psum

Psum

Eyeriss Accelerator System

34 of 39
## Chip Spec & Measurement Results

<table>
<thead>
<tr>
<th>Technology</th>
<th>TSMC 65nm LP 1P9M</th>
</tr>
</thead>
<tbody>
<tr>
<td>On-Chip Buffer</td>
<td>108 KB</td>
</tr>
<tr>
<td># of PEs</td>
<td>168</td>
</tr>
<tr>
<td>Scratch Pad / PE</td>
<td>0.5 KB</td>
</tr>
<tr>
<td>Core Frequency</td>
<td>100 – 250 MHz</td>
</tr>
<tr>
<td>Peak Performance</td>
<td>33.6 – 84.0 GOPS</td>
</tr>
<tr>
<td>Word Bit-width</td>
<td>16-bit Fixed-Point</td>
</tr>
<tr>
<td>Natively Supported</td>
<td></td>
</tr>
<tr>
<td>CNN Shapes</td>
<td></td>
</tr>
<tr>
<td>Filter Width:</td>
<td>1 – 32</td>
</tr>
<tr>
<td>Filter Height:</td>
<td>1 – 12</td>
</tr>
<tr>
<td>Num. Filters:</td>
<td>1 – 1024</td>
</tr>
<tr>
<td>Num. Channels:</td>
<td>1 – 1024</td>
</tr>
<tr>
<td>Horz. Stride:</td>
<td>1–12</td>
</tr>
<tr>
<td>Vert. Stride:</td>
<td>1, 2, 4</td>
</tr>
</tbody>
</table>

AlexNet* Throughput vs. Power

* CONV layers only
## Comparison with GPU

<table>
<thead>
<tr>
<th></th>
<th>Eyeriss</th>
<th>NVIDIA TK1 (Jetson Kit)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Technology</strong></td>
<td>65nm</td>
<td>28nm</td>
</tr>
<tr>
<td><strong>Clock Rate</strong></td>
<td>200MHz</td>
<td>852MHz</td>
</tr>
<tr>
<td><strong># Multipliers</strong></td>
<td>168</td>
<td>192</td>
</tr>
<tr>
<td><strong>On-Chip Storage</strong></td>
<td>Buffer: 108KB</td>
<td>Shared Mem: 64KB</td>
</tr>
<tr>
<td></td>
<td>Spad: 75.3KB</td>
<td>Reg File: 256KB</td>
</tr>
<tr>
<td><strong>Word Bit-Width</strong></td>
<td>16b Fixed</td>
<td>32b Float</td>
</tr>
<tr>
<td><strong>Throughput</strong></td>
<td>34.7 fps</td>
<td>68 fps</td>
</tr>
<tr>
<td><strong>Measured Power</strong></td>
<td>278 mW</td>
<td>Idle/Active²: 3.7W/10.2W</td>
</tr>
</tbody>
</table>

1. AlexNet Convolutional Layers
2. Board Power
Image Classification on Eyeriss

[ISSCC 2016] Paper 14.5: Eyeriss Caffe Demo

This demo shows Caffe running with the MIT Eyeriss chip for Image Classification.

AlexNet: Krizhevsky, NIPS 2012
Summary

- **Eyeriss**: a reconfigurable accelerator for state-of-the-art deep CNNs at below 300mW

- **Reduce data movement & exploit data statistics** to achieve high energy efficiency

- **Integrated** with the **Caffe DL framework** and demonstrated an image classification system

Want to know more? Visit our project page:

Acknowledgement: funded by DARPA YFA, MIT CICS and a gift from Intel