Conv-RAM: An Energy-Efficient SRAM with In-Memory Convolution Computation for Low-Power Machine Learning Applications

Avishek Biswas, Anantha P. Chandrakasan

Massachusetts Institute of Technology
Motivation: Machine Learning for IoT

AI revolution & ML

- Face Recognition
- Speech Recognition
- Image Classification
- Object Detection

ML for IoT: *Edge Computing*

- Faster local decisions
- Less communication to cloud
- More secure (local data)

**Requirements:**

- Low power consumption
- Real-time processing

AI: Artificial Intelligence
ML: Machine Learning
CNNs: Accuracy vs. Energy

Convolutional Neural Networks (CNNs) provide state-of-the-art accuracy in a wide variety of AI tasks:

- CAT Classification
- CAT Classification + Localization
- CAT, DOG, DUCK Object Detection
- Speech recognition + translation

However, they are very computation intensive and energy hungry!
→ Not suitable for energy-constrained applications e.g. edge computing
Basic Operation for CNNs: Dot Product

Basic operation can be simplified to a **dot product** or **multiply-accumulate (MAC):**

\[ Y_k = \sum_i W_{k,i} \times X_i \]

\( k : \# \text{ of 3D filters in a layer} \)
Binary-Weight CNNs*: Size ↓

Simplification: Filter weights binary ‘±1’ (with a common scaling factor: $\alpha$)

<table>
<thead>
<tr>
<th>Type</th>
<th>Network Variation</th>
<th>Memory Savings (Inference)</th>
<th>Classification Accuracy†</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standard Convolution</td>
<td></td>
<td>1x</td>
<td>80.2%</td>
</tr>
<tr>
<td>Real-valued Inputs: $X$</td>
<td>0.11 -0.21 .... 0.34</td>
<td>0.12 -1.20 .... 0.41</td>
<td></td>
</tr>
<tr>
<td>Real-valued Weights: $W$</td>
<td>-0.25 0.61 .... 0.52</td>
<td>-0.20 0.50 .... 0.68</td>
<td></td>
</tr>
<tr>
<td>Binary-Weight Convolution*</td>
<td></td>
<td>~32x</td>
<td>79.4%</td>
</tr>
<tr>
<td>Real-valued Inputs: $X$</td>
<td>0.11 -0.21 .... 0.34</td>
<td>+1 -1 .... +1</td>
<td></td>
</tr>
<tr>
<td>Binary Weights: $w$</td>
<td>-0.25 0.61 .... 0.52</td>
<td>-1 +1 .... +1</td>
<td></td>
</tr>
</tbody>
</table>

*M. Rastegari et al. Arxiv ’16

$Y_k = \sum_i W_{k,i} \times X_i = \alpha_k \sum_i w_{k,i} \times X_i$

MAC ($\times, \pm$) replaced by add/sub ($\pm$)

†AlexNet: <1% reduction in accuracy
Memory Access Bottleneck in CNNs

Conventional all-digital implementation

Huge amount of data transfer to/from memory

Memory access & data movement energy $\gg$ Computation (ALU) energy!!
Energy Costs for Memory vs Compute

<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integer ADD (8b)</td>
<td>0.03</td>
</tr>
<tr>
<td>Integer ADD (16b)</td>
<td>0.05</td>
</tr>
<tr>
<td>Integer ADD (32b)</td>
<td>0.1</td>
</tr>
<tr>
<td>Integer MULT (8b)</td>
<td>0.2</td>
</tr>
<tr>
<td>Integer MULT (32b)</td>
<td>3.1</td>
</tr>
<tr>
<td>8KB SRAM Read (32b)</td>
<td>5</td>
</tr>
<tr>
<td>32KB SRAM Read (32b)</td>
<td>10</td>
</tr>
<tr>
<td>1MB SRAM Read (32b)</td>
<td>50</td>
</tr>
</tbody>
</table>

In 45nm CMOS at 0.9V

"Computing’s Energy Problem (and what we can do about it)", M. Horowitz, ISSCC 2014

Memory access energy >> Computation energy!!
Alternative: Make Memory Pro-active in Computation

Embedding computation inside memory
- Less data transfer ⇒ More Energy-efficient
- Massively parallel operations ⇒ Higher Bandwidth

Conventional all-digital implementation

Memory
Buffer
Compute
Input ➔ Output

Huge amount of data transfer to/from memory

Memory with Embedded Computation
Buffer
Input ➔ Output

No explicit data read

Proposed Memory-embedded implementation
Outline

• Concept of SRAM Embedded Compute
  • Conv-SRAM Architecture & Key Features
  • Conv-SRAM Operation
  • Measurement Results
Binary-Weight CONV as Averaging in SRAMs

$Y_k = \sum_i W_{k,i} \times X_i = \alpha_k \sum_i w_{k,i} \times X_i$

$= \frac{M_k}{N} \sum_i w_{k,i} \times X_i$

$\forall k \in \mathbb{Z} \backslash \{0\}, M_k, N \in \mathbb{I}$

$Y_{OUT,k} = \frac{1}{N} \sum_i w_{k,i} \times X_{IN,i}$

Y: Convolution Output || W: Filter Weight || X: Convolution Input

Binarization

Effective convolution output
Binary-Weight CONV as Averaging in SRAMs

\[ Y_{OUT,k} = \frac{1}{N} \sum_i w_{k,i} \times X_{IN,i} \]

\[ V_{Y\_AVG,k} = \frac{1}{N} \sum_i w_{k,i} \times V_{a,i} \]

**Digital Domain**

**Analog Domain**

**Y_{OUT}:** Convolution Output || **w:** Binary Filter Weight || **X_{IN}:** Convolution Input

**1 DAC**

**2 MAV**

**3 ADC**

**MAV:** Multiply-and-Average
Outline

• Concept of SRAM Embedded Compute
• Conv-SRAM Architecture & Key Features
• Conv-SRAM Operation
• Measurement Results
Overall Architecture: Conv-SRAM

- Conv-SRAM stores binary filter weights in 16 local arrays
  - Each local array ↔ Different filter in a CNN layer
Processes 64 Conv inputs ($X_{IN}$) and 16 Conv outputs ($Y_{OUT}$) in parallel.
Key Features of This Work

1. Robust to SRAM bit-cell $V_t$-variation

⇒ Weight multiplication using full-swing locally, unlike [4,5]
⇒ Bit-cell $V_t$-variations don’t affect computation accuracy

Mitigation of SRAM Bit-cell $V_t$-variation

Bit-cell discharge current:

$I_{cell} \propto (V_{WL} - V_t)$

$V_a \propto I_{cell}$

Conventional [4]

$V_a$ varies widely due to $I_{cell}$ variation

Proposed

$V_a$ has no variation due to $I_{cell}$

*Assumption: Both DACs are ideal

Directly apply $V_a$ on GBL(LBL) by DAC

$V_a \propto I_{cell}$
Key Features of This Work

2. Improved dynamic range without any bit-cell disturb issue

**Conventional [5]**
Multi-row read using shared global bit-lines

**Proposed**
Multi-column read using independent local bit-lines and 10T bit-cells

- No write disturb
- Very wide range of bit-line voltage for analog computation

Bit-line dynamic voltage range is limited to prevent write disturb

Write disturb of this 6T bit-cell if $V_{GBLT}$ is low
Key Features of This Work

1. Robust to SRAM bit-cell $V_t$-variation
2. Improved dynamic range without any bit-cell disturb issue
3. Inherent SRAM bit-line $\text{cap}$ for analog computation
   ⇒ Less impact of variation (vs. $V_t$ based approaches)
   ⇒ Less area overhead vs. [5] (no extra cap)
4. Supports multi-bit output precision
   ⇒ Better classification accuracy vs. [4] (1bit output)

Outline

• Concept of SRAM Embedded Compute
• Conv-SRAM Architecture & Key Features
• Conv-SRAM Operation
  ❑ Phase 1: DAC
  ❑ Phase 2: MAV
  ❑ Phase 3: ADC
• Measurement Results
Conv-SRAM Operation Phase 1: DAC

Digital conv input ($X_{IN}$) to analog voltage ($V_a$) conversion

\[ Y_{OUT,k} = \frac{1}{N} \sum_{i} w_{k,i} \times X_{IN,i} \]
\[ V_{Y_AVG,k} = \frac{1}{N} \sum_{i} w_{k,i} \times V_{a,i} \]
Global Bit-Line DAC Circuit

For better linearity of $X_{IN}$ vs $V_{GRBL}$, single ON pulse needed for all $X_{IN}$ codes
Digital-to-Time Conversion

Share 8:1 mux and timing signals in 2 charging phases:
- **A** (3 MSBs), **B** (3 LSBs)

**Digital Inputs**

**Phase A**
- $X_{IN}[5:3] = 63$ or '111 111'
- $X_{IN}[2:0] = 0, 1, ..., 7$

**Phase B**
- $X_{IN}[5:3] = 24$ or '011 000'
- $X_{IN}[2:0] = 0, 1, ..., 7$

**Select Signal for 2 Phases**

$$X_{IN} = 8 \times k_A + k_B$$

- $k_A = \text{Dec}(X_{IN}[5:3])$
- $k_B = \text{Dec}(X_{IN}[2:0])$

**Shared Timing Signals**

$TD_{9k} = 8 \times kt_0 + kt_0$

$t_0$: min. time resolution
Global Bit-Line DAC Operation

**Global Timing Signals**

Same PMOS's for all codes & global timing signals
⇒ more robust to variation than binary-weighted PMOS DACs [4]
Outline

• Concept of SRAM Embedded Compute
• Conv-SRAM Architecture & Key Features
• Conv-SRAM Operation
  o Phase 1: DAC
  o Phase 2: MAV
  o Phase 3: ADC
• Measurement Results
Conv-SRAM Operation Phase 2: MAV

Row-wise 1-bit weight ($w$) multiply and average (MAV)

\[
Y_{OUT,k} = \frac{1}{N} \sum_i w_{k,i} \times X_{IN,i}
\]

\[
V_{Y\_AVG,k} = \frac{1}{N} \sum_i w_{k,i} \times V_{a,i}
\]
Row-wise Multiply & Average Architecture

- 1-bit weight multiplication by SRAM bit-cell through local bit-line discharge
- \( Q = '0' \) for \( w = +1 \) / ‘1’ for \( w = -1 \)

\[
\Delta V_{LBL,i} = V_{LBL,i} - V_{LBF,i} = V_{a,i} \times w_i
\]
Row-wise Multiply & Average Architecture

- Charge-sharing of local bit-lines implements averaging
- Separate voltage rails for ‘+ve’ ($V_{p_AVG}$) and ‘-ve’ ($V_{n_AVG}$) parts of the average
Variation of SRAM Local BL discharge time

- Local BL discharge time (i.e. weight evaluation time) has small variation
- Even at SS corner: \( t_{\text{dis,LBL}} (6\sigma) \approx 500 \text{ ps} \ll t_{\text{clk}} (\sim 100\text{ns}) \)
- Bit-cell variations do not dominate computation time

\[
\mu \approx 296\text{ps} \\
\sigma \approx 26\text{ps}
\]

\( V_{BL} = V_{WL} = 1\text{V} \)

SS 25°C

*massively parallel in-memory compute mode*
Outline

• Concept of SRAM Embedded Compute
• Conv-SRAM Architecture & Key Features
• Conv-SRAM Operation
  - Phase 1: DAC
  - Phase 2: MAV
  - Phase 3: ADC
• Measurement Results
Conv-SRAM Operation Phase 3: ADC

Analog average voltage ($V_{Y\_AVG}$) to digital conv output ($Y_{OUT}$) conversion

\[ Y_{OUT,k} = \frac{1}{N} \sum_{i} w_{k,i} \times X_{IN,i} \]

\[ V_{Y\_AVG,k} = \frac{1}{N} \sum_{i} w_{k,i} \times V_{a,i} \]

\[ V_{Y\_AVG} \triangleq V_{pAVG} - V_{nAVG} \]
Charge-sharing based ADC Architecture

- Integrating ADC architecture to save area, energy ($\alpha |Y_{OUT}|$)
- Charge-sharing using replica bit-lines for better matching and less impact of variation

Local SRAM Array

Charge-sharing + integration

ADC

Convolution output $Y_{OUT}$

LeNet-5 CNN Conv Layer #2

\[ \mu(|Y_{OUT}|) \sim 1.3 \]

Full scale: $\pm 31$
Charge-sharing based ADC Operation

\[ \Delta V_{ADC} \approx \frac{V_{dd}}{N} \]

From ADC logic

To ADC logic

Local SRAM Array

Local MAV\(_a\)

Charge-sharing + integration

ADC Logic

Convolution output \(Y_{OUT}\)

Comparison Integrate EOC

N: # SRAM Cols
Outline

• Concept of SRAM Embedded Compute
• Conv-SRAM Architecture & Key Features
• Conv-SRAM Operation
• Measurement Results
# Test-chip Photo and Summary

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Technology</td>
<td>65nm</td>
</tr>
<tr>
<td>CSRAM size</td>
<td>16Kb</td>
</tr>
<tr>
<td>CSRAM area</td>
<td>0.063mm²</td>
</tr>
<tr>
<td>Array organization</td>
<td>256×64 (10T bit-cells)</td>
</tr>
<tr>
<td># column DAC’s</td>
<td>64</td>
</tr>
<tr>
<td># row ADC’s</td>
<td>16</td>
</tr>
<tr>
<td>Max. # MAV’s</td>
<td>64×16</td>
</tr>
<tr>
<td>Supply voltages</td>
<td>1V (main), 1.2V (DAC), 0.8V (array)</td>
</tr>
<tr>
<td>Main clock freq. (compute mode)</td>
<td>5MHz</td>
</tr>
<tr>
<td>ADC clock freq.</td>
<td>250MHz</td>
</tr>
</tbody>
</table>
Measured GBL_DAC Performance

- 64 DACs averaged to estimate $V_a$
- 1 time calibration to set $V_{a,\text{max}} \sim 1\text{V}$ (for $X_{IN} = 31$; 5b mode)
- Good linearity, $DNL < 1\text{LSB}$
• Good linearity between $X_{IN}$ and $Y_{OUT}$ for both $w: \pm 1$
• Energy of CSH_ADC scales linearly with the output $Y_{OUT}$ value
Test Case: MNIST Digit Classification

**LeNet-5 CNN** with 2 convolution \((C1, C3)\) and 2 fully-connected \((F5, F6)\) layers
Classification Test Examples (Measured)

All CNN layers implemented using Conv-SRAM
Classification Accuracy of 97.4% (v1), 98.4% (v2), after 4 layers with 10000 test images
Measured Energy for CONV/FC Layers

- Peak energy-efficiency of 38.8 TOPS/W (v1) for layer C3, running 50x16x2 operations/cycle
## Comparison with Prior Work for MNIST

<table>
<thead>
<tr>
<th></th>
<th>Tech. (nm)</th>
<th>ML Model</th>
<th>Comp. Mode</th>
<th>Classification Accuracy</th>
<th>Input/Filter</th>
<th>SRAM Size (KB)</th>
<th>Throughput (GOPS)</th>
<th>Energy Eff. (TOPS/W)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>This work</strong></td>
<td>65</td>
<td>CNN</td>
<td>Analog</td>
<td>98.35%</td>
<td>7b/1b</td>
<td>2</td>
<td>8</td>
<td>38.8*</td>
</tr>
<tr>
<td><strong>ISSCC ’17 [1]</strong></td>
<td>65</td>
<td>FC DNN</td>
<td>Digital</td>
<td>98.36%</td>
<td>8b/8b</td>
<td>1024</td>
<td>10.7</td>
<td>1.86</td>
</tr>
<tr>
<td><strong>VLSI ’16 [2]</strong></td>
<td>40</td>
<td>CNN</td>
<td>Digital</td>
<td>98%</td>
<td>6b/4b</td>
<td>144</td>
<td>102</td>
<td>1.75</td>
</tr>
<tr>
<td><strong>JSSC’18 [5]</strong></td>
<td>65</td>
<td>k-NN</td>
<td>Analog</td>
<td>92%</td>
<td>6b/8b</td>
<td>16</td>
<td>10.2</td>
<td>0.98</td>
</tr>
<tr>
<td><strong>JSSC’18 [3]</strong></td>
<td>65</td>
<td>FC NN</td>
<td>Digital</td>
<td>90%</td>
<td>1b/1b</td>
<td>102</td>
<td>1380</td>
<td>2.3</td>
</tr>
</tbody>
</table>

*Does not include input/output memory access energy

More than **16× improvement** in energy-efficiency, with similar classification accuracy as digital implementations.

1 MAV, MAC, SAD = 2 OPs.
Conclusions

• Demonstrated on silicon **SRAM-embedded convolution computation** for binary-weight CNNs with MNIST dataset

• **Variation-tolerant architecture** and **multi-bit output resolution** improve classification accuracy

• **More than 16× improvement in energy-efficiency** vs. prior work, due to less data transfer and in-place computations

• Potential for **low-power, real-time machine learning applications** for “smart” IoT devices
Acknowledgements

• Intel for funding

• Prof. Vivienne Sze and Prof. Harry Lee for helpful technical discussions

Thank you for your attention!

Questions?
References

[1] P. Whatmough et al., ISSCC 2017
[3] K. Ando et al., JSSC 2018
[5] M. Kang et al., JSSC 2018