CHIMERA: Efficient DNN Inference and Training at the Edge with On-Chip Resistive RAM

Kartik Prabhu

Machine Learning at the Edge

Benefits
- private
- responsive
- user-specific
- low-power

Challenges
- on-device training
- energy-efficient
- high throughput
- large DNN model sizes

Challenging with today’s memory technologies
(SRAM, DRAM, Flash)
On-Chip Non-Volatile Resistive RAM (RRAM)

Voltage controlled resistance

Bit cell and resistance state

Top electrode
Conductive filament
Bottom electrode

Low 0
High 1

[Wong and Salahuddin, Nat. Nanotech ‘15]
On-Chip Non-Volatile Resistive RAM (RRAM)

**Benefits**
- Non-volatile
- High density ($53F^2$)
- Low read energy (8 pJ/Byte)
- High bandwidth (6 Gbits/s)

**Challenges**
- High write energy (40.3 nJ/Byte)
- High write latency (0.4 μs/Byte)
- Low endurance (10k-100k writes)

[Chou et al., ISSCC '18]
Key Contributions

**RRAM-Optimized ML SoC**

- 2 MB On-Chip RRAM with DNN Accelerator

**Illusion System for Scale Out**

- C1, C2, C3, C4, C5
- ≤5% Energy and Latency Overheads

**First On-Device RRAM Training**

- RRAM-Aware Training Algorithm (LRT)
- Accuracy (%)
  - On-device incremental training
  - Previous learned classes

Stanford University
Key Contributions

RRAM-Optimized ML SoC

- 2 MB On-Chip RRAM with DNN Accelerator

Illusion System for Scale Out

- ≤5% Energy and Latency Overheads

First On-Device RRAM Training

- RRAM-Aware Training Algorithm (LRT)

Stanford University
CHIMERA SoC

- 2 MBytes foundry RRAM
  - DNN weights
  - CPU instructions
- 512 KBytes SRAM
- DNN accelerator
  - 0.92 TOPS
- 64b RISC-V CPU
- 2 chip-to-chip (C2C) links
  - Inter-chip communication
**Design Space Exploration Methodology**

**Design Space Exploration**
- Neural Network Specification
- Design Space Specification
- Component Energy Models (MAC, RF, Memories)
- Architectural Design Space Exploration (Interstellar)
- Optimal Architectural Parameters (# MACs, Memory sizes, etc.)

**Hardware**
- Parameterized DNN Accelerator HLS Code (Catapult HLS)
- DNN Accelerator RTL
- Integration into RISC-V SoC
- VLSI Flow

**Software**
- Quantization
- Optimal Loop Tiling (Interstellar)
- Low Level Accelerator Driver Code
- Power & Performance Analysis

Stanford University
DNN Accelerator

Stanford University
Conventional DRAM-based DNN Accelerator

[Yang et al., ASPLOS ‘20]
1. On-chip RRAM-based DNN Accelerator

- **DNN weights loaded directly from RRAM**
- **Off-chip SRAM weight buffer removed**

Buffers sized for min. energy

[Yang et al., ASPLOS ‘20]
1. On-chip RRAM-based DNN Accelerator

DNN weights loaded directly from RRAM

SRAM weight buffer removed

2x smaller input and accumulation buffers

Buffers sized for min. energy

Input Double Buffer

16x16 Systolic Array

Accumulation Buffer

Input Double Buffer

Accumulation Buffer

Energy (μJ)
2. PE Filter Unrolling to Reduce Input Buffer

Conventional accelerators only unroll input and output channels.

**Processing Element**

- **Inputs**
- **Weights**

- Multiply
- Accumulate

**16x16 Systolic Array**

- PE

**Equations**

\[ X \cdot Y \cdot C_0 \cdot C_1 \]

**Graphs**

- Buffers sized for min. energy

**Stanford University**
2. PE Filter Unrolling to Reduce Input Buffer

Filter dimensions ($F_x=F_y=3$) are unrolled in the PE to reduce input buffer by 9x, while achieving same data reuse.
3. Stride Registers to Reduce Input Accesses

In-PE filter unrolling allows for immediate reuse of overlapped pixels during striding.

Buffers sized for min. energy

16x16 Systolic Array

Energy (μJ)

Capacity (MB)

Input Double Buffer

PE

Stanford University
3. Stride Registers to Reduce Input Accesses

Exploit horizontal input reuse
Reduce input buffer accesses and energy by 3x

Stride

Input Double Buffer

3x3 Input

Stride Register

16x16 Systolic Array

Buffers sized for min. energy

<table>
<thead>
<tr>
<th>Capacity (MB)</th>
<th>Energy (μJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.5</td>
<td>50</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>1.5</td>
<td>150</td>
</tr>
<tr>
<td>2</td>
<td>200</td>
</tr>
<tr>
<td>2.5</td>
<td>250</td>
</tr>
</tbody>
</table>

Weight Acc.

Input

RRAM

Unrolling

Stride reg.

Stanford University
ResNet-18 Results

Measured on ImageNet:
Energy: 8.1 mJ/image
Latency: 60 ms/image
Average power: 136 mW
Efficiency: 2.2 TOPS/W

Simulated Power Breakdown

- DA 69%
- MACs 16%
- Systolic Array Interconnect 19%
- Address gen. control 12%
- RRAM + Controller 16%
- Main SRAM 3%
- DA others 7%
- RISC-V 2%
- IO 6%
- Others 4%
- Input and Acc. Buffers 15%

With on-chip RRAM, total power dominated by DNN accelerator, NOT by memory accesses
CHIMERA\textsuperscript{v2} and CHIMERA\textsuperscript{vS}

- CHIMERA\textsuperscript{v2} implements several optimizations:
  - Increased RRAM to 4MBytes
  - Data prefetching
  - Loop replication
- CHIMERA\textsuperscript{vS} replaces CHIMERA\textsuperscript{v2}'s 4MB of RRAM with SRAM

<table>
<thead>
<tr>
<th></th>
<th>v1</th>
<th>v2</th>
<th>vS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weights Memory Capacity (MB)</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Weights Memory Area (mm\textsupersquare)</td>
<td>10.8</td>
<td>20.2</td>
<td>30.5</td>
</tr>
<tr>
<td>Total Die Area (mm\textsupersquare)</td>
<td>29.2</td>
<td>29.2</td>
<td>39.5</td>
</tr>
</tbody>
</table>
# ResNet-18 Inference Results

<table>
<thead>
<tr>
<th></th>
<th>v1</th>
<th>v2</th>
<th>vS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Energy (mJ)</td>
<td>8.1</td>
<td>1.8</td>
<td>1.5</td>
</tr>
<tr>
<td>Latency (ms)</td>
<td>60.0</td>
<td>15.2</td>
<td>15.1</td>
</tr>
<tr>
<td>Average Power (mW)</td>
<td>135</td>
<td>118</td>
<td>99</td>
</tr>
</tbody>
</table>

7.2x mainly from replication
2.8x mainly from pre-fetching
2.9x mainly from increased RRAM bandwidth
On-Chip RRAM vs Traditional SRAM

![Graph showing comparison between RRAM and SRAM models](chart)

- **4MB model**
  - Single chip
  - CHIMERAv2
    - 4MB RRAM
  - CHIMERAvS
    - 4MB SRAM
  - Flash

Inference Energy (mJ)

- **Shutdown + load from Flash**
- **CHIMERAvS**
- **CHIMERAv2**

Time between inferences (ms)

- **Server**: 75 ms, 13 fps
- **Edge**

Stanford University
On-Chip RRAM vs Traditional SRAM
RRAM-Optimized DNN Accelerator

1. On-chip RRAM overcomes memory access bottleneck

2. Low read-energy RRAM weights reduces SRAM buffer sizes and energy

3. New dataflow: filter unrolling for larger filters improves input reuse

= Efficient on-device inference
Key Contributions

RRAM-Optimized ML SoC

- 2x4 RRAM Macros (1 MB)
- RRAM Controllers
- LP SRAMs 512 KB
- RV IS/DS
- 64b RISC-V
- DNN Accelerator
- Chip-to-Chip link
- IO PADS
- 5.4 mm
- 5.4 mm
- 2 MB On-Chip RRAM with DNN Accelerator

Illusion System for Scale Out

- ≤5% Energy and Latency Overheads

First On-Device RRAM Training

- Samples
- Prev. learned classes
- On-device incremental training
- RRAM-Aware Training Algorithm (LRT)
CHIMERA: Scale Out to Larger DNNs

<table>
<thead>
<tr>
<th>DNN</th>
<th>Approach</th>
<th>Benefits from on-chip RRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤2MBytes</td>
<td>CHIMERA</td>
<td>✔</td>
</tr>
</tbody>
</table>

[Radway et al., Nat. Electronics ‘21]
## CHIMERA: Scale Out to Larger DNNs

<table>
<thead>
<tr>
<th>DNN</th>
<th>Approach</th>
<th>Benefits from on-chip RRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤2MBytes</td>
<td>CHIMERA</td>
<td>✓</td>
</tr>
<tr>
<td>≥2MBytes</td>
<td>Single CHIMERA, more RRAM</td>
<td>✓</td>
</tr>
</tbody>
</table>

“Dream Chip”: **Moving target** as DNNs continue growing

[Radway et al., *Nat. Electronics* ‘21]
## CHIMERA: Scale Out to Larger DNNs

<table>
<thead>
<tr>
<th>DNN</th>
<th>Approach</th>
<th>Benefits from on-chip RRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤2MBytes</td>
<td>CHIMERA</td>
<td>✓</td>
</tr>
<tr>
<td>≥2MBytes</td>
<td>Single CHIMERA, more RRAM</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>CHIMERA + Off-chip Memory (DRAM)</td>
<td>X</td>
</tr>
</tbody>
</table>

[Radway et al., Nat. Electronics ‘21]
CHIMERA: Scale Out to Larger DNNs

<table>
<thead>
<tr>
<th>DNN</th>
<th>Approach</th>
<th>Benefits from on-chip RRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤2MBytes</td>
<td>CHIMERA</td>
<td>✓</td>
</tr>
<tr>
<td>≥2MBytes</td>
<td>Single CHIMERA, more RRAM</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>CHIMERA + Off-chip Memory (DRAM)</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>Multiple CHIMERAAs in an Illusion System</td>
<td>✓</td>
</tr>
</tbody>
</table>

Creates “Illusion” of Dream, single chip

[Radway et al., Nat. Electronics ‘21]
CHIMERA Illusion System for 12MByte DNNs

Chip to Chip (C2C) links: (77 pJ/bit; 1.9 Gbits/s)

Power gating devices
RRAM Leads to Quick Chip Wakeup/Shutdown

SoC ready from wakeup in 33 $\mu$s

Full SoC shutdown in 10 ns

SoC instructions & DNN weights preserved through shutdown

**CHIMERAs on only when actively computing**
Minimizing Inter-Chip Messaging Overheads

Traditional: High message traffic to keep all chips occupied

CHIMERA: *Enough* weight RRAM for infrequent data transfers

**Illusion:** Map DNN to minimize inter-chip traffic
Mapping 1: Minimizes Energy/Inter-Chip Messages

Energy-efficient ResNet-18 mapping for sporadic inference

Minimal inter-chip traffic (373 KBytes vs. 12MByte DNN)
Mapping 2: Minimizes Latency

Low-latency ResNet-18 mapping for high-throughput inference
Higher inter-chip traffic (1.5 MBytes)

Input split, weights duplicated for parallel compute
CHIMERA Illusion System Measurements

Minimal energy mapping: ≤ 4% energy and ≤ 5% execution time

Normalized to the ResNet-18 per-layer total measured on CHIMERA
## Compared to Traditional MCMs

<table>
<thead>
<tr>
<th></th>
<th>CHIMERAv2</th>
<th>Simba</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Technology</strong></td>
<td>40 nm</td>
<td>16 nm</td>
</tr>
<tr>
<td><strong>Core Energy (mJ)</strong></td>
<td>5.18</td>
<td>16.3</td>
</tr>
<tr>
<td><strong>Single Chip Latency (ms)</strong></td>
<td>5.29*</td>
<td>4.64</td>
</tr>
<tr>
<td><strong>Multi Chip Throughput (inferences/ms)</strong></td>
<td>1.74**</td>
<td>2.06</td>
</tr>
</tbody>
</table>

*Scaled for iso-frequency
**Scaled for iso-MACs and frequency

[Shao et al., MICRO ‘19]
[Zimmer et al., VLSI ‘19]
[Zimmer et al., JSSC ‘20]
[Venkatesan et al., Hot Chips ‘19]
CHIMERA Illusion System

1. A fast, energy-efficient RRAM-optimized DNN accelerator

2. Quick chip wakeup/shutdown: < 58 μJ / 33 μs

3. Mappings with minimal messages: < 1/30\textsuperscript{th} of DNN weights

= Scalable inference system enabled by RRAM
Key Contributions

RRAM-Optimized ML SoC

- 2 MB On-Chip RRAM with DNN Accelerator

Illusion System for Scale Out

- ≤5% Energy and Latency Overheads

First On-Device RRAM Training

- RRAM-Aware Training Algorithm (LRT)

Stanford University
Training DNN Off-Line

**Generic classes** (abundant data)
- 1M images
- Full ImageNet1000 dataset

**User-specific classes** (limited data)
- 1 image/class
- Subset of Flowers102

DNN model
- Feature extractor
- Fully connected layer
- Classifier

Stanford University
On-Device Incremental Training

**Generic classes** (limited data)

6 images/class

Random ImageNet images

**User-specific classes** (abundant data)

60 image/class

Full Flowers102 dataset

DNN model deployed at the edge

Feature extractor

Only fully connected layer is updated

Fully connected layer

Classifier

Stanford University
## On-device Training Challenges with RRAM

### SGD challenging on RRAM-based edge-devices

<table>
<thead>
<tr>
<th>Batch = 1</th>
<th>Large Batches</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Green" /></td>
<td>Immediate classification</td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Red" /></td>
<td>Frequent weight updates</td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Red" /></td>
<td>Fixed-point training difficult</td>
</tr>
</tbody>
</table>

### Our RRAM-aware Low-Rank Training (LRT) algorithm

<table>
<thead>
<tr>
<th>Batch = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Green" /></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Green" /></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Green" /></td>
</tr>
<tr>
<td><img src="https://via.placeholder.com/15" alt="Green" /></td>
</tr>
</tbody>
</table>

---

**SGD:** Stochastic Gradient Descent

On-Device Training

Images

Feature extraction

1
2
3
i
i+1

Features

Correct class

Fully connected layer

Classes
probabilities

Gradients
Low-Rank Training (LRT): Initialization

Features

Gradients

Rank

F

G
Low-Rank Training (LRT): New Sample

Append new sample’s features and gradients

Features
G

Gradients
F

Rank + 1

Features

Gradients
Low-Rank Training (LRT): QR Decomposition

Decomposition reduces matrix dimensions **much smaller than** number of features/gradients

Features

\[
\begin{align*}
F & \rightarrow Q_F \cdot R_F \\
\text{Features} & \quad \text{Rank + 1}
\end{align*}
\]

Gradients

\[
\begin{align*}
G & \rightarrow Q_G \cdot R_G \\
\text{Gradients} & \quad \text{Rank + 1}
\end{align*}
\]
Low-Rank Training (LRT): QR Decomposition

Features

Gradients

Rank + 1
LRT: Singular Value Decomposition

Singular Value Decomposition (SVD) finds most significant components

\[ F = Q_F \cdot R_F \]

\[ G = Q_G \cdot R_G \]

\[ R_F R_G^T \approx U \cdot \begin{pmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_{\text{Rank} + 1} \end{pmatrix} \cdot V \]

Rank + 1

Stanford University
LRT: Dimensionality Reduction

Singular Value Decomposition (SVD) finds most significant components

Dimensionality reduction

Minimal loss of information
LRT: SRAM Update

LRT compresses gradients stored in small SRAM (6-22 Kbytes depending on Rank)

\[
\begin{align*}
F &= Q_F \cdot R_F \\
G &= Q_G \cdot R_G \\
F' &= U \cdot R_{F}R_{G}^T \\
G' &= V
\end{align*}
\]
LRT: Repeat for Multiple Images

SRAM for compressed gradient does not increase with set size

Up to 1024 images in update set
LRT: RRAM Weight Update

At end of sample set:
expand features & gradients to update RRAM weights

\[ W = W - \alpha (F \cdot G^T) \]

\( \alpha = \) learning rate

Larger set \( \rightarrow \) fewer RRAM updates
Optimal Rank and Set-Size Search

Larger rank -> more computation, but larger set size

LRT: 340x lower EDP and 101x fewer RRAM weight updates

All lines iso-SGD accuracy
### Optimizing Training at the Edge

<table>
<thead>
<tr>
<th></th>
<th>Weight update every sample</th>
<th>Batch weight update</th>
<th>Store Input and output FC</th>
<th>Low Rank Training</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Capacity</strong></td>
<td>$m \cdot n \sim 1E6$</td>
<td>$m \cdot n \sim 1E6$</td>
<td>$B \cdot (m + n) \sim 2E5$</td>
<td>$R \cdot (m + n) \sim 1E3$</td>
</tr>
<tr>
<td><strong>Weight updates per sample</strong></td>
<td>$m \cdot n \sim 1E6$</td>
<td>$m \cdot n / B \sim 1E4$</td>
<td>$m \cdot n / B \sim 1E4$</td>
<td>$m \cdot n / B \sim 1E4$</td>
</tr>
<tr>
<td><strong>SRAM writes/sample</strong></td>
<td>Not needed</td>
<td>Not needed</td>
<td>$(m + n) \sim 2E3$</td>
<td>$R \cdot (m + n) \sim 2E3$</td>
</tr>
<tr>
<td><strong>Ops/sample</strong></td>
<td>$1E6$</td>
<td>$1E6$</td>
<td>$1E6$</td>
<td>$(R+1)^2(m+n + R) \sim 5E4$</td>
</tr>
</tbody>
</table>

**Reduction in memory usage and mathematical complexity!**

---

**Stanford University**
LRT Accuracy Measured On-Device

Accuracy (%)

Pre-Trained Off-line

ImageNet (1000 classes)

Flowers102 (102 classes)

Maximum achievable accuracy with limited field data
LRT Accuracy Measured On-Device

Negligible ImageNet1000 degradation (< 2 %) as Flower102 are learned

LRT accuracy equivalent to conventional SGD
LRT Measured Energy and Latency

LRT rank = 14 and set size = 64

Training energy per image
- DNN Inference: 8.1 mJ (46%)
- LRT: 4.5 mJ (25%)
- RRAM Update: 5.5 mJ (29%)
- Total: 18.1 mJ

Training latency per image
- DNN Inference: 60 ms (38%)
- LRT: 44 ms (27%)
- RRAM Update: 55 ms (35%)
- Total: 159 ms

Set Size (S)
- 1000
- 100
- 10
- 1

Energy-Delay Product (J\cdot s)
- Minimal training EDP

RRAM

FE

LRT

RRAM Update

LRT

Total

Training overhead

Stanford University
Low-Rank Training Algorithm

1. Gradients are compressed with minimal information loss
   +
2. High write-energy RRAM updates amortized over large update sets
   +
3. Achieves conventional SGD equivalent accuracy
   =

Efficient RRAM-based on-device incremental training
Conclusion

• RRAM-optimized ML SoC with DNN accelerator for efficient on-device inference
  - CHIMERAv2 achieves ResNet-18 inference at 15.2 ms and 1.8 mJ

• Multi-chip Illusion System for scalable DNN inference
  - CHIMERAv2 consumes 3x less energy than traditional MCMs for ResNet-50 inference

• Low-rank training for efficient RRAM-based on-device incremental training
  - LRT achieves 340x lower EDP and 101x fewer RRAM weight updates compared to SGD at iso-accuracy