AUTOHET: An Automated Heterogeneous ReRAM-Based Accelerator for DNN Inference

*Tong Wu*, Shuibing He, Jianxin Zhu, Weijian Chen, Siling Yang, Ping Chen, Yanlong Yin, Xuechen Zhang*, Xian-He Sun$, Gang Chen
Background

Von Neumann Architecture

• Separate computing and storage units

[Diagram showing data flow between Processor and Memory]

Massive data movement

• DNN inference includes massive MVMs

[Diagram with matrices showing data flow]

High latency and high energy

Processing in Memory (PIM)

• In-situ computing

[Diagram with multiple Computing Memory units]

Low energy and low latency

• High-parallel MVMs

[Diagram showing Matrix multiplication using ReRAM Array]
Crossbars and peripheral circuits (PCs) cooperate to perform DNN inference together.

- Hierarchical topology
- In-situ computing
- Homogeneous crossbars
- Tile-based allocation

Background


1. Homogeneous crossbars cannot cope with different DNN layers.
2. Square crossbars are not well-matched the most common kernels.
Motivation

DNN layer mapping and crossbar allocation

1. Homogeneous crossbars cannot cope with different DNN layers.
2. Square crossbars are not well-matched the most common kernels.
The tile-based allocation scheme compromises resource utilization.

3. Crossbar wastage in tiles reduces the utilization of crossbars, thereby decreasing the RUE value.
• **Our goal:** Using heterogeneous crossbars for DNN inference to achieve higher RUE.

1. Homogeneous crossbars
2. Square crossbars
3. Tile-based allocation

1. Heterogeneous crossbars
2. Square + Rectangular
3. Tile-shared allocation
Goals and Challenges

• **Challenge 1:** The conflict between utilization and energy efficiency.

<table>
<thead>
<tr>
<th>DNN Layer</th>
<th>XB32</th>
<th>XB64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Utilization</td>
<td>9/16</td>
<td>9/64</td>
</tr>
<tr>
<td>ADC</td>
<td>128</td>
<td>64</td>
</tr>
</tbody>
</table>

- DNN Layer:
  - Input Channel: 4
  - Output Channel: 64
  - Kernel Size: 3 × 3

<table>
<thead>
<tr>
<th>Tile 1</th>
<th>Tile 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>27 × 32</td>
<td>36 × 64</td>
</tr>
<tr>
<td>X832</td>
<td>X864</td>
</tr>
<tr>
<td>ADC</td>
<td>ADC</td>
</tr>
<tr>
<td>9 × 32</td>
<td>9 × 32</td>
</tr>
<tr>
<td>X832</td>
<td>X832</td>
</tr>
<tr>
<td>ADC</td>
<td>ADC</td>
</tr>
</tbody>
</table>

- ADC: 128 → 64

• **Challenge 2:** The search space for determining crossbar sizes can be vast.

- **N** DNN layers + **C** crossbar candidates
- Search space: $C^N$
Design: AUTOHET

AUTOHET: an automated heterogeneous ReRAM-based accelerator for DNN inference.
Design 1: RL-based Decision Scheme

**Reinforcement learning (RL):**

- A well-established ML method for NAS
- Deep deterministic policy gradient (DDPG) algorithm
- Latency feedback

**Work flow:**

1. DNN Model → State Space
2. State Space → Actor, Critic
3. Actor, Critic → action
4. action → Strategy
5. Strategy → Feedback
6. Feedback → Reward Function
7. Reward Function → Reward
8. Reward → Experience Pool
9. Experience Pool → DDPG
10. DDPG → Experience Pool

**Heterogeneous Accelerator**

- ReRAM Bank
- Input Buffer
- Output Buffer
- Global Controller

Design 1: RL-based Decision Scheme

- Action space:
  \[ a_k \rightarrow \text{Crossbar types} \]

- Reward function:
  \[ R = \frac{u}{e} \]

- State space:
  \[ S_k = (k, t, inc, outc, ks, s, w, ins, a_k, u_k) \]

Table 1: Symbols used in the RL state space.

<table>
<thead>
<tr>
<th>No.</th>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( k )</td>
<td>layer index</td>
</tr>
<tr>
<td>2</td>
<td>( t )</td>
<td>layer type: CONV:1 ; FC: 0</td>
</tr>
<tr>
<td>3</td>
<td>( inc )</td>
<td>number of channels in the input feature map</td>
</tr>
<tr>
<td>4</td>
<td>( outc )</td>
<td>number of channels produced by the CONV</td>
</tr>
<tr>
<td>5</td>
<td>( ks )</td>
<td>number of elements of a convolution kernel</td>
</tr>
<tr>
<td>6</td>
<td>( s )</td>
<td>stride of the convolution</td>
</tr>
<tr>
<td>7</td>
<td>( w )</td>
<td>number of weights in layer ( k )</td>
</tr>
<tr>
<td>8</td>
<td>( ins )</td>
<td>size of the input feature map</td>
</tr>
<tr>
<td>9</td>
<td>( a_k )</td>
<td>action of layer ( k )</td>
</tr>
<tr>
<td>10</td>
<td>( u_k )</td>
<td>crossbar utilization of layer ( k )</td>
</tr>
</tbody>
</table>
Design 2: Heterogeneous Crossbar Size Selection

• Utilization equation:

\[
u = \frac{C_{\text{in}} \times k^2 \times C_{\text{out}}}{r \times \left\lfloor \frac{C_{\text{in}}}{r k^2} \right\rfloor \times c \times \left\lfloor \frac{C_{\text{out}}}{c} \right\rfloor}\]
Design 2: Heterogeneous Crossbar Size Selection

\[ u = \frac{C_{in} \times k^2 \times C_{out}}{r \times \left\lfloor \frac{C_{in}}{r/k^2} \right\rfloor \times c \times \left\lfloor \frac{C_{out}}{c} \right\rfloor} \]

- \( r \) needs to be multiples of \( k^2 \)
- \( c \) needs to be divisible by \( C_{out} \)

**For CONV layers:** \( k = 3 \), \( C_{out} = 2^n \) → **Rectangular crossbars**

**For FC layers:** \( k = 1 \), \( C_{out} = 2^n \) → **Square crossbars**

**Hybrid crossbars:** \( 32 \times 32, 36 \times 32, 72 \times 64, 288 \times 256, 576 \times 512 \)

**Table 2: The structure of three popular DNN models.**

<table>
<thead>
<tr>
<th>Network</th>
<th>Structure</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>C3-64, C3-192, C3-384, 2C3-256, F4096, F4096, F10</td>
<td>62%</td>
</tr>
<tr>
<td>VGG16</td>
<td>2C3-64, 2C3-128, 3C3-256, 6C3-512, F4096, F1000, F10</td>
<td>81%</td>
</tr>
<tr>
<td>ResNet152</td>
<td>C7-64, 3C1-64, 8C1-128, 40C1-256, 12C1-512, 37C1-1024, 4C1-2048, 3C3-64, 8C3-128, 36C3-256, 3C3-512, F1000</td>
<td>32%</td>
</tr>
</tbody>
</table>
**Design 3: Tile-shared Crossbar Allocation Scheme**

**Key idea:** allowing multiple DNN layers to be mapped onto the same tile, thereby reducing the number of empty crossbars by **tile sharing**.
## Experiments

<table>
<thead>
<tr>
<th>Experimental Setup</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Experimental Platform</strong></td>
<td>Software: Ubuntu+anaconda_envs(Pytorch)</td>
</tr>
<tr>
<td></td>
<td>Hardware: MNSIM_Python(simulator)</td>
</tr>
<tr>
<td><strong>Models and Datasets</strong></td>
<td>DNN Model: AlexNet, VGG16, ResNet152</td>
</tr>
<tr>
<td></td>
<td>Dataset: MNIST, CIFAR10, ImageNet</td>
</tr>
<tr>
<td><strong>Baselines</strong></td>
<td>Homogeneous: 32<em>32, 64</em>64, 128<em>128, 256</em>256, 512*512</td>
</tr>
<tr>
<td></td>
<td>AUTOHET: 32<em>32, 36</em>32, 72<em>64, 288</em>256, 576*512</td>
</tr>
</tbody>
</table>
Experiments: Overall Performance

(a) RUE

(b) Crossbar Utilization

(c) Energy Consumption
Experiments: Impact of Individual Techniques

- **Base**: the best homogeneous SXB
- **+Hetero**: RL+SXBs
- **+Hybrid**: RL+SXBs+RXBs
- **All**: RL+SXBs+RXBs+tile_shared
Experiments: Sensitivity Analysis

Various ratios of SXBs and RXBs

Various numbers of crossbar candidates

Various numbers of Pes in each tile
⇒ We propose **AUTOHET**, an automated heterogeneous ReRAM-based accelerator for DNN inference.

⇒ We introduce **hybrid crossbar shapes** (i.e., square and rectangle crossbars) to enhance the matching between the weight matrices and crossbars.

⇒ We propose the **tile-shared allocation scheme** to improve crossbar utilization further.

⇒ Experimental results demonstrate that AUTOHET effectively improves the crossbar utilization by up to $3.1 \times$ while reducing the energy consumption by up to **94.6%**.
AutoHet: An Automated Heterogeneous ReRAM-Based Accelerator for DNN Inference

Thanks for your attention!
Contact me at: wu.tong@zju.edu.cn