OmniDRL: An Energy-Efficient Mobile Deep Reinforcement Learning Accelerators with Dual-mode Weight Compression and Direct Processing of Compressed Data

Juhyoung Lee, Sangyeob Kim, Ji-Hoon Kim, Sangjin Kim, Wooyoung Jo, Donghyeon Han and Hoi-Jun Yoo

Semiconductor System Lab.
School of EE, KAIST
Deep Reinforcement Learning (DRL)

- No Pre-labelled Data ➔ Training with Trial-and-errors!
  - Sequential decision making problems @ Unknown environments
  - Applications: gaming agent, autonomous systems, agent adaptation

- Decision tree, SVM
- Deep neural network

- Classical Machine Learning vs. Classical RL
  - Known Answer vs. Policy to solve

<Applications>
- Gaming Agents
- Autonomous Systems
- Agent Adaptation
# Abstract – OmniDRL: DRL Processor

- Minimizing both External & Internal Memory Access!
  - Concept: compress data & direct utilization of compressed data

<table>
<thead>
<tr>
<th>Algorithm Level</th>
<th>Data-Level Compression</th>
<th>Bit-Level Compression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Group-Sparse Training</td>
<td>Exponent Mean Delta</td>
</tr>
<tr>
<td></td>
<td>(GST)</td>
<td>Encoding (EMDE)</td>
</tr>
<tr>
<td></td>
<td>Group-Sparse Training</td>
<td>Compression Network</td>
</tr>
<tr>
<td></td>
<td>Core (GSTC) Architecture</td>
<td>Interface (CNI)</td>
</tr>
<tr>
<td></td>
<td>Sparse Weight Transposer</td>
<td>Decoding After Fetching</td>
</tr>
<tr>
<td></td>
<td>(SWT)</td>
<td>(DAF)</td>
</tr>
</tbody>
</table>

**Weight Compression ↑**

**SRAM Power ↓**

![Diagram showing OmniDRL architecture](image)
DRL with Multiple DNNs

- Trends: Utilizing Multiple DNNs (>3) for DRL
  - Environment modelling w/ “Digital Twin” gathers experiences for training

<Multiple DNNs in DRL>  <Overall flow of DRL training scenario>
Challenges of DRL Processor

- 1. Limited Computational Intensity (~60 Ops/byte)
- 2. Dominant SRAM Power Consumption (> 53.6 %)
  - Reason: Multiple DNNs (mainly FC) require complex, sequential execution

Frequent Access of Weight & F.Map

Cannot run together

Frequent Access of Weight & F.Map

Cannot run together
Overall Architecture of OmniDRL

- 24 Group-Sparse Training Core (GSTC)
- 2-D Mesh Network-on-chip w/ 2 External I/F
- PRNG, DRL task scheduler, Top RISC Ctrlr.
Key Features for Energy-Efficiency

- 1. 24 Group-Sparse Training Core (GSTC)
- 2. Exponent Mean-Delta Encoding (EMDE)
- 3. Sparse Weight Transposer (SWT)
Proposed Data-Level Compression

- **Group-Sparse Training (GST)**
  - ~41.5 %p higher weight compression ratio than iterative pruning 😊

@ Early Iterations:
**Block-circulant + Pruning**

@ Late Iterations:
**Only Pruning**

Opportunity of GST

- **Direct Utilization of Compressed Weight is Required!**
  - Opportunity 1: $\approx \times 4$ additional data reuse for grouped weight
  - Opportunity 2: $\approx \times 10$ speed-up by skipping computation of sparse weight

---

**Group-Sparse Weight**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td>0</td>
<td>d</td>
<td>b</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>d</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>f</td>
<td>g</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>e</td>
<td>0</td>
<td>g</td>
<td></td>
</tr>
</tbody>
</table>

**Compression Process**

- **Excluding Grouped Weight**
  - |   |   |   |   |
  - | b | d |   |   |
- **Excluding Sparse Weight**
  - |   |   |   |   |
  - | e | f | g |   |

**DRL Chip**

- SRAM
- Direct Processing Required!
- PE Array
Group-Sparse Training Core (GSTC)

- **Component**
  - 8×8 Bfloat16 PU array
  - Prefetchers & Router

- **Output Stationary**

- **Row-wise Allocation**
  - Ch\textsubscript{out} parallelism
  - Weight broadcasting

- **Column-wise Allocation**
  - Batch parallelism
  - Input multicasting
**1. Weight Zero Skipping @ Prefetchers (CLK 0)**

### Input Pref.

- **Batch 0**
  - Skip Skip
  - \( I_{20} \) \( I_{30} \) \( I_{40} \) \( I_{50} \) \( I_{60} \) \( I_{70} \)

- **Batch 1**
  - Skip Skip
  - \( I_{21} \) \( I_{31} \) \( I_{41} \) \( I_{51} \) \( I_{61} \) \( I_{71} \)

### Weight Prefetcher

- **Ch_{out0}**
  - Skip Skip
  - \( W_{20} \) 0 \( W_{40} \) \( W_{41} \) \( W_{60} \) 0

- **Ch_{out1}**
  - Skip Skip Skip
  - \( W_{20} \) \( W_{41} \) \( W_{40} \) 0 \( W_{60} \)

- **Ch_{out2}**
  - Skip Skip Skip
  - \( W_{23} \) 0 \( W_{43} \) \( W_{62} \) \( W_{63} \)

- **Ch_{out3}**
  - Skip Skip
  - \( W_{23} \) 0 \( W_{43} \) 0 \( W_{63} \) \( W_{62} \)

### W Reg.

- \( W_{20} \)
- \( W_{41} \)
- \( W_{43} \)
- \( W_{23} \)

### Weight Router

- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
- PU
### GSTC Operations

- **2. Weight Group Reuse @ Weight Routers (CLK 1)**

<table>
<thead>
<tr>
<th>Batch 0</th>
<th>Batch 1</th>
<th>Input Pref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skip</td>
<td>Skip</td>
<td>(I_{20} \quad I_{30} \quad I_{40} \quad I_{50} \quad I_{60} \quad I_{70})</td>
</tr>
<tr>
<td>(I_{21} \quad I_{31} \quad I_{41} \quad I_{51} \quad I_{61} \quad I_{71})</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weight</th>
<th>Prefetcher</th>
<th></th>
<th></th>
<th>Weight Router</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(Ch_{out0})</td>
<td>Skip</td>
<td>Skip</td>
<td>(W_{20} \quad 0 \quad W_{40} \quad W_{41} \quad W_{60} \quad 0)</td>
<td>Reuse!</td>
<td>Reuse!</td>
<td>(W_{20})</td>
</tr>
<tr>
<td>(Ch_{out1})</td>
<td>Skip</td>
<td>Skip</td>
<td>(W_{20} \quad W_{40} \quad W_{41} \quad W_{40} \quad 0 \quad W_{60})</td>
<td>Reuse!</td>
<td>Reuse!</td>
<td>(W_{41})</td>
</tr>
<tr>
<td>(Ch_{out2})</td>
<td>Skip</td>
<td>Skip</td>
<td>(W_{23} \quad 0 \quad W_{43} \quad W_{62} \quad W_{63})</td>
<td>Reuse!</td>
<td>Reuse!</td>
<td>(W_{43})</td>
</tr>
<tr>
<td>(Ch_{out3})</td>
<td>Skip</td>
<td>Skip</td>
<td>(W_{23} \quad W_{43} \quad 0 \quad W_{63} \quad W_{62})</td>
<td>Reuse!</td>
<td>Reuse!</td>
<td>(W_{23})</td>
</tr>
</tbody>
</table>
Results of GSTC

- Throughput Increase with Weight Sparsity Exploitation
- WMEM Access Decrease with Weight Group Exploitation

<Normalized GSTC Throughput>

~×5.4 Throughput Increased!

<Normalized WMEM Access>

- 50%
- 75%

w/o GSTC  Group Size 2  Group Size 4
Proposed Bit-level Compression

- **3-bit Exponent Mean-Delta Encoding (EMDE)**
  - Efficient searching, ×1.6 times high CR for both weight and f.map

1. **Find Mean Exponent**
2. **Mean-Delta Comp.**

<table>
<thead>
<tr>
<th>F.Map</th>
<th>Mean Monitor</th>
<th>Exp. Dist.</th>
<th>2. Mean-Delta Comp.</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>Only Accumulation</td>
<td>Efficient ☺</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Baseline D.Type:**
  - 8'b Man. Exp.
  - 8'b Exp.

- **3-bit Delta Encoding**
  - 3'b Man.
  - 3'b Idx

- **Ratio [%]**
  - Exponent Values

- **Mean Monitor**
  - Adder
  - Reg.

- **Result**
  - High CR*!

1. **1. Find Mean Exponent**
   - Find Mean Exponent Exp. Dist.
   - Mean-Delta Comp.

2. **2. Mean-Delta Comp.**
   - High CR*!

- **Other Exponents**
  - 8'b Man.
  - 8'b Exp.
  - 3'b Idx
Decoder of the EMDE

- Decoding after Fetching (DAF)
  - Simple adder based decoder ➔ DAF ➔ Memory access power 23.3 % ↓
Motivation: Training w/ Sparse Weight

- Back-propagation Stage: Transposed Weight Required
- Irregularity Due to Storing Only Non-zero Values

---

<Ideal Sparse W Transpose>

```
Ideal Sparse W

<table>
<thead>
<tr>
<th>Ch_out</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>14</td>
<td>22</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>9</td>
<td>24</td>
<td>26</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>9</td>
<td>30</td>
<td>33</td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>24</td>
<td>35</td>
<td>34</td>
</tr>
<tr>
<td>11</td>
<td>26</td>
<td>30</td>
<td>33</td>
<td>34</td>
</tr>
</tbody>
</table>
```

```
Ideal Sparse W^T

<table>
<thead>
<tr>
<th>Ch_in</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>9</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>30</td>
<td>33</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>24</td>
<td>34</td>
<td>26</td>
</tr>
<tr>
<td>11</td>
<td>33</td>
<td>35</td>
<td>33</td>
<td>??</td>
</tr>
</tbody>
</table>
```

→ Irregular Pattern 😞

<Sparse W Transpose @ Hardware>

```
Sparse W in SRAM

<table>
<thead>
<tr>
<th>SRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>30 33 34 35</td>
</tr>
<tr>
<td>11 12 14 22 24 26</td>
</tr>
<tr>
<td>1 2 3 4 7 9</td>
</tr>
</tbody>
</table>
```

Storing only non-zero!
Sparse Weight Transposer (SWT) Arch.

- **Block-wise Division of Sparse W + Hierarchical Transpose**
  - Intra-block transpose @ Transpose Register Array (TRA)
  - Inter-block transpose @ Transpose Translation Lookaside Buffer (T-TLB)

**<Block-wise Division>**

**<SWT Architecture>**

**Ideal Sparse W**

- $Ch_{in}$
- $Ch_{out}$

**Divided Sparse W**

- $Ch_{in}$
- $Ch_{out}$

**8x8 Transpose Reg. Array (TRA)**

**Transpose Translation Lookaside Buffer (T-TLB)**

**Sparse W Decoder**

**Sparse $W^T$ Decoder**
Chip Performance and Summary

- ×2.5 Higher DRL Training Efficiency than Previous SOTA

### DRL Core Performance

<table>
<thead>
<tr>
<th>DRL Core</th>
<th>6~11</th>
<th>12~17</th>
<th>18~23</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ext I/F 0</td>
<td>Top RISC Ctrl + PRNG</td>
<td>Ext I/F 1</td>
<td></td>
</tr>
</tbody>
</table>

#### Chip Specifications

<table>
<thead>
<tr>
<th>Technology</th>
<th>28nm Logic CMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Die Area</td>
<td>3.6 mm × 3.6 mm</td>
</tr>
<tr>
<td>SRAM</td>
<td>1.5 MB</td>
</tr>
<tr>
<td>Supply Voltage</td>
<td>0.68 V ~ 1.1 V</td>
</tr>
<tr>
<td>Frequency</td>
<td>~ 250 MHz</td>
</tr>
<tr>
<td>Peak Performance</td>
<td>0.77* – 4.18** TFLOPS @ 250MHz</td>
</tr>
<tr>
<td>Power Consumption [mW]</td>
<td>3.1** – 3.9* @ (5MHz, 0.68V)</td>
</tr>
<tr>
<td>Energy Efficiency [TFLOPS/W]</td>
<td>4.2* – 29.3** @ (10MHz, 0.68V)</td>
</tr>
</tbody>
</table>

---

* = Weight Sparisty 0%, No Weight Group, Exp Comp 0%
** = Weight Sparisty 90%, Weight Group 4, Exp Comp 90%
Demonstration System

- Humanoid Adaptation to Sudden Environment Change*

OmniDRL Demo Scenario

Baseline

w/o DRL Training

Head Size ↑

Fail 😞

w/ DRL Training

Train!

Success 😊

* https://www.youtube.com/watch?v=qpnu1k8jqSQ
Conclusion

1. For Low Memory Bandwidth
   - Group-sparse training $\Rightarrow$ weight compression $\sim 41.5\% \uparrow$
   - Exponent-mean-delta-encoding $\Rightarrow$ exponent compression $\sim 1.6\times \uparrow$
   - World-first on-chip sparse weight transposer $\Rightarrow$ weight EMA $22.8\% \downarrow$

2. For Low Memory Power Consumption
   - Group-sparse training core $\Rightarrow$ Energy-efficiency $\sim 4.4\times \uparrow$
   - Decoding-after-fetching $\Rightarrow$ SRAM power consumption $23.3\% \downarrow$

OmniDRL: A 29.3 TFLOPS/W DRL Training Processor for Mobile Applications
Thank You!

- Questions? Feel Free to Contact Me!
  - E-mail: juhyoung@kaist.ac.kr
  - LinkedIn: https://www.linkedin.com/in/juhyounglee
  - Zoom Meeting: https://us02web.zoom.us/j/3466650389?pwd=c1RWaXFTWGljaVU1MiNtcDhKaGg0dz09 (Password: HC_JHLEE)

  - System Demonstration Video Link: https://www.youtube.com/watch?v=qpnu1k8jqSQ