Accelerating Weightless Neural Network Inference and Training on a Real Processing-in-Memory System

WNN-PiM

Juan Gómez Luna
Master Student

Onur Mutlu
Advisor

Xavier Servot
Advisor

SAFARI

ETH Zürich
Problem & Motivation

Deep Neural Networks are not suitable for mobile inference and training Because of ❶ High Flop Count and ❷ Large Memory Footprint

Weightless Neural Networks (WNN) emerged as a viable alternative They learn non-linear activations via lookup-tables

However, mobile inference and training of WNNs remain a challenge Because of ❶ Random Memory Accesses with ❷ Low Reuse and ❸ High Strides

Computing systems with Processing-in-Memory (PiM) capabilities can alleviate this data movement related costs by placing computation near the data

Contributions

We implement WNN inference and end-to-end training of WNNs a Real-World PiM architecture

We propose a model distribution scheme that results in efficient use of the PiM architecture

We find an algorithmic improvement that boosts accuracy and allows for further model partitioning

Results

We outperform the CPU baseline by an average of 3.6x for inference and 2.9x for training with less than 256 DPUs

We improve model accuracy by an average of 1.03 percentage points
Motivation. Mobile Inference and Training

Edge Inference is a crowded space.
Most devices that have DRAM also have NN Accelerators, GPUs and powerful CPUs

On-Device Training is valuable.

1. To fine-tune to User Behaviours
2. To adapt to System Usage Patterns

On-Device Training is hard.

1. Edge devices are power constrained
2. TinyML focuses on quantization, pruning, … → Not trainable!
Motivation. Weightless Neural Networks as a viable alternative

They learn non-linear activations via lookup tables
Motivation. *WNN Inference and Training Challenges*

- Encode
- Shuffle
- Hash
- Filter
  - Reduce
- Pop Count
  - & Argmax

**Random memory accesses**
- with large strides and no immediate reuse

**Parameters with no immediate reuse but reused among request**: Thresholds, Order, Hash Parameters, Lookup Tables

*WNN Bottlenecks*
- Executed only during inference
- & done in host
Background. Real-World PiM Architecture

We use the UPMEM PiM architecture with general-purpose processing cores called DPUs.

32-bit integer arithmetic
8-bit multiplication only
64 mB DRAM bank (MRAM)
64 kB scratchpad (WRAM)

We use \( \leq 256 \) DPUs to represent an edge system.
Motivation. *UPMEM PiM Friendliness of WNNs*

- **Encode**
- **Shuffle**
- **Hash**
- **Filter Red.**
- **Reduction**

**Random memory accesses**
- Random memory accesses with large strides and no immediate reuse

**Parameters with no immediate reuse but reused among request**
- Thresholds
- Order
- Hash Params
- Lookup Tables

**Low Arithmetic Intensity**
- Simple Operations
  - memory ops
  - memory ops
  - memory ops
  - memory ops
  - comparison
  - bit logic
  - simple integer arithmetic

**UPMEM PiM Friendliness**
- Executed only during inference & done in host
Partitioning. Why?

Random memory accesses with large strides and no immediate reuse

Random memory accesses with large strides and no immediate reuse

 DDRx Interface

IRAM 24 kB

DMA

64 bits
Partitioning. Implementation

Model Parameters

PiM System

DPU WRAM 64 kB

<table>
<thead>
<tr>
<th></th>
<th>Thresholds</th>
<th>Order</th>
<th>Hash Params</th>
<th>Lookup Tables</th>
<th>Intermediate Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Values</td>
<td>2 %</td>
<td>2 %</td>
<td>1 %</td>
<td>58 %</td>
<td>34 %</td>
</tr>
</tbody>
</table>

DPU 1
- Filters 1/12
- Class 1
- Input 1/12

DPU 109
- Filters 1/12
- Class 10
- Input 1/12

DPU 120
- Filters 12/12
- Class 1
- Input 12/12

DPU 12
- Filters 12/12
- Class 1
- Input 12/12

...
Partitioning. Benefits

Random memory accesses with large strides and no immediate reuse

Random mem access \(\leq 1\) per cycle
Algorithmic Improvement.

Standard Reordering

Encoded
Shuffled

Block-wise Reordering

Encoded
Shuffled

VS.

Enables Model Partitioning

Improves Model Accuracy
Results. Algorithmic Improvement

### Accuracy improvements in points

#### MNIST-Small

- Baseline: 93.22%
- **Reordering by 1D-Blocks**
  - Accuracy improvement: +1.06%

#### MNIST-Medium

- Baseline: 95.35%
- **Reordering by 1D-Blocks**
  - Accuracy improvement: +0.21%

#### MNIST-Large

- Baseline: 94.46%
- **Reordering by 1D-Blocks**
  - Accuracy improvement: +0.4%

#### MNIST-Small

- Baseline: 93.22%
- **Reordering by 2D-Blocks**
  - Accuracy improvement: +1.43%

#### MNIST-Medium

- Baseline: 95.35%
- **Reordering by 2D-Blocks**
  - Accuracy improvement: +0.64%
Results. Performance

**Inference**

Speedup of inference of WNNs on DPUs compared to CPU

- ~3.6x with ≤256 DPUs

**Training**

Speedup of training WNN on DPUs compared to CPU

- ~2.9x with ≤256 DPUs
Conclusion

❶ Weightless Neural Networks and Processing-in-Memory are cool

❷ Research is awesome

❸ Please send me hate-mails: xavier.servot0@gmail.com