### Problem & Motivation

Deep Neural Networks are not suitable for edge inference and training because of limited memory footprint. They use non-linear activations via lookup-tables. However, edge inference and training of WNNs remain a challenge. Computation systems with Processing-in-Memory (PiM) capabilities can alleviate this data movement related cost by placing computation near the data.

### State-of-the-Art PiM System

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

#### Methodology: System

We conduct our performance evaluation on a UPMEM PiM system that includes a 2-socket Intel Xeon Silver 4110 CPU at 2.10 GHz (host CPU), standard main memory (32GB) of 158 GB, and 20 UPMEM PiM DMUs with 160 GB PiM-capable memory and 2560 DPU. As our baseline CPU, we use a 2015 Quad-Core Intel Core i7 at 2.50 GHz, in a system with 4110 CPU at 2.10 GHz (host CPU), standard main memory (16GB) of 128 GB, and 20 UPMEM PiM DMUs with 160 GB PiM-capable memory and 2560 DPU.

#### Methodology: Experiments

To this end, we write CPU kernels for [Encoding](https://github.com/Xavier0301/Cbthowen), [Shuffling](https://github.com/Xavier0301/Cbthowen), [Hashing](https://github.com/Xavier0301/Cbthowen) and [Filter Reduction](https://github.com/Xavier0301/Cbthowen) while pop count and argmax are reduction steps only executed during inference and done in host.

#### Proposed Solution

We implement inference and end-to-end training of WNN on the UPMEM PiM system. To that end, we write CPU kernels for those operations, and the model is partitioned by classes and by filters (i.e., lookup tables) so that each part of the model fits in WMAM. Random memory accesses are then executed at a throughput of 1 per cycle, thus alleviating the main data movement related cost.

### Results: Accuracy Benefits of Algorithmic Improvements

#### Methodology: Inference

- **Algorithmic Improvements**: We conduct a performance evaluation on a UPMEM PiM system that includes a 2-socket Intel Xeon Silver 4110 CPU at 2.10 GHz (host CPU), standard main memory (32GB) of 158 GB, and 20 UPMEM PiM DMUs with 160 GB PiM-capable memory and 2560 DPU. As our baseline CPU, we use a 2015 Quad-Core Intel Core i7 at 2.50 GHz, in a system with 4110 CPU at 2.10 GHz, standard main memory (16GB) of 128 GB, and 20 UPMEM PiM DMUs with 160 GB PiM-capable memory and 2560 DPU.

#### Methodology: Experiments

- **Model Distribution**: We introduce inferences specific optimizations that reduce memory footprint of WNNs and enables to fit 16x larger models in WMAM.

#### Methodology: Accuracy Gains

- **Inference-specific optimizations**: We write our own C library for Weightless Neural Networks and validate it against an existing Python library. We use the C library for our baseline timing.

### Summary

Weightless Neural Networks are emerging as a promising alternative to Deep Neural Networks. We implement inference and training of WNNs on the UPMEM PiM system and find that this architecture can accelerate training and inference with a careful implementation.

### Future Work

- In general, more work needs to be done to understand how WNNs can be scaled up in size and adapt to more complex datasets. We also need to be done to understand the theory behind WNNs.
- Evaluating the accuracy improvements of block-wise reordering on NVA trained using backpropagation.
- Unifying the inference and training implementations to illustrate on-device continual learning.

### Github Links

- Cbthowen: A complete and fast WNN inference and training library fully written in C and validated against its Python counterpart. [https://github.com/Xavier0381/Cbthowen](https://github.com/Xavier0381/Cbthowen)
- BiM (BiThowen in Memory). An (Incomplete) implementation of WNN inference on UPMEM DPU. [https://github.com/Xavier0381/BiM](https://github.com/Xavier0381/BiM)
- BiM (BiThowen in Memory - Training). An (Incomplete) implementation of WNN training on UPMEM DPU. [https://github.com/Xavier0381/BiM](https://github.com/Xavier0381/BiM)

### Acknowledgements

We acknowledge the generous gifts provided by our industrial partners, including AMD, Facebook, Google, Huawei, Intel, Microsoft, and VMware. We acknowledge support from the Semiconductor Research Corporation and the ETH Future Computing Laboratory.