High Performance Computing 2015

Hiroki Kanezashi

Tokyo Institute of Technology

Dept. of mathematical and computing sciences

Matsuoka Lab.

Reviewed Paper 1

DaDianNao: A Machine-Learning Supercomputer

[2014 47th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)] Best Paper

Yunji Chen[1], Tao Luo[1,3], Shaoli Liu[1], Shijin Zhang[1], Liqiang He[2,4], Jia Wang[1], Ling Li[1], Tianshi Chen[1], Zhiwei Xu[1], Ninghui Sun[1], Olivier Temam[2]

[1] SKL of Computer Architecture, ICT, CAS, China [2] Inria, Scalay, France [3] University of CAS, China [4] Inner Mongolia University, China

Reviewed Paper 2

Mariana: Tencent Deep Learning Platform and its Applications

[Proceedings of the VLDB Endowment 7.13 (2014)]

Yongqiang Zou, Xing Jin, Yi Li, Zhimao Guo, Eryu Wang, Bin Xiao

Tencent Inc.

Reviewed Paper 3

Performance Modeling and Scalability Optimization of Distributed Deep Learning Systems

[21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2015)]

Feng Yan[1], Olatunji Ruwase[2], Yuxiong He[2], Trishul Chilimbi[2]

[1]College of William and Mary Williamsburg, [2]Microsoft Research

Abstract

- The most popular machine-learning algorithms for large amounts of data are CNNs and DNNs.
  - Computationally and memory intensive.
  - Their memory footprint is not beyond the capacity of the on-chip storage.
- This article introduced a custom multi-chip machine-learning architecture.
  - Achieve a speedup of 450.65x over a GPU.
  - Reduce the energy by 150.31x on average for a 64-chip system.

Outline

1. Introduction
2. State-of-the-art Machine-Learning Techniques
3. The GPU Option
4. The Accelerator Option
5. A Machine-Learning Supercomputer
6. Methodology
7. Experimental Results
8. Related Work
9. Conclusions and Future Work

- My Impression

1. Introduction

- Machine-Learning algorithms are popular in many applications and cloud services.
- Deep Learning (Convolutional and Deep Neural Networks) has been used in a broad range of applications.

Deep Learning Accelerators

- Some research groups designed accelerators for heterogeneous multi-cores.
  - Neural network accelerator for multi-layer perceptrons
  - Hardware neural network for approximating any program function
  - Accelerator for Deep Learning (for CNNs and DNNs)
- These accelerators have neural network size limitations.
  - Only few tens of neurons can be executed.
  - Neurons and synapses intermediate values have to be stored in the main memory.

Contributions of This Paper

- Presented an architecture
  - Composed of interconnected nodes, each containing computational logic, eDRAM, and the router fabric.
  - The node is implemented down to the place and route at 28nm
- Evaluated it with up to 64 nodes
  - Using the largest existing neural network layers.
  - Possible to achieve 450.65x speed-up
  - 150.31x energy reduction

2. State-of-the-art Machine-Learning Techniques

- The most popular machine-learning algoritms are CNNs and DNNs.
- CNNs and DNNs are also distinguished by their implementation of convolutional layers detailed thereafter.
  - CNN: Particularly efficient for image applications and any application which can benefit from the implicit translation invariance properties of their convolutional layers.
  - DNN: more complex neural networks but widely applicable to speech recognition, web search, etc.

A. Main Layer Types

- A CNN or a DNN is a sequence of multiple instances of four types of layers.
  - Convolutional layers (CONV)
  - Local response normalization layers (LRN)
  - Pooling layers (POOL)
  - Classifier layers (CLASS)

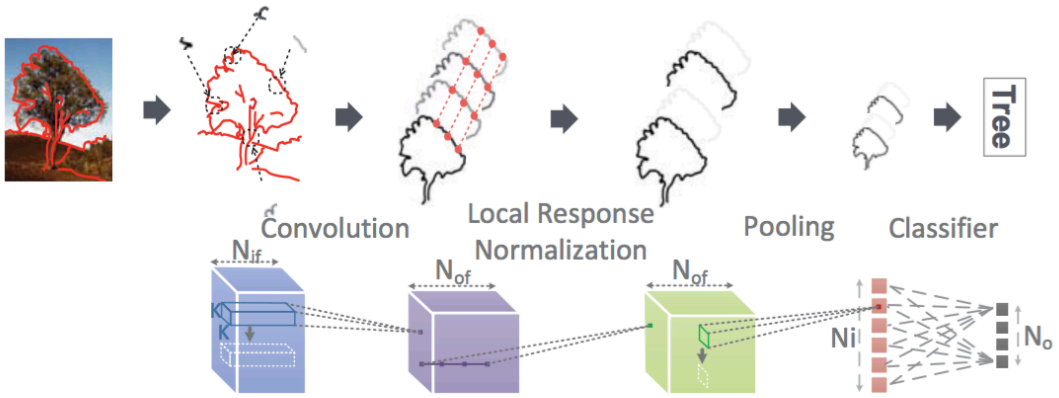

Figure 1: *The four layer types found in CNNs and DNNs.*

Convolutional Layers (CONV)

- They implement a set of filters to identify characteristic elements of the input data.
  - Filter is defined by K_x * K_y coefficients (kernel).
  - Form the layer synaptic weights from learning.

$$out(x,y)^{f_o} = \sum_{f_i=0}^{N_{if}} \sum_{k_x=0}^{K_x} \sum_{k_y=0}^{K_y} w_{f_i,f_o}(k_x,k_y) * in(x+k_x, y+k_y)^{f_i}$$

- DNN: The kernels usually have different synaptic values for each output neuron.
- CNN: The kernels are shared across all neurons of the same output feature map.

Local Response Normalization Layers (LRN)

- LRN implements competition between neurons at the same location, but in different (neighbor) feature maps.

$$out(x,y)^f = in(x,y)^f \Big/ \left(c + \alpha \sum_{g=\max(0,f-k/2)}^{\min(N_f-1,f+k/2)} (a(x,y)^g)^2 \right)^\beta$$

where k determines the number of adjacent feature maps considered, and c, α and β are constants.

Pooling Layers (POOL)

- They compute the max or average over a number of neighbor points.
  - Reduce the input layer dimensionality, which allows coarse-grain features to emerge
  - Identified by filters in the next convolutional layers.
  - They have no learned parameter (no synaptic weight).

$$out(x,y)^f = \max_{0 \le k_x \le K_x, 0 \le k_y \le K_y} in(x+k_x, y+k_y)^f$$

Classifier Layers (CLASS)

- The result of the sequence of CONV, POOL and LRN layers is then fed to one or multiple these layers.
  - Typically fully connected to its N_i inputs (and it has N_o outputs)
- They correlate the different features extracted from the other steps and the output categories.

$$out(j) = t\left(\sum_{i=0}^{N_i} w_{ij} * in(i)\right)$$

where $t()$ is a transfer function, e.g., $\frac{1}{1+e^{-x}}$, $tanh(x)$, $max(0, x)$ for ReLU [32], etc.

B. Benchmarks

- Used 10 of the popular layers as benchmarks.
  - The sliding window strides of CONV are 1, but the first CONV layer of the full NN, where they are 4.

Layer	N_x	N_y	K_x	K_y	N_i or N_{if}	N_o or N_{of}	Synapses	Description
CLASS1	-	-	-	-	2560	2560	12.5MB	Object recognition and speech recognition tasks (DNN) [11].
CLASS2	-	-	-	-	4096	4096	32MB	Multi-Object recognition in natural images (DNN), winner 2012 ImageNet competition [32].
CONV1	256	256	11	11	256	384	22.69MB	
POOL2	256	256	2	2	256	256	-	
LRN1	55	55	-	-	96	96	-	
LRN2	27	27	-	-	256	256	-	
CONV2	500	375	9	9	32	48	0.24MB	Street scene parsing (CNN) (e.g., identifying building, vehicle, etc) [18].
POOL1	492	367	2	2	12	12	-	
CONV3*	200	200	18	18	8	8	1.29GB	Face Detection in YouTube videos (DNN), (Google) [34].
CONV4*	200	200	20	20	3	18	1.32GB	YouTube video object recognition, largest NN to date [8].

Table I: *Some of the largest known CNN or DNN layers (CONVx* indicates convolutional layers with private kernels).*

C. Inference vs. Training

- A frequent and important misconception about neural networks is that on-line learning is necessary for many applications.
  - On-line learning: training or backward
  - Off-line learning: testing or feed-forward
- Designed the architecture to support the most common learning algorithms in order to also serve as an accelerator for machine-learning researchers.

3. The GPU Option

- The most favored approach for implementing CNNs and DNNs are GPUs.
  → Implemented in CUDA the layers.
  → Also implemented in C++ as baseline.

GPU Performance

- The GPU can provide a speedup of 58.82x over a SIMD on average.
- The GPU is particularly efficient on LRN layers.
  - It has a dedicated exponential instruction, a computation which accounts for most the LRN execution time on SIMD.

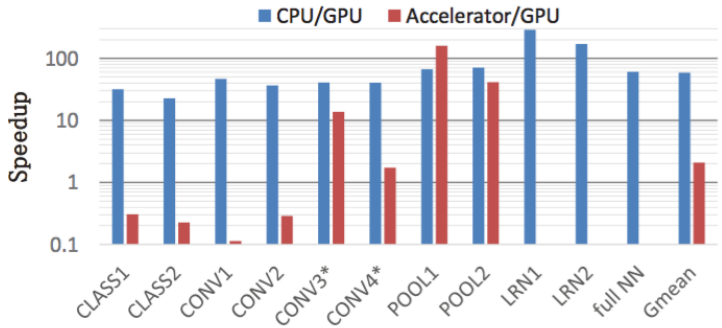

Figure 2: *Speedup of GPU over CPU (SIMD) and DianNao accelerator [5].*

GPU Problems

- GPU area cost is high
  - The number of hardware operators.
  - The need to remain reasonably general-purpose.
- The total execution time remains large
  - Up to 18.03 seconds in the CLASS1 layer.
  - Not compatible with the milliseconds response time required by many industrial applications.
- The GPU energy efficiency is moderate
  - With an average power of over 74.93W for the NVIDIA K20M GPU.

4. The Accelerator Option

- Recently, the DianNao[1] accelerator was proposed for the fast and low-energy execution of the inference of large CNNs and DNNs.
- Reimplemented a cycle-level bit-level version of DianNao, and we use the memory latency parameters.
- Introduced even larger classifier layers
  - They are large convolutional layers with respectively shared and private kernels.

[1] T. Chen, et al.: A Small-Footprint High-Throughput Accelerator for Ubiquitous Machine-Learning. In *International Conference on Architectural Support for Programming Languages and Operating Systems*, 2014.

DianNao Architecture

- Neural Functional Unit (NFU): a pipelined version of the typical computations.
- NBin, NBout: input/output buffer for input/output neurons.

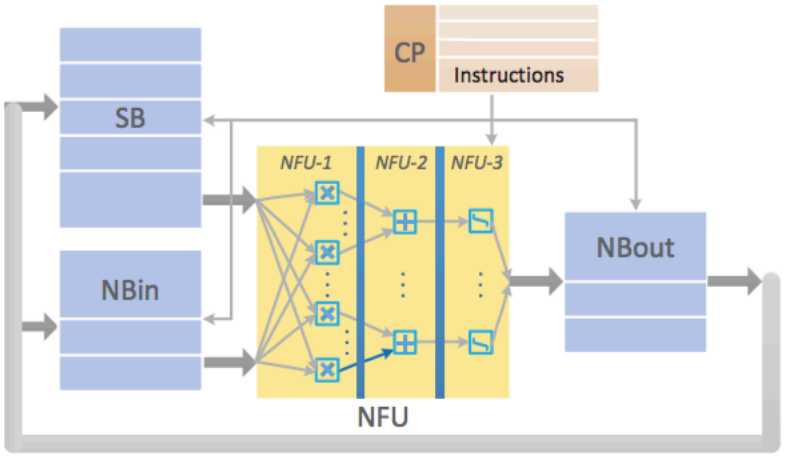

Figure 3: *Block diagram of the DianNao accelerator [5].*

5. A Machine-Learning Supercomputer

- We design the architecture around the central property, specific to DNNs and CNNs, that the total memory footprint of their parameters, while large (up to tens of GB), can be fully mapped to on-chip storage in a multi-chip system with a reasonable number of chips.

A. Overview

- Adopted the following design principles to tackle memory storage and bandwidth issue.
  1. Synapses are always stored close to the neurons which will use them, minimizing data movement, saving both time and energy.
  2. Asymmetric architecture where each node footprint is massively biased towards storage rather than computations.
  3. Transfer neurons values rather than synapses values because the former are orders of magnitude fewer.
  4. Enable high internal bandwidth by breaking down the local storage into many tiles.
- The general architecture is a set of nodes, one per chip, all identical, arranged in a classic mesh topology.

B. Node

- Synapses Close to Neurons
- High Internal Bandwidth
- Configurability (Layers, Inference vs. Training)

Synapses Close to Neurons

- To locate the storage for synapses close to neurons and to make it massive is fundamental for the architecture.
  - For both inference and training.
  - Having all synapses next to computational operators provides little data transfers and high internal bandwidth.

eDRAM Layout

- Split the eDRAM into four banks and interleaved the synapses rows among the four banks.
- Placed and routed this design at 28nm.

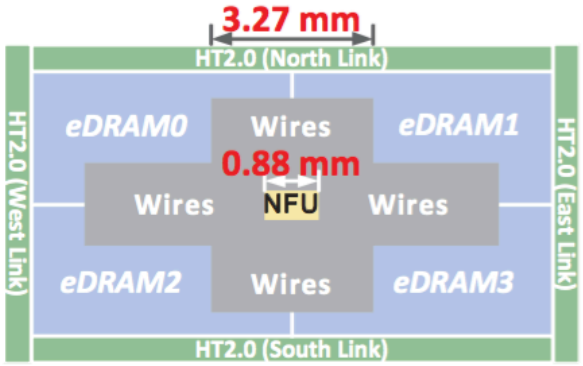

Figure 4: *Simplified floorplan with a single central NFU showing wire congestion.*

High Internal Bandwidth

- Adopt a tile-based design to avoid this congestion.
- The output neurons are spread out in the different tiles.
  - Each NFU can simultaneously process 16 input neurons of 16 output neurons.
  - All the tiles are connected through a fat tree for broadcasting and collecting.

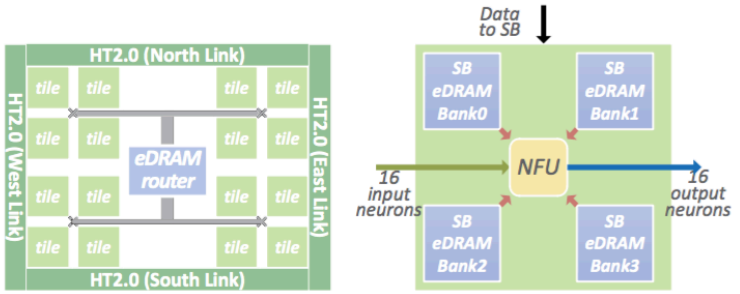

Figure 5: *Tile-based organization of a node (left) and tile architecture (right). A node contains 16 tiles, two central eDRAM banks and fat tree interconnect; a tile has an NFU, four eDRAM banks and input/output interfaces to/from the central eDRAM banks.*

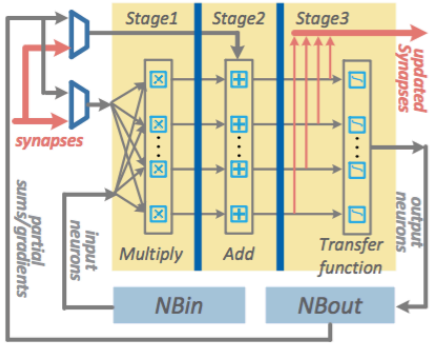

Figure 6: *The different (parallel) operators of an NFU: multipliers, adders, max, transfer function.*

Transferring Neuron Values

- The intermediate values of these neurons are saved back locally in the tile eDRAM.
- When the computation of an output neuron is finished, the value is sent through the fat tree to the center of the chip to the corresponding central eDRAM bank.

Configurability (Layers, Inference vs. Training)

- Adapt the tile, and the NFU pipeline in particular, to the different layers and the execution mode.
  - Both inference and training
- NFU is decomposed into a number of hardware blocks.
  - adder block
  - multiplier block
  - max block
  - transfer block

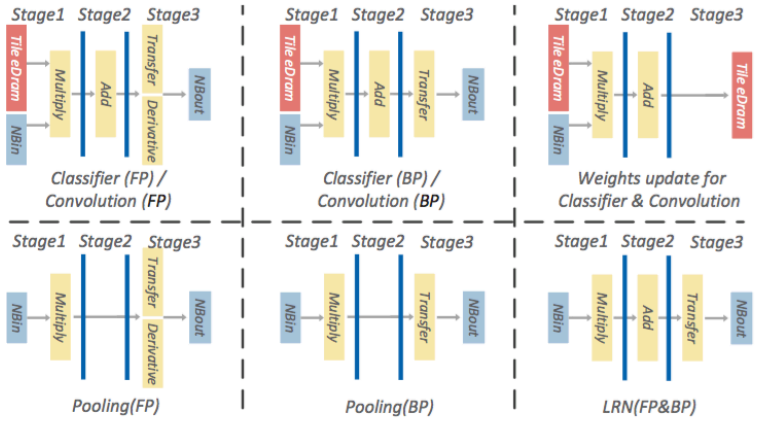

Figure 7: *Different pipeline configurations for CONV, LRN, POOL and CLASS layers.*

Fixed-Point Computations

- Each hardware block is designed to allow the aggregation of 16-bit operators into fewer 32-bit operators.
  - The overhead cost of aggregable operators is very low.
  - They may either reduce the accuracy and/or increase the convergence of training.

Inference	Training	Error
Floating-Point	Floating-Point	0.82%
Fixed-Point (16 bits)	Floating-Point	0.83%
Fixed-Point (32 bits)	Floating-Point	0.83%
Fixed-Point (16 bits)	Fixed-Point (16 bits)	(no convergence)
Fixed-Point (16 bits)	Fixed-Point (32 bits)	0.91%

Table II: *Impact of fixed-point computations on error.*

C. Interconnect

- The amount of communications is not a bottleneck except for a few layers and many-node systems.
  - Neurons are the only values transferred and heavily reused within each node.
- Used commercially available high-performance interfaces and a HyperTransport (HT) 2.0 IP block.
- Topology: 2D mesh
  - Each chip must connect to four neighbors via four HT2.0 IP blocks.
  - May be later revisited in favor of a more efficient 3D.

Router

- Next to the central block of the tile, we implement the router (Figure 5 at p.28)
  - Wormhole routing
  - Five input/output ports
  - Each input port contains 8 virtual channels
  - The router has four pipeline stages: RC, VA, SA, ST

D. Overall Characteristics

- 16 tiles per node
- 4 eDRAM banks contains 1024 rows of 4096 bits per tile
  - The total eDRAM capacity in one tile is 2MB
- Clock the NFU at the same frequency (606MHz) as the eDRAM available in the 28nm technology
  - To avoid the circuit and time overhead of asynchronous transfers

Parameters	Settings	Parameters	Settings
Frequency	606MHz	tile eDRAM latency	~3 cycles
# of tiles	16	central eDRAM size	4MB
# of 16-bit multipliers/tile	256+32	central eDRAM latency	~10 cycles
# of 16-bit adders/tile	256+32	Link bandwidth	6.4x4GB/s
tile eDRAM size/tile	2MB	Link latency	80ns

Table III: *Architecture characteristics.*

E. Programming, Code Generation and Multi-Node Mapping

- These node instructions themselves drive the control of each tile.
  - The control circuit of each node generates tile instructions and sends them to each tile.
  - The spirit of a node or tile instruction is to perform the same layer computations on a set of contiguous input data.
- The control provides *processing one row at a time* or *batch learning* modes.

E. Programming, Control and Code Generation

- The programming requirements are low, the architecture essentially has to be configured and the input data is fed in.
  - The input data is initially partitioned across nodes and stored in a central eDRAM bank.
  - The neural network configuration is implemented in the form of a sequence of node instructions.

CP	central eDRAM								SB				NBin				NBout				NFU				
Inst Name	READ OP	WRITE OP	READ ADDR	WRITE ADDR	READ STRIDE	WRITE STRIDE	READ ITER	WRITE ITER	READ OP	WRITE OP	ADDR	STRIDE	READ OP	WRITE OP	ADDR	STRIDE	READ OP	WRITE OP	ADDR	STRIDE	NFU-1 OP	NFU-2 OP	NFU-3 OP	NFU-2-IN	NFU-2-OUT

Table IV: *Node instruction format.*

CP	central eDRAM								SB				NBin				NBout				NFU				
Class	LOAD	WRITE	0	256	0	4	64	4	READ	NULL	0	1	READ	NULL	0	1	NULL	WRITE	0	1	MUL	ADD	NULL	0	1
Class	LOAD	WRITE	64	256	0	4	64	4	READ	NULL	256	1	READ	NULL	0	1	READ	WRITE	0	1	MUL	ADD	NULL	1	1
Class	LOAD	WRITE	128	256	0	4	64	4	READ	NULL	512	1	READ	NULL	0	1	READ	WRITE	0	1	MUL	ADD	NULL	1	1
Class	LOAD	STORE	192	256	0	4	64	4	READ	NULL	768	1	READ	NULL	0	1	READ	WRITE	0	1	MUL	ADD	SIGMOID	1	0

Table V: *An example of classifier code ($N_i = 4096$, $N_o = 4096$, 4 nodes).*

E. Multi-Node Mapping

- At the end of a layer, each node contains a set of output neurons values, stored back in the central eDRAM.
- The input neurons are distributed across all nodes, in the form of 3D rectangles corresponding to all feature maps of a subset of a layer.
- Finally, communications can be high for classifier layers because each output neuron uses all input neurons.

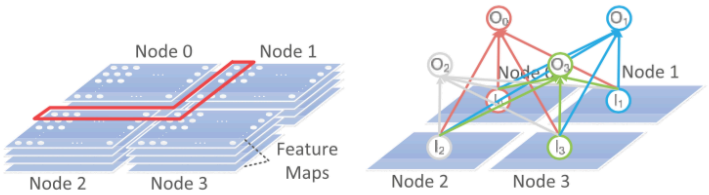

Figure 8: *Mapping of (left) a convolutional (or pooling) layer with 4 feature maps; the red section indicates the input neurons used by node 0; (right) a classifier layer.*

6. Methodlogy

- For a series of experiments, the following tools are used.
  - Verilog (CAD Tools)
  - Time, eDRAM and inter-node measurements
  - GPU
- To maximize the quality of the baseline, the following programming environment are used.
  - CUDA Convnet
  - Intel SIMD CPU

A. Measurements

- Verilog (CAD Tools)
  - Synopsys Design Compiler for the synthesis
  - ICC Compiler for the layout
  - Synopsys PrimeTime PX for power consumption estimation
- Time, eDRAM and inter-node measurement
  - VCS to simulate the node RTL
  - eDRAM model which includes destructive reads
  - Periodic refresh of a banked eDRAM running
  - Cycle-level Booksim2.0 interconnection network simulator
- GPU (as a baseline)
  - NVIDIA K20M GPU of Section III
  - Report its own power usage
  - Used CUDA SDK 5.5 to compile the CUDA version of neural network codes

B. Baseline

- Extracted the CUDA versions from *CUDA Convnet* (a tuned open-source version).
  - To maximize the baseline quality.
- Compared it against the C++ version run on the Intel SIMD CPU.
  - To assess the quality of this baseline
- Compared the SIMD version against a non-SIMD version
  - To observe an average speedup of the SIMD version

7. Experiment Results

- In this chapter, the following characteristics and result of experiments are presented.
  - The main characteristics of the node layout.
  - The performance and energy results of the multi-chip system.

A. Main Characteristics

- Chip area
  - 44.53% is used by the 16 tiles
  - 26.02% by the four HT IPs
  - 11.66% by the central block (including 4MB eDRAM, router and control logic)
  - (47.55% by memory cells)
- The peak power consumption: 15.97 W
  - tiles: 38.53%
  - memory cells (tile eDRAMs + central eDRAM): 38.30%
  - combinational logic: 37.97%
  - registers: 19.25%

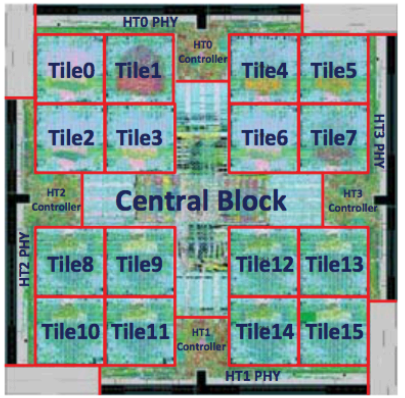

Figure 9: *Snapshot of the node layout.*

Component/Block	Area (μm^2)	(%)	Power (W)	(%)
WHOLE CHIP	67,732,900		15.97	
Central Block	7,898,081	(11.66%)	1.80	(11.27%)
Tiles	30,161,968	(44.53%)	6.15	(38.53%)
HTs	17,620,440	(26.02%)	8.01	(50.14%)
Wires	6,078,608	(8.97%)	0.01	(0.06%)
Other	5,973,803	(8.82%)		
Combinational	3,979,345	(5.88%)	6.06	(37.97%)
Memory	32207390	(47.55%)	6.12	(38.30%)
Registers	3,348,677	(4.94%)	3.07	(19.25%)
Clock network	586323	(0.87%)	0.71	(4.48%)
Filler cell	27,611,165	(40.76%)		

Table VI: *Node layout characteristics.*

B. Performance

- Compared to the GPU baseline, the speedup is up to 2595.23x for 64 nodes in CONV1 layer.

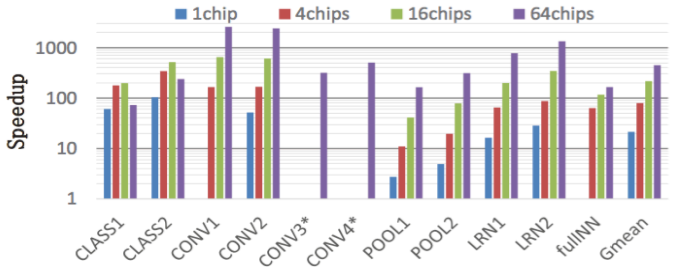

Figure 10: *Speedup w.r.t. the GPU baseline (inference). Note that CONV1 and the full NN need a 4-node system, while CONV3* and CONV4* even need a 36-node system.*

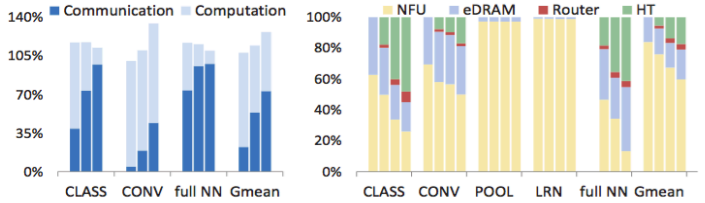

Figure 11: *Time breakdown (left) for 4, 16 and 64 nodes, (right) breakdown for 1, 4, 16, 64 nodes; CLASS, CONV, POOL, LRN stand for the geometric means of all layers of the corresponding type, Gmean for the global geometric mean.*

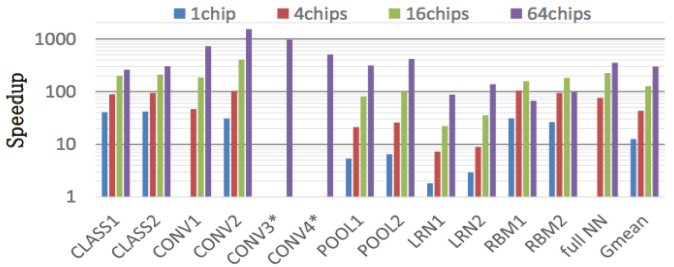

Figure 12: *Speedup w.r.t. the GPU baseline (training).*

C. Energy Consumption

- These architectures can reduce the energy by up to 330.56x for inference.
  - The minimum energy improvement is 47.66x for CLASS1 with 64 nodes.
  - For these layers, the energy benefit remains relatively stable as the number of nodes is scaled up.
- For training and initialization, the energy reduction of our architecture with respect to the GPU baseline on training is up to 180.42x.

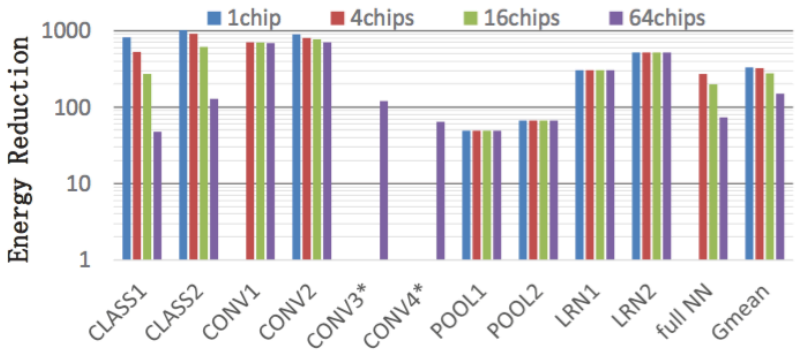

Figure 13: *Energy reduction w.r.t. the GPU baseline (inference).*

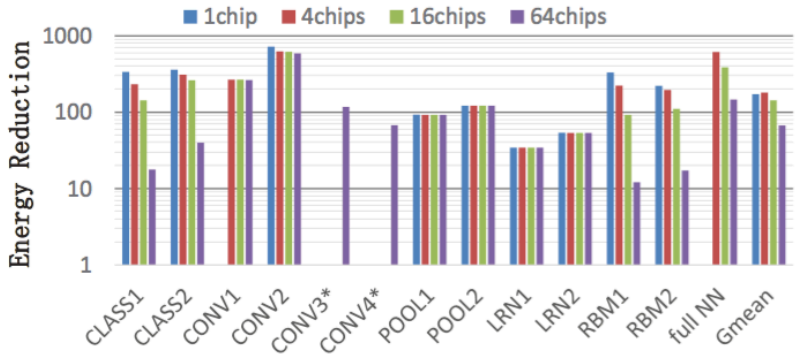

Figure 14: *Energy reduction w.r.t. the GPU baseline (training).*

8. Related Work (Machine-Learning)

- Services that are computationally intensive, and considering the energy and operating costs of data centers, custom architectures could help from both a performance and energy perspective.
- But such web services are only the most visible applications.
- Even there is an inherit risk in hardware,
  - Hardware can rapidly evolve with machine-learning progress, much like it currently (and rapidly) evolves with technology progress.
  - That hardware needs not implement and follow each and every evolution.
  - End users are already accustomed to the notion of software libraries, and they can always choose between fast library.

Related Work (Custom Accelerators)

- Architecture customization is increasingly viewed as one of the most promising paths forward.
- Closer to the target algorithms of this paper, a number of studies have recently advocated the notion of neural network accelerators
  - Either to approximate any function of a program, for signal-processing tasks or specifically for machine-learning tasks

Related Work (Large-Scale Custom Architectures)

- There are few examples of custom architectures targeting large-scale neural networks.
  - A wafer-scale design capable of implementing thousands of neurons and millions of synapses.
  - The SpiNNaker system is a multi-chip supercomputer where each node contains 20+ ARM9 cores linked by an asynchronous network.
  - The IBM Cognitive Chip is a functional chip capable of implementing 256 neurons and 256K synapses.
- Their goal is the emulation of biological neurons, not machine-learning tasks.

9. Conclusions

- Showed the possibility to design a multi-chip architecture for CNNs and DNNs.
  - Outperform a single GPU by up to 450.65x.
  - Reduce energy by up to 150.31x using 64 nodes.
- On both GPUs and recently proposed accelerators, such algorithms exhibit good speedups and area savings respectively, but they remain largely bandwidth-limited.

Future Work

- To improve that architecture:
  - Increasing the clock frequency of the NFU.
  - Multi-dimensional torus interconnects to improve the scalability of large classifier layers.
  - Investigating more flexible control in the form of a simple VLIW core per node.
  - Associated toolchain.
- A tape-out of a node chip is planned soon.

My Impression

- The architecture well considers Deep Learning layers and algorithms.
  - Four types of layers (CONV, LRN, POOL, CLASS)
  - It minimizes communication overhead by the characteristic of neural network structure.
- The performance evaluations are performed only in simulators.
  - Actual machine experiments should be performed even a few nodes.
  - What if we run thousands of those nodes?