In-memory Computing with thousandfold energy-efficiency
— an algorithm-architecture co-design approach

Li Jiang
Associate Professor
Department of Computer Science and Engineering
Shanghai Jiao Tong University
Outline

1. Introduction

Darkness Before the Dawn

2. Scalability limits deployment of large DNN

3. Imperfect Device -> inaccurate compute

4. AI based Approximate for adaptivity
Computing power is the new bottleneck for AI

- 1470 * V100 card (300W) training for 1 hour
- One game, 6,380 * TPU, power consumption cost 3.5 million USD

Computing power requirement doubles in 3-4 months

Moore's law: performance doubles in > 2 years

Microarchitectural innovation (More parallelism, Data reuse, simpler control) becomes unstorable!
Energy-efficiency wall

Energy wall: conventional technology and architecture cannot exceed 10s of TOPS/w

- 80% energy consumed by AI inference in Facebook;
- >4% power consumed by IDC;
- Low PUE in IDC: 30% storage, 20% movement, 50% computation
Memory wall

Massive data movement of Deep learning application within chip and across system

- Bandwidth/op eration decreases by 3 times
- Development of Memory technology is slower

Applications across ~10^6 performance range

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Intel-Xeon 8180</td>
<td>14nm</td>
<td>0.28T</td>
<td>0.56T</td>
<td>-</td>
<td>2.24T</td>
<td>205W</td>
<td>6xDDR4</td>
</tr>
<tr>
<td>GPU</td>
<td>nVIDIA-V100</td>
<td>14nm</td>
<td>7.5T</td>
<td>15T</td>
<td>30T</td>
<td>125T</td>
<td>250W</td>
<td>HBM</td>
</tr>
<tr>
<td>FPGA</td>
<td>Xilinx VC1902</td>
<td>7nm</td>
<td>-</td>
<td>8T</td>
<td>33T</td>
<td>133T</td>
<td>HBM</td>
<td></td>
</tr>
<tr>
<td>ASIC</td>
<td>Nervana NNP</td>
<td>28nm</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>38T</td>
<td>HBM</td>
<td></td>
</tr>
<tr>
<td>TPU-3</td>
<td></td>
<td>14nm</td>
<td>-</td>
<td>-</td>
<td>90T</td>
<td>180T</td>
<td>?</td>
<td>HBM 2</td>
</tr>
<tr>
<td>Neuromorphic</td>
<td>IBM TrueNorth</td>
<td>90nm</td>
<td>46.6T Neural Operations</td>
<td>18.2M Neural Operations</td>
<td>1W</td>
<td>PIM</td>
<td>1W</td>
<td>PIM</td>
</tr>
<tr>
<td>Neuromorphic</td>
<td>IFAT-UCSD</td>
<td>90nm</td>
<td>18.2M Neural Operations</td>
<td></td>
<td></td>
<td>25uw</td>
<td>PIM</td>
<td></td>
</tr>
</tbody>
</table>

[HiSilicon]

[Deephi]

New architecture and low-precision is essential!
AI Chips towards mobile and IoT

Technology
- Processor with tensor computing
- Domain Specific DNN Accelerator
- Reconfigurable Accelerator
- Algorithm (SW) x Arch. co-design
- Scalable Heterogeneous design

Silicon Products
- GPUs
  - Programmable
  - Parallel Compute
  - Power hungry
- TPUs
  - v1 released in 2015, inference
  - v2/v3 generally Available for Training and inf.
- Efficient CGRA
  - Dataflow
  - Reconfigurable for bit-width ...
- Highly custom hardware
  - Ultra-low power
  - Capable of train even at end/edge
- Computing in memory Arch.
  - Avoid VN wall
  - Algorithm, Arch., circuits, compiler codesign

Deployment Location
- Data center
- Server/Edge
- Edge/Mobile End
- IoT / Robot
- One for all

- Extremely high energy-efficiency
- Cost and technology scaling tradeoff performamce request!
Analog Comp. based IMC: >>100x Energy–efficiency

ReRAM:
5nm Filament, BEOL, 15ns, 10-year retention

Vector Matrix multiplication
RRAM Crossbar
RRAM Crossbar schematic
M x N Multiply-Add Tree

With the same TOPS
Transistors: 1.8M VS. 54B
Power: 70w VS. 400w

54 billion transistor, 400 w, 480K multiplier, peak 624 TOPS (8bit)

Analog computing, Kirchhoff's current law
• Non-Von Neumann architecture: weight in the memory
• Read memory = multiplication (I = V x (1/R))

CMOS:
thousands of Transistor form 1 multiplier, adder, register
0.1~1B transistors form 1K*1K Matrix
Memristive process technologies

Gartner 2019

<table>
<thead>
<tr>
<th>Process</th>
<th>ReRAM</th>
<th>Nor Flash</th>
<th>SRAM</th>
<th>MRAM</th>
<th>PCM</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Read Perf.</td>
<td>Fast</td>
<td>Slow</td>
<td>Fast+</td>
<td>Fast</td>
<td>Slow</td>
</tr>
<tr>
<td>Power</td>
<td>Ex. Low</td>
<td>Large</td>
<td>Medium</td>
<td>Ex. Low</td>
<td>Low</td>
</tr>
<tr>
<td>Integration</td>
<td>on-chip 3D</td>
<td>off-chip</td>
<td>on-chip</td>
<td>on-chip</td>
<td>on-chip 3D</td>
</tr>
<tr>
<td>Cost</td>
<td>$1-4</td>
<td>$2.8</td>
<td>$200</td>
<td>$38</td>
<td>$1.6</td>
</tr>
</tbody>
</table>

ReRAM technology

- Superior in cost, power, latency
- Multi-level cell suitable for AI
- Anology computing
- On-chip 3D integration
Darkness Before the Dawn

1. Introduction

2. Scalability limits deployment of large DNN
   - 1T1R device density
   - IR drop and sneak path

3. Imperfect Device -> inaccurate compute

4. AI based Approximate for adaptivity
Recent advancement of DNN compression

- **Sole SW optimization**
  - Less storage and less bandwidth
  - No obvious computation speedup

- **Sole HW optimization**
  - Compression and decompression
  - Skipping the “x 0” operations needs complex and costly hardware design (e.g., EIE by Stanford)

- A new thinking of Algorithm & Hardware codesign is desired!
Structural compression of DNN [DATE’18]

- No obvious speed up
  - Non-structured sparsity needs fine-grained scheduling
  - High Parallelism has coarse-grained “structure”
- Cannot save storage
  - Hardware coupled

5~10x compression rate, in practice < 2x Speedup
Structural compression of DNN [DATE’18]

- No obvious speed up
  - Non-structured sparsity needs fine-grained scheduling
  - High Parallelism has coarse-grained “structure”
- Cannot save storage
  - Hardware coupled

**Structural pruning algorithm:**
Group lasso in loss function

**First for crossbar**

PE-Buffer Interface design:
Sparse input Fetching & Output alignment

5~10x compression rate, in practice < 2x Speedup
Structural compression of DNN

5~10x compression rate, in practice < 2x Speedup

Up to 10.66x speedup
Up to 9.43x energy saving
< 1% accuracy lost

A typical in-memory computing technique.
---Cited by Bill Day (a joint work of Stanford/MIT in DAC2019)
More fine-grained pruning granularity, better compression

Coarse-grained pruning on the whole weight matrix

Fine-grained pruning on the weights mapped on a crossbar
PIM-Prume: Fine-grain Pruning

- Fine-grain pruning in both row and column cause dislocation problem
- Pruning algorithm coupled with crossbar structure
- Crossbar column/row after pruning and shifting contains different Filter/Channel
PIM-Prume: Fine-grain Pruning

- Fine-grain pruning in both row and column cause **dislocation** problem
- A warm-up solution
  - Matrix column-level structural pruning, mapping, and Crossbar row-level pruning
  - Matrix row-level structural pruning, mapping, and Crossbar column-level pruning

1. Matrix column pruning

2. Mapping

3. Crossbar row pruning

1. Matrix row pruning

2. Mapping

3. Crossbar column pruning
PIM-Prume: Fine-grain Pruning

- Fine-grain pruning in both row and column cause **dislocation** problem
- A warm-up solution
  - Matrix column-level structural pruning, mapping, and Crossbar row-level pruning
  - Matrix row-level structural pruning, mapping, and Crossbar column-level pruning
- **Block based pruning with both dimension: decoupling hardware and model**

1. **Matrix column pruning**
   - 2. **Mapping**
   - 3. **Crossbar row pruning**

1. **Matrix row pruning**
   - 2. **Mapping**
   - 3. **Crossbar column pruning**

1. **Block based pruning**
   - 2. **Mapping**
# Compression result w./w.o quantization (8bit)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Network</th>
<th>Baseline Acc</th>
<th>Crossbar Size</th>
<th>Compression Rate of XBs</th>
<th>Acc after Pruning</th>
<th>Acc after Quantization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Speech Command</td>
<td>GRU_S</td>
<td>93.62%</td>
<td>16*16</td>
<td>2.40x 4.01x</td>
<td>93.15% 92.92%</td>
<td>93.29% 92.82%</td>
</tr>
<tr>
<td></td>
<td>GRU_B</td>
<td>94.62%</td>
<td>16*16</td>
<td>4.93x 9.92x</td>
<td>94.40% 93.97%</td>
<td>94.42% 93.97%</td>
</tr>
<tr>
<td></td>
<td>CRNN_S</td>
<td>93.50%</td>
<td>16*16</td>
<td>3.99x 7.79x</td>
<td>93.56% 93.11%</td>
<td>93.33% 93.05%</td>
</tr>
<tr>
<td></td>
<td>CRNN_B</td>
<td>94.56%</td>
<td>16*16</td>
<td>5.12x 10.44x</td>
<td>94.62% 94.15%</td>
<td>94.66% 94.03%</td>
</tr>
<tr>
<td>Voxceleb 100</td>
<td>TDNN</td>
<td>89.78%</td>
<td>16*16</td>
<td>8.41x 14.02x 19.14x</td>
<td>90.33% 90.00% 90.11%</td>
<td>90.47% 89.81% 89.87%</td>
</tr>
</tbody>
</table>

## NLP
- Up to 20 X reduction on crossbars
- Accuracy drop < 0.1%
- Tested in industry production line

## CV
- Up to 27 X reduction on crossbars
- Averagely **Accuracy drop < 0.1%**
- maximumly < 0.2%
Short Summary for Pruning

- structural pruning
  - nonstructural random
  - vector-wise
  - row-wise
  - column-wise
  - channel-wise
  - block-wise
  - reduced randomness

Row-balanced Irregular Pruning, [NVIDIA, A100]

Irregular & random

Load balance
Short Summary for Pruning

- structural pruning

- nonstructural random
- vector-wise
- row-wise
- column-wise
- channel-wise
- block-wise

reduced randomness

Row-balanced Irregular Pruning, [NVIDIA, A100]

Column-balanced Irregular Pruning

Irregular & random

Load balance

Pruned Matrix

Dense Matrix

Pruned Matrix

In Winograd domain

Pruned Matrix;
Load balanced;
Simplify the architecture
DNN quantization methods

Weight Clustering [EIE ...]
- Reduce the number of values that weights can represent
- Weight sharing
- Need costly indexing

Fixed quantization [INQ, INT8, PACT ...]
- Quantize values using the fixed bit-width
- At the cost of accuracy

Adaptive quantization [AdaBits, FXP ...]
- Quantize weights using different bit-width
- Lack of exploration on input feature maps

Dilemma of accuracy and computing cost
DRQ: Dynamic Region-Based Quantization [ISCA-2020]

Weight Clustering [EIE ...]
- Reduce the number of values that weights can represent
- Weight sharing
- Need costly indexing

Fixed quantization [INQ, INT8, PACT ...]
- Quantize values using the fixed bit-width
- At the cost of accuracy

Adaptive quantization [AdaBits, FXP ...]
- Quantize weights using different bit-width
- Lack of exploration on input feature maps

Different pixels have different impact on accuracy
- Large values have greater impact on accuracy (sensitive values)
- The sensitive input values aggregate in space (sensitive regions)

High/low precision model for Sensitive/insensitive regions
Algorithm
• Light-weight predictor for the sensitivity region
• Dynamic quantized weight (e.g., 4/8 bit)

Architecture:
• 4/8bit data alignment in line buffer
• Multi-precision and Variable-speed MAC (8bit = 4bit<< + 4bit)
• Data path sync. by stalling
Explore the Bit-level sparsity for energy-efficiency [GLVLSI-2020]

- Bit-level sparsity is difficult to exploit
- Unbalanced bit-level energy consumption induced by peripherals (e.g., ADC)
- **Key observation:** reading bit-1 and bit-0 incurs drastic different energy-consumptions
- **Key idea:** search (quantize) the binary code of weight to unbalance the 0 and 1 bit count
- Method: energy-aware gradient search to guarantee accuracy and group update to keep the unbalance

<table>
<thead>
<tr>
<th>Power</th>
<th>[VLSI’14]</th>
<th>[TCS’10]</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit-1</td>
<td>300</td>
<td>136</td>
</tr>
<tr>
<td>bit-0</td>
<td>0.99</td>
<td>1.91</td>
</tr>
</tbody>
</table>

- **Weight**
  - MSB: 0 0 0 0 1 1 1 1
  - LSB: 0 0 0 1 0 0 0 0

- **Graphs**
  - VGG16
  - ResNet50
  - InceptionV3
Outline

1. Introduction

Darkness Before the Dawn

2. Scalability limits deployment of large DNN

3. Imperfect Device -> inaccurate compute
   - Resistance variation (device-to-device variation)
   - Highlight: Retraining method for mainstream adoption

4. AI based Approximate for adaptivity
Approximate computing for adaptivity

- Many applications are error tolerant
  - Input are noisy e.g. sensors
  - Outputs are probabilistic estimates e.g. machine learning
  - User facing application output e.g. Images
- Use Neural network to approximate a code block/function
  - Amdahl law: performance limited by serial code
  - NN has high parallelism, e.g., FPGA, ASIC, GPU
  - A Striking facts: Neural network can compute any function
Approximate computing with Quality control

Original program

Input

Original program

Output

Approximate

Input

Accelerator

Output

With quality control

Input

Classifier

Output

NN-based

error < threshold

Original program

NN-based

error > threshold

Accelerator

[ISCA’13, ISLPED’16]

[Micro’16]

Train NN based approximator & classifier

Initialize

Train accelerator

Predict accelerator

Record error

Find the best accelerator

Change topology

Original data

Train the “best” accelerator to fit the input data

Train the “best” classifier to predict the approximate error

Q1: Does the “best” accelerator and the “best” classifier lead to the best overall energy-efficiency?

Q2: How multiple tasks/models cooperate for a single mission?

Q3: How to train them?

Q4: How to design the chip for them?
Our work on Approximate DNN framework

- Two neural networks have different points of views on the input data space [DAC’17]
  - Iterative Collaborate training for two models
  - Efficient Network structure search
  - Up 3X speed up!
- Too little agreement on the Safe-to-approximate input data [FPGA’18, ICCAD’18]
  - Multi-class classifier and multiple approximator
  - Architecture support for multiple models
  - Up 15X speed up, but 5X more training time!
- Multiple collaborated neural networks are difficult to train [ICCAD’18²]
  - End-to-end training for multiple models
  - Architecture support for the sophisticated execution
  - Up 20X speed up, but as low as 10% training time!
Complex computing demand near sensors

LBP/Edge extraction [VLSIC2018]
Always on Image Processing

Intelligent Vision Sensor (ISP+AI Processing)

“General” domain-specific CiM

<table>
<thead>
<tr>
<th>Operation Modes</th>
<th>“General” domain-specific CiM</th>
</tr>
</thead>
<tbody>
<tr>
<td>LBP Extraction 128x108</td>
<td>Compression</td>
</tr>
<tr>
<td>Edge Detection 128x108</td>
<td>Data encryption</td>
</tr>
<tr>
<td>Normal Imaging 256x216</td>
<td>FIR</td>
</tr>
<tr>
<td></td>
<td>Coding</td>
</tr>
</tbody>
</table>

| CTRL                             | Bit-rate estimation           |
|----------------------------------| Image Processing              |
| MUX (x108)                       | Audio processing              |
| CAP Array (2x108)                | stereoscopic vision           |
| MIMO COMP (x108)                 |                               |
| OBUF (x18)                       |                               |

Inherent weakness of Adaptness for CiM can be resolved by NN-approximation.

Near sensor function blocks can be approximated by AI algorithms!

Adaptivity

Efficiency
The Cross-layer codesign way is a challenging way

- Advocate **Domain Specific Architecture (DSA)**
- Conventional VLSI has a thin layer of **Instruction Set Architecture**, we optimizes the ISA.
- DSA has much more abstraction and interface

David Patterson Turning awardee

**Conventional hardware-software interface**

**Hardware**  
| ISA | Software |

Conventional: Compiler and architecture ISA is in between  
Co-design by optimizing scheduling of instruction  
- VLIW (static)  
- OoO (dynamic)

**Hardware-software codesign in DSA**

**Device**  
| Hardware | ISA | Compile | Numerical | App. | API Lib. | Alg. | Model |

Now: Many abstraction and interface  
- New device is considered (Memristor, Optic)  
- Cross-layer (device, architecture, algorithm) co-optimization

Very challenge!
Ongoing project: tape-out

ReRAM array tape out on April.
• 16K ReRAM array
• Based on ICRD@HLMC 40nm
• 1bit resolution each cell
• Test chip for verification of our innovation
Tiny AI compiler with sparsity support

<table>
<thead>
<tr>
<th>Framework</th>
<th>Inference</th>
<th>Training</th>
<th>Structural sparsity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paddle Mobile</td>
<td>✓</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>MNN</td>
<td>✓</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>Arm Compute Library</td>
<td>✓</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td>Tensorflow Lite</td>
<td>✓</td>
<td>✘</td>
<td>✘</td>
</tr>
</tbody>
</table>

- No training framework can be deployed in mobile/edge
- No DNN framework support structural sparsity
- Sparsity is a **big open problem** in DSA/DSL --by John Hennecy, CNCC 2020

- Support inference/training with structural sparsity deployed on mobile
- Small memory occupation and high performance under limited resource
- Front-end modification Based on TVM and migrated to MindSpore
Conclusion and Future work

DCNN  Inception  Residual
LSTM  Bert  GCNN

Tiny DNN

Deployed

CIM chip

Compressed
Prune and quant.

Co-design
Algorithm
Architecture

Accuracy
Cost
Power
Performance
Conclusion and Future work

Near sensor/Domain specific
DCNN  Inception  Residual
LSTM  Bert  GCNN

Approximated
Compressed
Prune and quant.

Tiny DNN

Deployed
CiM chip

Co-design
Algorithm

Accuracy
Cost
Power
Performance

Architecture
Conclusion and Future work

Near sensor/Domain specific
DCNN, Inception, Residual, LSTM, Bert, GCNN

Approximated
Compressed
Prune and quant.

Tiny DNN

Deployed

Co-design

Algorithm

Accuracy, Cost, Power, Performance

Architecture

Testing, Fault-tolerance, Encoding, Compiling

Tool-chain
Acknowledgment

Sponsors:

Thanks to collaborators:
Prof. Rui Yang, Weikang Qian, Yanan Sun, Xiaoyao Liang, Naifeng Jing and our students
Thank you for your attention!

jiangli@cs.sjtu.edu.cn

It's QUESTION TIME!!
Sponsors

Premier Sponsor & tinyML Strategic Partner

arm

Gold Sponsor

EDGECORTIX®

Silver Sponsor

SynSense
Arm: The Software and Hardware Foundation for tinyML

- Connect to high-level frameworks
- Supported by end-to-end tooling
- Connect to Runtime

Profiling and debugging tooling such as Arm Keil MDK

1. Application
2. Optimized models for embedded
3. Runtime (e.g. TensorFlow Lite Micro)
   - Optimized low-level NN libraries (i.e. CMSIS-NN)
   - RTOS such as Mbed OS
   - Arm Cortex-M CPUs and microNPUs

AI Ecosystem Partners

Stay Connected

@ArmSoftwareDevelopers
@ArmSoftwareDev

Resources: developer.arm.com/solutions/machine-learning-on-arm
Dynamic Neural Accelerator™

- 10x more compute with single DNA engine
- More than 20x better energy-efficiency
- Ultra-low latency
- Fully-programmable with INT 8bit support

www.edgecortix.com

© 2020 EDGECORTIX. ALL RIGHTS RESERVED
SynSense builds ultra-low-power (sub-mW) sensing and inference hardware for embedded, mobile and edge devices. We design systems for real-time always-on smart sensing, for audio, vision, IMUs, bio-signals and more.

https://SynSense.ai
Partners

Conference Partner

Media Partners
Questions?
Or to join tinyML WeChat Group

Please add staff to join our official tinyML WeChat Group
Copyright Notice

This presentation in this publication was presented at tinyML® Asia 2020. The content reflects the opinion of the author(s) and their respective companies. The inclusion of presentations in this publication does not constitute an endorsement by tinyML Foundation or the sponsors.

There is no copyright protection claimed by this publication. However, each presentation is the work of the authors and their respective companies and may contain copyrighted material. As such, it is strongly encouraged that any use reflect proper acknowledgement to the appropriate source. Any questions regarding the use of any materials presented should be directed to the author(s) or their companies.

tinyML is a registered trademark of the tinyML Foundation.

www.tinyML.org