Deep Learning with Low Precision Hardware
Challenges and Opportunities for Logic Synthesis

Luca Benini
ETHZ & UNIBO

http://www.pulp-platform.org
Deep Learning: Why?

First, it was machine vision…

Now it’s everywhere!

Growing Use of Deep Learning at Google [J. Dean]

Deep Learning appears (green dots)
Deep neural networks (DNNs)

Application components:
- Task objective: e.g. Identify face
- Training data: 10-100M images
- Network architecture: ~10 layers, 1B parameters
- Learning algorithm: ~30 Exaflops, ~30 GPU days

Luca Benini
Key operation is dense $M \times V$

Layer-by-layer computation

\[ b_i = W_{ij} \times a_j \]

Output activations
Weight matrix
Input activations

Repeat for each layer

Training by recursive backpropagation of error on fitness function
GPUs are Great for Vanilla CNNs

Why?

Because they are good at matrix multiply → 90% utilization is achievable (on lots of “cores”)

Pascal GP100
3840 “cores”
3840 MAC/cycle
@ 1.4GHz
5.3 TMACS (FP)
@300W

*Volta with tensor engine claims 4x better E
HW for deep Networks: Frenzy

DNN Processing Units

- CPUs
- GPUs
- Soft DPU (FPGA)
- Hard DPU
- ASICs

FLEXIBILITY ➔ EFFICIENCY

Datacenter ➔ High-performance embedded ➔ Mobile

Mostly inference acceleration

Luca Benini
Information Processing in the IoT
Algorithmic Opportunities
DNNs Are Evolving Rapidly

Many efforts to improve efficiency

**Batching**
- Reduce bitwidth
- Sparse weights
- Sparse activations
- Compression
- Compact network

All applicable for inference
Some for training

**Deeper**
- More params?
- Larger model?

**Before 2000**
- LeNet5
  - 5 layers
  - Params: 1M
  - Model: 4MB

AlexNet
- (~80% Top5)
- 8 layers
- Params: 60M
- Model: 240MB

2012

2013

VGG
- (~89% Top5)
- 19 layers
- Params: 140M
- Model: 500MB

GoogleLeNet
- (~89% Top5)
- 22 layers
- Params: 6M
- Model: 24MB

2014

2015

ResNet
- (~94% Top5)
- 152 layers
- Params: 60M
- Model: 240MB

2016

[TernaryConnect [ICLR’16]
Spatially
SparseCNN
[CIFAR-10 winner ‘14]

SqueezeNet + DeepCompression:
- 6-bit, 20-50% sparse
- AlexNet accuracy, ~500x smaller (0.5MB)

XNORNet (1-bit) → ~2% AlexNet

TernaryNet (2-bit, 50% sparse) → ~1% ResNet

SqueezeNet + DeepCompression:
- 6-bit, 20-50% sparse
- AlexNet accuracy, ~500x smaller (0.5MB)

XNORNet (1-bit) → ~2% AlexNet

TernaryNet (2-bit, 50% sparse) → ~1% ResNet

Orders of magnitude compute effort and memory reduction with no loss in accuracy

Luca Benini
Toward Micropower CNN HW
Outline

- Near Threshold Multiprocessing
- Non-Von Neumann Accelerators
- Aggressive Approximation
- From Frame-based to Event-based Processing
- Outlook and Conclusion
Near-Threshold Multiprocessing

- 4-stage OpenRISC & RISC-V RV32IMC
- Shared L1 DataMem + Atomic Variables
- DMA Tightly Coupled DMA
- 1.. 8 PE-per-cluster, 1…32 clusters
- Near-Threshold Multiprocessing

Luca Benini
Extending RISC-V for CNNs

<32-bit precision → SIMD2/4 opportunity
1. Dot product between SIMD vectors
2. Shuffle operations for vectors
3. Packed-SIMD ALU operations
4. Bit manipulations
5. Rounding and Normalization

V1  Baseline RISC-V RV32IMC
    HW loops
V2  Post modified Load/Store
    Mac
V3  SIMD 2/4 + DotProduct + Shuffling
    Bit manipulation unit
    Lightweight fixed point

Small Power and Area overhead → Energy reduction in NT >3x
The Dot-Product instruction:
- Hardware-loops eliminate loop overhead

Sum-of-dot-product units:
- 8b version
- 16b version
- 1 cycle execution

- 7 Sum-of-dot-product
- 4 move
- 1 shuffle
- 3 lw/sw
- ~ 5 control instructions

20 instr. / output pixel
Scalar version >100 instr. / output pixel
**PULP-CNN ISA-Extensions**

Convolution Performance on PULP with 4 cores

- **speedup**: 3.9x
- **Energy gains**: 4.2x

*a) Cycles per Output Pixel*

*b) Energy for convolutions on a 64x64 image*

- 5x5 convolution in only 6.6 cycles/pixel

Luca Benini
The Memory Optimization Challenge

The canonical CNN 6-loop nest

- Memory hierarchy optimization
  - Sizing
  - Banking factors
- Data Movement engines
  - Shuffle instructions
  - DMA, prefetchers
- Loop optimizations
  - Reordering
  - Tiling
  - Double-buffering
Outline

- Near Threshold Multiprocessing
- Non-Von Neumann Accelerators
- Aggressive Approximation
- From Frame-based to Event-based Processing
- Outlook and Conclusion
Computational Effort

- **Computational effort**
  - 10-class scene labeling on Stanford-BG
  - 7.5 GOp/frame for 320x240 image (#Op=2 \times #MAC)
  - 260 GOp/frame for FHD
  - 1050 GOp/frame for 4k UHD

\[ \sim 90\% \text{ workload is Conv} \]

Origami CNN ASIC
**Origami: A CNN Accelerator**

- **FP not needed**: 12-bit signals sufficient
- Input to classification double-vs-12-bit accuracy loss < 0.5% (80.6% to 80.1%)
CNNs: typical workload

Example: ResNet-34
- classifies 224x224 images into 1000 classes
- ~ trained human-level performance
- ~ 21M parameters
- ~ 3.6G MAC operations

Scaling Origami to 28nm FDSOI

Performance for 10 fps: ~73 GOPS/s
Energy efficiency: ~2300 GOPS/W efficiency

Origami core in 28nm FDSOI → 10 fps ResNet-34 with ~32mW
Outline

- Near Threshold Multiprocessing
- Non-Von Neumann Accelerators
- Aggressive Approximation
- From Frame-based to Event-based Processing
- Outlook and Conclusion
Pushing Further: YodaNN

- Approximation at the algorithmic side → Binary weights

- **BinaryConnect** [Courbariaux, NIPS15], **XOR NET** [Rastegari, arXiv16]
  - Reduce weights to a binary value -1/+1
  - Stochastic Gradient Descent with Binarization in the Forward Path

\[
\begin{align*}
    w_{b,\text{stoch}} &= \begin{cases} 
    -1 & p_{-1} = \sigma(w) \\
    1 & p_1 = 1 - p_{-1} 
    \end{cases} \\
    w_{b,\text{det}} &= \begin{cases} 
    -1 & w < 0 \\
    1 & w > 0 
    \end{cases}
\end{align*}
\]

- Learning large networks is still challenging, but starts to become feasible:
  
  ResNet-18 on ImageNet with **83.0%** (binary-weight) vs. **89.2%** (single-precision) top-5 accuracy; and **60.8%** vs. **69.3%** top-1 accuracy

- **Ultra-optimized HW is possible!**
  - Major arithmetic density improvements: **MAC → 2s compl. & Accum.**
    - Area can be used for more energy-efficient weight storage
  - Storage reduction → SCM memories for lower voltage → E goes with \(1/V^2\)

\[\text{Later added: After the Yedi Master from Star Wars - “Small in size but wise and powerful” cit. www.starwars.com}\]

Luca Benini
SoP-Unit Optimization

Equivalent for 7x7 SoP

Image Mapping (3x3, 5x5, 7x7)


Luca Benini
YODANN Energy Efficiency

Same area 8→32 SoP units + all SCM

Core Energy Efficiency [TJ/op/W]

Supply Voltage $V_{core}$ [V]

0.6 | 0.7 | 0.8 | 0.9 | 1 | 1.1 | 1.2

Core Energy Eff. Q2.9/8x8/SRAM (●●●), Bin./32x32/SCM (●●●●), Throughput Q2.9/8x8/SRAM (●●●●), Bin./32x32/SCM (●●●●)

12x Boost in core energy efficiency (single layer)

0.03pj/OP
Outline

- Near Threshold Multiprocessing
- Non-Von Neumann Accelerators
- Aggressive Approximation
- From Frame-based to Event-based Processing
- Outlook and Conclusion
Back to System-Level

Smart Visual Sensor $\rightarrow$ idle most of the time (nothing interesting to see)

- **Event-Driven Computation**, which occurs only when relevant events are detected by the sensor
- **Event-based sensor interface** to minimize IO energy (vs. Frame-based)
- **Mixed-signal event triggering** with an ULP imager, cochlea with internal processing AMS capability

A Neuromorphic Approach for doing *nothing* VERY well
GrainCam Imager

Pixel-level spatial-contrast extraction

\[ V_C = V_{PE}(t_1) - V_{PO}(t_1) = (V_R - V_{TH}) \left( \frac{I_{PO} - I_{PE}}{I_{PE}} \right) \]

Analog internal image processing

- Contrast Extraction
- Motion Extraction, differencing two successive frames
- Background Subtraction with the reference image stored in pixel memory

[Gottardi, JSSC09]
Graincam Readout

Readout modes:
- **IDLE**: readout the counter of asserted pixels
- **ACTIVE**: sending out the addresses of asserted pixels (address-coded representation), according raster scan order

**Event-based sensing**: output frame data bandwidth depends on the external context-activity

---

Ultra Low Power Consumption e.g. 10-20uW @10fps
Even-driven CNNs? Yes!

Binary Neural Networks reduce precision of weights and post-activation neurons to 1-bit precision while leading to a limited performance drop.

Benefits:

- Reduced storage costs by **32x** either for synaptic weights and temporary input/output data.
- Reduced computation complexity.

Convolution reduce to:

\[ C = \text{popcount} \left( \text{NOT} (\text{weights XOR image}) \right) \]

Binarization:

\[ \text{output} = C > 0 \ ? \ 1 \ else \ 0 \]

Performing spatial filtering and binarization on the sensor die through mixed-signal sensing! \( \rightarrow \) in-sensor first stage of the binary NN!!

‘Moving’ pixel window

Gradient extraction

Per-pixel circuit for filtering and binarization
Event-Driven Binary Deep Network

Digital pixel sampling

Spatial-local filtering and binarization

Mixed-Signal Sensing

Digital Signal Processing

Layers with binary inputs and binary weights

Binary Neural Network (BNN)

Event-based Binarized Neural Network

Binary Input Layer

Output Layer

Integer data

Integer Input Layer

Output Layer

Binary Input Layer

Input Layer

Luca Benini
Training challenge

**Training** Event-based Binarized Neural Network:

[**ISSUE**] Absence of huge amount of data for training

Modelling the “graincam filter” as a digital filter

\[
V_C \sim \frac{\max(|p_E-p_O|, |p_N-p_O|)}{\max(p_E,p_O,p_N)}
\]

Binary Output

\[
V_O = sgn(V_C - V_{th})
\]

Evaluation on **CIFAR-10** (10 classes, 45k training, 5k valid, 10k testing)

<table>
<thead>
<tr>
<th>Model Type</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Baseline with RGB input</strong></td>
<td>92%</td>
</tr>
<tr>
<td><strong>BNN with RGB input</strong></td>
<td>86%</td>
</tr>
<tr>
<td><strong>Baseline with binary input</strong></td>
<td>72%</td>
</tr>
<tr>
<td><strong>BNN with binary input</strong></td>
<td>68%</td>
</tr>
</tbody>
</table>

Model VGG-like with 12 Convolutional layers and 3 Fully Connected Layers

18% performance drop because of input representation but still converges

Original RGB image  Synthetic image  Graincam image
Results

TRAINING DATASETS

Training: 60k samples
Validation: 900 samples

KITTI dataset [1] – autonomous driving

ACCURACY

Model: VGG-like architecture

<20kB "binary weight program"

CONV3x3(#c,16) + POOLING
CONV3x3(16,32) + POOLING
CONV3x3(32,48) + POOLING
CONV3x3(48,64) + POOLING
CONV3x3(64,96) + POOLING
FULLY-CONNECTED (384,64)
FULLY-CONNECTED (64,3)

BNN with RGB input

84.6%

Training converges with a 3% performance drop

BNN with binary input

81.6%


Luca Benini
BNN implementation on PULP

Basic Convolution Operation:

\[ \rho(x, y) = \sum_{(d,i,j)\in C} w(d, i, j) \ast I(d, i, j, x, y) \]

\[ I, w \in \{0,1\} \]

\[ \rho(x, y) = \text{popcount}\{ \text{NOT} (w \text{ XOR } I(x, y)) \} \]

Batch Normalization and binarization:

\[ o_z(x, y) = \frac{\rho(x, y) + b - \mu}{\sigma} \gamma + \beta \geq 0 \]

\[ \rho(x, y) \in N \]

\[ \text{if } \gamma \geq 0 \text{ then } o_z(x, y) = \rho(x, y) \leq \left[ \mu - b - \frac{\beta \ast \sigma}{\gamma} \right] \text{ else } o_z(x, y) = \rho(x, y) \geq \left[ \mu - b - \frac{\beta \ast \sigma}{\gamma} \right] \]

just logic operation and integer comparison!

Major opportunity for HW acceleration!
Preliminary Results

<table>
<thead>
<tr>
<th>Scenario</th>
<th>BNN with RGB input</th>
<th>Event-based BNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image Sensor Power Consumption</td>
<td>1.1mW @ 30fps</td>
<td>100μW @ 50fps</td>
</tr>
<tr>
<td>Image Size</td>
<td>632446 bits</td>
<td>8192 bits</td>
</tr>
<tr>
<td>Image Sensor Energy for frame capture</td>
<td>66.7 μJ</td>
<td>2 μJ</td>
</tr>
<tr>
<td>Transfer Time (4bit SPI @ 50MHz)</td>
<td>3.1 msec</td>
<td>0.04 msec</td>
</tr>
<tr>
<td>Transfer Energy (8.9mW @ 0.7V)</td>
<td>28 μJ</td>
<td>2 μJ</td>
</tr>
<tr>
<td>BNN Execution Time (168MHz)</td>
<td>81.3 msec</td>
<td>75.3 msec</td>
</tr>
<tr>
<td>BNN Energy consumption (8.9mW @ 0.7V)</td>
<td>725 μJ</td>
<td>671 μJ</td>
</tr>
<tr>
<td>Total System Energy for Classification</td>
<td>820 μJ</td>
<td>674 μJ</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Statistics per frame</th>
<th>Frame-Based</th>
<th>Event-based</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Idle (no motion)</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Power</td>
<td>1.1mW</td>
<td>20μW</td>
</tr>
<tr>
<td>Avg Sensor Data</td>
<td>19764 Bytes</td>
<td></td>
</tr>
<tr>
<td>Transfer Time</td>
<td>790μsec</td>
<td></td>
</tr>
<tr>
<td>Processing Time</td>
<td>3.02 msec</td>
<td></td>
</tr>
<tr>
<td>Avg Processor Power</td>
<td>1.45mW</td>
<td>0.3mW (sleep)</td>
</tr>
<tr>
<td><strong>Detection</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Power</td>
<td>1.1mW</td>
<td>60μW</td>
</tr>
<tr>
<td>Avg Sensor Data</td>
<td>19764 Bytes</td>
<td>~536 Bytes</td>
</tr>
<tr>
<td>Transfer Time</td>
<td>790μsec</td>
<td>21.4μsec</td>
</tr>
<tr>
<td>Processing Time</td>
<td>3.47 msec</td>
<td>187.μsec</td>
</tr>
<tr>
<td>Avg Processor Power</td>
<td>1.57mW</td>
<td>0.511mW</td>
</tr>
<tr>
<td><strong>Classification</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sensor Power</td>
<td>2mW</td>
<td>60μW</td>
</tr>
<tr>
<td>Avg Sensor Data</td>
<td>79056 Bytes</td>
<td>1024 Bytes</td>
</tr>
<tr>
<td>Transfer Time</td>
<td>3.16 msec</td>
<td>41μsec</td>
</tr>
<tr>
<td>Processing Time</td>
<td>81.3 msec</td>
<td>75.3 msec</td>
</tr>
<tr>
<td>Processor Energy</td>
<td>760 μJ</td>
<td>677 μJ</td>
</tr>
</tbody>
</table>

84.6% vs. 81.6% Accuracy
Outline

- Near Threshold Multiprocessing
- Non-Von Neumann Accelerators
- Aggressive Approximation
- From Frame-based to Event-based Processing
- Outlook and Conclusion
Conclusions

- Near-sensor processing for the IoT
- **CNNs can be taken into the ULP (mW power envelope) space**
  - Non-von-Neumann acceleration
  - Very robust to low precision computations (deterministic and statistical)
  - fJ/OP is in sight!
- **Major synthesis challenges**
  - Memory optimization: automatic exploration of Archi+Loop
  - Automatic precision tuning of datapath
  - Boolean training
- **Open Source HW & SW approach** → innovation ecosystem
Morale:
Plenty of room at the bottom

Thanks!!!

www.pulp-platform.org
www-micrel.deis.unibo.it/pulp-project
iis-projects.ee.ethz.ch/index.php/PULP
Origami, YodaNN vs. Human

The «energy-efficient AI» challenge (e.g. Human vs. IBM Watson)

<table>
<thead>
<tr>
<th>Type</th>
<th>Analog (bio)</th>
<th>Q2.9 Precision</th>
<th>Q2.9 Precision</th>
<th>Binary-Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top-1 error [%]</td>
<td></td>
<td>21.53</td>
<td>30.7</td>
<td>39.2</td>
</tr>
<tr>
<td>Top-5 error [%]</td>
<td>5.1</td>
<td>5.6</td>
<td>10.8</td>
<td>17.0</td>
</tr>
<tr>
<td>Hardware</td>
<td>Brain</td>
<td>Origami</td>
<td>Origami</td>
<td>YodaNN</td>
</tr>
<tr>
<td>Energy-eff. [uJ/img]</td>
<td>100.000(*)</td>
<td>1086</td>
<td>543</td>
<td>31</td>
</tr>
</tbody>
</table>

*P_{brain} = 10W, 10% of the brain used for vision, trained human working at 10img/sec

- Game over for humans also in energy-efficient vision?
- .... Not yet! (object recognition is a super-simple task)
CNN Workloads

Better networks are not necessarily more complex

[Culurciello16]
Recovering silicon efficiency

<table>
<thead>
<tr>
<th>Area for same GOPS</th>
<th>Throughput Computing</th>
<th>Non-Von Neumann</th>
<th>Fully hardwired</th>
</tr>
</thead>
<tbody>
<tr>
<td>General-purpose Computing</td>
<td>CPU</td>
<td>GPGPU</td>
<td>Accelerator Gap</td>
</tr>
</tbody>
</table>

Closing The Accelerator Efficiency Gap with Agile Customization

Luca Benini