CUTIE – Beyond PetaOp/s/W Ternary DNN Acceleration

Moritz Scherer
scheremo@iis.ee.ethz.ch


Special Thanks to:
Georg Rutishauser, Lukas Cavigelli, Luca Benini
Energy Efficiency is Everything

- TinyML requires ultra-low power & tiny memory footprint
  - Typical battery-based applications: single digit mW
  - Energy harvesting systems: 100s of μW
  - Typical sensor node: 100s of KB of memory

- Trend points to ultra-low precision
  - Fixed point and sub-byte networks optimize energy-accuracy tradeoff
  - Even binarized networks can achieve reasonable accuracy

- Acceleration is key to unlocking full potential
Acceleration = Exploiting Parallelism

- **Textbook State of the Art: Systolic array**
  - Scales to many-chip systems
  - Designed for flexibility
  - Heavily pipelined

- **Most energy is still NOT spent on computations**
  - Huge overheads in clocking & data movement
  - Core computational energy between 10-30%
  - All the rest is memory, data movement and control!

Source: EyeRiss homepage, https://eyeriss.mit.edu/

Source: "Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices", Chen et al., 2018
Motivation & Contribution

How can we maximize energy-efficiency in low power digital CNN Accelerators on the architectural level?
Motivation & Contribution

How can we maximize energy-efficiency in low power digital CNN Accelerators on the architectural level?

- Minimize data movement
  - Keep weights and partial results local
Motivation & Contribution

How can we maximize energy-efficiency in low power digital CNN Accelerators on the architectural level?

- Minimize data movement
  - Keep weights and partial results local

- Maximize computational efficiency
  - Ultra-low precision operands
  - Completely unrolled, parallel architecture
  - Minimize switching activity
Motivation & Contribution

How can we maximize energy-efficiency in low power digital CNN Accelerators on the architectural level?

- **Minimize data movement**
  - Keep weights and partial results local

- **Maximize computational efficiency**
  - Ultra-low precision operands
  - Completely unrolled, parallel architecture
  - Minimize switching activity

- **Leverage irregular sparsity in computation**
  - Use ternary weights & activations over binary
  - Sparsity-aware training
Acceleration: The current trend in TinyML

System Architecture – OCU

- The compute core of CUTIE
  - Layer-by-layer network execution
  - Keep all weights in local buffer
  - Multiply-and-add activations and weights
System Architecture – OCU

- Minimize switching activity
  - **Completely unrolled** inner products instead of MACs: All MACs in one cycle
  - Zeros in weights and activations reduce switching activity
System Architecture – OCU

- **Support for Pooling**
  - Support max and average pooling
  - Silence additional hardware if no pooling
System Architecture – OCU

- Re-ternarize results with dual thresholds
  - Fold convolutional and batchnorm biases into thresholds
  - Fold batchnorm scaling into thresholds
System Architecture – Data Path

- Completely unrolled compute architecture
  - Limit maximal number of channels
  - One compute unit per channel
  - Local storage minimizes data movement
System Architecture – Data Path

- **Completely unrolled compute architecture**
  - Limit maximal number of channels
  - One compute unit per channel
  - Local storage minimizes data movement

- **Compressed ternary storage**
  - Minimize required memory → 1.6 Bits / Operand
System Architecture – Data Path

- Completely unrolled compute architecture
  - Limit maximal number of channels
  - One compute unit per channel
  - Local storage minimizes data movement

- Compressed ternary storage
  - Minimize required memory → 1.6 Bits / Operand

- Very light pipelining
  - Keep everything as close to combinational as possible
  - Minimize clocking overheads
  - Use registers to silence unused units
System Architecture – Data Path

- Completely unrolled compute architecture
  - Limit maximal number of channels
  - One compute unit per channel
  - Local storage minimizes data movement

- Compressed ternary storage
  - Minimize required memory → 1.6 Bits / Operand

- Very light pipelining
  - Keep everything as close to combinational as possible
  - Minimize clocking overheads
  - Use registers to silence unused units

- Dedicated weight & feature map memory
  - Large enough to store worst-case feature maps + weights
  - I/O is extremely expensive
Implementation

- **CUTIE is highly parametrizable**
  - Channels, kernel shapes, pipeline depth, memory sizes, ...

- **Configuration parameters**
  - 128 channels
  - 3 x 3 kernels
  - 32 x 32 pixels feature maps

- **Evaluated network**
  - 9 layers
  - Convolution $\rightarrow$ BatchNorm $\rightarrow$ Hardtanh
Results – Numbers

- **TSMC 7 nm:**
  - Avg. energy efficiency: 2.1 POp/s/W
  - **Peak energy efficiency:** 3.1 POp/s/W
  - Area: 1.2 mm²

- **Inference on CIFAR-10:**
  - Accuracy: 88% vs. 86%[1]
  - Clock frequency: 66 MHz
  - Average inference power: 7.8 mW
  - FPS: 13.68k
  - Energy per inference: 0.52 µJ vs 13.76 µJ[1]
  - Peak Throughput: 16 TOp/s

[1]: Moons et al.: BinarEye: An Always-On Energy-Accuracy-Scalable Binary CNN Processor With All Memory On Chip in 28nm CMOS
Results – Insights

- **Ternary > Binary:**
  - 50% higher energy efficiency for same network & accelerator
  - 4% higher accuracy for same network architecture
  - 4.8x lower energy per inference at iso-accuracy

- **Fully unrolled > Iterative**
  - Significantly less data movement
  - Spatial smoothness reduces switching activity by > 2x

- **Training matters**
  - 1.5x higher energy efficiency with sparser networks

- **Stay tuned for our tape-out in GF 22 nm!**