Introducing the Arm Machine Learning (ML) Processor

- Optimized ground-up architecture for machine learning processing
- Massive efficiency uplift from CPUs, GPUs and DSPs
- Open-source stack enables easy deployment
- Architecture scales from IoT to server and automotive
- First design targets mobile with derivatives for additional segments
Arm’s ML Processor

- 16 Compute Engines
- ~ 4 TOP/s of convolution throughput (at 1 GHz)
- Targeting > 3 TOP/W in 7nm and ~2.5mm²
- 8-bit quantized integer support
- 1MB of SRAM
- Support for Android NNAPI and ARMNN
- Optimized for CNNs, RNN support
- To be released 2018
4 Key Ingredients for a Machine Learning Processor

- Static scheduling
- Efficient convolutions
- Bandwidth reduction mechanisms
- Programmability/flexibility
4 Key ingredients for a Machine Learning Processor

• Static scheduling

• Efficient convolutions

• Bandwidth reduction mechanisms

• Programmability/flexibility
Arm’s ML processor: Static Scheduling

- CNNs are statically analyzable
- Compiler takes a NN and maps it to a command stream consumed by the ML processor

NN Compiler ➔ Command Stream

Command Stream

- DMA X
- DMA Y
- WAIT for DMA (X,Y)
- Conv X, Y
- etc

ML Processor

- DMA Engine
- Control Unit
- SRAM
- MAC Engine
- Weight Decoder
- Programmable Layer Engine
- Compute Engine 1
- Compute Engine 16
Arm’s ML processor: Static Scheduling

- No caches
- Simplified flow control
- Simplified hardware (but requires careful co-design with the compiler)
- Relatively predictable performance

NN Compiler

Command Stream

DMA X
DMA Y
WAIT for DMA (X,Y)
Conv X, Y
etc

ML Processor

DMA Engine
Control Unit
Weight Decoder
MAC Engine
Programmable Layer Engine
SRAM
Compute Engine 1
Compute Engine 16
4 Key ingredients for a Machine Learning Processor

- Static scheduling
- Efficient convolutions
- Bandwidth reduction mechanisms
- Programmability/flexibility
Convolutions

- Output Feature Maps (OFMs) are interleaved across the compute engines (each CE working on a different OFM)

- The weights for OFM-X will be resident in the SRAM of the CE which is processing OFM-X

- Input Feature Maps (IFMs) are interleaved across all SRAM banks
Convolutions

- MAC Engine capable of eight 16-wide dot products (8b)
  - MAC Engine = 2 * 8 * 16 = 256 ops/cycle
  - 16 MAC Engines. = 16 * 256 = 4096 ops/cycle
  - 4.1 TOPs @ 1 GHz
  - 32b accumulators

- The utilization of the MAC engine depends on conv parameters

- Datapath gating for zeros (~50% power reduction)
Each "16 Wide DP" unit performs an 8b, 16 deep dot product operation.
Convolutions

Activations from other Compute Engines → Broadcast Network →

IFMs → Input Activation Read
Compressed Weights → Weight Decoder
SRAM → MAC Engine
OFMs → Programmable Layer Engine

Accumulators

16 Wide DP → 16 Wide DP → 16 Wide DP → 16 Wide DP

MAC Engine

A tensor of activations is assembled in the broadcast network and sent to all MAC engines.
The weights for a specific OFM are resident in the local SRAM slice paired with the MAC Engine.

The weights are read, decompressed and sent to the MAC Engine.
When the final output activation values have been communicated, the 32b values are scaled back to 8b and sent to the Programmable Layer Engine.
Convolutions

• POP IP for the MAC Engines, Tuned for 16nm and 7nm
  • Providing 40% area reduction and 10-20% power improvements
4 Key ingredients for a Machine Learning Processor

- Static scheduling
- Efficient convolutions
- Bandwidth reduction mechanisms
- Programmability/flexibility
Importance of Weight and Feature Map Compression

- DRAM power can be nearly as high as the processor power itself
- ML processor supports
  - Weight Compression
  - Activation Compression
  - Tiling

![Power Breakdown Chart]

- Weight DDR Power
- Activation DDR Power
- ML Processor Power
ML Processor Feature Map Compression

ML processor saves average of 3x with lossless compression

- Compression per 8x8 block
- 3.3x compression for Inception V3

Many maps have repeating non-zeros, again aiding compression

Standard padding behaviors for tensors introduce more zeros

High zero count indicates good compression behavior

Source: Arm Machine Learning group

Count of zeros per 8x8 block

Unique non-zero values

Frequency
Weight Compression and Pruning

- Weight bandwidth dominates later layers of networks
- Pruning during the training phase increases the number of zeros
- Clustering can “snap” the remaining non-zero weights to a smaller set of possible NZ values
- Models are compressed offline during compilation phase to our format which exploits both clustering and pruning
- Weights stay compressed until read from internal SRAM

Inception v4

<table>
<thead>
<tr>
<th>Convolution layer #</th>
<th>Input</th>
<th>Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>120</td>
<td></td>
<td></td>
</tr>
<tr>
<td>160</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Learning both weights and connections for efficient neural networks

Han et al
October 2015
Tiling

• Compiler-based scheduling further reduces bandwidth
  • Scheduling tuned to keep working set in SRAM
  • Tiled or wide scheduling avoids trips to DRAM
  • Multiple outputs calculated in parallel from same input
  • Intermediate stages are pipelined between MAC and PLE
  • Possible because of static scheduling (compile time)

Szegedy et al
Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning
February 2016
4 Key ingredients for a Machine Learning Processor

- Static scheduling
- Efficient convolutions
- Bandwidth reduction mechanisms
- Programmability/flexibility
Programmable Layer Engine (PLE)

- State of the art in neural networks is still evolving

- Programmable Layer Engine
  - Provides design future-proofing
  - Benefits from existing Arm technology

- No hardware assumptions on operator ordering
• We have extended ARM CPU technology with vector and NN extensions targeted for non-convolution operators (pooling, relu, etc)
Programmable Layer Engine (PLE), cont.

- The results of MAC computation are sent to the PLE
  - The PLE register file is populated directly
  - Interrupts are sent to activate PLE processing
  - The majority of operators are performed by a 16-lane vector engine – as they often pool or reduce

- Results are emitted back to SRAM
  - A micro-DMA unit writes data out
  - They are then fetched back into CE for subsequent processing
Scalability

- Multiple ways to scale
  - Number of Compute Engines
  - MAC Engine throughput
  - Number of ML processors
Arm’s ML processor: Summary

- 16 Compute Engines
- ~ 4 TOP/s of convolution throughput (at 1 GHz)
- Targeting > 3 TOP/W in 7nm and ~2.5mm²
- 8-bit quantized integer support
- 1MB of SRAM
- Support for Android NNAPI and ARMNN
- To be released 2018
Thank You
Danke
Merci
谢谢
ありがとうございます
Gracias
Kiitos
감사합니다
धन्यवाद
תודה