Arm's ML processor architecture key features

Efficient convolutions

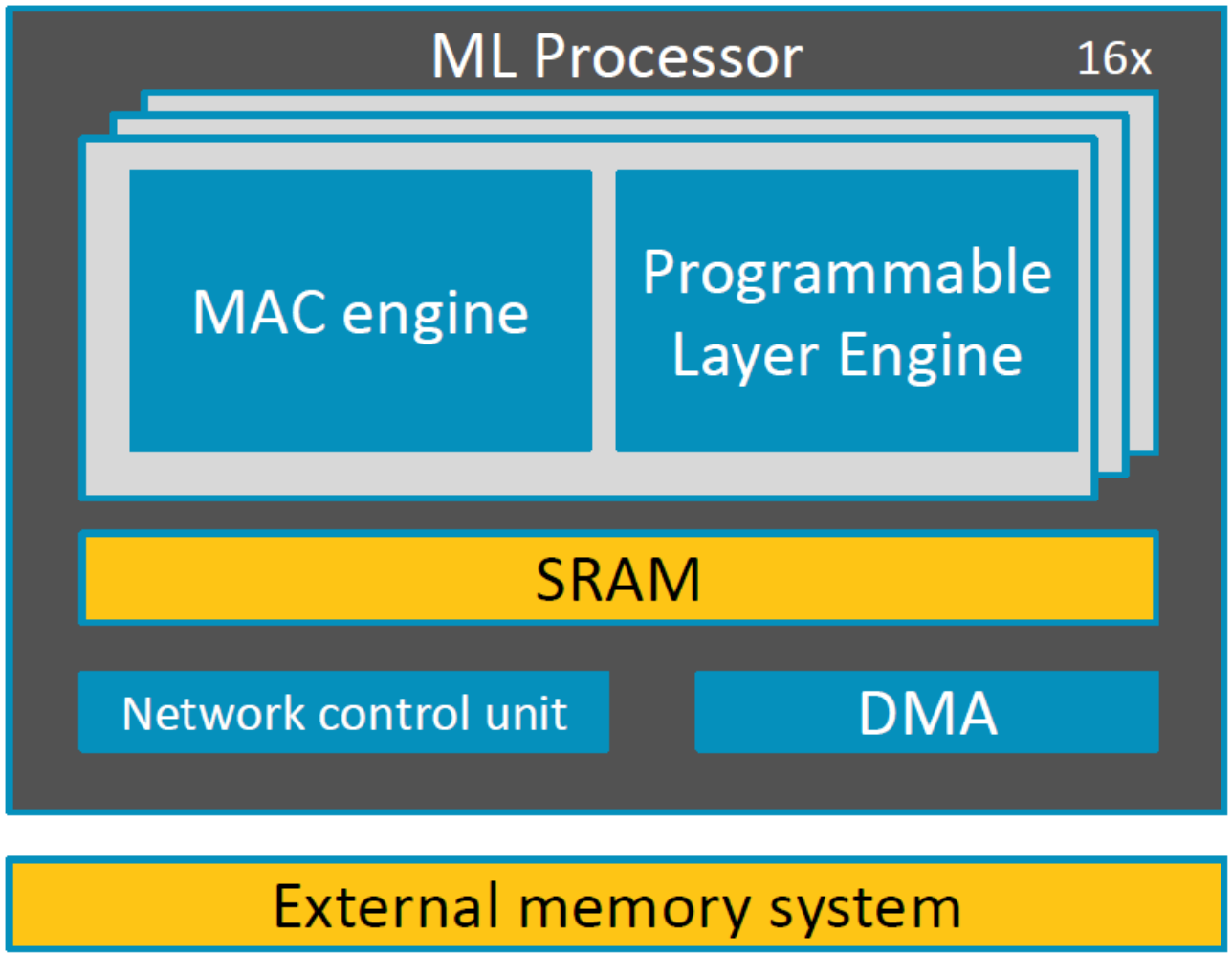
- Convolutions represent the bulk of computation
- We provide dedicated 8-bit hardware for convolutions

Efficient data movement

- More energy is spent moving data than computing
- We amortize activation accesses and compress weights

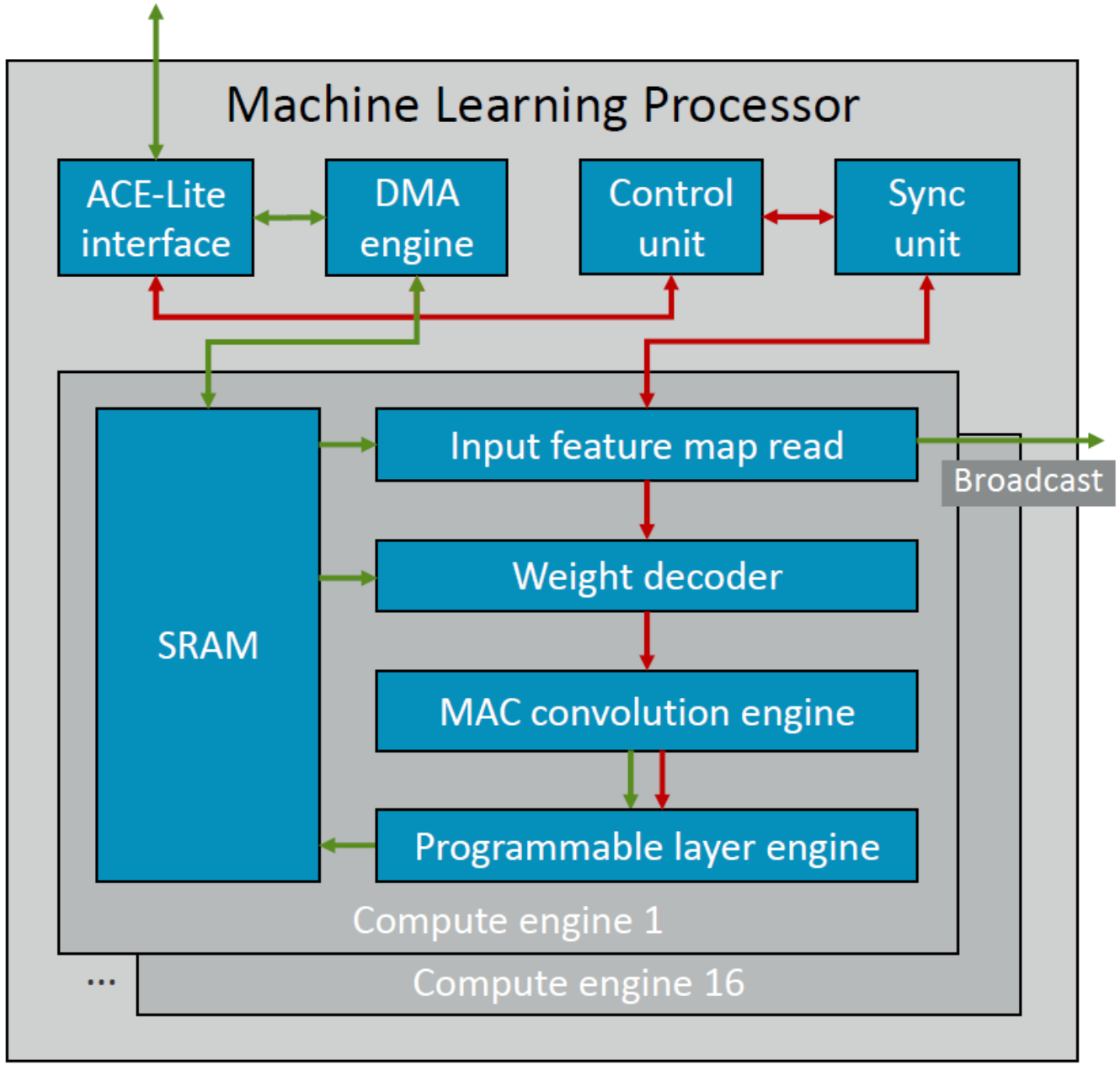
Sufficient programmability

- New operators are invented and topology is changed frequently
- We provide programmability to future-proofed as new network architectures appear



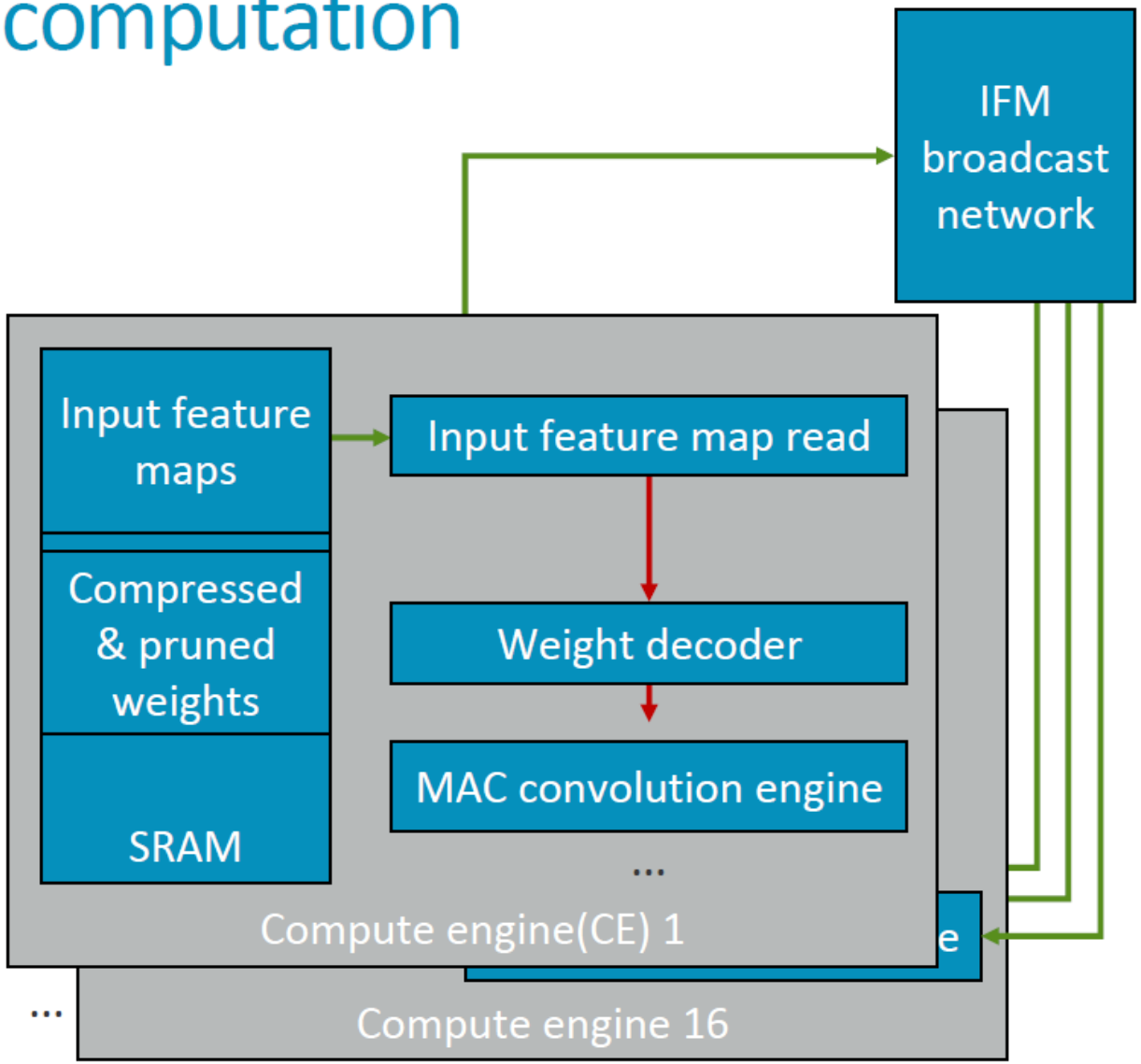
Arm's ML processor

- A microcontroller and DMA engine manage overall network scheduling
- The compute engine processes major sections of the neural network
  - Stores weights
  - Stores and manipulates activation data
  - Handles convolution in 128-wide MAC units
  - Handles other layer operators via PLE
  - Pipelines data to and from SRAM
- Internal broadcast network manages SRAM population and synchronization



Convolution and vector product computation

- Feature maps are read from SRAM
- The same map is broadcast to each CE
  - Direct to the MAC unit
  - Effectively reducing bandwidth / energy use
- Each CE applies different weights
  - Producing 16 unique Output Feature Maps
- Output is issued direct to the PLE
- Results from the PLE written back to SRAM



Programmable Layer Engine (PLE)

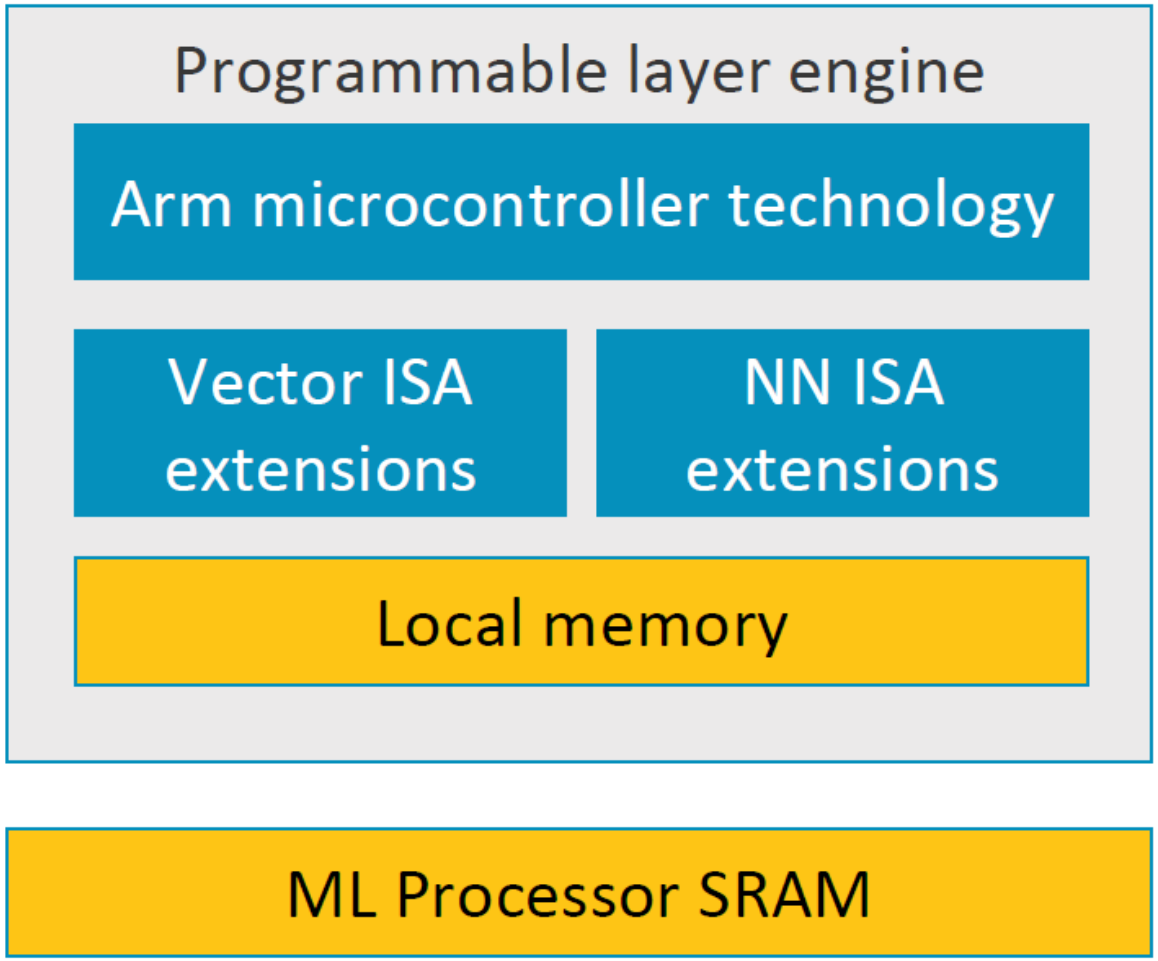
State of the art in neural networks is still evolving

High-efficiency programmability is a key benefit for the ML processor

- This provides design future-proofing
- Benefiting from existing Arm technology for better development and toolchains

Key operations are further accelerated

- Pooling, activations and compression



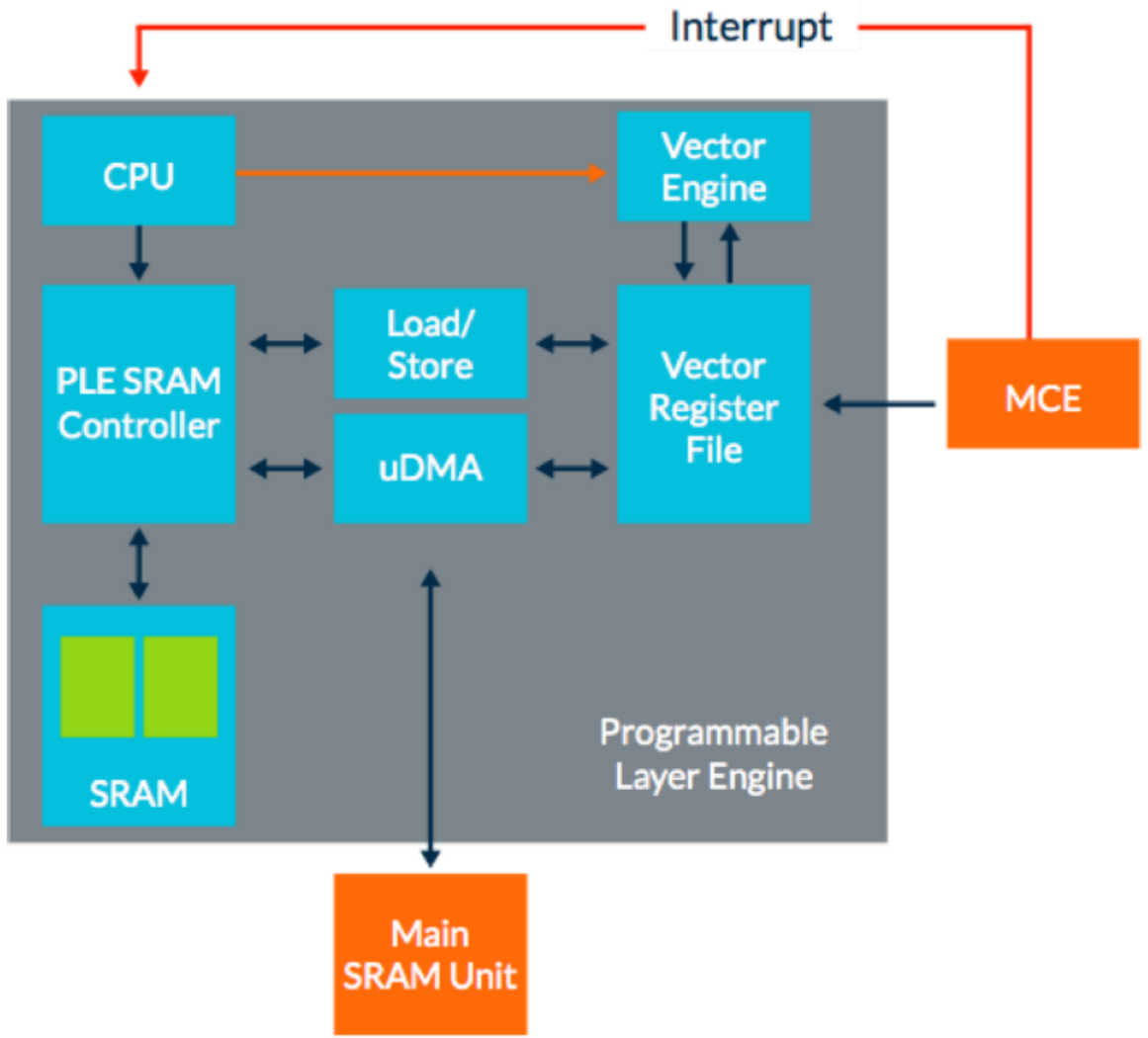
Programmable Layer Engine (PLE), cont.

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



Efficient convolution

Power and area efficiency for key operations

The ML processor works on 8-bit datatypes

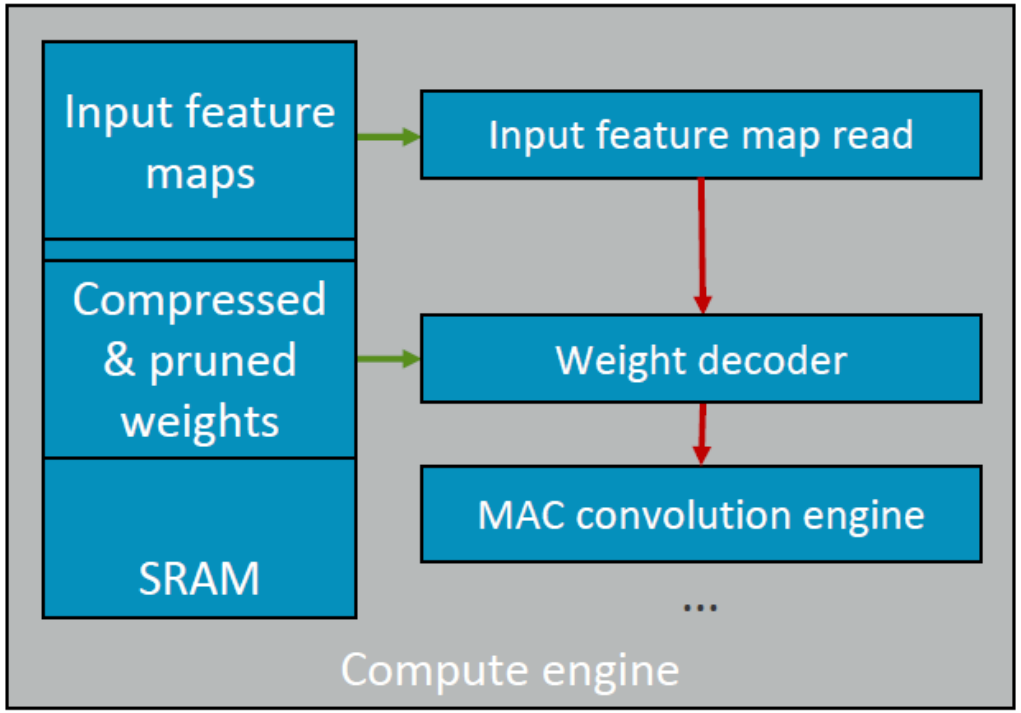
- Processing in the quantized domain
- With varied internal precision

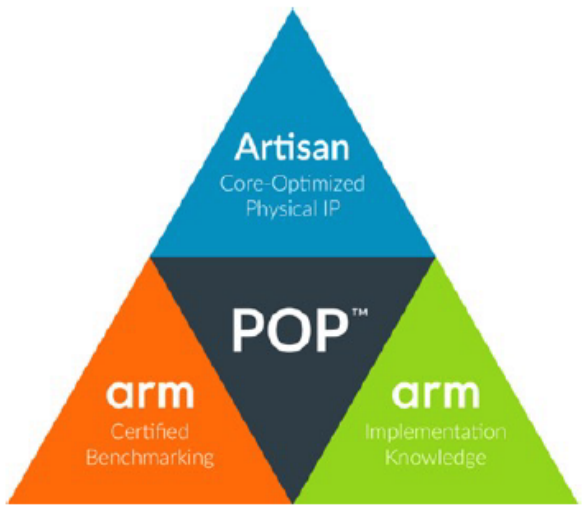
Exploiting common data for convolutions

- Decompressing weights once to minimize bandwidth
- Broadcast sharing data for activations between compute engines

Tuned for maximum efficiency

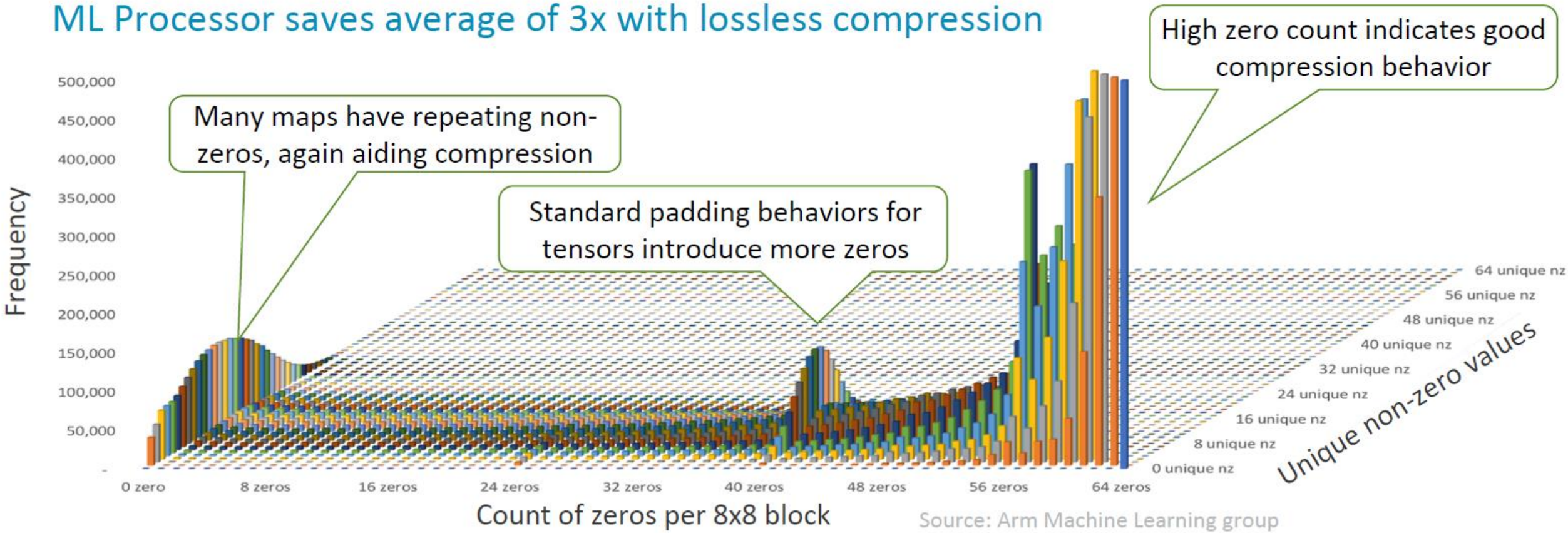
- With POP IP for the compute engines, Tuned for 16nm and 7nm
- Providing 40% area reduction and 10-20% power improvements





ML Processor feature map compression

ML Processor saves average of 3x with lossless compression

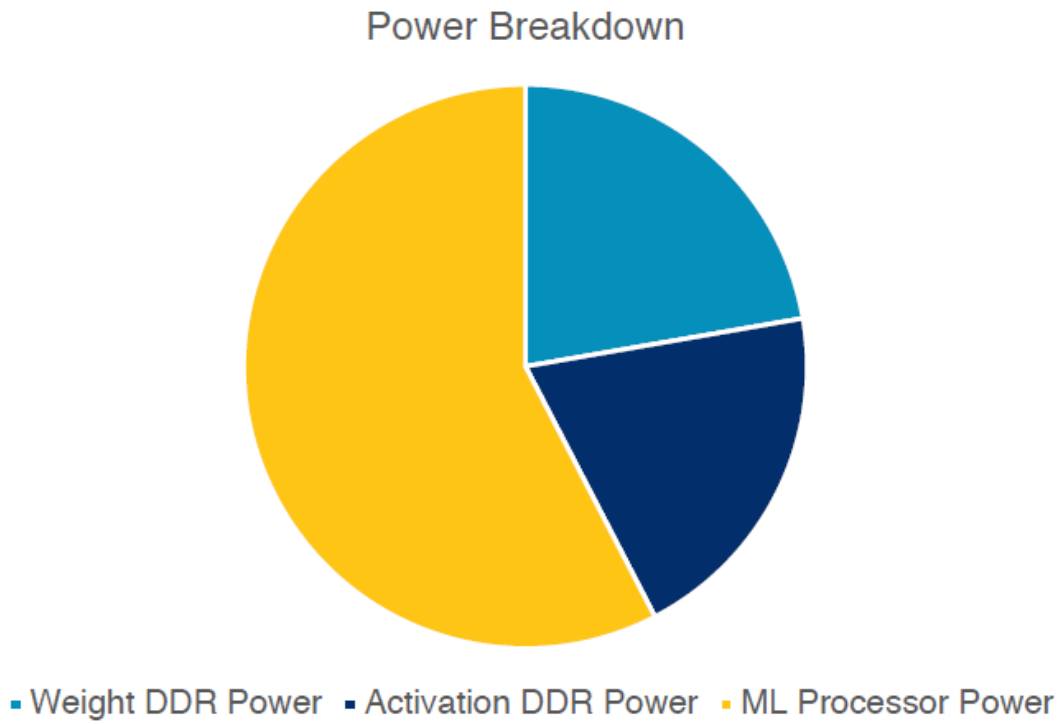


Source: Arm Machine Learning group

- Compression works in a block-specific way
- For GoogleNet V3 we average about 3.3x compression including metadata

Importance of weight and feature map compression

- DRAM power can be nearly as high as the accelerator power itself
- Both weights and feature maps are compressed losslessly on ML processor
- Data compression also
  - Lowers processor power by reducing SRAM accesses
  - Reduces internal storage requirements
- External traffic is reduced further through optimal network traversal generated by our network compiler



GoogleNet Inception v3 estimated power breakdown

* Assumes 125mW/GB/s for DRAM power

Weight compression and pruning

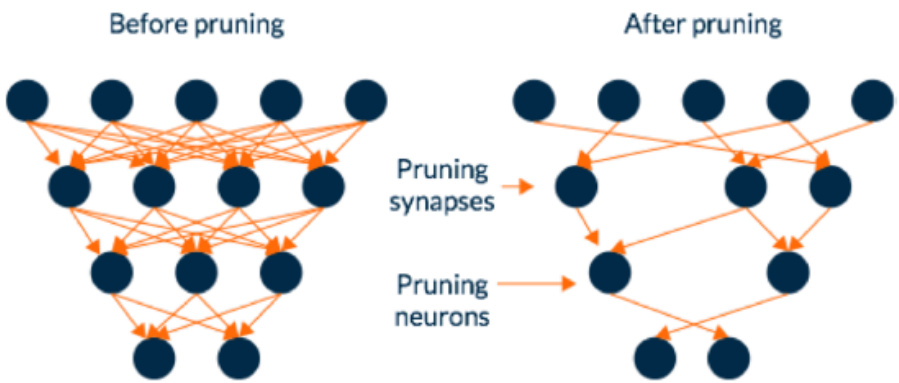
Weight bandwidth dominates later layers of networks

- Weights have many zeros if well trained
- Both compression and pruning are applied

Weight data is compressed offline as zero-mask + indices

- Network static and dynamic storage compresses ~4x
- This format also means computation can be skipped

Both pruning and clustering are supported in hardware and can significantly reduce number of MACs performed



Han et al
Learning both weights and Connections for Efficient Neural Networks
October 2015

Arm's ML processor

Efficient convolutions

- Targeting >3 TOP/W in 7nm
- 4.6 TOP/s design throughput
- Operates on 8-bit quantized integers

Efficient data movement

- Scalable SRAM size

Sufficient programmability

- Supports Android NNAPI and Arm NN
- Able to add new operators

To be released mid-2018