CMix-NN: Mixed Low-Precision CNN Library for Memory-Constrained Edge Devices

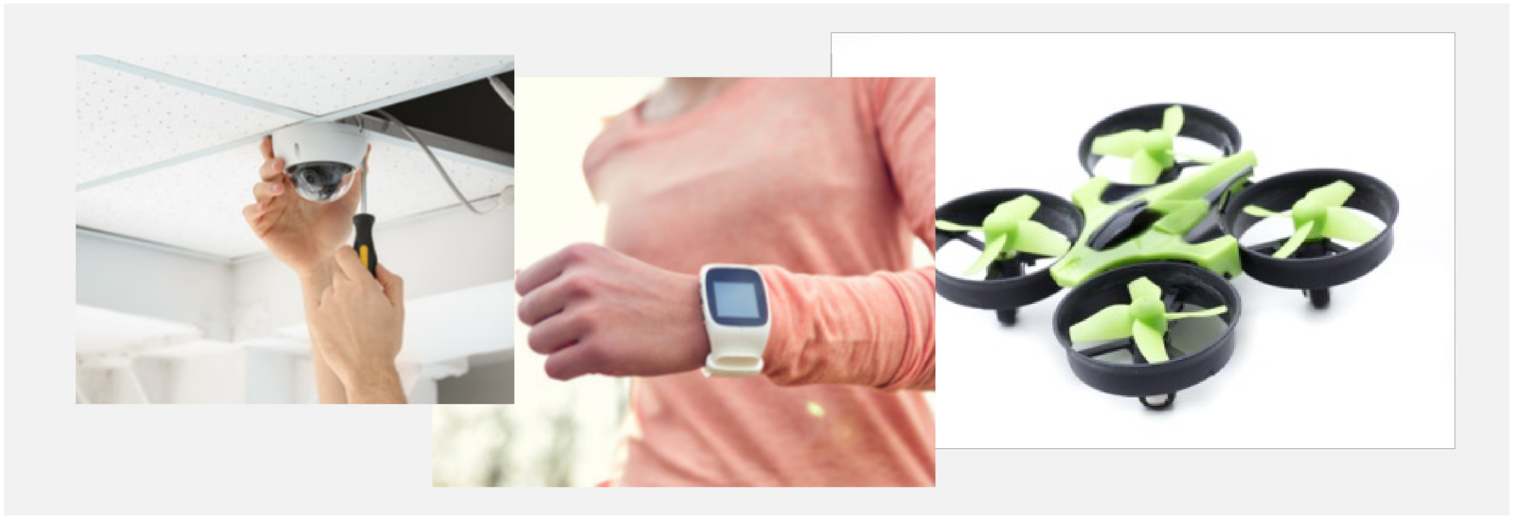
Alessandro Capotondi*, Manuele Rusci, Marco Fariselli, Luca Benini

**alessandro.capotondi@unimore.it*

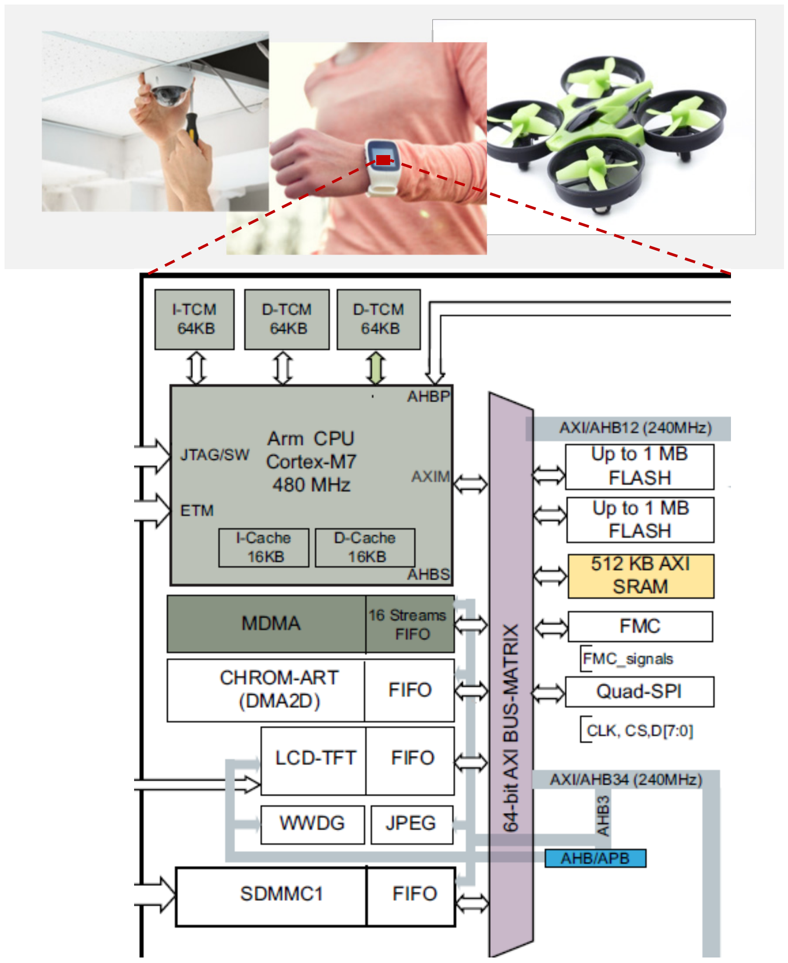
Hipert Lab
Department of Physics, Mathematics and Informatics
Università di Modena e Reggio Emilia

Energy-Efficient Embedded Systems Laboratory
Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione
Università di Bologna

Edge Smart Devices



Microcontrollers for tinyML at the edge

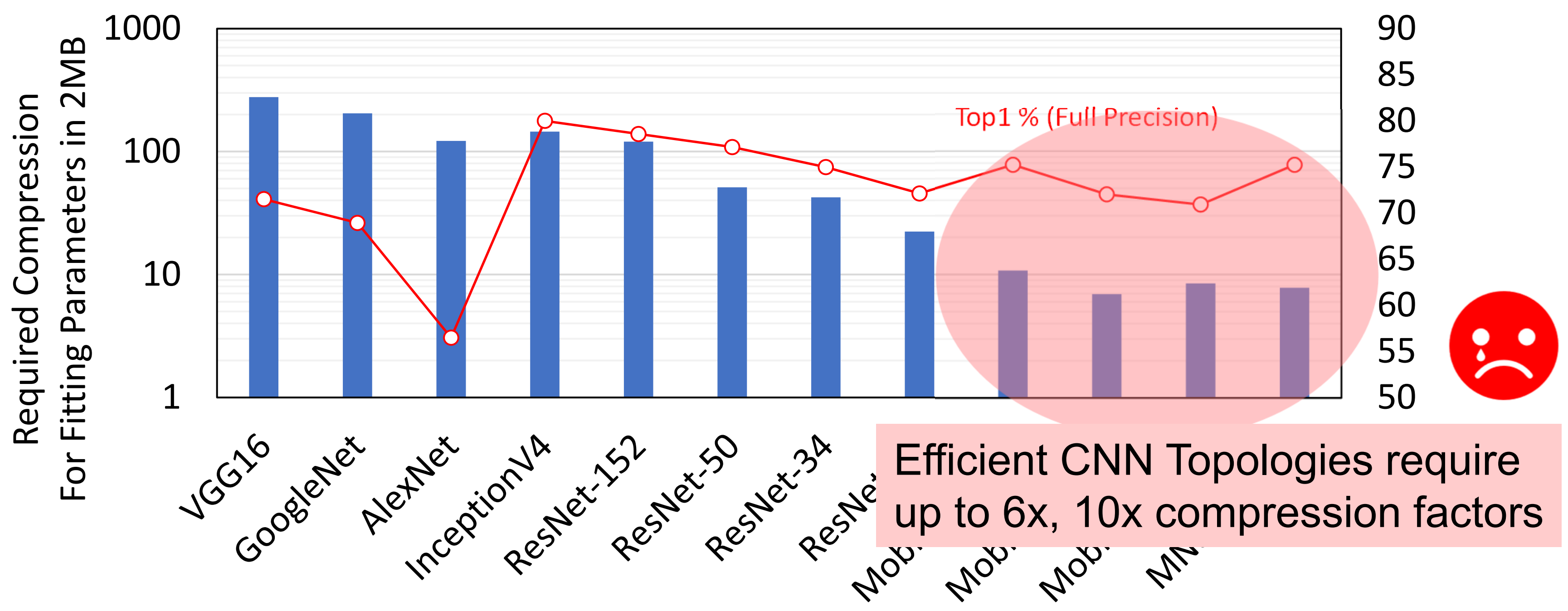


Source: STM32H7 datasheet

- ❑ Low-power (<10-100mW) & low-cost
  - ❑ Smart device are battery- operated
- ❑ Highly-flexible (SW programmable)
- ❑ But **limited resources(!)**
  - ❑ few MB of memories
  - ❑ single RISC core up to few 100s MHZ (STM32H7: 480MHz) with DSP SIMD instructions and optional FPU
- ❑ Currently, only tinyML tasks on MCUs (visual wake words, CIFAR10)

Challenge: Run ‘complex’ and ‘big’ (Imagenet-size) DL inference on MCU ?

Deep Learning Model Size



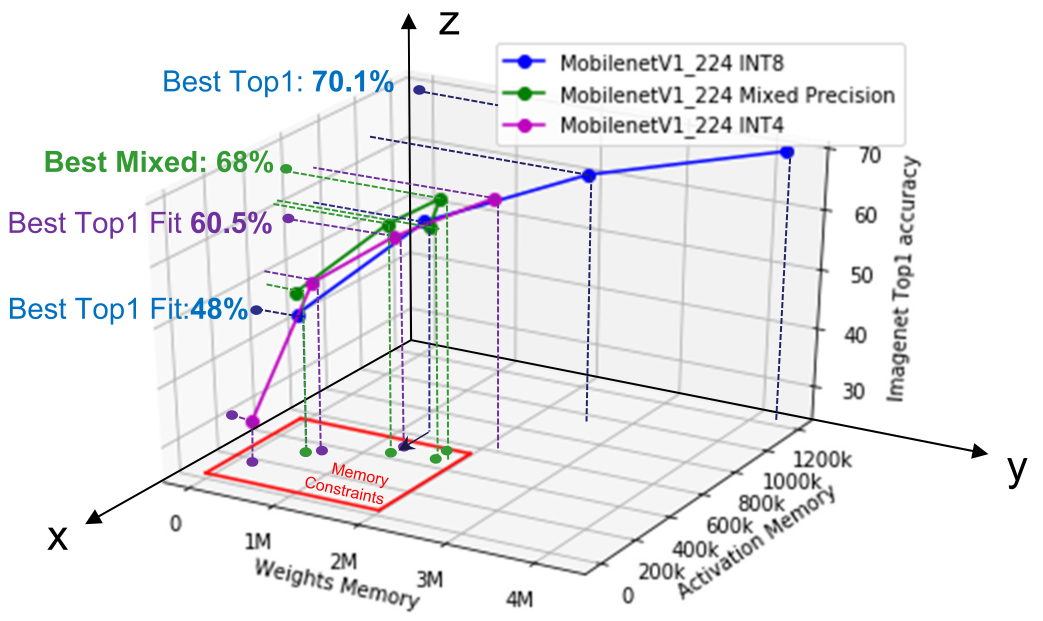
Efficient CNN Topologies require up to 6x, 10x compression factors

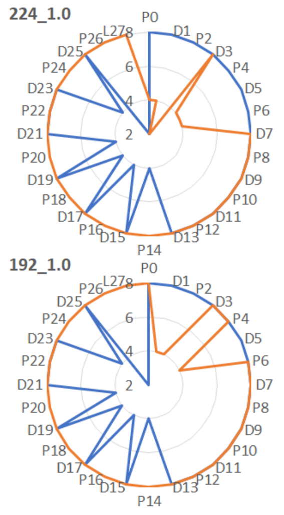
Rusci, Manuele, et al. "Leveraging Automated Mixed-Low-Precision Quantization for tiny edge microcontrollers." arXiv preprint arXiv:2008.05124 (2020).

Fact 1: Integer-only model needed for deployment on low-power MCUs
Fact 2: 8-16 bit are **not** sufficient to bring 'complex' models on MCUs (memory!!)

Mixed-Precision Quantization

Using less than 8 bits…





HAQ (Wang et al 2018)	Accuracy-aware mixed-precision policy search with RL agent. Relies on non-uniform quantization, weight-only.
HAWQ (Dong et al 2019)	Hessian-based metric to compute the quantization policy. Less performant than HAQ.
(Rusci et al 2020)	Mixed Precision Quantizatoin appling minimum tensor-wise quantization ≤8bit to fit the **memory constraints**

(Wang et al 2018) Wang, Kuan, et al. "Haq: Hardware-aware automated quantization with mixed precision." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019.
(Dong et al 2019) Dong, Zhen, et al. "Hawq: Hessian aware quantization of neural networks with mixed-precision." *Proceedings of the IEEE International Conference on Computer Vision*. 2019.
(Rusci et al. 2020) "Memory-driven mixed low precision quantization for enabling deep network inference on microcontrollers." *arXiv preprint arXiv:1905.13082* (2019).

Fact 3: Mixed Low-Precision Improves Accuracy on memory constrained device

CMix-NN: Mixed Low-Precision CNN Library

- ✓ **Mixed-precision sub-byte** (2, 4, 8 bit) quantization for weights and activations
- ✓ **Per-Layer** *(PL),* and **Per-Channel** *(PC)* Quantization
- ✓ **Optimized** for ARM ARMv7-M ISA
- ✓ **Open-Source:** https://github.com/EEESlab/CMix-NN

CMix-NN Source Code Structure

- Based on CMSIS-NN
  - Same computational model (im2col + GEMM)
  - Same HWC tensor layout
- C Library (not C++)
  - NN Graph is compiled! Good for Latency!
- Convolutional Layer Supported
  - 2D Conv, Depthwise
  - FC

More that 81 different convolutional kernel configurations!

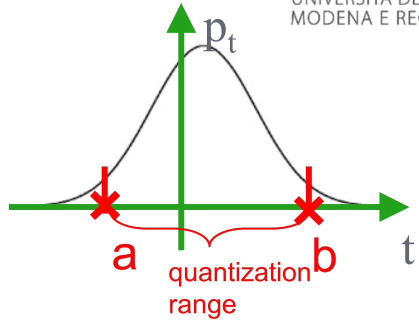
Cmix-NN final source code is generated using templates (mako, python)

Uniform Interger Quantization Flavors

real value tensor or sub-tensor → quantized tensor (INT-Q)

$$t = S_t \times (T_q - Z_t)$$

$$S_t = \frac{b-a}{2^Q - 1}$$



$$T_q = Z_t + round\left(\frac{clamp(t,a,b)}{S_t}\right)$$

a, b are learned or deterministic params
In general *a* != *b*, **asymmetric quantization**

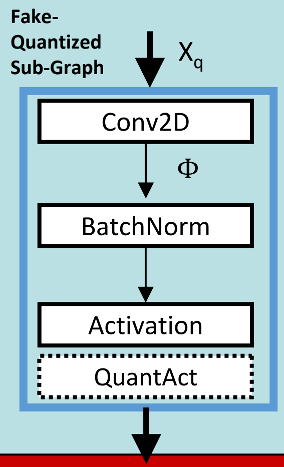
weights

Per-Layer (PL)	: asym *[a,b]* of the whole weight tensor *w*
Per-Channel (PC)	: asym *[a,b]* of the weight sub-tensor along the outer dim *w[i]*

activations

Typically asymmetric by definition (e.g. ReLU), b is learned via PACT

$$Y_q = quant_{act}(x) = Z_y + floor\left(\frac{clamp(x,0,b)}{S_x}\right) \cdot S_x$$



To turn a fake-quantized (sub-byte) sub-graph into an integer-only Representation we use **Integer Channel-Normalization** (Rusci et al. 2019) :

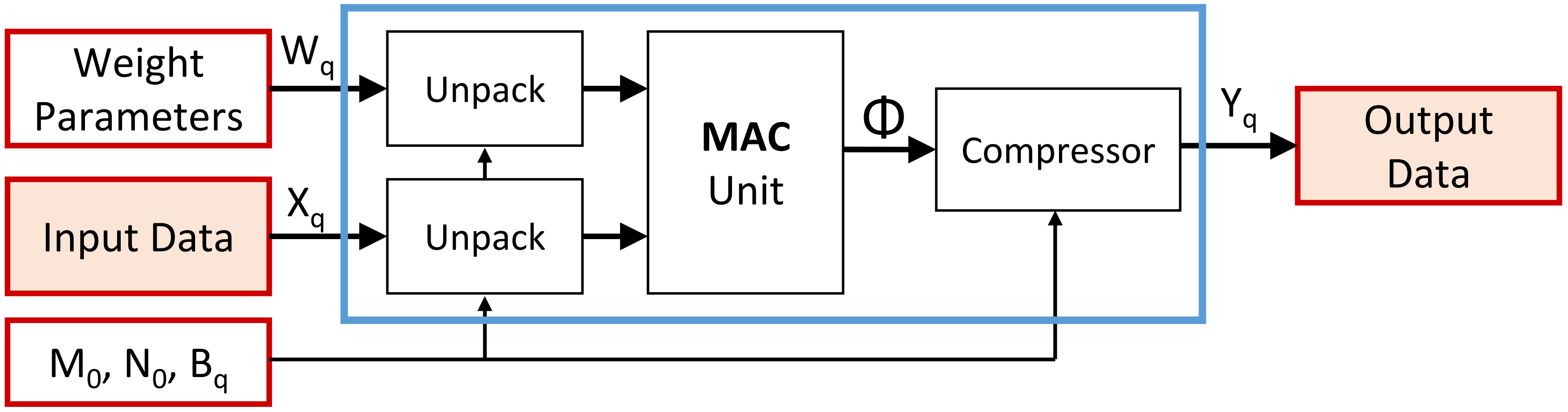
$$Y_q = Z_y + clamp\left(floor\left(M_0 2^{N_0}\left(\Phi + B_q\right)\right), 0, 2^{Q-1}\right)$$

M_0, N_0, B_q are channel-wise integer params

(Rusci et al. 2019) "Memory-driven mixed low precision quantization for enabling deep network inference on microcontrollers." arXiv preprint arXiv:1905.13082
(Jacob, 2018) Jacob, Benoit, et al. "Quantization and training of neural networks for efficient integer-arithmetic-only inference." *CVPR* 2018

CMix-NN Quantized Convolutional Layer

Layer-by-layer execution



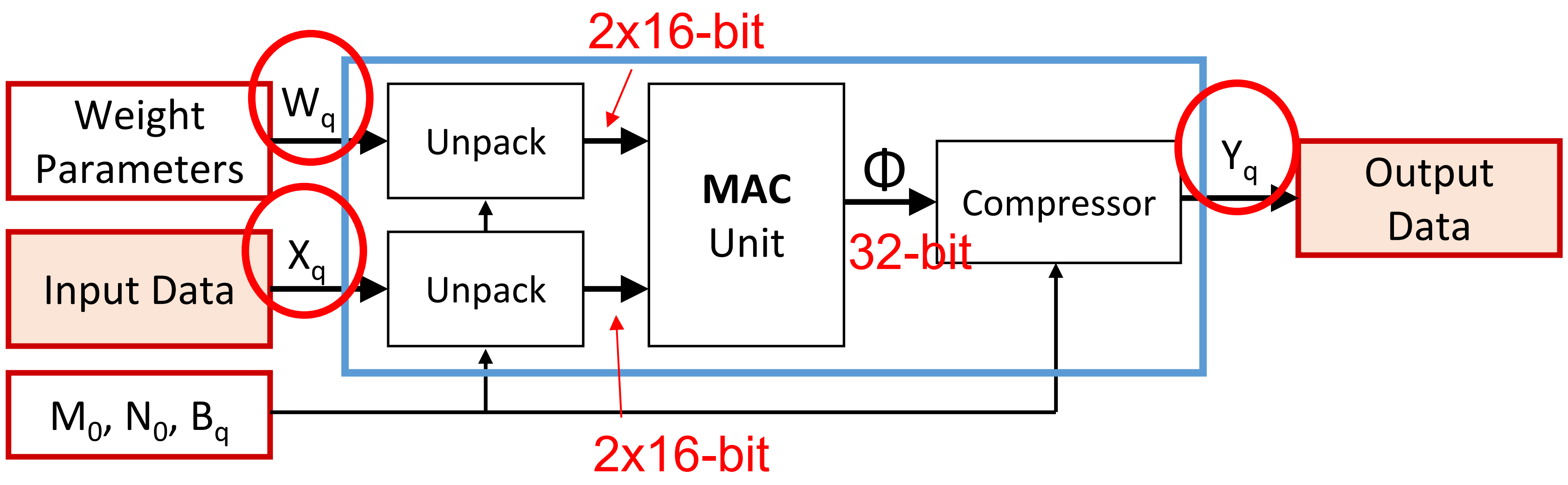
Layer transfer function

$$Y_q = Z_y + clamp\left(floor\left(M_0 2^{N_0}\left(\Phi + B_q\right)\right), 0, 2^{Q-1}\right)$$

$$\Phi = \sum(W_q - Z_w) \cdot (X_q - Z_x)$$

NB. This formulation holds either for PL or PC quantization of weights, only Z_w differs from scaler to array

CMix-NN Quantized Convolutional Layer (2)



- W_q, X_q and Y_q has *arbitrary* precision (2-, 4-, 8-bit)
- MAC Unit exploits *2x16-bit* SIMD MAC (ARMv7-M ISA)
- Compession (or *Activation Function*) brings 32-bit accumulator to *arbitrary* precision (2-, 4-, 8-bit)

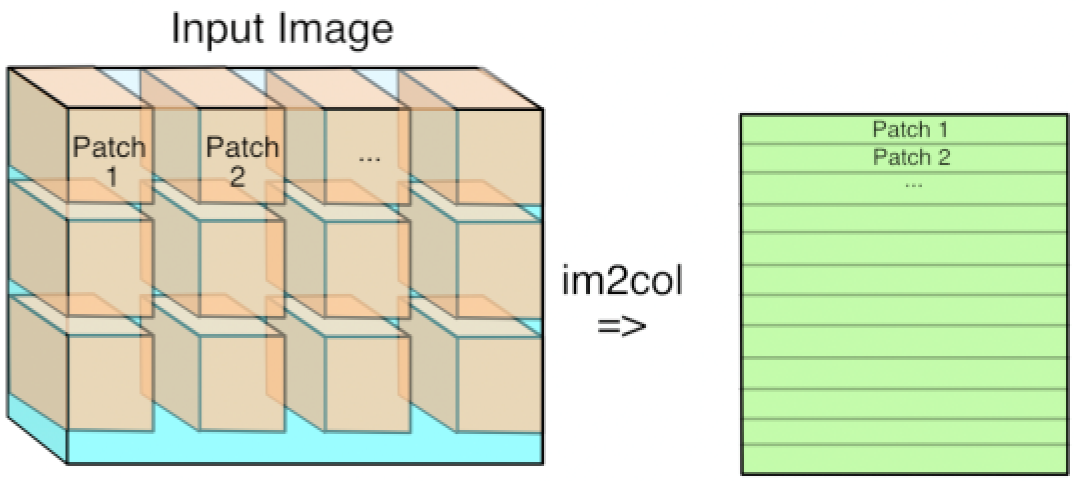
CMix-NN Computational Scheme

```
for $x,y=0; x,y<dimOut; x,y{+}{+}$ do



end for
```

CMix-NN Computational Scheme

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

```
for x,y=0; x,y<dimOut; x,y++ do
im2col {
    u32 ptrBuff0 ← im2colQ(tensorIn[x,y], Zin, kw, kh, InFeat, nCol)
    u32 ptrBuff1 ← im2colQ(tensorIn[x+1,y], Zin, kw, kh, InFeat, nCol)
```



NOTE! Patches are also packed from original bitwidth to 2x16-bit Vectors

```
end for
```

Figure: https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

CMix-NN Computational Scheme

```
im2col {  for x,y=0; x,y<dimOut; x,y++ do
              u32 ptrBuff0 ← im2colQ(tensorIn[x,y], Zin, kw, kh, InFeat, nCol)
              u32 ptrBuff1 ← im2colQ(tensorIn[x+1,y], Zin, kw, kh, InFeat, nCol)
MatMul {      for each co in OutFeat; co+=2 do

              end for
          end for
```

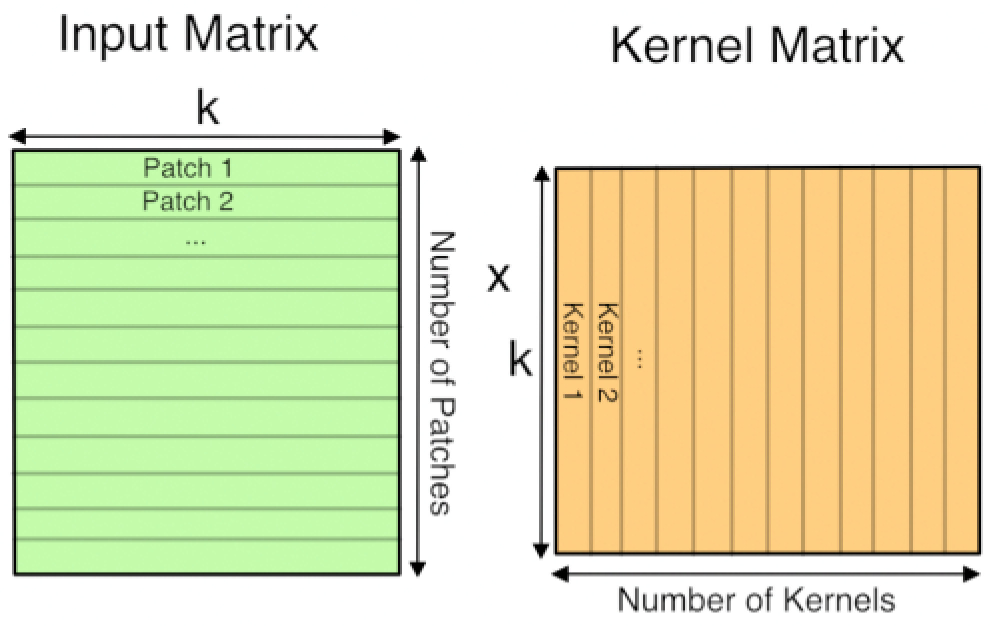
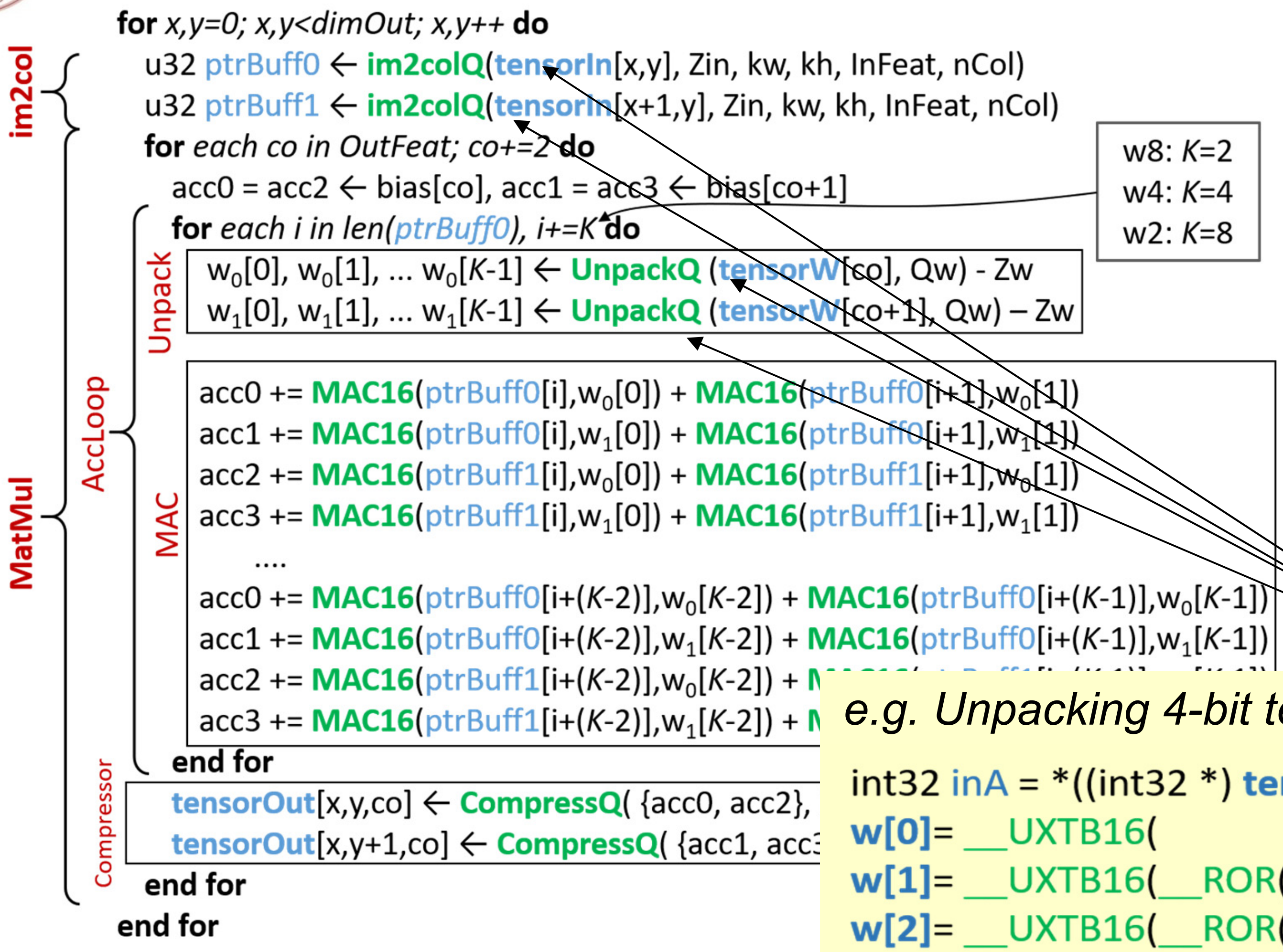


Figure: https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

CMix-NN Computational Scheme

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA


im2col
MatMul
AccLoop
Unpack
MAC
Compressor

```
for x,y=0; x,y<dimOut; x,y++ do
  u32 ptrBuff0 ← im2colQ(tensorIn[x,y], Zin, kw, kh, InFeat, nCol)
  u32 ptrBuff1 ← im2colQ(tensorIn[x+1,y], Zin, kw, kh, InFeat, nCol)
  for each co in OutFeat; co+=2 do
    acc0 = acc2 ← bias[co], acc1 = acc3 ← bias[co+1]
    for each i in len(ptrBuff0), i+=K do
      $w_0$[0], $w_0$[1], ... $w_0$[K-1] ← UnpackQ (tensorW[co], Qw) - Zw
      $w_1$[0], $w_1$[1], ... $w_1$[K-1] ← UnpackQ (tensorW[co+1], Qw) – Zw

      acc0 += MAC16(ptrBuff0[i],$w_0$[0]) + MAC16(ptrBuff0[i+1],$w_0$[1])
      acc1 += MAC16(ptrBuff0[i],$w_1$[0]) + MAC16(ptrBuff0[i+1],$w_1$[1])
      acc2 += MAC16(ptrBuff1[i],$w_0$[0]) + MAC16(ptrBuff1[i+1],$w_0$[1])
      acc3 += MAC16(ptrBuff1[i],$w_1$[0]) + MAC16(ptrBuff1[i+1],$w_1$[1])
          ....
      acc0 += MAC16(ptrBuff0[i+(K-2)],$w_0$[K-2]) + MAC16(ptrBuff0[i+(K-1)],$w_0$[K-1])
      acc1 += MAC16(ptrBuff0[i+(K-2)],$w_1$[K-2]) + MAC16(ptrBuff0[i+(K-1)],$w_1$[K-1])
      acc2 += MAC16(ptrBuff1[i+(K-2)],$w_0$[K-2]) + M
      acc3 += MAC16(ptrBuff1[i+(K-2)],$w_1$[K-2]) + M
    end for
    tensorOut[x,y,co] ← CompressQ( {acc0, acc2},
    tensorOut[x,y+1,co] ← CompressQ( {acc1, acc
  end for
end for
```

w8: *K*=2
w4: *K*=4
w2: *K*=8

e.g. Unpacking 4-bit tensors.

```
int32 inA = *((int32 *) tensorW);
w[0]= __UXTB16(          inA        & 0x000F000F);
w[1]= __UXTB16(__ROR(inA, 4)  & 0x000F000F);
w[2]= __UXTB16(__ROR(inA, 8)  & 0x000F000F);
w[3]= __UXTB16(__ROR(inA, 12) & 0x000F000F);
```

Low-bitdwith Extraction is not a free meal...

Experimental Setup

- **Low-bitwidth Performance Characterization**
- **End2End Network Inference SoA Comparison**

Target Board

- STM32H7
  - ARM Cortex-M7 480 MHz, 2MB FLASH, 512kB SRAM

CNN Network Configuration

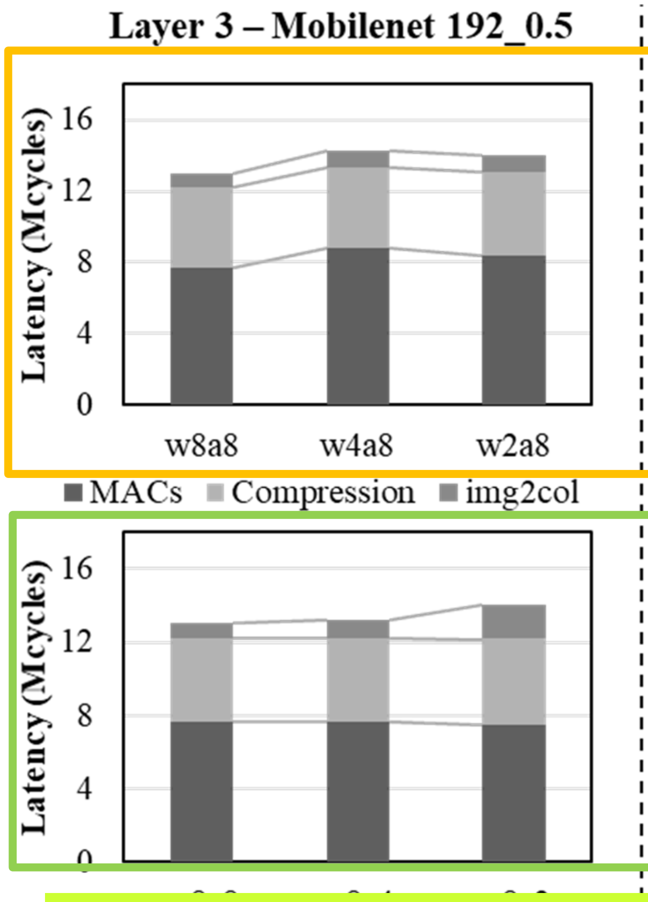
- Mobilenet v1 and Mobilenet v2 Family
- Imagenet (1000-class) Classification Task

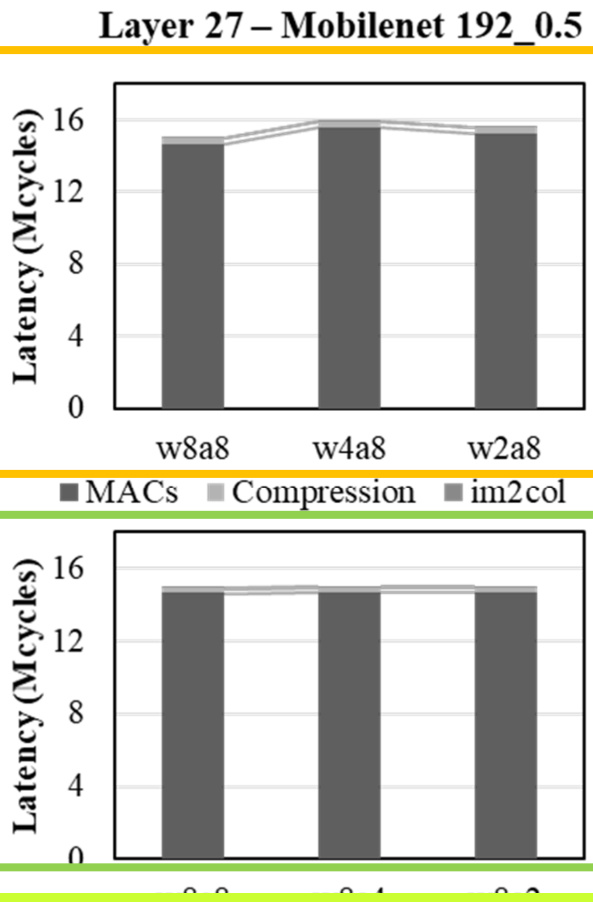
Inference Libraries SOA

- CMSIS-NN
- STM CubeMX.AI

Low-bitwidth Performance Characterization

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA





Compression and **im2col** are insensitive

MAC loop up to 14% slower (due to unpack)

Varying W bitwidth

Compression and **MAC** are insensitive

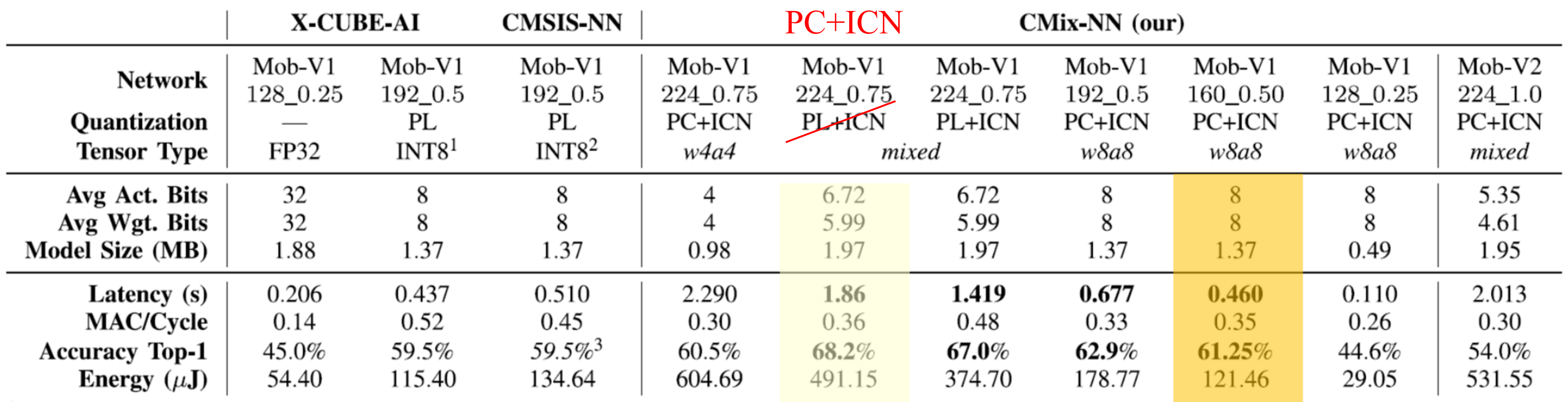
im2col loop up to 10% slower

Varying Act bitwidth

↗ Input dimension = ↘ load/MAC (for this benchmark)
↘ W bits = ↗ unpacking instructions
The spatial stationary computation model makes convolution efficiency independent by activation bit

End2End SoA Comparison

Errata Corrige



	X-CUBE-AI		CMSIS-NN	CMix-NN (our)	PC+ICN					
Network	Mob-V1 128_0.25	Mob-V1 192_0.5	Mob-V1 192_0.5	Mob-V1 224_0.75	Mob-V1 224_0.75	Mob-V1 224_0.75	Mob-V1 192_0.5	Mob-V1 160_0.50	Mob-V1 128_0.25	Mob-V2 224_1.0
Quantization	—	PL	PL	PC+ICN	~~PL+ICN~~	PL+ICN	PC+ICN	PC+ICN	PC+ICN	PC+ICN
Tensor Type	FP32	INT8[1]	INT8[2]	*w4a4*	*mixed*		*w8a8*	*w8a8*	*w8a8*	*mixed*
Avg Act. Bits	32	8	8	4	6.72	6.72	8	8	8	5.35
Avg Wgt. Bits	32	8	8	4	5.99	5.99	8	8	8	4.61
Model Size (MB)	1.88	1.37	1.37	0.98	1.97	1.97	1.37	1.37	0.49	1.95
Latency (s)	0.206	0.437	0.510	2.290	**1.86**	**1.419**	**0.677**	**0.460**	0.110	2.013
MAC/Cycle	0.14	0.52	0.45	0.30	0.36	0.48	0.33	0.35	0.26	0.30
Accuracy Top-1	45.0%	59.5%	*59.5%*[3]	60.5%	**68.2%**	**67.0%**	**62.9%**	**61.25%**	44.6%	54.0%
Energy (μJ)	54.40	115.40	134.64	604.69	491.15	374.70	178.77	121.46	29.05	531.55

Accuracy

Best Cmix-NN solution (MobV1-224-0.75, PC+ICN, Mixed, Top1% 68.2%) up to

- **+23% Accuracy** compared to the best FP32 solution (CubeMX.AI)
- **+9% Accuracy** compared to the best INT8 solution (CubeMX.AI, CMISIS-NN)

Per-Channel quantization is always more accurate compare to Per-Layer (up to **+1.75%**)

Efficiency

Per-Channel quatization is approx. 24% less efficient in MAC/s (CubeMX.AI 0.52 MAC/s, Cmix-NN PL+ICN 0.48, Cmix-NN PC+ICN 0.36)

But, **MobV1-160-0.50**, PC+ICN, has the same or lower latency of any other competitors and provide a **more accurate solution!**

Conclusions

- CMix-NN, flexible inference library for MCU-based edge devices
  - optimized backend to deploy state-of-the-art mixed low-precision deep neural networks for ARM Cortex-M MCUs
  - Support different flavor of quantization
    - Different Datatypes, for weights and activations
    - Per-Channel and Per-Layer quantization

- Open-Source https://github.com/EEESlab/CMix-NN

- CMix-NN enables the deployment of a 68% Imagenet MobilenetV1 into a MCU with 2MB FLASH and 512 kB RAM.

What’s next

Supporting Complex ML problems on MCU-class require

- Optmized byte and sub-byte operations on ISA!
  - A. Garofalo “XpulpNN: Accelerating Quantized Neural Networks On RISC-V Processors Through ISA Extensions”
  - ARMv8-M (MAC 8/16-bit SIMD arithmetic)

- Novel Quantization Mechanisms to improve accuracy degradation on low-precision (sub-byte) models

- Quantization is only one of the levels for DL Compression
  - Pruning
  - NAS and AutoML
  - How integrate them properly!

Check it out and Contribute to Cmix-NN!

https://github.com/EEESlab/CMix-NN

Alessandro Capotondi
Università di Modena e Reggio Emilia
alessandro.capotondi@unimore.it