Memory-driven mixed low precision quantization for enabling deep inference networks on microcontrollers

Manuele Rusci*, Alessandro Capotondi, Luca Benini
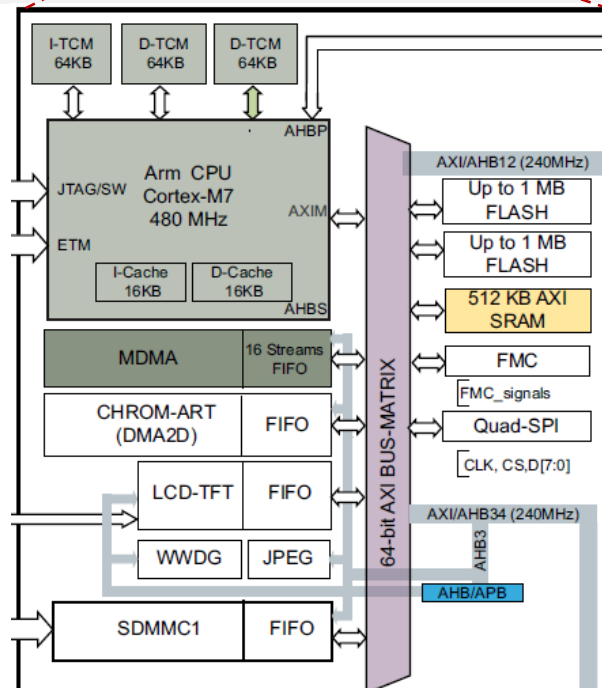
manuele.rusci@unibo.it

Energy-Efficient Embedded Systems Laboratory

Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione "Guglielmo Marconi" – DEI – **Università di Bologna**

Microcontrollers for smart sensors
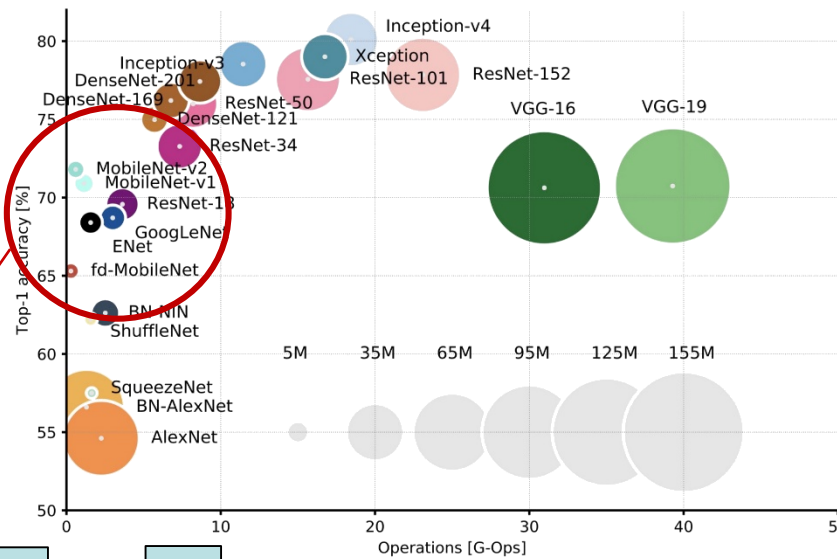
Microcontrollers for smart sensors

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: 400MHz) with DSP SIMD instructions and optional FPU

- ❑ Currently, tiny visual DL tasks on MCUs (visual wake words, CIFAR10)

Challenge: Run 'complex' and 'big' (Imagenet-size) DL inference on MCU ?
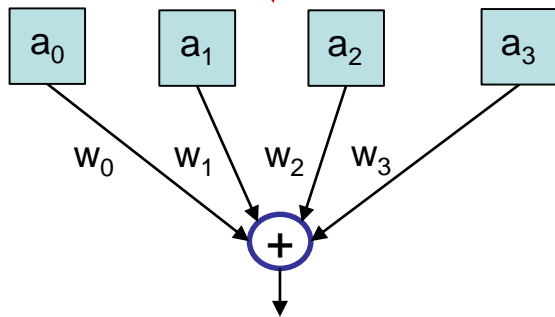
Deep Learning for microcontrollers

"Efficient" topologies: Accuracy **vs MAC vs Memory**

But **quantization** is also essential…

Source: https://towardsdatascience.com/neural-network-architectures-156e5bad51ba

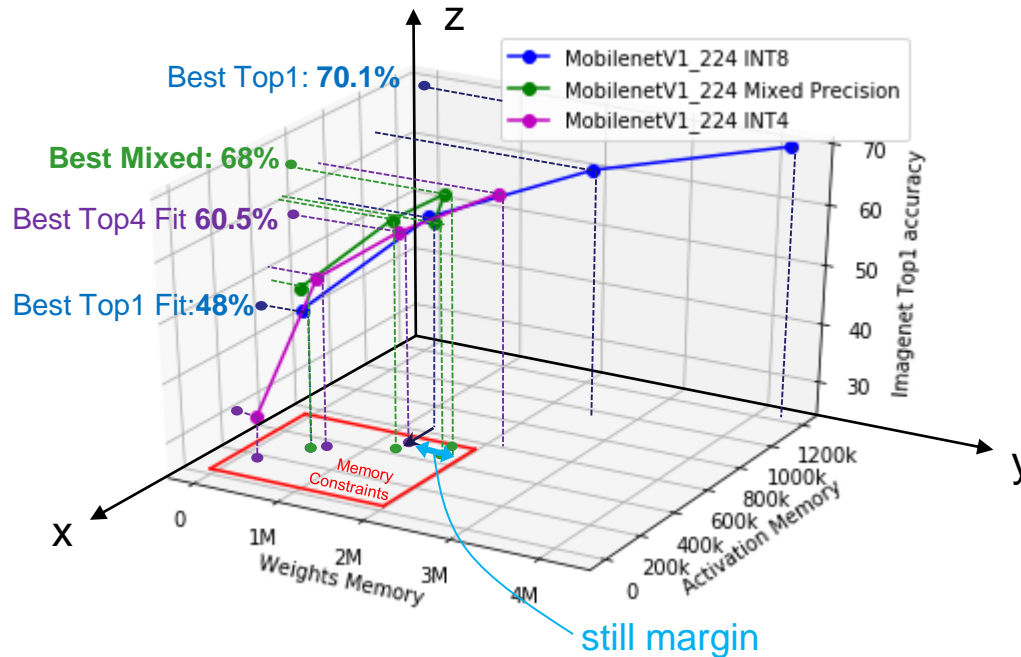| | FP32 | : 4 instr + 32 bytes | Memory | Accuracy | Compute |
| INT16 | : 2 instr + 16 bytes |
| INT8 | : 1 instr + 8 bytes |

Reducing bit precision

(if ISA MAC SIMD available)

Issue1: Integer-only model needed for deployment on low-power MCUs

Issue2: 8-16 bit are **not** sufficient to bring 'complex' models on MCUs (memory!!)

Memory-Driven Mixed-Precision Quantization

Using less than 8 bits…

apply minimum tensor-wise quantization ≤8bit to fit the **memory constraints** with very-low accuracy drop
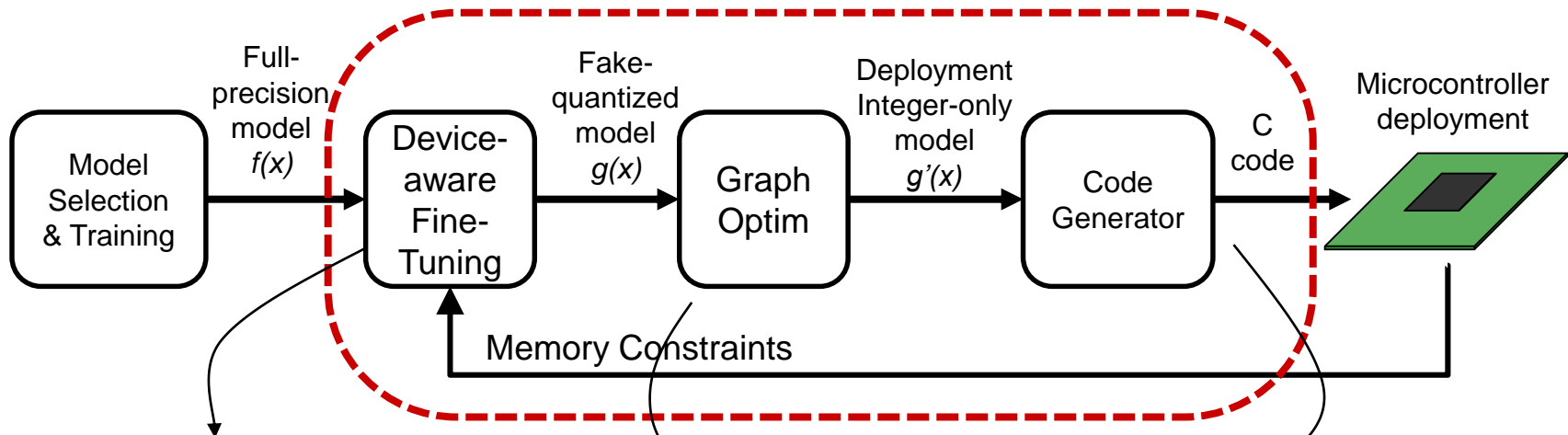
➢ **Challenges**:
  – How to define the quantization policy
  – Combine quantization flow this with integer only transformation

End-to-end Flow & Contributions

Goal: Define a design flow to bring Imagenet-size models into an MCU device while paying a low accuracy drop.

DNN Development Flow for microcontrollers

Device-aware Fine-Tuning
We define a rule-based **methodology** to determine the **mixed-precision quantization policy** driven by a memory objective function.

Deployment on MCU
A **latency-accuracy tradeoff** on iso-memory mixed-precision networks belonging to the **Imagenet MobilenetV1** family when running on a **STM32H7** MCU.
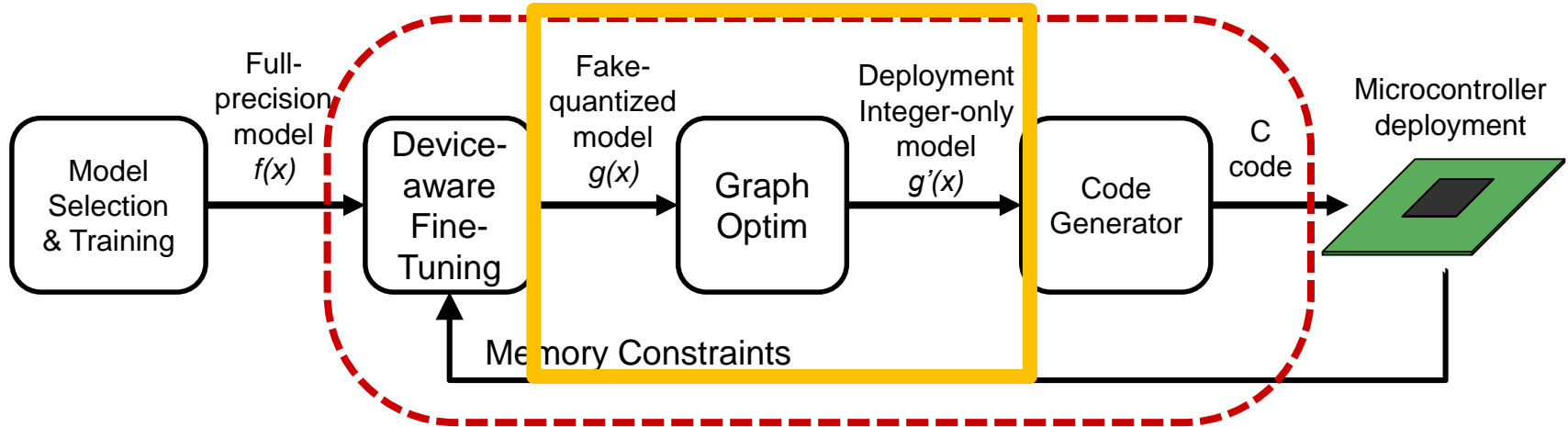
Graph Optimization
We introduce the **Integer Channel-Normalization** (ICN) activation layer to generate an **integer-only deployment** graph when applying **uniform sub-byte quantization**.

DNN Development Flow for microcontrollers

Graph Optimization

INTEGER-ONLY W/ SUB-BYTE QUANTIZATION

State of the Art

- ❑ Inference with Integer-only arithmetic (Jacob, 2018)
  - ❑ Affine transformation between real value and (<u>uniform</u>) quantized parameters
  - ❑ Quantization-aware retraining
  - ❑ Folding of batch norm into conv weights + rounding of per-layer scaling parameters

real value tensor or sub-tensor ⟶

quantized tensor (INT-Q)

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

- ☺ Almost lossless with 8 bit on Image classification and detection problems. Used by TF Lite.

- ☹ 4 bit MobilnetV1: Training collapse when folding batch norm into convolution weights

- ☹ Does not support Per-Channel (PC) weight quantization

Integer-Only MobilenetV1_224_1.0

Quantization Method	Top1	Weights (MB)
Full-Precision	70.9	16.8
w8a8	70.1	4.06
w4a4	0.1	2.05

(Jacob, 2018) Jacob, Benoit, et al. "Quantization and training of neural networks for efficient integer-arithmetic-only inference." *CVPR* 2018

Integer-Channel Normalization (ICN)

Fake-Quantized Sub-Graph

$\downarrow$ X$_q$

```
Conv2D
  │ Φ
  ▼
BatchNorm
  │
  ▼
Activation
  ┊
QuantAct
```

$\downarrow$ Y$_q$
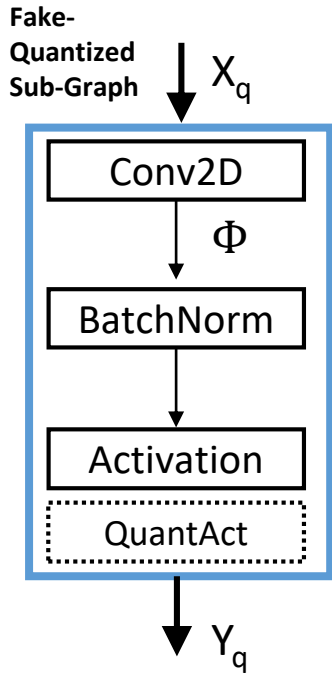
$$Y_q = quant_{act}\left(\frac{\phi - \mu}{\sigma} \cdot \gamma + \beta\right)$$

$$\phi = \sum w \cdot x$$

$\mu, \sigma, \gamma, \beta$ are channel-wise batchnorm parameters

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

S_w is scalar if PL, else array
S_i, S_o are scalar

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

$$Y_q = Z_y + quant_{act}\left(\frac{S_i S_w}{S_o} \frac{\gamma}{\sigma}\left(\Phi + \left[\frac{1}{S_i S_w}\left(B - \mu + \frac{\beta \, \sigma}{\gamma}\right)\right]\right)\right)$$

$$M_0 2^{N_0}\left(\Phi + B_q\right)$$

M_0, N_0, B_q are channel-wise integer params
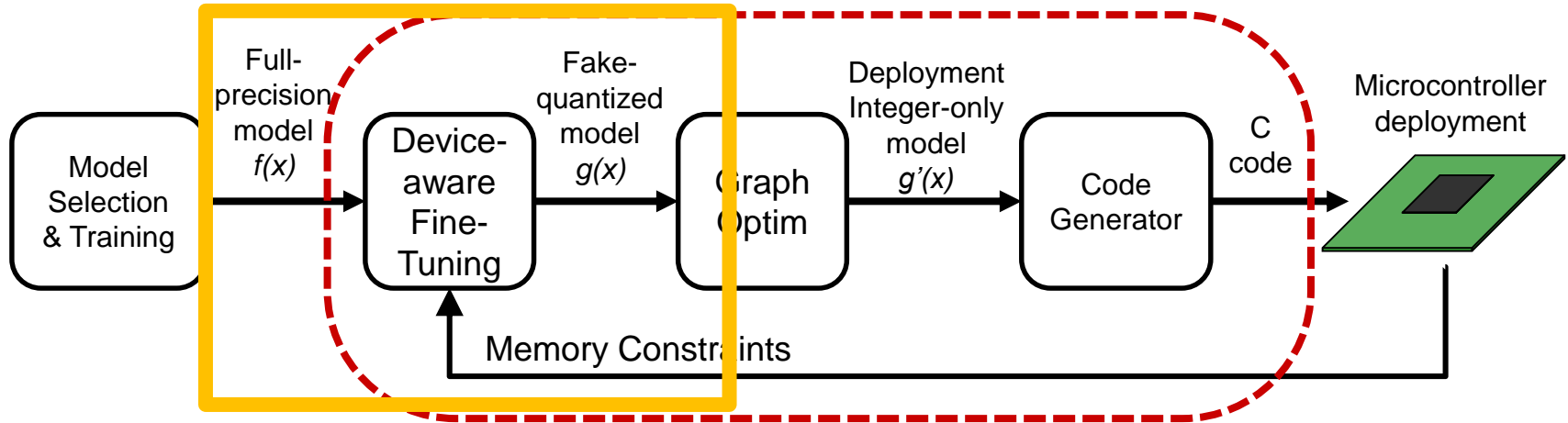
Integer-Only MobilenetV1_224_1.0

Quantization Method	Top1	Weights (MB)
Full-Precision	70.9	16.8
PL+ICN w4a4	61.75	2.10
PC+ICN w4a4	66.41	2.12

Integer Channel-Normalization (ICN) activation function
➤ holds either for PL or PC quantization of weights
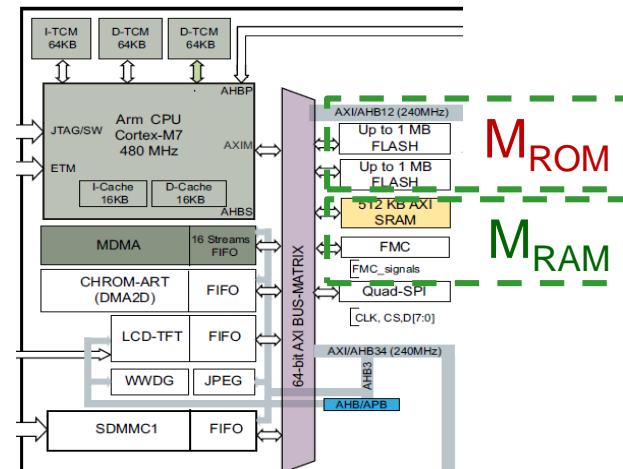
Device-aware Fine-Tuning

MIXED-PRECISION QUANTIZATION POLICY

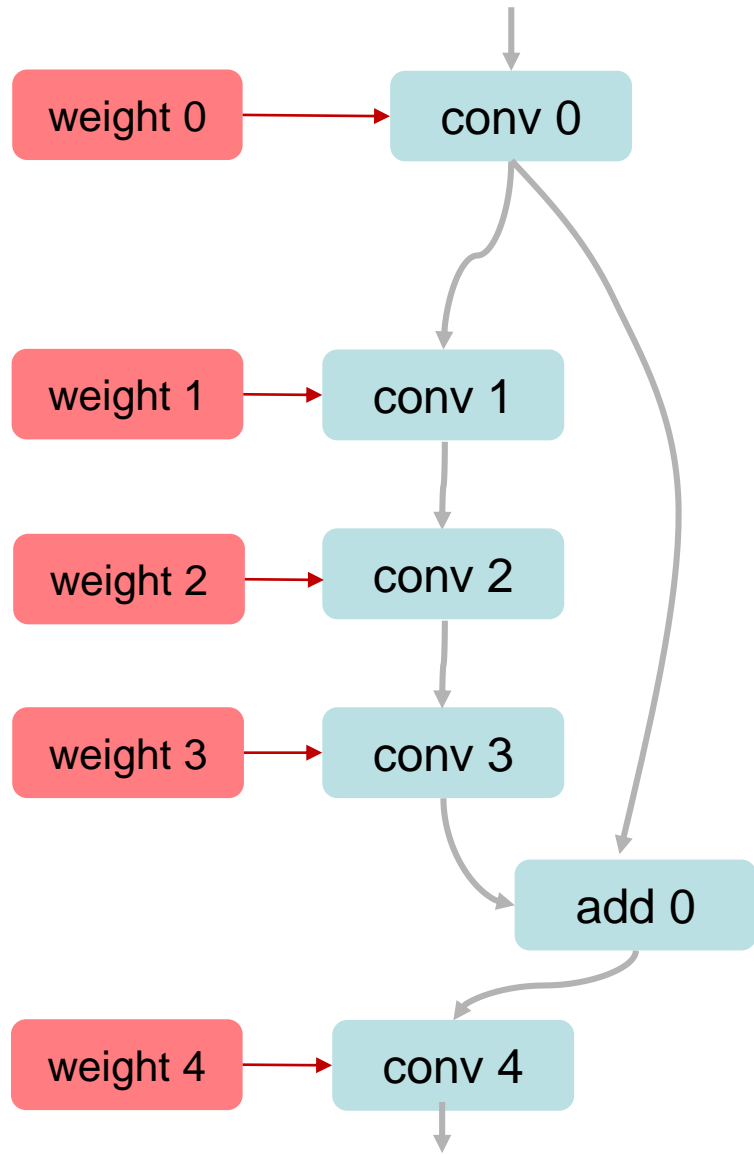# Deployment of an integer-only graph
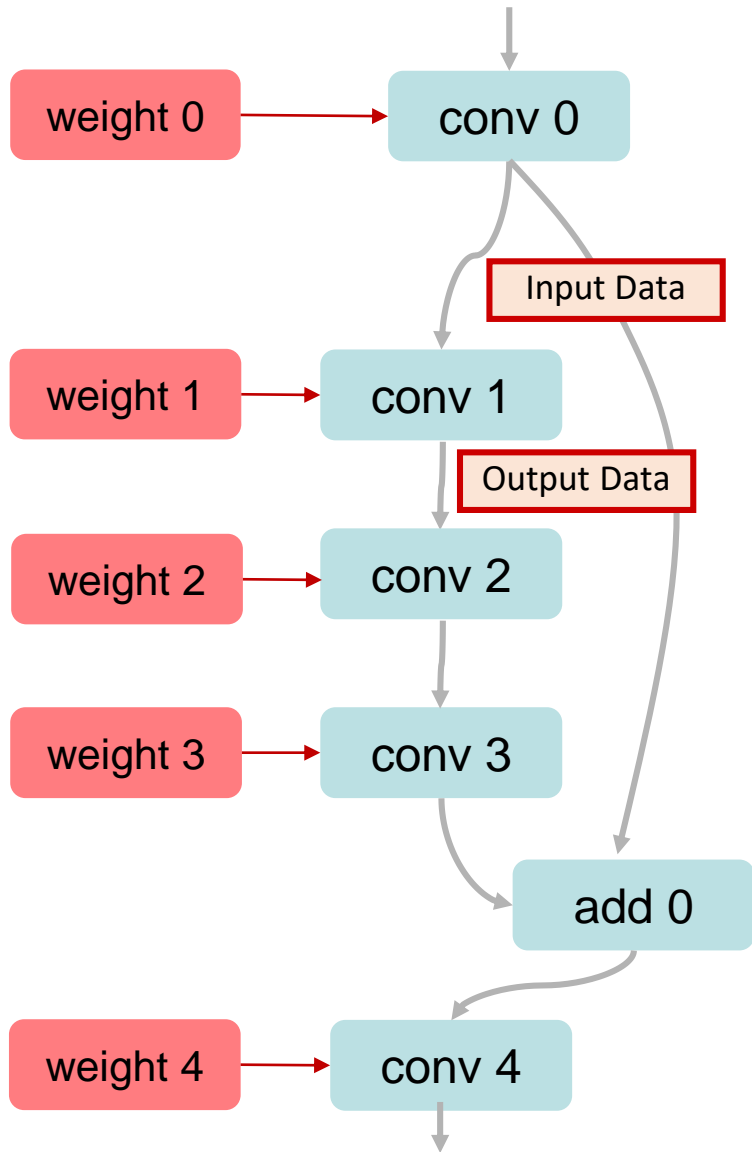
Problem

Can this graph fit the memory constraints of our MCU device?

Deployment of an integer-only graph

Problem

Can this graph fit the memory constraints of our MCU device?

Deployment of an integer-only graph

[M1]
$$\sum_{i=0}^{L-1} mem(W_i, Q_w^i) + mem(M_0, N0, B_q) < M_{ROM}$$

[M2]
$$mem(X_i, Q_x^i) + mem(Y_i, Q_y^i) < M_{RAM}, \qquad \forall i$$

[M1] weights must fit M_{ROM}

[M2] I/O of a node must fit M_{RAM}

weight 0 → conv 0

weight 1 → conv 1

weight 2 → conv 2

weight 3 → conv 3

add 0

weight 4 → conv 4

Problem Formulation
Find the quantization policy Q_x^i, Q_y^i, Q_w^i
to satisfy [M1] and [M2]

$Q_x^i, Q_y^i, Q_w^i \in \{2,4,8\}$ bits

Rule-Based Mixed-Precision

Set $Q_w^i = 8$

[M1] satisfied ? — yes

no

Compute mem occupation
$$r_i = \frac{mem(w_i, Q_w^i)}{tot_{MEM}}$$
$$R = \max r_i$$

Cut Q_w^i of the lower layer with a mem occupation $r_i > R - \delta$

Weights Quantization Policy

[M1] : size(w0) + size(w1) + size (w2) + size(w3) < M_{ROM}

$\delta = 5\%$

$Q_w^0 = 8$
13%

w0 → conv 0

$Q_w^1 = 8$
15%

w1 → conv 1

$Q_w^2 = 8$
22%

w2 → conv 2

$Q_w^3 = 8$
50%

w3 → conv 3

Rule-Based Mixed-Precision

Goal
Maximize memory utilization

Any cut reduces the bit precision by one step: 8→4, 4→2

Set $Q_w^i = 8$

[M1] satisfied ? — **yes**

no

Compute mem occupation
$$r_i = \frac{mem(w_i, Q_w^i)}{tot_{MEM}}$$
$$R = \max r_i$$

Cut Q_w^i of the lower layer with a mem occupation $r_i > R - \delta$
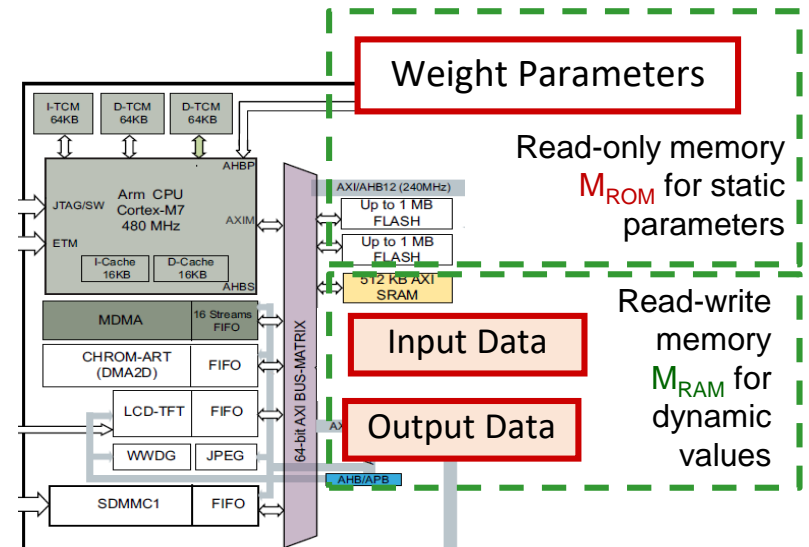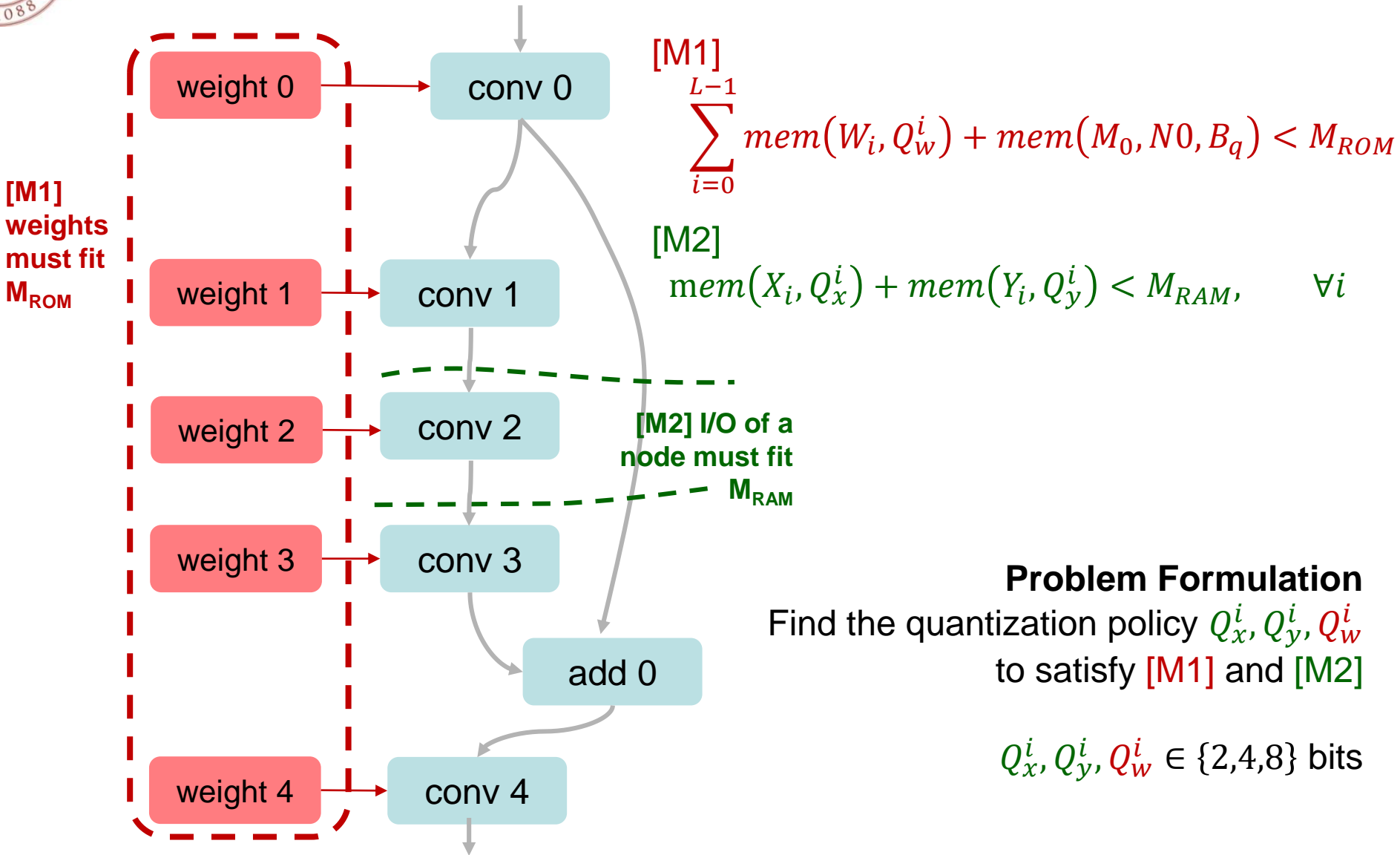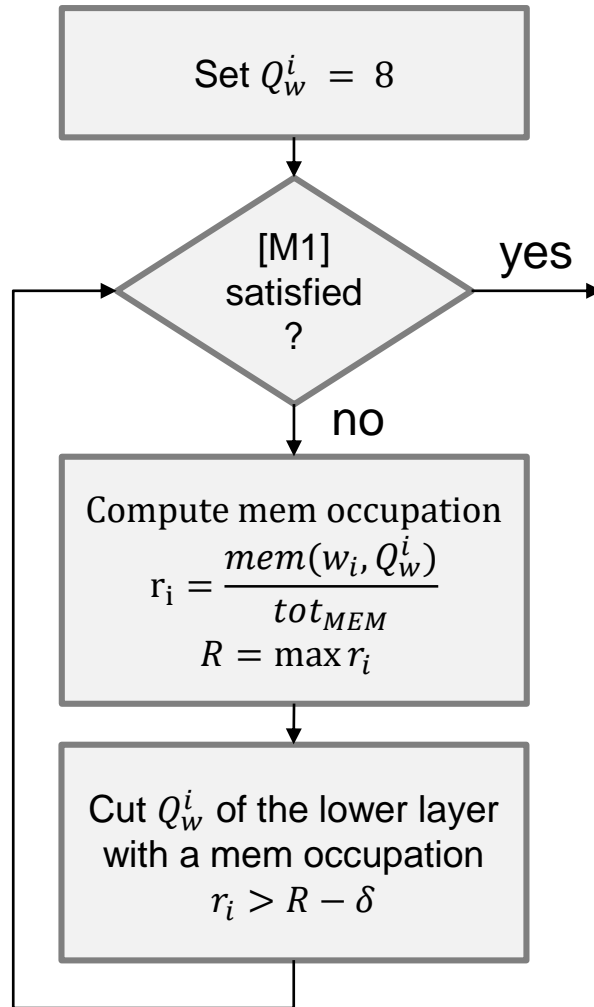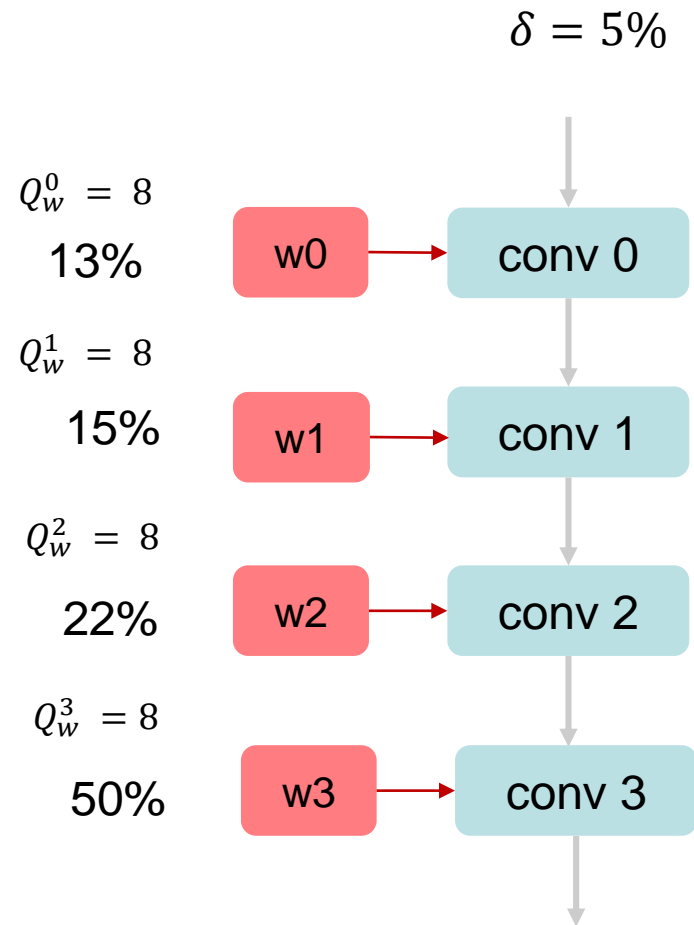
Weights Quantization Policy
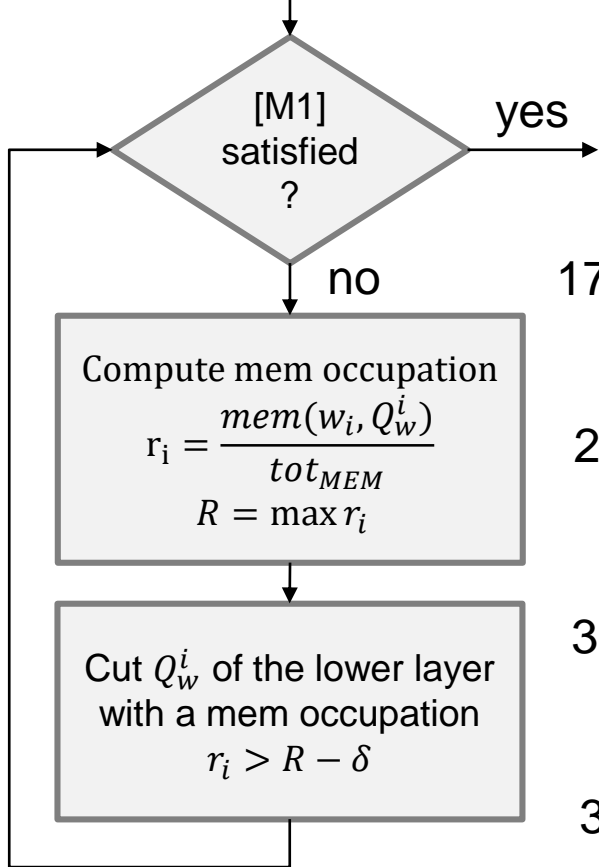
[M1] : size(w0) + size(w1) + size (w2) + size(w3) < M_{ROM}

$\delta = 5\%$

$Q_w^0 = 8$
17% ← 13%

$Q_w^1 = 8$
20% ← 15%

$Q_w^2 = 8$
30% ← 22%

$Q_w^3 = 4$
33% ← 50%
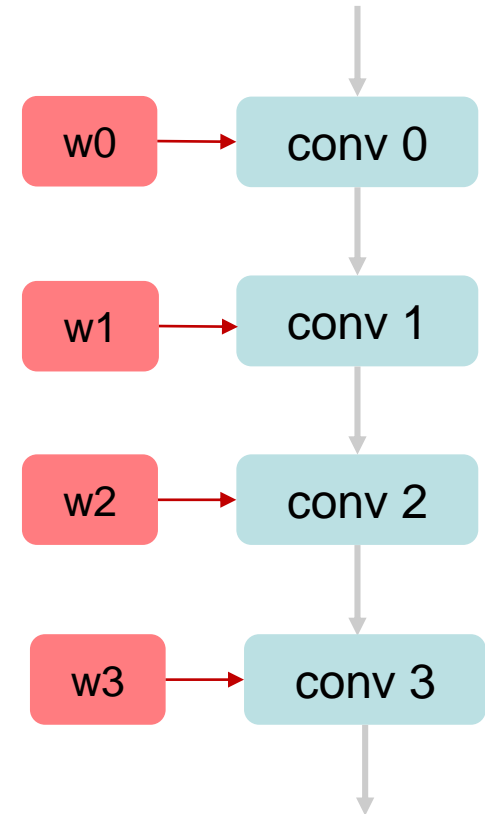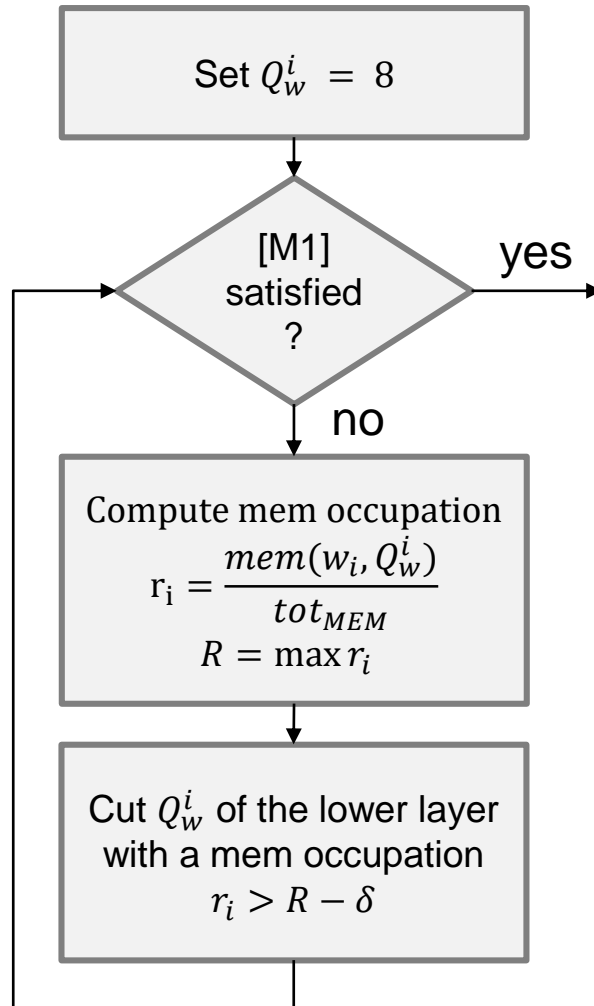
w0 → conv 0

w1 → conv 1

w2 → conv 2

w3 → conv 3

Cut layer 3!

Rule-Based Mixed-Precision

Goal
Maximize memory utilization

Any cut reduces the bit precision by one step: $8 \rightarrow 4$, $4 \rightarrow 2$

Set $Q_w^i = 8$

$\downarrow$

[M1] satisfied ? — yes $\rightarrow$

no $\downarrow$

Compute mem occupation
$$r_i = \frac{mem(w_i, Q_w^i)}{tot_{MEM}}$$
$$R = \max r_i$$

$\downarrow$

Cut Q_w^i of the lower layer with a mem occupation $r_i > R - \delta$

Weights Quantization Policy

[M1] : size(w0) + size(w1) + size (w2) + size(w3) < M_{ROM}

$\delta = 5\%$

$Q_w^0 = 8$
17%

$Q_w^1 = 8$
20%
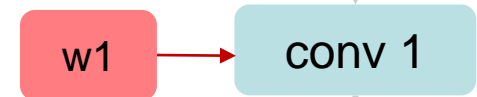
$Q_w^2 = 4$
30%
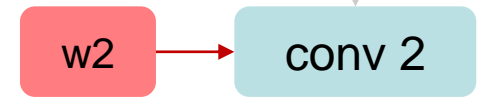
$Q_w^3 = 4$
33%

w0 $\rightarrow$ conv 0

w1 $\rightarrow$ conv 1

w2 $\rightarrow$ conv 2

w3 $\rightarrow$ conv 3

Cut layer 2!

Rule-Based Mixed-Precision

Goal
Maximize memory utilization

conv 0

size y_0

$\geq$

conv 1

size y_1 ← Cut this?

conv 2

size y_2

conv 3

size y_3

forward

Any cut reduces the bit precision by one step: 8→4, 4→2

Set $Q_x^i = Q_y^{i-1} = 8$

[M2] satisfied ? — yes

no

For any layer i
While $mem(y_i, Q_y^i) > mem(x_i, Q_x^i)$ and !M2:
Cut Q_y^i

forward

For any layer i
While $mem(y_i, Q_y^i) < mem(x_i, Q_x^i)$ and !M2:
Cut Q_x^i

backward

Activation Quantization Policy

[M2]
size y_0 +size $y_1 < M_{RAM}$
size y_1 +size $y_3 < M_{RAM}$
size y_2 +size $y_3 < M_{RAM}$

Rule-Based Mixed-Precision

Goal
Maximize memory utilization

forward

conv 0

size y_0

conv 1

size y_1

conv 2 ≥

size y_2 ← Cut this?

conv 3

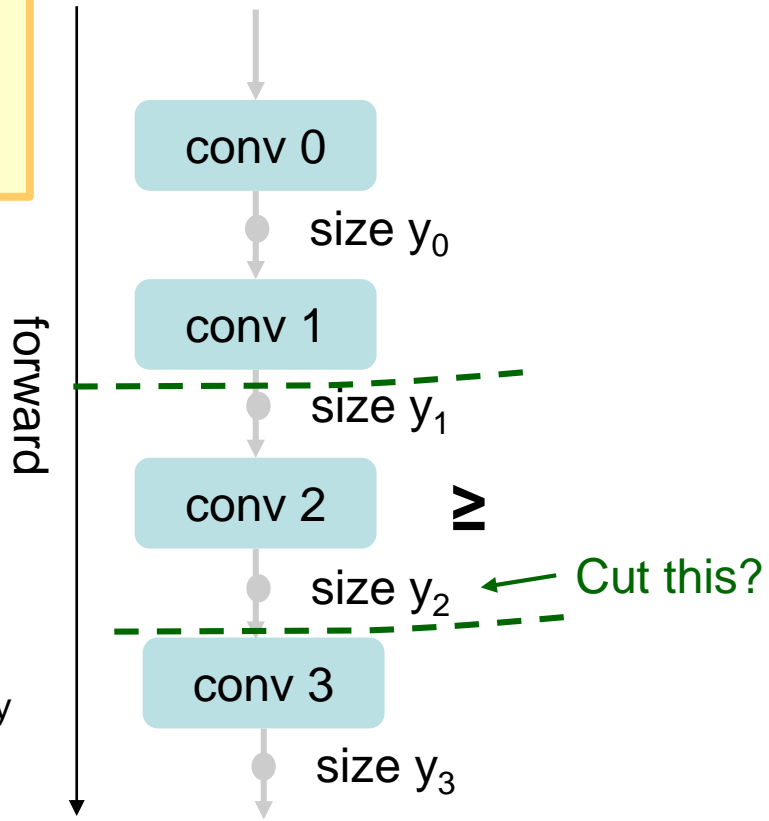size y_3

Any cut reduces the bit precision by one step: 8→4, 4→2

[M2]
size y_0 +size $y_1 < M_{RAM}$
size y_1 +size $y_3 < M_{RAM}$
size y_2 +size $y_3 < M_{RAM}$

Set $Q_x^i = Q_y^{i-1} = 8$

[M2] satisfied ? — yes

no

For any layer i
While $mem(y_i, Q_y^i) > mem(x_i, Q_x^i)$ and !M2:
Cut Q_y^i

forward

For any layer i
While $mem(y_i, Q_y^i) < mem(x_i, Q_x^i)$ and !M2:
Cut Q_x^i

backward
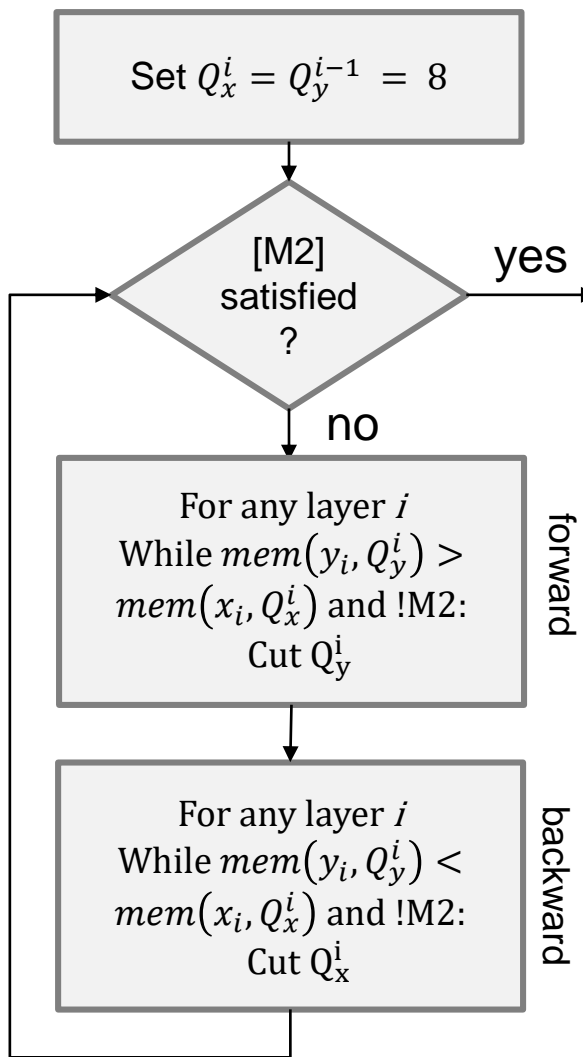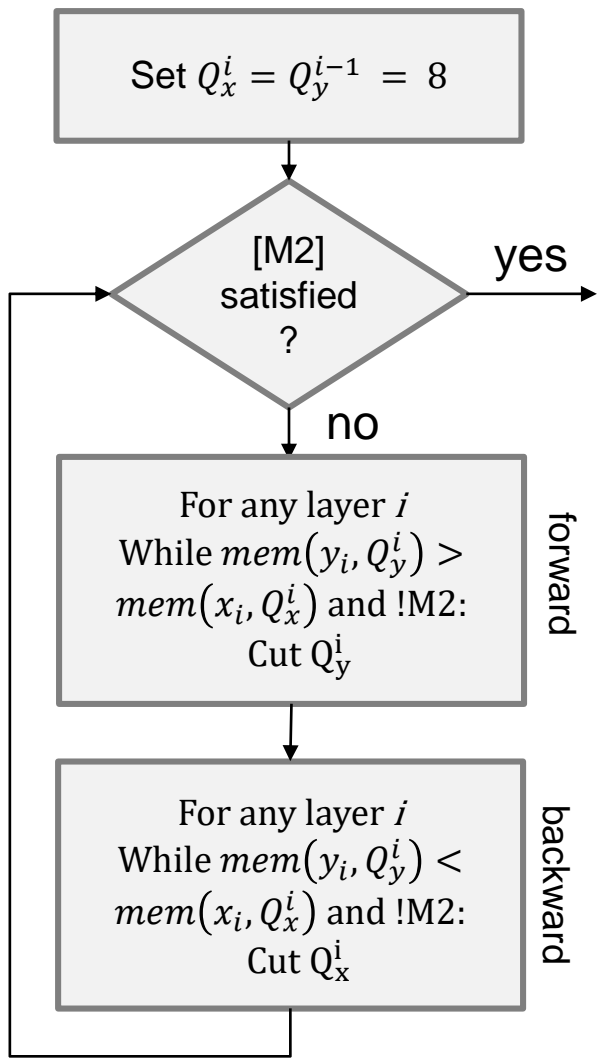
Activation Quantization Policy

Rule-Based Mixed-Precision

Goal Maximize memory utilization

conv 0

size y_0

conv 1

size y_1 ← Cut this?

conv 2 $\geq$

size y_2

conv 3

size y_3

backward

Any cut reduces the bit precision by one step: $8 \rightarrow 4$, $4 \rightarrow 2$

Set $Q_x^i = Q_y^{i-1} = 8$

[M2] satisfied? — yes

no

For any layer i
While $mem(y_i, Q_y^i) > mem(x_i, Q_x^i)$ and !M2:
Cut Q_y^i

forward

For any layer i
While $mem(y_i, Q_y^i) < mem(x_i, Q_x^i)$ and !M2:
Cut Q_x^i

backward

Activation Quantization Policy

[M2]
size y_0 +size $y_1 < M_{RAM}$
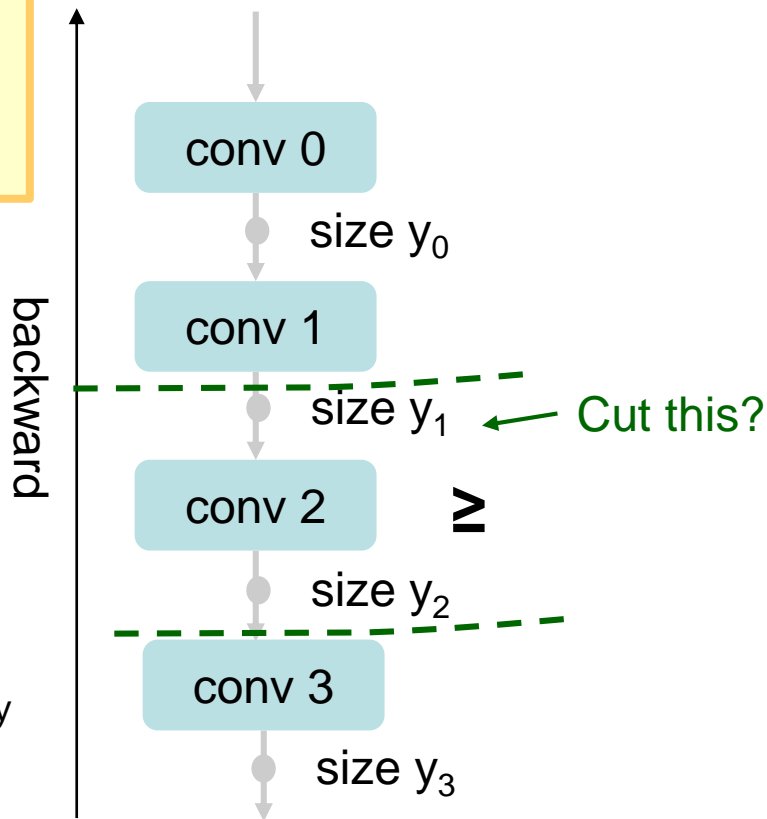size y_1 +size $y_3 < M_{RAM}$
size y_2 +size $y_3 < M_{RAM}$

Experimental Results on MobilenetV1

Iso-memory MobilenetV1 models with 2MB FLASH and 512kB RAM.

| | | | Integer-only | |
Model	Mparams	Full-Prec	Mix-PC	Mix-PL
224_1.0	4.24	**70.9**	64.3	59.6
192_1.0	4.24	70.0	65.9	61.9
224_0.75	2.59	68.4	**68.0**	**67.0**
192_0.75	2.59	67.2	67.2	64.8
224_0.5	1.34	63.3	63.5	63.1
192_0.5	1.34	61.7	62.0	59.5

Quantization-aware Fine-Tuning recipe:
- ❑ Init w/ pre-trained params
- ❑ 8H on 4 NVIDIA Tesla P100
- ❑ ADAM, lr=1e-4 (5e-5 @5eph, 1e-5 at 8 eph)
- ❑ Frozen batch norm stats after 1 eph
- ❑ Asymmetric quant on weights, either PC (min/max) or PL (PACT)
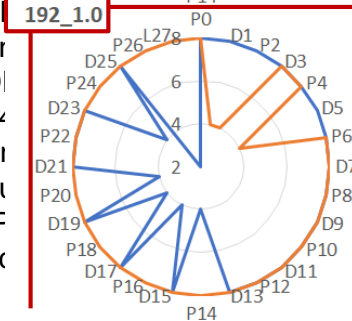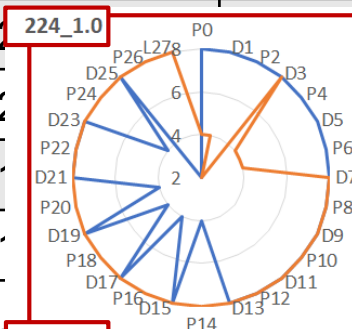- ❑ Asymmetric activation (PACT)

Open source: https://github.com/mrusci/training-mixed-precision-quantized-networks

Experimental Results on MobilenetV1

Iso-memory MobilenetV1 models with 2MB FLASH and 512kB RAM.

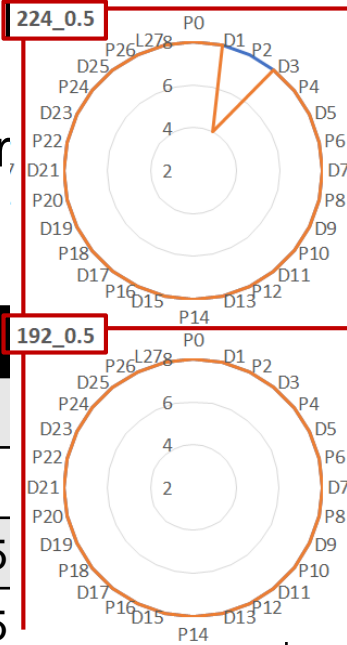| Model	Mparams	Full-Prec	Integer-only	
			Mix-PC	Mix-PL
224_1.0	4.24	**70.9**	64.3	59.6
192_1.0	4.24	70.0	65.9	61.9
224_0.75			**68.0**	**67.0**
192_0.75			67.2	64.8
224_0.5			63.5	63.1
192_0.5			62.0	59.5

Higher drop due to more aggressive cuts

Quantization-aware
- Init w/ pre-trai
- 8H on 4 NVIDI
- ADAM, lr=1e-4
- Frozen batch r
- Asymmetric qu (min/max) or F
- Asymmetric ad

Open source: https://github.com/mrusci/training-mixed-precision-quantized-networks

Experimental Results on MobilenetV1

Iso-memory MobilenetV1 models with 2MB FLASH and 512kB RAM.

| Model | MB | Full-Prec | Integer-only | |
			Mix-PC	Mix-PL
224_1.0		9	64.3	59.6
192_1.0		0	65.9	61.9
224_0.75		4	**68.0**	**67.0**
192_0.75		2	67.2	64.8
224_0.5	1.34	63.3	63.5	63.1
192_0.5	1.34	61.7	62.0	59.5

Higher drop due to more aggressive cuts

Lossless, ~8 bit fits the memory constraints

Quantization-aware Fine-Tuning recipe:
- ❑ Init w/ pre-trained params
- ❑ 8H on 4 NVIDIA Tesla P100
- ❑ ADAM, lr=1e-4 (5e-5 @5eph, 1e-5 at 8 eph)
- ❑ Frozen batch norm stats after 1 eph
- ❑ Asymmetric quant on weights, either PC (min/max) or PL (PACT)
- ❑ Asymmetric activation (PACT)

Open source: https://github.com/mrusci/training-mixed-precision-quantized-networks

Experimental Results on MobilenetV1

Iso-memory MobilenetV1 models with 2MB FLASH and 512kB RAM.

| Model | Mparams | Full-Prec | Integer-only | |
			Mix-PC	Mix-PL
224_1.0	4.24	**70.9**	64.3	59.6
192_1.0	4.24	70.0	65.9	61.9
224_0.75	2.59	68.4	**68.0**	**67.0**
192_0.75	2.59	67.2	67.2	64.8
224_0.5			63.5	63.1
192_0.5			62.0	59.5

Higher drop due to more aggressive cuts

Nearly lossless, few but 'significant' cuts ☺

Lossless, ~8 bit fits the memory constraints
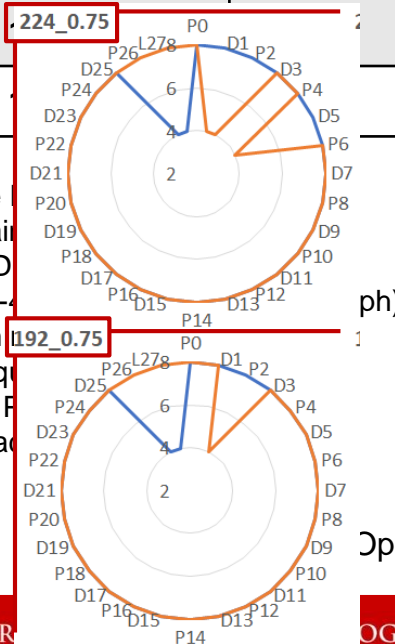
Quantization-aware
- ❑ Init w/ pre-trai
- ❑ 8H on 4 NVID
- ❑ ADAM, lr=1e-4 ...ph)
- ❑ Frozen batch
- ❑ Asymmetric q... (min/max) or F
- ❑ Asymmetric a

Overall, an integer-only network running on MCU with -2.9% accuracy drop wrt to the most precise model

Open source: https://github.com/mrusci/training-mixed-precision-quantized-networks

DNN Development Flow for microcontrollers
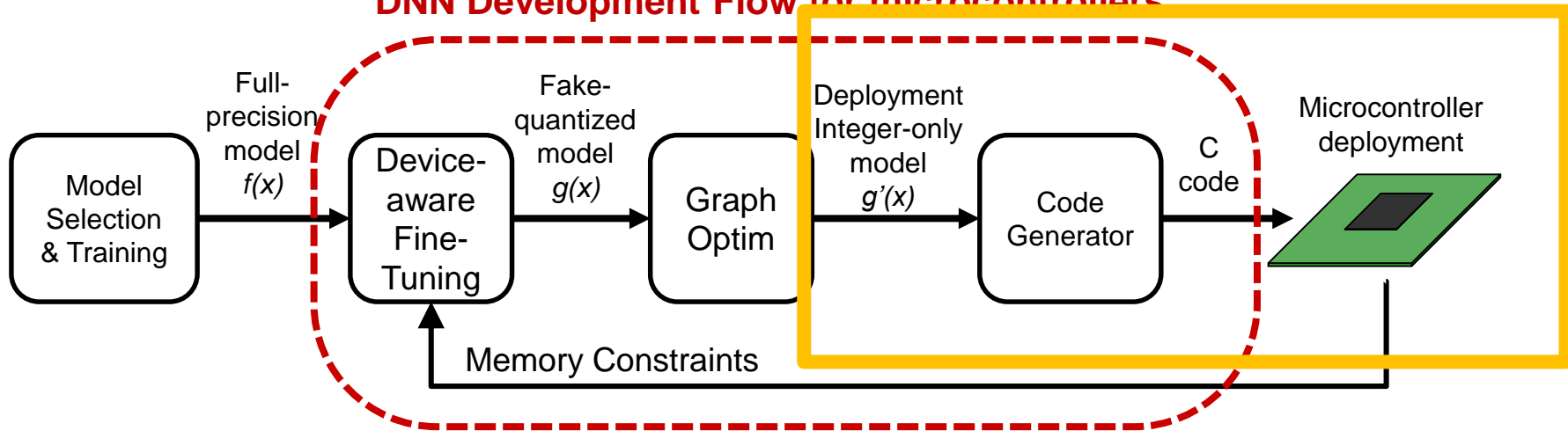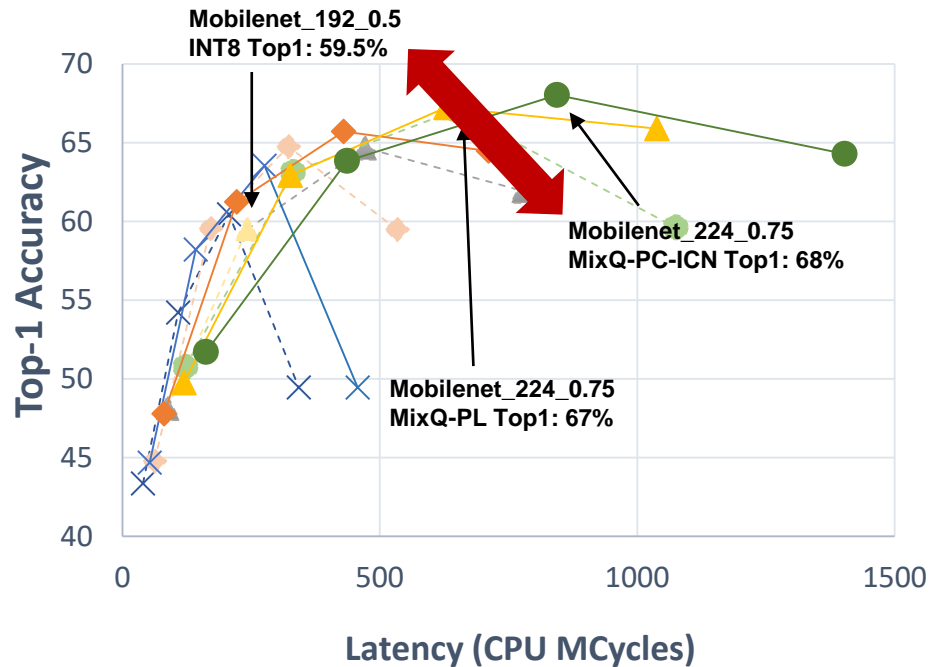
Deployments on MCUs

LATENCY-ACCURACY TRADE-OFF ON A STM32H7 MCU

Latency-Accuracy Trade Off

Experiments runs on a STM32H743 (400MHz clk)

Mobilenet_192_0.5
INT8 Top1: 59.5%

Mobilenet_224_0.75
MixQ-PC-ICN Top1: 68%

Mobilenet_224_0.75
MixQ-PL Top1: 67%

Top-1 Accuracy vs **Latency (CPU MCycles)**

Legend:
- --✗-- 128-MixQ-PL
- --◆-- 160-MixQ-PL
- --▲-- 192-MixQ-PL
- --■-- 224-MixQ-PL
- ──✗── 128-MixQ-PC-ICN
- ──◆── 160-MixQ-PC-ICN
- ──▲── 192-MixQ-PC-ICN
- ──●── 224-MixQ-PC-ICN

➤ The implementation is based on the sw lib for mixed-precision inference (based on Cmsis-NN):
  ❑ Cmix-NN:
    https://github.com/EEESlab/CMix-NN
  ❑ UINT2-4 software emulated
  ❑ MAC 2x16 bits
➤ PC on the pareto
➤ But PC slower than PL by 20-30%

$$\Phi = \sum (X_q - Z_x) \cdot (W_q - Z_w)$$
$$= \sum X_{im2col} \cdot (W_q - Z_w) \quad \textbf{PC}$$
$$= \sum X_{im2col} \cdot W_q - \sum X_{im2col} \cdot Z_w \quad \textbf{PL}$$

Overall +8% with respect to best 8-bit integer-only MobilenetV1 fitting the device (Jacob et al. 2018)

Wrap-up

- We proposed an **end-to-end methodology** to train and deploy 'complex' DL models on **tiny MCUs**.
    - **sub-byte** uniform quantization
    - **mixed-precision** settings
    - a **memory-driven** rule-based method for determine the quantization policy
    - integer-only transformation with **ICN** activation layers
    - mixed precision **software** library for MCU

- Deployment of a 68% Imagenet MobilenetV1 into a MCU with 2MB FLASH and 512 kB RAM.