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

- 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...

**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.

---

**Device-aware Fine-Tuning**

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

**Graph Optimization**

We introduce the Integer Channel-Normalization (ICN) activation layer to generate an integer-only deployment graph when applying uniform sub-byte quantization.

**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.

---

**DNN Development Flow for microcontrollers**

1. **Model Selection & Training**
   - Full-precision model $f(x)$

2. **Device-aware Fine-Tuning**
   - Fake-quantized model $g(x)$

3. **Graph Optim**
   - Deployment Integer-only model $g'(x)$

4. **Code Generator**
   - C code

5. **Microcontroller deployment**

---

**Memory Constraints**

Goal: Define a design flow to bring Imagenet-size models into an MCU device while paying a low accuracy drop.
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 (uniform) quantized parameters
  - Quantization-aware retraining
  - Folding of batch norm into conv weights + rounding of per-layer scaling parameters

\[ t = S_t \times (T_q - Z_t) \]

- 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

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

\[ Y_q = quant_{act} \left( \frac{\phi - \mu}{\sigma} \cdot \gamma + \beta \right) \]

\[ \phi = \sum_{w} x \]

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

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

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

\( S_w \) is scalar if PL, else array
\( S_i, S_o \) are scalar

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

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

\( M_0, N_0, B_q \) are channel-wise integer params

Integer-Only MobilenetV1_224_1.0

<table>
<thead>
<tr>
<th>Quantization Method</th>
<th>Top1</th>
<th>Weights (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full-Precision</td>
<td>70.9</td>
<td>16.8</td>
</tr>
<tr>
<td>PL+ICN w4a4</td>
<td>61.75</td>
<td>2.10</td>
</tr>
<tr>
<td>PC+ICN w4a4</td>
<td>66.41</td>
<td>2.12</td>
</tr>
</tbody>
</table>

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?

**Weight Parameters**
- weight 0
- weight 1
- weight 2
- weight 3
- weight 4

**Input Data**

**Output Data**

**Graph Representation**
- conv 0
- conv 1
- conv 2
- conv 3
- add 0
- conv 4

**Memory Constraints**
- \( M_{\text{ROM}} \)
- \( M_{\text{RAM}} \)
Deployment of an integer-only graph

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

- **Weight Parameters**
- **Read-only memory** $M_{ROM}$ for static parameters
- **Input Data**
- **Read-write memory** $M_{RAM}$ for dynamic values
- **Output Data**
Deployment of an integer-only graph

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

Goal
Maximize memory utilization

Weights
Quantization Policy

Set \( Q_w^i = 8 \)

\[ [M1] : \text{size}(w_0) + \text{size}(w_1) + \text{size}(w_2) + \text{size}(w_3) < M_{ROM} \]

\( \delta = 5\% \)

\( Q_w^0 = 8 \)
\( Q_w^1 = 8 \)
\( Q_w^2 = 8 \)
\( Q_w^3 = 8 \)

\( 22\% \)
\( 15\% \)
\( 13\% \)
\( 50\% \)

Conv 0
Conv 1
Conv 2
Conv 3

\( R = \max r_i \)

Compute mem occupation
\( r_i = \frac{\text{mem}(w_i, Q_w^i)}{\text{totMEM}} \)

Cut \( Q_w^i \) of the lower layer with a mem occupation
\( r_i > R - \delta \)
Rule-Based Mixed-Precision

Goal
Maximize memory utilization

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

Weights
Quantization Policy

Cut layer 3!

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

δ = 5%

w0 → conv 0
w1 → conv 1
w2 → conv 2
w3 → conv 3

Set $Q_w^i = 8$

Compute mem occupation

$R = \max r_i$

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

[M1] satisfied?

[17% → 13%]

$Q_w^0 = 8$

$Q_w^1 = 8$

$Q_w^2 = 8$

$Q_w^3 = 4$

17% → 13%

20% → 15%

30% → 22%

33% → 50%
Rule-Based Mixed-Precision

Goal
Maximize memory utilization

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

Compute mem occupation
\[ r_i = \frac{\text{mem}(w_i, Q_w^i)}{\text{tot}_{MEM}} \]
\[ R = \max r_i \]

Cut \( Q_w^i \) of the lower layer with a mem occupation
\[ r_i > R - \delta \]

Weights
Quantization Policy

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

\[ \delta = 5\% \]

Cut layer 2!
Rule-Based Mixed-Precision

**Goal**
Maximize memory utilization

- **[M2]**
  - $\text{size } y_0 + \text{size } y_1 < M_{RAM}$
  - $\text{size } y_1 + \text{size } y_3 < M_{RAM}$
  - $\text{size } y_2 + \text{size } y_3 < M_{RAM}$

For any layer $i$

- While $\text{mem}(y_i, Q_y^i) > \text{mem}(x_i, Q_x^i)$ and !M2:
  - Cut $Q_y^i$ for forward
  - Cut $Q_x^i$ for backward

Any cut reduces the bit precision by one step: $8 \rightarrow 4, 4 \rightarrow 2$
Rule-Based Mixed-Precision

Goal
Maximize memory utilization

Conv 0
size $y_0$

Conv 1
size $y_1$

Conv 2
size $y_2$

Conv 3
size $y_3$

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

Activation Quantization Policy

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

[M2] satisfied?

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

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

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

$\text{size } y_0 + \text{size } y_1 < M_{\text{RAM}}$
$\text{size } y_1 + \text{size } y_3 < M_{\text{RAM}}$
$\text{size } y_2 + \text{size } y_3 < M_{\text{RAM}}$
Rule-Based Mixed-Precision

Goal
Maximize memory utilization

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

Goal
Maximize memory utilization

Conv 0
size $y_0$

Conv 1
size $y_1$

Cut this?

Conv 2
size $y_2$ ≥

Conv 3
size $y_3$

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

[M2] $\text{size } y_0 + \text{size } y_1 < M_{\text{RAM}}$

$\text{size } y_1 + \text{size } y_3 < M_{\text{RAM}}$

$\text{size } y_2 + \text{size } y_3 < M_{\text{RAM}}$

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

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

Forward

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

Activation Quantization Policy
Experimental Results on MobilenetV1

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Mparams</th>
<th>Full-Prec</th>
<th>Mix-PC</th>
<th>Mix-PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>224_1.0</td>
<td>4.24</td>
<td>70.9</td>
<td>64.3</td>
<td>59.6</td>
</tr>
<tr>
<td>192_1.0</td>
<td>4.24</td>
<td>70.0</td>
<td>65.9</td>
<td>61.9</td>
</tr>
<tr>
<td>224_0.75</td>
<td>2.59</td>
<td>68.4</td>
<td>68.0</td>
<td>67.0</td>
</tr>
<tr>
<td>192_0.75</td>
<td>2.59</td>
<td>67.2</td>
<td>67.2</td>
<td>64.8</td>
</tr>
<tr>
<td>224_0.5</td>
<td>1.34</td>
<td>63.3</td>
<td>63.5</td>
<td>63.1</td>
</tr>
<tr>
<td>192_0.5</td>
<td>1.34</td>
<td>61.7</td>
<td>62.0</td>
<td>59.5</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Mparams</th>
<th>Full-Prec</th>
<th>Mix-PC</th>
<th>Mix-PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>224_1.0</td>
<td>4.24</td>
<td>70.9</td>
<td>64.3</td>
<td>59.6</td>
</tr>
<tr>
<td>192_1.0</td>
<td>4.24</td>
<td>70.0</td>
<td>65.9</td>
<td>61.9</td>
</tr>
<tr>
<td>224_0.75</td>
<td></td>
<td></td>
<td>68.0</td>
<td>67.0</td>
</tr>
<tr>
<td>192_0.75</td>
<td></td>
<td></td>
<td>67.2</td>
<td>64.8</td>
</tr>
<tr>
<td>224_0.5</td>
<td></td>
<td></td>
<td>63.5</td>
<td>63.1</td>
</tr>
<tr>
<td>192_0.5</td>
<td></td>
<td></td>
<td>62.0</td>
<td>59.5</td>
</tr>
</tbody>
</table>

Higher drop due to more aggressive cuts

Integer-only

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

Experimental Results on MobilenetV1

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

<table>
<thead>
<tr>
<th>Model</th>
<th>Mparams</th>
<th>Full-Prec</th>
<th>Mix-PC</th>
<th>Mix-PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>224_1.0</td>
<td>4.24</td>
<td>70.9</td>
<td>64.3</td>
<td>59.6</td>
</tr>
<tr>
<td>192_1.0</td>
<td>4.24</td>
<td>70.0</td>
<td>65.9</td>
<td>61.9</td>
</tr>
<tr>
<td>224_0.75</td>
<td>2.59</td>
<td>68.4</td>
<td>68.0</td>
<td>67.0</td>
</tr>
<tr>
<td>192_0.75</td>
<td>2.59</td>
<td>67.2</td>
<td>67.2</td>
<td>64.8</td>
</tr>
<tr>
<td>224_0.5</td>
<td>1.34</td>
<td>63.3</td>
<td>63.5</td>
<td>63.1</td>
</tr>
<tr>
<td>192_0.5</td>
<td>1.34</td>
<td>61.7</td>
<td>62.0</td>
<td>59.5</td>
</tr>
</tbody>
</table>

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)

Higher drop due to more aggressive cuts
Lossless, ~8 bit fits the memory constraints

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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Mparams</th>
<th>Full-Prec</th>
<th>Mix-PC</th>
<th>Mix-PL</th>
</tr>
</thead>
<tbody>
<tr>
<td>224_1.0</td>
<td>4.24</td>
<td>70.9</td>
<td>64.3</td>
<td>59.6</td>
</tr>
<tr>
<td>192_1.0</td>
<td>4.24</td>
<td>70.0</td>
<td>65.9</td>
<td>61.9</td>
</tr>
<tr>
<td>224_0.75</td>
<td>2.59</td>
<td>68.4</td>
<td>68.0</td>
<td>67.0</td>
</tr>
<tr>
<td>192_0.75</td>
<td>2.59</td>
<td>67.2</td>
<td>67.2</td>
<td>64.8</td>
</tr>
<tr>
<td>224_0.5</td>
<td>1.34</td>
<td>63.3</td>
<td>63.5</td>
<td>63.1</td>
</tr>
<tr>
<td>192_0.5</td>
<td>1.34</td>
<td>61.7</td>
<td>62.0</td>
<td>59.5</td>
</tr>
</tbody>
</table>

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 8eph)
- Frozen batch norm stats after 1 epoch
- Asymmetric quant on weights, either PC (min/max) or PL (PACT)
- Asymmetric activation (PACT)

Lossless, ~8 bit fits the memory constraints.

Higher drop due to more aggressive cuts.

Nearly lossless, few but ‘significant’ cuts 😊

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
Deployments on MCUs

LATENCY-ACCURACY TRADE-OFF ON A STM32H7 MCU
Latency-Accuracy Trade Off

Experiments runs on a STM32H743 (400MHz clk)

- The implementation is based on the sw lib for mixed-precision inference (based on Cmsis-NN):
  - Cmix-NN: [https://github.com/EEESlab/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%

\[
\Phi = \sum(X_q - Z_x) \cdot (W_q - Z_w)
\]
\[
= \sum X_{im2col} \cdot (W_q - Z_w) \quad \text{PC}
\]
\[
= \sum X_{im2col} \cdot W_q - \sum X_{im2col} \cdot Z_w \quad \text{PL}
\]

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.