MEMORY-DRIVEN MIXED LOW PRECISION QUANTIZATION FOR ENABLING DEEP NETWORK INFERENCE ON MICROCONTROLLERS

Manuele Rusci [1] **Alessandro Capotondi** [2] **Luca Benini** [1 3]

ABSTRACT

This paper presents a novel end-to-end methodology for enabling the deployment of high-accuracy deep networks on microcontrollers. To fit within the memory and computational limitations of resource-constrained edge-devices, we exploit mixed low-bitwidth compression, featuring 8, 4 or 2-bit uniform quantization, and we model the inference graph with integer-only operations. Our approach aims at determining the minimum bit precision of every activation and weight tensor given the memory constraints of a device. This is achieved through a rule-based iterative procedure, which cuts the number of bits of the most memory-demanding layers, aiming at meeting the memory constraints. After a quantization-aware retraining step, the fake-quantized graph is converted into an inference integer-only model by inserting the Integer Channel-Normalization (ICN) layers, which introduce a negligible loss as demonstrated on INT4 MobilenetV1 models. We report the latency-accuracy evaluation of mixed-precision MobilenetV1 family networks on a STM32H7 microcontroller. Our experimental results demonstrate an end-to-end deployment of an integer-only Mobilenet network with Top1 accuracy of 68% on a device with only 2MB of FLASH memory and 512kB of RAM, improving by 8% the Top1 accuracy with respect to previously published 8 bit implementations for microcontrollers.

1 INTRODUCTION

Enabling machine learning on extreme-edge-devices is challenging due to their tight memory and computing power constraints. When envisioning smart sensors operating on batteries, the target power envelope must be below tens of mWs to guarantee a battery lifetime of years. This requirement impacts the system architecture design: adding computational units (e.g. floating-point units) or memory banks contributes increasing the complexity and the power cost, and hence the energy, of a system.

Nowadays, microcontroller units (MCUs), such STMicroelectronics STM32 devices, feature an energy consumption compliant with the requirement of smart autonomous sensors and include energy-efficient computational units for running machine learning workloads. However, the typical size of the embedded memory cuts is limited to a few MB (a STM32H7 MCU features 2MB of FLASH memory) and the computation core (commonly a single ARM Cortex-M CPU) runs up to few hundreds of MHz. To boost the performance of this class of MCUs while leveraging the high flexibility of software-programmability, ARM recently released a software library, CMSIS-NN (Lai et al., 2018), which enabled the efficient computation of deep networks on tiny microcontrollers. The optimized routines composing the library realize convolutional operations in fixed-point representations, to exploit instruction-level parallelism. Unfortunately, due to memory constraints, only a small set of relatively complex networks has been ported to the microcontroller domain yet (Zhang et al., 2017). For what concerns deep inference models tailored for complex tasks, e.g. 1000 classes image classification, the deployment on memory-constrained MCUs is still an open problem.

To address this, recent works focused on designing novel network topologies optimized not only in terms of accuracy but also for computational and memory costs (Howard et al., 2017; Ma et al., 2018; Wu et al., 2018). In addition, a variety of compression techniques can be applied to further shrink a trained model. Among these, the quantization of either activations values and parameters to a low-bitwidth format, i.e. 8 bit or less, is extremely effective because, besides reducing the memory footprint, it allows to operate with low precision integer operations, which can be efficiently mapped on the limited instruction-set of tiny microcontrollers. Figure 1 highlights a typical development flow to deploy a deep network design into a resource-constrained device. A pretrained network $f(x)$ is quantized by means of an initial device-aware fine tuning process, which can include also a re-training step. The resultant fake-

[1]DEI, Universita' di Bologna, Bologna, Italy [2]Universita' di Modena e Reggio Emilia, Italy [3]D-ITET, ETH Zurich, Switzerland. Correspondence to: Manuele Rusci <manuele.rusci@unibo.it>.

Proceedings of the 3^{rd} MLSys Conference, Austin, TX, USA, 2020.

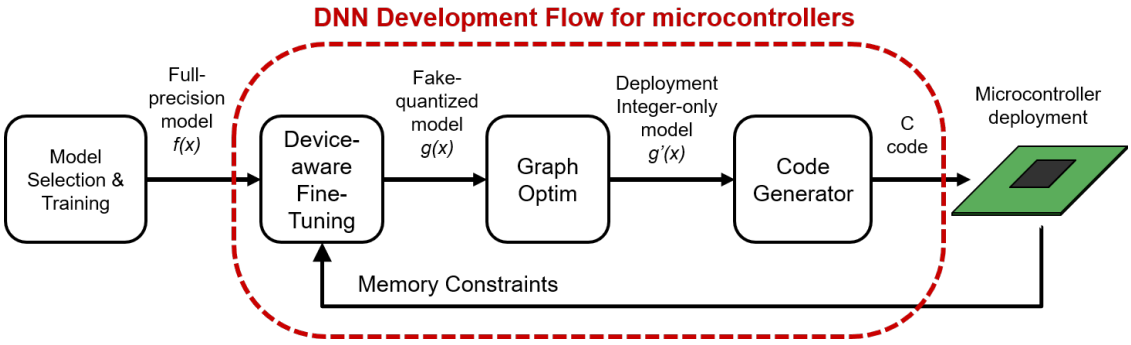

Figure 1. Design flow to bring deep neural networks into tiny microntrollers.

quantized model $g(x)$, emulating quantized values during the forward pass, is turned into an integer-only deployment model $g'(x)$ by means of an additional optimization step. Ideally, $\text{loss}(g'(x)) \approx \text{loss}(g(x)) \approx \text{loss}(f(x))$.

State-of-the-art quantization approaches lead to almost-zero accuracy loss if approximating a deep models with 8 bits arithmetic (Jacob et al., 2018). This compression level is however not sufficient to bring deep models with high accuracy into memory-constrained microcontrollers. As an example, a 8 bit MobilenetV1 (Howard et al., 2017) with the highest accuracy requires more than 4 MB of embedded memory, which is prohibitive for the majority of microcontroller devices available today. If homogeneously lowering the number of bits below 8 bits on a per-network base, the accuracy degradation becomes not negligible (Krishnamoorthi, 2018). To keep the accuracy level high, the bit precision of individual tensors should be tuned such as i) to fit the memory constraints and ii) to minimize the reduction of the bitwidth (Dong et al., 2019). These needs motivate our proposed heterogeneous sub-byte quantization approach, denoted as Mixed Low Precision Quantization, which finely controls the per-tensor bit precision in accordance to the memory budget. Moreover, the compression scheme must be combined with novel techniques for deriving integer-only inference models, required to accelerate deep learning workloads on microntrollers.

In this work we present a methodology for quantizing deep networks based on a mixed-precision scheme. The selection of the bit precision of every individual tensor is automated such as to satisfy the memory limitations of a given device. Moreover, we improve the methodology (Jacob et al., 2018) for integer-only inference networks by supporting sub-byte per-channel quantization. Our experimental evaluation is conducted over the MobilenetV1 family networks on the 1000 classes Imagenet classification task (Howard et al., 2017). We argue that this is a representative problem for tiny microntrollers, not yet solved (Jain et al., 2019), and much harder than quantizing over-parameterized networks (Choi et al., 2018).

This paper places the following contributions:

- We introduce the Integer Channel-Normalization (ICN) activation layer to achieve an efficient conversion of the fake-quantized graph into an integer-only deployment graph, also supporting per-channel quantization and quantization-aware training strategies.
- We present a mixed-precision quantization methodology driven by the memory constraints of a target architecture, which aims at selecting the bit precision of every weight and activation tensor of an integer-only network.
- We studied the latency-accuracy tradeoff on iso-memory mixed-precision networks belonging to the MobilenetV1 family when running on a STM32H7 microcontroller device.

Our methodology demonstrates, for the very first time, an integer-only deployment of a MobilenetV1 network on a STM32H7 microcontroller, featuring only 2MB of FLASH memory and 512kB of RAM, with 68% Top1 accuracy, which is 8% higher than previous reported 8 bit integer-only implementations fitting into the same memory constraints (Jacob et al., 2018).

2 Related Work

Quantized Neural Networks. Early works on quantization of deep networks targeted 16 bits fixed-point implementations (Lin et al., 2016), which result in an almost lossless approximation of full-precision trained networks, or extreme binarized networks, which, despite the fascinating low-computational and memory requirements, showed major accuracy losses when applied on image classification benchmarks (Courbariaux et al., 2016; Rastegari et al., 2016). Several studies demonstrated that 8 bit quantization

of weights and activations results in a good trade-off between latency, compression and a near-zero accuracy degradation, also if applied to efficient Imagenet classification networks (Jacob et al., 2018; Migacz, 2017; Jain et al., 2019). Among the employed methodologies, TensorRT (Migacz, 2017) approximates the parameters tensor by the minimization of the KL divergence metric between quantized and full-precision values. On the contrary, (Jacob et al., 2018) quantizes values within a range defined by the tensor min and max values. Concerning activations, the PACT approach (Choi et al., 2018) demonstrated the highest efficiency by leveraging backpropagation to learn the quantization ranges. Recently, to fit stringent memory requirements, more aggressive sub-byte precision quantization approaches, i.e. less than 8 bit, are under investigation (Choukroun et al., 2019; Jain et al., 2019; Esser et al., 2019; Krishnamoorthi, 2018; Liu & Mattina, 2019). The works (Jain et al., 2019; Esser et al., 2019) exploits learning-based approaches for determining the quantization ranges of activation and weights at low-bitwidth precision. State-of-the-art accuracy level on the efficient MobilenetV1 model has been reported by (Krishnamoorthi, 2018; Liu & Mattina, 2019), by making use of per-channel quantization when moving to 4 bits precision. It is also worth to mention as non-uniform quantizers have resulted as the best approximators when reducing the bit precision (Zhang et al., 2018; Wang et al., 2018; Han et al., 2015). However, a high-precision (floating point) arithmetic is needed on uncompressed values within the datapath, hence these methods results not suitable for the microcontroller domain. In this work, we leverage existing techniques and show the insights, concerning either computational and memory aspects, when bringing fake-quantized networks to the integer-only arithmetic domain, which is not taken into consideration by this class of works.

Mixed Low Precision Quantization. Mixed-precision techniques make use of multiple bit precision throughout a quantized network, motivated by the fact that a lossy and aggressive linear cut is not necessary to reach a given compression rate. The method (Fromm et al., 2018) targeted per-pixel binarization based on a defined tensor mask. Despite achieving an extreme quantization level, a per-pixel quantization cannot be efficiently handled on a microcontroller, due to the control-based nature of the required dataflow. The HAWQ (Dong et al., 2019) method relies on a second order Hessian metric to define prioritization of tensor's bit precision to reduce, but without choosing the optimal per-tensor quantization level. On the same direction, HAQ (Wang et al., 2018) dynamically explores multiple low-bitwidth precision at training time by means of reinforcement learning. When optimizing for memory constraints, a non-uniform quantization is used. Compered to this, our methodology for bit precision selection applies statically, before quantization-aware retraining, and it is based on a rule-based iterative procedure. Both (Dong et al., 2019) and (Wang et al., 2018) reports superior accuracy than ours when compressing networks to a 1MB of memory footprint, but they rely on a non-uniform clustering quantization of floating-point parameters, therefore not fully-comparable with our work in terms of microcontroller readiness, as current MCUs are not equipped with the hardware needed for manipulation and computation on these data formats.

Deep networks for resource-constrained devices. To bridge the gap between the complexity on deep networks and the limitations of resource-constrained devices, device-aware optimization strategies have also been presented. The work (Blott et al., 2018) introduced FINN-R to quantize and deploy a generic model into constrained FPGA architectures. Their quantization approach makes use of integer thresholds (Umuroglu & Jahre, 2017; Gao et al., 2018; Rusci et al., 2018) for data compression. This method enabled a lossless integer representation of a fake-quantized networks, but demands larger memory footprint with respect to our proposed method. In contrast, the integer-only deployment in (Jacob et al., 2018) presented a compact fixed-point 8 bit quantization strategy, which performs the folding of batch-normalization and scaling factors into weights before applying a uniform quantizer. Additionally, per-layer fixed-point parameters are needed for adapting the dynamic range when passing data from a layer to the next one. In contrast with this work, our methodology generalizes the deployment process when a more effective quantization strategy is used, i.e. per-channel mixed-precision quantization.

3 Background on Low-Bitwidth Quantization

The quantization process aims at quantizing either the network parameters and the activations values, i.e. the temporary input and output values of the network layers. While the parameters can be quantized just before the inference (forward) pass (Migacz, 2017), the quantization of the activations requires the insertion of *fake-quantized* activation layers within the network graph. These additional layers are responsible for recording the activation range statistics, optionally via backpropagation (Choi et al., 2018), and apply quantization during the forward pass depending on the collected statistics. Because of injected quantization noise, the original full-precision network f is approximated with the correspondent fake-quantized function g. A quantization-aware retraining of a fake-quantized model is essential to recover accuracy, especially when low-bitwidth precision is employed (Jacob et al., 2018).

In the remainder of the paper we only focus on uniform quantization because its arithmetic is naturally supported by the instruction-set of general-purpose programmable MCUs. Hence, without loosing generalities, any tensor $t \in R^N$,

Table 1. Memory Requirements of a Quantized Convolutional Layer

Label	Z_x	Weights	Z_w	B_q	M_0	N_0	Z_y	Thresholds
PL+FB (Jacob et al., 2018)	1	$c_O \cdot k_w \cdot k_h \cdot c_I$	1	c_O	1	1	1	-
PL+ICN (our)	1	$c_O \cdot k_w \cdot k_h \cdot c_I$	1	c_O	c_O	c_O	1	-
PC+ICN (our)	1	$c_O \cdot k_w \cdot k_h \cdot c_I$	c_O	c_O	c_O	c_O	1	-
PC+Thresholds (Umuroglu & Jahre, 2017)	1	$c_O \cdot k_w \cdot k_h \cdot c_I$	c_O	-	-	-	1	$c_O \cdot 2^Q$

either representing weights or activations or only a subset of them, can be quantized across the range $[a, b]$ with a given number of Q bits (Jacob et al., 2018) as:

$$T \cdot S_t = \text{quant}(t) = \text{round}(\frac{\text{clamp}(t, a, b)}{S_t})S_t \quad (1)$$

where $S_t = \frac{b-a}{2^Q-1}$ is a real scaling parameter and T is an integer tensor.

Equation (1) derives from the mapping:

$$t = S_t \cdot (T_q - Z_t) \quad (2)$$

where Z_t is a bias parameter required to shift the numeric domain of the quantized tensors T_q into $[0, 2^Q - 1]$ or $[-2^{Q-1}, 2^{Q-1} - 1]$ ranges, representative of the UINT-Q and INT-Q datatypes. If $a = -b, b > 0$, the quantization range is symmetric and Z_t is zero.

In the case of weights, the parameters a and b can be computed as the min and max values of a tensor (Jacob et al., 2018) or by means of more sophisticated statistic analysis (Migacz, 2017) or via backpropagation (Choi et al., 2018). A Per-Layer (PL) quantization exploit single values a and b for the whole full-precision tensor, hence the Equation 1 is applied layer-wise. A Per-Channel (PC) procedure results more effective by independently approximating a given tensor along the outer dimension (Krishnamoorthi, 2018). This corresponds to compute the a and b parameters in correspondence of any output channel of the tensor.

To determine the quantization range of the activation values, statistics can be collected at training time during the forward pass, or against a specific calibration dataset. The PACT strategy demonstrated the effectiveness of learning b via backprogation while $a = 0$ to reproduce the non-linearity of the ReLU function. In our implementation, the $\text{round}()$ of Equation 1 is replaced by $\text{floor}()$ because of the lighter software implementation (the operand gets simply truncated, i.e. a shift operation), becoming: $\text{quant\_act}(x) = \text{floor}(\frac{\text{clamp}(x,0,b)}{S_x}) \cdot S_x, S_x = \frac{b}{2^Q-1}$.

4 Integer-Only Inference

Previous work (Jacob et al., 2018) discussed the training and integer-only deployment of a fake-quantized network with 8 bit per-layer quantization. The weight quantization is applied after folding the batch-norm parameters into the convolutional weights. However, when reducing the bit precision below 8 bit using per-layer quantization, the folding process itself can lead to accuracy drop because it can drastically affects the range of the parameters to quantize. As a reference, Table 2 shows the collapse of the training process for INT4 MobilenetV1 with the folding of the batch-norm parameters enabled.

With the aim of an integer-only deployment, we extend (Jacob et al., 2018) to a) prevent the folding of batch normalization parameters into convolutional weights and b) support per-channel low-bitwidth weight quantization. We observe that any fake-quantized network's sub-graph composed by a convolutional layer, a batch-normalization layer and a fake-quantizer activation module can be modeled by the transfer function:

$$y = \text{quant\_act}(\frac{\phi - \mu}{\sigma} \cdot \gamma + \beta) \quad (3)$$

where $\phi = \sum x \cdot w$ is the output of a full-precision convolution and $\mu, \sigma, \gamma, \beta$ are channel-wise full-precision parameters of a batch normalization layer. It is worth to note that this kind of formulation holds for any feature-wise or layer-wise scaling factor applied to the convolution's output tensor.

When applying a per-layer quantization of either input/output activations and weights, the Rule 2 is injected into Equation 3 that becomes:

$$Y = Z_y + \text{quant\_act}(\frac{S_i S_w}{S_o}\frac{\gamma}{\sigma}(\Phi + \frac{1}{S_i S_w} \cdot (B - \mu + \beta\frac{\sigma}{\gamma}))) \quad (4)$$

where $\Phi = \sum(X - Z_x) \cdot (W - Z_w)$ is the integer output of a low-bitwidth convolution. We define the arrays $B_q = \text{round}(\frac{1}{S_i S_w} \cdot (B - \mu + \beta\frac{\sigma}{\gamma}))$, i.e. the quantized bias, and $M = \frac{S_i S_w}{S_o}\frac{\gamma}{\sigma}$. As done by (Jacob et al., 2018), each element m_i of M can be decomposed as $m_i = m0_i \cdot 2^{n0_i}$, where $m0_i$ is a signed fractionary fixed-point number with $0.5 \leq \text{abs}(m0_i) < 1.0$. For the sake of notation, we indicate as M_0 and N_0 the two vectors such as $M = M_0 \cdot 2^{N_0}$. Given this, Equation 4 can be rewritten as:

Algorithm 1 Cut Activation Bits

Require: a fake-quantized network g of L stacked quantized convolutional layers, a $\mathrm{M_{RW}}$ memory constraint, a $Q_{a,min}$ minimum quantization level
Ensure: the bit precion $Q_x^i, Q_y^i, i = 0, ..L-1$ to satisfy (7)

$Q_y^i \equiv Q_x^{i+1} \leftarrow 8 \quad i = 0, ..L-1$ ▷ Initialization
while (7) is not True for every layer **do** ▷ Stop Condition
  for $i = 0$ to $L-2$ **do** ▷ Forward pass
    while $\mathrm{mem}(x_i, Q_x^i) + \mathrm{mem}(y_i, Q_y^i) > \mathrm{M_{RW}}$ **AND** $\mathrm{CutBits}(x_i, Q_x^i, y_i, Q_y^i)$ **do**
      Q_y^i and Q_x^{i+1} are decremented by one step
    end while
  end for
  for $i = L-1$ to 1 **do** ▷ Backward pass
    while $\mathrm{mem}(x_i, Q_x^i) + \mathrm{mem}(y_i, Q_y^i) > \mathrm{M_{RW}}$ **AND** $\mathrm{CutBits}(y_i, Q_y^i, x_i, Q_x^i)$ **do**
      Q_x^i and Q_y^{i-1} are decremented by one step
    end while
  end for
end while

function CUTBITS($x_1, Q_{x_1}, x_2, Q_{x_2}$) ▷ Return True if Q_{x_2} have to be decremented
  if $Q_{x_2} > Q_{a,min}$ **then**
    if $Q_{x_2} > Q_{x_1}$ **OR** ($Q_{x_2} == Q_{x_1}$ **AND** $\mathrm{mem}(x_2, Q_{x_2}) > \mathrm{mem}(x_1, Q_{x_1})$) **then**
      return True
    end if
  end if
  return False
end function

$$Y = Z_y + \mathrm{clamp}(\mathrm{floor}(\mathrm{M_0} \cdot 2^{\mathrm{N_0}} \cdot (\Phi + \mathrm{B_q})), 0, 2^{\mathrm{Q}} - 1) \tag{5}$$

Note that every value in Equation 5 is an integer or a fixed-point value, so that a quantized convolutional layer can be computed with an integer-only arithmetic. Since the static parameters M_0, N_0, B_q vary along the channel dimension, we name this activation function (Equation 5) as **Integer Channel-Normalization activation**, indicated as ICN. If weight parameters get quantized per-channel (PC), i.e. every output channel weight bank has its own S_w and Z_w values, Equation (5) still holds after deriving the B_q, M_0 and N_0 vector parameters accordingly.

4.1 Memory Requirement

Table 1 schematizes the memory requirements to compute the transfer Function 5, considering both per-layer (PL) or per-channel (PC) quantization and the ICN layer. The table reports the amount of parameters of a convolution operation with a k_w x k_h receptive field, c_I input channels and c_O output channels. The weight-parameters are stored in memory as UINT-Q, where Q denotes the number of bits, so that the represented numeric domain corresponds to [0, $2^Q - 1$]. Z_x, Z_w and Z_y are in a UINT8 format (Z_w as INT16 if PC is applied), B_q and M_0 are stored as INT32 and N_0 is a INT8 array. For comparison purpose, Table 1 reports also the higher memory requirement of a quantized convolutional layer if using the thresholding method proposed by (Umuroglu & Jahre, 2017; Gao et al., 2018), which exponentially increases with Q.

5 Memory-Driven Mixed Low Precision Methodology for MCU Deployment

To run deep networks on microcontrollers, the memory footprint is a stringent constraint. Given common microcontroller architectures (Zhang et al., 2017), we distinguish:

- **Read-Only (RO) Memory**, to store frozen inference parameters, i.e. parameters that will not change during the lifetime of a smart device.
- **Read-Write (RW) Memory**, to store temporary values, i.e. input and output of any quantized convolutional layer that depends on the current sensor data.

At any step of the inference pass, a pair of temporary activation tensors, i.e. the input and output of a layer, and the whole set of fixed parameters must be present in the memory. If considering a network of L stacked quantized convolutional layers and a device with $\mathrm{M_{RO}}$ and $\mathrm{M_{RW}}$ memory budget (expressed in bytes), the above requirement is translated as:

$$\sum_{i=0}^{L} \mathrm{mem}(w_i, Q_w^i) + M_{T_A^i} \leq \mathrm{M_{RO}} \tag{6}$$

Algorithm 2 Cut Weights Bits

Require: a fake-quantized network g of L stacked quantized convolutional layers, a $\mathrm{M_{RO}}$ memory constraint, a $Q_{w,min}$ minimum quantization level, a δ margin
Ensure: The bit precision $Q_w^i, i = 0, ..L-1$ to satisfy (6)
$Q_w^i \leftarrow 8$ ▷ Initialization
while $\sum_{i=0}^{L-1} \mathrm{mem}(w_i, Q_w^i) + M_{T_A^i} > \mathrm{M_{RO}}$ **do**
  Compute $r_i = \mathrm{mem}(w_i, Q_w^i) / \sum_{i=0}^{L-1} \mathrm{mem}(w_i, Q_w^i)$ for every layer with $Q_{w_i} > Q_{w,min}$
  Find $R = \max_i r_i$
  Among the layers with $r_i > (R - \delta)$, select the k-th with the smallest index i
  Q_w^k is decremented by one step
end while

where i indicates the i-th quantized convolutional layer and $\mathrm{mem(t, Q)}$ returns the memory footprint of a tensor t with bit precision Q. $M_{T_A^i}$ is the memory footprint of the additional set of layer's static parameters (see Table 1) with datatype detailed in Section 4.1. Concerning activation values:

$$\mathrm{mem}(x_i, Q_x^i) + \mathrm{mem}(y_i, Q_y^i) \leq \mathrm{M_{RW}} \quad i = 0, .., L-1 \tag{7}$$

to ensure input and output of any block fitting the available memory footprint. Our methodology aims determining the bit precision Q_w^i, Q_x^i, Q_y^i of any input x_i, output y_i and weight w_i tensor of the i-th layer, to match the memory constraints (6) and (7). Only the values of $Q = \{2, 4, 8\}$ are admittable solutions; Q_x^0 is fixed to 8. Note that $y^i \equiv x^{i+1}$, hence fixing Q_y^i is equivalent to set Q_x^{i+1}. Initially, the bit precision of every tensor is set as $Q = 8$. Algorithm 1 and Algorithm 2 reports the pseudo-code of the procedure to cut the bit precision of, respectively, activations and weights, under the hypothesis that exists a solution that satisfy (6) and (7). The procedure in Algorithm 1 iterates over the L quantized convolutiona layers in a forward and backward fashion: the bit precision of output tensors $Q_y^i \equiv Q_x^{i+1}$ are cut during the forward pass, reductions of the input tensors' precision $Q_x^i \equiv Q_y^{i-1}$ are applied during the backward pass. Any cut consists of reducing the bit precision by a single step, i.e. from 8 to 4 and from 4 to 2 bits, and it is applied if the number of bits of the intended tensor (output during forward or input during backward) is lower or equal, but with a higher footprint, than the other activation tensor of the i-th layer.

Algorithm 2 details the iterative procedure for cutting bits of the weights parameters. At any iteration, a layer score r_i is computed as the ratio between the layer's footprint of the i-th layer and the total occupation. Among the highest scores r_i within a δ margin, the layer with the lowest layer's index is selected for the cut. This heuristic rule is intended to balance the quantization level between the central layers and the last layers, which are more subject to aggressive cuts due to the typically higher number of parameters.

6 Experimental Results

We run experiments on the MobilenetV1 family networks (Howard et al., 2017) on Imagenet using the PyTorch framework. In the following, any model of the MobilenetV1 family is marked with a label x_y, where $x = \{128, 160, 192, 224\}$ is the spatial resolution of the input data and $y = \{0.25, 0.5, 0.75, 1.0\}$ refers to the width channel multiplier. The quantization-aware retraining starts from pre-trained weights[1]. Every training session executes on a compute node equipped with 4 NVIDIA-Tesla P100 GPUs for 8 hours. ADAM is chosen as optimizer with an initial learning rate of 1e-4, which is decreased in a fixed schedule to 5e-5 and 1e-5 at, respectively, the 5th and 8th epochs. Running statistics and learned parameters of batch-normalization layers are frozen after the first training epoch. Batch size is 128. An asymmetric uniform quantization is applied on weights: the PACT method is used in case of PL quantization while min/max statistics are employed in case of PC quantization. PPQ (Liu & Mattina, 2019) is applied for refining pre-trained weights before the quantization-aware retraining. Folding of batch-normalization parameters into weights, when applied layer-wise, starts from the 2nd training epoch. Activations are quantized with the PACT strategy. The code to reproduce our experiments is open-source [2].

[1] Pretrained weights are downloaded from https://github.com/tensorflow/models/blob/master/research/slim/nets/mobilenet_v1.md

[2] https://github.com/mrusci/training-mixed-precision-quantized-networks

Table 2. Integer-Only MobilenetV1_224_1.0

Quantization Method	Top1 Accuracy	Weight Memory Footprint
Full-precision (Jacob et al., 2018)	70.9%	16.27 MB
PL+FB INT8 (Jacob et al., 2018)	70.1%	4.06 MB
PL+FB INT4 (our)	0.1%	2.05 MB
PL+ICN INT4 (our)	61.75%	2.10 MB
PC+ICN INT4 (our)	66.41%	2.12 MB
PC W4A4 (Liu & Mattina, 2019)	64.3%	-
PC W4A8 (Krishnamoorthi, 2018)	65%	-
PC+Thresholds INT4 (our)	66.46%	2.35 MB

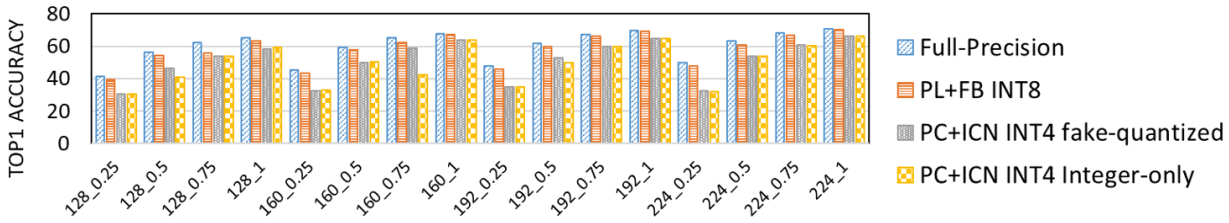

Figure 2. Top1 Accuracy of INT4 integer-only MobilenetV1 models, compared with full-precision, INT8 integer-only and INT4 fake-quantized models.

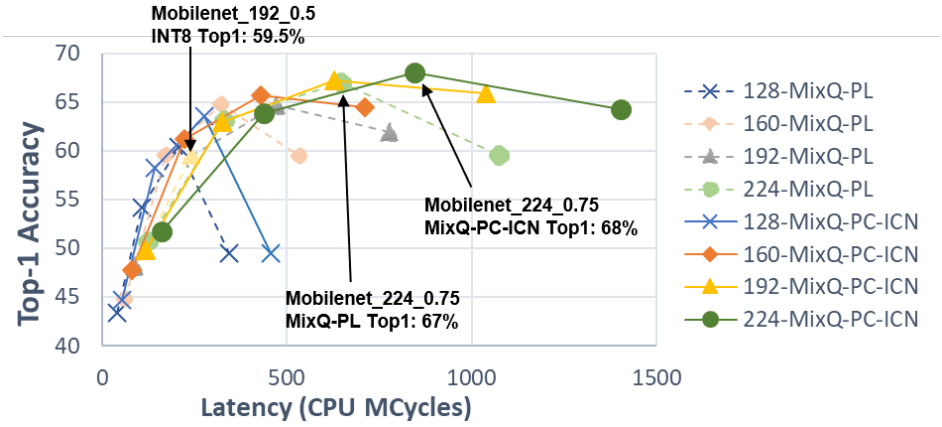

Figure 3. Accuracy-latency tradeoff of Mixed-Precision MobilenetV1 networks running on a STM32H7 device with $\mathrm{M_{RO}} = 2MB$ and $\mathrm{M_{RW}} = 512kB$.

To prove the effectiveness of the ICN layers, we quantize weights and activations of every layers of a MobilenetV1 224_1.0 model to 4 bits and we measure the accuracy achieved in case of integer-only approximation. Table 2 reports the accuracies for the following configurations: PL+FB stands for per-layer quantization and folding of batch-norm parameters into weights, PL+ICN indicates per-layer quantization with ICN layers and PC+ICN refers to per-channel quantization with ICN layers. First we can note that only thanks to the proposed ICN layers, the folding of the batch-norm parameters, which causes the collapse of the training process (PL+FB INT4), can be avoided, therefore enabling the convergence of the training algorithm (PL+ICN INT4 and PC+ICN INT4). Secondly, the insertion of the ICN layer introduces an almost negligible accuracy drop of 0.3% on PL+ICN and 0.05% on PC-ICN with respect to the fake-quantized graph. Moreover, by means of PC quantization, the accuracy of our 4 bit model is higher than other reported implementations (Krishnamoorthi, 2018; Liu & Mattina, 2019). In addition, Table 2 also reports the memory footprint of our PC+ICN INT4, which results to be 10% less memory-demanding than using the integer thresholds based methodology.

More in details, Figure 2 shows the Top1 accuracy of the family of INT4 integer-only PC+ICN Mobilenets. Compared with the related INT4 fake-quantized models, using ICN activations results into negligible loss. Only for the 160_0.75 case a relevant accuracy drop was observed. To recover it, we found effective to change the datatype of the quantized bias parameters to $Q30.2$, hence paying only an additional shift operation on the accumulator before of the bias addition.

After validating the ICN solution, we evaluate our proposed memory-driven methodology for the deployment of deep networks on microcontrollers. To this end, we apply our mixed-precision technique on all the Mobilenet configurations after setting the memory constraints $\mathrm{M_{RO}} = 2MB$ and $\mathrm{M_{RW}} = 512kB$, corresponding to the memory characteristics of an STM32H7 device. The trained integer-only models are deployed and bechmarked on the STM32H7 MCU running at 400MHz, to assess the implications for inference implementations. To this aim, we leverages an extended version of the ARM CMSIS-NN (Lai et al., 2018)

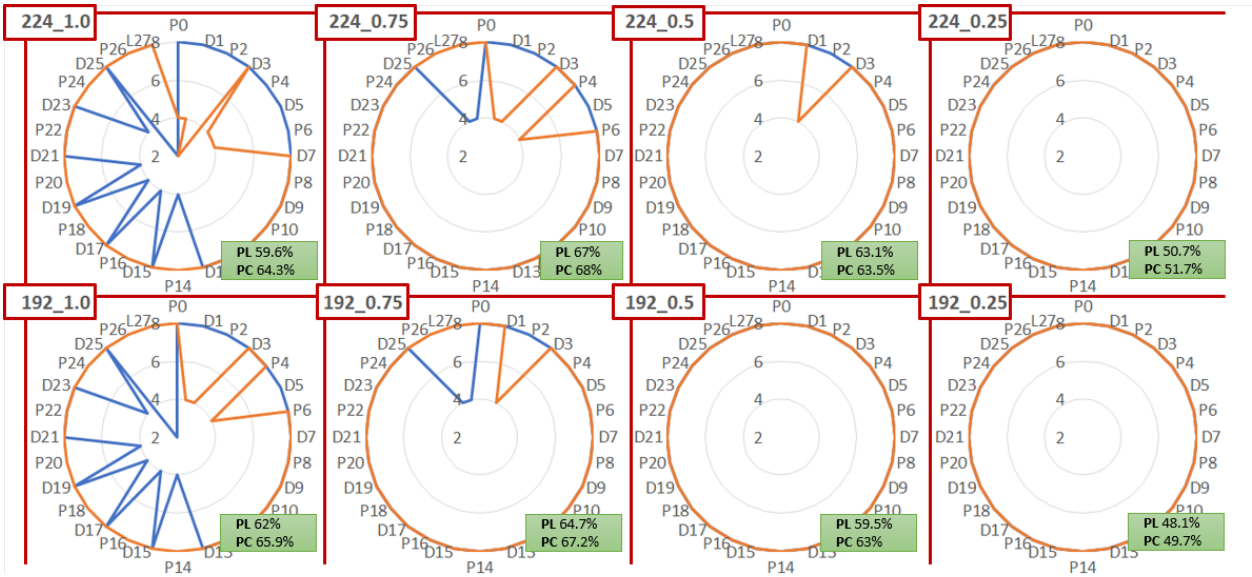

Figure 4. Bit precision (on the radial axes) of weights (blue curves) and activation output tensors (orange curves) for any layer (numbered from 0 to 27) of the MobilenetV1 models with input size 224 or 192, after setting the memory constraints $\mathrm{M_{RO}} = 2MB$ and $\mathrm{M_{RW}} = 512kB$. Top1 Accuracies on Imagenet are reported in the green boxes in case of Per-Layer (PL) or Per-Channel (PC) Mixed Precision Quantization.

library, featuring an output stationary dataflow, and we measure latency in terms of clock cycles. Figure 3 plots the accuracy-latency tradeoff measured on two configurations. MixQ-PL indicates per-layer quantization with either the folding of batch-norm parameters or ICN for layers with $Q_y < 8$ or $Q_w < 8$. On the contrary, MixQ-PC-ICN indicates integer-only models with per-channel quantization and ICN as activation layers. Every curve represents a group of Mobilenet models with same input resolution. Increasing the width multiplier causes a longer latency because of the increasing amount of MAC operations. When applying our mixed-precision method under this memory constraints, Mobilenet models with width multipliers of 0.25 and 0.5, with the exception of 224_0.5, features no cuts of bit precision. Hence, under the configuration MixQ-PL, these points corresponds to the 8 bit integer-only models described in (Jacob et al., 2018).

Figure 4 details the individual tensors bit-precision for larger MobilenetV1 models after applying memory-driven mixed-precision quantization (Algorithms 1 and 2) on both MixQ-PL and MixQ-PC-ICN configurations. Models with higher number of parameters or activations are more affected by the bit reduction procedure. Typically, first layers feature large spatial maps but weight tensors with low number of parameters. On the contrary, last layers feature small activation tensors, but high number of weight parameters, with the exception of the depthwise layers.

Pareto frontiers of Figure 3 are mostly populated by MixQ-PC-ICN configurations. The most accurate model, PC+ICN 224_0.75, scores 68% Top1 accuracy by featuring 4 bit weight on the last convolutional pointwise and on the linear layers, in addition to $Q_y^1, Q_y^2, Q_y^5 = 4$, as determined by the memory-driven procedure of Section 5. This score is 8% higher than the more accurate INT8 Mobilenet (192_0.5) fitting into the same device. Note that all the configurations featuring width multiplier 1.0 suffers of a dramatic accuracy degradation with respect to full-precision settings (from 2% to 15%) due to aggressive quantization required to fit into the memory constrains. On the latency side, the fastest inference model (128_0.25 MixQ-PL), which features a homogeneous 8 bit quantization, runs at 10fps, 20× higher than the the most precise configuration (224_0.75 PC+ICN), but only achieves 43% of Top1 accuracy. We can observe that the MixQ-PC-ICN quantization introduces a latency overhead of approx. 20% with respect to the MixQ-PL setting, due to the additional subtractions of Z_w biases within the inner loop of the convolution. On the other hand, MixQ-PC-ICN provides up to 4% more accuracy for classification.

To further test our proposed mixed-precision method, we set the memory constrain to $\mathrm{M_{RO}} = 1MB$ and compare with other mixed-precision methodologies in Table 3. Our best models feature up to 7% lower accuracy with respect to (Wang et al., 2018), but, in contrast with this and similar works, we remark that we only use integer operations also

Table 3. Comparison with state-of-the-art mixed precision models when M_{RO} is 1MB

Model	Quantization Method	Top1 Accuracy	Memory
MobilenetV1_224_0.5	MixQ-PC-ICN	62.9%	1MB M_{RO} + 512kB M_{RW}
MobilenetV1_192_0.5	MixQ-PC-ICN	60.2%	1MB M_{RO} + 256kB M_{RW}
MobilenetV1_224_0.5 (Jacob et al., 2018)	INT8 PL+FB	60.7%	1.34 MB
MobilenetV1_224_0.25 (Jacob et al., 2018)	INT8 PL+FB	48.0%	0.47 MB
MobilenetV1 (Wang et al., 2018)	MIX *not-uniform*	57.14% / 67.66%	1.09 / 1.58 MB
MobileNetV2 (Wang et al., 2018)	MIX *not-uniform*	66.75% / 70.90%	0.95 / 1.38 MB
SqueezeNext (Dong et al., 2019)	MIX *not-uniform*	68.02%	1.09 MB

thanks to the exploited uniform quantization. Moreover, our solution features a 2% higher accuracy than INT8 models with comparable memory footprint and tailored for integer-only deployments.

7 Conclusion

By mixing quantization methodologies, it is possible to execute complex deep neural networks such as MobilenetV1 on memory constrained MCU edge devices. To pursue this objective, in this work we introduced a mixed-precision quantization technique tailored for memory-constrained microcontroller devices, leveraging the formulation of a quantized activation layer, i.e. the Integer Channel-Normalization activation, to enable sub byte integer-only deployments. The experimental results show a MobilenetV1 network running on a microcontroller equipped with 2MB of Flash and 512kB of RAM and featuring a Top1 accuracy of 68%, which is 8% higher than state-of-the-art integer-only 8 bit implementations fitting the same memory constraints.

References

Blott, M., Preußer, T. B., Fraser, N. J., Gambardella, G., Obrien, K., Umuroglu, Y., Leeser, M., and Vissers, K. Finn-r: An end-to-end deep-learning framework for fast exploration of quantized neural networks. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 11(3):16, 2018.

Choi, J., Wang, Z., Venkataramani, S., Chuang, P. I.-J., Srinivasan, V., and Gopalakrishnan, K. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

Choukroun, Y., Kravchik, E., and Kisilev, P. Low-bit quantization of neural networks for efficient inference. *arXiv preprint arXiv:1902.06822*, 2019.

Courbariaux, M., Hubara, I., Soudry, D., El-Yaniv, R., and Bengio, Y. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Dong, Z., Yao, Z., Gholami, A., Mahoney, M., and Keutzer, K. Hawq: Hessian aware quantization of neural networks with mixed-precision. *arXiv preprint arXiv:1905.03696*, 2019.

Esser, S. K., McKinstry, J. L., Bablani, D., Appuswamy, R., and Modha, D. S. Learned step size quantization. *arXiv preprint arXiv:1902.08153*, 2019.

Fromm, J., Patel, S., and Philipose, M. Heterogeneous bitwidth binarization in convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 4006–4015, 2018.

Gao, H., Tao, W., Wen, D., Chen, T.-W., Osa, K., and Kato, M. Ifq-net: Integrated fixed-point quantization networks for embedded vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 607–615, 2018.

Han, S., Mao, H., and Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

Jain, S. R., Gural, A., Wu, M., and Dick, C. Trained uniform quantization for accurate and efficient neural network inference on fixed-point hardware. *arXiv preprint arXiv:1903.08066*, 2019.

Krishnamoorthi, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.

Lai, L., Suda, N., and Chandra, V. Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus. *arXiv preprint arXiv:1801.06601*, 2018.

Lin, D., Talathi, S., and Annapureddy, S. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pp. 2849–2858, 2016.

Liu, Z.-G. and Mattina, M. Learning low-precision neural networks without straight-through estimator (ste). *arXiv preprint arXiv:1903.01061*, 2019.

Ma, N., Zhang, X., Zheng, H.-T., and Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 116–131, 2018.

Migacz, S. 8-bit inference with tensorrt. In *GPU Technology Conference*, volume 2, pp. 7, 2017.

Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pp. 525–542. Springer, 2016.

Rusci, M., Capotondi, A., Conti, F., and Benini, L. Work-in-progress: Quantized nns as the definitive solution for inference on low-power arm mcus? In *2018 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*, pp. 1–2. IEEE, 2018.

Umuroglu, Y. and Jahre, M. Streamlined deployment for quantized neural networks. *arXiv preprint arXiv:1709.04060*, 2017.

Wang, K., Liu, Z., Lin, Y., Lin, J., and Han, S. Haq: hardware-aware automated quantization. *arXiv preprint arXiv:1811.08886*, 2018.

Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. *arXiv preprint arXiv:1812.03443*, 2018.

Zhang, D., Yang, J., Ye, D., and Hua, G. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 365–382, 2018.

Zhang, Y., Suda, N., Lai, L., and Chandra, V. Hello edge: Keyword spotting on microcontrollers. *arXiv preprint arXiv:1711.07128*, 2017.