ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

Work-in-Progress: Quantized NNs as the Definitive solution for inference on low-power ARM MCUs?

This is the final peer-reviewed author’s accepted manuscript (postprint) of the following publication:

Published Version:
Work-in-Progress: Quantized NNs as the Definitive solution for inference on low-power ARM MCUs? / Rusci, Manuele; Capotondi, Alessandro; Conti, Francesco; Benini, Luca. - ELETTRONICO. - (2018), pp. 8525915.1-8525915.2. (Intervento presentato al convegno 2018 ACM/IEEE International Conference on Hardware/Software Codesign and System Synthesis CODES+ISSS 2018 tenutosi a Torino Incontra Congress Center, Torino, Italia nel September 30 – October 5, 2018) [10.1109/CODESISSS.2018.8525915].

Availability:
This version is available at: https://hdl.handle.net/11585/652922 since: 2018-12-19

Published:
DOI: http://doi.org/10.1109/CODESISSS.2018.8525915

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

25 April 2024

This is the post peer-review accepted manuscript of:

M. Rusci, A. Capotondi, F. Conti and L. Benini, "Work-in-Progress: Quantized NNs as the Definitive Solution for Inference on Low-Power ARM MCUs?", in Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis, Turin, Italy — September 30 - October 05, 2018.
doi:10.1109/CODESISSS.2018.8525915

The published version is available online at: https://ieeexplore.ieee.org/abstract/document/8525915

© 2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works

Work-in-Progress: Quantized NNs as the Definitive Solution for Inference on Low-Power ARM MCUs?

Manuele Rusci
DEI, University of Bologna
manuele.rusci@unibo.it

Alessandro Capotondi
DEI, University of Bologna
alessandro.capotondi@unibo.it

Francesco Conti
DEI, University of Bologna
D-ITET, ETH Zurich
f.conti@unibo.it

Luca Benini
DEI, University of Bologna
D-ITET, ETH Zurich
luca.benini@unibo.it

***Abstract*—High energy efficiency and low memory footprint are the key requirements for the deployment of deep learning based analytics on low-power microcontrollers. Here we present work-in-progress results with Q-bit Quantized Neural Networks (QNNs) deployed on a commercial Cortex-M7 class microcontroller by means of an extension to the ARM CMSIS-NN library. We show that *i)* for $Q = 4$ and $Q = 2$ low memory footprint QNNs can be deployed with an energy overhead of 30% and 36% respectively against the 8-bit CMSIS-NN due to the lack of quantization support in the ISA; *ii)* for $Q = 1$ native instructions can be used, yielding an energy and latency reduction of ~3.8× with respect to CMSIS-NN. Our initial results suggest that a small set of QNN-related specialized instructions could improve performance by as much as 7.5× for $Q = 4$, 13.6× for $Q = 2$ and 6.5× for binary NNs.**

***Index Terms*—Machine Learning, Quantized Neural Network, ARM Microcontrollers**

I. Introduction

Energy efficiency is key for enabling Deep Learning (DL) based data analytics performed directly at the edge in unobtrusive, low-power devices such as Micro-Controller Units (MCUs). However, commercial MCUs are typically limited in terms of computational capabilities and onboard memory, making aggressive software and algorithmic optimizations absolutely necessary if one wants to deploy popular DL algorithms such as deep convolutional neural networks (CNNs). To tackle the first requirement, ARM recently proposed the optimized CMSIS-NN library [1], which maximizes performance and energy efficiency of common DL kernels on top of Cortex-M series cores. This library, however, only supports 16-bit and 8-bit fixed-point data, which means that even relatively small CNN topologies may exceed on-chip memory.

A recent trend to reduce the footprint of CNNs is the aggressive quantization of parameters weights and/or activations with a precision lower than 8 bits. Competitive end-to-end accuracy has been demonstrated on several public datasets even using purely binary weights and activations [2]. However, despite the memory savings of binary NNs, it is still under active debate whether they represent the best energy-accuracy tradeoff, as 2- and 4-bit quantized NNs have demonstrated superior accuracy with comparable energy cost in other cases, including complex visual classification and detection problems [3], [4].

This work presents the deployment and evaluation on a ARM Cortex-M7 of low-precision convolutional kernels using 4, 2 or a single bit to represent data, defining the INT-4/2/1 formats accordingly (Sec. II). To this end, we extended the ARM CMSIS-NN library and ran our experiments on an STM32-H743 MCU. Our initial results indicate that: *i)* despite the reduction in memory bandwidth requirements, the INT-4/2 convolutional kernels suffer significant runtime overhead with respect to the hardware-supported 8-bit quantization, which could be removed only if the ISA supported efficient sub-byte scalar product operations; *ii)* the INT-1 kernel shows a 3.8× higher energy efficiency thanks to the reduced amount of memory accesses and the *bitwise* operations featured by the CPU datapath. Still, efficient *popcount* support at the ISA level would lead to significant speed-ups in binary network inference; this is already available in ARM Cortex-A class cores (VCNT instruction) and has been shown also in the context of low-power MCU-class cores [5].

II. INT-Q format & QNN implementation

The approach of Hubara et al. [2] generalizes the quantization of real-valued weights/activations w to Q-bit signed integers by means of the following function q:

$$q(w) = \mathrm{clip}_{[-1,1)}\big(2^{-(Q-1)} \cdot \mathrm{round}(w \cdot 2^{Q-1})\big), \qquad (1)$$

where $\mathrm{clip}_{[a,b)}(x) = \max(a, \min(x, b))$. $q(w)$ is a fixed-point fractionary number; we define the integer $W = q(w) \cdot 2^{Q-1}$ as the INT-Q representation of w. If both weights and activation inputs are INT-Q values, the convolution becomes an integer sum-of-products operation:

$$\varphi(w, x) = 2^{-2(Q-1)} \sum_{i \in C} W_i X_i \doteq 2^{-2(Q-1)}\, \Phi(W, X) \qquad (2)$$

where C is the set of input channels, φ indicates the convolution operation. The partial results for the accumulation of $\Phi(W, X)$ feature a higher precision, INT-16 for the purpose of this work.

We focus on the case of $Q = 1, 2, 4$ bits as these are the most natural sub-byte data types to fit within the 32-bit registers of a common MCU. Similarly to the CMSIS-NN, we developed a INT-Q convolutional kernel which includes an *Unpack* function to *i)* extract INT-Q parameters and cast them to an INT-16 representation and *ii)* reshape a receptive field tile into an element array. The convolution is then implemented as a matrix multiplication, which leverages the 2xINT-16 MAC instruction supported by the ARMv7-M ISA DSP extention (SMALD). Compared with the baseline CMSIS-NN scenario, the extraction of 2-,4-bit parameters presents nearly a 2.5× higher number of operations.

TABLE I
SIZE OF 256×128×3×3 CONV ON 128×16×16 INPUT

Datatype	Weights	Activations	Thresholds	Total
FIXED-8	288 kB	96 kB	0.25 kB	384.25 kB
INT-4	144 kB	48 kB	4 kB	196 kB
INT-2	72 kB	24 kB	1 kB	97 kB
INT-1	36 kB	12 kB	0.5 kB	48.5 kB

Source code is available at https://github.com/EEESlab/CMSIS_NN-INTQ. This project was supported in part by the EU's H2020 programme under grant no. 732631 (OPRECOMP).

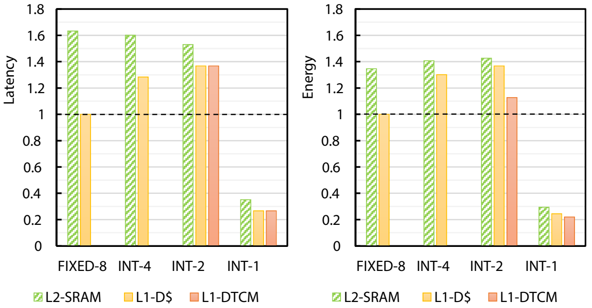

Fig. 1. Normalized latency and energy consumption of 2D spatial convolution benchmark on STM32-H743.

Output values, which stores the accumulation of partial results (INT-16), have to be compressed back into Q bits using a staircase function that generalizes (1):

$$Y = q\big(\varphi(x)\big) = \sum_{p=-2^{Q-1}}^{2^{Q-1}-1} p \cdot \chi_{[\tau_p,\tau_{p+1})}\big(\Phi(W,X)\big) \tag{3}$$

where $\chi_s(\cdot)$ is the characteristic function of the interval s. $\Phi(W,X)$ and the τ thresholds are INT-16 and hence neither (2) nor (3) need any floating point operation[1]– the whole layer can be executed entirely in the integer domain. The staircase function is optimally implemented by a balanced binary tree where an INT-16 comparison takes place at every node.

As a special case, INT-1 binarization can be redefined equivalently as $W = \text{sign}(w)$; binary convolution reduces to:

$$\Phi_{bin}(X) = \text{popcount}(W \text{ xnor } X) \tag{4}$$

Contrary to the general INT-Q case, the xnor operation can be performed bitwise with no need of casting to INT-16. The accumulation, performed by popcount, still requires INT-16 precision. The final binarization, as in the INT-Q case, is performed by comparing $\Phi_{bin}(X)$ with an INT-16 threshold [6].

III. EXPERIMENTAL RESULTS & DISCUSSION

To evaluate our quantized extension to CMSIS-NN, we profiled the execution of a 256×128×3×3 convolutional layer on a 128×16×16 input tensor, sweeping all the INT-Q datatypes supported and the CMSIS-NN baseline (FIXED-8). We used a high-performance STM32-H743 MCU, equipped with an ARM Cortex-M7 running at 384 MHz, 16 kB of L1 data cache (which can be deactivated), 128 kB of L1 scratchpad called Data Tightly Coupled Memory (L1-DTCM), 512 kB of SRAM memory. Table I shows the memory footprint for weights, activations and thresholds for FIXED-8 and INT-4/2/1; only INT-2/1 fit in L1-DTCM.

Figure 1 reports latency and energy consumption using SRAM with no cache (L2-SRAM), with active cache (L1-D$) and, when the problem fits it, using the manually managed scratchpad (L1-DTCM). All results are normalized with respect to FIXED-8 with L1-D$. Naturally, the lower bandwidth required when the precision is dropped results in lower impact from memory latency in the L2-SRAM case. Execution time, however, does not drop linearly because of the additional instructions for the INT-4/2 to INT-16 casting. The full impact of this effect is visible in the L1-D$ case, where memory latency is hidden. This results in a latency increase of 28% and 36% for INT-4 and INT-2, respectively. Conversely, the INT-1 kernel shows a ∼3.8× lower execution time by exploiting the bitwise operations supported by the ISA. Similar results to the L1-D$ can be seen in the L1-DTCM case, when the full problem fits in. For what concerns the energy consumption, INT-4 and INT-2 cases are again less optimal than FIXED-8 (+30% and +36%, respectively), due to the extra-costs of data extraction and casting. Using L1-DTCM instead of L1-D$ provides a marginal energy advantage.

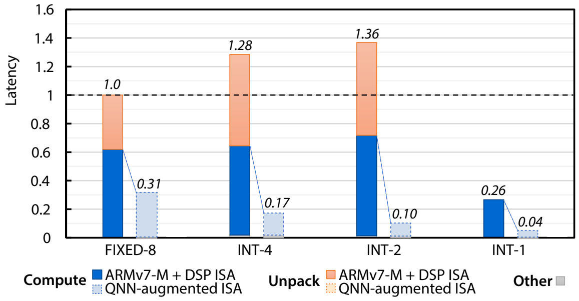

Fig. 2. Breakdown of contributions to latency in the L1-D$ case and projection to a QNN-augmented ISA.

The breakdown of the contributions to latency for the L1-D$ case is reported in Figure 2. Except for the INT-1 case, the overall workload is evenly split between *Unpack* operations and *Compute* 2xINT-16 SIMD instructions, as supported by the ARMv7-M ISA. The INT-1 does not have *Unpack* overheads because XNOR convolutions are supported by the current ISA. However the popcount is software-emulated, leaving further space for improvements.

Our estimates show that the performance of low-precision kernels can be greatly enhanced by extending the ISA with QNN-oriented instructions, on the line of what is done for INT-16 – this would fully eliminate *Unpack* latency, as well as compress *Compute* time thanks to SIMD vectorization. We estimate the latency gains in case of a *QNN-augmented ISA* supporting vectorized MAC instruction for low precision data types INT-8/4/2/1. The analysis reveals potential improvements up to 3.2× (FIXED-8), 7.5× (INT-4), 13.6× (INT-2), 6.5× (INT-1) in terms of execution time, enabling highly desirable downscaling of both execution time and memory footprint together with data type precision.

REFERENCES

[1] L. Lai *et al.* Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus. *arXiv:1801.06601*, 2018.

[2] I. Hubara *et al.* Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv:1609.07061*, 2016.

[3] B. Moons *et al.* Minimum energy quantized neural networks. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pages 1921–1925, Oct 2017.

[4] T. B. Preußer *et al.* Inference of quantized neural networks on heterogeneous all-programmable devices. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 833–838, March 2018.

[5] M. Gautschi *et al.* Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(10):2700–2713, October 2017.

[6] M. Rusci *et al.* Design automation for binarized neural networks: A quantum leap opportunity? In *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018.

---

[1] The τ_p thresholds absorb bias, batch normalization, and the $2^{-2(Q-1)}$ factor coming from $\varphi = 2^{-2(Q-1)}\Phi$. Specifically, considering the batch-normalized $y = \gamma/\sigma(b + \varphi - \mu) + \beta)$ (where b is the bias, $\gamma, \beta, \sigma, \mu$ are the batch normalization parameters), the thresholds are

$$\tau_p = \left\lceil 2^{Q-1}\left(p \cdot \sigma/\gamma - 2^{Q-1} \cdot (b - \mu) + \beta \cdot \sigma/\gamma\right)\right\rceil$$