CoopNet: Cooperative convolutional neural network for low-power MCUs

# CoopNet: Cooperative Convolutional Neural Network for Low-Power MCUs

Luca Mocerino, Andrea Calimera
Politecnico di Torino, 10129, Torino Italy

***Abstract*—Fixed-point quantization and binarization are two reduction methods adopted to deploy Convolutional Neural Networks (CNNs) on end-nodes powered by low-power micro-controller units (MCUs). While most of the existing works use them as stand-alone optimizations, this work aims at demonstrating there is margin for a joint cooperation that leads to inferential engines with lower latency and higher accuracy. Called *CoopNet*, the proposed heterogeneous model is conceived, implemented and tested on off-the-shelf MCUs with small on-chip memory and few computational resources. Experimental results conducted on three different CNNs using as test-bench the low-power RISC core of the Cortex-M family by ARM validate the CoopNet proposal by showing substantial improvements w.r.t. designs where quantization and binarization are applied separately.**

## I. Introduction

Inference engines built upon end-to-end deep learning methods represent the state-of-the-art in several application domains. Deep Convolutional Neural Networks (CNNs), in particular, have brought about breakthroughs in the field of computer vision, speech recognition and natural language processing [1]. Many Internet-of-Things (IoT) services rely on CNNs to infer information from the raw data gathered by end-user portable devices and/or embedded sensors. While the majority of IoT frameworks run CNNs in the cloud, namely, on centralized data centers physically very far from the source of data, to have CNNs on hand is a means to higher efficiency and more user privacy [2]. Enabling the inferential stage on the mobile edge is challenging as it requires the processing of CNNs, large in size and computationally intensive, with limited hardware resources. The picture gets even more complicated when considering applications, like wearable [1] or ambient and infrastructural sensors [3], which must run on tiny cores with few hundreds of kByte of on-chip memory and an active power consumption below the 100 mW mark. As practical example, this work considers the micro-controller units (MCUs) of the Cortex-M family designed by ARM for the IoT segment[1]. In such cases, the only available option is to shrink down the cardinality of the CNN model until it fits the underlying hardware architecture.

Among the available algorithmic optimizations, post-training quantization via integer arithmetic has become a must-do stage: most of the MCU cores deployed on the end-nodes do not have floating-point units indeed. The use of arithmetic representations with scaled bit-widths helps to reduce the memory footprint, but above all it ensures a larger memory bandwidth as multiple data can be packed within the same word. This ensures lower latency and hence smaller energy consumption w.r.t 32-bit floating point. In [4] the authors demonstrate that 8-bit fixed-point integer guarantees near-to-zero accuracy loss with 4× memory reduction. Extreme quantization to 1-bit [5], [6], [7] leads to Binary CNNs with the smallest footprint, but also the lightest workload as (some) integer arithmetic get replaced with bit-wise Boolean operators. However, binary CNNs come with significant accuracy loss: from 2%, up to 10%, 20%, and even more, depending on the original CNN and the complexity of the training data-set. This represents a key limiting factor.

This work aims to address this drawback demonstrating there exist margins to exploit binary CNNs for building highly accurate, yet fast inferential models that can be deployed on the edge. The proposed solution, called *Cooperative Convolutional Neural Network* (CoopNet), consists of a joint combination of binary and 8-bit fixed-point CNN models controlled by a probabilistic thresholding policy. The resulting heterogeneous inferential model improves the classification accuracy and meets the hardware constraints of low-power MCUs. Experimental results, conducted on three CNNs trained to run classification on three data-set belonging to different domains, reveal CoopNet deployed on the Cortex-M cores by ARM outperforms classical homogeneous CNN, therefore achieving an improved accuracy-latency tradeoff.

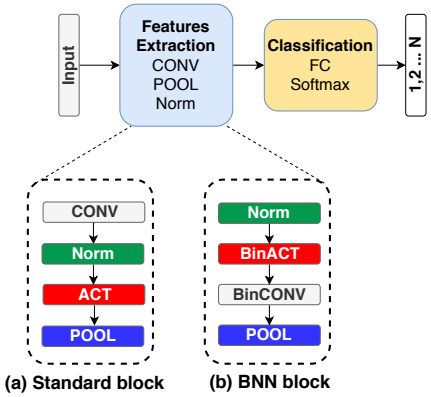


Figure 1. Structure of a typical CNN. (a) Basic block of a standard Integer CNN; (b) Basic block of a BNN with binarized weights and activations

## II. Background

### A. Convolutional Neural Networks (CNNs)

CNNs are a special class of end-to-end trainable models mostly suited for the classification of multi-dimensional spatial inputs, like multi-channel images. They consist of several computational layers chained to form a deep architecture. Existing CNNs mainly differ for their internal topology, namely, how different kinds of hidden layers are sized and connected. It is however possible to recognize a common structure which is made up of two macroblocks (Fig. 1): *Feature Extraction*, where relevant features learned during the training stage are

[1]https://os.mbed.com/platforms

extracted layer-wise using kernel convolutions; *Classification*, where the extracted features get classified.
Within the feature extraction block, the most commonly adopted layers are: *convolutional* layers (CONV), which perform multidimensional convolutions between the output tensor generated by the previous layer (also called feature map) and local filter tensors; *pooling* layers (POOL), e.g. max pooling or average pooling, which reduce the dimension of feature maps; *normalization* layers (Norm), that normalize the distribution (mean and standard deviation) of the activation maps; *activation* function (ACT), e.g. ReLU or tanh, which introduces non-linearity. The classification block is built upon *fully-connected* layers (FC), which implement a geometric separation of the extracted features, and *softmax*, that produces a probability distribution over the available classes.

### B. Fixed-Point Quantization

While a CNN training is usually run using a 32-bits floating-point representation, recent studies, e.g. [4], [8], demonstrate that fixed-point integers with lower arithmetic precision are enough for inference. Fixed-point quantization is becoming a consolidated standard when the target hardware are low-power cores with small memory footprint and reduced instruction set (8/16-bit integer). A detailed review of all the quantization schemes in literature is out of the scope of this work and interested readers may refer to [4], [8], [9]. This work adopts the $q$-bit fixed-point quantization proposed in [8]. The convolution run in a CONV layer between the input feature map $\mathbf{x} \in \mathbb{R}^{c \times w_{in} \times h_{in}}$ and the local weights $\mathbf{w} \in \mathbb{R}^{c \times kw \times kh}$ is as follows:

$$\mathbf{x} * \mathbf{w} = 2^{-2(q-1)} \sum_{i \in C} X_i \cdot W_i \tag{1}$$

with $C$ as the number of channels. We set $q = 8$ for both weights and activations, and $q = 16$ for intermediate results accumulation.

### C. Binarized Neural Networks

Several works proposed CNNs with binary weights and/or activations. BinaryConnect [5] represents the ancestor: weights are binarized using *hard sigmoid* function, while activations remain in full-precision to avoid accuracy drop. The Binarized Neural Networks proposed in [6] are the first example of fully binary CNN: both weights and activations are binarized via *sign* function. The CONV layers are simplified through bit-wise XNOR and bit-count. This allows to achieve the highest compression ($\sim 32\times$), yet with substantial accuracy loss (up to 28.7%). The authors of XNOR-Net [7] addressed this drawback introducing a new topology where the binary output of each CONV layer is first re-scaled through a full-precision Norm layer. Fig. 1-b gives a pictorial description of the basic block deployed in the XNOR-Net, where the suffix *Bin* highlights binarized layers.
The mathematical description of a binary convolution is as follows. Given $\mathbf{x} \in \mathbb{R}^{c \times w_{in} \times h_{in}}$ as the input feature and $\mathbf{w} \in \mathbb{R}^{c \times kw \times kh}$ as the weight tensor, their convolution is approximated as follows:

$$\mathbf{x} * \mathbf{w} \approx \text{popcount}\ (\mathbf{X}\ \text{xnor}\ \mathbf{W}) \cdot K \cdot \alpha \tag{2}$$

where $K$ and $\alpha$ are scaling factors. While weights ($\mathbf{W}$) are binarized with the *sign* function only, the activations are first normalized and then binarized. These stages can be fused into a single layer that includes all batch normalization parameters: variance $\sigma^2$, mean $\mu$, scale $\gamma$, shift factor $\beta$, and $\epsilon$ for numerical stability. A feature map $\mathbf{x}$ is binarized as follows:

$$\mathbf{X} = BinACT_{0,1}(x) = \begin{cases} 1 & x \geq c \\ 0 & x < c \end{cases} \tag{3}$$

where $c = \mu - \beta/\gamma\sqrt{\sigma^2 + \epsilon}$ is constant at inference time. We do not use $K$, that is the activations scaling factor, for computational efficiency reasons. We represent $c$ and $\alpha$ with 8-bit integers.
An efficient processing of XNOR-Net requires data-paths capable of performing bit-wise xnor, bit-counter and comparison. These operators can be implemented with specialized units in case of custom hardware [10], or through software routines compiled using the instruction-set available on the target general purpose core [11].

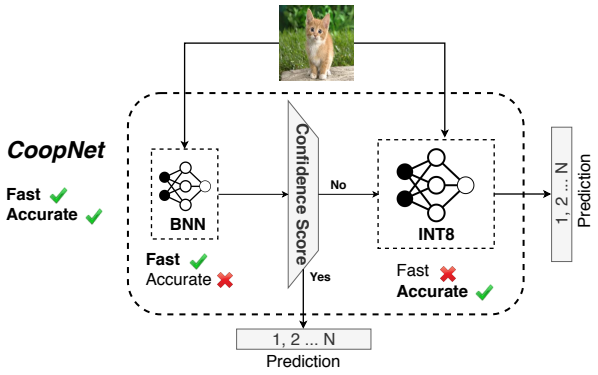


Figure 2. CoopNet concept and abstract architecture.

## III. CoopNet

### A. Concept and Architecture

The CoopNet inference concept is intuitive, yet very efficient. As graphically depicted in Fig. 2, it is based on the cooperation of two convolutional models: a binary net BNN, fast and small but less accurate, an integer net INT8, slower and larger but more accurate. The BNN processes the input data first. Then, if the prediction satisfies a certain criterion of confidence it is forwarded to the output as the outcome, otherwise the input is re-processed by the INT8 to produce a more confident output score. The criterion used to control the execution flow is called *Confidence Score (CS)* and it is defined as follows:

$$CS = P_{BNN}(y_i|x) - P_{BNN}(y_j|x) \tag{4}$$

where $P_{BNN}(y_n|x)$ is the probability produced by the BNN that a given input $x$ belongs to a given class $n \in 1, 2 \ldots N$; $i$ and $j$ refer to the indexes of the first and second highest scored classes. Intuitively, a high *CS* means the BNN was able to classify the given input with enough confidence, on the contrary, a low *CS* means the topmost scored classes get very close to each other, which reveals a certain level of uncertainty, as the BNN was not able to make a clear distinction among the available classes. For the latter case, the INT8 model is activated for aid. The thresholding policy is controlled through a *Confidence Threshold (CT)* which might be changed dynamically for run-time adjustments.

For a given task and application, the pre-trained 32-bit Floating Point model is used as basis to generate the INT8 model, obtained with the quantization method introduced in [8] using $q$=8-bit. The BNN model is built using the XNOR-Net method presented in [7]. It is worth emphasizing that, according to [7], the first and last layers of the BNN model are kept to 8-bit. A key design aspect concerns the setting of the threshold $CT$ as it affects the accuracy-latency trade-off. The parametric analysis reported in the experimental section provides a proper understanding of this important relationship.

### B. Extra-functional Metrics

**Latency.** Given a generic CoopNet, its latency is modeled through the following equation:

$$L_{CoopNet}(CT) = \begin{cases} L_{BNN} + L_{CS} & CS \geq CT \\ L_{BNN} + L_{CS} + L_{INT8} & CS < CT \end{cases} \quad (5)$$

$L_{BNN}$ and $L_{INT8}$ are the latency of the BNN and INT8 models respectively, while $L_{CS}$ is the contribution due to $CS$ computation and comparison with $CT$. To notice that $L_{CS}$ is the latency of a single integer subtraction and comparison, hence its contribution is negligible w.r.t. the latency of BNN and INT. Both $L_{BNN}$ and $L_{INT8}$ can be simply estimated as the sum of the latency of each internal layer collected from on-board measurements. The layers characterization has been implemented using an extended version of the CMSIS-NN library by ARM [12] which supports binary convolutions [11]. When considering batch inference, Equation 5 can be generalized as:

$$L_{CoopNet}(BS, CT) = \sum_{i=1}^{BS} L_i(CT) \quad (6)$$

where *BS* is the cardinality of the batch and $L_i(CT)$ is the latency of the $i$-th batch sample.

**On-chip Memory.** The hardware cores targeted by this work are the smallest low-power MCUs equipped of the Cortex-M family by ARM. These MCUs are usually equipped with limited RAM ($\leq$ 1 MByte). The memory footprint of CoopNet is the sum of the RAM taken by the BNN model ($M_{bnn}$) and INT8 model ($M_{int}$). The two contributions include the RAM taken by the weights buffer, the activations buffer and *im2col* buffer as the model provided by ARM in [12]. The $CT$ parameter is one Byte, therefore negligible.

## IV. Experimental Results

### A. Experimental Setup

CoopNet has been evaluated on the following three tasks:
**CIFAR-10 -** Image classification task; it consists of 60k $32 \times 32$ RGB images classified with 10 labels.
**Google Speech Command (GSC) -** Keyword spotting from speech; the data set [13] collects 65k one-second long samples classified with 30 classes.
**Facial Expression Recognition (FER13) -** Emotion recognition from facial expression; the data set [14] is made up of 36k $48 \times 48$ grayscale facial images classified by 7 labels.
Different lightweight CNNs suited for tiny cores are deployed for the three tasks. An overview is reported in Table I; for sake of space, the table reports the CONV and FC layers together with their size, although there are activation, pooling and regularization (normalization and dropout) layers. Moreover, Table I shows the top-1 accuracy (%) and the memory footprint (kB) for full-precision (FP32), 8-bit fixed-point (INT8) and binary (BNN) models.

| | **Dataset** | CIFAR-10 | GSC | FER13 |
|---|---|---|---|---|
| | **Model** | CaffeNet[2] | GscNet [13] | FerNet |
| **FP32** | Accuracy (%) | 80.25 | 90.30 | 65.16 |
| | Mem. Size (kB) | 550 | 1060 | 2345 |
| **INT8** | Accuracy (%) | 80.20 | 89.50 | 64.70 |
| | Mem. Size (kB) | 120 | 250 | 577 |
| **BNN** | Accuracy (%) | 76.52 | 87.60 | 62.86 |
| | Mem. Size (kB) | 94 | 90 | 118 |
| | input | 3x32x32<br>CONV 32x5x5<br>32x5x5<br>64x5x5<br>FC 1024x10 | 1x32x32<br>CONV 32x5x5<br>32x5x5<br>64x5x5<br>64x5x5<br>FC 1024x31 | 1x44x44<br>3× CONV 32x3x3<br>3× CONV 64x3x3<br>3× CONV 128x3x3<br>FC 128x7 |

Table I
Benchmarks: Datasets and CNNs

### B. Performance Assessment

The conducted experiments aimed at assessing the latency-accuracy trade-off. With this purpose, we first provide a parametric analysis that leverages the confidential threshold $CT$ as main knob. The line plot in Fig. 3 shows the delta accuracy achieved by CoopNet using as ground the accuracy of the baseline model INT8. The three tasks show the same trend: the CoopNet gets more accurate (positive delta) for larger values of $CT$. The break-even point $CT_{be}$ (for which delta is 0) may change depending on the complexity of the data-set and the classification capability of the CNN adopted: $CT_{be} = 0.2$ for FER and GSC, $CT_{be} = 0.4$ for CIFAR-10. To notice that the use of a confidential threshold $CT > CT_{be}$ guarantees substantial accuracy improvement. This suggests that CoopNet does not just improve over standard binarized CNNs, but it can also go beyond 8-bit quantization.

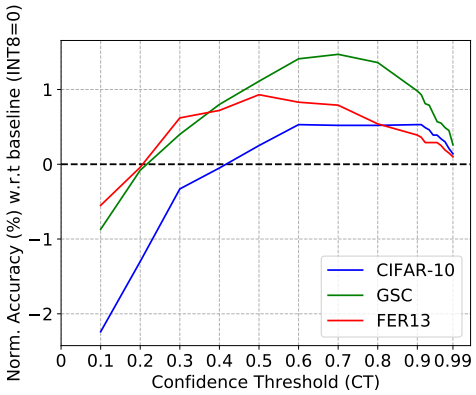


Figure 3. Normalized Accuracy (%) w.r.t the INT8 baseline vs Confidence Threshold (CT)

Even more interesting is the gain in terms of latency. Fig. 4 shows the delta accuracy w.r.t. the baseline (INT8) as function of the average speed-up measured over the test set. The colored bullets drawn over the lines correspond to the actual value of $CT \in [0, 1)$; to notice that bullets size is inversely proportional to the $CT$ value adopted: the smaller the $CT$, the larger the

[2]Inspired by https://code.google.com/archive/p/cuda-convnet/

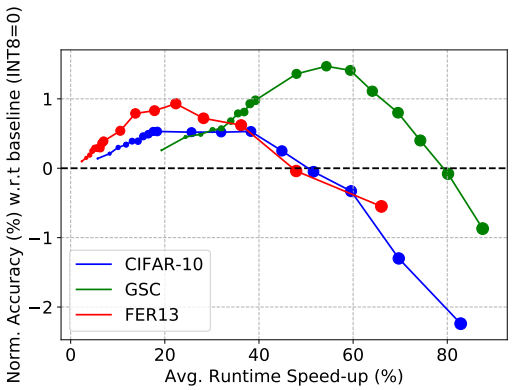


Figure 4. Normalized Accuracy (%) and Average Speed-up w.r.t baseline INT8 on test-set

speed-up. Indeed, a large $CT$ implies that the BNN results get accepted as gold even if highly uncertain, hence the INT8 computation is skipped for speed-up. At the break-even, i.e. $CT = CT_{be}$, CoopNet shows impressive performance boost: 47.90% for FER, 51.58% for CIFAR-10, 80.16% for GSC.

Table II gives a summary of some key results achieved by CoopNet. More specifically, it shows the evaluated extra-functional metrics (Speed-up and RAM footprint) under three accuracy level scenarios: $(i)$ CoopNet meets the accuracy of FP32, $(ii)$ CoopNet meets the accuracy of INT8 (the ground, i.e. $\Delta = 0$), $(iii)$ CoopNet with the highest accuracy. As one can see, CoopNet guarantees substantial gains even under very high accuracy constraints. For instance, for GSC it achieves the same accuracy of the FP32 model with an average speed-up of 69.53%; more interesting CoopNet can even overtake the FP32 model (+1.47%). We observed that the joint action of BNN and INT8 helps to recognize inputs for which the FP32 model fails.

## V. Related works

Fixed-point quantization [4], as well as binarization [6], [5], [7], represent a valuable solution to deploy CNN on ultra-low-power commercial MCUs. While 8-bit are almost sufficient to guarantee the same accuracy of floating-point models, lower bit-widths lead to inaccurate inference. Recent works investigated on arbitrary bit-widths (i.e. $2 \leq bit < 8$). More specifically, they aim at finding the optimal balance between accuracy, resource utilization and performances assigning different bit-widths to different layers [9]. Unfortunately, commercial low-power MCUs do not have programmable data-paths and memory interfaces to support arbitrary bit-width arithmetic efficiently [11]. Amiri et al. in [15] proposed a system level mixed-precision solution which exploits heterogeneous CPU and FPGA accelerators. The overhead, both on-line (due to a tuning procedure) and off-line (during training), and the resources required make this approach less suitable for low-end MCUs. Combining multiple CNN models into *ensemble* results in a winning solution for many tasks [16]. However, the resources required to host several models and execute them in parallel make this approach practically not scalable on a low-end device. On the contrary, CoopNet enables an efficient and accurate solution for off-the-shelf MCUs proposing a flexible architecture adaptable to the user-defined constraint.

| Dataset (Net) | Accuracy Level | Δ (%) | Mem. Size (kB) | Speed-up (%) | ST |
|---|---|---|---|---|---|
| CIFAR-10 (CaffeNet) | FP32 | +0.05 | 214 | 51.20 | 0.42 |
| | INT8 | 0 | | 51.58 | 0.4 |
| | Max-accuracy | +0.53 | | 18.40 | 0.9 |
| GSC (GscNet) | FP32 | +0.8 | 360 | 69.53 | 0.4 |
| | INT8 | 0 | | 80.16 | 0.2 |
| | Max-accuracy | +1.47 | | 54.31 | 0.7 |
| FER13 (FerNet) | FP32 | +0.46 | 695 | 36.20 | 0.3 |
| | INT8 | 0 | | 47.90 | 0.2 |
| | Max-accuracy | +0.93 | | 22.34 | 0.5 |

Table II
CoopNet: Main achievements and Final Results

## VI. Conclusions

CoopNet is a novel network architecture that integrates a fast and unreliable model with a slower but accurate one to improve the processing efficiency of inference models. The joint cooperation of binary and 8-bit quantized models guarantees higher accuracy and substantial speed-up, also offering a valuable option for adaptive energy-accuracy inference on the edge.

## References


[1] J. Gu *et al.*, "Recent advances in convolutional neural networks," *CoRR*, vol. abs/1512.07108, 2015.

[2] W. Shi *et al.*, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[3] M. Fayaz and D. Kim, "A prediction methodology of energy consumption based on deep extreme learning machine and comparative analysis in residential buildings," *Electronics*, vol. 7, no. 10, 2018.

[4] D. D. Lin *et al.*, "Fixed point quantization of deep convolutional networks," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16, 2016, pp. 2849–2858.

[5] M. Courbariaux *et al.*, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, 2015, pp. 3123–3131.

[6] M. Courbariaux *et al.*, "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1," *arXiv preprint arXiv:1602.02830*, 2016.

[7] M. Rastegari *et al.*, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 525–542.

[8] I. Hubara *et al.*, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, pp. 6869–6898, 2017.

[9] A. Zhou *et al.*, "Incremental network quantization: Towards lossless cnns with low-precision weights," *CoRR*, vol. abs/1702.03044, 2017.

[10] Y. Umuroglu *et al.*, "FINN: A framework for fast, scalable binarized neural network inference," *CoRR*, vol. abs/1612.07119, 2016.

[11] M. Rusci *et al.*, "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)*. IEEE, 2018, pp. 1–2.

[12] L. Lai *et al.*, "CMSIS-NN: efficient neural network kernels for arm cortex-m cpus," *CoRR*, vol. abs/1801.06601, 2018.

[13] T. N. Sainath *et al.*, "Convolutional neural networks for small-footprint keyword spotting," in *Interspeech*, 2015.

[14] I. Goodfellow *et al.*, "Challenges in representation learning: A report on three machine learning contests," 2013.

[15] S. Amiri *et al.*, "Multi-precision convolutional neural networks on heterogeneous hardware," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018, pp. 419–424.

[16] R. Minetto *et al.*, "Hydra: an ensemble of convolutional neural networks for geospatial land classification," *ArXiv*, vol. abs/1802.03518, 2018.