Two-stage Human Activity Recognition on Microcontrollers with Decision Trees and CNNs

Two-stage Human Activity Recognition on Microcontrollers with Decision Trees and CNNs

Francesco Daghero
DAUIN, Politecnico di Torino
Turin, Italy
francesco.daghero@polito.it

Daniele Jahier Pagliari
DAUIN, Politecnico di Torino
Turin, Italy
daniele.jahier@polito.it

Massimo Poncino
DAUIN, Politecnico di Torino
Turin, Italy
massimo.poncino@polito.it

***Abstract*—Human Activity Recognition (HAR) has become an increasingly popular task for embedded devices such as smartwatches. Most HAR systems for ultra-low power devices are based on classic Machine Learning (ML) models, whereas Deep Learning (DL), although reaching state-of-the-art accuracy, is less popular due to its high energy consumption, which poses a significant challenge for battery-operated and resource-constrained devices. In this work, we bridge the gap between on-device HAR and DL thanks to a hierarchical architecture composed of a decision tree (DT) and a one dimensional Convolutional Neural Network (1D CNN). The two classifiers operate in a cascaded fashion on two different sub-tasks: the DT classifies only the easiest activities, while the CNN deals with more complex ones. With experiments on a state-of-the-art dataset and targeting a single-core RISC-V MCU, we show that this approach allows to save up to 67.7% energy w.r.t. a "stand-alone" DL architecture at iso-accuracy. Additionally, the two-stage system either introduces a negligible memory overhead (up to 200 B) or on the contrary, reduces the total memory occupation.**

***Index Terms*—Machine Learning, Adaptive Inference, Microcontrollers, Energy Efficiency**

I. Introduction and Related Works

Human Activity Recognition (HAR) based on Inertial Measurements Units (IMUs) has become an increasingly popular feature for smart devices such as fitness trackers and smartwatches. A HAR task consists of determining the activity performed by an user in a specific time-window of the input signal. This is generally performed through Machine Learning (ML) [1], [2] and more recently Deep Learning (DL) [1], [3], with the latter generally achieving state-of-the-art accuracy.

As for many other tasks in the Internet of Things (IoT) world, HAR benefits from an edge-centric approach, that is, from performing the full ML inference on the same device collecting the sensors readings. In this way, there is no need to transmit the data over possibly unstable or unreachable networks, thus making the inference latency more predictable. Additionally, possibly sensitive information don't have to be transmitted, increasing the application security. Finally, avoiding the data transmission brings an additional advantage in terms of energy efficiency: wireless connectivity is in fact power-hungry, generally consuming much more than performing the computations locally. This is particularly beneficial since edge devices are often battery-powered.

However, directly deploying a HAR ML model tailored for the cloud on an embedded device is often difficult. Those devices are in fact typically based on ultra-low-power Microcontrollers (MCUs) with tight memory constraints and relatively low clock frequencies. For this reason, most HAR implementations on embedded devices are based on simple classifiers such as Decision Trees (DTs). These shallow ML models require only a few compare-and-branch operations for inference, and resulting in low latency and energy consumption. Additionally, their memory requirements are also orders of magnitude less than other types of classifiers. However, they rarely reach very high accuracies on complex tasks.

On the other hand, DL approaches have shown promising results for HAR [1]–[3], achieving state-of-the-art accuracies (significantly higher than tree-based models) on several datasets. However, most DL models proposed in other works are impossible to deploy on MCUs, requiring $>$ 1 million parameters and floating point operations for inference [3]. This, in turn, translates into unacceptable latency and energy consumption for IoT devices.

To bridge this gap, several works have introduced optimizations aimed at reducing the complexity of DL models and making them compatible with edge devices. Among the most popular there are quantization and pruning [4], [5], which modify the models statically before deployment, either at training time or post-training, respectively reducing the complexity of the network through the use of low-precision arithmetic or removing redundant parameters.

Orthogonal to such approaches, *dynamic* (or *adaptive*) optimization techniques leverage the varying complexity of inputs in order to save energy at runtime. The key intuition is that performing an inference with an accurate yet energy demanding model is wasteful for easy inputs, while naively shrinking the classifier may lead to a deterioration in predictions' quality for hard inputs. Additionally, easy inputs are often far more common than hard ones in real-word situations (e.g. for HAR the activity "laying" is far more frequent than "walking upstairs"), thus making adaptive approaches even more effective in terms of energy efficiency.

Most existing dynamic inference systems are based on sequentially executing multiple models of increasing complexity on each input, all performing the *same task*, and stopping the process as soon as a sufficient *output confidence* is reached.

Accepted as a conference paper at the IEEE 2022 PRIME Conference.

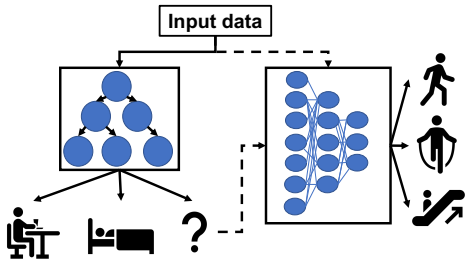

Fig. 1: High-level overview of the two-stage hierarchical inference architecture. Easy classes are classified by the DT (left), while complex ones are left to the CNN (right).

The multiple models can be separate [6] or share weights, using variable-width [7], variable-precision [8], or early-exit [9], [10] mechanisms. Alternatively, so-called "hierarchical classifiers" *split an inference task into different sub-tasks* based on the input. An example of this approach is [11], which introduces a framework based on several cascaded classifiers of increasing complexity, each working on an increasingly more difficult sub-task, constructed so that easier and more common inputs stop earlier in the cascade, reducing the average energy consumption.

In this paper, we follow the latter approach, proposing a novel type of adaptive architecture tailored for HAR tasks. Specifically, we combine a DT and a 1-dimensional (1D) Convolutional Neural Network (CNN). The first is trained to select only among the easiest activities, and output a generic *fallback* class for all other inputs. Only when the fallback class is predicted by the DT, the CNN is activated, performing an inference pass on the same input. We test our approach on a single-core RISC-V MCU and on a state-of-the-art HAR dataset [12], showing that we are able to save up to 67.7% energy at iso-accuracy, and with a negligible memory overhead, with respect to a static DL approach. More generally, our adaptive system is Pareto optimal when compared to multiple static CNN architectures, both in the energy versus accuracy and memory versus accuracy planes.

II. Proposed Method

This work proposes the first two-stage hierarchical, hybrid ML/DL system for HAR, to our knowledge.

Fig. 1 shows an high-level scheme of the solution, which mixes a DT with a 1D-CNN, each working on a different sub-task, rather than on the whole set of activities. Specifically, the DT always performs the first inference on each new input datum, looking for *easy* activity classes. For example, inputs corresponding to the "laying" activity can be intuitively recognized looking just at the orientation and amount of movement detected by the accelerometer. To cover more complex activities, the DT is trained to output a generic "fallback" class, which can be interpreted as: "the input does not belong to any of the easy classes". When the DT output is different from "fallback", the inference terminates immediately, using the predicted class as final result. On the contrary, when the fallback class is predicted, the 1D CNN is activated and applied to the same input, in order to distinguish among more complex activities (e.g., stairs climbing, running, cycling, etc).

The average energy consumption of this hierarchical model can be computed with the following equation:

$$E = E_{DT} + P_{fallback} * E_{CNN} \quad (1)$$

where E_x denotes the inference energy of x, and $P_{fallback}$ denotes the probability of an input being classified as "fallback" by the DT, thus enabling the CNN. Since the DT is significantly less computationally expensive than the CNN ($E_{DT} << E_{CNN}$), large energy savings are obtained for easy inputs, which only require the execution of the first "stage". Moreover, the latter are often the majority in real-life scenarios (we sleep for more hours than we jump or climb stairs), meaning that the *total* savings will also be considerable.

Additionally, with respect to a standard, single-stage classifier, our second-stage CNN operates on a reduced set of classes (only the complex ones), which reduces both the number of inference operations and the number of parameters of last fully-connected layer. The former further cuts the energy cost of the system, while the latter helps compensating the memory overhead introduced by the presence of the DT.

A. Training and Deployment Flow

Fig. 2 shows the training and deployment flow that we propose for the two-stage adaptive system just described, which is composed of five main steps.

1. DT-based Task Decomposition: In the first phase, we identify the easy and hard classes that will be later assigned to the DT and CNN respectively. To this end, we train and optimize the hyper-parameters of a DT on the *full* task (i.e., on all M activity classes). Namely, we select the tree depth that yields the best classification results on the full training set, by means of an exhaustive grid search, made possible by the limited size of HAR datasets. In our experiments, we explore depth values in [2,10]. Then, we select as easy classes the $M_{easy} < M$ static activities for which the complete DT achieves the largest F1 score. In general, the number of easy classes is a design parameter that should be explored, but in this preliminary work, we fix it to $M_{easy} = 2$. All other classes are considered hard.

2. Sub-tasks Training Sets Generation: Next, we generate the training sets for the DT and CNN. The former is obtained keeping samples relative to all easy classes unchanged, and modifying hard samples so that they all have the same label, corresponding to the fallback class. The CNN training set, instead, is simply obtained removing all easy classes samples, since those are not considered in the second inference stage.

3. Final DT Selection and Training: We then perform a second set of trainings and depth optimizations on the DT, this time on the modified training set containing only easy-classes and fallback examples. We look once again for the model that maximizes the classification score, using grid search.

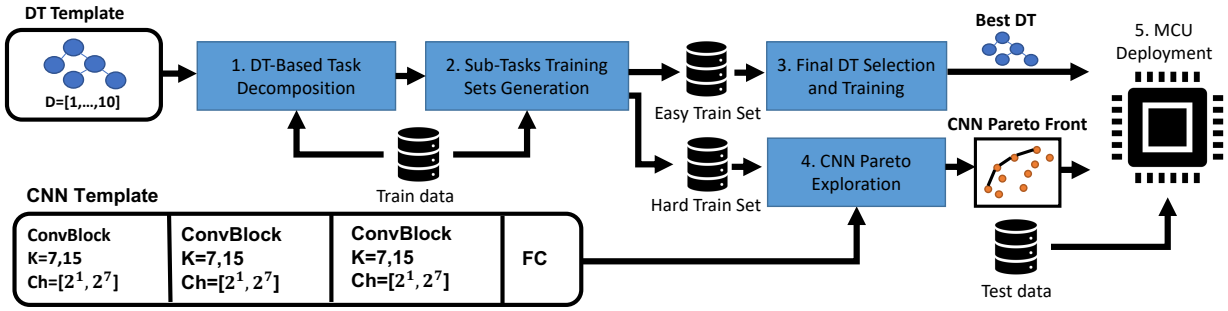

Fig. 2: Schematic view of the proposed training and deployment flow.

4. CNN Pareto Exploration: For what concerns the CNN, we perform a similar search, but in this case rather than only selecting the most accurate model, we extract the entire Pareto frontier in the energy versus classification score space. This difference comes from the fact that the second stage of the hierarchical model is by far the most computationally expensive. Therefore, building the hierarchical system starting from different Pareto-optimal CNNs is the easiest way to obtain different accuracy versus energy consumption trade-offs, as shown in Sec. III. The CNN architecture exploration is performed starting from a common template, based on a classic network structure [13], composed of three convolutional blocks (each including a convolution, batch normalization and max-pooling) and a final fully-connected layer. We explore variants of this template obtained changing the number of channels and the kernel size in each convolutional layer. Specifically, we consider all possible combinations of power-of-2 channels in $[2, ..., 128]$ and kernel sizes 7 or 15. The max-pooling stride and width are left fixed at 2.

5. MCU Deployment: Lastly, we deploy the selected DT and one of the Pareto-optimal CNNs onto the target MCU, combining them with the scheme of Fig 1, using optimized libraries (described below) to implement both classifiers.

III. Experimental Results

We target a popular public HAR dataset named UCI HAPT [12], featuring 958500 samples collected from the accelerometer and gyroscope of an Android phone. The data has been collected with a sampling frequency of 50 Hz from 30 subject performing 12 different activities. We keep the same train/test split proposed by the authors and split the data in non overlapping windows of 5 s, obtaining inputs of size 250x6. For this dataset, the 2 classes identified as easy by our procedure, hence those classified by the DT, are *sitting* and *laying*, which result in the highest F1-score (above 85%) on the training set. Together, these two activities account for $\approx$ 30% of both training and testing samples.

As deployment target, we select the single-core, ultra-low-power, RISC-V MCU Quentin [14], with characteristics close to those of common embedded HAR devices. The MCU has 520 kB of memory, and our results are obtained with the core running at 205.1 MHz, with a supply voltage of 0.54 V. In these conditions, the system consumes 3.8 mW of active power.

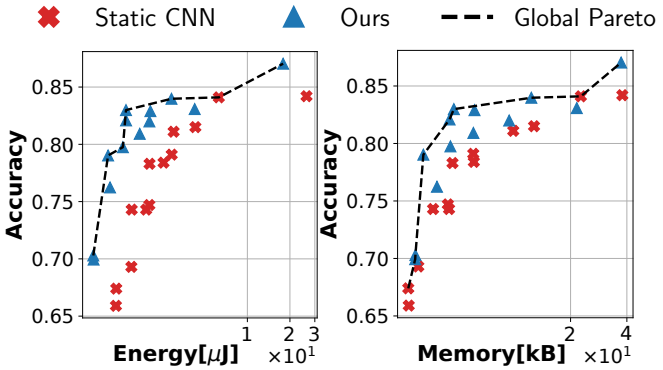

Fig. 3: Accuracy versus average energy consumption and total memory occupation results, for the proposed two-stage method and for a static CNN-based approach.

We train our DTs and CNNs in Python, before converting the trained models to optimized C code for the target MCU. For this conversion, we use the library described in [15] for the 1D CNN, while the DT and RF implementations are described in [1]. Both libraries are specifically tailored for the target hardware, leveraging the available SIMD and DSP-oriented instruction set extensions.

A. Energy and memory comparison

Fig. 3 shows the results obtained by our two-stage architecture in terms of classification accuracy versus inference energy consumption and memory occupation on the target MCU. Specifically, each blue triangle represents a hierarchical system built combining the DT determined in Step 3 of Fig. 2 (always the same) and one of the different Pareto-optimal CNN architectures found in Step 4 of Sec. II-A. For comparison, the two graphs also report the results obtained with different *static* CNNs derived from the same search space, and trained directly on *all* activity classes (red crosses). The dashed black line identifies the global Pareto front. Energy results refer to the average consumption for a single inference, over the entire test set.

On the energy side, Fig. 3 shows that our approach occupies most of the Pareto frontier, with the exception of a single architecture around 85% accuracy. Namely, we are able to reduce the average energy per inference by up to 67.7% at iso-accuracy, or to improve the accuracy by up to 12.34% for

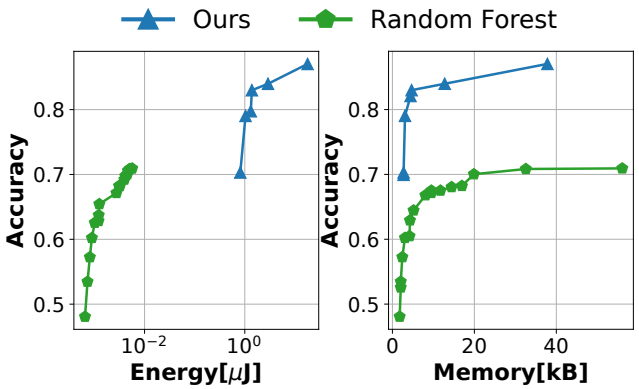

Fig. 4: Accuracy versus average energy consumption and total memory occupation results for the proposed two-stage method and for a Random Forest classifier.

the same average energy. Further, our most accurate adaptive system outperforms the best static CNN (+3% accuracy) with lower energy consumption. This result is obtained thanks to the fact that the DT achieves a classification accuracy of 92% on the "easy activities" sub-task, while having negligible inference energy costs (341x less than the smallest CNN).

Memory results are similar, showing that the presence of an additional DT in our two-stage system yields negligible overheads. Specifically, the DT requires a total of 0.2kB of memory, yielding a maximum overhead of around 9% w.r.t to the smallest static CNN ($\leq$ 3 kB). At the opposite side of the Pareto front, thanks to the reduced dimension of the last CNN fully-connected layer (given by the reduced number of activities classified), the total memory of the system is instead reduced by up to 64% for the same accuracy.

Figure 4 compares our method with a fully tree-based solution, consisting either of a single DT or of a Random Forest (RF) ensemble [1]. Precisely, the green curve shows the Pareto front obtained training all RFs with a number of trees from 1 (single DT) to 15, and depths in the $[2, 20]$ interval. The blue curve is the Pareto front of our adaptive method. As explained in Sec. I, tree-based models are expected to have lower energy consumption than DL ones, due to their lightweight computations, as confirmed by the graphs. However, also as expected, these models are unable to achieve comparable accuracy to a DL-based method within the memory constraints of a typical MCU. Indeed, even the largest RFs, which occupy more memory than our adaptive system, cannot achieve more than 71% accuracy.

B. Detailed Deployment Results

Table I reports the detailed deployment results, in terms of memory occupation, energy consumption, and inference latency, of the most accurate version of our adaptive system, and of the most accurate static CNN (*Base* rows). The results show that, while obtaining a +3% accuracy increase, the adaptive system can reduce the average energy and latency per inference by 31.6% even at maximum accuracy. Additionally,

	Mode	Acc.	Memory[kB]	Energy[μj]	Latency[ms]
Base	Ours	0.87	37.2	17.9	4.7
	Static	0.84	37.8	26.2	6.9
10x	Ours	0.89	37.2	6.2	1.6
	Static	0.85	37.8	26.2	6.9
20x	Ours	0.90	37.2	4.4	1.2
	Static	0.86	37.8	26.2	6.9

TABLE I: Detailed deployment results on the original and synthetically augmented test data.

in order to show the potential advantages of such a system in a real-world scenario, we repeat the experiment after synthetically over-sampling *laying* and *sitting* inputs (i.e., the easy classes) in the test set by 10x and 20x. In those settings, the accuracy improvement reaches +4%, and the energy/latency reductions increase to 76% and 83% respectively.

IV. Conclusions

We have presented a novel, two-stage HAR system for MCUs, featuring a small DT tree cascaded with a 1-dimensional CNN. Our approach leverages the different input complexity to save energy at inference time, switching on the CNN only when needed. Compared to a static CNN, we reduce the average energy per inference by up to 67.7% at iso-accuracy on a state-of-the-art HAR dataset. With respect to a pure tree-based model, our approach occupies a different portion of the design space, yielding much higher accuracy for the same memory occupation.

References

[1] F. Daghero *et al.*, "Ultra-compact binary neural networks for human activity recognition on risc-v processors," in *18th ACM CF*, 2021, p. 3–11.
[2] W. Sousa *et al.*, "A comparative analysis of the impact of features on human activity recognition with smartphone sensors," in *WebMedia*, 2017, pp. 397–404.
[3] H. Thu *et al.*, "Hihar: A hierarchical hybrid deep learning architecture for wearable sensor-based human activity recognition," *IEEE Access*, vol. 9, pp. 145 271–145 281, 2021.
[4] B. Jacob *et al.*, "Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference," in *IEEE CVPR*, 2018.
[5] P. Molchanov *et al.*, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.
[6] E. Park *et al.*, "Big/little deep neural network for ultra low power inference," in *IEEE CODES+ ISSS*, 2015, pp. 124–132.
[7] H. Tann *et al.*, "Runtime configurable deep neural networks for energy-accuracy trade-off," in *IEEE CODES+ ISSS*, 2016, pp. 1–10.
[8] D. Jahier Pagliari *et al.*, "Dynamic Bit-width Reconfiguration for Energy-Efficient Deep Learning Hardware," in *ACM ISLPED*, 2018, pp. 47:1—-47:6.
[9] S. Teerapittayanon *et al.*, "Branchynet: Fast inference via early exiting from deep neural networks," in *23rd IEEE ICPR*, 2016, pp. 2464–2469.
[10] F. Daghero *et al.*, "Adaptive Random Forests for Energy-Efficient Inference on Microcontrollers," in *29th VLSI-SoC*, 2021, pp. 1–6.
[11] K. Goetschalckx *et al.*, "Optimized hierarchical cascaded processing," *IEEE JETCAS*, vol. 8, no. 4, pp. 884–894, 2018.
[12] J.-L. Reyes-Ortiz *et al.*, "Transition-aware human activity recognition using smartphones," *Neurocomputing*, vol. 171, pp. 754–767, 2016.
[13] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[14] A. Di Mauro *et al.*, "Always-on 674μ w@ 4gop/s error resilient binary neural networks with aggressive sram voltage scaling on a 22-nm iot end-node," *IEEE TCAS-I*, vol. 67, no. 11, pp. 3905–3918, 2020.
[15] A. Burrello *et al.*, "TCN Mapping Optimization for Ultra-Low Power Time-Series Edge Inference," in *IEEE/ACM ISLPED*, 2021, pp. 1–6.