

Dendron: Enhancing Human Activity Recognition with On-Device TinyML Learning

Hazem Hesham Yousef Shalby
Politecnico di Milano
Milan, Italy
hazemhesham.shalby@polimi.it

Manuel Roveri
Politecnico di Milano
Milan, Italy
manuel.roveri@polimi.it

Abstract—Human activity recognition (HAR) is a research field that employs Machine Learning (ML) techniques to identify user activities. Recent studies have prioritized the development of HAR solutions directly executed on wearable devices, enabling the on-device activity recognition. This approach is supported by the Tiny Machine Learning (TinyML) paradigm, which integrates ML within embedded devices with limited resources. However, existing approaches in the field lack in the capability for on-device learning of new HAR tasks, particularly when supervised data are scarce. To address this limitation, our paper introduces *Dendron*, a novel TinyML methodology designed to facilitate the on-device learning of new tasks for HAR, even in conditions of limited supervised data. Experimental results on two public-available datasets and an off-the-shelf device (STM32-NUCLEO-F401RE) show the effectiveness and efficiency of the proposed solution.

Index Terms—Human activity recognition, TinyML, smart glasses, wearable sensing, hierarchical classification

I. INTRODUCTION

Human activity recognition (HAR) is a research area focusing on developing systems that can automatically identify user activities (e.g., lying, standing, walking, or running) by using Machine Learning (ML) techniques. In the last few years, research in the field focused on the development of HAR solutions operating on wearable devices, so as to support the on-device recognition of human activities directly on tiny devices [1]–[6].

This novel research field enhances the Tiny Machine Learning (TinyML) perspective, which integrates ML within embedded devices, constrained by limited memory, low computational power, and low power consumption [7].

Interestingly, existing works in the field suffer from the absence of an on-device adaptation and integration of new tasks to the HAR model at runtime with limited supervised data availability [8]–[11]. However, this is essential in on-device HAR, as we cannot collect and store large datasets directly on the device.

This paper aims to introduce a novel methodology, called *Dendron*, that allows the HAR model to learn new tasks directly on the wearable device in supervised data scarcity conditions. Specifically, the proposed architecture employs a hierarchical approach that, unlike the traditional methods that utilize a single model for HAR, takes into account the similarities between activities and decomposes the multi-class classification problem into multiple stages of sub-classification

[2]. For example, the first stage of the hierarchical classification predicts a general-activity category (e.g., walking vs running), while subsequent stages refine the prediction by identifying specific sub-activities (e.g., walking upstairs vs downstairs). With its specifically-designed architecture, *Dendron* efficiently and effectively supports the on-device learning of new tasks. Differently from traditional hierarchical approaches (e.g., [2]) that require multiple training processes (i.e., one for each sub-task), *Dendron* has a unique training process, hence simplifying and reducing the complexity of the training phase. Additionally, the unified training process of *Dendron* allows for the effective and efficient learning of tasks with limited data availability.

The experimental results show that *Dendron* has superior performance in on-device learning compared to existing solutions, particularly under conditions of limited supervised data, as evidenced by experimental results on two publicly available datasets [12], [13] and in the porting on a resource-constrained device (STM32-NUCLEO-F401RE). Additionally, *Dendron* uses $5\times$ less memory, requires $2\times$ less computational load, and takes $2\times$ less time per inference compared to other hierarchical solutions.

Summing up, the novel contributions of this paper are as follows:

- a novel hierarchical architecture that allows the efficient and effective on-device learning of new activities under conditions of supervised data scarcity;
- a novel learning algorithm to train a hierarchical HAR in a unified process, simplifying the intricate training process typically linked with such architectures.

The paper is organized as follows. Section II provides an overview of the related literature. Section III delves into the proposed architecture and its key features. In Section IV experimental results are provided. Finally, Section V discusses the main findings of this research and addresses future research directions.

II. RELATED WORKS

A. Human Activity Recognition

HAR techniques present in the literature [1], [14] can be categorized into two main approaches: vision-based HAR and sensor-based HAR. Vision-based HAR involves the analysis

of images or videos captured by optical sensors, while sensor-based HAR exploits data from wearable and environmental sensors, such as accelerometers and gyroscopes [15]. In this paper, we focus on sensor-based HAR, and the related literature will explore this specific approach.

More specifically, [14] proposes an extensive study of the state-of-the-art methods for HAR together with their relative challenges. In particular, the authors divided HAR techniques into two main categories: ML-based (e.g., Support Vector Machine, K-Nearest Neighbour, and Decision Trees) and Neural Network (NN)-based (e.g., CNNs and Recurrent Neural Networks). In [16] a dataset consisting of data produced by accelerometer and gyroscope sensors of smartphones is used to show the effectiveness of different ML algorithms. This analysis showed that CNNs provide the largest recognition abilities and, for this reason, the proposed *Dendron* is designed starting from a CNN.

Remarkably, sensor-based HAR is typically conducted over a fixed-duration window of signals and, in this perspective, the selection of the window size and sampling frequency becomes crucial when designing a HAR pipeline. In [17] the authors extensively explore how different windowing techniques impact the recognition accuracy. The findings reveal that shorter windows, specifically those lasting 2 seconds or less, yield the highest accuracy in HAR. In [18] a study on 15 public datasets shows that a frequency of 15 – 20 Hz may be sufficient for HAR on a wearable device (specificity/sensitivity higher than 95%). Moreover, [19] shows that increasing the sampling rate above 20 Hz improves the recognition accuracy by just 1%.

One of the most promising architectures in HAR is the hierarchical approach [2]. This method enhances the discrimination between similar activities without explicitly assuming temporal relationships between actions and activities. Hierarchical HAR is a tree-based activity recognition model that, first infers the abstract activity of the user and, subsequently, identifies the specific activity in a top-down scheme [3]–[6]. The development of *Dendron* is inspired by the hierarchical HAR solutions. This aspect will be cleared in Section III.

B. TinyML

The popularity of TinyML derives from its capability to process data locally on the device, improving privacy, and security, reducing latency, and enabling offline operation without a constant internet connection [7]. Most of the TinyML solutions present in the literature aim at reducing the size and complexity of the ML models [20], [21]. These solutions encompass precision scaling, which involves techniques like quantization [22] and model compression [23], and task dropping to alleviate computational burdens [20]. Another area of exploration in TinyML involves redesigning the network architecture, such as implementing approximate or dilated convolutions [24], [25].

Regarding the on-device learning of TinyML models, several studies in the literature introduced techniques targeting either specific layers or the entire architecture of Fully Connected Neural Networks [8], [9]. Moreover, contributions fo-

cus on the training of all the layers of Convolutional Neural Networks (CNNs) can be found in the literature [10]. All these approaches focused on performing on-device learning by optimizing the memory and the computational demand, while our approach aims to enhance the on-device learning process with limited supervised data availability and adding new tasks for HAR.

III. PROPOSED METHODOLOGY

This section introduces the proposed methodology. More specifically, Section III-A formalizes the proposed solution. Section III-B details the proposed *Dendron* architecture. Section III-C, and III-D detail the initial off-device learning and on-device inference process of the proposed architecture, respectively. Finally, Section III-E discusses the on-device learning process.

A. Formalization

Following the formalization introduced in [2], the HAR multi-class classification task $T^{(0)}$ can be decomposed into n sub-tasks $\{T^{(1)}, \dots, T^{(n)}\}$. Formally, a task $T^{(i)}$ is a classifier that maps an input window of size T of sequential data $\{x_t, \dots, x_{t-(T-1)}\}$ to its label $y_t^{(i)}$ as follows:

$$T^{(i)}: \{x_t, \dots, x_{t-(T-1)}\} \rightarrow y_t^{(i)}$$

being:

- $x_t \in \mathbb{R}^{N_{in}}$ the input of size N_{in} at time t ,
- $y_t^{(i)}$ a label belonging to the label set $\Omega^{(i)} = \{\Omega_{k_1}^{(i)}, \dots, \Omega_{k_{k_i}}^{(i)}\}$, where k_i is the total number of classes related to the task $T^{(i)}$.

We emphasize that the sequential execution of a subset of the sub-tasks set $\{T^{(1)}, \dots, T^{(n)}\}$ is equivalent to the execution of $T^{(0)}$. Indeed, in the hierarchical process, the predicted label $y_t^{(L)}$ from the last executed sub-task $T^{(L)}$ in the hierarchy belongs to the label set $\Omega^{(0)}$ of $T^{(0)}$, as follow:

$$y_t^{(L)} \in \Omega^{(0)}, \text{ being } T^{(L)} \text{ the last executed sub-task.}$$

To express the dependencies between tasks, we introduce the matrix D of dimension $n \times n$, representing the dependencies among tasks as follows:

$$D = \begin{bmatrix} d_1^{(1)} & \dots & d_n^{(1)} \\ \vdots & \ddots & \vdots \\ d_1^{(n)} & \dots & d_n^{(n)} \end{bmatrix}$$

where $d_j^{(i)} \in \{\Omega^{(j)} \cup \emptyset\}$ denotes whether the task $T^{(i)}$ depends on the task $T^{(j)}$. More specifically:

- $d_j^{(i)} = \emptyset$ indicates the absence of dependencies of the task $T^{(i)}$ from the task $T^{(j)}$;
- $d_j^{(i)} \in \Omega^{(j)}$ indicates that a dependency exists between $T^{(i)}$ and $T^{(j)}$. Moreover, the label $d_j^{(i)}$ of $T^{(j)}$ must be predicted to trigger the activation of the task $T^{(i)}$.

In Figure 1 an example of HAR schema with $n = 6$ is presented. Both the subset of the sub-tasks to be executed and

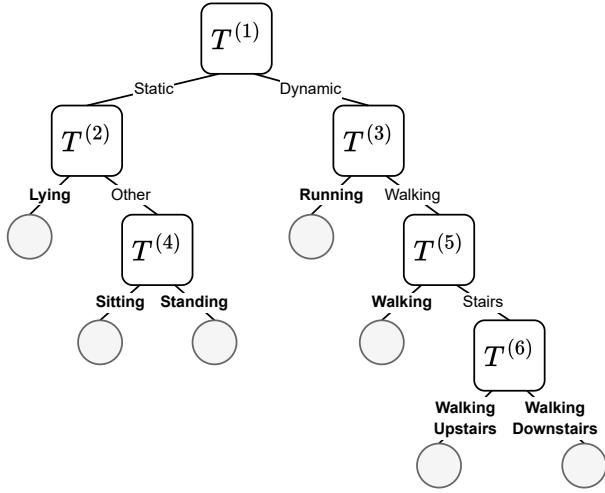


Fig. 1. Example of hierarchical HAR schema.

their execution order are determined through a dependency structure D introduced above. As an example, to illustrate the concept of dependencies, consider task $T^{(6)}$ shown in Figure 1. Being $\Omega^{(5)} = \{\text{walking, stairs}\}$, the row of D associated with $T^{(6)}$ is $d^{(6)} = [\emptyset, \emptyset, \emptyset, \emptyset, \text{stairs}, \emptyset]$ indicating that, to activate $T^{(6)}$, $T^{(5)}$ must predicts *stairs*.

B. Proposed architecture

The proposed *Dendron* architecture aims to reduce both the computational and the memory requirements of the TinyML modules for HAR (i.e., one module for each sub-task) by employing a single feature extractor (FE) module, which is used by all the tasks, and a set of multiple fully connected (FC) modules, one for each specific task. This approach leads to a general FE learned without specific task dependencies and FC modules specific to their respective tasks. More formally, we refer to FE as $g(x_t, \dots, x_{t-(T-1)})$, and to the FC related to the task $T^{(i)}$ as $h^{(i)}(\cdot)$. Therefore, each task $T^{(i)}$ is implemented as:

$$\tilde{y}_t^{(i)} = h^{(i)}(g(x_t, \dots, x_{t-(T-1)}))$$

being $\tilde{y}_t^{(i)}$ the predicted label for task $T^{(i)}$ at time t .

In the proposed architecture, the function $g(\cdot)$ is implemented through a convolutional FE. Differently, the function $h^{(i)}(\cdot)$ of task $T^{(i)}$ is designed with $b^{(i)}$ dense layers, each having $q_{u_i}^{(i)}$ dense units with $u_i \in [1, b^{(i)}]$. The final layer of FC is a softmax dense layer that yields k_i outputs (i.e., $q_{b^{(i)}}^{(i)} = k_i$).

C. Off-device training process

The training process of $g(\cdot)$ and $h^{(i)}$ s is jointly carried out by considering the problem as a multi-output classification problem, as shown in Figure 2. Therefore, after defining a proper loss function $L^{(i)}(y^{(i)}, \tilde{y}^{(i)})$ for each sub-task $T^{(i)}$, the total loss function L is defined as:

$$L = \sum_{i=1}^n \alpha^{(i)} L^{(i)},$$

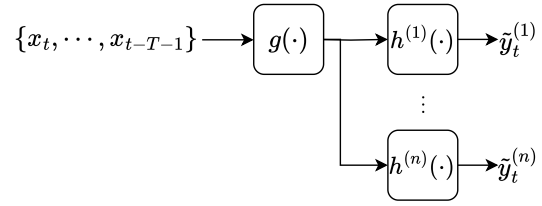


Fig. 2. Overview of the off-device training process.

being $\alpha^{(i)} \in \mathbb{R}^+$ a parameter that determines how much the optimization process should focus on minimizing the loss $L^{(i)}$ associated with task $T^{(i)}$ for the given sample. In our approach, $\alpha^{(i)}$ denotes the probability of activating the corresponding task $T^{(i)}$, which is implemented as follows:

$$\alpha^{(i)} = \begin{cases} 1 & \text{if } d_j^{(i)} = \emptyset, \forall j \in [1, n], \\ \sum_{j=1}^n c(d_j^{(i)}) \alpha^{(j)} & \text{otherwise} \end{cases}$$

where $c(d_j^{(i)}) \in [0, 1]$ is the confidence level with which $d_j^{(i)}$ is predicted. In our implementation, the final layer of all FC modules exploits a softmax operation, hence allowing us to compute the confidence $c(d_j^{(i)})$ as the softmax score for the corresponding label $d_j^{(i)}$. Note that $c(\emptyset) = 0$.

D. On-device inference process

The inference process of *Dendron* is carried out hierarchically on the device. Specifically, Algorithm 1 reports the on-device inference process of *Dendron*. In particular, $g(\cdot)$

Algorithm 1 On-device inference process

- 1: **procedure** DENDRON_PREDICTION($x_t, \dots, x_{t-(T-1)}$)
 - 2: $f_t = g(x_t, \dots, x_{t-(T-1)})$ ▷ Extract feature
 - 3: $T^{(i)} = \text{root task}$
 - 4: **do**
 - 5: $\tilde{y}_t^{(i)} = h^{(i)}(f_t)$ ▷ prediction with $T^{(i)}$
 - 6: $T^{(i)} = T^{(j)}$ ▷ where: $\tilde{y}_t^{(i)} = d_j^{(j)}$
 - 7: **while** $\tilde{y}_t^{(i)} \notin \Omega^{(0)}$
 - 8: **return** $\tilde{y}_t^{(i)}$
 - 9: **end procedure**
-

is executed on the input window $\{x_t, \dots, x_{t-(T-1)}\}^1$. Then, following the hierarchical schema defined in D , a subset of $\{h^{(1)}(\cdot), \dots, h^{(n)}(\cdot)\}$ is sequentially executed on the output of $g(\cdot)$. More in details, the first FC module to be executed (i.e., the root task $T^{(i)}$) is the one without any dependency, hence $d_j^{(i)} = \emptyset$, with $j \in [1, n]$. Then, we iterate through the dependency schema D until the predicted label $\tilde{y}_t^{(i)}$ is the algorithm's final prediction (i.e., $\tilde{y}_t^{(i)} \in \Omega^{(0)}$).

¹During the inference process in HAR, input windows are usually partially overlapped. However, this does not affect the proposed solution.

E. On-Device learning process

In this work, our objective is to support the on-device learning of an additional task $T^{(new)}$ in limited supervised-data availability conditions. Consequently, when a user chooses to add a new task $T^{(new)}$, our objective is to minimize the amount of data labeling required to learn $T^{(new)}$. In the proposed *Dendron* architecture, learning a new task $T^{(new)}$ requires learning $h(\cdot)^{(new)}$, and deciding how to update the hierarchical schema D . Both aspects are detailed in what follows.

1) *Learn the FC module $h(\cdot)^{(new)}$* : *Dendron* is designed in a way such that only the learning of the FC $h(\cdot)^{(new)}$ layer for $T^{(new)}$ is needed. Consequently, learning a new task $T^{(new)}$ allows us to keep fixed $g(\cdot)$ and fine-tune the parameters of $h^{(new)}(\cdot)$ to minimize the corresponding loss $L^{(new)}$. In particular, $h^{(new)}(\cdot)$ is optimized using the standard Gradient Descent algorithm, and the memory m_w required to store the weights of $h^{(new)}$ is computed as (see formalism in Section III-B):

$$m_w = \sum_{u=1}^{b^{(new)}} q_u^{(new)} \times q_{u-1}^{(new)}.$$

being $b^{(new)}$ the number of dense layers of $h^{(new)}(\cdot)$, each having $q_u^{(new)}$ dense units with $u \in [1, b^{(new)}]$. Note that $q_0^{(new)}$ is the number of feature extracted by $g(\cdot)$.

2) *Update the hierarchical schema D* : Determining where the newly learned task $T^{(new)}$ must be integrated within the hierarchical schema established by D is done as described in Algorithm 2. In particular, $T^{(new)}$ is added to the most

Algorithm 2 On-device node selection process

Require: S \triangleright acquired dataset for the new task $T^{(new)}$

- 1: count = $[0, \dots, 0]$ \triangleright len(count) = num of labels of $T^{(0)}$
- 2: **for** $s \in S$ **do**
- 3: $\tilde{y} = \text{DENDRON_PREDICTION}(s)$
- 4: count[i] = count[i] + 1 $\triangleright \tilde{y} = \Omega_i^{(0)}$
- 5: **end for**
- 6: count_tmp = SORT(count)
- 7: $f' = \text{count_tmp}[0]$ \triangleright frequency of the most predicted label
- 8: $f'' = \text{count_tmp}[1]$ \triangleright frequency of the 2nd most predicted label
- 9: **if** $f' - f'' > \delta$ **then** \triangleright with $\delta \in [0, 1]$
- 10: add to the node associated with f'
- 11: **else**
- 12: add to the nodes associated with both f' and f''
- 13: **end if**

frequently predicted label if the frequency difference with the second most predicted label exceeds a threshold δ . Otherwise, $T^{(new)}$ is added to both the top predicted two labels.

Figure 3 reports an example of this node selection process. Specifically, we aim to add the task of distinguishing between walking upstairs and downstairs. Given that the majority of the data corresponds to walking (with $f' = \frac{959}{1209}$, and $f'' = \frac{247}{1209}$ the frequency of the two most predicted labels respectively),

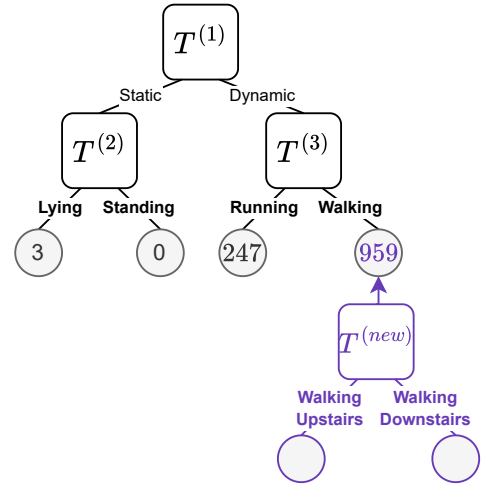


Fig. 3. Example of the process for selecting the node to add a new task $T^{(new)}$.

the walking node is considered a potential candidate for the new task $T^{(new)}$. Subsequently, if we set $\delta = 0.5$ the second heuristic is also satisfied. Indeed, the difference in frequency between the top two predicted labels is ≈ 0.59 , which exceeds $\delta = 0.5$. Note that if δ were set to 0.6, the second heuristic would not be met, and $T^{(new)}$ would have been added to both running and walking.

IV. EXPERIMENTAL RESULTS

This section evaluates the effectiveness and efficiency of the proposed architecture using the UCA-EHAR [12], and the UCI-HAPT [13] datasets.

As a comparison, the following architectures have been employed:

- 1) **Traditional single-model solution** [12]: a single neural network where the output corresponds to the performed task.
- 2) **Hierarchical solution** [2]: multiple neural networks organized hierarchically, one for each sub-task.

To ensure consistency across experiments for *Dendron*, the traditional and the hierarchical solution the evaluation employs the architecture proposed in [12], a modified version of a one-dimensional ResNetv1-6 [26]. Specifically, the feature extractor $g(\cdot)$ is the same as the one reported in Figure 4, while the fully connected module $h^{(i)}(\cdot)$ comprises a single dense layer having the same number of neurons as the number of classes (k_i) to be classified.

This section is organized as follows. Section IV-A details the datasets used for the evaluation. Section IV-B evaluates the classification capability. Section IV-C evaluates the performances of adding a new task during the operational life of HAR. Finally, Section IV-D discusses the memory usage, computation requirements, and mean latency.

A. Datasets

1) *UCA-EHAR dataset*: The UCA-EHAR dataset [12] includes gyroscopic and accelerometer data collected from smart

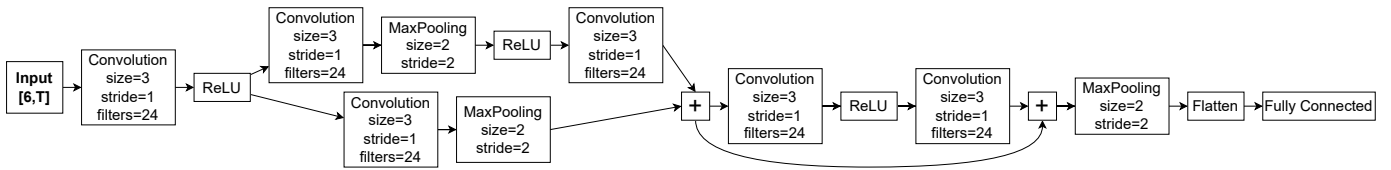


Fig. 4. One-dimensional ResNetV1-6 model architecture used in the evaluation conducted in Section IV.

glasses worn by 20 people engaged in 8 different activities. Specifically, the dataset has been collected at 26 Hz, and has been divided into approximately 77% for training (14 subjects) and 23% for testing (6 subjects) of the total samples, respectively, and, the window size has been set to 2s as suggested in [12].

2) *UCI-HAPT dataset*: The UCI-HAPT dataset [13] is an extension of the UCI-HAR dataset [27]. It includes gyroscope and accelerometer data collected from 30 people engaged in 6 basic activities, similar to the original UCI-HAR dataset, with a waist-mounted smartphone. Additionally, the UCI-HAPT dataset introduces 6 postural transition activities, expanding on the original activity set. Specifically, the dataset has been collected at 50 Hz, and has been divided into approximately 70% for training (21 subjects) and 30% for testing (9 subjects) of the total samples, respectively, and, the window size has been set to 2.56s as suggested in [27].

Figure 5 reports the distribution of samples per class for the classes used in the evaluation across both the UCA-EHAR and UCI-HAPT datasets.

B. Off-device learning performance evaluation

In this section, we compare the classification capability of *Dendron* with the one of the traditional and hierarchical solutions. Specifically, the following classes have been employed in this study: standing, sitting, walking, lying down, walking downstairs, walking upstairs, and running. Furthermore, for

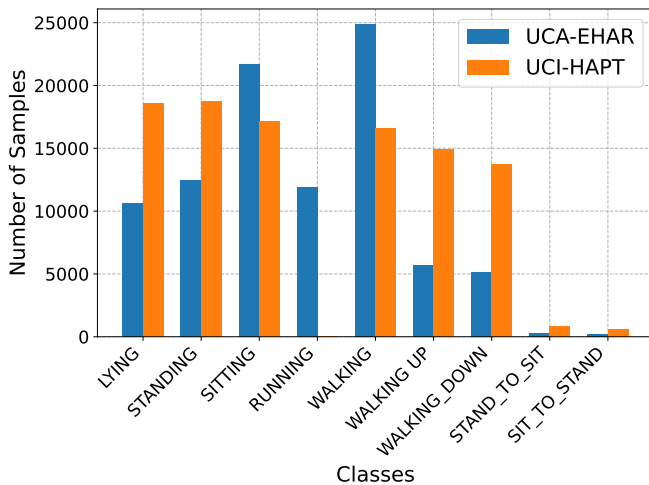


Fig. 5. Number of Samples per Class for UCA-EHAR and UCI-HAPT Datasets

TABLE I
OFF-DEVICE CLASSIFICATION PERFORMANCES COMPARISON BETWEEN THE SINGLE-MODEL SOLUTION, THE HIERARCHICAL SOLUTION, AND DENDRON.

Dataset	Metric	Single-model	Hierarchical	Dendron
UCA-EHAR	Accuracy	0.73 ± 0.015	0.75 ± 0.011	0.77 ± 0.012
	Precision	0.74 ± 0.004	0.75 ± 0.007	0.77 ± 0.005
	Recall	0.73 ± 0.015	0.75 ± 0.011	0.77 ± 0.012
	F1-score	0.72 ± 0.009	0.75 ± 0.009	0.76 ± 0.007
UCI-HAPT	Accuracy	0.90 ± 0.008	0.90 ± 0.004	0.91 ± 0.007
	Precision	0.90 ± 0.007	0.90 ± 0.003	0.91 ± 0.006
	Recall	0.90 ± 0.008	0.90 ± 0.004	0.91 ± 0.006
	F1-score	0.89 ± 0.008	0.90 ± 0.004	0.91 ± 0.007

both the hierarchical solution and the proposed *Dendron*, the employed hierarchical schema is the one reported in Figure 1, which has been introduced in [3]. The results for both datasets are summarized in Tables I and II.

Table I compares accuracy, precision, recall, and F1-score across the three methods. Overall, *Dendron* demonstrates the best performances, outperforming the compared approaches by at least 1 – 2%.

Table II compares the accuracy-per-class across the three methods. Specifically, the hierarchical solution generally outperforms the single-model approach, as its structure facilitates distinguishing between closely related activities. However, it struggles when classes have few supervised data, such as walking upstairs and walking downstairs for the UCA-EHAR where its performances are 4 – 6% worse than the one of the single-model approach. On the contrary, while maintaining the hierarchical framework and having less memory usage, *Dendron* achieves higher performance, even in cases where few supervised data are available for specific classes.

C. On-device learning performance evaluation

We now conduct an analysis to demonstrate the advantages of utilizing *Dendron* for the on-device learning of new tasks during the operational life of HAR. The experimental results, shown in Table III, focus on adding two new tasks, which contain some of the classes with the least amount of data in both datasets, as indicated in Figure 5. These tasks are: walking downstairs vs walking upstairs, and sit-to-stand vs stand-to-sit. Specifically, two solutions are analyzed for comparison:

- 1) *Dendron*, using the previously learned $g(\cdot)$ and learning only $h(\cdot)^{(new)}$ and
- 2) the traditional approach, where both $g(\cdot)^{(new)}$, and $h(\cdot)^{(new)}$ must be learned. This approach is employed solely for comparison purposes and is impractical for

TABLE II
OFF-DEVICE PER-CLASS-ACCURACY COMPARISON BETWEEN THE SINGLE-MODEL SOLUTION, THE HIERARCHICAL SOLUTION, AND DENDRON.

Dataset	Class	Single-model	Hierarchical	Dendron
UCA-EHAR	Lying	0.99 ± 0.022	0.99 ± 0.029	1.00 ± 0.000
	Standing	0.24 ± 0.069	0.26 ± 0.025	0.36 ± 0.093
	Sitting	0.73 ± 0.081	0.73 ± 0.028	0.68 ± 0.084
	Running	0.84 ± 0.031	0.93 ± 0.014	0.94 ± 0.014
	Walking	0.80 ± 0.061	0.90 ± 0.027	0.89 ± 0.025
	Up	0.70 ± 0.050	0.66 ± 0.011	0.70 ± 0.024
	Down	0.77 ± 0.035	0.71 ± 0.042	0.75 ± 0.031
UCI-HAPT	Lying	0.95 ± 0.001	0.99 ± 0.000	1.00 ± 0.000
	Standing	0.84 ± 0.068	0.84 ± 0.031	0.85 ± 0.029
	Sitting	0.76 ± 0.081	0.78 ± 0.050	0.79 ± 0.056
	Walking	0.96 ± 0.018	0.97 ± 0.016	0.98 ± 0.005
	Up	0.94 ± 0.042	0.94 ± 0.011	0.95 ± 0.024
	Down	0.93 ± 0.036	0.94 ± 0.008	0.91 ± 0.026

real-world applications due to its excessive memory requirements (as we will prove in Section IV-D) and the large amount of supervised data needed.

Moreover, to emulate the on-device incremental collection of the data, the experimental results evaluated the classification ability across various data partition percentages, starting from an initial 10% and increasing incrementally by 10% up to 100%. Specifically, for the sit-to-stand vs stand-to-sit task, the UCA-EHAR dataset provides a maximum of 4 min, while the UCI-HAPT dataset offers up to 24 min. For the walking downstairs vs walking upstairs task, the UCA-EHAR dataset provides a maximum of 40 min, whereas the UCI-HAPT dataset includes up to 48 min². This emulation process is necessary to assess the performance degradation relative to the amount of supervised data. We emphasize that memory is a critical constraint in TinyML applications, as one minute of gyroscopic and accelerometer data collected at 50Hz consumes approximately 20KB of memory. Therefore, one of the key objectives of *Dendron* is to enhance classification performance while minimizing the data required for learning new tasks.

The evaluation shows that *Dendron* achieves higher accuracy than the traditional approach in limited supervised data availability conditions. Indeed, the traditional approach surpasses in accuracy *Dendron* only when we use at least 40 – 50% of the available data, and even in that case the traditional approach exhibits only an increase of 2 – 4% in accuracy over *Dendron*. Furthermore, in the case of extreme supervised data-scarcity conditions, as in the case of the task of sit-to-stand vs stand-to-sit where we have only 4 and 24 min of data respectively for the two datasets, the traditional approach never surpasses the proposed *Dendron* architecture. Moreover, the experimental results prove that *Dendron* ensures stable performances across various data availability conditions. In contrast, when *Dendron* is not employed, the performances are highly dependent on data availability, therefore less data availability leads to a significant degradation in performance.

²The UCI-HAPT dataset has \approx 480 minutes for walking upstairs vs. downstairs, but only 10% was used for the evaluation to simulate supervised data scarcity condition.

TABLE III
COMPARISON BETWEEN THE TRADITIONAL APPROACH AND DENDRON IN TERMS OF ACCURACY WITH DIFFERENT DATA PERCENTAGES FOR THE TRAINING. THE TASKS USED FOR THE EVALUATION ARE: WALKING DOWNSTAIRS VS WALKING UPSTAIRS, AND SIT-TO-STAND VS STAND-TO-SIT.

UCA-EHAR				
%	downstairs vs upstairs		sit-to-stand vs stand-to-sit	
	Traditional	Dendron	Traditional	Dendron
	10	0.72 ± 0.019	0.84 ± 0.045	0.68 ± 0.057
20	0.76 ± 0.028	0.85 ± 0.009	0.74 ± 0.062	0.83 ± 0.053
30	0.80 ± 0.022	0.88 ± 0.016	0.76 ± 0.054	0.83 ± 0.030
40	0.82 ± 0.039	0.87 ± 0.020	0.71 ± 0.065	0.82 ± 0.072
50	0.88 ± 0.025	0.87 ± 0.016	0.79 ± 0.080	0.87 ± 0.011
60	0.88 ± 0.027	0.87 ± 0.008	0.75 ± 0.068	0.87 ± 0.011
70	0.90 ± 0.032	0.88 ± 0.008	0.79 ± 0.028	0.82 ± 0.038
80	0.88 ± 0.058	0.89 ± 0.003	0.75 ± 0.056	0.84 ± 0.038
90	0.88 ± 0.043	0.89 ± 0.010	0.72 ± 0.050	0.88 ± 0.033
100	0.89 ± 0.015	0.89 ± 0.005	0.80 ± 0.027	0.83 ± 0.030

UCI-HAPT				
%	downstairs vs upstairs		sit-to-stand vs stand-to-sit	
	Traditional	Dendron	Traditional	Dendron
	10	0.88 ± 0.015	0.89 ± 0.023	0.72 ± 0.039
20	0.91 ± 0.024	0.93 ± 0.007	0.77 ± 0.026	0.85 ± 0.022
30	0.92 ± 0.026	0.94 ± 0.006	0.79 ± 0.024	0.88 ± 0.010
40	0.95 ± 0.027	0.94 ± 0.007	0.78 ± 0.012	0.86 ± 0.030
50	0.97 ± 0.009	0.94 ± 0.006	0.81 ± 0.018	0.88 ± 0.010
60	0.97 ± 0.019	0.94 ± 0.007	0.81 ± 0.014	0.88 ± 0.023
70	0.98 ± 0.009	0.94 ± 0.005	0.81 ± 0.020	0.90 ± 0.020
80	0.98 ± 0.005	0.95 ± 0.006	0.81 ± 0.015	0.88 ± 0.018
90	0.98 ± 0.006	0.94 ± 0.004	0.83 ± 0.022	0.89 ± 0.007
100	0.99 ± 0.004	0.95 ± 0.005	0.84 ± 0.023	0.88 ± 0.017

For instance, for the task of walking upstairs versus downstairs in the UCA-EHAR dataset when *Dendron* is employed, the accuracy ranges from 0.84 to 0.89, with only a 0.05 difference between the maximum and minimum accuracy. Conversely, when employing the traditional approach, the accuracy ranges from 0.72 to 0.90, resulting in a larger 0.18 difference between the maximum and minimum accuracy.

D. Computation, memory, and latency evaluation

This section analyzes the proposed solution focusing on memory usage, computation requirements, and mean latency. Specifically, the assessment is carried out by using the STM32-NUCLEO-F401RE evaluation board, commonly utilized in TinyML applications. This board serves as a representative platform for evaluating the feasibility and effectiveness of the proposed architecture within the TinyML context.

1) *On-device inference evaluation*: The findings of this evaluation are summarized in Table IV. Specifically, The results show that our solution outperformed the existing hierarchical solution in terms of memory, computation, and mean latency. Specifically, *Dendron* has 5× less memory usage, 2× less computation load, and requires 2× less time per inference compared to the hierarchical solution. Additionally, *Dendron* exhibits only 7 KiB increase in memory compared to the traditional single-model solution, while showing a reduced computational load compared to the traditional single-model.

2) *On-device learning evaluation*: We evaluate the memory and time required for the on-device learning of a new task

TABLE IV

COMPARISON IN TERMS OF MEMORY, COMPUTATION, AND LATENCY USING THE STM32-NUCLEO-F401RE FOR INFERENCE. $M_{FE}^{(i)}$, $C_{FE}^{(i)}$, $M_{FC}^{(i)}$, and $C_{FC}^{(i)}$ are respectively the memory, and computation load of the i -th task for the feature extractor (FE), and the fully connected (FC) modules.³

Solution	Memory	Computation (MACC)	Mean Latency
Single-model	$M_{FE} + M_{FC}$ ≈ 60 KiB	$C_{FE} + C_{FC}$ $\approx 427, 242$	≈ 47 ms
Hierarchical	$\sum_i^n M_{FE}^{(i)} + M_{FC}$ ≈ 324 KiB	$\sum_i C_{FE}^{(i)} + C_{FC}$ $\approx 851, 292$	≈ 94 ms
Dendron	$M_{FE} + \sum_i^n M_{FC}^{(i)}$ ≈ 67 KiB	$C_{FE} + \sum_i C_{FC}^{(i)}$ $\approx 426, 444$	≈ 47 ms

TABLE V

ANALYSIS OF DENDRON ON-DEVICE LEARNING OF A NEW BINARY TASK $T^{(new)}$ FOR ONE SAMPLE OF SIZE 2s ON THE STM32-NUCLEO-F401RE. $g(\cdot)$ IS THE FEATURE EXTRACTOR, AND $h(\cdot)$ IS THE FULL CONNECTED PART OF DENDRON.

	Time	Additional Memory	
Forward pass	$g(\cdot)$	≈ 46 ms	0 Bytes
	$h(\cdot)$	≈ 3 ms	0 Bytes
Backward pass	$g(\cdot)$	Not Required	Not Required
	$h(\cdot)$	≈ 7 ms	8 Bytes

$T^{(new)}$ using *Dendron*. We emphasize that performing on-device training using traditional approaches is not feasible in a TinyML scenario, due to the memory requirements associated with the training of the entire network. The findings of the evaluation are summarized in Table V. Specifically, a forward-backward pass for one window sample $\{x_t, \dots, x_{t-(T-1)}\}$ of 2s requires approximately 56ms and only 8 Bytes overhead w.r.t the inference. By storing the output of $g(\cdot)$ (the intermediate extracted feature) and performing only the forward-backward pass of $h(\cdot)$, the forward-backward pass time can be reduced to approximately 10ms. For instance, if a user is required to collect 30s of data per class to learn a new binary task $T^{(new)}$ on-device, it would take approximately 1.2s to complete one epoch over 2s windows with 50% overlap.

V. CONCLUSIONS AND FUTURE WORKS

This paper focused on the HAR task on wearable devices, specifically emphasizing the on-device learning of new tasks. Specifically, *Dendron* outperforms existing solutions in conditions of supervised data scarcity. Indeed, *Dendron* can achieve comparable performance levels in new tasks learning on-device as other solutions, but using only 10% of the data required by those alternative methods to achieve similarly high-performance levels. Additionally, *Dendron* requires less memory and less computation to perform the on-device training compared to other solutions which are in general not feasible in a TinyML scenario, due to the memory requirements associated with the required training process.

Future work will encompass introducing a dynamic hierarchical schema generation procedure, implementing a sensor

³The notation \sum_i denotes a summation over a specific subset of task indexes, while \sum_i^n represents the summation over all task indexes.

drift detection mechanism, and extending the on-device learning process to initialize weights based on similar tasks.

ACKNOWLEDGMENT

This work was carried out in the EssilorLuxottica Smart Eyewear Lab, a Joint Research Center between EssilorLuxottica and Politecnico di Milano.

REFERENCES

- [1] Antonio Bevilacqua, Kyle MacDonald, Aamina Rangarej, Venessa Wijaya, Brian Caulfield, and Tahar Kechadi. Human Activity Recognition with Convolutional Neural Networks. volume 11053, pages 541–552. 2019. arXiv:1906.01935 [cs, stat].
- [2] Aiguo Wang, Shenghui Zhao, Chundi Zheng, Huihui Chen, Li Liu, and Guilin Chen. HierHAR: Sensor-Based Data-Driven Hierarchical Human Activity Recognition. *IEEE Sensors Journal*, 21(3):3353–3365, February 2021.
- [3] Aiguo Wang, Guilin Chen, Xi Wu, Li Liu, Ning An, and Chih-Yung Chang. Towards Human Activity Recognition: A Hierarchical Feature Selection Framework. *Sensors*, 18(11):3629, November 2018.
- [4] Oresti Banos, Miguel Damas, Hector Pomares, Fernando Rojas, Blanca Delgado-Marquez, and Olga Valenzuela. Human activity recognition based on a sensor weighting hierarchical classifier. *Soft Computing*, 17(2):333–343, February 2013.
- [5] Heeryon Cho and Sang Yoon. Divide and Conquer-Based 1D CNN Human Activity Recognition Using Test Data Sharpening. *Sensors*, 18(4):1055, April 2018.
- [6] Yanglin Pu, Yongqiang Jiang, and Hai-Miao Hu. HR-HAR: A hierarchical relation representation for human activity recognition based on Wi-Fi. *IET Communications*, 17(1):29–44, January 2023.
- [7] tinymml.org. Tinymml foundation, 2023.
- [8] Massimo Pavan, Eugeniu Ostrovan, Armando Caltabiano, and Manuel Roveri. TyBox: an automatic design and code-generation toolbox for TinyML incremental on-device learning. *ACM Transactions on Embedded Computing Systems*, page 3604566, June 2023.
- [9] Haoyu Ren, Darko Anicic, and Thomas Runkler. TinyOL: TinyML with Online-Learning on Microcontrollers, April 2021. arXiv:2103.08295 [cs, eess].
- [10] Michele Craighero, Davide Quarantiello, Beatrice Rossi, Diego Carrera, Pasqualina Fragneto, and Giacomo Boracchi. On-Device Personalization for Human Activity Recognition on STM32. *IEEE Embedded Systems Letters*, pages 1–1, 2023.
- [11] Laith Alzubaidi, Jinshuai Bai, Aiman Al-Sabaawi, Jose Santamaría, A. S. Albahri, Bashar Sami Nayyef Al-dabbagh, Mohammed A. Fadhel, Mohamed Manoufali, Jinglan Zhang, Ali H. Al-Timemy, Ye Duan, Amjed Abdullah, Laith Farhan, Yi Lu, Ashish Gupta, Felix Albu, Amin Abosh, and Yuantong Gu. A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications. *Journal of Big Data*, 10(1):46, April 2023.
- [12] Pierre-Emmanuel Novac, Alain Pegatoquet, Benoît Miramond, and Christophe Caquineau. UCA-EHAR: A Dataset for Human Activity Recognition with Embedded AI on Smart Glasses. *Applied Sciences*, 12(8):3849, January 2022.
- [13] Jorge-Luis Reyes-Ortiz, Luca Oneto, Alessandro Ghio, Albert Samá, Davide Anguita, and Xavier Parra. Human Activity Recognition on Smartphones with Awareness of Basic Activities and Postural Transitions. In Stefan Wermter, Cornelius Weber, Włodzisław Duch, Timo Honkela, Petia Koprinkova-Hristova, Sven Magg, Günther Palm, and Alessandro E. P. Villa, editors, *Artificial Neural Networks and Machine Learning – ICANN 2014*, volume 8681, pages 177–184. Springer International Publishing, Cham, 2014.
- [14] Human Activity Recognition: A Survey. *Procedia Computer Science*, 155:698–703, January 2019.
- [15] L. Minh Dang, Kyungbok Min, Hanxiang Wang, Md. Jalil Piran, Cheol Hee Lee, and Hyeonjoon Moon. Sensor-based and vision-based human activity recognition: A comprehensive survey. *Pattern Recognition*, 108:107561, December 2020.
- [16] Khimraj, Praveen Kumar Shukla, Ankit Vijayvargiya, and Rajesh Kumar. Human Activity Recognition using Accelerometer and Gyroscope Data from Smartphones. In *2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3)*, pages 1–6, February 2020.

- [17] Oresti Baños, Juan Manuel Galvez, Miguel Damas, Alberto Guillén, Luis Javier Herrera, Héctor Pomares, and Ignacio Rojas. Evaluating the effects of signal segmentation on activity recognition. In Ignacio Rojas and Francisco M. Ortuño Guzman, editors, *International Work-Conference on Bioinformatics and Biomedical Engineering, IWBBIO 2014, Granada, Spain, April 7-9, 2014*, pages 759–765. Copicentro Editorial, 2014.
- [18] José Antonio Santoyo-Ramón, Eduardo Casilari, and José Manuel Cano-García. A study of the influence of the sensor sampling frequency on the performance of wearable fall detectors. *Measurement*, 193:110945, April 2022.
- [19] Lei Gao, A. K. Bourke, and John Nelson. Evaluation of accelerometer based multi-sensor versus single-sensor activity recognition systems. *Medical Engineering & Physics*, 36(6):779–785, June 2014.
- [20] Manuel Roveri. Is Tiny Deep Learning the New Deep Learning? In Rajkumar Buyya, Susanna Munoz Hernandez, Ram Mohan Rao Kovvur, and T. Hitendra Sarma, editors, *Computational Intelligence and Data Analytics*, volume 142, pages 23–39. Springer Nature Singapore, Singapore, 2023.
- [21] Colby R. Banbury, Vijay Janapa Reddi, Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang, Robert Hurtado, David Kanter, Anton Lokhmotov, David Patterson, Danilo Pau, Jae-sun Seo, Jeff Sieracki, Urmish Thakker, Marian Verhelst, and Poonam Yadav. Benchmarking TinyML Systems: Challenges and Direction, January 2021. arXiv:2003.04821 [cs].
- [22] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Tijmen Blankevoort. A White Paper on Neural Network Quantization, June 2021. arXiv:2106.08295 [cs].
- [23] James O’ Neill. An Overview of Neural Network Compression, August 2020. arXiv:2006.03669 [cs, stat].
- [24] François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions, April 2017. arXiv:1610.02357 [cs].
- [25] Fisher Yu and Vladlen Koltun. Multi-Scale Context Aggregation by Dilated Convolutions, April 2016. arXiv:1511.07122 [cs].
- [26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition, December 2015. arXiv:1512.03385 [cs].
- [27] D. Anguita, Alessandro Ghio, L. Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *The European Symposium on Artificial Neural Networks*, 2013.