IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 19 June 2023, accepted 4 July 2023, date of publication 10 July 2023, date of current version 18 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3294097

RESEARCH ARTICLE

A Microcontroller-Based Platform for Cognitive Tracking of Sensorimotor Training

MATTEO ANTONIO SCRUGLI[1], **BOJAN BLAŽICA**[2], **LUIGI RAFFO**[1], **(Member, IEEE), AND PAOLO MELONI**[1]
[1]Department of Electrical and Electronic Engineering (DIEE), University of Cagliari, 09124 Cagliari, Italy
[2]Computer Systems Department, Jožef Stefan Institute, 1000 Ljubljana, Slovenia

Corresponding author: Matteo Antonio Scrugli (matteo.scrugli@unica.it)

This work was supported in part by the Sardegna Ricerche, Bando "PROOF of CONCEPT - Valorizzazione dei Risultati Della Ricerca in Biomedicina" - PO FESR 2014-2020 - Deep-ECGEE Project; in part by Slovenian Research Agency (research core grant number P2-0098); and in part by the D-Rabbit Project, a Cross-Domain Technology Transfer Experiment funded by the European Union's Horizon 2020 Research and Innovation Program under the SMART4ALL under Grant 872614.

ABSTRACT A new generation of fitness trackers is pervasively invading different aspects of our life, taking profit from wireless technology, embedded sensors, and increasingly accurate AI-based data analysis. The most crucial aspects concerning the design of these systems include energy efficiency and accuracy. In this paper, we propose a system relying on two microcontroller-based sensor nodes to track the physical activity during sensorimotor training, a type of exercise that challenges the user's balance skill, which has been proven to be very effective in improving performance, preventing injuries and recovering from them. One of the sensor nodes is integrated into a custom wobble-board and the second is wearable by the user. The nodes are adaptable to be set in different operating modes, depending on the use case needs, enabling different steps of near-sensor pre-processing. The most power-efficient operating mode executes a CNN-based analysis directly on the microcontroller, to recognize physical exercises. The algorithm provides an accuracy of respectively 99.4% and 97.6% on the two nodes. In-place execution of the CNN saves up to 65% power consumption with respect to the transmission of raw data for on-cloud analysis.

INDEX TERMS Adaptive system, fitness activity tracking, sensorimotor training, low power electronics, neural network, remote sensing, runtime.

I. INTRODUCTION

In recent years, fitness tracking has become increasingly important in healthcare and sports. Wearable wireless sensors can be designed to track physical exercises and monitor performance, taking profit from the widespread availability of microcontrollers and sensors of unprecedented quality, as well as from the increasing accuracy and flexibility of artificial intelligence (AI) and neural networks, opening up new uses cases and possibilities.

In this paper, we focus on sensorimotor training, a type of physical exercise that challenges an individual's balance, coordination, proprioception, and reaction time through a series of progressively difficult exercises.

The associate editor coordinating the review of this manuscript and approving it for publication was Vyasa Sai.

Sensorimotor training is often used in rehabilitation settings to help recover from pathological conditions or by athletes and fitness enthusiasts to enhance their performance and prevent injuries.

A specific research problem, common to other tracking tasks, stems from the need for wearable and accurate sensor solutions to support this kind of training. Despite being very effective, sensorimotor training exercises can be repetitive and challenging, which can lead to a lack of motivation and a high dropout rate among trainees.

A useful solution could employ a methodology reliant on sensor-based tracking devices. This allows precise monitoring of the training and adequate reporting, enabling smoother and more effective interactions between athletes/patients and coaches/trainers. This method can be easily used to implement gamification, making sensorimotor training more engaging and enjoyable.

To simultaneously ensure adequate accuracy and portability, we plan to combine smart microcontroller-endowed sensor nodes and AI algorithms. These will be meticulously optimized to execute on resource-constrained processing platforms. In more detail, in this paper, we present the design of a hardware and software processing system composed of two sensor nodes, each executing an AI-based algorithm for exercise classification at the edge. The proposed system includes a balance board with an integrated sensor to detect movement and execution of pre-defined exercises. Additionally, a companion node worn by the athlete is capable of classifying exercises executed while standing on the balance board.

Each node is implemented to execute an analysis of the sensor data at-the-edge, processing the acquired signals on the microcontroller, reducing the power consumption required for communication, and improving responsiveness. We use SensorTile, a low-power microcontroller-based module integrating inertial sensors commercialized by STMicroelectronics. As a data-analysis algorithm, we use convolutional neural networks (CNNs), nowadays commonly adopted in embedded classification tasks for being highly accurate and feature-free. Moreover, we execute an adaptive firmware that adapts the power consumption to different operating modes to save battery lifetime, changing the hardware setup of the processing platform by adapting power-relevant settings such as clock frequency, supply voltage, and peripheral gating.

As the main finding of our work, we show, with our experimental results, that lightweight CNNs, executable at-the-edge on a microcontroller node, can provide the required accuracy for the task of supporting sensorimotor training while providing adequate portability. We also show that, taking profit from in-place data analysis and adaptive system management, it is possible to increase significantly power/energy efficiency and improve battery lifetime. Moreover, we provide a detailed breakdown of the contributions to the overall power consumption to highlight the benefits of edge-based execution and to enable the reuse of our results in different domains.

The remainder of this paper is organized as follows: In Section II, we delve into the related work, exploring the current state of research on wireless fitness trackers implementing edge processing. We also review existing solutions for energy efficiency and event detection accuracy in wearable devices. Moving on to Section III, we provide an overview of our proposed microcontroller-based wearable device, describing the overall architecture and how it interacts with the wobble board and the mobile application. We preliminarily show the software-hardware structure of the node. In Section IV, we discuss the design of the application, explaining the available operating modes. We offer a detailed description of the neural network model used in the device. Section V presents the experimental results, showcasing the outcomes of our evaluation of the neural network's accuracy and its impact on power consumption. Finally, in Section VI, we draw conclusions by summarising the main findings and contributions of our work, emphasizing the key role of cognitive on-edge processing.

II. RELATED WORK

In literature, the field of study in computer science and artificial intelligence that applies machine learning algorithms to identify and classify different types of human activities based on sensor data is usually referred to as Human Activity Recognition (HAR). In general, HAR methods can be divided into two main groups: skeletal-based and inertial sensor-based methods.

Skeletal-based HAR uses sensors to capture 3D skeletal data from a person's body and then uses algorithms to recognize the activity being performed. Since the release of *Microsoft Kinect depth sensor and body tracking SDK* in 2010, real-time and accurate RGB-based human pose estimation techniques have been developed [1], [2], [3], [4]. This type of HAR is useful for applications such as healthcare, sports, and gaming. It can be used to monitor the health of a person, track the performance of athletes, and provide feedback to gamers. However, the use of devices such as the Kinect and cameras does not make this methodology sufficiently versatile and user-friendly. Due to the different technological nature of these kinds of approaches, they are not directly comparable to our results and have to be considered complementary instruments.

On the other hand, inertial sensors measuring acceleration, angular velocity, or magnetic field strength are small, low-cost, and low-power. By combining data from these sensors, HAR systems can accurately recognize and classify human activities such as walking, running, and jumping. In particular, the authors in [5] show how inertial sensor-based devices have many advantages over vision-based methodologies, which pose privacy issues and are limited by computational requirements. Accelerometer, gyroscope, and magnetometer sensors are the most commonly used sensors in literature [5], [6].

More recent work has focused on sophisticated and accurate algorithms, such as those based on artificial intelligence or deep learning provide impressive performance in human activity recognition challenges. Some of the most accurate approaches adopt Convolutional neural networks (CNNs) as analysis algorithms. In [7] the authors propose an approach to leverage multiple data sources for robust and accurate activity segmentation, exercise recognition, and repetition counting, achieving good results in terms of accuracy with multimodal deep learning models. In [8], the authors compare in terms of accuracy of results some classical signal processing techniques with deep learning models, showing how the latter leads to better results. They use inertial sensors plus an ECG in order to recognize ten manipulative gestures performed in a car maintenance scenario. The authors in [9] apply a CNN-based model to recognize eight different types of activities: still, walk, run, bike, bus, car, train, and subway. As in other works in literature, the authors use pre-processed

TABLE 1. **Comparison of different studies working on human activity detection using mainly microcontrollers and convolutional neural networks.**

Work	Dataset	Activity	Classess	Accuracy in %	Device	Model	Power consumption
Coelho et al. [10]	MHEALTH	body motion and physical activities	3 - 9	97.27	STM32F4-based	CNN	11.95 mW - 67.78 mW only MCU during inference
Dao et al. [11]	Bao et al. [12]	daily activities	13	99.412	ESP32-based	Random Forest	∼ 60mW
Ghibellini et al. [13]	custom	fall detection and physical activities	3	97	Arduino BLE 33 Sense	CNN + LSTM	–
Wenzheng et al. [14]	WISDM	daily activities	5	93.25	STM32L4-based	CNN	–
Warunsin et al. [15]	MobiAct	fall	13	96.55	ESP32-based	CNN	–
Belousov et al. [16]	WISDM	daily activity	6	91.81	STM32F4-based	CNN	–
Daghero et al. [17], [18]	UCI HAPT - WISDM - UniMiB-SHAR - WALK	various	12 - 6 - 17 - 2	85.63 - 98.81 - 86.24 - 95.74	RISC-V MCU Quentin based	CNN	∼ 3.8 mW only MCU
Our work	custom	physical activities	3 - 5	99.427 - 97.652	SensorTile (STM32L4-based)	CNN	∼3 mW

data as input to the neural network. In our work, we avoided this approach so as not to overly impact the workload on the resource-limited platform we selected. *As a main difference with our work, these approaches are based on CNNs executed remotely on high-performance computing platforms. To make HAR more pervasive, requirements in terms of power consumption, privacy, and responsiveness must be taken into account, enabling at-the-edge processing of the sensor data.*

Other examples in literature are more aligned with our work, as they execute the algorithm at-the-edge, dealing with the efficiency of resource-constrained platforms. The main characteristics of these solutions are presented in Table 1.

In most of the studies, HAR is aimed at tracking the behavior of the user and monitoring his daily activities. In [11], the authors promote a real-time human activity recognition system based on wearable devices. Classification is done directly on the device, and the results can be accessed via the Internet. The authors use the Random Forest algorithm, the dynamic window method applied by the proposed model allows to change in the data sampling time and increases the performance in activity classification. Experimental results show that the proposed system can classify 13 activities with a high accuracy of 99.4%.

In [14], the authors use an accelerometer sensor and a neural network to recognize personal activity behaviors. Using ST Microelectronics' X-CUBE-AI development tool, the trained model is imported into the firmware of the low-power microcontroller to realize the classification and recognition of activities such as jogging, standing, walking, standing, and climbing stairs. Experimental results show that this method can detect such classes with an accuracy of 93.25%.

In [17] and [18], the authors bridge the gap between on-device HAR and deep learning by proposing a set of efficient one-dimensional convolutional neural networks (CNNs) that can be implemented on general-purpose microcontrollers (MCUs). The proposed CNNs are achieved by combining hyper-parameter optimization with sub-byte and mixed-precision quantization to find a good trade-off between classification results and memory occupancy. In addition, the adaptive inference is exploited as orthogonal optimization to adjust the complexity of inference according to the input processed, making the HAR system more flexible. They achieve good results both in terms of memory occupancy, which varies between 0.05 and 23.17 kB, and in terms of power consumption.

In [16], the authors present a case study in which the use of a pre-trained CNN feature extractor under realistic conditions is evaluated. The case study consists of two main steps: (1) evaluation of different topologies and parameters to identify the best candidate models for HAR, thus obtaining a pre-trained CNN model, and (2) use of the pre-trained model as a feature extractor, evaluating its use with a real large-scale dataset. The results show an accuracy of 91.81%.

Another promising task exploiting HAR is the detection of risk-related events at home or in workplaces. In [15], the authors developed a fall detection system using an accelerometer as a sensor and a deep learning algorithm for fall pattern recognition in Ambient Assisted Living (AAL). The authors used an ESP32 microcontroller; if it detects a fall, it sends a fall alert to the provider via Wi-Fi and the LINE application. The proposed fall detection model demonstrated 96.55% accuracy. In [13], the authors propose the use of on-board artificial intelligence (AI) to move data analysis closer to the sensing units, thereby reducing the impact on user privacy and network load of the mobile or wearable device. The authors developed a HAR system based on a low-power microcontroller unit to detect critical movements (e.g., falls, runs) in industrial environments. They use a DL model based on convolutional neural networks (CNNs) and a quantization technique to reduce the model size. Preliminary results show 97% accuracy for the CNN model, outperforming non-DL techniques, and a 53% reduction in model size due to quantization.

Finally, another exploitation sector is fitness tracking, where near-sensor AI is used to classify physical exercises and sports activities. In [10], the authors present a lightweight framework for human activity recognition on low-power devices, examining the impact of different system configurations and complexity of deep learning models on computational cost and energy consumption. This study highlights the importance of developing energy-efficient deep learning models to realize effective mobile HAR applications.

Although all the mentioned studies provide a useful starting point for understanding the necessary level of accuracy in human activity recognition (HAR) and suggest that CNN is a promising algorithm to use, they cannot be directly applied to sensorimotor training. To the best of our knowledge, our work is the first that focuses on supporting such a kind of training. Additionally, none of these studies offer a thorough analysis of the energy-related factors involved in the system, which is key to understand when on-board processing is actually beneficial.

The usefulness of sensor-assisted tracking of sensorimotor training is suggested by several works in literature. In [19], the authors highlight in a critical review how patient motivation has been taken into account in relation to rehabilitation for strokes, fractures, rheumatic disease, aging, and cardiac and neurological issues. The monotonous nature of the ankle exercises and the inability to track one's progress during the training process may, at least in part, account for some people's lack of motivation [20]. To encourage people to use a wobble board, which is crucial for ankle rehabilitation, the authors in [21] and [22] suggest using it in creative ways, enabling gamification and making the fitness activity process more accessible.

In our work, we take the mentioned approach one step further, adding AI-based on-board processing and using a companion sensor for detecting body-weight exercises executed when standing on the board. This increases the variety of training routines that can be assigned to the user and enables to create a more engaging training experience.

Summarizing, our work advances state-of-the-art approaches in the following respects:

- it is the first work using AI-based at-the-edge HAR for sensorimotor training, with power consumption and accuracy aligned with other state-of-the-art HAR approaches;
- it is the first work that uses inertial sensors to enable simultaneously the classification of the wobble board movement and the recognition of bodyweights exercises executed while standing on the board;
- combines state-of-the-art deep learning techniques and dynamic management of hardware and software to minimize power consumption;
- demonstrates the usefulness of on-board processing by studying all the contributions to power consumption, including processing and data communication.

While a preliminary presentation of our approach was presented in [23], in this paper, we further advance the system capabilities, since:

- combined with the wobble board, we use the same deep learning approach for the recognition of physical exercises such as squats and push-ups through the use of a companion wearable device placed on the arm, exploiting the creation of a custom adequate dataset;
- we present in detail the impact in terms of power consumption and system accuracy regarding the extension of new features;
- we describe the engineering of the produced wobble board and smartphone client application.

III. SYSTEM OVERVIEW

The objective of the system presented in our work is to track the activity and exercises performed during sensorimotor training. This kind of training typically exploits a wobble board, a balance board that is designed to be unstable and challenging to stand on. Common sensorimotor training exercises require adopting a particular posture on the wobble board or making precise adjustments to body weight to maintain balance and perform specific movement patterns. More advanced routines envision the execution of dynamic workout exercises, such as squats, push-ups, and lunges while standing on the balance board.

Exploiting sensor nodes to monitor and evaluate user performance, our system can be connected to an interactive user interface, to make the sensorimotor training experience more engaging for users, to foster their commitment to their training or rehabilitation program.

Figure 1 visually represents the components of the system. At the core of the tracking functionality, we use two wireless sensor nodes, which are small battery-powered microcontroller-based devices that gather data from sensors and connect via Bluetooth low-energy to transmit/receive data from the rest of the system.

The first sensor node, called hereafter *board sensor node*, is embedded in a custom wobble board. The sensor in the board tracks its movements and provides data on the athlete's performance. The wobble board is extensively described in Section III-A. The second sensor node, called hereafter *body sensor node*, is designed to be worn on the athlete's forearm. It tracks the athlete's movements during exercises like squats and push-ups, providing data on the number of repetitions and the quality of the exercise. We chose the upper limbs as the

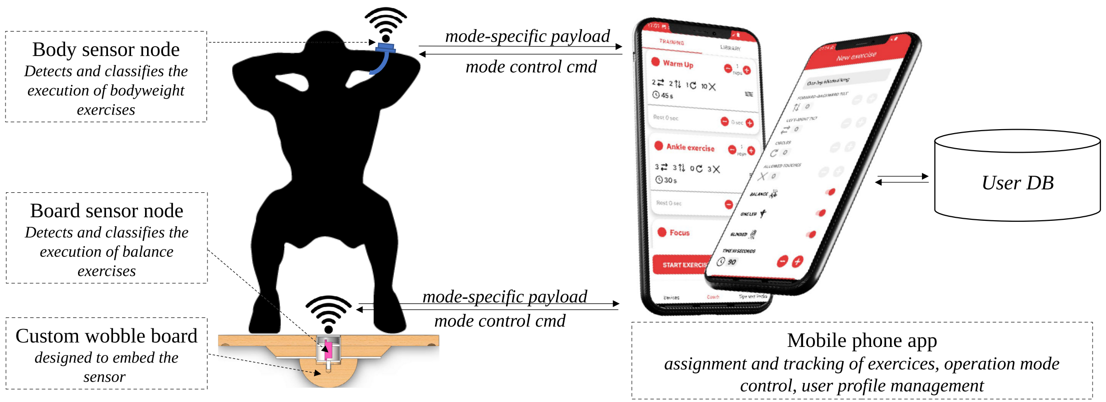

Body sensor node
Detects and classifies the execution of bodyweight exercises
Board sensor node
Detects and classifies the execution of balance exercises
Custom wobble board
designed to embed the sensor
mode-specific payload
mode control cmd
mode-specific payload
mode control cmd
User DB
Mobile phone app
assignment and tracking of exercises, operation mode control, user profile management

FIGURE 1. A graphical representation of the system architecture, consisting of two IoT sensor nodes that gather data from various sensors and transmit it via Bluetooth low-energy.

sensor placement because of their ability to better represent several physical exercises, including those covered in the present work and others that could be added for detection in the future. This placement was chosen to allow the sensor to be worn comfortably and to minimize invasiveness while performing the exercises.

Both sensor nodes, described in further detail in Section III-B, are designed to execute some on-board pre-processing of the sensed data. To conserve battery power, the sensor nodes have a hardware and software architecture that enables real-time reconfiguration, to adjust processing resources on-the-fly setting the system in a specific operating mode, depending on the modality chosen by the user.

Sensor nodes communicate with a dedicated mobile phone app, described in Section III-C. The app collects the data from both sensors and displays it to the user. It can also be used by a coach or trainer to assign specific training routines to the athlete. Depending on the selected training modality, the app is in charge to send operating mode change commands to the nodes, thus expecting a differently pre-processed payload for each chosen configuration. The mobile application serves as a main user interface. First, the tracking data can be used by coaches and trainers to monitor the athlete's progress over time and adjust their training program as needed, even remotely. Second, it can share the athlete's results with a cloud database, to allow the athletes to compete against other users in a virtual leaderboard or to earn rewards for achieving certain milestones. Moreover, the app can use the board as a controller for simplistic video games.

A. CUSTOM WOBBLE BOARD

The basic idea behind sensorimotor training is to have the trainee try to be as stable as possible on unstable surfaces such as Bosu balls or wobble boards while introducing increasing difficulties. For example, standing on one leg instead of two, standing with eyes closed, trying to catch a ball while standing on a wobble board, trying to perform a squat without the board touching the ground on the edge, etc. The physical design of the board impacts how difficult the exercises on the board will be. A half sphere underneath the board allows movement in all directions, while a half cylinder only in two, making the exercise easier and more specific. Next, the inclination and size of the sphere control the sensibility of the whole board; a slightly flattened half-sphere will be more forgiving of small oscillations in balance by the trainee.

We have designed and physically prototyped a custom wobble board for the purpose of this research. Similarly to the microcontroller, we opted for an adaptable design also for the board itself. Figure 2 shows the building blocks of the board: 1) the upper part upon which the trainee stands consists of a wooded plate or a standard 10 kg bumper, 2) a placeholder for the sensor, so that it can be removed and used for recognition of activities not related to the wobble board, and finally, 3) interchangeable lower part (currently cylinder or sphere, others could be added later on). The possibility to change the geometry of the board is an additional requirement for the exercise recognition algorithms.

B. SENSOR NODE ARCHITECTURE

The architecture that we designed for the sensor nodes is a layered structure composed of hardware, firmware, and software components.

Regarding the hardware, we selected SensorTile by STMicroelectronics, a microcontroller-based low-power platform with an ARM Cortex-M4 32-bit. The LSM303AGR accelerometer sensor, integrated into the SensorTile, is optimally configured for our applications at a 100 Hz sampling rate, to ensure accurate data acquisition and analysis. As mentioned, communication with the sensor node happens wirelessly via the Bluetooth Low Energy standard implemented by an on-board dedicated BlueNRG-MS module, and the device is battery-powered with a Lithium Polymer battery. The overall size of the SensorTile node is 13.5 × 13.5 mm.

The firmware layer exploited in the sensor node is essential for managing the data collected by the hardware, ensuring efficient operation and communication. To enable

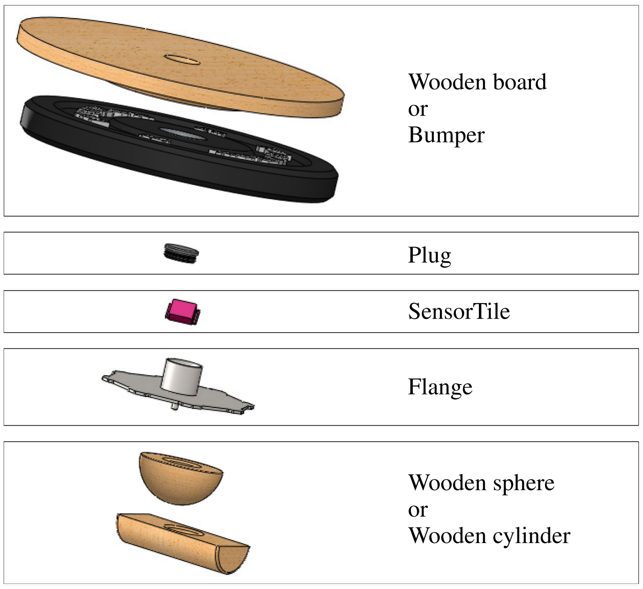


FIGURE 2. Adaptability of the wobble board with interchangeable bottom parts for different levels of difficulty.

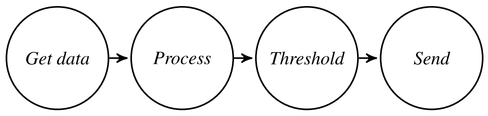


FIGURE 3. Basic tasks identified in our model, providing the foundational operation structure within our system.

task-level abstraction, we use Free Real-Time Operating System (FreeRTOS) on the node. This allows to activate/deactivate real-time tasks without having a significant impact on the application's memory footprint. The operating system is between 4 and 9 kB in size. It includes features such as real-time scheduling, interprocess communication, synchronization, and time measurement. With the support of this system, it is possible to create the processing tasks to be performed on the platform as dynamically activatable threads and readily manage their scheduling at runtime. Moreover, a key component of the firmware layer is the CMSIS, in particular, CMSIS-NN [24], that we exploit for the efficient execution of neural network processing.

Regarding the software layer, we have devised an application structure based on a network of processes. Tasks are represented as distinct processes that communicate with each other using FIFO structures and blocking read-and-write communication primitives. Using a software pipeline, processes can run concurrently if sufficient processing resources are available, potentially improving performance. To enable adaptivity of the system, we create a series of tasks that operate on the sensed data for each sensor that needs to be monitored (Figure 3). For each sensor node, a chain of processes is created. Processes can be dynamically activated or deactivated according to their utility, the morphology of the chain depends on the selected operating mode. For each sensor, we devise four general types of tasks:

- *Get data task:* takes care of taking data from the sensing hardware integrated into the node;
- *Process task:* this type of task can be used to perform different stages of a data processing algorithm. By having different levels of processing available, a potential user can choose the level of analysis, which affects the required communication bandwidth, the level of detail of the information collected, and the amount of power/energy used;
- *Threshold task:* With this task data can be filtered, its purpose is to limit the node's ability to send data. It allows data to be transferred to the cloud only in case of certain events or alarm circumstances, e.g., sending the processed data if and only if they are collocated within a predetermined range of values;
- *Send task:* is the task responsible for outward communication.

This allows users to choose from a range of application configurations, using different operating modes with varying degrees of in-place computation, bandwidth requirements, and monitoring accuracy.

As an additional component of the software layer, we have developed ADAM (ADAptive runtime Manager), which is executed as an additional periodic thread on the platform and takes care of the management of dynamic hardware and software reconfiguration. Although a timer is used by default within ADAM to activate it periodically, it is also possible to activate it under other circumstances, such as reconfiguration commands received from the mobile application or other relevant events (e.g. change of the battery status).

Under these conditions, ADAM is able to react by performing certain actions:

- Change the operating frequency of the microcontroller to increase or decrease performance;
- Activate or deactivate each task in the sensor task chain separately or all together;
- Redirect the FIFO-managed data flow according to the current operating mode;
- Decide whether or not to put the microcontroller in sleep mode.

For example, to switch from an operating mode that sends pre-processed data to the cloud, to one that sends raw data, ADAM can reconfigure the system by disabling a processing task, as shown in Figure 4.

C. USER INTERFACES: MOBILE APPLICATIONS AND GAMES

Following the basic idea behind sensorimotor training describe earlier, we devised a set of possible user interfaces that permit interaction with the system to implement an enjoyable and more effective training experience. A first UI is a simple *Training* mobile application targeting coaches and personal trainers. A mobile app is a widely available and easy portable implementation for the user interface, directly

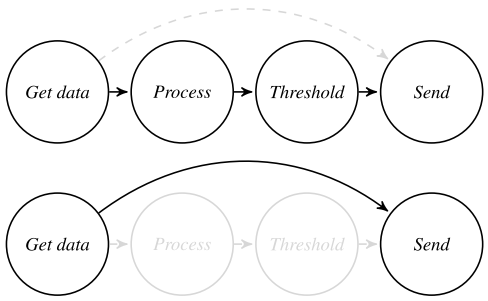


FIGURE 4. Task representation of a generic system capable of morphological changes in the running process network, facilitated by ADAptive runtime Manager.

fostering the pervasive use of the sensors by a very vast scope of possible users. A second kind of interface receives messages from the sensor node so that the board can be used as a controller to play some games.

1) TRAINING APP

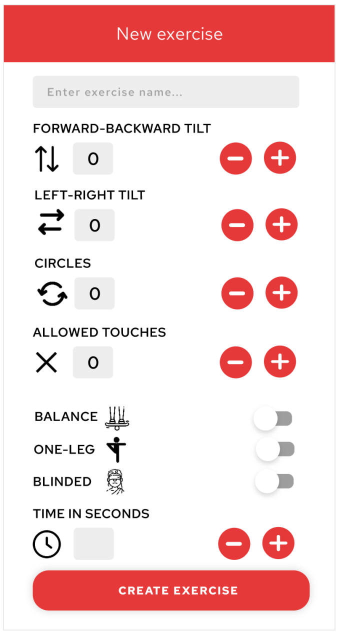
The application enables them to create training sessions by defining and combining different elements of the training session. The screenshot from Figure 5 highlights these elements: training duration, number of allowed touches, number of requested tilts front and back and/or left and right, number of requested circles on the board, one-legged execution, blindfolded execution. When this kind of use is selected, the app needs to receive a message from the board sensor every time one of the requested exercises is performed. Thus the app sends a reconfiguration message to the board sensor node to set it in an adequate operating mode, that performs in-place classification of the movement (such mode will be indicated as *Classification* mode in the following) and simply counts repetitions.

Another possible option is to request the trainee to hold as still as possible and the application will give a score on how well the trainee performed. In this case, the app needs to receive an indication of the inclination of the board over time and provides a score depending on how much and how often inclination changes. An adequate reconfiguration message is sent to the sensor node to set it in the required operating mode (see *detection* mode hereafter).

A session is successfully completed if all the requested criteria are met. Each so-defined session can be saved for later reuse and exercises can be combined into sets so it is possible to define a complete training session composed of warm-up exercise, main training session, and relaxation exercises. Sets can also be saved for later reuse. Multiple users can be related in the user database to implement competition and rankings. An application add-on can count exercises classified by the body sensor.

2) GAMES

The board has been tested to work as a controller/joystick in different games. In this case, the board sensor is also set in an operating mode providing inclination and direction


New exercise

Enter exercise name...

FORWARD-BACKWARD TILT
0

LEFT-RIGHT TILT
0

CIRCLES
0

ALLOWED TOUCHES
0

BALANCE

ONE-LEG

BLINDED

TIME IN SECONDS

CREATE EXERCISE

FIGURE 5. Screenshot of a mobile application for the coach: composition of sensorimotor exercise with elements recognized on the edge by the microcontroller.

values, to be used to command the games. Some preliminary users have played with a Python version of Pacman implemented on a PC. Others have tried more advanced wobble-board action games developed with Unity using LEGO Microgame [25]. Playability has been verified, however, it is deeply impacted by the shape of the bottom layer of the wobble-board. A lower semisphere is easy to play with, while more difficult shapes require advanced user skills.

IV. SENSOR NODE SOFTWARE DESIGN

For the implementation of the *board* and *body* sensor nodes, we have set up the general previously described software application model to support three different operating modes, namely *raw*, *detection*, and *classification*. Figure 6 shows the three different operating modes configuration and which tasks participate in the process chain. The first from the top, operating mode *raw*, enables more detailed downstream analysis on the cloud by transmitting all data to it. This approach ensures comprehensive information is available for further processing but may require more bandwidth for data transfer. The second operating mode, *detection*, allows for basic edge analysis, sacrificing the detailed accelerometer trace data that could be processed later on the cloud. This mode reduces communication with the cloud, conserving bandwidth but offering limited analytical capabilities. Lastly, the operating mode *classification* combines the main advantages of the first two operating modes, providing detailed analysis directly on the device while simultaneously reducing the communication bandwidth with the cloud. By employing a Convolutional Neural Network (CNN) on the edge device, this mode strikes a balance between the need for detailed analysis and efficient data transfer, maximizing both processing capabilities and communication efficiency.

In further detail, Table 2 provides an overview of the operating modes for both sensors.

In the operating mode *raw*, the system does not perform any processing and transmits accelerometer data directly. The data for each accelerometer axis has a size of 2 bytes. To save power consumption when sending the data, it was decided to merge two acquisitions into a single Bluetooth packet, so a dimension of 4 bytes for each axis of the accelerometer will be shown in Table 2.

The operating mode *detection* detects the direction and inclination of the board or the absence of motion. If no movement is detected, the system will not proceed to the next task, which involves the use of the threshold task or CNN task.

The operating mode *classification* uses AI-based detection to identify various movements when a sensory node is placed on a wobble board or on the arm for body exercises. The different recognizable physical exercises are classified into distinct categories for board and body movements. For board movements, the class labels are B (Basic stance balance), S (Side tilt), FB (Forward/Backward), and R (Two leg tilts). For body movements, the class labels are SQ (Squats) and P (Push-ups). In both cases, the G label represents unmatched actions or lack of movement. An approach based on an AI-based analysis applied to data acquired by a simple and affordable sensor, delegates accuracy to the software instead of exploiting multiple sensors or more expensive hardware, improving the usability of the device.

A notable observation from Table 2, which will be further substantiated in Section V-B with power consumption findings, pertains to the data rate reduction when incorporating data processing. With the first level of processing, activated in the operating mode *detection*, there is a tenfold decrease in the volume of data transmitted externally. For the second level of processing, enabled by the *classification* operating mode, the data rate experiences at least a 40-fold reduction. This decline in data rate results in significant power consumption savings due to the diminished utilization of the wireless communication module.

A. NEURAL NETWORK DESIGN

To implement the classification of board and bodyweight exercises envisioned in the *classification* operating mode, due to their ability to capture complex patterns and hierarchical features from data [26], we chose to use a Convolutional Neural Network. CNNs are particularly well suited for working with raw signals, as they can automatically learn relevant features without the need for manual preprocessing. Furthermore, the use of CNN models for activity recognition applications is widely discussed and used in the literature, as reported in Section II, further confirming the validity of choosing deep learning for this system.

We have performed a design space exploration to select a convenient CNN topology, comparing with each other tens of

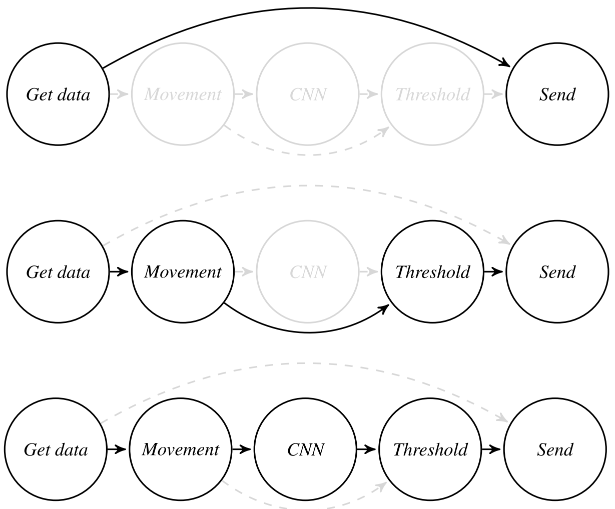


FIGURE 6. Specific representation of the task chain for each operating mode in our selected use case, offering insight into the practical implementation and functioning of our system. From top to bottom: the *raw* operating mode, the *detection* operating mode, and the *classification* operating mode.

different models featuring different number of layers, number of neurons per layer, and kernel size. Comparison has considered accuracy and computational complexity, taking into account the memory storage inside SensorTile and the possibility of being executed in real-time by the target processor. The selected network topology is displayed in Figure 7.

We partitioned the data into independent training, validation, and test sets, guaranteeing that the evaluation of the model's performance was on unseen data, thus providing a realistic assessment of its predictive power. The setup of the training environment is indicated in Table 3. We used a training set consisting of 70% of the total dataset elements, randomly selected. Additionally, 15% of the dataset is allocated for the validation set, and another 15% is reserved for the testing set. To verify and prevent overfitting in our deep learning approach, we implemented a few strategies. Primarily, we employed the early stopping (ES) algorithm. This technique halts the training process if an increase in the validation loss is detected for a specified number of consecutive epochs, known as the *patience* [26]. In our case, we chose a patience value of 5, meaning the training would cease if a loss increment is observed for five consecutive epochs. If the model's performance on the training set continues to improve while its performance on the validation set starts to deteriorate, it indicates overfitting. In such cases, the training process would be stopped early, thus preventing the model from overfitting.

Quantization is an important technique used to optimize neural networks, as it significantly reduces memory and computational requirements without significantly affecting the model's performance. The benefits of applying quantization include reduced memory footprint, lower power consumption, faster inference times, and the ability to exploit the SIMD (Single Instruction Multiple Data) capabilities of the

TABLE 2. Summary of operating modes for board and body movement detection.

Operating mode	Description	Board	Body
Raw	Processing level	No processing	
	Bluetooth sending rate	50 Hz	
	Bluetooth packet	4 byte X-axis + 4 byte Y-axis + 4 byte Z-axis + 4 byte timestamp	
	Data rate	800 B/s	
	Example usage	Detailed offline analysis; Dataset collection/enrichmen	
Detection	Processing level	• *Direction and inclination detection.* The first angle indicates the tilt of the board relative to the ground, while the second angle identifies the direction of the tilt. • *Motion detection.* Detection of motion below a predetermined threshold, within a specified tolerance level, to prevent activation of the next activity in case of undetected movement.	• *Motion detection.* Detection of motion below a predetermined threshold, within a specified tolerance level, to prevent activation of the next activity in case of undetected movement.
	Bluetooth sending rate	10 Hz	–
	Bluetooth packet	2 byte tilt direction + 2 byte inclination + 4 byte timestamp	–
	Data rate	80 B/s	–
	Example usage	Controller interacting with games; Providing input to the mobile app to grade the stability over time	
Classification	Processing level	• *AI-based: board exercise classification.* The neural network is capable of identifying various movements when a sensory node is placed on a wobble board, including different types of tilts and rotations, as well as accounting for other unexpected actions or lack of movement	• *AI-based: body exercise classification.* The neural network is capable of recognizing typical physical exercises when a sensory node is placed on the arm, currently identifying two specific exercises and a generic class for other movements.
	AI classes	• *Basic stance balance (B):* Upright torso, neutral spine, board balancing. • *Side tilt (S):* Weight shift, left-right tilt, controlled movement. • *Forward/Backward (FB):* Front-back tilt, slow and controlled. • *Two leg tilts (R):* 360-degree rotation, alternating directions. • *Other (G):* Unmatched actions or lack of movement.	• *Squats (SQ):* Involves bending the knees, lowering the torso with a straight back, and pushing the hips back before returning to a standing position. • *Push-ups (P):* Executed in the prone position, facing down, and entails raising and lowering the body by extending and flexing the arms. • *Other (G):* Unmatched movements.
	Bluetooth sending rate	$0 \sim 1$ Hz	$0 \sim 4$ Hz
	Bluetooth packet	1-byte AI classification + 4-byte timestamp	
	Data rate	$0 \sim 5$ B/s	$0 \sim 20$ B/s
	Example usage	Providing input to the mobile app for counting of exercise repetitions	

TABLE 3. Neural network hyperparameters used.

Hyperparameter	Value	Hyperparameter	Value
Training set	70%	Validation set	15%
Testing set	15%	Optimizer	Adam
Batch size	32	Learning rate	0.0001
Loss criterion	Cross Entropy	Betas	(0.9, 0.999)
ES patience	5	Eps	1e-08
ES evaluation	Every epoch	Weight decay	0

microcontroller, making it particularly suitable for deployment on resource-constrained devices such as microcontrollers. For deployment, we exploited static quantization [27] since the model inference is executed on the target processor using the CMSIS-NN optimized function library, which assumes 8-bit resolution inputs. Thus the deployment process requires a quantized CNN model, built by converting weights and biases from floating-point to integer format. To convert the weights, we introduced *MinMax* observers at the quantization phase, whose task is to analyze the outputs of each layer. We set the biases to zero to minimize the computational load on the node. This decision reduces the overall computational complexity of the neural network, which is particularly beneficial for resource-constrained devices as it allows for more efficient processing and lower power consumption. The observer returns a value of *scale* and *zero point* by analyzing the distribution of the outputs of each layer. They are used to resize the output of the various layers so that they can take advantage of the 8-bit resolution and avoid saturation. For

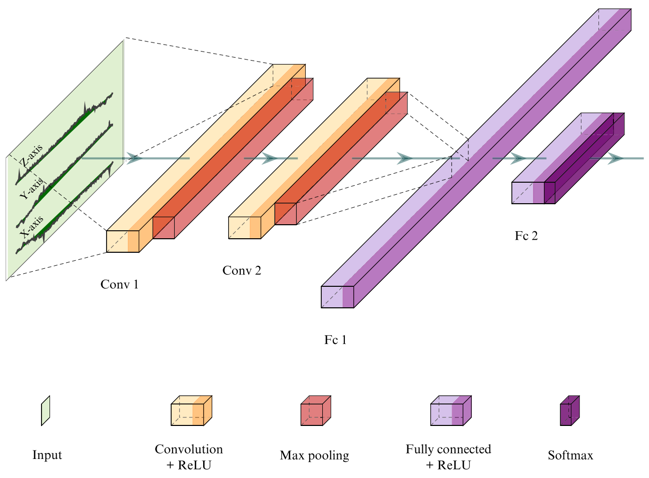


FIGURE 7. **CNN structure and classes description.**

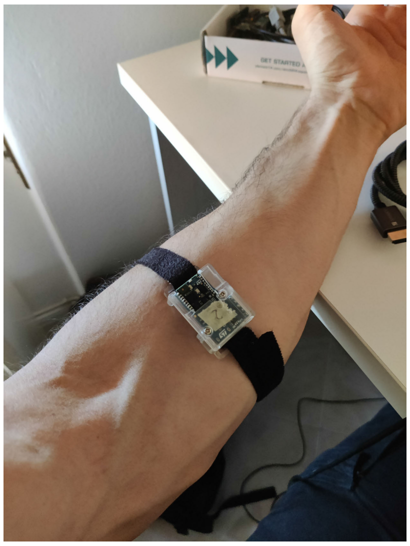


FIGURE 8. **Possible positioning of the sensory node.**

both convolutional and fully connected layers, CMSIS-NN provides the logical shift on the output. In this way, the output data is efficiently represented with only 8 bits. The quantization process in PyTorch, however, returns a value of *scale* that is not necessarily equal to a power of two, making the logical shift on the output not usable. Consequently, we made a small modification to the CMSIS-NN functions to adapt them to the PyTorch scale values, representable with *float* type variables. The inference execution time is slightly longer because of this change. We estimated an increase in execution time of 2.87% after testing inference with and without this modification.

The CNN layers are slightly different between the *board* and the *body* sensor nodes. Table 4 describes the differences between the two versions. A main difference is related to the input features: in the body version, all three axes are taken into account, while, in the wobble board version, only two axes, X and Y, are used, namely those parallel to the ground. For the *board* model, the total parameters amount to 98,460. Since the data is stored with 8-bit variables, each parameter requires 1 byte of memory. Thus, the memory requirement for the *board* model is approximately 98,460 bytes or around 96.11 KB. Similarly, for the *body* model, the total parameters amount to 34,600. The memory requirement for the *body* model is approximately 34,600 bytes or around 33.79 KB. As both models operate independently, each of them can fit within the 128 KB SRAM of the microcontroller. However, it is essential to consider that the firmware itself also requires memory, which reduces the available memory for the neural network models. In the case of the *board* model, the first Fully Connected Layer, which has 94,000 parameters, might not entirely fit within the available SRAM due to the memory consumed by the firmware. To address this issue, some of the weights from the first Fully Connected Layer can be stored in the microcontroller's flash memory. However, it is important to note that using flash memory might result in a slight decrease in performance, as accessing flash memory is generally slower than accessing SRAM.

1) DATASET

In literature, there are few datasets available that could be useful for our work. In [7], the authors make their dataset with numerous recordings of different exercises publicly available, including those selected in our work, namely squats, and push-ups. For each exercise, the authors provide several recordings, each made with a different device, i.e. camera, earbud, two smartwatches (one per arm), and two smartphones (different models). Although the dataset is well structured and contains numerous recordings, we chose to create one from scratch. The creation of a custom dataset was necessary because no publicly accessible dataset featured data collected from sensors placed at the same location specified in our study. The sensor placement was an important factor in our research, which required us to generate a dataset in line with our specific needs, rather than relying on an existing dataset that would not meet our technical requirements. We chose to place the sensor on the left arm as shown in Figure 8, as it is a good compromise between comfort and detection accuracy. Our dataset [28] includes 108 recordings that are generally one minute long but can vary between 30 seconds and 90 seconds. The exercises in question will be described in Table 2. The recordings are in *json* format and contain information regarding: the type of exercise; the number of samples in the recording; the timestamp; any notes; the samples and a list of events corresponding to the samples in which the repetitions are performed. We recruited eight members from a fitness gym, consisting of both males and females, aged between 25 to 50 years. These individuals exhibited a wide range of experience levels, from beginners to those with moderate experience. The number of subjects and recordings involved is comparable with other datasets available in literature, e.g. MM-Fit and MHEALTH, mentioned in Section II. The fact that more people were included, increases the generality of the dataset and thus the ability of the neural network to detect an exercise performed in different ways.

TABLE 4. Neural network model parameters for the wobble board and body version.

Layer	Input dimension		Output dimension		Input features		Output features		Kernel size		Number of parameters	
	Board	Body	Board	Body	Board	Body	Board	Body	Board	Body	Board	Body
Convolutional	215	76	207	72	2	3	20	20	9	5	360	300
Max pooling	207	72	103	36	20	20	20	20	2	2	–	–
Convolutional	103	36	95	32	20	20	20	20	9	5	3600	2000
Max pooling	95	32	47	16	20	20	20	20	2	2	–	–
Fully connected	940	320	100	100	–	–	–	–	–	–	94000	32000
Fully connected	100	100	5	3	–	–	–	–	–	–	500	300
Total number of parameters											98460	34600

TABLE 5. Augmentation parameters in the wobble board version.

Parameter	Value
Shifting, temporal distance between frames	0.25 s
Rotation, X and Y axis rotation	$\angle -4$, $\angle 0$, $\angle 4$
Time dilation, downsampling	6, 7, 8

TABLE 6. Augmentation parameters in the body version.

Exercise	Parameter	Value
Squat	Shifting, temporal distance between frames	From -0.5 s to 0.5 s with intervals of 0.1 s,
Push-up	Shifting, temporal distance between frames	From -0.2 s to 0.2 s with intervals of 0.1 s.

2) DATA AUGMENTATION & GENERALIZATION

The process of labeling the exercises in the dataset was done manually, centering labels as precisely as possible within the frames of samples corresponding to an exercise repetition. If no countermeasures are taken during training or during dataset preparation, this determines that the network classifies accurately only frames that are precisely coherent, in terms of centering, with the manually labeled frames used in training. In other words, the neural network is not robust to input temporal shifts.

The trivial solution to this issue is to execute the inference very frequently, to increase the probability of performing at least one inference on an adequately centered input frame, and to actually classify correctly each single exercise repetition. This solution obviously is not convenient when it comes to the power consumption of the device.

Another way to make the network more robust is to train it to recognize different exercises even with inputs that are not perfectly centered within the frame. This is possible by exploiting some data augmentation techniques on the dataset. Operations like shifting the training signal for a few samples in each direction can often greatly improve generalization [26]. Through the utilization of this data augmentation technique, it is possible to tolerate a lower execution frequency (f_{CNN}), improving the overall power efficiency of the system. On the other hand, an excessive degree of augmentation could lead to an increase in false positives. It is worth emphasizing that the data augmentation techniques, designed to increase the diversity and representation of our dataset, were applied exclusively to the training set. This procedure enriched the model's learning phase without introducing potential bias into the validation and test stages.

The parameters chosen for this type of augmentation can be seen in Table 5.

As for the squat and push-up exercises, only the shifting transformation was used. Note that in this case a distinction of shifting parameters was made according to the type of exercise, this is due to the different execution and duration of the exercise. The parameters chosen for this type of augmentation can be seen in Table 6.

V. EXPERIMENTAL RESULTS

In this section, we present an experimental research study aimed at evaluating the performance and power consumption of our proposed hardware/software model. The primary goal is to assess the effectiveness of our dynamically managed low-power node in performing in-place analysis of sensed data while maintaining high accuracy and low energy consumption. We hypothesize that the proposed model will be able to efficiently manage hardware and software reconfiguration, leading to significant energy savings.

To test our hypothesis, we have conducted experiments using the neural network for both wobble board and body versions. The experimental setup includes the microcontroller-based node running on-edge CNN processing and the ADAM component handling device reconfiguration. We also assessed the effects of using data augmentation techniques on accuracy and power consumption.

Our experiments involve the following steps:

- training the quantized neural network on a custom dataset;
- evaluating the accuracy of detecting wobble board movements and body exercises;
- assessing the power consumption for each selected operating mode;
- analyzing the energy savings achieved by enabling in-place analysis and efficient device management.

A. NEURAL NETWORK ACCURACY

We have first evaluated the classification accuracy obtainable by the CNN on the board sensor, comparing a floating-point and fixed-point implementation, when recognizing four types of exercises performed over the wobble board: basic stance balance, side tilt, forward/backward tilt, two leg tilts, and a

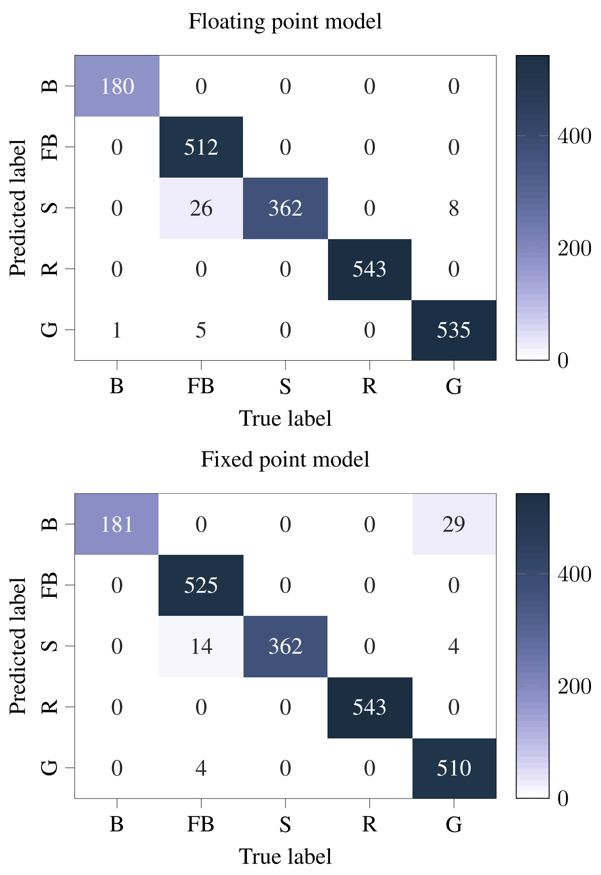


FIGURE 9. Neural network confusion matrix for the wobble board version. The classes were defined in Table 2.

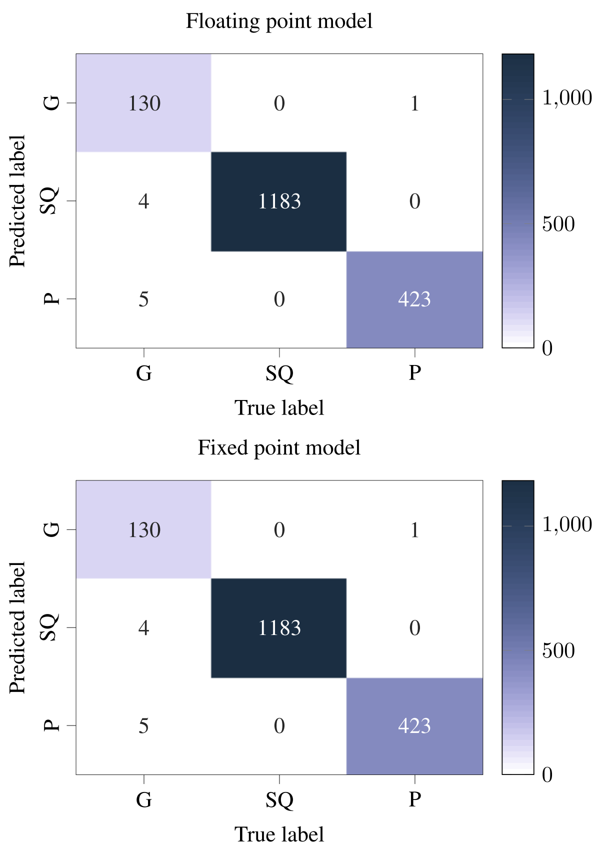


FIGURE 10. Neural network confusion matrix for the body version. The classes were defined in Table 2.

TABLE 7. Summary of neural network model accuracies.

Augmentation	Model type	Board	Body
No	Floating-point	98.02%	97.864%
	Fixed-point	97.05%	97.153%
Yes	Floating-point	98.158%	99.427%
	Fixed-point	97.652%	99.427%

generic movement. First, the neural network models were trained without utilizing any data augmentation techniques. The resulting accuracy achieved for the floating-point model was 98.02%, and for the fixed-point model was 97.05%. Subsequently, data augmentation techniques were incorporated into the training process, resulting in improved accuracy for the model. The resulting floating-point model achieved an accuracy of 98.158%, while the fixed-point model version achieved an accuracy of 97.652%. These results confirm that the incorporation of data augmentation techniques can lead to improved accuracy in neural network models and that the quantization process, although beneficial for computational reasons, introduced a slight degradation in accuracy due to the inherent trade-off between precision and efficiency. The corresponding confusion matrix for the obtained models is shown in Figure 9. The limitations of the quantized model mostly derive from the difficulty in distinguishing between the basic stance balance and generic movement classes due to the reduced numerical precision and approximations introduced during the quantization process.

Second, we have assessed the *body* version of the neural network, designed to classify between squats, push-ups, and a generic motion. A comparative analysis has been conducted on the training results before and after the implementation of data augmentation techniques. The neural network trained with the un-augmented dataset achieves an accuracy of 97.864% in the case of the floating-point model, with a reduction in accuracy down to 97.153% in the fixed-point case. After augmentation, the model achieved an impressive accuracy of 99.427%. The conversion process to fixed-point did not lead to a noticeable degradation in performance. Figure 10 shows the corresponding confusion matrix, which further supports the minimal impact of the conversion on the model's overall performance. The improved performance suggests that data augmentation has successfully enhanced the model's ability to generalize from the training set to unseen data, thus improving the robustness and reliability of our system.

Table 7 summarizes the results of performance analysis in terms of classification accuracy of neural networks based on the use of data augmentation techniques, model type (floating-point or fixed-point), and system considered (board or body). Augmentation techniques lead to increased accuracy for both systems. For instance, when comparing the floating-point model with augmentation (98.158% for Board and 99.427% for Body) to the one without augmentation (98.02% for Board and 97.864% for Body), there is a noticeable improvement in accuracy. A similar trend can be observed for fixed-point models. Regarding quantization, it is important to note that the process of converting floating-point numbers to fixed-point numbers can introduce some degree of information loss. This loss is generally due to the reduction in numerical precision, which may impact the model's ability to accurately represent and process the data. However, the table indicates that this impact on accuracy is within a reasonable range, suggesting that the benefits of quantization, such as reduced memory and computational requirements, might outweigh the minor loss in accuracy for certain applications.

Finally, we report another experiment to visualize the effect of augmentation. We evaluated the two models trained with augmented data and with non-augmented data on two test traces, respectively corresponding to 30 seconds of squats and 30 seconds of push-ups. The test was repeated for two different inference frequencies, 1 Hz and 4 Hz. The results of these tests are presented in Figure 11 In the graphs, vertical lines represent the ground truth, indicating when the subject has actually performed the exercise. To correctly detect a repetition we expect at least one inference execution to classify the input frame as a squat or a push-up for each of the vertical lines. When executing the inference 4 times per second, on both input traces, both the non-augmented and the augmented versions are capable of detecting at least one squat/push-up for each repetition. However, when reducing the execution rate, the network trained without augmentation does not allow the exercise to be detected unless it is optimally centered within the frame. This aspect is clearly visible in Figure 11, where the network was tested with f_{CNN} equal to 1 Hz. In these cases, the network struggles to detect the exercise, since for some of the repetitions, highlighted with rectangles, none of the inference executions has classified the frame to be a squat or a pushup. The version trained with augmentation, on the other hand, detects every repetition correctly even when the inference is executed once per second, thus it permits to reduce in the overall workload for the microprocessor, highlighting how the augmentation has beneficial effects on the energy-related aspects of the system.

In summary, our accuracy assessments emphasize the reasonable negative impact of quantization on the model's accuracy and demonstrate how augmentation techniques have contributed to achieving more proficient models in terms of accuracy. This illustrates the feasibility of utilizing artificial intelligence-based algorithms on platforms with constrained capabilities without compromising high accuracy in the results.

As an additional validation of the approach based on CNNs, we also compared our model with a simple two-layer Artificial Neural Networks (ANN). This ANN, designed with a number of parameters comparable to that of our CNN models, would have fit into the available memory. However, after testing, the performance of the ANNs did not match that of our CNN models. Despite having a similar memory footprint, the accuracy of the ANNs for the *board* version was 93.432 %, and for *body* version was 98.025 %, lower than the CNNs accuracy (97.652% for *board* version and 99.427% for *body* version). This disparity in performance can be attributed to the inherent strengths of CNNs when dealing with lattice-type data, such as those of our time-series sensors. CNNs are better suited for identifying hierarchies or spatial patterns within this type of data.

B. POWER CONSUMPTION

We have assessed the power-related features of the system, to verify the related effectiveness of in-place processing and of dynamic management. To do so, we monitored the voltage drop across a shunt resistor connected in series to the SensorTile node power line using an ANALOG Discovery 2 digital oscilloscope. We have performed multiple measurements to determine the power consumption for each operating mode. The results are shown in Figure 13. The power consumption due to BLE transmission is highlighted, visualizing the impact of communication on the overall consumption. Moreover, it helps to understand the effectiveness of pre-processing, which enables (by means of the *threshold* task in the application model) to send in output only relevant data. Finally, we highlighted the additional power that would be consumed if the frequency is not optimized dynamically by ADAM, using the same system frequency required to support all operating modes (in this case 4 MHz). As may be noticed, the most efficient modes are those that implement some on-board pre-processing of the acquired data. Especially the CNN mode is convenient. It provides detailed analysis capabilities, used for exercise classification, but exploiting in-place processing reduces the communication requirements, saving around 65% power consumption in the case of the *body* sensor and around 50% in the case of the *board* sensors, with respect to the raw data transmission that would be required to perform the same analysis on the cloud. In the following, we comment on each operating mode in more detail.

1) OPERATING MODE *raw*

As outlined in Section IV, this operating mode allows the transmission of all data taken from the sensor via Bluetooth Low Energy (BLE). In this case, the contribution related to transmission does not depend on pre-processing results and is continuously dissipated by the system. For supporting such a continuous communication, the adequate clock frequency (f_{sys}) is set to 4 MHz. To maximize data transmission via Bluetooth, two sensor acquisitions are combined into one BLE packet, reducing the sending frequency from 100 Hz (equal to the sampling rate) to 50 Hz. Nevertheless, the *raw* mode is the most power-consuming operating mode, consuming

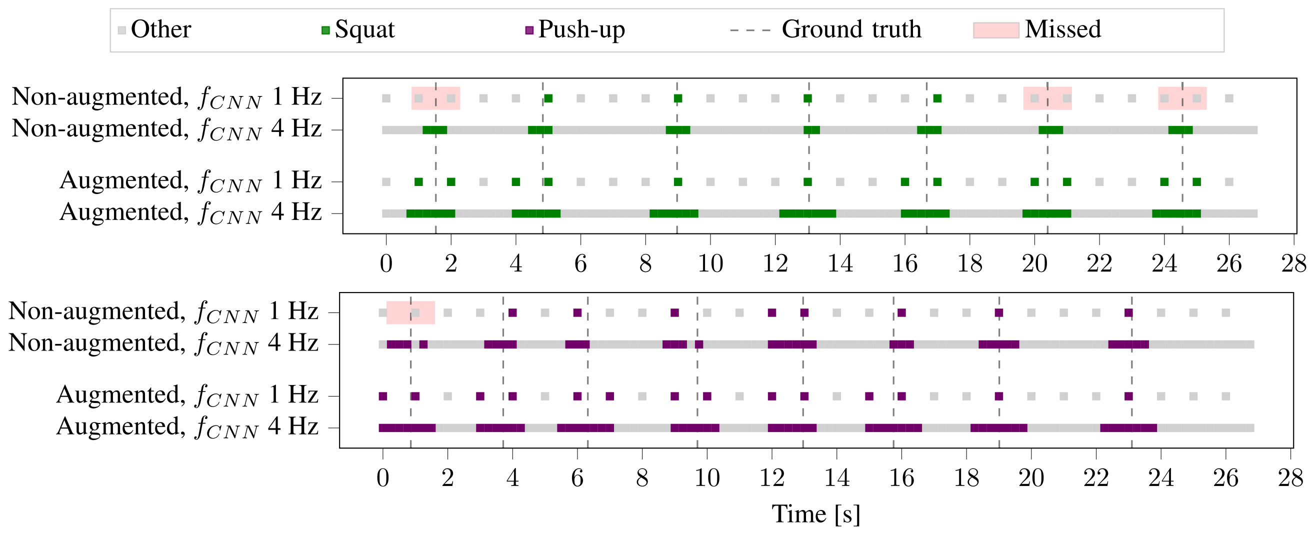


FIGURE 11. Tests on two different traces for squats and push-ups execution. Each was tested with a network trained with augmented and non-augmented data and tested with f_{CNN} equal to 4 or 1.

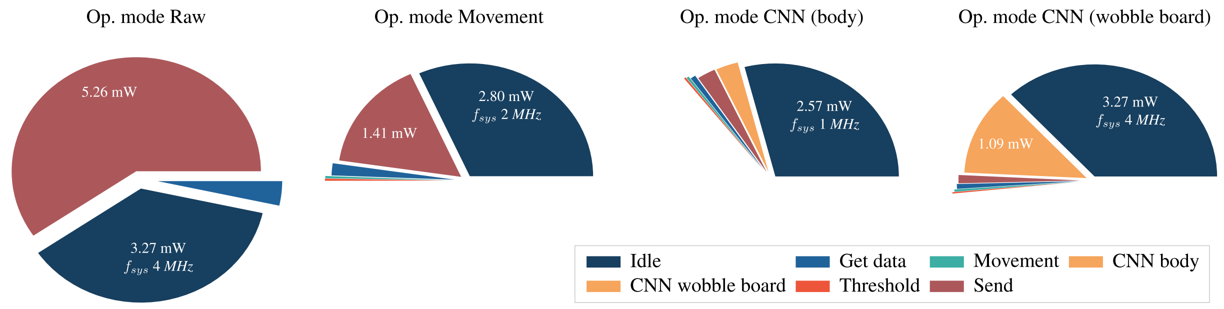


FIGURE 12. Contribution of each task in terms of power consumption for each operating mode.

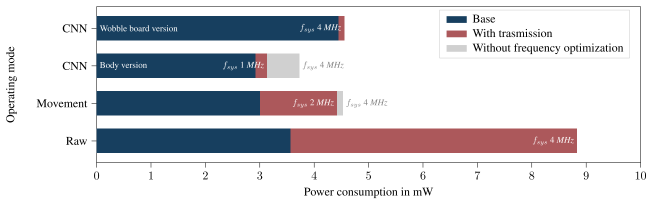


FIGURE 13. Power consumption for each operating mode.

8.83 mW, with a contribution of 3.5 mW dissipated for transmission.

2) OPERATING MODE *detection*

As described in Section IV, when the system is set in this operating mode, the node communicates dome pre-processed inclination and direction data to the mobile app. The processing task *Movement* is executed 10 times per second. If no threshold is applied, the task send is also executed with a frequency of 10 Hz, requiring a clock frequency f_{sys} set to 2 MHz. In this case, the node dissipates 4.4 mW, which can decrease to around 3 mW if no relevant inclination or direction changes are detected. The savings obtained by means of dynamic clock frequency optimization, in this case, derive from a reduction from 4 MHz to 2 MHz, corresponding to only around 3% improvement.

3) OPERATING MODE *classification*

Whenever neural network inference is executed, information about the results of exercise classification is transmitted. Again, as mentioned in Section in Section IV, there are two types of CNN-based processing, board movement classification, and body movement classification. As shown in Table 4, they have different sizes, the *board* version requiring more computational capabilities. For this reason, the two cases are considered distinct to show the power consumption results. Board movement classification requires f_{sys} to be set at a minimum of 4 *MHz*, while for body movement classification, the optimal clock frequency is 1 *MHz*. For both cases, f_{CNN} is equal to 1 *Hz*. This is the most power-efficient operating mode (3.13 mW for the *body* sensor and 4.55 mW for the *board*), demonstrating how enabling edge processing can lead to improved power consumption. As may be

TABLE 8. Energy values for each task in the process network and the power consumption of the platform.

Task type	Number of cycles	Energy contribution
Get data	863	$E_g = 3.05\,\mu J$
Movement	831	$E_m = 2.91\,\mu J$
CNN, wobble board version	2 429 623	$E_{cw} = 1\,086.18\,\mu J$
CNN, body version	623 605	$E_{cb} = 259.26\,\mu J$
Threshold	910	$E_t = 2.73\,\mu J$
Send data (f_{sys} *= 1 MHz*)	–	$E_s = 213.29\,\mu J$
Send data (f_{sys} *= 2 MHz*)	–	$E_s = 170.73\,\mu J$
Send data (f_{sys} *= 4 MHz*)	–	$E_s = 105.18\,\mu J$

Device	Power consumption		
	1 ***MHz***	**2** ***MHz***	**4** ***MHz***
Platform in idle state	$2.568\,mW$	$2.802\,mW$	$3.267\,mW$

noticed, in-place processing has significantly decreased the communication-related needs, the transmission of data only contributes 0.21 mW (*body* version, around 7% of the total) and 0.11 mW (*board* version, around 2,5% of the total). Dynamic frequency optimization is used only in the *body* version and permits saving around 0,6 mW (16% of the total) by reducing the clock frequency to 1 MHz from the 4MHz static value.

4) POWER CONSUMPTION BREAKDOWN

In order to better understand the different contributions to the power figures and to be capable of understanding the effects of the proposed approach on different use cases, we have built a model showing how each task contributes to power consumption. To derive it, we have conducted a wide range of measurements of the energy consumed by the node under different configuration circumstances. In Table 8, the energy values for each task in the process network and the power consumption of the platform as a function of the chosen system frequency f_{sys} are shown. Below are the equations that estimate the power consumption for all operating modes:

$$P_{raw} = (E_g + \alpha E_s) \cdot f_s + P_{idle}, \quad (1)$$

$$P_{detection} = E_g \cdot f_s + (E_m + E_t + E_s) \cdot f_m + P_{idle}, \quad (2)$$

$$P_{classification} = E_g \cdot f_s + E_m \cdot f_m + (E_{cx} + E_t + E_s) \cdot f_{CNN} + P_{idle}\,. \quad (3)$$

In Equations 1, 2 and 3, the following operators are used: f_s is the sampling frequency; f_{CNN} is the frequency of convolutional neural network execution; f_m is the movement task frequency execution; α^{-1} is the number of samples inserted in a BLE package; P_{idle} power consumption of the platform in idle state, depends on the system frequency.

An example of the application of such power estimation equations is shown in Figure 12, from it, the contribution of each task can be observed. As can be noticed from the graph, the operating mode *raw* has the highest power consumption among the different modes. The power consumption of the *Send* task alone in this mode is greater than the combined power consumption of all other tasks in the *detection* and *classification* operating modes. This is due to the high data transmission rate associated with the operating mode *raw*. In some operating modes, it is possible to reduce the system frequency by as much as 4 times compared to the operating mode *raw* without violating real-time constraints. Another noteworthy aspect is the low power consumption resulting from the *CNN* task, which is able to dramatically reduce the usage of the *send* task. This is achieved through efficient processing that simplifies the information, subsequently reducing the frequency of data transmission to external systems.

The implementation of edge computing, as demonstrated by the efficient processing in the *detection* and *classification* operating modes, leads to significant energy savings compared to scenarios where processing is not enabled on the node, as in the case of the operating mode *raw*. This approach demonstrates the potential benefits of leveraging edge computing in real-time applications.

VI. LIMITATIONS AND FUTURE PERSPECTIVES

In forthcoming developments, our intention is to improve our system by diversifying the dataset to encompass a range of activities, environments, and participants, thereby enhancing its adaptability to new circumstances. This will lead to an improvement in the generality of the dataset and in the robustness of the neural network without necessarily having to give up data augmentation techniques. Additionally, we plan to incorporate handheld sensors to expand our exercise tracking capabilities, and to provide more comprehensive, personalized feedback for each user. This approach aims to facilitate the development of more efficient and individually customized training programs. Building on these future developments and improvements, the envisioned enhancement of our system will not only improve individual user experiences, but also pave the way for broader applications.

To move into a wider area of our research, our approach is designed to seamlessly fit into an Internet of Things (IoT) environment, thus facilitating the evolution of mHealth applications. The authors in [29] and [30] reveal some of the most important critical issues in this area, our work could solve some of these criticisms. Specifically, key concerns addressed include energy efficiency in wearable sensors and data accuracy, both of which our system mitigates through innovative data processing techniques and employing state-of-the-art AI methodologies, respectively. To tackle the critical issue of data privacy and security, we've constructed our model to minimize data transmission, thereby reducing potential data breach risks. Our system further accommodates the necessity for decision support in mHealth by offering real-time data analysis to assist users with their exercise routines. Finally, we cater to the demand for ambient assisted living solutions, with our system thoughtfully designed to provide sensorimotor training support for users within their homes, promoting health and independence.

Compatibility with IoT systems is further strengthened by our previous work [31]. In that study, we successfully implemented an interconnected network of ECG sensors in an IoT environment using a similar approach.

VII. CONCLUSION

We have developed a system relying on two microcontroller-based sensor nodes, capable of tracking exercises and activity performed during sensorimotor training. The sensors, respectively integrated into a custom wobble-board and worn by the user, can monitor the training to finely track the results, enable interaction with a trainer, and foster user engagement.

Each node is capable of running on-edge CNN processing and is able to detect basic movements performed on a wobble board. Depending on the operating mode and the required workload, the device can reorganize itself automatically. They dynamically switch between operating modes, corresponding to different levels of pre-processing applied to the sensor data, to adapt to different usage requirements.

In-place execution of a quantized neural network on the nodes achieves over 97% accuracy in detecting wobble board movements and over 99% accuracy in identifying body exercises on a custom dataset. This demonstrates the possibility of using lightweight AI algorithms to implement activity recognition in this scope.

Moreover, we were able to save up to 65% of energy by enabling in-place analysis and efficiently managing hardware and software reconfiguration of the device, confirming the potential of using sensor data in-place processing to increase battery life.

REFERENCES

[1] R. Saini, P. Kumar, P. P. Roy, and D. P. Dogra, "A novel framework of continuous human-activity recognition using Kinect," *Neurocomputing*, vol. 311, pp. 99–111, Oct. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231218306179

[2] J. K. Aggarwal and L. Xia, "Human activity recognition from 3D data: A review," *Pattern Recognit. Lett.*, vol. 48, pp. 70–80, Oct. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865514001299

[3] S. Herath, M. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," 2016, *arXiv:1605.04988*.

[4] R. Poppe, "A survey on vision-based human action recognition," *Image Vis. Comput.*, vol. 28, no. 6, pp. 976–990, Jun. 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885609002704

[5] F. Demrozi, G. Pravadelli, A. Bihorac, and P. Rashidi, "Human activity recognition using inertial, physiological and environmental sensors: A comprehensive survey," *IEEE Access*, vol. 8, pp. 210816–210836, 2020.

[6] S. Herath, M. Harandi, and F. Porikli, "Going deeper into action recognition: A survey," *Image Vis. Comput.*, vol. 60, pp. 4–21, Apr. 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885617300343

[7] D. Strömbäck, S. Huang, and V. Radu, "MM-Fit: Multimodal deep learning for automatic exercise logging across sensing devices," *Proc. ACM Interact., Mobile, Wearable Ubiquitous Technol.*, vol. 4, no. 4, pp. 1–22, Dec. 2020, doi: 10.1145/3432701.

[8] S. Ha, J.-M. Yun, and S. Choi, "Multi-modal convolutional neural networks for activity recognition," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2015, pp. 3017–3022, doi: 10.1109/SMC.2015.525.

[9] G. Dogan, S. S. Ertas, and I. Cay, "Human activity recognition using convolutional neural networks," in *Proc. IEEE Conf. Comput. Intell. Bioinf. Comput. Biol. (CIBCB)*, Oct. 2021, pp. 1–5.

[10] Y. L. Coelho, F. de Assis Souza dos Santos, A. Frizera-Neto, and T. F. Bastos-Filho, "A lightweight framework for human activity recognition on wearable devices," *IEEE Sensors J.*, vol. 21, no. 21, pp. 24471–24481, Nov. 2021.

[11] T.-H. Dao, H.-Y. Hoang, V.-N. Hoang, D.-T. Tran, and D.-N. Tran, "Human activity recognition system for moderate performance microcontroller using accelerometer data and random forest algorithm," *EAI Endorsed Trans. Ind. Netw. Intell. Syst.*, vol. 9, no. 4, p. e4, Nov. 2022. [Online]. Available: https://publications.eai.eu/index.php/inis/article/view/2571

[12] L. Bao and S. S. Intille, "Activity recognition from user-annotated acceleration data," in *Pervasive Computing*, A. Ferscha and F. Mattern, Eds. Berlin, Germany: Springe, 2004, pp. 1–17.

[13] A. Ghibellini, L. Bononi, and M. Di Felice, "Intelligence at the IoT edge: Activity recognition with low-power microcontrollers and convolutional neural networks," in *Proc. IEEE 19th Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2022, pp. 707–710.

[14] Z. Wenzheng, "Human activity recognition based on acceleration sensor and neural network," in *Proc. 8th Int. Conf. Orange Technol. (ICOT)*, Dec. 2020, pp. 1–5.

[15] K. Warunsin and T. Phairoh, "Wristband fall detection system using deep learning," in *Proc. 7th Int. Conf. Comput. Commun. Syst. (ICCCS)*, Apr. 2022, pp. 223–227.

[16] I. I. Belousov and A. A. Smirnov, "HAR CNN using accelerometer data set with debugging board based on STM32F407 microcontroller," in *Proc. 8th Int. Young Researcher' Conf.-Phys., Technol., Innov.*, 2022, Art. no. 090004, doi: 10.1063/5.0088699.

[17] F. Daghero, A. Burrello, C. Xie, M. Castellano, L. Gandolfi, A. Calimera, E. Macii, M. Poncino, and D. J. Pagliari, "Human activity recognition on microcontrollers with quantized and adaptive deep neural networks," *ACM Trans. Embedded Comput. Syst.*, vol. 21, no. 4, pp. 1–28, Aug. 2022, doi: 10.1145/3542819.

[18] F. Daghero, D. J. Pagliari, and M. Poncino, "Two-stage human activity recognition on microcontrollers with decision trees and CNNs," in *Proc. 17th Conf. Ph.D Res. Microelectron. Electron. (PRIME)*, Jun. 2022, pp. 173–176.

[19] N. Maclean and P. Pound, "A critical review of the concept of patient motivation in the literature on physical rehabilitation," *Social Sci. Med.*, vol. 50, no. 4, pp. 495–506, 2000.

[20] S. E. Asp, K. O. Halldorsdottir, C. Hägg, M. L. Møller, B. P. Mickelsson, L. Boldt, and D. Skaarup, "WobbleActive," in *Proc. 1st Int. Symp. Ludic Engagement Design*, 2007.

[21] N. C. Nilsson, S. Serafin, and R. Nordahl, "Gameplay as a source of intrinsic motivation for individuals in need of ankle training or rehabilitation," *Presence*, vol. 21, no. 1, pp. 69–84, Feb. 2012.

[22] B. Blazica and P. Krivec, "OLOK boardy—Gamified sensorimotor training with affordable smart balance board," in *Proc. 3rd Annu. Sci. Prof. Int. Conf. 'Health Children Adolescent'*, Sep. 2019, p. 185.

[23] M. A. Scrugli, B. Blažica, and P. Meloni, "An adaptable cognitive microcontroller node for fitness activity recognition," in *Proc. Int. Workshop Design Archit. Signal Image Process.*, K. Desnos and S. Pertuz, Eds. Cham, Switzerland: Springer, 2022, pp. 149–161.

[24] L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient neural network kernels for arm cortex-M CPUs," 2018, *arXiv:1801.06601*.

[25] Unity. *Lego Microgame*. Accessed: Apr. 18, 2023. [Online]. Available: https://learn.unity.com/project/lego-template

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (Adaptive Computation and Machine Learning series). Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: https://books.google.it/books?id=Np9SDQAAQBAJ

[27] Pytorch. *Static Quantization With Eager Mode in PyTorch*. Accessed: Oct. 18, 2022. [Online]. Available: https://pytorch.org/tutorials/advanced/static_quantization_tutorial.html

[28] M. Scrugli. *Wobble Board Dataset*. Accessed: Apr. 18, 2023. [Online]. Available: https://github.com/matteoscrugli/wobbleboard-dataset

[29] O. S. Albahri, A. S. Albahri, K. I. Mohammed, A. A. Zaidan, B. B. Zaidan, M. Hashim, and O. H. Salman, "Systematic review of real-time remote health monitoring system in triage and priority-based sensor technology: Taxonomy, open challenges, motivation and recommendations," *J. Med. Syst.*, vol. 42, no. 5, p. 80, 2018, doi: 10.1007/s10916-018-0943-4.

[30] Y. A. Qadri, A. Nauman, Y. B. Zikria, A. V. Vasilakos, and S. W. Kim, "The future of healthcare Internet of Things: A survey of emerging technologies," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 2, pp. 1121–1167, 2nd Quart., 2020.

[31] M. A. Scrugli, D. Loi, L. Raffo, and P. Meloni, "An adaptive cognitive sensor node for ECG monitoring in the Internet of Medical Things," *IEEE Access*, vol. 10, pp. 1688–1705, 2022.

MATTEO ANTONIO SCRUGLI received the M.S. degree (Hons.) and the Ph.D. degree in electrical engineering from the University of Cagliari, in 2018 and 2022, respectively. He is currently a Research Fellow with the DIEE, University of Cagliari. His research mainly concerns the development of systems capable of managing at runtime the hardware and software configuration of low-power devices in order to adapt it to the required operating mode. His current research interest includes the cognitive IoT devices, based on single-core or multi-core platforms.

BOJAN BLAŽICA received the Ph.D. degree. He is a researcher in the fields of human–computer interaction, user experience design, usability testing, and artificial intelligence. Most recently, he has been involved in projects related to active aging, where individualized targeted training (SI4Care) and gardening (turntable) have been used to improve the quality of life of the elderly, and projects related to crowd-sourcing food composition data with a mobile application, and developing usable interfaces for decision support systems in agriculture (DEXiWare). He is the Co-Founder of BMP3 (focused on producing biomechanical measurement devices) and Proventus (software company developing web and mobile platform for gardening Tomappo).

LUIGI RAFFO (Member, IEEE) received the Laurea degree in electronic engineering and the Ph.D. degree in electronics and computer science from the University of Genoa, Italy, in 1989 and 1994, respectively. He is currently a Full Professor in electronics with the Department of Electrical and Electronic Engineering, University of Cagliari, Italy, where he joined the Department of Electrical and Electronic Engineering, as an Assistant Professor, in 1994, an Associate Professor, in 1998, and a Full Professor in electronics, since 2006. Since 2012, he has been a Rector's delegate for International Research Projects. His research interests include the study, design, and development of platforms, and systems and integrated circuits for several applications mainly in the field of biomedical engineering, with focus on high-performances, high-efficiency, and adaptability. He is the author of more than 200 scientific papers and five international patents.

PAOLO MELONI is currently an Associate Professor with the University of Cagliari. His research interests include the development of advanced digital systems, on the application-driven design and programming of multi-core on-chip architectures and FPGAs.

Open Access funding provided by 'Università degli Studi di Cagliari' within the CRUI CARE Agreement