IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 5, 2021, accepted December 9, 2021, date of publication December 20, 2021, date of current version January 6, 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3136793

An Adaptive Cognitive Sensor Node for ECG Monitoring in the Internet of Medical Things

MATTEO A. SCRUGLI, DANIELA LOI, LUIGI RAFFO, AND PAOLO MELONI, (Member, IEEE)
Department of Electrical and Electronic Engineering (DIEE), University of Cagliari, 09123 Cagliari, Italy

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

This work was supported in part by the European Union (EU) Commission for funding ALOHA Project H2020 under Grant 780788, in part by the Joint Research and Development under Project F/050395/01-02/X32, in part by the INtelligent Systems for Integrated hEalth ManagEment (INSIEME) through the Italian Ministero dello Sviluppo Economico (MISE), D.M. 01/06/2016, Axis 1, action 1.1.3. of the National Operative Program "Imprese e Competitività" 2014–2020 FESR, Horizon 2020—PON I&C 2014–2020 under Grant CUP:B28I17000060008.

ABSTRACT The Internet of Medical Things (IoMT) paradigm is becoming mainstream in multiple clinical trials and healthcare procedures. Cardiovascular diseases monitoring, usually involving electrocardiogram (ECG) traces analysis, is one of the most promising and high-impact applications. Nevertheless, to fully exploit the potential of IoMT in this domain, some steps forward are needed. First, the edge-computing paradigm must be added to the picture. A certain level of near-sensor processing has to be enabled, to improve the scalability, portability, reliability and responsiveness of the IoMT nodes. Second, novel, increasingly accurate data analysis algorithms, such as those based on artificial intelligence and Deep Learning, must be exploited. To reach these objectives, designers, and programmers of IoMT nodes, have to face challenging optimization tasks, in order to execute fairly complex computing tasks on low-power wearable and portable processing systems, with tight power and battery lifetime budgets. In this work, we explore the implementation of a cognitive data analysis algorithm, based on a convolutional neural network trained to classify ECG waveforms, on a resource-constrained microcontroller-based computing platform. To minimize power consumption, we add an adaptivity layer that dynamically manages the hardware and software configuration of the device to adapt it at runtime to the required operating mode. Our experimental results show that adapting the node setup to the workload at runtime can save up to 50% power consumption. Our optimized and quantized neural network reaches an accuracy value higher than 97% for arrhythmia disorders detection on MIT-BIH Arrhythmia dataset.

INDEX TERMS Adaptive system, health information management, Internet of Things, low power electronics, neural network, remote sensing, runtime, wearable sensors.

I. INTRODUCTION

The Internet of Things (IoT) paradigm, declined in the so-called Internet of Medical Things (IoMT), enables seamless collection of a wide range of data streams, that can be analyzed to extract relevant information about the patient's condition. However, in order to make IoMT really ubiquitous and effective, a step forward is needed to improve scalability, responsiveness, security, privacy. Most of the efforts aiming in this direction focus on the adoption of an edge-computing approach. Data streams, acquired by sensors, can be processed, at least partially, at the edge, before being sent to the cloud, on adequate portable/wearable processing platform. This provides several advantages. First, it reduces bandwidth requirements. Near-sensor processing can extract from raw data more compact information. In this way, less communication bandwidth is required to the centralized server, and, at the same time, the energy consumption related to wireless data transmission is drastically reduced. Second, near-sensor processing can improve reliability. Monitoring must not rely necessarily on connection availability and, if immediate feedback to the user and/or local actuation is needed, the delays through the network can be avoided. Moreover, pre-processed information can be delivered to the cloud, preserving user privacy avoiding the propagation of sensitive raw data.

An extremely important field of application of IoMT is related to the treatment of cardiovascular diseases (CVD), a major public health problem that generates millions of deaths yearly and impacts significantly on health-related public costs. As an example, in 2016, ≈17.6 million (95% CI,

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

17.3–18.1 million) deaths were attributed to CVD globally, representing an increase of 14.5% (95% CI, 12.1%–17.1%) since 2006 [1]. In Europe, the CVD impact on the economy is estimated to be around €210 billion [2]. It is commonly accepted that machine learning and IoT are important for creating a novel assisted living methodology [3], this is also true for CVD monitoring. In [3], many aspects regarding assisted living and how to improve the quality of this category of devices are discussed, some of them are: the introduction of machine learning can allow the device to adapt autonomously to the environment and reduces manual interventions by an operator; the combination of artificial intelligence (AI) and IoT leads to improvements from the point of view of comfort and energy saving, allows constant monitoring of the environment and learning from its behavior. CVD treatment with remote monitoring involves in most cases analysis of electrocardiogram (ECG) signals. Creating embedded platforms implementing such kind of analysis is promising, but, at the same time, very challenging, for several reasons:

- **Requires edge computing at low energy/cost budget:** Sensor nodes must be wearable and affordable to implement ubiquitous patient monitoring. Given the high data rate produced by ECG sensors, raw data wireless data transmission requires an energy budget that cannot be negligible when the task is implemented in a portable and inexpensive computing device.
- **Requires cognitive computing:** state-of-the-art anomalies detection tasks are based on the analysis of manually designed features with are hard to craft and extract online from the ECG waveforms. Thus the community is shifting focus to techniques based on neural networks and deep learning, that rely on automatically learned features. However, existing approaches that use deep learning for the recognition of anomalies on the ECG trace, rarely pay attention to energy consumption to be deployed on low-power processing systems. Thus, pretty often do not take into account workload reduction and post-deployment accuracy evaluation.
- **Requires adaptivity:** Intensity of the processing workload is very dependent on the needed level of detail and also intrinsically data-dependent. Information to be analyzed is usually contained in waveform shapes of ECG peaks, thus the rate of sample frames to be analyzed is directly dependent on the patient's heartbeat rate. This paves the way to energy consumption reduction by means of an adaptive management of the system, that reconfigures itself on the basis of the detected data and on the chosen operating mode (OM).

In this work, we explore the implementation of a system for at-the-edge cognitive processing of ECG data. We have conceived a hardware/software setup for the processing system inside the IoMT node. We have used SensorTile, a compact processing device developed by STMicroelectronics, as a reference microcontroller platform. The system makes use of a quantized convolutional neural network, specifically sized and trained to run on a low-power microcontroller, that has been validated in post-deployment and recovers accuracy drops that arise in real online utilization. Moreover, we take a step further in hardware/software optimization using adaptivity, allowing the system to reconfigure itself, to suit different operating modes and data processing rates. To this aim, besides executing the tasks that implement sensor monitoring and on-board processing, the system includes a component called ADAM (ADAptive runtime Manager), able to dynamically manage the hardware/software configuration of the device optimizing power consumption and performance. ADAM creates and manages a network of processes that communicate with each other via FIFOs. The morphology of the process network varies to match the needs of the operating mode in execution. ADAM can be triggered by re-configuration messages sent by the external environment or by specific workload-related variables in the sampled streams (e.g. patient's heartbeat pace). When triggered, ADAM changes the morphology of the process network, switching on or off processes, and reconfigures the inter-process FIFOs. Moreover, depending on the new configuration it changes the hardware setup of the processing platform, adapting power-relevant settings such as clock frequency, supply voltage, peripheral gating.

The remainder of this paper is as follows: Section II describes the landscape of related work in literature, Section III describes an overview of the overall SoS picture, Section IV presents the proposed template for the node and the reference target platform and the reference application model. Moreover it presents the details of the ADAM component. Section V describes how the chosen template has been declined to implement ECG monitoring, the proposed operating modes and the processing tasks coexisting in the application. Section VI discusses our experimental results. Section VII shows a comparison of our experimental results with work similar to our own, in addition to highlighting the main limitations of our system. Finally, Section VIII outlines our conclusions.

II. RELATED WORK

In this section we will mention some of the works proposed in literature that offer IoT solutions for health care. Many of these solutions involve collecting and analyzing data only on the cloud. The cognitive edge-computing paradigm on the other hand introduces many advantages in terms of responsiveness, accuracy and data security. We will focus on works dealing with ECG monitoring and anomalies detection, showing how deep learning-based analysis achieves detection accuracy values equal to or better than more traditional methods.

Multiple solutions involving the use of sensor networks in hospitals or at home and the IoMT are proposed in literature [4]–[6]. Most of these studies exploit a cloud-based analysis: data is usually encapsulated in standard formats and sent to remote servers for data mining. Most research work takes into account wearability and portability as main objectives when developing IoMT-based data sensing archi-

tectures, thus devices available on the market can guarantee autonomy for days or weeks [7], [8]. Commercial devices such as Medtronic SEEQ Mobile Cardiac Telemetry (MCT) System and ViSi Mobile System have been designed to fit non-invasively into the skin and body, both devices allow vital sign monitoring, in particular, the first one, focuses on cardiac activity. These devices provide for raw data streaming to the cloud and arrhythmias recognition, however, power consumptions in different operating modes are not specified, and no data relating to the detection accuracy are available.

Although artificial intelligence (AI) is not a new concept, its application in smart homes and smart environments still has critical issues. On the other hand, the combination of AI and IoT opens up new possibilities, allowing for new types of intelligent pervasive systems and platforms, providing the highest level of comfort, energy savings, and new personalized services for residents of intelligent environments. To really use cognitive computing at the edge, more complex and accurate algorithms, such as those exploiting artificial intelligence or deep learning, must be targeted. Their efficiency has been widely demonstrated on high-performance computing platforms. Some examples are [9], where an NVIDIA GeForce GTX 1080 Ti (11 GB) is used, [10], that uses a 3.5 *GHz* Intel Core i7-7800X CPU, RAM 32 GB, and a GPU NVIDIA Titan X (Pascal, 12 GB), or [11], based on an i7-4790 CPU at 3.60 *GHz*. However, how to map state-of-the-art cognitive computing on resource-constrained platforms is still an open question. There is an ever-increasing number of approaches focusing on machine learning and artificial intelligence to identify specific events in sensed data. In [12] and [13] authors exploit ANN (artificial neural networks) to detect specific conditions from the proposed data. In [13], an ANN is used to identify the emotional states (happiness or sadness) of the patient. However, network topologies are still very basic and highly tuned, and customized to fit on the target device.

As already described in Section I, cardiovascular diseases are one of the most frequent causes of death worldwide, which is why there is also a strong interest among the scientific community to address this issue. There are several works that implement ECG monitoring on customized chips, it is shown that with low energy consumption it is possible to classify cardiac anomalies in real-time even using AI methods. In [14] wavelet theory was adopted to perform features extraction and classification, an accuracy of 97.25% was achieved on arrhythmias recognition. In [15] an excellent job of researching the compromise between complexity and performance on different classifiers with different lead configurations was made. In particular, they obtained good results with a linear discriminant analysis (LDA) using the spectral energy of the PQRST complexes as features. In [16] a Naive Bayes classifier is exploited, they obtained an accuracy equal to 86% on arrhythmias recognition using the PQRST points detected with a Pan-Tompkins algorithm as features. In [17] a wavelet-based algorithm is used to extrapolate morphological and temporal features, a Principal Component Analysis (PCA) is used to reduce the dimensionality and redundancy of the features. An accuracy of 97.4% was obtained with an evolutionary ANN (Artificial Neural Network). In [18] and [19] possible implementations of arrhythmia classification algorithms based on spiking neural networks are shown.

Other works focus on implementing efficient off-the-shelf commercial devices to facilitate easier community adoption of these techniques. Several target technologies have been used in the literature, such as FPGAs or microcontroller-based boards. A substantial number of research works are dedicated to studying IoT devices in the medical field, in particular ECG monitoring and anomalies detection. In [20] and [21], authors deal with simple ECG monitoring on wearable devices. In [22] and [23], authors treat with particular attention the aspect of Signal Quality Assessment (SQA), identifying the signal quality level is useful for knowing when to ignore the input data or even when to put the device into a sleep state. In [24] a system that uses Compressive Sensing (CS) to compress bio-signals in a power-efficient way is proposed. In [25], authors propose a monitoring device with a particular focus on low energy consumption.

In some works proposed in the literature dealing with ECG signal monitoring, local processing is used only for implementing easy checks on raw data and/or marshaling tasks for wrapping the sensed data inside standard communication protocols [26]–[29]. On the other hand, there are numerous works in the literature that exploit cognitive computing for CVD detection, even at the edge. Some of this work and its experimental results will be compared with our own. The cognitive approach that involves the use of convolutional neural networks (CNNs) shows promise in terms of accuracy in detecting ECG signal arrhythmias compared to other traditional strategies based or not on artificial intelligence algorithms [30]–[32]. Moreover, in most cases, the use of CNNs allows to classify an ECG signal even if not pre-processed. The most common strategies present in many state-of-the-art works that allow to improve the efficiency of these IoT nodes are: moving the inference operations at the edge, choosing a low-power device, quantization techniques to speed up the network execution of the inference stage.

In [33] the importance of real-time monitoring is discussed, they propose a system capable of recognizing anomalies on ECG, testing their methodology with various machine learning techniques.

In [31], in addition to the comparison with other techniques used to analyze the ECG trace, Latent Semantic Analysis techniques were used to improve the accuracy of the network. Both training and inference take place on the cloud side, our aim is to move the inference to the edge of a low-power device in order to reduce latency times and reduce energy consumption due to wireless communication.

In [34] excellent results were obtained for ventricular arrhythmias and supraventricular arrhythmias classification: 99.6% and 99.3% for accuracy value, 98.4% and 90.1% for sensitivity value, 99.2% and 94.7% for positive predictive

value, respectively. In [34] a double CNN is used, one of them takes as input the frequency domain information of the ECG signal (a fast Fourier transform is performed). This methodology, despite the excellent results in terms of accuracy, was not taken into consideration in our case because it's particularly expensive to perform on a microcontroller.

In [35], the authors obtain excellent results in terms of accuracy, they used a method similar to the one used in [36] to combine the output features of three independent neural networks (VGG19, AlexNet, and Inception-v3) and then classify them as a single output. The methodology used allows a significant increase in terms of accuracy (97.6%) compared to the use of neural networks chosen individually, however, this method is too resource-expensive in terms of memory and computational load to be executed in a microcontroller such as the one we selected. In [32], again, there is proof of how neural networks obtain good results if compared with methods such as K-nearest neighbors (KNN) and random forest (RF) (95.98% on MIT-BIH Supraventricular Arrhythmia Database) and the inference occurs directly from the IoT node but the power consumption remain relatively high once again, they are used in fact non-low power devices such as Raspberry Pi 4 or Jetson Nano. Always in [32], a good job of research has been done on the morphology of the CNN network that was more suitable for inference on ECG signals, the network we used provides a structure very similar to the one chosen in [32].

In [37], good results are obtained in terms of accuracy (96% using MIT Arrhythmia dataset), the inference does not occur on the Cloud side, nor on the sensorial device at the edge, but from a device located within the same WBAN network. The sensory device, therefore, remains in constant communication with the outside to send the raw data of the signal. An approach similar to [38] was chosen, an embedded device was chosen that is able to perform the inference directly on the node. In [38], a study was made on the variation of accuracy as a function of different quantization levels, they choose a precision of 12-bit with an accuracy of 97%, but it's visible that already from 6-bit upwards the accuracy levels exceed the 90%. Power consumption is around 200 mW during computation and the node is based on FPGA technology.

In [39] ResNet, AlexNet, and SqueezeNet neural networks are used, an accuracy of 97% is achieved. In addition to the use of much more expensive computational neural networks, there is an overhead due to encoding the raw ECG signal into a JPEG image. The platforms chosen are Arduino UNO for signal acquisition and Raspberry Pi 3B+ to process the raw signal, making the power consumption considerably higher than a solution based only on microcontroller.

Other works with which we are confronted are [40]–[43].

Table 1 lists and describes the main works we were confronted with. On one hand, the community has designed novel ultra-low power processing platforms, providing previously unmatched computation capabilities on typical AI and data analysis workloads. In this work, we extend [44] taking into account that in the current state-of-the-art landscape, network topologies, processing platforms and software tools can be much more complex. In summary, as a main novel contribution, we propose:

- The definition of a hardware/software/firmware architectural template for the implementation of a remotely-controlled sensory node, allowing for near-sensor cognitive data processing for cardiac anomalies detection.
- Its validation on a state-of-the-art data analysis based on a Convolutional Neural Network.
- The evaluation of the effectiveness of in-place computing and operating mode dynamic optimization on ARM microcontroller platform, as a method to reduce the power consumption of the node.

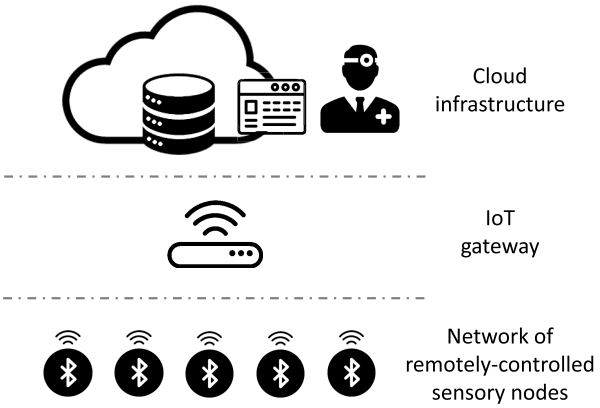


FIGURE 1. General overview of the proposed system.

III. ADAPTIVE SENSOR NODE ARCHITECTURE

Figure 1 shows an overview of the system architecture as envisioned in this work. We see the network as composed of three levels. The lower level is composed of the sensor nodes, which acquire information from the environment. They are connected to the upper level using Bluetooth technology. The nodes are capable of reacting, reconfiguring their operating mode, to commands sent from higher levels, or to workload changes that can be detected near-sensor, thanks to the internal component called ADAptive runtime Manager which will be described later. The intermediate level consists of several gateways, in charge of collecting the data from the sensor nodes and send them to the upper level. To test the approach presented in this work, the gateway was implemented with a Raspberry Pi 3 running a Linux operating system.

For the same purpose, the cloud-based infrastructure, on top of the stack, has been implemented using Google App Engine. Data is stored securely on the cloud, and can be used for analysis or simply visualized by a healthcare professional. Such kind of user, accessing a web-based interface, can also send downstream commands to the nodes, to communicate a required change of the operating mode, e.g. changing the needed detail of acquisition of the patient's parameters.

TABLE 1. Structure of some devices proposed in literature.

Reference	*Node technology*	*Processing placement*	*Power consumption*	*Accuracy and Dataset*	*Classification method*
[34]	–	–	–	MIT-BIH ventricular arrhythmias and supraventricular arrhythmias classification: 99.6% and 99.3% for accuracy value, 98.4% and 90.1% for sensitivity value, 99.2% and 94.7% for precision value, respectively	CNN
[31]	–	training and inference on cloud	–	94% for sensitivity vale, 99,3% for accuracy value on custom dataset	CNN + LSA method
[40]	–	–	–	accuracies of 93.63% for ventricular ectopic beats and 95.57% for supraventricular ectopic beats on MIT Arrhythmia dataset	RBM and DBN
[41]	–	–	–	MIT-BIH dataset, NSVFQ classes: 99,09%, 98,55% and 99,52% for accuracy, sensitivity and specificity value, respectively	DNN
[35]	–	–	–	97.6% accuracy on MIT-BIH dataset for ventricular tachyarrhythmia desease	CNN
[42]	Intel I7-4700MQ at 2.4 GHz (eight CPUs) and 16-Gb memory, but designed to run on cheaper and less powerful architectures	inference on edge	–	accuracy up to 99% for ventricular ectopic beats and up to 97.6% for supraventricular ectopic beats on MIT Arrhythmia dataset	adaptive implementation of 1-D CNNs
[37]	hierarchical structure	training on cloud, inference on edge	–	96% for accuracy value on MIT Arrhythmia dataset	CNN
[38]	FPGA	inference on edge	$200\,mW$ [I]	97% accuracy value for NLRAV classes on MIT Arrhythmia dataset	quantized CNN
[32]	Jetson Nano (Quad-core ARM A57 @ 1.43GHz), Raspberry Pi 4 (Quad-core CortexA72 @ 1.5GHz), Raspberry Pi 3 (Quad-core Cortex-A53 @ 1.4GHz)	inference on edge	–	95.27% accuracy value for NSVF classes on MIT-BIH Supraventricular Arrhythmia Database	CNN
[39]	Arduino UNO and Raspberry Pi 3B+	inference on edge	–	97% for accuracy value for: rhythm, QRS widening, ST depression, and ST elevation categories for detecting arrhythmia disorders on MIT Arrhythmia dataset	DNN
[43]	Raspberry Pi 3	training on cloud, inference on edge	–	98% for accuracy value for: normal(NOR), Left Bundle Brunch Block(LBB), Right Bundle Brunch Block (RBB), Paced beat(PAB), Premature Ventricular Contraction(PVC), Atrial Premature Contraction(APC), Ventricular Flutter Wave(VFW) and Ventricular Escape Beat(VEB) beats for detecting arrhythmia disorders on MIT Arrhythmia dataset	CNN
Our work	ST SensorTile	inference on edge	$9\,mW$ [II]	MIT-BIH dataset, NLRAV and NSVFQ classes: 97.42% and 96.98% for accuracy value, 98.26% and 98.22% for sensitivity value, 98.28% and 98.52% for precision value, respectively	quantized CNN

[I] When fully active.
[II] The device adapts itself to the workload and operating mode, the reported value is the power consumption in case of maximum workload.

In this paper, attention will be focused only on the sensor node.

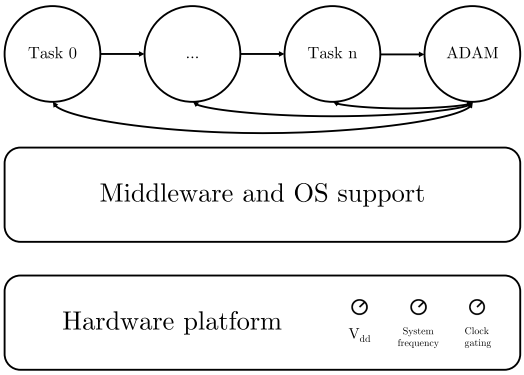


FIGURE 2. IoMT node architecture overview.

IV. IoMT NODE ARCHITECTURE

The sensor node architecture itself can be seen as a layered structure, schematized in Figure 5. In the following sections, a detailed description of each level is provided. The bottom layer is the hardware platform, which may be any kind of programmable microcontroller, that integrates sensors to take care of data acquisition, one or more processing elements, to manage housekeeping and pre-processing, and an adequate set of communication peripherals, implementing transmission to the gateway. The hardware platform is managed at runtime by a firmware/middleware level, potentially including some operating system (OS) support, to enable the management and scheduling of software threads. Moreover, this level must expose a set of low-level primitives to control hardware architecture details (e.g. access to peripherals, frequency, power operating mode, performance counting, etc.), and a set of monitoring Application Programming Interfaces (APIs) to continuously control the status of the hardware platform (e.g. energy and power status, remaining battery lifetime) and to characterize the performance of the different application tasks on it.

At the top of the node structure, there is the software application level, which executes tasks designed according to an adequate application model based on process networks, to be easily characterized and dynamically changed at runtime.

To implement adaptivity, we add to the application an additional software agent, that we call ADAM (ADAptive runtime Manager), which is in charge of monitoring all the events that may trigger operating mode changes (workload changes, battery status, commands from the cloud) and reconfigures the process network accordingly, to minimize power/energy consumption. Reconfiguration actions may involve changes in the process network topology (activation/deactivation of tasks and restructuring of the inter-task connectivity) and playing with the power-relevant knobs exposed by the architecture (e.g. clock frequency, power supply, supply voltage). As mentioned, to assess the feasibility of our approach based on dynamic reconfiguration, we have used a single-core microcontroller, namely an off-the-shelf platform designed by STMicroelectronics named SensorTile. In the following, we will describe the main features of such platforms, exploited in this work.

We have chosen SensorTile to represent a class of platforms available on the market equipped with a single-core low-power IoT node, usually integrating a wide scope of sensors and peripherals to increase usability. These solutions often integrate mid- to low-end processing elements, capable of executing simpler near-sensor processing tasks on a low energy budget, using optimized libraries to recover performance and lightweight operating systems to enable the coexistence of multiple software processes.

A. HARDWARE PLATFORM LAYER

The SensorTile measures $13.5 \times 13.5\ mm$. It's equipped with an ARM Cortex-M4 32-bit low-power microcontroller. The small size and low power consumption allow the device to be powered also by the battery and obtaining good results in terms of autonomy without having to give up portability. Several architectural knobs can be used to adapt the platform to different conditions. SensorTile can work in two main modalities: *run mode* and *sleep mode*, in which different subsets of the hardware components are active. Moreover, in each mode, the chip can be set to a different system frequency (from 0.1 MHz to 80 MHz). Depending on the chosen system frequency and operating state, the device uses different voltage regulators to power the chip.

In Table 2, we list some configurations selectable using the mode-management APIs offered by the platform vendor.

TABLE 2. SensorTile current consumption in different operating states.

RUN (Range 1) at 80 MHz	120 $\mu A/\ MHz$
RUN (Range 2) at 26 MHz	100 $\mu A/\ MHz$
LPRUN at 2 MHz	112 $\mu A/\ MHz$
SLEEP at 26 MHz	35 $\mu A/\ MHz$
LPSLEEP at 2 MHz	48 $\mu A/\ MHz$

For our experiments, we have chosen to use two approaches to dynamically reduce power consumption:

- to change system frequency (and consequently voltage regulator settings) over time according to the workload.
- to use the sleep mode of the microcontroller whenever possible. The operating system automatically sets a sleep state when there are no computational tasks queued to be performed and a timer-based awakening can be used to restart the run mode when needed.

B. MIDDLEWARE/OS LAYER

In addition to the API offered by the manufacturer, we used other middleware components to manage multiple computation tasks at runtime and to execute CNN-based near-sensor processing with an adequate performance level.

1) FREERTOS

SensorTile runs FreeRTOS as RTOS (Real-time Operating System). This firmware component is aimed at developers who intend to have a real-time operating system without too much impact on the memory footprint of the application. The size of the operating system is between 4 kB and 9 kB. Some features offered by the operating system are real-time scheduling functionality, communication between processes, synchronization, time measurements. One of the most important aspects that led us to choose FreeRTOS is that of having the possibility to enable thread-level abstraction to represent processing tasks to be executed on the platform and to timely manage their scheduling at runtime.

FreeRTOS creates a system task called *idle* task, which is set with the lowest possible execution priority. When this task is executed, the system tick counter is deactivated and the microcontroller is put in a sleep state. Due to the priority setting, the idle task is only executed if there are no other tasks waiting to be called by the scheduler.

FreeRTOS does not natively support the frequency variation of the system. Once the frequency has changed, timing functions would be completely de-synchronized. We had to modify part of the OS support, to enable system frequency changes without impact on the rest of the OS functionality.

2) CMSIS

In order to be capable of executing in-place processing of the sensed data, we have exploited the Cortex Microcontroller Software Interface Standard (CMSIS), an optimized library specifically targeting Cortex-M processor cores [45]. It includes several modules having many libraries capable of optimizing mathematical functions based on the type of architecture used. Of particular interest is the CMSIS-NN module, inside there are various optimized functions that allow cognitive computational implementations. While CMSIS provides quite extensive support for neural network execution, we had to add some changes to support the use cases that are described in the following, namely to enable mono-dimensional convolutions on one-dimension sensor data streams.

C. APPLICATION MODEL

In this section, we describe the application model that we have used to create and analyze the application, the source code is available at our public repository.[1] We selected an application structure based on process networks. Tasks are represented as independent processes, communicating with each other via FIFO structures, using blocking *read* and *write* communication primitives to avoid data loss in case of busy pipeline stages. Processes may be potentially executed in parallel, in case of available processing resources, potentially improving performance using a software pipeline.

[1] https://github.com/matteoscrugli/adam-iot-node-on-stm32l4

In particular, for each sensed variable to be monitored, we build a chain of tasks that operate on the sensed data (Figure 3).

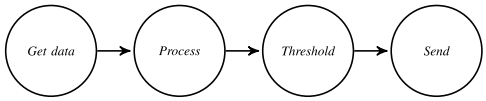


FIGURE 3. Simple task chain.

A chain of processes is generated for each sensor node, so that, if required by changes in the operating mode, it's possible to dynamically turn on and off the useful and non-useful components.

For each sensor, we envision four types of general tasks:

- *Get Data Task:* takes care of taking data from the sensing hardware integrated into the node.
- *Process Task:* it's possible to have multiple tasks of this type, representing multiple stages of in-place data analysis algorithm. Having more than one task of this type allows a prospective user to select, for example, a certain depth of analysis, which determines an impact on the required communication bandwidth, detail of the extracted information, and power/energy consumption.
- *Threshold Task:* this task allows to filter data depending on the results of the in-place analysis. For example, a threshold task may be used to send data to the cloud only when specific events or alert conditions are detected. Its purpose is to limit data transfers from the node.
- *Send Task:* is the task in charge of outwards communication to the gateway.

Considering the selected process network model, activation/deactivation of tasks or entire chains corresponding to sensors can be implemented by:

- enabling/stopping the periodic execution of the involved task;
- reconfiguring the FIFOs to reshape the process chain accordingly.

In this way, it's possible to select multiple application configurations, corresponding to operating modes characterized by different levels of in-place computing effort, bandwidth requirements, monitoring precision.

D. ADAPTIVITY SUPPORT: THE ADAPTIVE RUNTIME MANAGER

Within the process network, a task was exclusively dedicated to the management of dynamic hardware and software reconfiguration of the platform. We have implemented such reconfiguration in a software agent called ADAptive runtime Manager (ADAM). ADAM can be activated periodically by means of an internal timer. It evaluates the status of the system, monitoring:

- reconfiguration commands from the gateway;
- changes in the workload, e.g. rate of events to be processed. For example, a task may have to be executed

periodically, with a rate that depends on the frequency of certain events in the sensor data. This poses real-time constraints that may be varying over time in a data-dependent manner.
- other relevant variables (e.g. battery status).

Depending on such input, ADAM can react to change the platform settings, performing different operations:
- Enable or disable the individual tasks of the sensor task chain or the entire chain;
- Choose whether to set the microcontroller in a sleep mode or not;
- Set the operating frequency of the microcontroller to increase/reduce performance level;
- Reroute the data-flow managed by the FIFOs according to the active tasks.

Figure 4 shows an example of the reconfiguration of the system that may be applied by ADAM, deactivating a *process* task, to switch from an operating mode that sends pre-processed information to the cloud to another sending raw data.

This proposed application model can be easily declined in different use cases and application domains involving sensor monitoring and near-sensor processing. Adapting to new scenarios could require for example to add different or additional *process data* tasks and/or connecting them in a different order. Nevertheless, the possibility of switching between different configurations by reconfiguring the process network would be preserved.

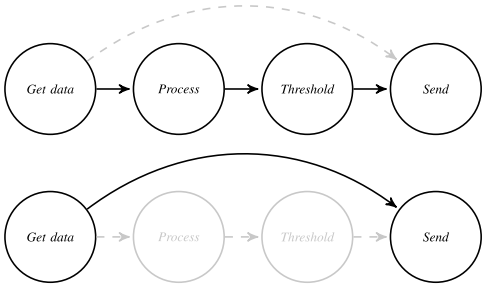


FIGURE 4. Two possible configurations of a generic system.

V. DESIGNING THE APPLICATION: OPERATING MODES AND PROCESSING TASKS

To implement ECG monitoring, we have applied the previously described application model to deploy an adequate waveform analysis application on SensorTile. A single-lead configuration was chosen due to the practical convenience of not having too many electrodes attached to the skin. Lead II is often read individually and therefore adequate in case that as little data as possible needs to be processed. In particular, the modified limb lead II (MLII) configuration was chosen, the same configuration present in the records of the MIT-BIH arrhythmia database, these recordings will be taken into consideration during the training. We built a prototype using an AD8232 sensor module from Analog Devices, connected to the ADC converter integrated into the reference platform. The device supports a single-lead configuration, with two or three electrodes. The official documentation proposes different solutions to manage the signal noise, the compromise is between signal distortion and rejection of motion artifacts. The chosen configuration assumes that the patient remains relatively still during the measurement, and therefore, motion artifacts are less of an issue. The AD8232 is configured with a 0.5 Hz two-pole high-pass filter followed by a two-pole, 40 Hz, low-pass filter. A third electrode is driven for optimum common-mode rejection.

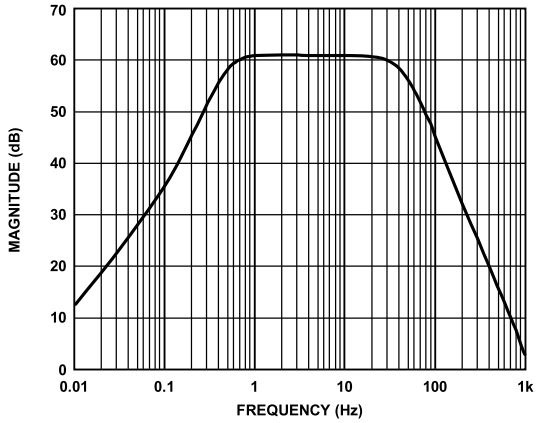


FIGURE 5. Frequency response of the circuit configuration chosen for the AD8232.

In this section, we describe the supported operating modes, that can be selected at runtime, and the processing tasks coexisting in the different operating modes.

A. OPERATING MODES

We have enabled three different operating modes to be selectable by the user, by sending adequate commands from the cloud. Operating modes are shown in Figure 6.

1) OPERATING MODE: RAW DATA

The first operating mode envisions sending the entire data stream acquired by the sensor node to the gateway. There is therefore no near-sensor data analysis enabled, and it poses fairly high requirements in terms of bandwidth. In this operating mode are:
- Multiple samples are been grouped and inserted into a packet of 20 Bytes (8 ECG data 16 bit, 1 timestamp 32 bit).
- The sample rate of the ADC is set to 330 *Hz*, considering sending multiple samples at a time, one Bluetooth packet is sent every 24 *ms*.

2) OPERATING MODE: PEAK DETECTION

This operating mode does not provide visual access to the whole ECG waveform. A healthcare practitioner, when

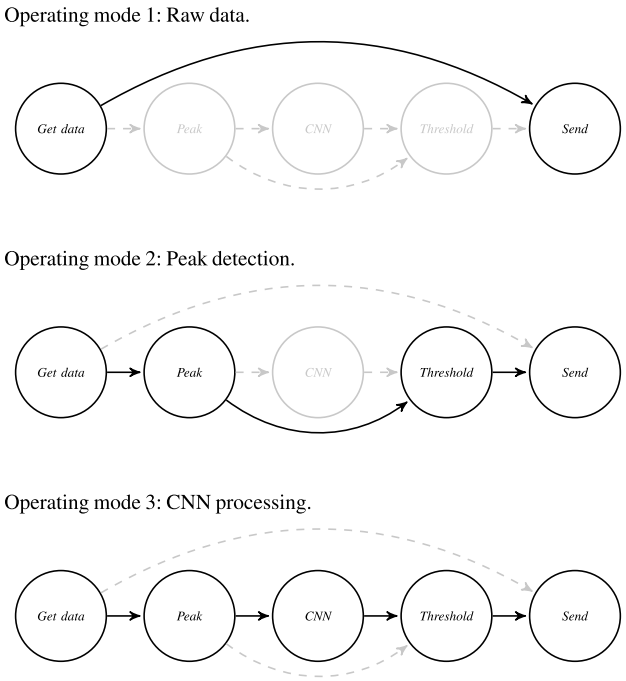


FIGURE 6. ECG application model.

selecting this mode when accessing the data, can select to monitor only heartbeat rate, requiring a lower level of detail in the information sent to the cloud. As a common technique to reduce the power consumption related to communication when sampled values are not interesting, a healthcare practitioner could also set thresholds and receive a notification only when thresholds are exceeded. In this operating mode, four tasks are active:

- Get data task
- Process data (peak detection)
- Threshold task (alert heartbeat rate evaluation)
- Send task

This operating mode processes samples to search for signal peaks and consequently computes the heartbeat rate. The first task (Figure 6) collects data from the sensor (as in raw data operating mode), the second analyzes the signal analysis and calculates the heart rate, and the fourth allows data transmission. The threshold task is used to determine if data must be sent to the cloud. For example, no data is sent if the heartbeat rate is controlled between two high and low alert values. The peak detection algorithm it's not very critical in terms of time and power consumption, it will be better discussed in Section V-B. The size of the package sent is 5 Bytes packet (1 heartbeat rate value, represented on 8 bit, 1 time-stamp 32 bit). The transmission rate is given dependent, in the worst case a package is sent for each peak detected. Thanks to the threshold task, the communication-related power consumption is heartbeat-dependent, since the execution of the send task is triggered only when the heartbeat exceeds the preset threshold defined by the medical staff.

3) OPERATING MODE: CNN PROCESSING

In the latter operating mode, a further level of analysis is introduced. An additional task implements a convolutional neural network, classifying the ECG waveform to recognize physically relevant conditions. Using such classification technique, the practitioner can monitor the morphology of the signal without the need of sending the entire data stream to the cloud, saving transmission-related power/energy consumption. The neural network implemented recognizes anomalous occurrences in the ECG tracing, in this case, communications with the gateway occur only in case of anomaly detection. The enabled tasks are:

- Get data task
- Process data 1 task (peak detection)
- Process data 2 task (CNN)
- Threshold task (anomalous shapes in the ECG waveform)
- Send task

The required communication bandwidth is more similar to *peak detection OM* than *raw data OM*, however, with respect to *peak detection OM*, computing effort is higher. The node executes the 1D convolution neural network similar to the one described in [46]. We have designed the system be capable of classifying ECG peaks according to alternative sets of categories, each composed by 5 classes, named *NLRAV* and *NSVFQ* (see Figure 9). The design process used to select, train and deploy the specific neural network topology is explained in Section V-C.

The size of the data transferred to the cloud is 6 Bytes (1 heartbeat data 8 bit, 1 label data 8 bit, 1 timestamp 32 bit). The CNN, threshold and send tasks are executed only if a peak is detected, the activation frequency of these tasks, therefore, depends directly on the heart rate value.

B. THE PEAK DETECTION ALGORITHM

The processing of the ECG signal is activated in *peak detection* and *CNN OM*, in both operating modes it's necessary to identify the R peaks in the signal, therefore a simplified version of the Pan Tompkins algorithm was used in order to obtain the position of the R peaks during data acquisition from the sensor. The reference study to implement the R peak recognition algorithm is [47].

An exploration was made with different combinations of filters and mathematical functions blocks in order to reduce the computational load as much as possible and at the same time obtain a good level of peak detection accuracy. The accuracy on the MIT-BIH Arrhythmia dataset was validated for each step of the exploration. The Figure 7 shows the block diagram representing the signal processing algorithm chosen after the exploration, composed by: DC filter, low pass filter ($f_c = 11$Hz), derivative and squared block. If CNN OM is enabled, the signal is segmented, a number of samples equal to the input size of the neural network are considered to generate a frame and the detected peak is centered in it.

Figure 8 shows the raw signal in blue color and the filtered one in red from two different recordings. A peak is detected

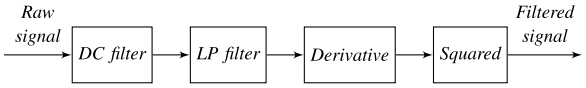


FIGURE 7. Filtering block diagram.

when a filtered signal exceeds a predefined threshold, then returns to a local minimum point and the delay introduced by the filter is taken into account, the threshold value may be set differently for each recording.

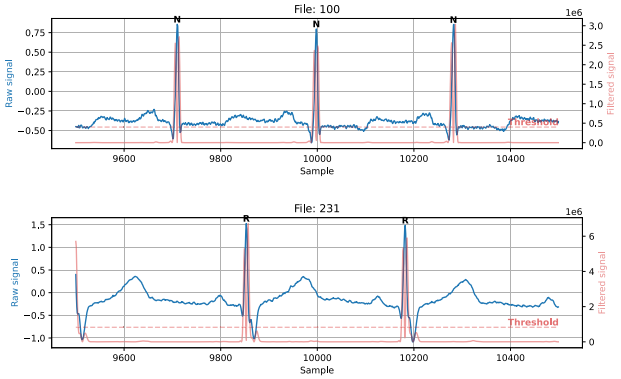


FIGURE 8. Raw signal and filtered one from two different recordings.

A detected peak is considered a true positive when it is associated with a dataset peak in a neighborhood of 50 samples within the track under analysis. Equation 1 and 2 shows the sensitivity (true positive rate) and precision (positive predictive value) data of the peak detection algorithm on the MIT-BIH arrhythmia database:

$$TPR = \frac{TP}{TP + FN} = 0.99674, \tag{1}$$

$$PPV = \frac{TP}{TP + FP} = 0.99421. \tag{2}$$

Table 3 shows true positive, false positive, false negative of the peak detection algorithm with a tolerance of 50 samples.

False positives are the peaks detected by the algorithm that cannot be matched to any in the dataset. False negatives are the peaks in the dataset that are not detected by the detection algorithm. We report in Table 4 the distribution of the type of peaks present in the dataset and in the false positives subset, the NSVFQ labels are reported since they cover all useful types of classes present in the dataset. The extreme case corresponds to analyze an ECG trace containing only V peaks, in this case, 0.97% of peaks are not detected. In literature, there are more advanced real-time algorithms that obtain sensitivity and precision values similar or higher than those obtained with our algorithm, such as [48] and [49]. In [49] more offline algorithms are shown with higher sensitivity and precision values, the authors state that it is easy to address the same results in the real-time case for their method. Our aim is to find an algorithm that achieves high sensitivity and accuracy values and at the same time it provided a low computational load and an easy integration on the microcontroller.

TABLE 3. True positives, false positives, and false negatives of our peak detection algorithm with a tolerance of 50 samples for each ECG recording on MIT-BIH arrhythmia database.

File	*TP*	*FP*	*FN*	*File*	*TP*	*FP*	*FN*
100	2273	0	0	201	1957	5	6
101	1865	7	0	202	2135	3	1
102	2187	0	0	203	2929	75	51
103	2084	0	0	205	2653	0	3
104	2219	27	11	207	1832	192	28
105	2559	61	14	208	2936	47	19
106	2025	6	2	209	3003	9	2
107	2134	1	3	210	2639	24	11
108	1639	30	118	212	2746	0	2
109	2531	5	1	213	3248	1	3
111	2123	3	1	214	2256	6	6
112	2538	4	1	215	3363	2	0
113	1794	0	1	217	2202	2	6
114	1877	2	2	219	2153	0	1
115	1953	0	0	220	2048	0	0
116	2391	7	21	221	2426	4	1
117	1535	6	0	222	2482	3	1
118	2278	6	0	223	2605	6	0
119	1987	1	0	228	2033	53	21
121	1861	2	2	230	2256	3	0
122	2476	1	0	231	1571	0	0
123	1518	3	0	232	1778	4	2
124	1619	0	0	233	3072	1	7
200	2599	10	2	234	2753	0	0

TABLE 4. Classes distribution over the dataset or the false negative subset.

Classes	*Classes distribution*		
	Dataset	*FN subset*	*FN subset / Dataset*
N	90369	255	0,28 %
S	2781	8	0,29 %
V	7230	70	0,97 %
F	803	4	0,5 %
Q	3895	7	0,18 %

C. DESIGNING THE CNN: TRAINING AND OPTIMIZATION

As seen in Section II, introducing deep learning into this type of device brings numerous benefits. Beyond that, we chose the neural network described in this section for its high accuracy, low computational load, latency, and power consumption. These results will be better described in the dedicated section (Section VI).

We have exploited a training procedure using and comprising a static quantization[2] step, the source code is available at our public repository.[3] This process enables the conversion of weights and activations from floating-point to integers and allows to implementation of the CNN using the CMSIS-NN optimized function library, which expects inputs represented with 8-bit precision. In static quantization[2], which takes place right after quantization, *float* values are converted to *qint8* format. We set the procedure to force bias values to be null,

[2]https://pytorch.org/tutorials/advanced/static_quantization_tutorial.html
[3]https://github.com/matteoscrugli/ecg-classification-quantized-cnn

while, to quantize weights, *MinMax* observers[4] are inserted inside the network to detect the output values dynamics in each layer. On the basis of the reported distribution, *scale* and *zero-point* values are selected and used to convert effectively and prevent data saturation.

The functions implementing convolution and fully connected layers in the CMSIS-NN library provide for output shifting operations to apply the *scale* factor on the outputs, allowing for scaling values ranging from −128 to 127. The quantization procedure in PyTorch, on the other hand, requires a *scale* value that is not necessarily a power of 2. For this reason, we slightly modified the CMSIS functions to support arbitrary *scale* values. Such modification has led to a limited increase in inference execution time. As an example of such performance degradation, we report here the execution time increase for two examples CNN topologies, named *20_20_100* and *4_4_100* networks (network name indicates the main topology parameters as *conv1OutputFeatures_ conv2OutputFeatures_ fc1Outputs*), corresponding to respectively 2,87% and 10.52%.

TABLE 5. Hyperparameters used during the training phase.

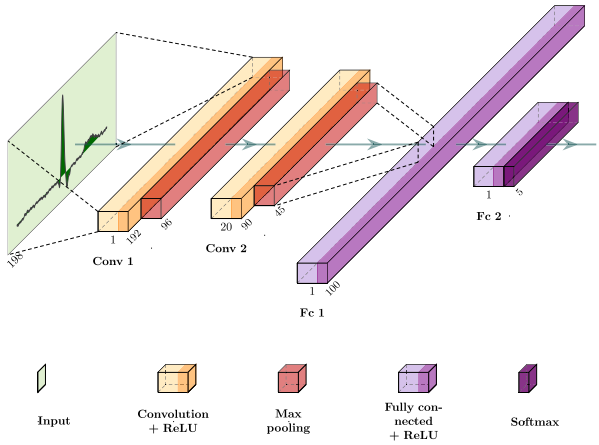
Hyperparameter	*Value*	*Hyperparameter*	*Value*
Epochs	200	Optimizer	SGD
Batch size	32	Learning rate	0.01
Loss criterion	Cross Entropy	Momentum	0.9
ES patience	5	ES evaluation	Every epoch

1) MODEL EXPLORATION

In order to select an optimized CNN topology implementing the classification task required for the system, we have carried out a design space exploration process, comparing tens of neural network topologies in terms of accuracy reached after training and in terms of computing workload associated with executing the inference task on SensorTile. We have explored multiple topologies composed by two convolution layers, two down-sampling layers, and two fully connected layers, as represented in Figure 9, the size of the input sample frame is equal to 198.

Explored topologies feature different numbers of output channels from each layer. The results are reported in Figure 10, showing the most interesting results for both the *NLRAV* and *NSVFQ* classes. Models *NLRAV_20_20_100* and *NSVFQ_20_20_100* achieve the highest accuracy value as shown in Equation 3 and 4. The training set is composed by 70% of the elements of the entire dataset and they are chosen randomly. Figure 11 shows the trend of the accuracy value during the training stage. As shown in Table 5, a maximum number of epochs has been set equal to 200 and, to avoid overfitting effects, the *early stopping* (ES) algorithm was chosen. This algorithm stops the training phase if it detects an increase in the loss value [50], the loss is evaluated every

[4]https://pytorch.org/docs/stable/_modules/torch/quantization/observer.html



(N) Normal beats	**(N)** Normal beats
(L) Left bundle branch block	**(S)** Superventricular ectopic beats
(R) Right bundle branch block	**(V)** Ventricular premature contraction
(A) Atrial premature contraction	**(F)** Fusion beats
(V) Ventricular premature contraction	**(Q)** Unclassificable beat

FIGURE 9. CNN structure and two possible classes of labels.

epoch and a *patience* value of 5 is chosen, i.e. the training stops only if a loss increment is detected for 5 consecutive epochs. The loss values for each epoch during the training phase are reported in Figure 11.

$$ACC_{\text{NLRAV\_20\_20\_100}} = \frac{TP + TN}{TP + TN + FP + FN} = 0.9922, \tag{3}$$

$$ACC_{\text{NSVFQ\_20\_20\_100}} = \frac{TP + TN}{TP + TN + FP + FN} = 0.9889. \tag{4}$$

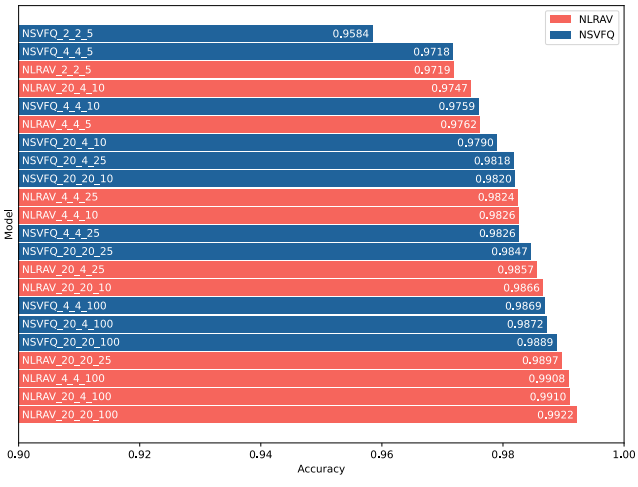
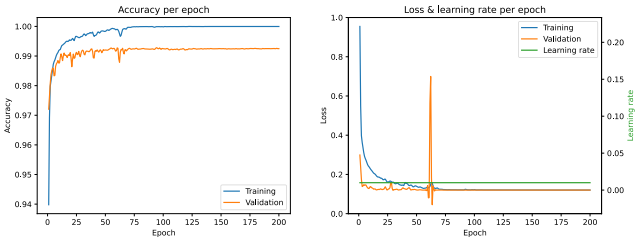
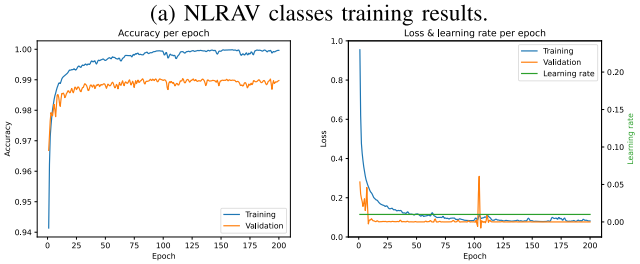


FIGURE 10. Exploration of the chosen neural network model, the name comes from *labels_ conv1OutputFeatures_ conv2OutputFeatures_ fc1Outputs*.

Figure 12 shows a Pareto plot representing accuracy and energy consumption for the most accurate topologies identified by the exploration.



(a) NLRAV classes training results.



(b) NSVFQ classes training results.

FIGURE 11. Results obtained from the training of the model having: 20 output features for *Conv1*, 20 output features for *Conv2* and 100 output for *Fc1*.

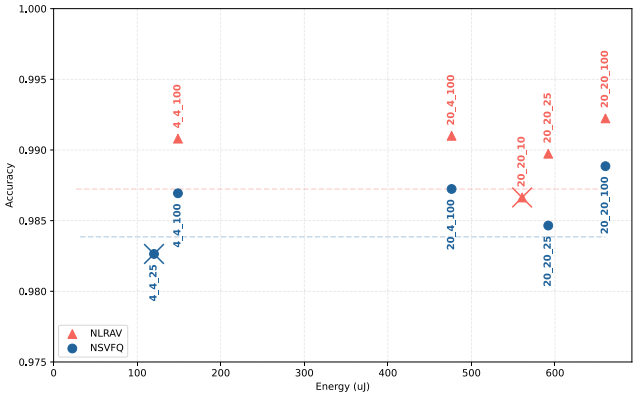


FIGURE 12. For the most accurate models, the energy consumption for a single CNN task call is shown. The dotted lines represents the maximum allowable drop in accuracy (0.5% with respect to the most accurate model) for NLRAV (red line) and NSVFQ (blue line) classes. The models marked with an "×" do not respect the constraints imposed on the minimum necessary accuracy value.

For both classes NLRAV and NSVFQ, only one neural network model must be selected which allows to reduce power consumption as much as possible but, at the same time, does not lead to an excessive drop in accuracy. A maximum accuracy drop equal to 0.5% with respect to the most accurate model (represented in Figure 12 by the dotted lines) was chosen. The reported energy consumption is associated with a single CNN inference task execution on SensorTile. Models that are above the 0.5% threshold are considered to be valid, and, for each set of labels, the valid model that consumes less energy is chosen to be refined in the next steps and deployed on the board. Eventually, we have selected *NLRAV_4_4_100* and *NSVFQ_4_4_100*. The accuracy values are reported in Equation 5 and 6.

$$ACC_{\text{NLRAV}} = 0.9908, \tag{5}$$

$$ACC_{\text{NSVFQ}} = 0.9869. \tag{6}$$

2) POST-DEPLOYMENT DEGRADATION AND REFINEMENT WITH AUGMENTATION

The ECG peaks in the reference dataset are perfectly centered in the frame of samples that is received in input by the CNN during the training stage. As a consequence, the network is trained to recognize the chosen classes as long as the peak is centered in the signal frame. The peak detection algorithm on the SensorTile, on the other hand, operates online on the incoming signals and would not always detect the peak in the same position specified in the dataset.

To assess the accuracy degradation after the deployment, we calculated post-deployment accuracy values by:

- considering false positive and false negative peaks produced by the peak detection algorithm, which need to be accounted for in Equation 5 and 6.
- using a post-deployment validation dataset, composed by the same samples in the original one, but modified to be centered as dictated by the peak detection algorithm during online analysis.

In these conditions, there is a degradation in accuracy, the results will be shown and discussed in Section VI. Pre-deployment accuracy, therefore, does not consider the aforementioned non-idealities.

To overcome the deriving inaccuracy, the chosen networks have been retrained for refining their precision in case of imperfectly centered input frames. We have used a data augmentation technique to create a larger dataset that contains not only perfectly centered peaks in the frame but also off-center ones. Operations like translating the training signal for a few samples in each direction can often greatly improve generalization [50]. We have chosen to create a dataset that contained peaks translated (with respect to those contained in the starting dataset) to the left or to the right by a number of samples multiple of 3, with a maximum translation of 48 samples. Once the dataset augmentation has been generated (larger than the one initially used), during the training phase, a number of elements equal to the size of the dataset initially used are taken into consideration, these elements are chosen randomly for each epoch.

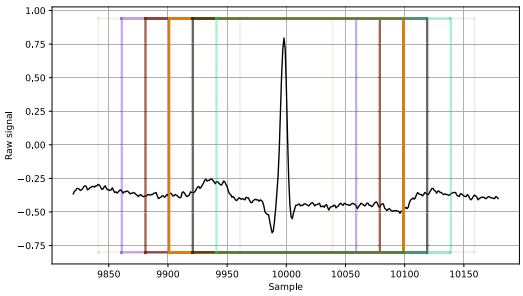


FIGURE 13. Qualitative example of augmentation.

VI. EXPERIMENTAL RESULTS

In this section, we show our main experimental results. We first show a detailed accuracy evaluation to show the

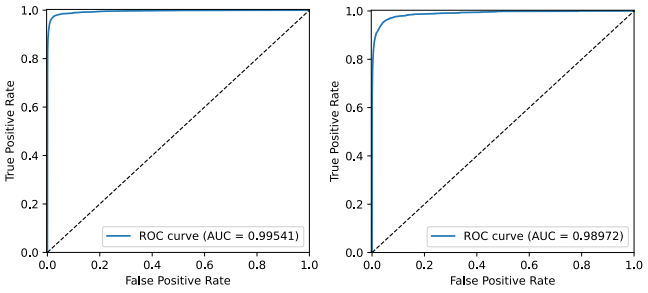
effectiveness of the data augmentation procedure and the class-level classification capabilities of the designed CNNs. Moreover, we present measures of the energy consumption of the entire system and we highlight the energy contributions of each task. To estimate power consumption in each operating mode, we have performed a thorough set of experiments measuring energy consumption in different setup conditions. The results were used to create a model highlighting the contribution of each task to the energy consumption of the node.

TABLE 6. Parameters used for *NLRAV_4_4_100* and *NSVFQ_4_4_100* training phase.

Layer	*Input dimension*	*Output dimension*	*Input features*	*Output features*	*Kernel size*
Convolutional	198	192	1	4	7
Max pooling	192	96	4	4	2
Convolutional	96	90	4	4	7
Max pooling	90	45	4	4	2
Fully connected	180	100	–	–	–
Fully connected	100	5	–	–	–

A. PRE-DEPLOYMENT CNN ACCURACY

As anticipated in Section V-C1, after the topological exploration on neural networks, we selected the *NLRAV_4_4_100* and *NSVFQ_4_4_100* models. A pre-deployment accuracy of 99.08% and 98.69% was obtained for classes *NLRAV* and *NSVFQ* respectively. Table 6 summarizes the layer parameters used during the training phase. Figure 14a and 14b, show the Receiver Operating Characteristic (ROC) curve and Area Under Curve (AUC) value for *NLRAV_4_4_100* and *NSVFQ_4_4_100* with augmentation. Eventually, Figure 15 shows the confusion matrix for the two selected networks.



(a) NLRAV (b) NSVFQ

FIGURE 14. ROC curve and AUC value for *NLRAV_4_4_100* and *NSVFQ_4_4_100* models with augmentation support.

B. POST-DEPLOYMENT CNN ACCURACY

As mentioned above, when considering the ideally centered samples in the dataset, the selected CNNs are very accurate. The precision of the classification, however, decreases significantly when peaks are detected online and imperfectly centered. In fact, for the selected neural network models, a post-deployment accuracy equals to 94.52% and 94.09%

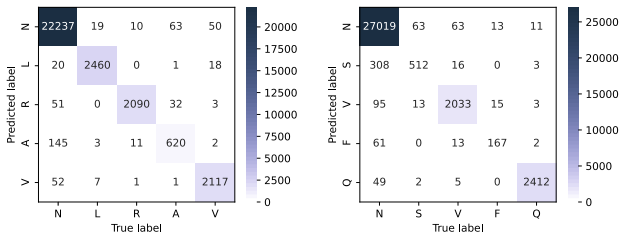


FIGURE 15. Consufion matrix for *NLRAV_4_4_100* and *NSVFQ_4_4_100*.

for *NLRAV_4_4_100* and *NSVFQ_4_4_100* respectively is obtained. As a solution to such accuracy degradation, we have enriched the training set with samples derived from the original ones by applying some artificial shifting as described in Section V-C2. Data augmentation techniques reduce the specialization of the CNN on the perfectly centered validation set, sightly dropping the accuracy to a value of 98.37% and 97.76% for *NLRAV* and *NSVFQ* respectively. On the other hand, ECG recordings with anomalous peaks, that are difficult to be perfectly centered by the peak detection algorithm, are expected to be classified much more accurately. To prove the obtained improvements, we report a detailed classification analysis. In Figure 16, we report the number of false positives and false negatives cases resulting from the peak detection algorithm, and we classify the remaining cases, true positives, with the neural network selected for *NLRAV* and *NSVFQ* classes. Such classification is executed on the post-deployment validation set mentioned in Section V-C.

The improvement in post-deployment accuracy after data augmentation is shown in Equations 7 and 8.

$$ACC_{\text{NLRAV\_post}} = 0.9742, \tag{7}$$

$$ACC_{\text{NSVFQ\_post}} = 0.9698. \tag{8}$$

Data augmentation techniques allows recovering most (around 2.9%) of the drop due to imperfect centering of the input ECG peaks. Data augmentation has obviously no effect on the drop due to misdetections, which still determine 1.7% degradation with respect to the pre-deployment phase.

Peak detection false positive	*Peak detection false negative*
189	107

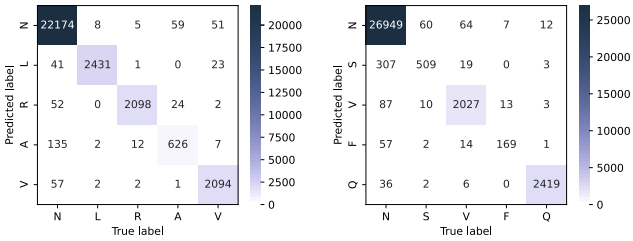


FIGURE 16. False positives and false negatives cases resulting from the peak detection algorithm and classification with remaining true positive cases for NLRAV and NSVFQ classes using CNNs trained with augmentation techniques.

Figure 17 shows a more detailed view of the effects of the quantization procedure and of the augmentation on the accuracy, focusing on the classification of the peaks detected online. The two leftmost plots represent the accuracy levels when no augmentation is exploited. The accuracy, as can be noticed in the leftmost bar of each plot is very high, with small variability over the different tracks, and is only slightly decreased when quantization is applied to obtain a fixed-point implementation. However, when considering the positioning of the peak as identified by the online detection, as shown in the two rightmost bars of each plot, precision degrades on some of the tracks, as can be noticed by the presence of multiple outlier tracks with very bad classification accuracy. This happens independently on the data representation format since the behavior is similar for both the fixed- and floating-point implementations. The two graphs on the right show the impact of data augmentation. As may be noticed by the rightmost bars in these two plots, general accuracy is significantly improved: classification works correctly for all the tracks and even the outliers show an accuracy higher than 90%.

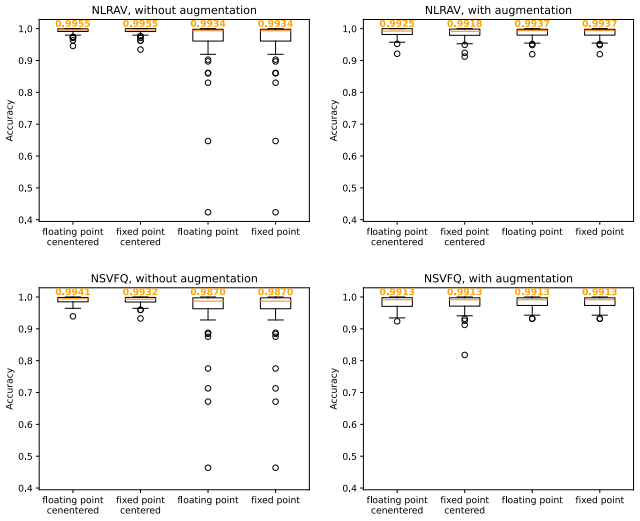


FIGURE 17. Taking into consideration the true positive peaks obtained with a tolerance equal to 50 samples, the statistical distribution of the accuracy values for each ECG recording, obtained from the classification on the validation set, is represented. The floating-point and fixed point models are tested, inference with centered and non-centered peaks is also tested. In orange, the median value.

C. POWER CONSUMPTION MEASURES

With the purpose of measuring the power consumption of our reference platform, the SensorTile board, we monitored the current absorption through an oscilloscope and a Shunt resistor. We used ANALOG Discovery 2 to measure the voltage on the shunt resistor, Figure 18 shows the circuit schematic used to measure the power consumption, in particular: V_{USB} is equal to $5V$, R_1 is the shunt resistor, LD39115J18 (SensorTile component) is a voltage regulator, V_{DD} is the power supply voltage of the entire chip. The voltage regulator holds V_{DD} at a stable voltage of $1.8V$. As input, it accepts a voltage between 1.5 and $5.5V$, then voltage drops on R_1 are expected. The Figure 19 shows some data on power consumption derived from experimental results, the individual cases will then be taken and discussed.

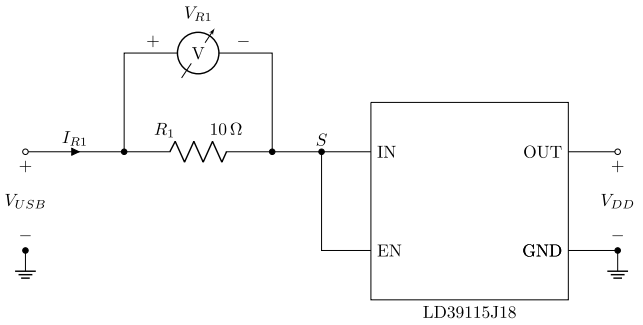


FIGURE 18. Circuit used to measure power consumption.

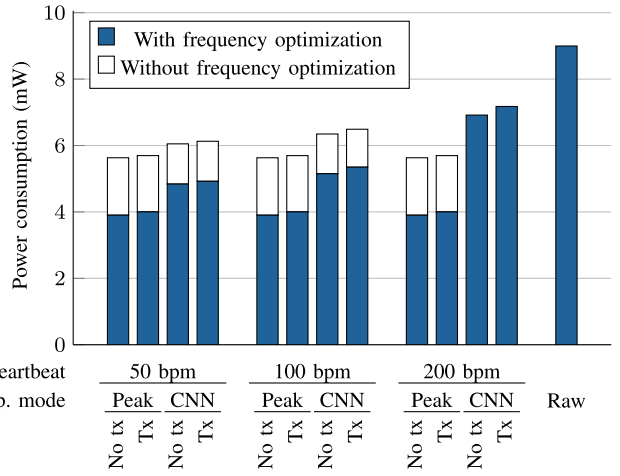


FIGURE 19. The graph summarizes the energy consumption for different heartbeat rates, when data sending is enabled (Tx) or not (No Tx). ***Raw OM*** does not depend on heartbeat nor on the threshold settings and the threshold task is disabled, so only one value is shown.

1) CASE: 50 *bpm*

With low heart rates values, considerable energy savings are obtained even without adapting the system frequency to the workload. In fact, *peak detection OM* and *CNN processing OM* are workload-dependent, which in this case is low. For the latter reason, they consume less than the *raw data OM*, which constantly sends data to the cloud. There is a further energy saving given by the reduction of the system frequency according to the workload, in this case the *peak detection OM* is set to 2 *MHz* and the *CNN processing OM* is set to 4 *MHz*. The *raw data OM*, the worst case, works always at 8 *MHz*. The Figure 19 also shows the power consumption values if data transmission to the cloud is present or not (Tx, No tx), as already said, the decision is up to the threshold task.

2) CASE: 100 *bpm*

Peak detection OM and *CNN processing OM* keep the same operating frequency of the previous case. Thus there is a

slight increase in power consumption for such modes, only due to the more intense data-dependent workload. Obviously, no change in *raw data OM* in terms of power consumption.

3) CASE: 200 *bpm*

Compared to the previous cases, again, there are no changes in the *raw data OM*. An increase of the working frequency to 8 *MHz* is required to sustain the *CNN processing OM*. The role of the threshold task, that implies the difference between the *Tx* and the *No Tx* bar for the *CNN processing OM*, is more important. Even with this very high rate, the CNN-based monitoring is still convenient with respect to the *raw data OM*, confirming the usefulness of near-sensor processing.

D. POWER MODEL AND OPERATING MODE POWER CONSUMPTION ESTIMATION

We have performed a thorough set of experiments measuring energy consumption in different setup conditions. The results were used to create a model highlighting the contribution of each task to the energy consumption of the node. By interpolating the experimental results on power consumption in the different use cases and knowing the duration of each task, we were able to build a model capable of estimating the energy consumption of the device under each possible use case. Table 7 shows the energy values for each task in the process network. Table 8 instead shows the power consumption of the platform in idle state and ECG sensor.

TABLE 7. Summary of consumption and execution time for each task.

Task type	*Number of cycle*	*Execution time (8 MHz)*	*Energy contribution*
Get data	841	$105\,\mu s$	$E_g = 2.96\,\mu J$
Get data + peak	$1\,550 + 841$	$300\,\mu s$	$E_{gp} = 3.76\,\mu J$
CNN 4_4_100	361 360	$45\,ms$	$E_c = 148.78\,\mu J$
CNN 20_20_100	1 719 582	$215\,ms$	$E_c = 660.37\,\mu J$
Threshold	910	$114\,\mu s$	$E_t = 2.73\,\mu J$
Send data	$\sim 25\,000$	$\sim 3\,ms$	$E_s = 83.96\,\mu J$

TABLE 8. Summary of consumption of peripherals.

Device	*Power consumption*		
	2 *MHz*	**4 *MHz***	**8 *MHz***
Platform in idle state	$2.609\,mW$	$3.101\,mW$	$4.546\,mW$
ECG sensor	$237\,uW$	$237\,uW$	$237\,uW$

At this point it's possible to easily estimate the power consumption relative to each operating mode, the resulting equations that calculate the power consumption for each operating mode are:

- $P_{raw\ data\ OM}$

$$(E_g + \alpha E_s) \cdot f_s + P_{idle} + P_{sensor}$$

- $P_{peak\ detection\ OM}$

$$E_{gp} \cdot f_s + (E_t + E_s) \cdot f_p + P_{idle} + P_{sensor}$$

- $P_{cnn\ processing\ OM}$

$$E_{gp} \cdot f_s + (E_c + E_t + E_s) \cdot f_{hr} + P_{idle} + P_{sensor}$$

where:

- f_s is the sampling frequency,
- f_{hr} is the heart rate,
- f_p is the peak data sanding frequency,
- α^{-1} it's the number of samples inserted in a BLE package,
- P_{idle} power consumption of the platform in idle state, depends on the system frequency,
- P_{sensor} energy consumption of the ECG sensor.

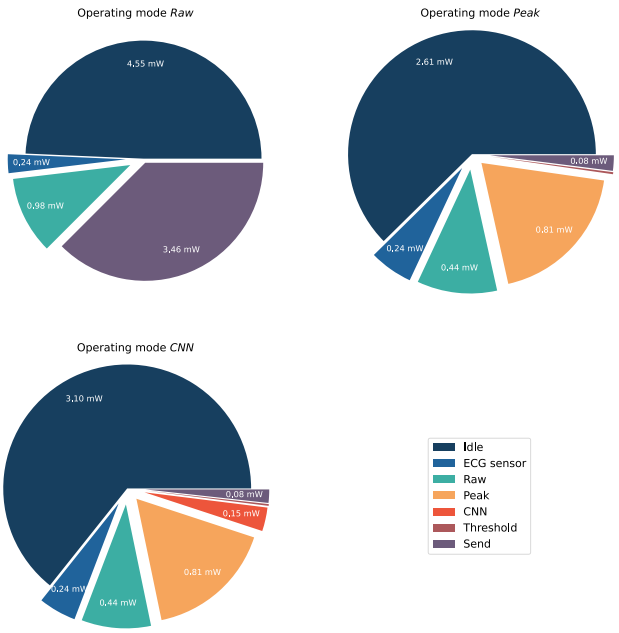


FIGURE 20. Estimation of energy consumption for each task of each operating modes at 60 bpm.

Figure 20 shows the estimate of the power consumption of the device and the contributions of each task in case the heart rate is around 60 *bpm*. The purpose of Figure 20 is to graphically show the power consumption contributions of the tasks for each operating mode. The following list shows the estimated battery life ($600\,mAh$, $3.7\,V$ Li-Ion) for each operating mode:

- *Raw data OM*: 10.29 days
- *Peak detection OM*: 23.49 days
- *CNN processing OM*: 20.20 days

Considering the results in terms of battery life just reported, it is possible to assert that the different optimizations have made possible higher energy efficiency for enabling on-edge processing OMs compared to raw OM. To summarize, the increase in efficiency is mainly due to the introduction of the ADAM component and various optimizations in the design

and development phase. ADMA, to this aim, sets the microcontroller in sleep/active mode and selects, at runtime, optimal clock frequency, and power supply to respects the real-time constraints. These hardware-related settings are also optimized at runtime to exploit the data-dependent nature of the workload (based on the heart rate). Finally, we have considered power optimization as the main objective when taking all the design and development choices. We have used a simplified peak detection algorithm. Moreover, we have explored a large number of neural network models to obtain good results in terms of accuracy values with smaller computational resources, with respect to those used in literature. Thanks to quantization and CMSIS APIs, the single instruction, multiple data (SIMD) microcontroller's capabilities have been exploited as much as possible, significantly increasing the performance of neural network inference and reducing the memory footprint.

VII. WORK COMPARISON

In this section we compare to the works discussed in Section II that deals with inference at-the-edge. Table 9 summarizes the results in terms of neural network accuracy on MIT-BIH dataset. As may be noticed, our system gets results higher or very close compared to the alternatives, despite being, to the best of our knowledge, the only work actually evaluating post-deployment accuracy, and considering all the contributions to errors deriving by all the steps in the online processing system. Only for the precision metric, there are works that report higher values, but exploiting platforms with much higher computational capabilities than those of a microcontroller and more complex neural networks.

In [32] a good results in terms of accuracy and precision is obtained, the F-score value is not reported and was therefore calculated from the reported confusion matrix. Power consumption is not provided but a higher value is expected compared to our as their methodology is tested on platforms such as Jetson Nano and RaspberryPi.

In [37] the inference does not occur in the Cloud side but from an intermediate device placed in the same WBAN network as the sensory node, the latter will have the task of transmitting all the data acquired in raw format thus leading to a possible excessive power consumption of the node. Also in this case the F-score parameter was calculated from the results proposed within the paper.

In [38], a good result in terms of accuracy has been obtained, however, they report far higher power consumption than our methodology. The higher consumption is due to the fact that they used an FPGA-based platform and their system is capable of classifying 335 beats per second.

In [43] they obtain good results in terms of accuracy, although in the work there are not many references to how they were calculated and there are no supporting confusing matrix, making the calculation of the remaining parameters not possible. Here too, a Raspberry is used as a reference platform, which leads to a significantly increasing of power consumption if compared to those obtained in our work.

TABLE 9. Results in terms of accuracy value on MIT-BIH dataset (see classes names in Figure 9).

Work	*Accuracy*	*Sensitivity*	*Precision*	*F-score*	*Diseases*
[32]	95.98%	–	95.9%	93.5%	NSVF
[37]	96%	76.18%	44, 51%	61, 6%	NSVFQ
[38]	97%	–	–	–	NLRAV
[43]	98%	–	–	–	[1]
Our	99.08%	98%	96, 31%	98, 12%	NLRAV
Our	98.69%	95, 52%	92, 6%	96, 16%	NSVFQ
			Post-deployment results		
Our	97.42%	96, 92%	91, 50%	94, 89%	NLRAV
Our	96.98%	95, 35%	85, 17%	91, 12%	NSVFQ

[1] Normal (NOR), Left Bundle Brunch Block (LBB), Right Bundle Brunch Block (RBB), Paced beat (PAB), Premature Ventricular Contraction(PVC), Atrial Premature Contraction (APC), Ventricular Flutter Wave (VFW) and Ventricular Escape Beat (VEB).

A. OUR METHODOLOGY LIMITATIONS

In this work a proof of concept system is presented. Beyond the aim of our research, the presented system shows some limitations that will be briefly commented in this section.

The major limitations are the movement artifacts, since we have chosen an acquisition configuration having a filtering that allows a low distortion of the signal, our system is therefore addressable in a hospital environment, where the patient remains relatively still during monitoring. An interesting future work is to make the most of the capabilities of the neural network to be able to recognize motion artifacts in the signal without having to manually search for the features that need to be monitored.

Another limitation is not having validated our methodology with that provided by the ANSI/AAMI EC57:2012 or BHD (British Hypertension Society) standards.

VIII. CONCLUSION

We have defined a hardware/software template for the development of a dynamically manageable IoMT node, studied to execute in-place analysis of the sensed physiological data. Its implementation has been tested on a low-power platform, able to exploit a CNN-based data analysis to recognize anomalies on ECG traces. The device is able to reconfigure itself according to the required operating modes and workload. The ADAM component, able to manage the reconfiguration of the device, plays a substantial role in energy saving. A quantized neural network reaches an accuracy value higher than 97% on MIT-BIH Arrhythmia dataset for NLRAV and NSVFQ diseases classification. We measured an energy-

saving up to 50% by activating in-place analysis and managing the hardware and software components of the device. This work demonstrates how the feasibility of increasing battery lifetime with near-sensor processing and highlighting the importance of data-dependent runtime architecture management.

REFERENCES

[1] E. J. Benjamin, P. Muntner, A. Alonso, M. S. Bittencourt, C. W. Callaway, A. P. Carson, A. M. Chamberlain, A. R. Chang, S. Cheng, S. R. Das, and F. N. Delling, "Heart disease and stroke statistics—2019 update: A report from the American Heart Association," *Circulation*, vol. 139, no. 10, pp. e56–e528, 2019.

[2] E. Wilkins, L. Wilson, K. Wickramasinghe, P. Bhatnagar, J. Leal, R. Luengo-Fernandez, R. Burns, M. Rayner, and N. Townsend, "European cardiovascular disease statistics 2017," Univ. Bath, Bath, U.K., Tech. Rep., 2017.

[3] R. Maskeliūnas, R. Damaševičius, and S. Segal, "A review of Internet of Things technologies for ambient assisted living environments," *Future Internet*, vol. 11, no. 12, p. 259, Dec. 2019. [Online]. Available: https://www.mdpi.com/1999-5903/11/12/259

[4] Z. Yang, Q. Zhou, L. Lei, K. Zheng, and W. Xiang, "An IoT-cloud based wearable ECG monitoring system for smart healthcare," *J. Med. Syst.*, vol. 40, p. 286, Dec. 2016, doi: 10.1007/s10916-016-0644-9.

[5] L. Roberts, P. Michalak, S. Heaps, M. Trenell, D. Wilkinson, and P. Watson, "Automating the placement of time series models for IoT healthcare applications," in *Proc. IEEE 14th Int. Conf. e-Sci. (e-Sci.)*, Oct. 2018, pp. 290–291.

[6] S. Macis, D. Loi, D. Pani, L. Raffo, S. La Manna, V. Cestone, and D. Guerri, "Home telemonitoring of vital signs through a TV-based application for elderly patients," in *Proc. IEEE Int. Symp. Med. Meas. Appl. (MeMeA)*, May 2015, pp. 169–174.

[7] K. Kaewkannate and S. Kim, *A Comparison of Wearable Fitness Devices*. London, U.K.: InTech, Oct. 2018.

[8] K. Kaewkannate and S. Kim, "A comparison of wearable fitness devices," *BMC Public Health*, vol. 16, no. 1, pp. 1–16, Dec. 2016.

[9] U. B. Baloglu, M. Talo, O. Yildirim, R. S. Tan, and U. R. Acharya, "Classification of myocardial infarction with multi-lead ECG signals and deep CNN," *Pattern Recognit. Lett.*, vol. 122, pp. 23–30, May 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016786551930056X

[10] R. D. Labati, E. Muñoz, V. Piuri, R. Sassi, and F. Scotti, "Deep-ECG: Convolutional neural networks for ECG biometric recognition," *Pattern Recognit. Lett.*, vol. 126, pp. 78–85, Sep. 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865518301077

[11] Y. Li, Y. Pang, J. Wang, and X. Li, "Patient-specific ECG classification by deeper CNN from generic to dedicated," *Neurocomputing*, vol. 314, pp. 336–346, Nov. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231218308063

[12] K. M. R. Tabal, F. S. Caluyo, and J. B. G. Ibarra, "Microcontroller-implemented artificial neural network for electrooculography-based wearable drowsiness detection system," in *Advanced Computer and Communication Engineering Technology*, H. A. Sulaiman, M. A. Othman, M. F. I. Othman, Y. A. Rahim, and N. C. Pee, Eds. Cham, Switzerland: Springer, 2016, pp. 461–472.

[13] M. Magno, M. Pritz, P. Mayer, and L. Benini, "DeepEmote: Towards multi-layer neural networks in a low power wearable multi-sensors bracelet," in *Proc. 7th IEEE Int. Workshop Adv. Sensors Interfaces (IWASI)*, Jun. 2017, pp. 32–37.

[14] S.-Y. Lee, J.-H. Hong, C.-H. Hsieh, M.-C. Liang, S.-Y. C. Chien, and K.-H. Lin, "Low-power wireless ECG acquisition and classification system for body sensor networks," *IEEE J. Biomed. Health Informat.*, vol. 19, no. 1, pp. 236–246, Jan. 2015.

[15] T. Chen, E. B. Mazomenos, K. Maharatna, S. Dasmahapatra, and M. Niranjan, "Design of a low-power on-body ECG classifier for remote cardiovascular monitoring systems," *IEEE J. Emerging Sel. Topics Circuits Syst.*, vol. 3, no. 1, pp. 75–85, Mar. 2013.

[16] N. Bayasi, T. Tekeste, H. Saleh, B. Mohammad, A. Khandoker, and M. Ismail, "Low-power ECG-based processor for predicting ventricular arrhythmia," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 5, pp. 1962–1974, May 2016.

[17] T. Ince, S. Kiranyaz, and M. Gabbouj, "A generic and robust system for automated patient-specific classification of ECG signals," *IEEE Trans. Biomed. Eng.*, vol. 56, no. 5, pp. 1415–1426, May 2009.

[18] A. Amirshahi and M. Hashemi, "ECG classification algorithm based on STDP and R-STDP neural networks for real-time monitoring on ultra low-power personal wearable devices," *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 6, pp. 1483–1493, Dec. 2019.

[19] E. Kolağasioğlu, "Energy efficient feature extraction for single-lead ECG classification based on spiking neural networks," Delft Univ. Technol., Delft, The Netherlands, Tech. Rep., 2018.

[20] G. R. Deshmukh and U. M. Chaskar, "IoT enabled system design for real-time monitoring of ECG signals using TIVA C-series microcontroller," in *Proc. 2nd Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, Jun. 2018, pp. 976–979.

[21] U. Arun, S. Natarajan, and R. R. Rajanna, "A novel IoT cloud-based real-time cardiac monitoring approach using NI myRIO-1900 for telemedicine applications," in *Proc. 3rd Int. Conf. Circuits, Control, Commun. Comput. (I C)*, Oct. 2018, pp. 1–4.

[22] U. Satija, B. Ramkumar, and M. S. Manikandan, "Real-time signal quality-aware ECG telemetry system for IoT-based health care monitoring," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 815–823, Jun. 2017.

[23] G. Xu, "IoT-assisted ECG monitoring framework with secure data transmission for health care applications," *IEEE Access*, vol. 8, pp. 74586–74594, 2020.

[24] V. Natarajan and A. Vyas, "Power efficient compressive sensing for continuous monitoring of ECG and PPG in a wearable system," in *Proc. IEEE 3rd World Forum Internet Things (WF-IoT)*, Dec. 2016, pp. 336–341.

[25] E. Spanò, S. Di Pascoli, and G. Iannaccone, "Low-power wearable ECG monitoring system for multiple-patient remote monitoring," *IEEE Sensors J.*, vol. 16, no. 13, pp. 5452–5462, Jul. 2016.

[26] H. Ghasemzadeh and R. Jafari, "Ultra low-power signal processing in wearable monitoring systems: A tiered screening architecture with optimal bit resolution," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 1, pp. 1–23, Aug. 2013, doi: 10.1145/2501626.2501636.

[27] T. Tekeste, H. Saleh, B. Mohammad, and M. Ismail, "Ultra-low power QRS detection and ECG compression architecture for IoT healthcare devices," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 66, no. 2, pp. 669–679, Feb. 2019.

[28] C. Wang, Y. Qin, H. Jin, I. Kim, J. D. Granados Vergara, C. Dong, Y. Jiang, Q. Zhou, J. Li, Z. He, Z. Zou, L.-R. Zheng, X. Wu, and Y. Wang, "A low power cardiovascular healthcare system with cross-layer optimization from sensing patch to cloud platform," *IEEE Trans. Biomed. Circuits Syst.*, vol. 13, no. 2, pp. 314–329, Apr. 2019.

[29] M. K. Adimulam and M. B. Srinivas, "Ultra low power programmable wireless ExG SoC design for IoT healthcare system," in *Wireless Mobile Communication and Healthcare*, P. Perego, A. M. Rahmani, and N. TaheriNejad, Eds. Cham, Switzerland: Springer, 2018, pp. 41–49.

[30] M. Deshmane and S. Madhe, "ECG based biometric human identification using convolutional neural network in smart health applications," in *Proc. 4th Int. Conf. Comput. Commun. Control Autom. (ICCUBEA)*, Aug. 2018, pp. 1–6.

[31] K. G. R. R. Devi, R. Mahendra Chozhan, and R. Murugesan, "Cognitive IoT integration for smart healthcare: Case study for heart disease detection and monitoring," in *Proc. Int. Conf. Recent Adv. Energy-Efficient Comput. Commun. (ICRAECC)*, Mar. 2019, pp. 1–6.

[32] S. Sakib, M. M. Fouda, Z. M. Fadlullah, and N. Nasser, "Migrating intelligence from cloud to ultra-edge smart IoT sensor based on deep learning: An arrhythmia monitoring use-case," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, Jun. 2020, pp. 595–600.

[33] A. Walinjkar and J. Woods, "Personalized wearable systems for real-time ECG classification and healthcare interoperability: Real-time ECG classification and FHIR interoperability," in *Proc. Internet Technol. Appl. (ITA)*, Sep. 2017, pp. 9–14.

[34] Y. Xitong, D. Yu, and Z. Jianxun, "A real–time ECG signal classification algorithm," in *Proc. 39th Chin. Control Conf. (CCC)*, Jul. 2020, pp. 7356–7361.

[35] M. Naz, J. H. Shah, M. A. Khan, M. Sharif, M. Raza, and R. Damaševičius, "From ECG signals to images: A transformation based approach for deep learning," *PeerJ Comput. Sci.*, vol. 7, p. e386, Feb. 2021.

[36] C. Ma, X. Mu, and D. Sha, "Multi-layers feature fusion of convolutional neural network for scene classification of remote sensing," *IEEE Access*, vol. 7, pp. 121685–121694, 2019.

[37] I. Azimi, J. Takalo-Mattila, A. Anzanpour, A. M. Rahmani, J.-P. Soininen, and P. Liljeberg, "Empowering healthcare IoT systems with hierarchical edge-based deep learning," in *Proc. IEEE/ACM Int. Conf. Connected Health, Appl., Syst. Eng. Technol.*, Sep. 2018, pp. 63–68.

[38] A. Burger, C. Qian, G. Schiele, and D. Helms, "An embedded CNN implementation for on-device ECG analysis," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops)*, Mar. 2020, pp. 1–6.

[39] L.-R. Yeh, W.-C. Chen, H.-Y. Chan, N.-H. Lu, C.-Y. Wang, W.-H. Twan, W.-C. Du, Y.-H. Huang, S.-Y. Hsu, and T.-B. Chen, "Integrating ECG monitoring and classification via IoT and deep neural networks," *Biosensors*, vol. 11, no. 6, p. 188, Jun. 2021.[Online]. Available: https://www.mdpi.com/2079-6374/11/6/188

[40] S. M. Mathews, C. Kambhamettu, and K. E. Barner, "A novel application of deep learning for single-lead ECG classification," *Comput. Biol. Med.*, vol. 99, pp. 53–62, Aug. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482518301264

[41] G. Sannino and G. De Pietro, "A deep learning approach for ECG-based heartbeat classification for arrhythmia detection," *Future Gener. Comput. Syst.*, vol. 86, pp. 446–455, Sep. 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17324548

[42] S. Kiranyaz, T. Ince, and M. Gabbouj, "Real-time patient-specific ECG classification by 1-D convolutional neural networks," *IEEE Trans. Biomed. Eng.*, vol. 63, no. 3, pp. 664–675, Aug. 2016.

[43] D. Hou, M. D. Raymond Hou, and J. Hou, "ECG beat classification on edge device," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2020, pp. 1–4.

[44] M. A. Scrugli, D. Loi, L. Raffo, and P. Meloni, "A runtime-adaptive cognitive IoT node for healthcare monitoring," in *Proc. 16th ACM Int. Conf. Comput. Frontiers*, Apr. 2019, pp. 350–357.

[45] L. Lai, N. Suda, and V. Chandra, "CMSIS-NN: Efficient neural network kernels for arm Cortex-M CPUs," *CoRR*, vol. abs/1801.06601, pp. 1–10, Jan. 2018.

[46] D. Li, J. Zhang, Q. Zhang, and X. Wei, "Classification of ECG signals based on 1D convolution neural network," in *Proc. IEEE 19th Int. Conf. e-Health Netw., Appl. Services (Healthcom)*, Oct. 2017, pp. 1–6.

[47] J. Pan and W. J. Tompkins, "A real-time QRS detection algorithm," *IEEE Trans. Biomed. Eng.*, vol. BME-32, no. 3, pp. 230–236, Mar. 1985.

[48] V. Gupta, M. Mittal, and V. Mittal, "R-peak detection using chaos analysis in standard and real time ECG databases," *IRBM*, vol. 40, no. 6, pp. 341–354, Dec. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1959031818303166

[49] J. Laitala, M. Jiang, E. Syrjälä, E. K. Naeini, A. Airola, A. M. Rahmani, N. D. Dutt, and P. Liljeberg, "Robust ECG R-peak detection using LSTM," in *Proc. 35th Annu. ACM Symp. Appl. Comput.*, New York, NY, USA, Mar. 2020, pp. 1104–1111, doi: 10.1145/3341105.3373945.

[50] 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

MATTEO A. SCRUGLI received the master's degree (Hons.) from the University of Cagliari, in 2018, where he is currently pursuing the Ph.D. degree in electronic engineering with the DIEE Department. His research interests include 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. Recently, the work is focused on cognitive IoT devices, based on single-core or multi-core platforms, and capable of enabling edge-computing.

DANIELA LOI received the M.Sc. degree in electronic engineering from the University of Cagliari, in 2006, and the Ph.D. degree in bioengineering from the University of Genoa, in 2010. She is currently the Project Manager with the Department of Electrical and Electronic Engineering (DIEE), University of Cagliari, involved in the implementation of deep learning algorithms on low-energy platforms.

LUIGI RAFFO 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. In 1994, he joined the Department of Electrical and Electronic Engineering, University of Cagliari, Italy, as an Assistant Professor, and as an Associate Professor, in 1998, where he has been a Full Professor in electronics with the Department of Electrical and Electronic Engineering, since 2006. He teaches courses on system/digital and analog electronic design and processor architectures for the Courses of studies in electronic and biomedical engineering. He was a Coordinator of the Project EU IST-FET—IST-2001-39266—BEST and the MADNESS EU Project (FP7/2007-2013). He was also an Unit Coordinator of the Project EU IST-FET—SHAPES—Scalable Software Hardware Architecture Platform for Embedded Systems. He is also a Local Coordinator of industrial projects in the field (among others: ST-Microelectronics—Extension of ST200 architecture for ARM binary compatibility and ST-Microelectronics—Network on chip). He is responsible for the cooperation programs in the field of embedded systems with several other European Universities. He was a Local Coordinator of the ASAM (ARTEMIS-JU) and ALBA projects (national founded project) and RPCT (regional founded project).

PAOLO MELONI (Member, IEEE) received the Ph.D. degree in electronic engineering and computer science, in October 2007, presenting the thesis "Design and Optimization Techniques for VLSI Network on Chip Architectures." He is currently an Assistant Professor with the Department of Electrical and Electronic Engineering, University of Cagliari, Cagliari, Italy. His research interests include the development of advanced digital systems, with special emphasis on the application-driven design of multicore on-chip architectures. He is the author of a significant record of international research papers and a tutor of many Bachelor and Master students thesis in electronic engineering. He is teaching the course of embedded systems with the University of Cagliari. He is also a Scientific Coordinator of the ALOHA Project (www.aloha-h2020.eu).