Hardware-assisted and Deep-Learning techniques for Low-Power Detection of Cardiovascular Abnormalities in Smart Wearables

A. Catalani*, I. Chatzigiannakis*, A. Anagnostopoulos*, G. Akrivopoulou†, D. Amaxilatis‡ and A. Antoniou‡
*Sapienza University of Rome, Rome, Italy
†University of Patras, Patras, Greece
‡Spark Works Ltd, Galway, Ireland

***Abstract*—The COVID-19 pandemic has significantly reduced visits to hospitals and clinics, forcing physicians and clinics to investigate how to move online using telemedicine and home monitoring. Wearable technologies can help by enabling homecare monitoring if they provide accurate and precise measurements. The monitoring of cardiac health problems is such an example and can be managed when patients are residing at home with the use of wearable cardiac monitoring equipment. Recent studies indicate that of various COVID-19 related complications, cardiac abnormalities in particular are associated with a significantly higher mortality rate. It is therefore important to develop smart wearables that are able to analyze and interpret the recorded signal to detect anomalies outside clinical environments where no external devices are available to analyze and store the signals, nor healthcare personnel is present to assist the identification of abnormal heart activity. This paper looks into two different approaches to enable smart wearables to analyze a high-definition electrocardiogram arriving from ECG sensors arrays in order to detect cardiovascular abnormalities. The first approach relies on techniques that enable the execution of deep-learning models within an embedded processor. The second approach uses heterogeneous multicore embedded processors that accelerate the execution of the classifiers. Results indicate the benefits of each approach and the interplay between the performance achieved in terms of event detection ratio and latency of classification.**

***Index Terms*—Cardiac Arrhythmias, ECG signal classification, Deep learning, Convolutional neural networks, Wearable Electronics, Experimental Evaluation**

I. Introduction

Obtaining reliable and useful datasets originating from low-end devices is one of the main challenges in the IoT domain, tackled by in-network aggregation and on-the-spot data management techniques. Yet, as the total number of interconnected nodes rises, an always increasing amount of barren data is collected and forwarded to the backend servers for further processing, which consumes bandwidth. As a result network resources become saturated by the constantly increasing information flow and hardware overprovisioning becomes the only way of effectively addressing this situation [34].

Medical applications have additional requirements, given the fact that minimal end-to-end latency, network bandwidth preservation and enhanced reliability are considered attributes of paramount importance [5]. In addition, robust data collection, storage and availability mechanisms, paired with failover schemes for uninterrupted services even in cases of intermittent cloud connectivity or resource-constrained wearable devices must also be integrated [6], [8]. As more data traverse the network, the possibility of errors proportionally rises, since bit error rate, data transmission latency and packet droppings are linked to the actual size of the transmitted data. Yet, in an emergency or safety-critical applications, there is no margin for errors. When these restrictions are combined with the prerequisites for (i) near real-time data processing, (ii) enhanced end-to-end data security, and (iii) data anonymization and privacy, formulate a set of challenges in which only radical, end-to-end solutions can be applied.

Cardiology is the medical branch in which applications based on wearable monitoring devices appear to thrive. Given the fact that 47% of all deaths in Europe are related to cardiovascular diseases [4], [29], it is essential to provide the means for doctors to closely and efficiently monitor heart conditions that will save thousands of lives annually [25]. To identify variations in heart rhythm along with other patterns of the heart's electrical impulses, wearables mostly use Electrocardiography monitoring (ECG). This technique monitors the electrical activity of the heart and records electrical impulses generated by the polarization and depolarization of cardiac tissue through properly placed electrodes.

Portable ECG devices that provide accurate real-time heart monitoring and detect sporadic events during doubtful sections of incidents have the potential to enhance the decision-making process of doctors from remote. The portable devices that are available today for providing accurate diagnosis have still certain limitations that prevent their widespread adoption.

First, currently available portable ECG devices, such AliveCor Heart Monitor[1] or the HeartCheck Pen[2], rely on 1-3 leads and have *limited accuracy* due to electromagnetic noise. Environmental noise from nearby appliances or noise originating from muscular activity frequently produces abnormalities in a recording [21]. One way to address noise is to increase the number of electrodes that record cardiac activity

[1]https://www.alivecor.com,
[2]http://www.theheartcheck.com/

from different heart regions. Of course, the additional ECG leads will require more energy to operate. Moreover, recording the signal arriving from multiple sensors will require increase memory capacity [7]. Under such circumstances, it remains a challenge to increase accuracy while at the same time also maintaining the longevity of the portable system.

Second, existing portable ECG devices *lack computational power and have limited battery life* to detect abnormal behaviour on the spot. Higher-end wearable ECG monitors, such as QardioCore[3] and Equivital[4] do not offer on-the-spot analysis of the signal. Instead, through a mobile phone, they wirelessly transmit the recorded signal to a cloud-based service that analyzes the signal and shares the results with doctors [22]. It is important to keep in mind that high-definition recordings that utilize more than six or more electrodes produce large amounts of data. Transmitting these data from the portable device to the mobile phone and eventually to the cloud service has an impact on the energy consumption of both the portable device and the mobile phone. In addition, the data transferred are usually diagnosed as normal, thus also wasting the bandwidth within the core network infrastructure.

Given the above observations, it is important to examine the possibility of creating a new generation of portable ECG devices that will have the capability to carry out a first-level analysis of the signal using the available embedded processor. The device will be therefore in a position to evaluate which parts of the recorded signals have a significant diagnostic value and are thus important to store and/or transmit to the cloud. Avoiding transmitting all the data that do not assist to deliver a diagnosis will reduce the network resources and consequently also the battery resources of the portable devices [2].

Today many methods exist to evaluate an ECG recording that is suitable for execution within an embedded processor. Unfortunately, these algorithms are characterised by very low accuracy, leading to a generation of an excessive number of alerts [36]. The approach of utilizing machine learning techniques [33] or deep neural networks (DNN) [18] have been examined. These approaches achieve high levels of accuracy for diagnosing heart abnormalities by utilizing elaborate models and relying on high-performance computing infrastructures.

One approach to make these models suitable for execution within low-power embedded processor that has limited memory capacity is to transform them to low-power variants: both at the hardware level (e.g., accelerators [10], [13]) and algorithmic level (e.g., network pruning [20], quantizing the weights in lower resolution integers [30] or stopping the inference chain to avoid unnecessary computations [11], [32]). These solutions remain very close to the original deep neural network approaches and have a limit in the level of power consumption reduction they can achieve.

Another approach is to implement these models using hardware accelerators able to carry out specialized operations with reduced power consumption and within short periods. Examples of such embedded processors that include hardware-assisted pattern matching modules are the NXP MPC8572E PowerQUICC III integrated processor or the Intel® Quark™ C1000 system-on-chip. Such systems-on-chips enable the execution of specific machine learning methods with significantly reduced power consumption requirements. Although each hardware-assisted classifier has limited memory capabilities, a system can be designed that builds upon multiple such modules to expand the accuracy of the resulting model.

In this study, two different methods for ECG heartbeat classification within a microcontroller have been evaluated. The first approach builds upon a hardware-assisted KNN implementation that allows the classification of ECG recordings with a very short latency and achieving low power efficiency. The second approach uses a deep convolutional neural network with residual connections for the arrhythmia classification task that is optimized to operate within a microcontroller. The results indicate that arrhythmia detection within low-power wearable devices can make predictions with reasonable high precision when compared to the state of the art methods in the literature of high-end processor architecture.

The rest of the paper is organized as follows. Sec. II provides an overview of recent results in the relevant literature. The formulation of the problem and the methodology of the two approaches investigated are presented in Sec. III. The experimental evaluation is presented in Sec. IV followed by a discussion of the results and future work directions in Sec. V.

II. Related work

The analysis of ECG signals for the diagnosis of heart diseases has attracted significant attention in the research community due to its importance. During the past years, several algorithms have been proposed that can detect heart arrhythmias as a specific cardiovascular disease [9]. For many years the use of signal-processing techniques was the predominant tool for detecting abnormal heartbeats and arrhythmias in general [36]. More recently, machine-learning approaches have been used to develop models [33]. Such approaches manage to achieve higher accuracy than the signal-processing-based approaches and tend to generate lower numbers of false alerts. For example, consider the detection of arrhythmias formulated as a classification problem. Then the k-nearest neighbour algorithm can be applied based on analyzing the QRS-complex in an ECG. Another approach is to develop artificial neural networks to classify cardiac anomalies. For example, in [28] an evolutionary neural system is developed that compared to a machine-learning-based classifier can achieve a recognition sensitivity, at a level of 90%. Pławiak [27] proposes to fragment the ECG signal into windows of $10sec$ and estimating the power spectral density to enhance the characteristic features. In the sequel, the extracted and enhanced features are used by a revolutionarily effective ensembles of classifiers. The evaluation of the model indicates that it can identify a total of 17 distinct heart disorders with a precision of 91.40%, which is the best results obtained to date, to our knowledge. In [39]

[3]https://www.getqardio.com/qardiocore-wearable-ecg-ekg-monitor-iphone/
[4]www.equivital.co.uk

the concept of using a 1-dimensional convolutional neural-network model is proposed. In contrast to previous attempts for designing deep neural network models, the proposed method does not use any elaborate feature extraction technique. As a result, classification can be carried out in real-time while at the same time achieving high accuracy. More recently, Rajpurkar et al. [31] propose a much deeper convolutional neural network that is made up of 34-layers. The network is trained to directly map ECG samples without conducting any kind of preprocessing or feature extraction to rhythm classes. The evaluation of the resulting system is carried out in comparison with experienced cardiologists that examined the recorded ECG against a broad range of cardiac anomalies. The results indicate that computerized detection based on a deep neural network can potentially outperform human cardiologists in terms of predictive value (precision) and sensitivity.

III. Methodology

An ECG recording is acquired by placing some electrodes on specific locations on the patient to obtain the correct view (or lead) of the operation of the heart. ECG devices that are capable of recording three, six or twelve channel ECGs are essentially capable of obtaining multiple views of the same cardiac cycles. The activity of the heart during a beat is based on a synchronized contraction of the ventricles to fills the atria. This activity is registered by the electrodes, the so-called $PQRST$ wave. In more details, the Sinoatrial (SA) node starts with a deflection that generates the P wave. Then the Atrioventricular (AV) junction is reached after a small delay due to the slower tissues. This is the PR segment of the wave that represents the electrical inactivity. Next, the depolarization of the ventricles, which are greater than the atria, require more energy to operate. This is reflected in the much larger QRS complex in comparison to the P wave. So the first negative deflection after the P wave defines the Q wave. Immediately after this, the first positive deflection follows which defines the R wave. Then the second negative deflection defines the S wave. Immediately after the depolarization, another wave in the ECG starts with the next repolarization phase.

The analysis of the $PQRST$ wave and the detection of arrhythmias is a difficult task due to the high variability of the heart's mechanisms. The assessment of cardiac activity is based on the morphology of the $PQRST$ wave of each heart-beat. The advancement of medical instrumentation (AAMI) standard [11] categorizes arrhythmias in two broad classes: *rhythmic* arrhythmias that are made up of a series of irregular beats and (2) *morphological* arrhythmias that represent a single abnormal beat. The morphological arrhythmias are further organized into five classes: the N, S, V, F and Q. N includes beats that result in a QRS complex that is longer than $120\,ms$ due to a blocked SA node, either in the left or the right bundle branch, that generates an impulse that does not reach the left or right ventricle. S includes the so-called supraventricular ectopic beats (SVEBs) where the P wave is upside-down. V includes the so-called ventricular ectopic beats (VEBs) where the QRS complex is tall and wide. F is a class made up of the fusion of normal and V. Finally, all other types of beats, as well as beats regulated by a pacemaker, form the Q class. It is therefore evident that the identification of the onset and offset points and the amplitude measurements of each part are diagnostically relevant. Moreover, the interval between the PQ waves and the PR waves, the QRS width, the QT interval, the amplitude of QRS, the ST level are also important when evaluating the cardiac rhythm.

A. Problem Formulation

The ECG arrhythmia detection task is a sequence-to-sequence task which takes as input an ECG signal that is decomposed into k fragments denoted as $\mathcal{S} = [s_1, \ldots, s_k]$, and outputs a sequence of labels, one for each segment, denoted as $\mathcal{L} = [l_1, \ldots, l_k]$, where each l_i can take on one of m different rhythm classes. Based on such a signal S and output sequence L, a training set is selected in a way such that the following cross-entropy objective function is optimized:

$$\text{loss}(\mathcal{S}, \mathcal{L}) = \frac{1}{k} \sum_{i=1}^{k} \log p\left(R = l_i | \mathcal{S}\right)$$

where $p(\cdot)$ represents the probability that the label l_i will be assigned to the ith segment.

In this paper, a "simplified" arrhythmia detection task is defined where the sequence of labels are just two: Normal (the F class stated above) and Abnormal (the classes N, S and Q). The simplified version of the problem is considered as a first step towards providing low-latency classification without relying on any external/cloud-based services that require continuous connectivity.

B. Hardware-assisted Detection

The prototype is based on the Intel® Quark™ SE C1000 system-on-chip, that combines an x86 microcontroller (MCU) operating at 32 MHz clock speed, with a sensor subsystem and pattern-matching capability through a hardware-accelerated engine. The hardware-accelerated engine offers optimized power management and low battery power, enabling the MCU to learn through pattern recognition and differentiate appropriate response events in real-time while being more energy-efficient. The system provides 80 KB SRAM, 384 KB integrated Flash, and 8 KB OTP.

The Zephyr [12] real-time operating system is used to develop the firmware. Zephyr OS is developed specifically for resource-constrained heterogeneous devices. It is released under the Apache License 2.0. The code controlling the hardware-accelerated Pattern Matching Engine (PME) inside the Quark is developed using the Intel® Curie™ ODK[5].

The PME can be trained and used on abitrary types of data. The core technology of the hardware is the so-called NeuroMem® [38] developed by General Vision Inc. The total number of labels used is limited to 255. Internally the module provides a total of 128 neurons that are used to store the model. During the training phase, each of these neurons is responsible

[5]https://software.intel.com/en-us/node/674972

for identifying specific aspects of the provided features. Each neuron uses a 128-byte array to store the selected features. The PME monitors the activation of the neurons and measures their influence in characterizing each input signal. Neurons that do not achieve a minimum influence, they are said to be "degenerated" and are removed. It is therefore possible that a smaller number of neurons is used after the end of the training phase. During the classification of a given signal, the neurons that have values close to those provided are actived. The other neurons remain idle. Given the numeric values associated with the pattern stored in the neuron, the provided signal is classificed based on the proximity to the patterns stored in the activated neurons.

The prototype presented here extends the one developed in [3]. The one used here implements a pre-processing that organizes the ECG recording in windows of 10 seconds. The concept is that each window will include at least four QRS consecutive complex that is required by experts to classify certain types of arrhythmias (e.g., PVC: Premature Ventricular Contraction, VF: Ventricular Fibrillation, BII: 2nd Heart Block) [37].

The recorded ECG trace is analyzed to provide global scalar and "spatial" parameters and detailed measurements. These parameters are derived from the representative cycles for each QRS complex present in the window. The parameters are used to construct a vector of features – or a pattern – that is used by the PME. The accuracy of the PME depends on this vector. However, since points are limited to 128 bytes, the number of features that can fit is limited. Moreover, the extraction of the features carried out by the generic x86 MCU needs to be conducted in real-time. So feature selection is important for the overall performance of the system, in terms of accuracy, latency and energy consumption. In this work, feature extraction is based on the algorithm of Pan and Tompkins [26], probably the most widely used algorithm and also fast to execute within an embedded processor environment. The QRS complexes are identified by analyzing the ECG segments in terms of slope, amplitude, and width. A total of eleven features are extracted for each segment: *number of QRS complexes included in the window*, *average R-R interval*, *average S amplitude*, *average R-S amplitude*, *average R-peak amplitude*, *last five R-R intervals*. As soon as the features vector are encoded into a 128-byte array, the hardware-assisted K-Nearest Neighbor (KNN) implementation classifies them within a very low latency and achieving low power efficiency.

During the learning/training phase, each 128-byte array corresponding to the vector of features extracted from the window of 10 seconds is assigned a label classifying the ECG beats recorded during the window. If a window includes at least one abnormal beat, it is characterized as *abnormal*. Otherwise, it is characterized as *normal*.

The byte arrays encoding the features of each 10-second window make up the training pattern used by the PME to construct a decision space. The decision space spans up by 128 orthogonal vectors, where each vector is 128-bytes. Thus the decision space is a 128 axes coordinate system with its axis limited to integers ranging from 0 to 255.

After the training is completed, during the normal operation, the PME identifies the k nearest points for each vector provided. The distance between two points is calculated using the Manhattan Distance (also known as rectilinear distance or Minkowski's L1 distance). The vector provided is classified by averaging the labels of the k closest points.

C. Convolutional Neural Network-based Detection

The prototype is based on a lot more powerful processor from Nordic Semiconductors, the nRF52840 system-on-chip built around the 32-bit ARM® Cortex™-M4 CPU with the floating-point unit running at 64 MHz. This allows larger programs to run on the prototype and with a lot more variables. Exceptionally low energy consumption is achieved using a sophisticated on-chip adaptive power management system.

The software is developed using the Segger [35], a real-time operating system for embedded systems in combination with the Nordic nRF5 SDK version 17.0.2. The Convolutional Neural Network (CNN) model is implemented using the TensorFlow computational library [1].

In contrast to the previous approach, a lightweight feature extraction method is used here thus requiring minimal processing cycles from the MCU. No form of processing is used that makes any assumption about the signal morphology or spectrum. In particular, the continuous ECG signal is split in windows of $10sec$ and the signal within each window is normalized to the range $[0, 1]$. Then the Pan and Tompkins [26] is applied to identify the R-peaks. For each R-peak identified, a signal of the length of $12sec$ is produced, centred on the R-peak. In this way, each window generates multiple different segments, depending on the number of R-peaks included in the period of $10sec$. This process ensures that all the extracted beats have identical lengths which are essential for the proper execution of the CNN model.

The extracted segments centred on an identified R-peak are fed into a CNN model that extends the one developed in [17]. First, a convolutional layer is used to apply a scalar product with the input fragments using 32 kernels of size five each. In contrast to CNN models used for image analysis, the model used here has one dimension. Then a predictor network is used that is made up of five blocks. Each block is made up of a convolutional layer, followed by a Rectified Linear Unit [24] that output only the positive elements of the input. Subsequently, a skip connection [16] is used and finally a max-pooling layer of size five and stride two. The output of the predictor network feeds two fully connected layers made up of 32 neurons that produce the final predictions. Finally, a softmax layer is used to convert the output into the corresponding labels.

During the training phase of the model, for the softmax layers, cross-entropy is used as a loss function. In addition the Adam optimization method [19] is used with the learning rate, beta-1, and beta-2 of 0.001, 0.9, and 0.999, respectively. The learning rate is decayed exponentially with the decay factor of 0.75 every 10000 iterations.

The approach followed in this prototype can be considered the opposite of the previous prototype. Instead of carrying out a complex pre-processing phase that is used by a fast classifier, here signals are processed briefly and analyzed by a deep classifier. This approach has higher requirements in terms of storing the weights of the kernels used.

IV. Experimental Evaluation

The two different approaches are evaluated in analyzing, and interpreting ECG traces through a series of experiments.

A. Data Source

There are various databases available for the evaluation of computerized diagnosis of heart diseases [14]. Among all the publicly available datasets, the *Massachusetts Institute of Technology-Beth Israel Hospital (MIT-BIH) Arrhythmia database* [23] is the most widely used for ECG signal analysis and arrhythmia detection since it includes recordings of all five arrhythmia classes suggested by the AAMI standards [11]. It includes 48 two-channel ECG recordings of $30min$ each, digitized using $360Hz$ and 11-bit resolution to create a total of $650K$ records. The recordings are annotated by two cardiologists indicating a total of $90,585$ beats belonging to the N class, $2,781$ for the S class, $7,235$ for the V class and 802 for the F class.

B. Data Preprocessing

The dataset used is relatively small compared to other use cases for deep learning, such as natural language processing or image analysis. Moreover, the N class is significantly larger than the others. These two issues can potentially lead to overfitting the model to the training dataset, thus failing to provide a model that is generic enough. Such a model would result in assigning beats to the N class with high probability. Although the accuracy of beats belonging to the N class would be very high, it would perform poorly for all other beats belonging to one of the other classes. The problem of class imbalance is predominant in biological datasets where the value of a model is connected to the ability to predict rare events that make up the minority of the total dataset [15].

Among the available methods to solve the class imbalance problem, here the oversampling method of Adasyn [15] is used. The dataset is pre-processed in order to identify less populated classes and introduce additional elements by partially replicating them. The algorithms take special care of samples for which a large fraction of neighbouring samples belong to another class. A *dataset* D made up of n samples is represented as $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$. The *classes* $Y = \{1, -1\}$ are such that $|y_i \in Y$. Then m_s is the number of samples beloging to the under-balanced class and m_l the number of over-balanced classes. In this case $m_s \leq m_l$. Clearly $m_s + m_l = n$.

The method of Adasyn characterizes a class as under- or over-balanced using the threshold function $d = \frac{m_s}{m_l} \in [0, 1]$. An appropriate threshold value d_{th} is selected to limit the maximum class unbalance when $d_i \leq d_{th}$. Given the characterization of each class, the method computes the total number of new samples that need to be generated based using $G = (m_l - m_s) \cdot \beta$, where $\beta \in [0, 1]$. Here β is user defined, for example if $\beta = 1$, the resulting dataset is completely balanced.

After identifying the number of new samples that need to be generated, the k-nearest neighbor algorithm is used to identify which samples to examine. For each sample x_i belonging to the class, the $r_i = \frac{\Delta_i}{K}$ $i \in \{1, \ldots, m_s\}$, is computed as an indication of how isolated the specific sample is from the rest of the points of the same class. Δ_i is the number of the K closest neighbours to x_i, belonging to the other classes. In the sequel r_i is nornalized by applying $\sum_{i=0}^{m_s} \hat{r_i} = 1$, where $0 \leq \hat{r_i} = \frac{r_i}{\sum_{i=0}^{m_s} r_i} \leq 1$. One could consider $\hat{r_i}$ as an indicator for characterizing whether x_i is an outlier within the specific portion of the dataset. The Adasyn methods wants to identify such samples and generate additional elements by replicating such samples.

Given the selection of samples that need to be replicated, the total number of copies is $\sum_{i=0}^{m_s} g_i = G$, where $g_i = \hat{r_i} \cdot G$. Finally, for each minority sample, the Adasyn method creates g_i samples s_j as using the formula $s_j = x_i + r \cdot d_j$.

C. Performance Measures

In a classification problem with n classes $y_1, \ldots, y_n$, it is possible to determine the quality of the classifier, in terms of $D_{i,j}$, that is the number of dataset's samples belonging to y_i, but predicted as y_j, where $1 \leq i, j \leq n$. In particular given a generic class: y_i, we define:

1) True Positive (TP): $y_{i,i}$
2) False Positive (FP): $\sum_{j \neq i} y_{j,i}$
3) False Negative (FN): $\sum_{j \neq i} y_{i,j}$

The evaluation of the performance of the methods considered here is based on the performance metrics used in the relevant bibliograph (e.g., [28], [31], [33], [36], [39]) which are: *Precision* computed as $\frac{TP}{TP+FP}$ and *Recall* computed as $\frac{TP}{TP+FN}$. Remark that precision and recall are computed for each class separately. So for the simplified arrhythmia detection, we have 2 pairs while for the standard arrhythmia detection we have 4 pairs. It is important to note that the MIT-BIH database only provides approximate detection point and beat type in its annotation files. Therefore, a classification provided by methods considered here needs to be temporally aligned with the detection points identified in the MIT-BIH database and the beat type needs to coincide with the annotation provided by the two doctors.

D. Hardware-assisted Detection

Each ECG $30min$ recording is split into 180 windows of $10sec$ each. The 65% of the resulting windows (i.e., 117 windows) is used for the *training set* and the remaining 35% windows (i.e., 63 windows) is used for the *validation test*. The training set is passed to the Pattern Matching Engine to be stored as neurons. Then, the validation set is submitted to the (trained) Pattern Matching Engine to classify each window into one of the predefined categories.

The experiments indicate that the extraction of some features is relatively lightweight, requiring on average between $5ms$ to $14ms$, while some other features (like the detection of the QRS complex) require on average between $80ms$ and $131ms$. Overall, the average time required to extract all the features for a single-window is $451ms$. The average time required to classify a single-window is $7ms$. The Intel® Quark™ SE 1000 Customer Reference Board is used to measure power consumption. The consumption of the x86 processor while performing the feature extraction was measured at $49.25mA$ while when the PME was classifying vectors, the consumption was measured at $53.87mA$.

After analyzing all the ECG recording, the average precision achieved is 88.86% and the average recall is 82.01%. For the specific case of arrhythmia detection, it is the number of TP that is most significant as they indicate the number of abnormal beats that have been properly labelled. In this experiment, the TP is above 80% in all cases. This is a very promising result since it is expected that with further fine-tuning, even higher success rates may be achieved.

Like any other machine learning technique, also here, the hardware-assisted detection depends on the features extracted. However, there is a trade-off between the number and type of features one should consider. It is critical to consider features that do not involve complicated computations and as a consequence will take a long processing time to be extracted. Such complex feature extraction will also increase the power consumption of the embedded system. Consider for example the analysis of the signal in the frequency domain. On the other hand side, features that are simple to extract may not help identify the specific morphologies of an ECG segment that will help the model to differentiate it from other samples. Such features are for example the identification of the R peaks, or the location of S part. Moreover, one must consider that due to the memory limitations of the embedded device there is an inherent limit to the total number of feature used. Finally, it is also important to assess and validate the truthfulness of the extraction due to noise inevitably affecting ECG recordings. Trying to increase the accuracy may require the amplification of the signal, or the execution of an extraction function multiple times, or allocating additional memory in order to increase the window sizes. This entails a very delicate feature engineering process in order to achieve a balance between the required memory resources (RAM), the computational capabilities of the MCU, the energy consumption, the execution of the classification process within specific latency constraints and the resulting precision and recall of the model.

E. Convolutional Neural Network approach

Like in the previous case, first each ECG recording is split into windows of 10sec and the signal is processed to identify the R-peaks and extract the $12sec$ fragments centred on each R-peak. In a way similar to the case of the hardware-assisted detection model, the resulting segments for each recording are split into two parts: training (65%) and validation (35%). The feature extraction of the CNN approach is very simple and requires less time to execute within the MCU, requiring an average of $95ms$. The Power Profiler Kit from Nordic Semiconductors is used to measure energy consumption during the preprocessing of the signal is measured at $11.51mA$.

The evaluation starts by measuring the performance of the simplified CNN model that classifies each ECG segment into two classes: normal or abnormal. The implementation of the model was based on the TensorFlow computational library [1] and then a version suitable for execution within the MCU was generated using the TensorFlow Lite converter. Finally, the post-training quantization conversion technique was used to reduce the model size while also reducing MCU utilisation, with little degradation in model accuracy. In particular, the *integer quantization* optimization strategy was selected that converted 32-bit floating-point numbers (such as weights and activation outputs) to the nearest 8-bit fixed-point numbers. This resulted in a smaller model and increased inferencing speed, which is valuable when executed within an MCU. The resulting model has a total of 3786 parameters, each of which is an unsigned int (32bit), therefore a total of 121152 bits are required to store it. The final output is 32 bit. The average precision achieved is 94% and the overall recall is 94.02%. The average time required to perform the simple preprocessing phase and execute the model for a single-window is $135ms$. The energy consumption for the execution of the classifier is measured at the same levels as the preprocessing phase.

The second part of the evaluation looks into the performance of the complete CNN model that classifies each segment into four classes: N, S, V, F. The same approach as the one reported above was followed to generate the optimized model for the MCU. The resulting model requires a total of 2046720 bits and the final output is 128 bit. In Tab. I the size of the model is depicted, along with the output size and the average execution time for each of the five passes of the model (see Sec. III-C). The first pass includes also the initial input and convolution operations, while the fifth pass includes the two FC, ReLU operations. The complete CNN model requires additional memory, about 20 times more than the simplified one. However, in terms of average execution time, the average time to execute the model for each segment is $246ms$. Once again, energy consumption for the execution of the classifier remains at the same levels.

Notice that each pass requires about the same number of bits to execute the 32 kernels of size 5 of each convolution operation and carry out the max-pooling of size 2 and stride 2 in all pooling layers. On the other hand, the output of each pass - which becomes the input for the next pass - reduces in size. This implies that the first pass involves convolution operations with more parameters than the following ones. As a result, the average execution time of each pass reduces significantly.

The average precision of the complete model for each of the 4 classes considered is depicted in Tab. II. The *TensorFlow* column refers to the results produced by executing the model on a desktop computer (i.e., not an MCU) using the "standard" TensorFlow computational library [1]. The results reported here are slightly lower than those reported in [17] due to

TABLE I: CNN Layers Profiling

Pass	Model size	Output size	Execution Time
1	474624 bit	5984 bit	136 ms
2	867328 bit	2944 bit	61 ms
3	1260288 bit	1408 bit	29 ms
4	1653248 bit	640 bit	14 ms
5	2046720 bit	256 bit	5 ms
Output		128 bit	1 ms

TABLE II: Achieved precision and recall of different models

	Tensorflow		TLite NQ-Model		TLite Q-Model	
	P	R	P	R	P	R
N	83.23	99.23	81.19	99.33	80.46	99.33
S	99.16	83.36	98.78	81.33	98.64	80.77
V	94.38	98.18	96.10	96.00	96.08	95.33
F	97.75	72.42	93.77	78.40	93.77	78.40

the data augmentation technique used. The column *TLite NQ-Model* refers to the conversion of the model using the TensorFlow Lite support library without carrying out any post-training quantization conversion technique. The third column *TLite Q-Model* refers to the application of the integer quantization optimization technique on the *TLite NQ-Model*. Based on the results depicted in Tab. II, the *TLite NQ-Model* has a slightly reduced accuracy than the original *TensorFlow*, while there is a further minor degradation of accuracy for the *TLite Q-Model*. Note that in all cases considered the precision for the N class is low (80 to 83 %) because the data is unbalanced towards N beats. This means the model is trained with a lot of beats classified as N, and this leads to a preference of the model to classify a beat as N in case of uncertainty, thus creating many false positive.

V. Conclusions

In this study, two different methods for ECG heartbeat classification within a microcontroller have been evaluated. The first approach builds upon a hardware-assisted KNN implementation that allows the classification of ECG recordings with a very short latency and achieving low power-efficiency. The second approach uses a deep convolutional neural network with residual connections for the arrhythmia classification task that is optimized to operate within a microcontroller. The results indicate that arrhythmia detection within low-power wearable devices can make predictions with reasonable high precision when compared to the state of the art methods in the literature of high-end processor architecture.

For the hardware-assisted approach, feature extraction that is carried out within the generic X86 MCU requires a significant amount of time and energy. On the other hand, the execution of the machine learning algorithm is very fast while the energy consumption is much lower. As discussed in the previous section, the number of features that need to be extracted needs to be carefully considered. On one hand side, the computational complexity and consequently the requirements in energy consumption become a limiting factor. On the other side, including many features that are simple to compute inevitably increases the memory requirements which also a limiting factor within the hardware-assisted machine learning algorithm. In fact, in this work an iterative process was followed for the selection of features, continuously refining the code and considering lightweight alternatives. Such a feature engineering and firmware fine-tuning process in some cases turned out to be time-consuming.

In contrast to the above, the deep-learning model considered in this work requires a very short preprocessing of the signal. On the other hand side, the type of kernels available within the embedded execution environment is not rich, thus limiting the types of models that can be supported. Therefore one needs to interact with the support library, in this work it was Tensorflow Lite, to design a model that can be transferred within the embedded environment. Moreover, due to the available memory (RAM) within the MCU, there are also certain limits to the size of the model. Another critical aspect is the difficulty to break down the execution of the model into smaller functions. Within the embedded system various handlers may require the preemption of the execution to avoid the disruption of certain functions. As an example consider the network connectivity based on BLE that requires very precise timing between packet transmissions thus requiring higher priority. Given that the execution of the model is considered a black box provided by the support library, i.e., Tensorflow Lite, this critically complicates firmware design. Such problems do not exist on the hardware-assisted prototype since the execution of the model is done in a very short period, while the extraction of the feature is implemented using multiple, short, functions that can be easily preempted to allow the processing of a higher task by the MCU.

The results of the experimental evaluation indicate that while the execution of the model to carry out a single classification has lower latency than the hardware-assisted detection model, the time needed to preprocess the recording and carry out the feature extraction is much shorter thus resulting in an overall shorted time of execution. As a result, the CNN-based approach leads to lower energy consumption.

Future work in this area is connected to examining alternative hardware-assisted modules and evaluate their performance in comparison to those presented here. It is also important to investigate potential improvements on the CNN model by examining alternative designs and layer compositions, in combination with additional features and different model optimization techniques. In terms of reducing the execution time of the deep-learning model, the possibility of using a smaller sampling rate needs to be considered and also the possibility of reducing the resolution of the ECG recordings.

Acknowledgment

This work has been supported by the European Union's research projects Secure and Seamless Edge-to-Cloud Analytics" (ELEGANT) and "Social Mining & Big Data Ecosystem" (SoBigData++) funded by the European Commission (EC) under the Horizon 2020 framework and contract number 957286 and 871042.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] Orestis Akrivopoulos, Dimitrios Amaxilatis, Athanasios Antoniou, and Ioannis Chatzigiannakis. Design and evaluation of a person-centric heart monitoring system over fog computing infrastructure. In *Proceedings of the First International Workshop on Human-centered Sensing, Networking, and Systems*, pages 25–30, 2017.

[3] Orestis Akrivopoulos, Dimitrios Amaxilatis, Irene Mavrommati, and Ioannis Chatzigiannakis. Utilising fog computing for developing a person-centric heart monitoring system. In *14th International Conference on Intelligent Environments, IE 2018, Roma, Italy, June 25-28, 2018*, pages 9–16. IEEE, 2018.

[4] Orestis Akrivopoulos, Dimitrios Amaxilatis, Irene Mavrommati, and Ioannis Chatzigiannakis. Utilising fog computing for developing a person-centric heart monitoring system. *Journal of Ambient Intelligence and Smart Environments*, 11(3):237–259, 2019.

[5] Orestis Akrivopoulos, Ioannis Chatzigiannakis, Christos Tselios, and Athanasios Antoniou. On the deployment of healthcare applications over fog computing infrastructure. In *Computer Software and Applications Conference (COMPSAC), 2017 IEEE 41st Annual*, volume 2, pages 288–293. IEEE, 2017.

[6] Fabio Angeletti, Ioannis Chatzigiannakis, and Andrea Vitaletti. Towards an architecture to guarantee both data privacy and utility in the first phases of digital clinical trials. *Sensors*, 18(12), 2018.

[7] Ioannis Chatzigiannakis, Emil Stoyanov Valchinov, Athanasios Antoniou, Athanasios Kalogeras, Christos Alexakos, and Panagiotis Konstantinopoulos. Advanced observation and telemetry heart system utilizing wearable ecg device and a cloud platform. In *2015 IEEE Symposium on Computers and Communication (ISCC)*, pages 25–30. IEEE, 2015.

[8] Mung Chiang and Tao Zhang. Fog and iot: An overview of research opportunities. *IEEE Internet of things journal*, 3(6):854–864, 2016.

[9] Eduardo José da S. Luz, William Robson Schwartz, Guillermo Cámara-Chávez, and David Menotti. Ecg-based heartbeat classification for arrhythmia detection: A survey. *Computer Methods and Programs in Biomedicine*, 127:144 – 164, 2016.

[10] Li Du, Yuan Du, Yilei Li, Junjie Su, Yen-Cheng Kuan, Chun-Chen Liu, and Mau-Chung Frank Chang. A reconfigurable streaming deep convolutional neural network accelerator for internet of things. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(1):198–208, 2017.

[11] Association for the Advancement of Medical Instrumentation. Testing and reporting performance results of cardiac rhythm and st segment measurement algorithms, 2013. American National Standard 2013, ANSI/AAMI EC57:2012.

[12] Linux Foundation. Zephyr Project. https://www.zephyrproject.org/, 2017.

[13] Vinayak Gokhale, Aliasger Zaidy, Andre Xian Ming Chang, and Eugenio Culurciello. Snowflake: An efficient hardware accelerator for convolutional neural networks. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2017.

[14] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[15] Haibo He, Yang Bai, Edwardo A Garcia, and Shutao Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*, pages 1322–1328. IEEE, 2008.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Mohammad Kachuee, Shayan Fazeli, and Majid Sarrafzadeh. ECG heartbeat classification: A deep transferable representation. *CoRR*, abs/1805.00794, 2018.

[18] Selcan Kaplan Berkaya, Alper Kursat Uysal, Efnan Sora Gunal, Semih Ergin, Serkan Gunal, and M. Bilginer Gulmezoglu. A survey on ecg analysis. *Biomedical Signal Processing and Control*, 43:216–235, 2018.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[20] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[21] Periyasamy M Mariappan, Dhanasekaran R Raghavan, Shady HE Abdel Aleem, and Ahmed F Zobaa. Effects of electromagnetic interference on the functional usage of medical equipment by 2g/3g/4g cellular phones: A review. *Journal of Advanced Research*, 7(5):727–738, 2016.

[22] Fen Miao, Yayu Cheng, Yi He, Qingyun He, and Ye Li. A wearable context-aware ecg monitoring system integrated with built-in kinematic sensors of the smartphone. *Sensors*, 15(5):11465–11484, 2015.

[23] George B Moody and Roger G Mark. The impact of the MIT-BIH arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001.

[24] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[25] Robert E O'Connor, Abdulaziz S Al Ali, William J Brady, Chris A Ghaemmaghami, Venu Menon, Michelle Welsford, and Michael Shuster. Part 9: acute coronary syndromes: 2015 american heart association guidelines update for cardiopulmonary resuscitation and emergency cardiovascular care. *Circulation*, 132(18_suppl_2):S483–S500, 2015.

[26] Jiapu Pan and Willis J Tompkins. A real-time qrs detection algorithm. *IEEE Trans. Biomed. Eng*, 32(3):230–236, 1985.

[27] Paweł Pławiak. Novel genetic ensembles of classifiers applied to myocardium dysfunction recognition based on ecg signals. *Swarm and evolutionary computation*, 39:192–208, 2018.

[28] Paweł Pławiak. Novel methodology of cardiac health recognition based on ecg signals and evolutionary-neural system. *Expert Systems with Applications*, 92:334–349, 2018.

[29] Piotr Ponikowski, Stefan D Anker, Khalid F AlHabib, Martin R Cowie, Thomas L Force, Shengshou Hu, Tiny Jaarsma, Henry Krum, Vishal Rastogi, Luis E Rohde, et al. Heart failure: preventing disease and death worldwide. *ESC heart failure*, 1(1):4–25, 2014.

[30] Haotong Qin, Ruihao Gong, Xianglong Liu, Xiao Bai, Jingkuan Song, and Nicu Sebe. Binary neural networks: A survey. *Pattern Recognition*, 105:107281, 2020.

[31] Pranav Rajpurkar, Awni Y. Hannun, Masoumeh Haghpanahi, Codie Bourn, and Andrew Y. Ng. Cardiologist-level arrhythmia detection with convolutional neural networks. *CoRR*, abs/1707.01836, 2017.

[32] Stephan R Richter and Stefan Roth. Matryoshka networks: Predicting 3d geometry via nested shape layers. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1936–1944, 2018.

[33] Indu Saini, Dilbag Singh, and Arun Khosla. Qrs detection using k-nearest neighbor algorithm (knn) and evaluation on standard ecg databases. *Journal of Advanced Research*, 4(4):331 – 344, 2013.

[34] Ryuichi Sakamoto, Thang Cao, Masaaki Kondo, Koji Inoue, Masatsugu Ueda, Tapasya Patki, Daniel Ellsworth, Barry Rountree, and Martin Schulz. Production hardware overprovisioning: Real-world performance optimization using an extensible power-aware resource management framework. In *2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 957–966. IEEE, 2017.

[35] SEGGER. RTOS & Embedded Software. https://www.segger.com/products/rtos-embedded-software/.

[36] Atman P. Shah and Stanley A. Rubin. Errors in the computerized electrocardiogram interpretation of cardiac rhythm. *Journal of Electrocardiology*, 40(5):385 – 390, 2007.

[37] Markos G Tsipouras, Dirnitrois I Fotiadis, and D Sideris. Arrhythmia classification using the rr-interval duration signal. In *Computers in Cardiology*, pages 485–488. IEEE, 2002.

[38] General Vision. Curieneurons library. https://www.general-vision.com/curieneurons/.

[39] Özal Yıldırım, Paweł Pławiak, Ru-San Tan, and U Rajendra Acharya. Arrhythmia detection using deep convolutional neural network with long duration ecg signals. *Computers in biology and medicine*, 102:411–420, 2018.