Runtime-Adaptive Cognitive IoT Nodes

Matteo Antonio Scrugli, Daniela Loi, Luigi Raffo, and Paolo Meloni

Department of Electrical and Electronic Engineering, University of Cagliari, Italy

Abstract. Current mainstream approach to sensor data monitoring usually relies on cloud access: samples are acquired by connected devices and data processing is performed on remote servers. To improve responsiveness, security and resilience, devices and programming methodologies must be improved, with the aim of enabling data analytics at the edge. Unfortunately this is not an easy task, especially in the IoT domain. In this paper, we present a research approach that manages at runtime the hardware/software configuration of a low-power processing system, with the aim of adapting to dynamically changing workloads optimizing power-relevant settings to the corresponding operating point. First, we present a first validation experiment, involving a hardware-software architecture for a connected sensor-processing node that allows the set of in-place processing tasks to be executed to be remotely controllable by an external user. The designed system is capable of dynamically adapting its operating point to the selected computational load, to minimize power consumption. The benefits of the proposed approach are tested on a use-case involving ECG monitoring, that, when selected, performs ECG classification using a lightweight convolutional neural network. Experimental results show how the proposed approach can provide more than 50% power consumption reduction for common ECG activity, with less than 2% memory footprint overhead and reconfiguring the system in less than 1 ms. Second we present our plans to extend this approach to more complex multi-core systems.

1 Introduction

Modern systems-of-systems (SoSs) are typically composed of multiple hierarchies of sensory nodes, that are in charge of collecting huge amounts of sensorial data from the physical environment. Near-sensor data analytics is often exploited, to convert acquired raw sensor data in more compact information, with the aim of reducing bandwidth requirements and communication-related power consumption, increasing responsiveness of the system and avoiding transmission of sensitive data for privacy reasons. A further improvement of this kind of approach is provided by recent advances in machine learning and cognitive computing techniques, such as those in the field of deep learning and neural networks, that provide very high performance in terms of accuracy when it comes to recognition and classification tasks applied to the sensed data.

To combine near-sensor processing with energy efficiency, data analytics tasks, that may be quite compute-intensive and power-hungry, must be activated only when needed. Nodes must adapt, even at runtime, to different operating modes corresponding to different usage cases, each one optimized for power efficiency.

In this paper we present a methodology enabling such kind of runtime dynamic management. A detailed description of this experiment is presented in [16]. Moreover, we present our future plans for extending the methodology to be used on more complex multi-core systems, using a multi-DSP chip as a target reference platform.

2 Related work

To validate our approach, we have chosen an application in the IoT health domain as a use case. The value of such market is projected to reach $136 billion by 2021 [14]. Network of sensors are expected to be deployed in hospitals and homes, many studies in the literature have presented IoT architectures for patient monitoring [20][15][9]. When implementing IoT-based data sensing architectures, wearability and portability of the sensing device is crucial, thus autonomy of the device and battery life are treated as main objectives to be considered within their design and implementation [5][18][19][1][2]. Several low-power devices available on the market are able to remain active for days [6], however some further optimization is still needed when near-sensor data processing has to be exploited to face power consumption, bandwidth, data privacy and security constraints.

Several research activities have started exploiting current advances in machine learning and artificial intelligence for detecting specific events/conditions in sensed data. Some interesting approaches can be found in [17][10]. Both projects involve the use of artificial neural networks (ANN). However, these works only use very simple ANN architectures and detect quite generic conditions. For example, in [10], by extracting the features of the ECG signal (PQRST peaks), good results are obtained regarding the classification of emotional states such as joy and sadness. When it comes to more detailed and reliable analysis, processing tasks may become much more complex and, thus, their execution must be carefully activated when needed, optimally implemented with adequate tools and libraries [12][7], on efficient target platform, very often custom-designed [11][13].

As main novel contribution in this work we propose:

- An implementation of a remotely-controlled sensory node allowing for near-sensor data processing inserted in an IoT context.
- Its validation on a state-of-the-art data analysis based on a Convolutional Neural Network as an example computational load.
- The evaluation of the effectiveness of in-place computing and operating mode dynamic optimization, as a method to reduce the power consumption of the node, on a case study involving classification of ECG data.

3 Node architecture

The overall system architecture is shown in Figure 1. In this paper, we focus on the architecture of the sensor node. The description of the cloud infrastructure and of the gateway is beyond the scope of the work presented here.

The multi-level structure of the sensor node architecture is schematized in Figure 2. In the following sections, a detailed description of each node level is provided.

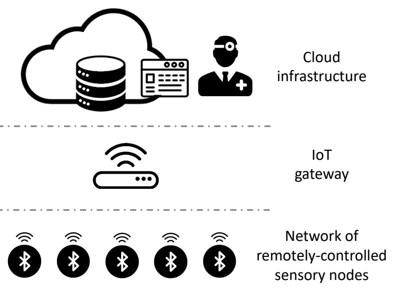


Fig. 1. General overview of the proposed system.

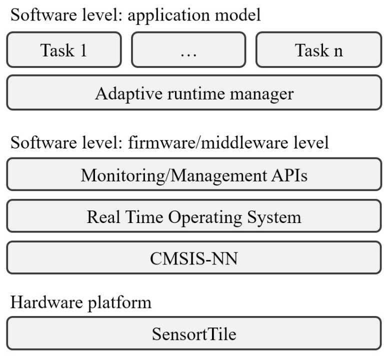

Software level: application model
Task 1
…
Task n
Adaptive runtime manager
Software level: firmware/middleware level
Monitoring/Management APIs
Real Time Operating System
CMSIS-NN
Hardware platform
SensorTile

Fig. 2. Structural composition of the single node.

3.1 Hardware platform

The SensorTile is a platform developed by STMicroelectronic, measuring $13.5 \times 13.5mm$, and equipped with a ARM Cortex-M4 32-bit low-power microcontroller ($80\,MHz$). Despite the small size, the device is endowed with a battery and with a Bluetooth Low Energy (BLE) module. The system exposes different operating modes, to be described in the next section.

Two main operating states are available, *run mode* and *sleep mode*, which can be further customized selecting the operating frequency (from 0.1 MHz to 80 MHz) and, depending on the selected frequency range, using a different voltage regulator.

Within our system, to dynamically reduce power consumption:

– we scale operating frequency and select accordingly the optimal voltage regulator, on the basis of the actual workload to be served,
– we exploit the sleep mode whenever possible. In sleep mode the microcontroller is in standby until it gets awakened by an interrupt. In our implementation the sleep mode is entered under the control of the operating system support, whenever all active software tasks are served.

3.2 Middleware/firmware layer

We have selected a set of middleware and firmware utilities, executable on top of the hardware platform, providing basic platform management capabilities and the needed optimized processing functions. Some of these libraries are part of the SensorTile Hardware Abstraction Layer, provided by the platform vendor. The others are specifically selected for the purpose of improving flexibility and implementing cognitiveness on the sensor node.

FreeRTOS In order to enable thread-level abstraction to represent processing tasks to be executed on the platform, and to timely manage their scheduling at runtime, the FreeRTOS operating system is included in the stack. It's small enough to operate on a microcontroller (generally, size is between $4\,kB$ and $9\,kB$). FreeRTOS primiteves implement real-time scheduling functionality, communication between processes, synchronization and time measurements.

In our architecture, as mentioned, FreeRTOS is able to schedule an *idle* task, that deactivates the system tick counter and sets the microcontroller in sleep mode. The idle task is entered whenever possible according to the scheduling queue, when no pending tasks have to be executed.

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), a optimized library specifically targeting Cortex-M processor cores [7]. It integrates several components of CMSIS, including the CMSIS-NN package, suitable to apply cognitive computing techniques on the sensor data, through efficient neural network kernels, optimized to maximize performance and minimize memory footprint.

Within CMSIS, we have added support for 1D convolution operators on input signals, with the aim of enabling cognitive sequence analysis.

3.3 Application model

We selected an application model based on process networks, where processes act as data-flow actors communicating through FIFOs. Similar models are widely used to improve analysis of concurrent tasks on embedded systems and for studying their dynamic reconfiguration [3]. For each sensed variable to be monitored, we build a chain of tasks that operate on the sensed data. Provided that this model is used, the architecture and the software stack can execute the tasks concurrently and manage dynamic switching between operating modes, activating/deactivating tasks and re-routing data-communication.

Tasks are envisioned to belong to four categories:

- *Get data task:* takes care to collect data from the sensors.
- *Process task:* enables in-place computing of the sensed data, e.g. to reduce bandwidth requirements and energy consumption. Multiple process tasks may be active to choose the depth of data analysis.

– *Threshold task:* operates on the results of the processing, selecting which items are relevant to be sent to the cloud.
– *Send task:* sends data to the cloud.

Activating/deactivating parts of the chain, the system can switch to different operating modes, featuring different trade-offs between in-place computing effort, bandwidth requirements and monitoring precision. Examples of possible alternative operating modes will be presented within the use-case discussion in Section 4.

ADAM Along with the chains of tasks processing the different sensor inputs, the platform executes a periodic task specifically dedicated to dynamic reconfiguration, the ADAptive runtime Manager (ADAM). This task periodically processes input messages received from the gateway and evaluates the status of the platform. On this basis, ADAM adapts the software and hardware configuration.

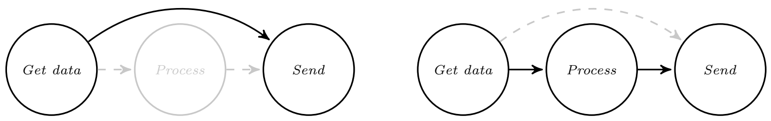


Fig. 3. Two possible configurations of a generic system.

Figure 3 shows an example of the reconfiguration of the system that may be applied by ADAM. ADAM can be triggered from the gateway, by sending a message to its input FIFO, or through a periodic timer, to implement a periodic status check. The system has the possibility to:

– Enable or disable entire sensor task chains.
– Set a run mode or sleep mode of the microcontroller.
– Set the operating frequency of the microcontroller (and consequently the appropriate voltage regulator).
– Adjust the message traffic between the FIFOs based on which tasks are active or deactivated.

In the second case, the system has the possibility, for example, to check the battery status and take reconfiguration decisions accordingly.

4 Use case evaluation

In order to assess the benefits of the proposed approach, we have selected a use case involving monitoring of a patient's ECG signal. In the prototype that we have implemented, an AD8232 sensor module from Analog Devices is connected to the ADC converter integrated in SensorTile, used to acquire samples to be coded using 16-bit width. Figure 4 shows components used in the prototype.

In this scenario, three possible operating modes have been identified (see Figure 5), described individually in the next sections.

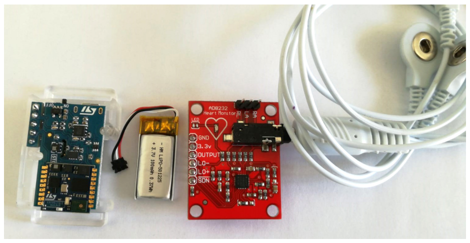


Fig. 4. Prototype hardware components.

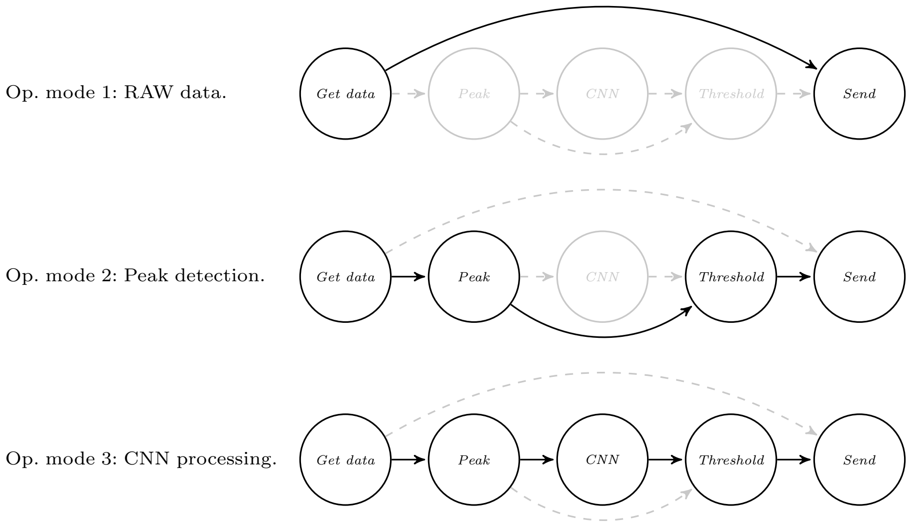


Fig. 5. EEG application model.

4.1 Operating mode 1: RAW data

This operating mode involves complete transmission of all the collected samples to the cloud. This requires maximum bandwidth but allows the healthcare practitioner to visually analyze the whole ECG waveform. Due to the high data transmission rate, for this operating mode the system must be configured to run at a minimum frequency of $8\,MHz$. To save energy and to obtain a more stable connection, 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 $330\,Hz$, so one Bluetooth packet is sent every $24\,ms$.

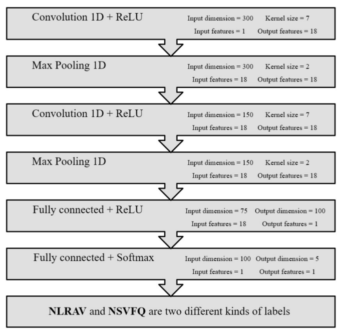
4.2 Operating mode 2: Peak detection

This operating mode reduces the detail of the information that is sent to the cloud. Instead of providing visual access to the whole ECG waveform, the practitioner may monitor only heartbeat rate, prospectively only getting a notification in case of alert conditions. This operating mode processes samples to search for signal peaks and consequently computes the heartbeat rate.

The algorithm for detecting a peak signal is based on the its derivative. It is not vey critical in terms of time and power consumption. The measurement carried out shows that the system is capable of respecting the real-time constraints exposed by the sampling rate at a frequency of $2\,MHz$. The data is sent to the cloud with a packet of 5 Bytes (1 heartbeat rate value, represented on 8 bit, 1 time-stamp 32 bit). In worst case, a packet is sent for every detected peak in the signal, thus the bandwidth requirement is dependent on the cardiac activity. Considering human tolerable heartbeat rates, the transmission rates are, however drastically reduced with respect to Operating Mode 1.

4.3 Operating mode 3: CNN processing

This operating mode is designed to combine bandwidth reduction with morphological analysis of the ECG waveform. This analysis is performed in-place by a processing task based on a convolutional neural network and communicates occurrence of specific patterns in the sample sequence. The implemented neural network can recognize occurrence of anomalous events in the ECG trace, so that communication to the cloud can take place only when a related alert has to be notified. Bandwidth requirements are reduced with respect to Operating Mode 1 and may be similar to Operating Mode 2 (worst case is one packet per peak, when all peaks are anomalous). However, with respect to Operating Mode 2, computing effort is higher. The node executes the 1D convolution neural network described in [8]. It consists of several convolution layers, down-sampling layers and one fully connected layer, as represented in Figure 6.


Convolution 1D + ReLU
Input dimension = 300 Kernel size = 7
Input features = 1 Output features = 18
Max Pooling 1D
Input dimension = 300 Kernel size = 2
Input features = 18 Output features = 18
Convolution 1D + ReLU
Input dimension = 150 Kernel size = 7
Input features = 18 Output features = 18
Max Pooling 1D
Input dimension = 150 Kernel size = 2
Input features = 18 Output features = 18
Fully connected + ReLU
Input dimension = 75 Output dimension = 100
Input features = 18 Output features = 1
Fully connected + Softmax
Input dimension = 100 Output dimension = 5
Input features = 1 Output features = 1
NLRAV and NSVFQ are two different kinds of labels

(N) Normal beats
(L) Left bundle branch block
(R) Right bundle branch block
(A) Atrial premature contraction
(V) Ventricular premature contraction

(N) Normal beats
(S) Superventricular ectopic beats
(V) Ventricular premature contraction
(F) Fusion beats
(Q) Unclassificable beat

Fig. 6. CNN structure.

The model can be used in three forms, using different size of the convolution kernel and the size of the output (number of features) at each layer. In [8] these three variants are examined:

- *Conv1D small:* Output size = 10, kernel size = 3.
- *Conv1D medium:* Output size = 18, kernel size = 7.
- *Conv1D large:* Output size = 50, kernel size = 13.

Input size is always set to 300 samples with stride set to 1. We chose *Conv1D medium* as reference network, a larger network would exceed memory availability in the device. The network classifies sliding windows of input samples, centered in each ECG peak, labeling peak shapes for two possible set of categories, named NLRAV and NSVFQ (see Figure 6).

The experimental results reported in [8] show that the proposed method achieves a promising classification accuracy, 97.5%, on the public MIT-BIH arrhythmia database, significantly outperforming several typical ECG classification methods. 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).

5 Experimental results

More details on the experimental results are reported in [16], Figure 7 shows the different levels of power consumption measurements in the three operating modes, for some specific heartbeat rate values (50, 100 and 200 bpm).

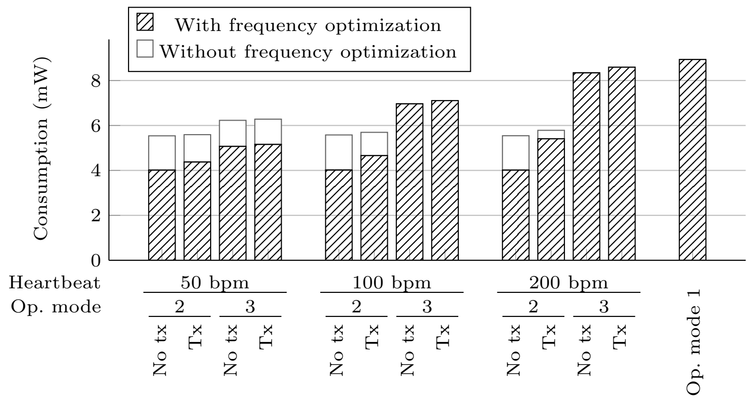


Fig. 7. The graph summarizes the energy consumption for different heartbeat rates, when sending of data is enabled for every beat (Tx) or when it is never enabled (No Tx) by the *threshold* task. Operating mode 1 does not depend on heartbeat nor on the threshold settings.

6 Future work

In order to validate our approach on state-of-the-art complex parallel processing platforms and on more complex CNN structures, we will extend the presented work to perform an experiment on a multi-core platform. This would require inevitaly several improvements of what was previously described. The reference platform taken into consideration was developed within the Orlando project by ST Microelectronics. We will briefly describe the hardware components inside the system used for the sake of this project in the following. Then, we will discuss two main required extensions needed to implement runtime adaptivity on this platform, namely:

– implementation of the adaptivity support on baremetal system, we will need to exploit the platform-specific set of APIs to manage the process chain and the related operating modes.
– adaptation of the parallelism level exploitable within the CNN structure, required to achieve an optimal partitioning of the workload on the available processing cores, using splitting/merging and pipeline methods.

6.1 Orlando

In order to deploy these technologies in mobile and wearable devices, an efficient hardware plays a critical role for real-time operation with very limited power consumption and with embedded memory overcoming the limitations of fully programmable solutions.

We have selected as reference example a high performance and energy efficient processor developed by ST microelectronics[4]. The chip is very flexible and represents a perfect test-case for our technique, since it integrates:

– An on-chip reconfigurable data-transfer fabric to improve data reuse and reduce on-chip and off-chip memory traffic.
– A power-efficient array of DSPs to support complete real-world computer vision applications.
– An ARM-based host subsystem with peripherals.
– A range of high-speed IO interfaces for imaging and other types of sensors.
– A chip-to-chip multilink to pair multiple devices together.

6.2 Baremetal system & application model

In general, we should assume (at least part of) the processing elements available in multi-core architectures to possibly not provide operating system support. Thus, when extending our work, we will need to manage the tasks in the process chain the thread-level abstractions granted by the RTOS. Moreover, we will need to consider tasks in the same chain to be prospectively executed by independent hardware cores communicating through a set/hierarchy of shared memories. The ADAM system will not use the APIs offered by the FreeRTOS middleware.

Instead, low-level primitives provided by STmicroeletronics will allow to manage the execution and change the operating mode at runtime. Similarly to what previously shown, when a core is idle, it is placed in a low-power state. The shared memories between the cores will be used to implement software FIFOs allowing a correct communication between the different cores.

6.3 CNNs structure

We will implement support for different kinds of parallelism, that will be exploited to optimize the partitioning of the workload on the available cores. A first level of parallelism that we will consider will be at layer level. We will take profit of software pipeline, assigning a core (if available) to each CNN layer. In case of significant pipeline bottlenecks, taking place when the execution time of a task is much longer than the others, we will allow the process network to be modified using split and merge transformations.

If used appropriately, splitting a layer can balance execution times of the pipeline states, allowing a more efficient utilization of the computing resources. In the example, the split transformation is performed on a Convolutional Layer block and divides it into two independent blocks. The input data is copied for every independent block. The output data stream coming from the two independent blocks is combined and synchronized by a concatenation node. The merge transformation is in the opposite direction. It composes two or more DNN graph nodes/blocks into one block and and is useful when two or more stages of the pipeline can be properly merged. We will evaluate combining design-time exploration and runtime management to find the optimal partitioning for the selected use-cases.

7 Conclusion

In this work we have presented our approach to the design of sensor monitoring nodes capable of adapting at runtime to different workloads. We have presented a first proof of concept, based on a hardware/software architecture for the design of dynamically reconfigurable IoT, that may adapt to different operating modes when triggered by an external command or when suggested by the monitoring of its internal status. The proposed approach has been deployed on a commercial microcontroller platform and evaluated on a use-case involving ECG monitoring and classification based on state-of-the art convolutional neural network. The use-case features three different operating modes, selectable by means of remotely-sent reconfiguration messages. The results obtained show how near-sensor processing combined with dynamic optimization can be effectively exploited, reducing power consumption up to a factor of 2 with respect to static simple monitoring, with limited overhead in terms of memory footprint and reconfiguration time. We have also introduced our plan for the extension of the adaptivity support to multi-core parallel computing platforms.

References

1. Adimulam, M.K., Srinivas, M.B.: Ultra low power programmable wireless exg soc design for iot healthcare system. In: Perego, P., Rahmani, A.M., TaheriNejad, N. (eds.) Wireless Mobile Communication and Healthcare. pp. 41–49. Springer International Publishing, Cham (2018)
2. Carta, N., Meloni, P., Tuveri, G., Pani, D., Raffo, L.: A custom mpsoc architecture with integrated power management for real-time neural signal decoding. IEEE Journal on Emerging and Selected Topics in Circuits and Systems **4**(2), 230–241 (2014). https://doi.org/10.1109/JETCAS.2014.2315881
3. Derin, O., Cannella, E., Tuveri, G., Meloni, P., Stefanov, T., Fiorin, L., Raffo, L., Sami, M.: A system-level approach to adaptivity and fault-tolerance in noc-based mpsocs: The madness project. Microprocessors and Microsystems **37**(6-7), 515–529 (2013). https://doi.org/10.1016/j.micpro.2013.07.007
4. Desoli, G., Chawla, N., Boesch, T., Singh, S., Guidetti, E., De Ambroggi, F., Majo, T., Zambotti, P., Ayodhyawasi, M., Singh, H., Aggarwal, N.: 14.1 a 2.9tops/w deep convolutional neural network soc in fd-soi 28nm for intelligent embedded systems. In: 2017 IEEE International Solid-State Circuits Conference (ISSCC). pp. 238–239 (Feb 2017). https://doi.org/10.1109/ISSCC.2017.7870349
5. Ghasemzadeh, H., Jafari, R.: Ultra low-power signal processing in wearable monitoring systems: A tiered screening architecture with optimal bit resolution. ACM Trans. Embed. Comput. Syst. **13**(1), 9:1–9:23 (Sep 2013). https://doi.org/10.1145/2501626.2501636, http://doi.acm.org/10.1145/2501626.2501636
6. Kaewkannate, K., Kim, S.: The Comparison of Wearable Fitness Devices (10 2018). https://doi.org/10.5772/intechopen.76967
7. Lai, L., Suda, N., Chandra, V.: CMSIS-NN: efficient neural network kernels for arm cortex-m cpus. CoRR **abs/1801.06601** (2018), http://arxiv.org/abs/1801.06601
8. Li, D., Zhang, J., Zhang, Q., Wei, X.: Classification of ecg signals based on 1d convolution neural network. In: 2017 IEEE 19th International Conference on e-Health Networking, Applications and Services (Healthcom). pp. 1–6 (Oct 2017). https://doi.org/10.1109/HealthCom.2017.8210784
9. Macis, S., Loi, D., Pani, D., Raffo, L., Manna, S.L., Cestone, V., Guerri, D.: Home telemonitoring of vital signs through a tv-based application for elderly patients. In: 2015 IEEE International Symposium on Medical Measurements and Applications (MeMeA) Proceedings. pp. 169–174 (May 2015). https://doi.org/10.1109/MeMeA.2015.7145193
10. Magno, M., Pritz, M., Mayer, P., Benini, L.: Deepemote: Towards multi-layer neural networks in a low power wearable multi-sensors bracelet. In: 2017 7th IEEE International Workshop on Advances in Sensors and Interfaces (IWASI). pp. 32–37 (June 2017). https://doi.org/10.1109/IWASI.2017.7974208
11. Meloni, P., Capotondi, A., Deriu, G., Brian, M., Conti, F., Rossi, D., Raffo, L., Benini, L.: Neuraghe: Exploiting CPU-FPGA synergies for efficient and flexible CNN inference acceleration on zynq socs. ACM Transactions on Reconfigurable Technology and Systems **11**(3), 1–24 (2018). https://doi.org/10.1145/3284357
12. Meloni, P., Loi, D., Deriu, G., Pimentel, A.D., Sapra, D., Moser, B., Shepeleva, N., Conti, F., Benini, L., Ripolles, O., Solans, D., Pintor, M., Biggio, B., Stefanov, T., Minakova, S., Fragoulis, N., Theodorakopoulos, I., Masin, M., Palumbo, F.: Aloha: An architectural-aware framework for deep

learning at the edge. In: Proceedings of the Workshop on INTelligent Embedded Systems Architectures and Applications. pp. 19–26. INTESA '18, ACM, New York, NY, USA (2018). https://doi.org/10.1145/3285017.3285019, http://doi.acm.org/10.1145/3285017.3285019

13. Pani, D., Meloni, P., Tuveri, G., Palumbo, F., Massobrio, P., Raffo, L.: An fpga platform for real-time simulation of spiking neuronal networks. Frontiers in Neuroscience **11** (2017). https://doi.org/10.3389/fnins.2017.00090
14. Research, A.M.: Internet of things (iot) healthcare market - global opportunity analysis and industry forecast, 2014 - 2020 (2016), https://www.alliedmarketresearch.com/iot-healthcare-market
15. Roberts, L., Michalák, P., Heaps, S., Trenell, M., Wilkinson, D., Watson, P.: Automating the placement of time series models for iot healthcare applications. In: 2018 IEEE 14th International Conference on e-Science (e-Science). pp. 290–291 (Oct 2018). https://doi.org/10.1109/eScience.2018.00056
16. Scrugli, M.A., Loi, D., Raffo, L., Meloni, P.: A runtime-adaptive cognitive iot node for healthcare monitoring. pp. 350–357 (04 2019). https://doi.org/10.1145/3310273.3323160
17. Tabal, K.M.R., Caluyo, F.S., Ibarra, J.B.G.: Microcontroller-implemented artificial neural network for electrooculography-based wearable drowsiness detection system. In: Sulaiman, H.A., Othman, M.A., Othman, M.F.I., Rahim, Y.A., Pee, N.C. (eds.) Advanced Computer and Communication Engineering Technology. pp. 461–472. Springer International Publishing, Cham (2016)
18. Tekeste, T., Saleh, H., Mohammad, B., Ismail, M.: Ultra-low power qrs detection and ecg compression architecture for iot healthcare devices. IEEE Transactions on Circuits and Systems I: Regular Papers **66**(2), 669–679 (Feb 2019). https://doi.org/10.1109/TCSI.2018.2867746
19. Wang, C., Qin, Y., Jin, H., Kim, I., Granados Vergara, J.D., Dong, C., Jiang, Y., Zhou, Q., Li, J., He, Z., Zou, Z., Zheng, L.R., Wu, X., Wang, Y.: A low power cardiovascular healthcare system with cross-layer optimization from sensing patch to cloud platform. IEEE Transactions on Biomedical Circuits and Systems pp. 1–1 (2019). https://doi.org/10.1109/TBCAS.2019.2892334
20. Yang, Z., Zhou, Q., Lei, L., Zheng, K., Xiang, W.: An iot-cloud based wearable ecg monitoring system for smart healthcare. Journal of Medical Systems **40**(12), 286 (Oct 2016). https://doi.org/10.1007/s10916-016-0644-9, https://doi.org/10.1007/s10916-016-0644-9