Embedding Temporal Convolutional Networks for Energy-efficient PPG-based Heart Rate Monitoring

Embedding Temporal Convolutional Networks for Energy-Efficient PPG-Based Heart Rate Monitoring

ALESSIO BURRELLO, Department of Electrical, Electronic and Information Engineering, University of Bologna, Italy
DANIELE JAHIER PAGLIARI, Department of Control and Computer Engineering at Politecnico di Torino, Italy
PIERANGELO MARIA RAPA, Department of Electrical, Electronic and Information Engineering, University of Bologna, Italy
MATILDE SEMILIA, Department of Electrical, Electronic and Information Engineering, University of Bologna, Italy
MATTEO RISSO, Department of Control and Computer Engineering at Politecnico di Torino, Italy
TOMMASO POLONELLI, PBL Center, Switzerland
MASSIMO PONCINO, Department of Control and Computer Engineering at Politecnico di Torino, Italy
LUCA BENINI, Department of Information Technology and Electrical Engineering at the ETH Zurich, Switzerland
SIMONE BENATTI, Department of Electrical, Electronic and Information Engineering, University of Bologna, Italy
Department of Sciences and Methods for Engineering, University of Modena e Reggio Emilia, Italy

Photoplethysmography (PPG) sensors allow for non-invasive and comfortable heart-rate (HR) monitoring, suitable for compact wrist-worn devices. Unfortunately, Motion Artifacts (MAs) severely impact the monitoring accuracy, causing high variability in the skin-to-sensor interface. Several data fusion techniques have been introduced to cope with this problem, based on combining PPG signals with inertial sensor data. Until know, both commercial and reasearch solutions are computationally efficient but not very robust, or strongly dependent on hand-tuned parameters, which leads to poor generalization performance.

Authors' addresses: Alessio Burrello, Department of Electrical, Electronic and Information Engineering, University of Bologna, Bologna, Emilia-Romagna, Italy, 40136, name.surname@unibo.it; Daniele Jahier Pagliari, Department of Control and Computer Engineering at Politecnico di Torino, Torino, Italy, 10129, name.surname@polito.it; Pierangelo Maria Rapa, Department of Electrical, Electronic and Information Engineering, University of Bologna, Bologna, Emilia-Romagna, Italy, 40136, name.surname@studio.unibo.it; Matilde Semilia, Department of Electrical, Electronic and Information Engineering, University of Bologna, Bologna, Emilia-Romagna, Italy, 40136, name.surname@studio.unibo.it; Matteo Risso, Department of Control and Computer Engineering at Politecnico di Torino, Torino, Italy, 10129, name.surname@studenti.polito.it; Tommaso Polonelli, PBL Center, Zurich, Switzerland, 8092, tommaso.polonelli@pbl.ee.ethz.ch; Massimo Poncino, Department of Control and Computer Engineering at Politecnico di Torino, Torino, Italy, 10129, name.surname@polito.it; Luca Benini, Department of Information Technology and Electrical Engineering at the ETH Zurich, Zurich, Switzerland, 8092, lbenini@iis.ee.ethz.ch; Simone Benatti, Department of Electrical, Electronic and Information Engineering, University of Bologna, Bologna, Emilia-Romagna, Italy, 40136,
Department of Sciences and Methods for Engineering, University of Modena e Reggio Emilia, Modena, Emilia-Romagna, Italy, 42123, name.surname@unibo.it.

2637-8051/2021/9-ART1 $15.00
https://doi.org/10.1145/3487910

In this work, we tackle these limitations by proposing a computationally lightweight yet robust deep learning-based approach for PPG-based HR estimation. Specifically, we derive a diverse set of Temporal Convolutional Networks (TCN) for HR estimation, leveraging Neural Architecture Search (NAS). Moreover, we also introduce ActPPG, an adaptive algorithm that selects among multiple HR estimators depending on the amount of MAs, to improve energy efficiency. We validate our approaches on two benchmark datasets, achieving as low as 3.84 Beats per Minute (BPM) of Mean Absolute Error (MAE) on PPGDalia, which outperforms the previous state-of-the-art. Moreover, we deploy our models on a low-power commercial microcontroller (STM32L4), obtaining a rich set of Pareto optimal solutions in the complexity vs. accuracy space.

Additional Key Words and Phrases: Temporal Convolutional Networks, Heart Rate Monitoring, Medical IoT, Wearable Devices, Deep Learning

ACM Reference Format:
Alessio Burrello, Daniele Jahier Pagliari, Pierangelo Maria Rapa, Matilde Semilia, Matteo Risso, Tommaso Polonelli, Massimo Poncino, Luca Benini, and Simone Benatti. 2021. Embedding Temporal Convolutional Networks for Energy-Efficient PPG-Based Heart Rate Monitoring. *ACM Trans. Comput. Healthcare* 1, 1, Article 1 (September 2021), 26 pages. https://doi.org/10.1145/3487910

1 INTRODUCTION

Wrist-worn devices are gaining remarkable traction in the wearable ecosystem [33], benefiting from the availability of accurate, compact sensors and energy-efficient microcontrollers [3]. While first-generation platforms relied mostly on accelerometers to perform activity recognition tasks, the novel paradigms of personalized healthcare and medical Internet of Things (IoT) [48] push in the direction of having wearables with richer sensor sets to monitor vital parameters such as electrodermal activity (EDA), and heart rate (HR).

In particular, monitoring HR and HR variability is paramount for clinical purposes and precise activity monitoring. The first wrist-worn HR tracking devices were connected to a chest band with a simple 1-3 leads Electrocardiogram (ECG) sensor. Albeit accurate, this solution is uncomfortable for users and even impossible to wear in certain conditions. More recently, the optimization and miniaturization of photoplethysmogram (PPG) sensors allowed integrating HR and blood oxygenation (SpO2) in wrist-worn devices, resulting in a more comfortable and user-friendly solution at a lower cost compared to conventional ECG strips. PPG sensors and HR estimation are now integrated on commercial wrist-worn devices such as the Apple Watch [4] or the Fitbit Charge 4 [20].

A major challenge in PPG-based HR estimation [13] is represented by motion artifacts (MA), which cause variability of sensor pressure on the skin or ambient light leaking into the gap between the photodiode and the skin. Besides, blood flow can vary considerably as the type of physical activity varies, contributing to a worsening of the light absorption measurement, which is necessary to estimate HR. Several studies compared ECG and PPG based approaches, evaluating the HR on 50 [21] and 30 [27] healthy subjects in three scenarios (i.e., sitting, walking, and jogging). These studies have shown that ECG chest straps outperform PPG-based platforms, affected by a mean error of up to 10 beats per minute (BPM). As a result, the ECG-based solutions are still considered the reference benchmark for wearable HR tracking [1].

To overcome these accuracy limitations, recent research has focused on obtaining precise HR measurements by integrating PPG and accelerometer data to mitigate MAs' effect. These studies have explored novel algorithmic approaches and collected vast datasets to validate them [36, 54]. On the algorithmic side, most solutions are based on classical approaches such as Independent Component Analysis (ICA), Kalman Filters, and Wavelet decomposition. These methods reduce the noise caused by MAs by creating models for the PPG signal and for the noise. TROIKA [54] and its evolutions, JOSS [53] and more recently CurToSS [55], are seminal works in this category. They estimate the noise via adaptive filtering and then apply a spectral peak tracking to detect the heartbeat frequency achieving a Mean Absolute Error (MAE) lower than 2 beats per minute. A major shortcoming

of the aforementioned approaches is related to the extensive hand-tuning of the model parameters, leading to lack of generalization.

Recently, deep learning approaches are receiving increasing attention also in this field, reaching accuracies comparable to those of classical methods with less hand-tuning of parameters. However, the deployment of these computationally-expensive models on resource-constrained platforms (such as those embedded in wrist-worn wearables) is still an open challenge since coupling a high accuracy with a small model is not trivial. In addition to that, data-driven algorithms necessitate a high amount of training data to reach satisfactory results, which are not trivial to collect. In 2019, a novel PPG-based HR tracking dataset called PPGDalia was presented [36], which covers many daily activities for 15 healthy subjects, paving the way to a more in-depth exploration of deep-learning solutions for this challenge.

In this paper, which extends the preliminary work of [38], we introduce a rich set of accurate yet computationally efficient Temporal Convolutional Networks (TCNs) for PPG-based HR estimation. Moreover, we integrate these TCNs in a novel framework, which adaptively combines multiple HR tracking models at runtime depending on an estimate of the MAs to reduce the overall energy consumption. In detail, the main contributions of this work are the following:

- *TimePPG*, a collection of Pareto-optimal TCNs for HR estimation based on raw PPG and acceleration data. All the TCNs are automatically derived from a single seed architecture [50] using a Neural Architecture Search (NAS) algorithm [22] to explore the accuracy vs. model size and the accuracy vs. complexity trade-offs.
- *ActPPG*, a new framework to adaptively combine TimePPG solutions, as well as other algorithms, depending on the user's movement conditions, using a bigger but more accurate model when higher MAs are expected. To this end, ActPPG employs a low-cost human activity recognition (HAR) model and uses the predicted activity as a discriminator to choose between different HR tracking models.
- A detailed comparison of our approach with state-of-the-art methods, including both model-driven approaches and deep-learning ones, on two popular benchmark datasets, i.e., PPGDalia [36] and SPC2015 [54].
- A complete deployment of both individual TimePPG models and of the ActPPG framework on a low-power microcontroller (MCU), demonstrating our algorithms' suitability for the edge execution in terms of latency and energy consumption.

The best performing TimePPG model, *TimePPG-Big*, coupled with simple smoothing post-processing, achieves a Mean Absolute Error (MAE) of 4.88 BPM on PPGDalia [36] (the largest public PPG dataset) and includes ≈ 232k trainable parameters. With an additional fine-tuning step, the MAE is further reduced to 3.84 BPM. At the other extreme, the smallest model in TimePPG, *TimePPG-Small*, uses only 5k parameters while still reaching an acceptable MAE of 5.63 BPM. When deployed on the STM32L4R9AII6, a popular low-power MCU by ST Microelectronics, *TimePPG-Small* and *TimePPG-Big* consume 232 μJ, and 17.57 mJ per inference, with a latency of 17.1 ms, and 1289.5 ms, respectively. *TimePPG-Small* outperforms previous deep learning solutions [36] on the PPG-Dalia dataset by 5.2-12000× and 3.8-4800× in terms of model size and the number of required operations respectively, while also improving the HR estimation accuracy (5.63 vs. 9.99/7.65 BPM of MAE). Thanks to the ActPPG adaptive approach, we obtain further Pareto points, reducing inference's computational complexity with a small accuracy degradation. For instance, using a small random forest as movement detector and *TimePPG-Small/Big* as HR estimators, we reduce the computation complexity by 50.3% while losing only 0.39 BPM of MAE (for a benchmark in which 50% of the samples have a "high" degree of MAs). Deploying such configuration on the STM32L4R9AII6 MCU, we obtain as low as 8.90 mJ of energy per inference.

The rest of the paper is organized as follows. Section 2 provides the necessary background on PPG, on TCNs, and on our target hardware platform. Section 3 gives an overview of state-of-the-art HR estimation algorithms. Section 4 and Section 5 describe our main contributions, i.e., TimePPG and ActPPG respectively. Finally, Section 6 presents the results and their discussion and Section 7 concludes the paper.

2 BACKGROUND

2.1 Photoplethysmography

Photoplethysmography (PPG) is a technique based on measuring the light absorption variations of blood vessels during the cardiac activity [45]. A PPG sensor consists of one or more Light-Emitting Diodes (LEDs) that continuously emit light to the skin and a photodetector (i.e., a photodiode) that measures variations of light intensity caused by blood flow, whose periodicity depends on the heart rate. More specifically, the larger the blood volume variation, the greater the attenuation of the light emitted by the LED, resulting in a lower the current output on the photodiode. Therefore, a heartbeat can be associated with each peak in the PPG signal. Indeed, many studies demonstrated that the second derivative of the PPG signal contains essential information for heart rate monitoring [54].

The simplicity of wearing a PPG sensor and the low-cost contribute to its increasing popularity as an alternative to ECG for HR monitoring [44]. One of the major challenges in employing PPG signals is their significant dependency on the subject's movements, which negatively affect the measurement quality during daily activities, as first shown in [52]. In particular, Motion Artifacts (MA) caused by the hands' movement alter the readings of the sensor and strongly impact the performance of the HR estimation, demanding ad-hoc algorithms for their remotion/reduction. As described in Section 3, the standard approach is to leverage additional intertial measurements (mostly, acceleration) to clean the PPG signal from MAs. For more details on using PPG for HR estimation please refer to [9].

2.2 Temporal Convolutional Network

In recent years, TCNs have achieved outstanding performance in many different time-series processing benchmarks, often resulting superior to Recurrent Neural Networks, which were previously considered the de-facto standard deep learning models for such tasks [6, 24, 50]. TCNs are a sub-class of 1D-Convolutional Neural Networks (CNNs) esplicitly designed for the processing of time-series, whose peculiarity is in the use of *causality* and *dilation* in convolutional layers [6, 28]. *Causality* constrains the convolution output $\mathbf{y}_t$ to depend only on inputs $\mathbf{x}_{\tilde{t}}$ with $\tilde{t} \leq t$, i.e., outputs are determined looking only to the "past". *Dilation* is a fixed gap d inserted between input samples processed by the filters that compose a convolutional layer. This has the benefit of increasing the receptive field of the convolution on the time axis, without requiring a larger number of parameters. In summary, a convolutional layer in a TCN implements the following function:

$$\mathbf{y}_t^m = \sum_{i=0}^{K-1} \sum_{l=0}^{C_{in}-1} \mathbf{x}_{t-d\,i}^l \cdot \mathbf{W}_i^{l,m} \tag{1}$$

where $\mathbf{x}$ and $\mathbf{y}$ are the input and output feature maps, t and m the output time-step and channel respectively, $\mathbf{W}$ the filter weights, C_{in} the number of input channels, d the dilation factor, and K the filter size. Fig. 1 shows a high-level scheme of the execution of a layer with $K = 3$, and $d = 4$. In the original paper [6], TCNs were proposed as fully-convolutional architectures, but modern embodiments also include other common layers such as pooling and linear ones [37, 50], which have been used in our architecture exploration as well.

2.3 Hardware platform

Our work's main objective is to assess the feasibility of embedding deep learning-based HR estimation using PPG sensors and accelerometers in a wrist-worn wearable device. Therefore, rather than merely focusing on improving the HR estimation accuracy, our goal is to derive models that can be executed with low latency and energy consumption on a typical wearable platform, while retaining high accuracy. Nowadays, most commercial wearables' digital "brain" consists of an ultra-low-power System-on-Chip, typically based on an ARM Cortex-M-class MCU. Accordingly, we deploy all our models on the STM32L4R9I-EVAL evaluation board from STMicroelectronics,

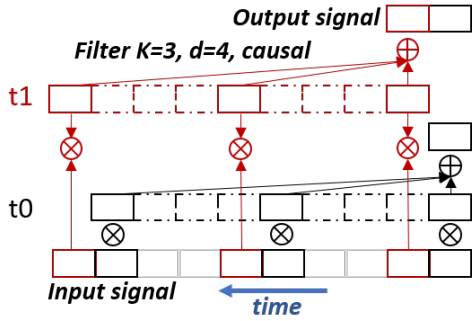

Fig. 1. Generation of the first 2 output time samples in a TCN layer with $K = 3$ and $d = 4$.

which features a Cortex-M4 core (the STM32L4R9AII6) with 640 kB of RAM, and 2MB of Flash. The board has an average power consumption of 13.63 mW at 80 MHz [31] [1].

The selection of this platform is also driven by the availability of *CUBE.AI*, a software toolchain provided by STMicroelectronics to deploy neural networks on their MCUs. *CUBE.AI* allows the automatic conversion of pre-trained neural networks from high-level frameworks such as TensorFlow Lite and Keras into optimized C code for the execution of the network on the target. Notably, at the time of submission of this article, *CUBE.AI* only allows the deployment of full-precision (*float32*) dilated convolutions, while *int8*-quantized convolutions with dilation are not supported.

3 RELATED WORK

Heart rate monitoring through wrist-worn PPG sensors is a relatively new application, attracting significant interest in industry and academia. The main challenges that this approach has to tackle are related to *i)* the accuracy, often measured as the $MAE = \mathbb{E}(|BPM_{true} - BPM_{prediction}|)$ and *ii)* the tight power and energy constraints that have to be met in order to execute locally on battery-operated wrist-worn devices; in fact, these algorithms are usually executed on a smartwatch, with a MCU operating at frequencies in the range of 10s of MHz and having a power envelope lower than 100 mW.

The first algorithms for PPG-based HR estimation were based on simple peak tracking methods. Among them, a representative example is found in [42], where the authors propose an improved peak detection algorithm called Adaptive Threshold (AT), which removes false peaks using an adaptive refractory period.

More recent algorithms can be split into two main categories: classical model-driven approaches, which are mostly based on adaptive-filtering combined with peak tracking, and data-driven ones, based on machine/deep learning. The leading solutions proposed in the literature are summarized in Table 1.

In the model-driven category, the seminal work of [54] paved the way to the algorithmic exploration in this field, proposing a new three-stage algorithm based on signal decomposition, spectrum estimation, and spectral peak tracking called TROIKA. TROIKA has been tested on the public dataset called SPC Cup 2015 (SPC2015), released together with the paper and comprising 12 subjects, achieving a MAE of 2.34 BPM. The same authors have also proposed an improvement of this algorithm in [53], where a spectral difference with the acceleration spectrum is used to clean the PPG spectrum from MAs, reducing the MAE on the same dataset to 1.28 BPM. Following the same MA cleaning approach, in [30], the authors propose to suppress them using a Singular Value Decomposition (SVD) applied on the acceleration data to extrapolate periodic artifacts. Coupled with an Iterative

[1]Notice that we use a platform with the same core but a larger memory space compared to the one adopted in our preliminary work of [38], to enable the deployment of all TimePPG and ActPPG models.

Table 1. State-of-the-art comparison table. Different MAE results correspond to the datasets in the Dataset column.

Work	Dataset	Activities	Sign.	Pre-Processing	Algorithm	Post-Proc.	MAE
Classical methods							
TROIKA, 2014 [54]	SPC2015*	Rest, Running	PPG, Acc.	0.5-4 Hz filtering, Downsampling	Signal decomp., reconstruct., spectral peak track	th., hist. track.	2.34 BPM
JOSS, 2015 [53]	SPC2015*	Rest, Running	PPG, Acc.	0.5-4 Hz filtering, Downsampling	MMV, spectral subtract	th., hist. track.	1.28 BPM
SpaMa, 2016 [40]	SPC2015* SPC2015[1] Chon Lab[2] PPG-Dalia[3]	Rest, Running, Rehab. ex., Rest, Running 8 daily activities	PPG, Acc.	0.5-3 Hz filtering, Downsampling	spectral filtering based on PSD	hist. track., interpol.	0.89 BPM 3.36 BPM 1.38 BPM 11.06 BPM
Schack2017 [41]	SPC2015* PPG-Dalia[3]	Rest, Running, 8 daily activities	PPG, Acc.	0.5-6 Hz filtering, Downsampling	Corr.-based Freq. indicating func., FFT	th.	1.32 BPM 20.5 BPM
FSM, 2018 [16]	SPC2015[1]	Rest, Running, Rehab. ex.	PPG, Acc.	0.5-4 Hz filtering, z-score scaling, Downsampling	Winer filtering	FSM	0.99 BPM
TAPIR, 2020 [23]	SPC2015* SPC2015[1] PPG-Dalia[3]	Rest, Running, Rehab. ex., 8 daily activities	PPG, Acc.	0.5-4 Hz filtering	Adaptive filter Peak detection Linear Transform.	Notch filter	2.5 BPM 5.9 BPM 4.6 BPM
Arunkmar, 2020 [5]	SPC2015* SPC2015[1]	Rest, Running, Rehab. ex.	PPG, Acc.	0.4-3.5 Hz filtering	RLS, NLMS MA reduction FFT based HR track.	Phase Voc.	1.03 BPM 1.89 BPM
CurToSS, 2020 [55]	SPC2015* SPC2015[1] PPG-Dalia[3]	Rest, Running, Rehab. ex., 8 daily activities	PPG, Acc.	0.5-4 Hz filtering	SSR Curve tracking	N/A	2.2 BPM 4.5 BPM 5.0 BPM
Deep Learning							
CNN, 2019 [36]	SPC2015* PPG-Dalia[3]	Rest, Running, 8 daily activities	PPG, Acc.	STFT, 0-4 Hz filtering	CNN	N/A	4 BPM 7.65 BPM
PPGNet, 2019 [43]	SPC2015* SPC2015[1]	Rest, Running, Rehab. ex.	PPG	0.4-18 Hz filtering, z-score scaling	Inception+LSTM	N/A	3.36 BPM 12.48 BPM
CorNet, 2019 [8]	SPC2015* SPC2015[1]	Rest, Running, Rehab. ex.	PPG	0.4-18 Hz filtering, z-score scaling	CNN+LSTM	N/A	4.67 BPM 5.55 BPM
BinCorNet, 2020 [39]	SPC2015* SPC2015[1]	Rest, Running, Rehab. ex.	PPG	0.4-18 Hz filtering, z-score scaling	Bin. CNN+LSTM	N/A	6.78 BPM 7.32 BPM
Chung, 2020 [15]	SPC2015*	Rest, Running.	PPG, Acc.	0.4-18 Hz filtering, z-score scaling	CNN+LSTM	N/A	1.46 BPM
DeepHeart, 2021 [14]	SPC2015*	Rest, Running.	PPG, Acc.	0.4-5 Hz filtering	DnCNN + spectrum analysis	err. check, calibration.	1.61 BPM
Our Work	PPG-Dalia[3]	8 daily activities	PPG, Acc.	0.5-4 Hz filtering	TCNBest	th, finetuning	3.84 BPM

* 12 subjects [1] 23 subjects [2] 10 subjects [3] 15 subjects

Method with Adaptive Thresholding (IMAT), this solution reduces the MAE on SPC2015 to 1.25 BPM. Similar MA reductions are also used in [5, 41] to clean the PPG signal, and a FFT to track the HR, achieving 1.32 and 1.03 BPM of MAE on SPC2015, whereas the authors of [16, 47] use Wiener filtering, further improving the performance on SPC2015 to 0.99 BPM of MAE. Lastly, the best performing approach on this dataset is SpaMa, proposed in [40] and based on a complex and computationally intensive five-step pipeline. This algorithm reduces the MAE to just 0.89 BPM.

More recently, Reiss et al. [36] released a new PPG-based HR tracking dataset, significantly larger than SPC2015. The dataset, called PPG-Dalia, includes 15 subjects monitored while performing eight different daily activities. In [23, 55] the authors proposed two new model-driven HR estimation solutions, tested both on SPC2015 and PPG-Dalia. The first, CurToSS [55], exploits sparse signal reconstruction and tracking of the curve in both acceleration and PPG signals, obtaining 2.2 BPM of MAE on SPC2015. The second, TAPIR [23], performs linear transformations in the time domain, strongly reducing the computational complexity and achieving a MAE of 2.5 BPM on the same dataset.

In general, all model-driven algorithms include many free parameters, leading to a high risk of overfitting. For instance, when two of the best-performing algorithms on the SPC2015 dataset are tested on the more extended PPG-Dalia, they obtain dramatically worse MAEs, from 0.89 to 11.06 BPM for [40], and from 1.32 to 20.5 BPM for [41]. Similarly, the only two model-driven algorithms directly developed for PPG-Dalia [23, 55] obtain the best performance on that dataset, 5.0 and 4.6 BPM, respectively, but achieve lower accuracy than the state-of-the-art for SPC2015. Moreover, adaptive filters, which are the core of nearly all the algorithms analyzed so far, are computationally intensive and not well suited for a real-time embedded application. Indeed, to the best of our knowledge, none of the aforementioned model-driven algorithms has been ported to a MCU-class platform such as those found on wrist-worn devices in order to evaluate its execution latency and energy consumption.

In the last years, motivated by the increasing success of machine/deep learning approaches in several other bio-signal processing applications, such as gesture recognition [50], seizure detection [10, 11] and brain-computer interfaces [51], researchers have started to explore solutions based on Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) for HR tracking. In [36], in conjunction with the publication of PPG-Dalia, the authors presented a CNN architecture coupled with a short-time Fourier transform, which outperforms two of the best classical methods [40, 41] on the new dataset. Furthermore, the works of [8, 14, 15, 43] achieve comparable results to the classical methods, using either pure deep learning architectures (e.g., CNN+LSTM) or denoising CNNs (DnCNNs) to remove motion artifacts followed by spectrum analysis applied to the clean PPG signals.

Deep learning solutions are known to achieve better generalization than classical ones on many tasks, but their application in this domain presents several challenges. Indeed, these models typically have many learned parameters (i.e., large memory footprints) and require a large number of operations for inference. Therefore, their deployment on memory-constrained MCUs, with low energy consumption and respecting real-time latency constraints, is not trivial. A first attempt to deploy a deep neural network for HR tracking has been made in BinaryCorNET [39], where the authors binarized the network of [8] and deployed it on dedicated hardware implemented in both ASIC and FPGA technologies, achieving just 56.1 μJ of energy consumption per classification, with a slightly higher MAE of 6.78 BPM on the SPC2015 dataset. To the best of our knowledge, we are the first to investigate the embedding of deep learning models for PPG-based HR tracking onto programmable, general purpose edge MCUs rather than custom hardware. Furthermore, we are also the first to use TCNs (described in Section 2.2) for this task. We explicitly focus on reducing the complexity and energy consumption of these models, exploring efficient architectures as well as an adaptive HR tracking solution that combines multiple models to improve efficiency (the *ActPPG* framework), while maintaining a low error.

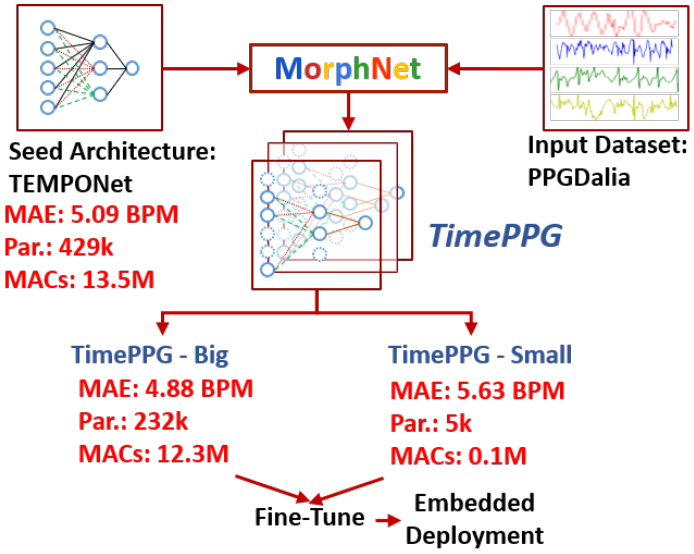

Fig. 2. Proposed NAS and deployment flow. The original TCN (seed architecture) and the input dataset are fed to the MorphNet NAS tool [22] to generate a set of optimized models called TimePPG. In red, the architectural parameters and MAE of the seed architecture and of two TimePPG Pareto points.

4 TIMEPPG: TEMPORAL CONVOLUTIONAL NETWORKS FOR HR ESTIMATION

This section describes the exploration of different Temporal Convolutional Networks (TCN) architectures for HR tracking in the accuracy vs. complexity space, aimed at finding accurate and light-weight solutions to be deployed on the hardware described in Section 2.3. We restrict the search space to architectures that can be executed on the target MCU, the STM32L4, while respecting a real-time constraint of 2s per inference. This constraint is in accordance with previous work [23, 36], and is equal to the time-shift (slide) between two consecutive input samples in the PPG-Dalia dataset (see Section 4.2).

Fig. 2 depicts the complete flow that we use to generate a set of TCNs. As shown, we explore the design space through a Neural Architecture Search (NAS) tool called MorphNet [22], which we enhanced to work on 1D dilated convolutions instead of its original target, i.e., the 2D layers found in standard CNNs for computer vision. MorphNet receives as input the training dataset, together with an original TCN, called "seed", which serves as the starting point for the architectural exploration. The NAS then optimizes the seed in different ways, generating a set of optimized models, offering different trade-offs in terms of HR tracking error versus complexity, which we call TimePPG. The MAE obtained by the seed on the PPG-Dalia dataset is reported in Fig. 2, together with its complexity in terms of the number of Parameters (Par.) and MAC operations. The same data are also reported for two extreme models in the TimePPG set, i.e., the most accurate network (TimePPG- Big) and the smallest one (TimePPG-Small), to show the positive effect of the NAS application. These results are analyzed in details in Section 6. TimePPG models, optionally fine-tuned on the patient under test, can then be deployed onto a wearable device, choosing the TCN version that best matches the constraints of the target hardware. We describe our design space exploration process in Section 4.1, while Section 4.2 describes the seed network and Section 4.3 discussed two additional steps to improve the accuracy of our models.

4.1 Design Space Exploration

NAS tools automatically generate novel NN architectures for a given task by acting on hyper-parameters such as the depth and the width of the network, the type of layers included, the connections between layers, etc [12, 29, 46]. While NAS is receiving increasing attention, standard tools mostly target complex computer vision tasks, leading to large and computationally-intensive NNs and requiring an enormous number of training iterations. Therefore, in this work, we decided to apply MorphNet [22] a tool that belongs to a category of recently developed light-weight NAS approaches called *DmaskingNAS*, whose complexity is comparable to that of a *single* NN training, at the price of a smaller search space. To obtain this result, DmaskingNAS tools generate their outputs as modified versions of a single *seed network*.

In particular, MorphNet limits the optimization to the *number of channels* (i.e., feature maps) in each layer, keeping the rest of the topology identical to the one of the seed. The number of channels is tuned to reduce the total memory footprint and number of required Multiply-and-Accumulate (MAC) operations of the network while retaining as much as possible the accuracy of the seed. This is obtained thanks to a two-step algorithm, which we extended to work with TCN layers. In the first *pruning* step, the network size is reduced by applying group Lasso [49], a training regularizer that forces the weights relative to entire channels (rather than single weights) to small magnitudes. After this regularization, channels whose total magnitude is inferior to a tunable threshold are eliminated. The second step is an *expansion*, in which the number of channels of all layers is up-scaled by a constant factor, called width multiplier. The goal is to recover part of the obvious performance penalty resulting from the first step. Noteworthy, as mentioned above, the complexity of MorphNet training is only slightly higher than that required to train the seed architecture. For more details on this tool, we refer the reader to the original paper [22].

Besides the seed network, MorphNet requires two other input parameters, i.e., the *pruning strength*, p_t, and the *pruning threshold* th_p, which control the strength for the group Lasso regularization and the number of eliminated channels in the pruning step. In order to explore the design space, we swept these two parameters between 10^{-6} and 10^{-3} (p_t) and 10^{-2} to 10^{-1} (th_p), further filtering all outputs that resulted in a computation time > 2 seconds. The results of this exploration are the different TCNs that compose TimePPG.

4.2 Seed Network

As a seed network for our exploration, we used TEMPONet [50] a TCN which shows impressive results on another bio-signal processing task, i.e., EMG-based gesture recognition. The TEMPONet architecture can be split into a modular *feature extractor*, composed of 3 convolutional blocks, and a *classifier*, composed of three linear layers. In turn, each convolutional block is formed by two dilated convolutions to increase the receptive field, a strided convolution, and an average pooling layer to shrink the time dimension. The output channels increase in each successive block to 32, 64, and 128, respectively. All layers use ReLU activations and batch normalization [25]. Further details on our seed network can be found in [50].

In order to adapt TEMPONet to our task, we changed the first layer to match the dimensions of each sample in the PPG-Dalia dataset. In particular, the network takes as input raw sensor data generated by the PPG-sensor and by a tri-axial accelerometer. The data provided to the algorithm are sampled at 32 Hz and fed to the model in sliding windows of 8 s, with a slide of 2s. Therefore, each input sample has four channels, ($Ch_{in} = 4$), and 256 time-steps ($t \in [0 : 255]$). We also replaced the final classification layer of TEMPONet with a single neuron in order to adapt it to a scalar regression task and we changed the loss function used for training from Categorical Cross Entropy to LogCosh. We found that, as in [34], LogCosh outperforms both RMSE and MAE losses, favoring the convergence near the minimum, thanks to its smoother behavior in that point.

4.3 Performance improvements

4.3.1 Post-Processing. Orthogonally to our design space exploration, we applied a low-complexity smoothing post-processing to the outputs of all models in TimePPG, to further improve their accuracy. This post-processing is motivated by the fact that, despite being more accurate on average than model-driven approaches, data-driven ones such as deep learning algorithms may sometimes incur huge errors, especially when the processed inputs differ from those seen in the training phase. Fortunately, in this particular task, these errors can be easily filtered by considering the compatibility of TCN estimations with human physiology, and in particular with the dynamics of the heart rate. Specifically, we impose a limit on the maximum variation of the estimated HR over time. To this end, we compare the latest TCN prediction with the mean of the previous N predictions, $E_{HR}[N]$: if the difference is larger than a threshold P_{th}, the HR is *clipped* to $E_{HR}[N] \pm P_{th}$. In this work, we set N to 10 and $P_{th} = {}^{E_{HR}[N]}/_{10}$, identical for all patients. When deployed on the target MCU, we found that this post-processing has negligible time and space complexity even with respect to the smallest HR estimation models.

4.3.2 Fine-Tuning. In one of our experiments of Section 6 we have also analyzed the performance of our models after an additional fine-tuning step. This is motivated by the fact that the lack of large datasets for PPG-based HR monitoring (public ones are limited to < 20 subjects) can hinder the performance of deep learning solutions, which notoriously benefit from large amounts of training data. In particular, input data never seen during training are challenging to predict. Therefore, subjects with *very large* HR values, that are rare in the rest of the dataset, are poorly tracked by our TCNs and, more generally, by data-driven algorithms. Very low HRs also have the same problem, but since they are typically associated to static activities with fewer MAs, the tracking problem is inherently easier for them. To underline this effect, Fig. 3 shows the accurate tracking obtained using the (not fine-tuned) TimePPG-Big outputs on a subject whose HR is in the interval $[50, 140]$ BPM (S7), and the much less accurate results obtained on S5, when the HR $>$ is 140 BPM, i.e., in the right tail of PPG-Dalia data distribution. We claim that this phenomenon could disappear in the presence of a more extended dataset and is not an intrinsic limitation of TimePPG. To demonstrate it, we *simulate* the effect of a larger dataset, which would include abundant samples of the entire range of possible HR values that our models are expected to predict, by means of fine-tuning. Specifically, we fine-tuned our networks on the first portion (25%) of each subject's data before testing on the remaining portion, obtaining a substantial performance improvement on patients whose HR is weakly represented in the training set. Note that, while this step is hardly reproducible in the field since fine-tuning would require collecting ground truth HR data from the user, e.g., through an ECG, we apply it only to mimic the results that could be obtained by training our models on a larger dataset, including data similar to all test subjects.

5 ACTPPG: REST-MOVEMENT BASED HR ESTIMATION

In this section we detail the *ActPPG* framework, specifically designed to adaptively combine multiple HR tracking models in a single prediction solution, using movement to assess the difficulty of the HR estimation. In Section 5.1 we provide the motivation and the intuition behind our framework, showing an example of its functionality by combining the AT model of [42] and the smallest TimePPG model, hereafter called TimePPG-Small. Then, we detail the different components of ActPPG in Section 5.2.

5.1 Motivation

Our framework aims to reduce the energy consumption of on-chip HR monitoring by combining the predictions of two different classifiers at runtime, one more accurate but more energy-hungry, and the other less accurate but more energy-efficient.

Inspired by big-little neural networks [18, 26, 32, 35], we aim at employing the heavier classifier only for those inputs for which the lighter one would fail. Achieving this goal would allow maintaining the same performance of

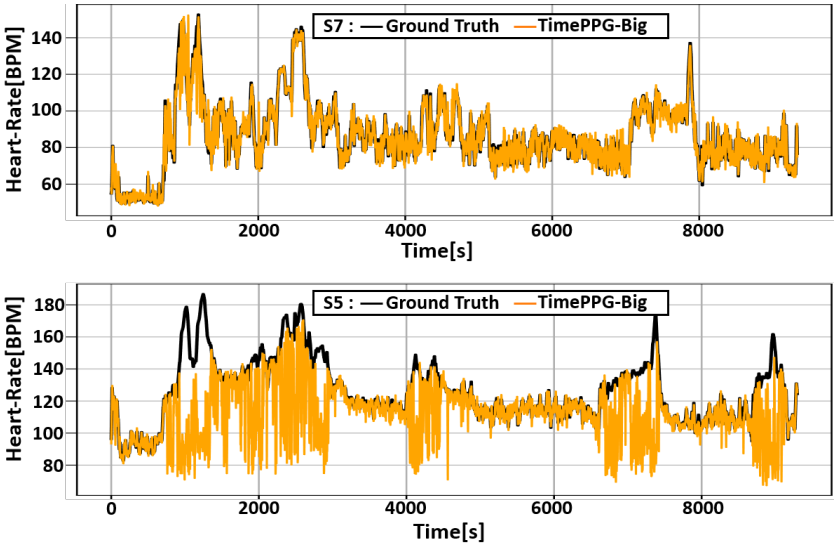

Fig. 3. Ground truth and prediction of a well tracked patient (S7) and of the worst one (S5). Values above 140 BPM are not well estimated by our algorithm.

the larger and more accurate classifier while decreasing the average energy consumption per input, depending on the frequency of "difficult" samples. Standard big-little models try to achieve this goal in an application-agnostic way, by sequentially executing first the little model and then optionally the big one, when the former's output has a low *confidence score* (see [18, 26, 32, 35] for details). Therefore, for "difficult" inputs, these approaches require executing *two models* and incur an energy overhead. In contrast, we propose an application-specific adaptive system that always requires a *single inference* on PPG data per input. Specifically, we use the amount of movement as a proxy of the HR prediction difficulty and use it to discriminate between easy and difficult samples. Our framework relies on two hypotheses:

1) MAE $\propto$ Mov. Regardless of the model used, as the amount of movement of a subject increases, the HR prediction becomes increasingly less accurate because of MAs.

2) Modelling Difficulty. The performance difference between a simple and small HR predictor and a larger and more complex one is mostly due to the latter's ability to obtain more accurate predictions for inputs affected by high movement. In other words, taken two models $M1$ and $M2$, with $\text{MAE}_{M1} > \text{MAE}_{M2}$, we observe that often $\text{MAE}^{rest}_{M1} \sim \text{MAE}^{rest}_{M2}$ for rest windows while $\text{MAE}^{mov}_{M1} >> \text{MAE}^{mov}_{M2}$ for high movement windows.

To demonstrate these intuitions, we propose an example comparing two algorithms of different complexity and accuracy: the Adaptive-Threshold (AT) described in [42] and the TimePPG-Small network, i.e., the most energy-efficient solution in TimePPG collection. First, we grouped samples relative to each of the 8 daily activities found in the PPG-Dalia dataset and sorted them by increasing movement, as shown in Figure 4. As an example, the sorting has been done using as a measure of the "amount of movement" the increasing average standard deviation (std) on the 3 axes of the accelerometer signal in each 8s window of the dataset. Then, we tested the two algorithms on each activity separately. Figure 5 reports the median (orange horizontal segments) and first/third quantiles of the algorithms' performance over all the different activities.

Coherently with our first hypothesis, the performance of both algorithms deteriorates for activities with higher movements, going from a minimum of 1.69 [0.70, 4.35] / 1.45 [0.60, 3.45] BPM (AT/TimePPG-Small medians with

[first quantile, third quantile]) for activity 1, i.e., *sitting*, to 17.88 [8.27, 31.31] / 11.0 [3.61, 25.60] BPM for activity 2, *climbing the stairs*, which has the second-largest acceleration std.

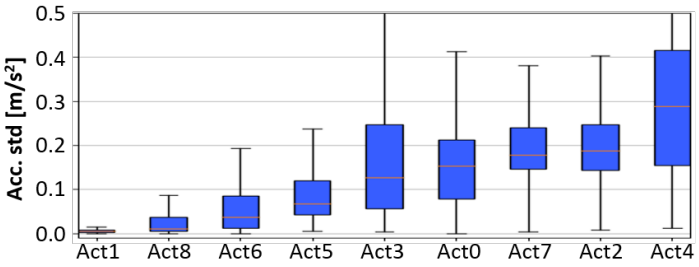

Fig. 4. PPG-Dalia activities ordered by increasing standard deviation of the accelerometer signal.

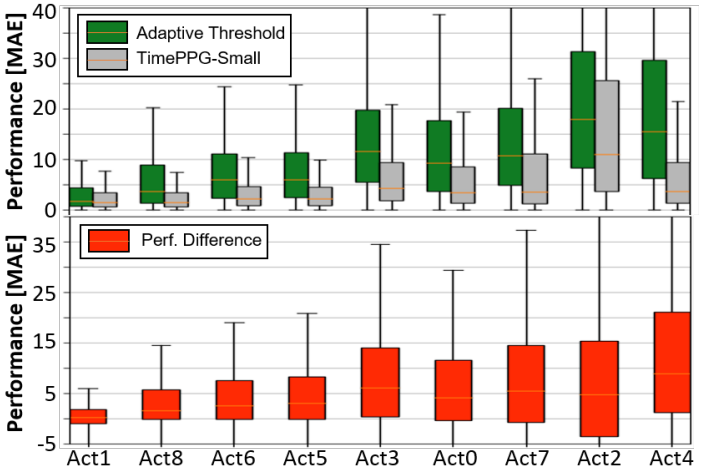

Fig. 5. Performance of two HR tracking algorithms (Adaptive Threshold, TimePPG-Small) on each PPG-Dalia activity (top). Performance difference between the same two algorithms (bottom).

Moreover, in accordance with our second hypothesis, the performance difference between the two algorithms also increases for high movement activities, as shown in the bottom of Figure 5. Specifically, it goes from a minimum difference of 0.23 [-0.95, 1.81] BPM for sitting to a maximum of 8.87 [1.26, 21.0] BPM for *cycling*, the activity with the highest acceleration std. Note that these assumptions also hold for different algorithms, e.g., for different TimePPG models, as demonstrated in the following sections.

5.2 ActPPG Modules

The ActPPG framework is composed of two main modules, as shown in Figure 6. The framework is fed with two signals, the PPG and the 3-axial acceleration. PPG data are the main inputs employed to predict HR in wrist-worn devices and are thus employed by all algorithms. In this work, we process a new window every 2s, and we use 8s

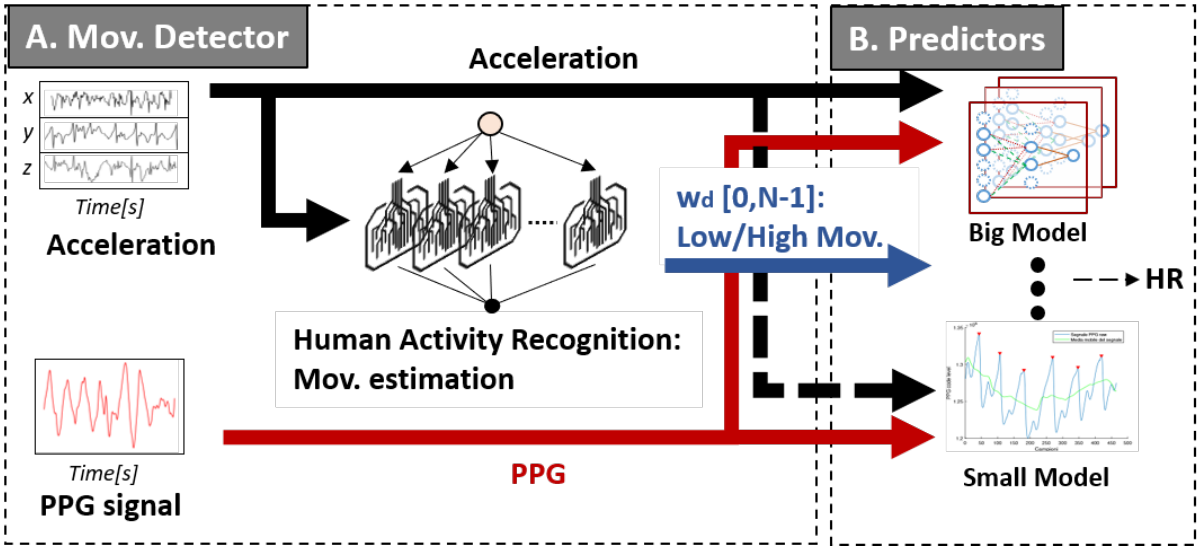

Fig. 6. Flowchart of the ActPPG framework. On the left, a human activity recognition (HAR) model (a random forest in our implementation) estimates the amount of MAs in the current input based on acceleration data. On the right, one of the N available heart rate tracking models is invoked based on the HAR result, with "bigger" models selected only for high-movement windows. In turn, the HR tracking model may use PPG data only or both PPG and acceleration.

as window length, following the setup of [36]. On the other hand, 3D acceleration does not contain essential features for HR but is used by many state-of-the-art algorithms to "clean" the PPG signal from the MAs caused by the subjects' hand movement, as detailed in Section 3. In our framework, we use acceleration data for two different tasks. First, we process them to determine whether a window is considered a "rest", i.e., a window with low MAs, or a "movement", with a high probability of MAs corruption. Second, the acceleration signal could also be (optionally) employed by the HR predictors directly to supplement the PPG signal for MA rejection [36, 54].

The core component of ActPPG is the *movement detector* module (Figure 6-A), whose goal is to assess the difficulty of the HR estimation for a given window. In particular, based on the assumptions of Section 5.1, we design a detector that uses the acceleration as input and gives a binary 0/1 output to distinguish between low and high movement windows. Note that this approach can be easily generalized to predict an ordinal output on a scale [0, N-1], where N-1 represents windows with the highest movement. One of N ordered HR predictors (from the smallest and less accurate to the largest and most accurate) can then be selected based on the window difficulty. In this work, we limit our exploration to the case of N=2. As detector, we use a lightweight random forest (RF) composed of 8 trees with a maxium depth of 5. We feed the RF with mean value, standard deviation, energy, and number of peaks of the acceleration signal. Combining these features with a RF allows to improve the final HR tracking performance, with respect to distinguishing between low and high movement with a simple threshold on a single feature (e.g., the acceleration std), and has a negligible impact on the overall inference complexity.

The *predictors* module (Figure 6-B) performs the actual HR prediction. This block receives as input the *window difficulty*, w_d, (0 to N-1 ordinal variable, with N=2 in our work), the PPG data, and optionally the 3D acceleration. The w_d variable is used as a switch to select the predictor to use for a given window. The predictors are sorted by performance offline, and more performant algorithms are associated with higher values of w_d. Noteworthy, all predictors are selected on the Pareto frontier in the accuracy vs. complexity plane. Therefore, when ordering by performance, we automatically associate more lightweight models to lower values of w_d.

6 EXPERIMENTAL RESULTS

In this section, we first describe in detail our target datasets. Then, we assess the performance of the individual HR estimation models from the TimePPG family and of the movement detector. Finally, we show how the complete ActPPG framework can be used to effectively trade-off the accuracy of HR tracking with the inference complexity. We then assess the energy consumption, memory footprint, and latency of our solutions when deployed on the platform described in Section 2.3. Finally, we compare the individual TimePPG models with recently developed techniques from the state-of-the-art. Since some of these were not tested on PPG-Dalia, for this comparison we also report results on the SPC2015 dataset. Software experiments are performed using Python 3.6 and TensorFlow 1.14 [2].

6.1 Target Datasets

We evaluate our models mainly on the *PPG dataset for motion compensation and heart rate estimation in Daily-Life Activities* (PPG-Dalia) [36]. At the time of writing, PPG-Dalia is, to the best of our knowledge, the largest publicly available dataset for PPG-based HR estimation. It includes PPG data, 3D acceleration, reference ECG, HR and activity labels, computed on a moving window of 8 seconds with 6 seconds overlap. A total of 37.5 hours have been monitored from 15 subjects, eight female and seven male, with age in the interval 21-55. Two commercial devices are used to collect data: the RespiBAN [7] for the reference ECG, and the wrist-worn Empatica E4 [19] for PPG and acceleration data. For further details on PPG-Dalia, refer to [36]. In Section 6.6, we also evaluate some of our TimePPG models on the smaller SPC Cup 2015 (SPC2015) dataset [54]. For both datasets, we follow the leave-one-subject-out validation schemes proposed in [36] and [54] respectively. Accordingly, all results reported in the rest of this section refer to the test subjects which, except for the case of fine-tuning on PPG-Dalia (see Section 6.6) are completely unseen during training. The state-of-the-art results have been taken directly from the original papers when available.

6.2 TimePPG Results

Figure 7 depicts some of the Pareto-optimal models extracted from the TimePPG family (black triangles) in the MAE vs. model size and the MAE vs. complexity (i.e., number of arithmetic operations – OPs) spaces on the PPG-Dalia dataset. These results refer to the algorithms' output after the smoothing post-processing step described in Section 4.3, without fine-tuning.

As shown, our automatic design space exploration technique based on NAS spans more than one order of magnitude, both in terms of TCN parameters (5k-232k) and OPs (0.1M-12.3M), despite starting from a single seed TCN. The red triangles in the figure underline two relevant Pareto models from TimePPG, i.e., the most accurate and the smallest. TimePPG-Big achieves the lowest overall MAE (4.88 BPM) while requiring around 232k parameters and 12.3M OPs. TimePPG-Small is the smallest model found in our design space exploration and is obtained from MorphNet using a regularizer strength of 1e-5 and a pruning threshold of 0.01. This TCN has just 5.09k parameters (46× compression with respect to TimePPG-Big) and requires less than 100k OPs per inference to achieve a MAE of 5.63 BPM (0.75 increase).

Finally, the figure also reports the results of the peak tracking algorithm of [42] (AT) as a red circle. Reporting this result in the figure is relevant, as AT will be later used as the "small" model in some ActPPG framework configurations. Although this algorithm achieves a much higher overall MAE (11.0 BPM, shown not in scale in the graph), it has the advantage of not using the acceleration input signal, with a complexity of less than 3k OPs.

The first rows of Table 2 report the accuracy results per subject obtained with AT, TimePPG-Small, and TimePPG-Big, whereas Table 3 reports the mean MAE across subjects and the number of operations per inference (OPs) of each model. Furthermore, in Fig. 8, we report the error distribution of the three algorithms. Noteworthy, despite the fact that the MAE of the three algorithms is comprised between 4.88 BPM and 11.0 BPM, most of the

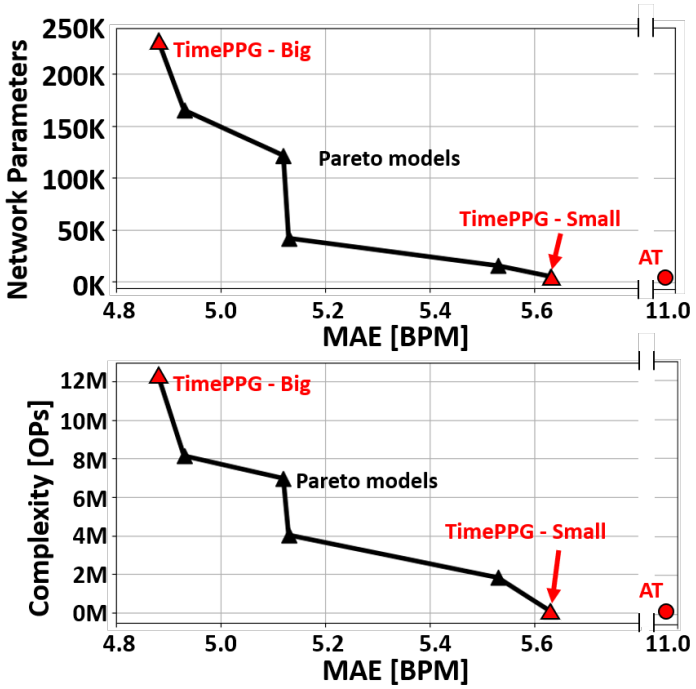

Fig. 7. TimePPG and AT Pareto models on the PPG-Dalia dataset in the MAE vs. Parameters space and in the MAE vs. complexity (number of OPs) space.

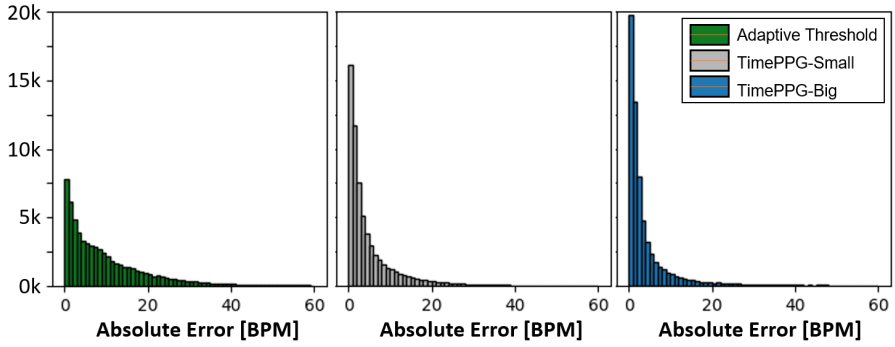

Fig. 8. Distribution of the absolute errors associated with the three different base algorithms. For clarity of visualization, we cut the distribution at a max error of 60 BPM.

errors are concentrated between 0 and 2 BPM. For instance, for TimePPG-Big, 30.5% of the windows show an error < 1 BPM and 51.3% less than 2 BPM.

6.3 Movement Detector Results

As anticipated in Section 5 our movement classifier is a random forest (RF) composed of 8 trees with a maximum depth of 5. We use the mean value, energy, standard deviation, and the number of peaks of the 3D acceleration as input features, after a grid search on a more extended set of classical fatures. As for HR estimation, we split the data using the cross-validation scheme proposed in [36] and report results as the average of all folds. Figure 9 shows the confusion matrix of our random forest on the 8 activities contained in PPG-Dalia, where the activities have been sorted by increasing movement in the same way as for Figure 4. Considering eight separated classes, thus only excluding the "transition" phase of PPGDalia data, we achieve a micro-averaging accuracy $^{TP+TN}/_{TS}$, of 60.6%, (where TP = true positives, TN = true negatives, and TS = total samples). Noteworthy, most of the erroneous predictions mistake one class with one of its two nearest neighbors in terms of movement. Therefore, these errors do not cause problems for ActPPG, where the goal is only to distinguish "low movement" and "high movement" windows, as shown in the following.

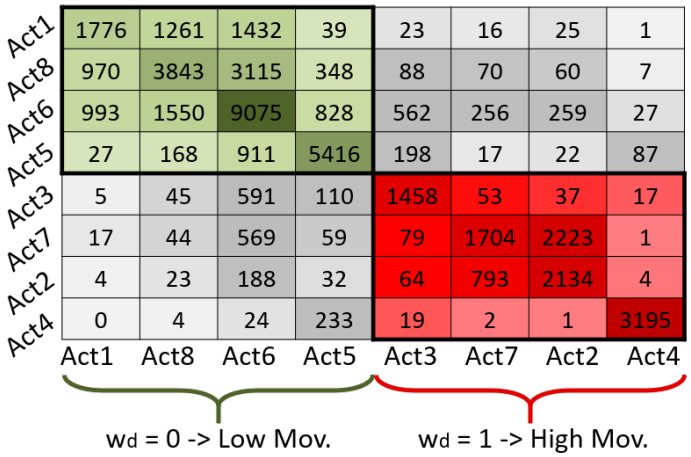

Fig. 9. Confusion matrix over the eight activities of all 15 subjects of PPG-Dalia predicted by our random forest.

Separating the activities in groups based on where the difference in terms of the average standard deviation of acceleration is the highest (i.e., between Act5 and Act3, see Figure 4), we define the two macro classes low and high movement, separated by thicker lines in Figure 9. In distinguishing these 2 classes, the same random forest classifier achieves a 92.2% accuracy. When testing ActPPG, we verified that using either the random forest predictions or directly the ground truth activity labels to select the most appropriate HR estimator yields almost identical results in terms of MAE and number of "big" model calls, proving that this accuracy is sufficient for our task. Therefore, we will employ the random forest predictions as a discriminator for the ActPPG framework in all the subsequent experiments. Note that for the case of N=2 considered in this work, even a simple threshold on the average standard deviation of the 3-axial acceleration can achieve decent performance in separating low/high movement windows. We use a RF not only to reach slightly more accurate results, but especially to make our framework easy to generalize to cases with N>2. Furthermore, the execution time and memory occupation of such a small RF is negligible compared to those of the HR tracking algorithms.

Table 2. In the first part of the table, the three algorithms used to test ActPPG. In the lower part, three configurations of ActPPG pairing two HR predictors and assigning Sitting, Working, Lunch and Driving activities to the smallest model in the pair. All results refer to PPG-Dalia dataset.

Alg.	S1 Mean ±sd	S2 Mean ±sd	S3 Mean ±sd	S4 Mean ±sd	S5 Mean ±sd	S6 Mean ±sd	S7 Mean ±sd	S8 Mean ±sd	S9 Mean ±sd	S10 Mean ±sd	S11 Mean ±sd	S12 Mean ±sd	S13 Mean ±sd	S14 Mean ±sd	S15 Mean ±sd
TimePPG & AT models: AT -1-, TimePPG-Small -2-, TimePPG-Big -3-															
-1-	8.49 ±9.7	12.97 ±10.4	9.13 ±8.7	9.40 ±8.5	22.23 ±19.1	17.55 ±16.2	6.17 ±6.5	9.83 ±9.6	13.58 ±11.0	9.99 ±14.1	14.80 ±15.2	6.82 ±6.6	7.16 ±7.7	8.59 ±8.0	8.07 ±9.5
-2-	5.57 ±6.1	4.57 ±6.1	2.92 ±3.5	5.70 ±6.9	12.53 ±18.3	6.38 ±9.5	3.12 ±4.6	6.76 ±6.6	7.60 ±8.0	3.57 ±5.6	6.87 ±9.3	8.47 ±8.1	2.69 ±3.2	3.47 ±4.4	4.34 ±7.1
-3-	4.01 ±4.9	3.16 ±4.2	2.27 ±3.0	4.62 ±5.7	14.96 ±22.4	4.28 ±8.9	2.58 ±3.7	6.02 ±6.9	7.61 ±8.8	2.89 ±4.8	4.79 ±7.1	6.95 ±8.8	2.54 ±2.9	3.01 ±4.0	3.46 ±6.3
ActPPG models (e.g., 1 + 2 = AT combined with TimePPG-Small)															
1 + 2	5.95 ±6.1	7.64 ±8.1	5.34 ±6.1	7.41 ±7.3	17.88 ±17.5	8.08 ±10.1	4.18 ±5.2	8.12 ±7.9	11.04 ±10.1	5.02 ±5.9	8.70 ±9.6	8.41 ±8.0	3.94 ±4.2	5.68 ±6.3	5.39 ±7.2
1 + 3	5.10 ±5.3	6.59 ±7.4	5.02 ±6.1	6.63 ±6.4	20.70 ±20.8	6.22 ±9.9	3.65 ±4.6	8.00 ±8.1	11.96 ±10.7	4.36 ±5.5	7.14 ±8.0	7.91 ±8.6	3.95 ±4.1	5.32 ±6.2	4.79 ±6.7
2 + 3	4.72 ±5.2	3.52 ±4.5	2.60 ±3.4	4.92 ±5.7	15.35 ±22.2	4.51 ±9.0	2.59 ±3.7	6.64 ±6.8	8.52 ±9.1	2.91 ±4.8	5.31 ±7.3	7.97 ±8.7	2.70 ±3.0	3.11 ±4.1	3.74 ±6.4

6.4 ActPPG: framework exploration

The lowermost rows of Table 2 and Table 3 report the results of applying the ActPPG framework for combining two out of the three HR prediction models highlighted in Figure 7 (AT, TimePPG-Small and TimePPG-Big). According to the previous movement detector results, these numbers are obtained by selecting the "small" HR predictor for "low movement" activities Act1 (Sitting), Act8 (Working), Act6 (Lunch), and Act5 (Driving), which span 51% of the total data.

Using the two TimePPG networks as elements of ActPPG (last row), we obtain a low MAE of 5.27 BPM with a complexity which is approximately 50% of that of the TimePPG-Big model alone on average. On the other hand, using the AT and TimePPG-Small allows reaching the lowest average complexity (∼ 38.8 kOPs) at the cost of an increased MAE (7.52 BPM), yet significantly lower than the 10.99 BPM achieved by AT alone.

6.4.1 ActPPG: ablation study. Here, we describe the two key elements that impact the performance and complexity of ActPPG: the number of activities assigned to the small model and the percentage of low movement windows.

Figure 10 shows the impact of moving the boundary between low and high movement activities. Specifically, activity classes are progressively added to the "low movement" set one by one, in the order of Figure 4. In the two extreme cases (leftmost and rightmost points), we exclusively use the framework's big/small model, respectively. Setting the threshold between activity 5 and 3, as shown in Figure 9, we obtain an error increase of 1.89, 2.28, and 0.39 BPM for the combinations AT/TimePPG-Small, AT/TimePPG-Big, and TimePPB Small/Big with respect to the big models alone, while reducing the computational complexity by ≈ 50%.

Notice that this threshold could also be changed dynamically at runtime, adapting the performance vs. complexity trade-off based on the system's status. For instance, we could envision an "energy-saving" mode with

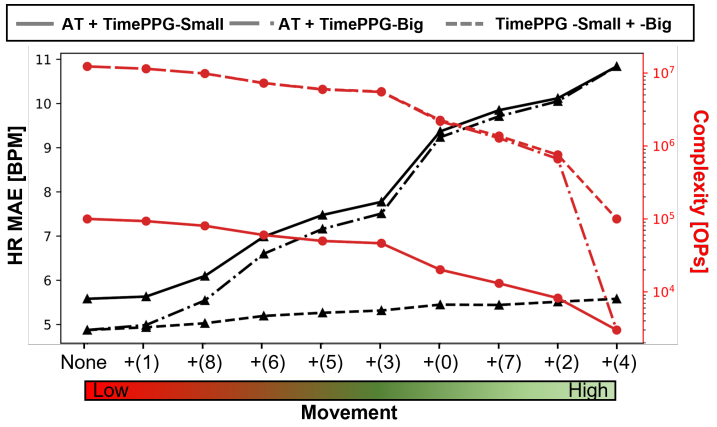

Fig. 10. ActPPG framework performance and complexity as a function of the low/high movement activities separation. From left to right, more activities are tracked using the "small" model of the pair.

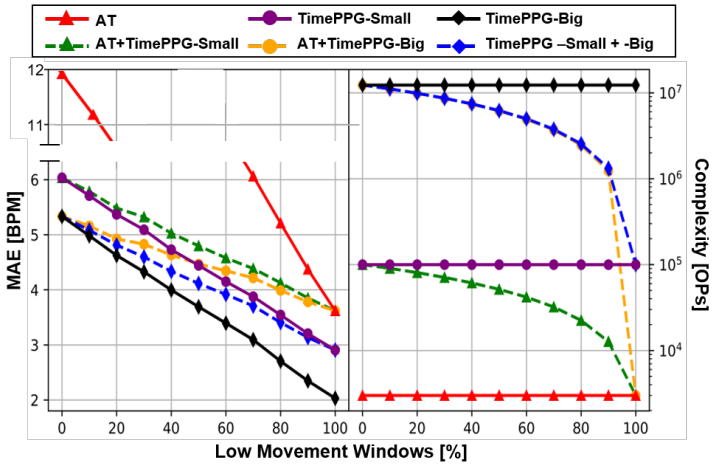

Fig. 11. Complexity and MAE of all our models for a changing percentage of low movement windows processed.

a high threshold (i.e., most activities in the low-movement set), activated only when the wrist-worn device's battery charge is low. Similarly, a "high-performance" mode with a lower threshold could be activated manually by users who need more precise HR tracking for a short time. Finally, a "normal-mode" with the default threshold between Act5 and Act3 could be active in all other scenarios.

Figure 11 analyzes the impact on our framework of users' daily routines, quantified as the number of low movement windows encountered by ActPPG. The percentage of windows has been artificially changed by sampling from the original PPGDalia windows a different portion of the two classes, considering only activity 1 as low movement and the others as a high movement. As the percentage of low movement windows increases, the algorithm's complexity always decreases, as expected, while the accuracy approximates the one of the smallest model. In the rest of this sub-section, we focus mainly on the case of AT + TimePPG-Small, which mostly

Table 3. Deployment of our models on the STM32L4R9AI.

	Ram [kB]	Flash [kB]	parameters	OPs	Cycles [#]	OPs/Cyc.	Time [ms]*	E. [mJ]*	MAE [BPM]+
Single Models									
AT (-1-)	4.09	0	0	3k	100k	0.03	1.25	0.017	10.99
TimePPG-Small (-2-)	7.68	18.54	5.09k	77.63k	1.36M	0.057	17.06	0.232	5.63
TimePPG-Small†	13.31	8.55	8.76k	224.8k	1.52M	0.148	19.02	0.259	5.60
TimePPG-Big (-3-)	129.64	902.21	232.6k	12.27M	103.16M	0.119	1289.5	17.57	4.88
TimePPG-Big†	34.06	884.26	902.2k	33.3M	104.14M	0.320	1301.85	17.74	4.87
ActPPG models (e.g., 1 + 2 = AT combined with TimePPG-Small)									
1 + 2	7.68	20.04	6.63k	38.81k	732.4k	0.053	9.16	0.125	7.52
1 + 3	129.64	903.71	234.14k	6.13M	51.63M	0.119	645.38	8.796	7.16
2 + 3	129.64	922.25	239.2k	6.17M	52.26M	0.118	653.28	8.903	5.27

† With int8 quantization-aware training. Dilation reduced to one, with 0-padded filters to maintain the receptive field, to cope with toolchain limitations.

* Measured on the STM32L4R9AI with a frequency of 80MHz and a power consumption of 13.63 mW.

+ Measured on the PPG-Dalia dataset.

highlights this aspect. Only executing AT, we have a constant low complexity of 3k OPs, but with quite bad performance results when high movement windows are frequent (leftmost part of the graph), reaching 12 BPM of MAE. On the other hand, only executing TimePPG-small, we have a constant complexity of 77.63k OPs, but with a good MAE both in high movement (6 BPM) and low movement conditions (2.91 BPM). In this case, the application of our framework (green dotted line of Figure 11) bounds the MAE in the [3.62, 6.03] BPM interval, avoiding bad performance in high movement situations while reducing the complexity of up to 33× during low movement situations compared to the execution of TimePPG-Small alone. In other words, ActPPG can bring the advantages of both small and big models by employing the appropriate model in every situation, thus minimally increasing the MAE (green vs. purple curves) while significantly reducing the mean complexity and therefore the energy consumption.

6.5 Algorithms Deployment

This section details the results obtained deploying some of our solutions (both individual models and ActPPG combinations using the default activity separation threshold fixed between Act5 and Act3) on the embedded platform described in Section 2.3. We report the memory occupation, the inference time, and the energy consumption for tracking an 8 seconds window in Table 3. The results refer to the deployment on the STM32L4R9AI MCU, clocked at 80 MHz, with a power consumption of 13.63 mW. Given that *int8* quantized deep neural networks reach iso-accuracies compared to their *float32* counterparts in literature [50], we show both the results of the original float32 and of int8 models (†). We used NEMO [17] to re-train our models with a quantization-aware training, achieving negligible accuracy loss compared to float32. For model deployment, we used CUBE.AI 5.1.2, which supports both full-precision and quantized models. However, the toolchain does not yet support dilation on quantized models. Therefore, we had to manually extend the convolution filters interleaving them with 0s to

replicate the effect of dilation[2]. Further energy and latency reductions (at iso-accuracy) could be achieved once the toolchain adds support for quantized dilation.

The simple AT algorithm of [42] achieves a very short latency of 1.25 ms, which is 13.65×/1041.5× lower than the ones of our two extreme TimePPG outputs, i.e., TimePPG-Small and TimePPG-Big, which reach 17.06 ms and 1289.5 ms in their float32 deployment. When quantized, both TCNs show a similar latency (~ 2 ms slower) despite the higher MACs/cycle achieved (0.148 vs 0.057 for TimePPG-Small and 0.320 vs. 0.119 for TimePPG-Big). This is due to the the larger total number of MACs executed (2.7× / 2.9× more respectively) because of the 0-interleaving described above. Using quantized models also reduces the memory occupation (although less than what would be achieved without the additional 0s), which is beneficial especially if the MCU has to store multiple networks. Notably, all the algorithms meet the requirement of having a total execution time < 2s and can be considered suitable for real-time HR tracking.

In terms of energy, AT consumes just 17 μJ, while TimePPG-Small and TimePPG-Big (deployed in float) consume 0.232 mJ and the 17.57 mJ respectively. Note in particular that both the inference time and the energy consumption of TimePPG-Big are two orders of magnitude larger than those of TimePPG-Small, due to the large number of computations performed by this TCN. Combining this model with TimePPG-Small through the ActPPG framework, we reduce its total energy consumption by ~ 50%, from 17.57 mJ to 8.903 mJ, with a MAE increase of just 0.39 BPM. Similarly, also the ActPPG combination of AT and TimePPG-Small yields relevant energy consumption reductions compared to the latter, with a significantly lower error than the former. All ActPPG results in Table 3 also include the memory and latency overheads due to the RF-based movement detector.

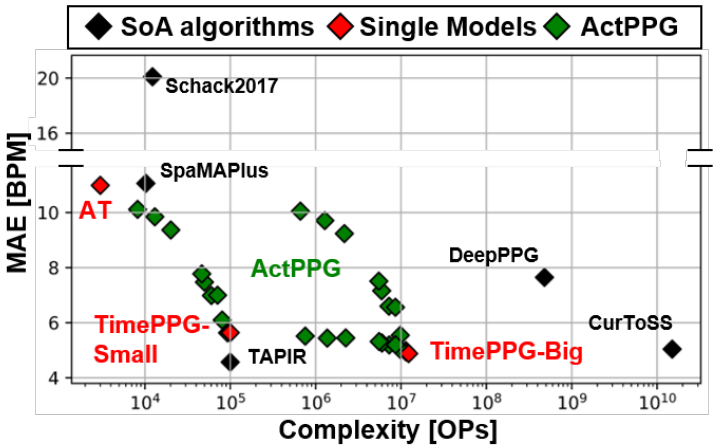

Fig. 12. Comparison with SoA algorithms in the Complexity vs. MAE space. Red points represent the three models employed in ActPPG, while green points represents Pareto-optimal solutions of ActPPG obtained by changing the threshold between low and high movement activities. For deep learning methods, we consider MACs as OPs.

6.6 State-of-the-art comparison

Figure 12 shows a full comparison of our models (green and red diamonds) with state-of-the-art algorithms (black ones), including both classical and deep learning solutions, benchmarked on the PPG-Dalia dataset. Green diamonds represent ActPPG results obtained varying the threshold between low and high movement activities as

[2]For instance, a filter with $C_{in} = 1$, $K = 3$, $d = 2$ and values {3,-7,43} had to be changed to $K = 5$ with values {3, 0, -7, 0, 43}

Table 4. Comparison of TimePPG models with state-of-the-art methods. The table reports the mean MAE over subjects ± the MAE standard deviation over subjects, when available. For our work, we report TimePPG-Small and TimePPG-Big obtained for PPG-Dalia (top) and SPC2015 (bottom). Abbreviations: Mem.: memory, B: Byte, E.: energy.

	MAE [BPM]		Deployment			
Algorithm	SPC2015	PPG-Dalia	[Ops/MACs]	Parameters [#]	Mem. [B]	E.[mJ]
Classical Algorithms						
TROIKA, 2014 [54]	2.34 ± 0.82	n.a.	n.a.	n.a.	n.a.	n.a.
JOSS, 2015 [53]	1.28 ± 2.61	n.a.	15.0B	n.a.	n.a.	n.a.
SpaMaPlus[1], 2016 [40]	4.25 ± 5.9	11.06 ± 4.8	10.3k	n.a.	n.a.	n.a.
Schack, 2017 [41]	2.91 ± 4.6	20.45 ± 7.1	12.3k	n.a.	n.a.	n.a.
TAPIR, 2020 [23]	2.5 ± 1.2	4.57 ± 1.4	100k	n.a.	n.a.	n.a.
Arunkumar, 2020 [5]	1.03 ± 1.49	n.a.	n.a.	n.a.	n.a.	n.a.
CurToSS, 2020 [55]	2.2	5.0 ± 2.8	15.0B	n.a.	n.a.	n.a.
AT	11.92 ± 5.39	10.99 ± 4.46	3k	0	0	17 μJ
Deep Learning Algorithms						
CNN, 2019 [36]	4.0 ± 5.4	7.65 ± 4.2	480M	60M	240M	n.a.
CNN constr., 2019 [36]	n.a.	9.99 ± 5.9	380k	26k	26k	n.a.
PPGNet, 2019 [43]	3.36 ± 4.1	n.a.	n.a.	765k	n.a.	n.a.
CorNET, 2019 [8]	4.67 ± 3.71	n.a	20M	250k	1M	n.a.
Bin CorNET, 2020 [39]	6.78 ± 5.29	n.a	20M	250k	260k	56.1 μJ[2]
Chung, 2020 [15]	1.46[3]	n.a.	17M	3.3M	n.a.	n.a.
DeepHeart, 2021 [14]	1.61	n.a.	1101M	4.3M	n.a.	n.a.
Our work						
TimePPG-Small	4.82 ± 2.75	5.63 ± 2.63	77.6k	5k	18.54k	232 μJ[4]
			1.77M	11k	43.31k	3.54 mJ[4]
TimePPG-Big	3.27 ± 2.04	4.88 ± 3.23	12.3M	232k	902.2k	17.57 mJ[4]
			16.4M	168k	658.8k	23.98 mJ[4]

[1] Original paper reported MAE = 0.89 BPM on SPC2015, but not using leave-one-subject-out validation.

[2] Deployed on ASIC.

[3] The SPC2015 dataset has been used only for training and thus the accuracy is not computed using leave-one-subject-out validation.

[4] Deployed on general purpose MCU STM32L4R9AI.

in Figure 10. Our models cover the space from 4.88 BPM to 10.99 BPM of MAE, with a complexity that spans from 12 MOps to 0.003 MOPs. As shown in the graph, both our individual models and our adaptive framework obtain results comparable with the state-of-the-art, sometimes significantly reducing the complexity compared to other approaches with similar accuracy. For instance, TimePPG-Big achieves a lower MAE than CurToSS (4.88

BPM vs. 5.0 BPM) but has 1000× lower complexity. Further, compared to DeepPPG (the previous state-of-the-art deep learning model tested on PPG-Dalia), TimePPG-Small achieves better MAE (5.63 BPM vs. 7.65 BPM), while reducing the size and the OPs by 12000× and 4800×, respectively.

Table 4 further strengthens this conclusion, by comparing our individual TimePPG models to state-of-the-art techniques tested on both the SPC2015 and the PPG-Dalia dataset[3]. Compared to previous deep learning algorithms [8, 14, 15, 43], our approach replaces more expensive layers such as LSTMs with dilated 1D convolutions, allowing a strong reduction in the number of parameters and in the computational time. As shown, both "extreme" TimePPG models are characterized by less parameters and MACs compared to the state-of-the-art, while achieving comparable or lower MAE on both the benchmark datasets. For instance, TimePPG-Small has 3.8×/4800× lower MACs and 5.2×/12000× lower parameters compared to others deep learning approaches, while achieving better performance on the PPG-Dalia dataset (5.63 vs. 7.65 BPM). Similarly, TimePPG-Big obtains comparable performance to PPGNet on the SPC2015 dataset (3.27 vs 3.36 BPM) with 4.6× less parameters. DeepHeart [14] outperforms our networks in terms of MAE on the SPC2015 dataset, obtaining a 1.61 BPM. However, this network can not be deployed on constrained MCUs such as the ones considered in this work, given that it requires 1101M MACs and has 4.3M of parameters, not fitting the typical < 1 MB memory of microcontrollers.

While our TCNs outperform previous deep learning approaches, some model-driven methods still obtain slightly better results on both datasets, reaching as low as 4.57 BPM of MAE on PPG-Dalia and 1.03 BPM on SPC2015. However, these methods have different shortcomings. Firstly, they perform poorly both when tested on different data compared to those on which they are tuned, or under slight parameters modifications. Indeed, most model-based algorithms (e.g., [40, 41]) tuned for SPC2015 perform much worse on PPG-Dalia, confirming that the tuning of their parameters is critical and not trivial. The few exceptions (e.g., [23, 55]) are only due to the fact that the parameters of those algorithms have been tuned on both datasets *simultaneously*, and without leave-one-subject-out validation. Secondly, none of the model-based algorithms (except the trivial AT) has been deployed for real-time execution on an edge device, neither a general purpose MCU or a dedicated ASIC/FPGA. Vice versa, they are only tested offline and without consideration of real-time execution or energy consumption.

Moreover, as previously discussed in Section 4.3, our models are strongly impaired by subject 5, whose large HR values are rarely encountered in training data and are therefore badly predicted by data-driven approaches. In fact, if we apply the additional fine-tuning step described in Section 4.3, TimePPG-Big (and consequently also the ActPPG models based on it) outperforms also classical methods, reaching 3.84 BPM of MAE on PPG-Dalia. Due to the smaller size of the dataset, and to the way in which it was collected (with the first 25% of all subjects' records containing only low-movement activities), the proposed fine-tuning strategy does not yield similar improvements on SPC2015. However, note that as explained in Section 4.3, the purpose of this experiment was not to reduce the MAE per se, but just to show that our proposed TCNs could achieve even better performance given the availability of a larger and more comprehensive dataset.

6.7 Limitations

In this section, we summarize the limitations of the proposed method, most of which have been addressed individually elsewhere in the manuscript. Similarly to all other PPG-based HR tracking algorithms (both classic and deep learning ones), also our TimePPG TCNs are negatively affected by intense MAs. This is discussed in Sections 5 and 6.4, and shown clearly by Figures 5 and 11, which demonstrate the better performance achieved by individual models for low-movement inputs. However, the ActPPG framework goes exactly in the direction of

[3]To allow a direct comparison to previous models, we also collected a Pareto-optimal frontier of TCN models on the SPC2015 12 subjects dataset always using the leave-one-subject-out validation and the parameters proposed in [36]. As for PPG-Dalia dataset, "small" and "big" models are extracted from the Pareto front and reported in Table 1.

addressing this intrinsic limitations of individual models, providing a mechanism to achieve good HR tracking performance at the minimum possible cost, automatically adapting to the amount of MAs.

Since we rely on pre-collected datasets (PPG-Dalia and SPC2015) for the training and evaluation of our models, the effectiveness of the proposed method with different PPG sensing hardware, e.g., not wrist-mounted, or in general providing an output signal with different characteristics, cannot be assessed. Once again, this limitation is shared with the majority of previous literature, which similarly provides results only on pre-collected, open-source, data. Our plans for future works go in the direction of addressing this limitation, through the collection of larger and more heterogeneous datasets in terms of activities, HR ranges, and sensing hardware.

This also relates with the last limitation of our method, i.e., the fact that our TCNs need large amounts of training data to be accurate. This is a common problem of data-driven algorithms, and in particular deep learning ones, and was discussed extensively in Sections 4.3.2 and 6.6. In particular, the need for large datasets is demonstrated by the much better results (in relation with the state-of-the-art) achieved by our method on PPG-Dalia, compared to the smaller SPC2015. The positive effect of fine-tuning on Dalia also confirms this point. Nonetheless, even without fine-tuning, our TCNs almost match the state-of-the-art on Dalia, and significantly outperform all previous data-driven solutions. This demonstrates that, thanks to the use of small and optimized neural networks, our method is less affected than previous deep learning ones by this problem. Moreover, this limitation can also be seen as unexpressed potential. In fact, even PPG-Dalia is a very small dataset by today's deep learning standards. So, the fact that we are competitive on it is an indication that, given the availability of more data, our method could achieve even better performance.

7 CONCLUSIONS

HR tracking is increasingly executed on wearable devices both in clinical contexts and in daily lives. However, the advanced HR tracking algorithms proposed in the literature are tested only offline and not always applicable in real-time on constrained devices. To the best of our knowledge, there is a lack of research results focusing on the deployment of advanced algorithms on MCU-class systems and on their optimization from the point of view of energy consumption.

In this work, we have presented a two-fold contribution that goes in this direction. First, we have presented TimePPG, a set of deep learning models for HR tracking based on TCN architectures, which achieve excellent performance with a limited number of parameters. All models are obtained, starting from a single seed using an automatic NAS approach. Second, we have proposed a run-time framework, ActPPG, to select among multiple HR prediction algorithms based on the movement intensity. The intuition behind this solution comes from the fact that a higher movement intensity causes more relevant MAs and is, therefore, more difficult to process. Both TimePPG and ActPPG obtain accurate results that are competitive with the state-of-the-art, while executing in real-time on a low-power MCU. On the PPGDalia dataset, the largest TimePPG model, coupled with post-processing and fine-tuning, reduces the MAE to as low as 3.84 BPM. Moreover, ActPPG generates a rich set of Pareto solutions that achieve MAEs in the range [4.88, 10.99] BPM and span four orders of magnitude in complexity, between 12.3 MOPs and 0.003 MOPs.

8 ACKNOWLEDGMENTS

Support was received from the Hasler Foundation under project 18082, and the EU's Horizon 2020 Research and Innovation Program through the project MNEMOSENE under Grant 780215.

REFERENCES

[1] 2020. Neurosky. 'http://neurosky.com/wp-content/uploads/2016/06/TOF-side-by-side-competitor-comparison.pdf'

[2] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz

Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[3] Rasha M Al-Eidan, Hend Al-Khalifa, and Abdul Malik Al-Salman. 2018. A review of wrist-worn wearable: Sensors, models, and challenges. *Journal of Sensors* 2018 (2018).

[4] Apple. [n.d.]. *Apple Watch Series.* https://www.apple.com/lae/watch/

[5] KR Arunkumar and M Bhaskar. 2020. Robust De-Noising Technique for Accurate Heart Rate Estimation Using Wrist-Type PPG Signals. *IEEE Sensors Journal* 20, 14 (2020), 7980–7987.

[6] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).

[7] BiosignalsPLUX. [n.d.]. *Respiban Professional 2019.* https://biosignalsplux.com/products/wearables/respiban-pro.html

[8] Dwaipayan Biswas, Luke Everson, Muqing Liu, Madhuri Panwar, Bram-Ernst Verhoef, Shrishail Patki, Chris H Kim, Amit Acharyya, Chris Van Hoof, Mario Konijnenburg, et al. 2019. CorNET: Deep learning framework for PPG-based heart rate estimation and biometric identification in ambulant environment. *IEEE transactions on biomedical circuits and systems* 13, 2 (2019), 282–291.

[9] D. Biswas, N. Simões-Capela, C. Van Hoof, and N. Van Helleputte. 2019. Heart Rate Estimation From Wrist-Worn Photoplethysmography: A Review. *IEEE Sensors Journal* 19, 16 (2019), 6560–6570. https://doi.org/10.1109/JSEN.2019.2914166

[10] Alessio Burrello, Simone Benatti, Kaspar Anton Schindler, Luca Benini, and Abbas Rahimi. 2020. An Ensemble of Hyperdimensional Classifiers: Hardware-Friendly Short-Latency Seizure Detection with Automatic iEEG Electrode Selection. *IEEE journal of biomedical and health informatics* (2020).

[11] Alessio Burrello, Kaspar Anton Schindler, Luca Benini, and Abbas Rahimi. 2020. Hyperdimensional Computing with Local Binary Patterns: One-shot Learning for Seizure Onset Detection and Identification of Ictogenic Brain Regions from Short-time iEEG Recordings. *IEEE transactions on bio-medical engineering* 67, 2 (2020), 601–613.

[12] Han Cai, Ligeng Zhu, and Song Han. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332* (2018).

[13] Denisse Castaneda, Aibhlin Esparza, Mohammad Ghamari, Cinna Soltanpur, and Homer Nazeran. 2018. A review on wearable photoplethysmography sensors and their potential future applications in health care. *International journal of biosensors & bioelectronics* 4, 4 (2018), 195.

[14] Xiangmao Chang, Gangkai Li, Guoliang Xing, Kun Zhu, and Linlin Tu. 2021. DeepHeart: A Deep Learning Approach for Accurate Heart Rate Estimation from PPG Signals. *ACM Trans. Sen. Netw.* 17, 2, Article 14 (Jan. 2021), 18 pages. https://doi.org/10.1145/3441626

[15] Heewon Chung, Hoon Ko, Hooseok Lee, and Jinseok Lee. 2020. Deep Learning for Heart Rate Estimation From Reflectance Photoplethysmography With Acceleration Power Spectrum and Acceleration Intensity. *IEEE Access* 8 (2020), 63390–63402.

[16] Heewon Chung, Hooseok Lee, and Jinseok Lee. 2018. Finite state machine framework for instantaneous heart rate validation using wearable photoplethysmography during intensive exercise. *IEEE journal of biomedical and health informatics* 23, 4 (2018), 1595–1606.

[17] Francesco Conti. 2020. Technical Report: NEMO DNN Quantization for Deployment Model. arXiv:2004.05930 [cs.LG]

[18] Francesco Daghero, Alessio Burrello, Daniele Jahier Pagliari, Luca Benini, Enrico Macii, and Massimo Poncino. 2020. Energy-Efficient Adaptive Machine Learning on IoT End-Nodes With Class-Dependent Confidence. In *Proceedings of the International Conference on Electronics Circuits and Systems (ICECS 2020)*. IEEE, 1–4.

[19] Empatica. [n.d.]. *E4 Wristband 2014.* https://www.empatica.com/en-eu/research/e4/

[20] Fitbit. [n.d.]. *Fitbit Charge 4.* https://www.fitbit.com/global/us/products/trackers/charge4

[21] Z Ge, PWC Prasad, N Costadopoulos, Abeer Alsadoon, AK Singh, and A Elchouemi. 2016. Evaluating the accuracy of wearable heart rate monitors. In *2016 2nd International Conference on Advances in Computing, Communication, & Automation (ICACCA)(Fall)*. IEEE, 1–6.

[22] Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. 2018. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1586–1595.

[23] Nicholas Huang and Nandakumar Selvaraj. 2020. Robust PPG-based Ambulatory Heart Rate Tracking Algorithm. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 5929–5934.

[24] T. M. Ingolfsson, M. Hersche, X. Wang, N. Kobayashi, L. Cavigelli, and L. Benini. 2020. EEG-TCNet: An Accurate Temporal Convolutional Network for Embedded Motor-Imagery Brain–Machine Interfaces. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2958–2965. https://doi.org/10.1109/SMC42975.2020.9283028

[25] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[26] Daniele Jahier Pagliari, Enrico Macii, and Massimo Poncino. 2018. Dynamic Bit-width Reconfiguration for Energy-Efficient Deep Learning Hardware. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED '18)*. ACM, New York, NY, USA. http://doi.acm.org/10.1145/3218603.3218611

[27] Hao-Yu Jan, Mei-Fen Chen, Tieh-Cheng Fu, Wen-Chen Lin, Cheng-Lun Tsai, and Kang-Ping Lin. 2019. Evaluation of Coherence Between ECG and PPG Derived Parameters on Heart Rate Variability and Respiration in Healthy Volunteers With/Without Controlled Breathing. *Journal of Medical and Biological Engineering* 39, 5 (2019), 783–795.

[28] Colin Lea, Rene Vidal, Austin Reiter, and Gregory D Hager. 2016. Temporal convolutional networks: A unified approach to action segmentation. In *European Conference on Computer Vision*. Springer, 47–54.

[29] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, et al. 2020. Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems* 33 (2020).

[30] Mahdi Boloursaz Mashhadi, Ehsan Asadi, Mohsen Eskandari, Shahrzad Kiani, and Farokh Marvasti. 2015. Heart rate tracking using wrist-type photoplethysmographic (PPG) signals during physical exercise with simultaneous accelerometry. *IEEE Signal Processing Letters* 23, 2 (2015), 227–231.

[31] ST Microelectornics. [n.d.]. *STM32L476*. https://www.st.com/resource/en/datasheet/stm32l476je.pdf

[32] L. Mocerino and A. Calimera. 2020. Fast and Accurate Inference on Microcontrollers With Boosted Cooperative Convolutional Neural Networks (BC-Net). *IEEE Transactions on Circuits and Systems I: Regular Papers* (2020), 1–12. https://doi.org/10.1109/TCSI.2020.3039116

[33] Benjamin W Nelson, Carissa A Low, Nicholas Jacobson, Patricia Areán, John Torous, and Nicholas B Allen. 2020. Guidelines for wrist-worn consumer wearable assessment of heart rate in biobehavioral research. *NPJ Digital Medicine* 3, 1 (2020), 1–9.

[34] Ralph Neuneier and Hans Georg Zimmermann. 1998. How to train neural networks. In *Neural networks: tricks of the trade*. Springer, 373–423.

[35] Eunhyeok Park, Dongyoung Kim, Soobeom Kim, Yong-Deok Kim, Gunhee Kim, Sungroh Yoon, and Sungjoo Yoo. 2015. Big/little deep neural network for ultra low power inference. In *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ ISSS)*. IEEE, 124–132.

[36] Attila Reiss, Ina Indlekofer, Philip Schmidt, and Kristof Van Laerhoven. 2019. Deep PPG: large-scale heart rate estimation with convolutional neural networks. *Sensors* 19, 14 (2019), 3079.

[37] Lei Ren, Yuxin Liu, Xiaokang Wang, Jinhu Lü, and M Jamal Deen. 2020. Cloud-Edge based Lightweight Temporal Convolutional Networks for Remaining Useful Life Prediction in IIoT. *IEEE Internet of Things Journal* (2020).

[38] Matteo Risso, Alessio Burrello, Daniele Jahier Pagliari, Simone Benatti, Enrico Macii, Luca Benini, and Massimo Poncino. 2021. Robust and Energy-efficient PPG-based Heart-Rate Monitoring. In *To appear at International Symposium on Circuit and Systems (ISCAS) 2021*. IEEE.

[39] Leandro Giacomini Rocha, Dwaipayan Biswas, Bram-Ernst Verhoef, Sergio Bampi, Chris Van Hoof, Mario Konijnenburg, Marian Verhelst, and Nick Van Helleputte. 2020. Binary corNET: accelerator for HR estimation from wrist-PPG. *IEEE Transactions on Biomedical Circuits and Systems* 14, 4 (2020), 715–726.

[40] Seyed Salehizadeh, Duy Dao, Jeffrey Bolkhovsky, Chae Cho, Yitzhak Mendelson, and Ki H Chon. 2016. A novel time-varying spectral filtering algorithm for reconstruction of motion artifact corrupted heart rate signals during intense physical activities using a wearable photoplethysmogram sensor. *Sensors* 16, 1 (2016), 10.

[41] Tim Schäck, Michael Muma, and Abdelhak M Zoubir. 2017. Computationally efficient heart rate estimation during physical exercise using photoplethysmographic signals. In *2017 25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2478–2481.

[42] Hang Sik Shin, Chungkeun Lee, and Myoungho Lee. 2009. Adaptive threshold method for the peak detection of photoplethysmographic waveform. *Computers in biology and medicine* 39, 12 (2009), 1145–1152.

[43] A Shyam, Vignesh Ravichandran, SP Preejith, Jayaraj Joseph, and Mohanasankar Sivaprakasam. 2019. PPGnet: Deep network for device independent heart rate estimation from photoplethysmogram. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 1899–1902.

[44] Nina Sviridova and Kenshi Sakai. 2015. Human photoplethysmogram: new insight into chaotic characteristics. *Chaos, Solitons & Fractals* 77 (2015), 53–63.

[45] Toshiyo Tamura, Yuka Maeda, Masaki Sekine, and Masaki Yoshida. 2014. Wearable photoplethysmographic sensors—past and present. *Electronics* 3, 2 (2014), 282–302.

[46] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. 2019. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2820–2828.

[47] Andriy Temko. 2017. Accurate heart rate monitoring during physical exercises using PPG. *IEEE Transactions on Biomedical Engineering* 64, 9 (2017), 2016–2024.

[48] Anjali S Yeole and Dhananjay R Kalbande. 2016. Use of Internet of Things (IoT) in healthcare: A survey. In *Proceedings of the ACM Symposium on Women in Research 2016*. 71–76.

[49] Ming Yuan and Yi Lin. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 68, 1 (2006), 49–67.

[50] Marcello Zanghieri, Simone Benatti, Alessio Burrello, Victor Kartsch, Francesco Conti, and Luca Benini. 2019. Robust Real-Time Embedded EMG Recognition Framework Using Temporal Convolutional Networks on a Multicore IoT Processor. *IEEE Transactions on Biomedical Circuits and Systems* (2019).

[51] Xiang Zhang, Lina Yao, Xianzhi Wang, Jessica Monaghan, David Mcalpine, and Yu Zhang. 2019. A survey on deep learning based brain computer interface: Recent advances and new frontiers. *arXiv preprint arXiv:1905.04149* (2019).

[52] Yangsong Zhang, Benyuan Liu, and Zhilin Zhang. 2015. Combining ensemble empirical mode decomposition with spectrum subtraction technique for heart rate monitoring using wrist-type photoplethysmography. *Biomedical Signal Processing and Control* 21 (2015), 119–125.

[53] Zhilin Zhang. 2015. Photoplethysmography-based heart rate monitoring in physical activities via joint sparse spectrum reconstruction. *IEEE transactions on biomedical engineering* 62, 8 (2015), 1902–1910.

[54] Zhilin Zhang, Zhouyue Pi, and Benyuan Liu. 2014. TROIKA: A general framework for heart rate monitoring using wrist-type photoplethysmographic signals during intensive physical exercise. *IEEE Transactions on biomedical engineering* 62, 2 (2014), 522–531.

[55] Menglian Zhou and Nandakumar Selvaraj. 2020. Heart Rate Monitoring using Sparse Spectral Curve Tracing. In *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. IEEE, 5347–5352.