End-to-end Automated Deep Neural Network Optimization for PPG-based Blood Pressure Estimation on Wearables

End-to-end Automated Deep Neural Network Optimization for PPG-based Blood Pressure Estimation on Wearables

FRANCESCO CARLUCCI and GIOVANNI POLLO, Politecnico di Torino, Torino, Italy
XIAYING WANG, ETH Zurich, Zurich, Switzerland
MASSIMO PONCINO and ENRICO MACII, Politecnico di Torino, Torino, Italy
LUCA BENINI, Università di Bologna, Bologna, Italy and ETH Zurich, Zürich, Switzerland
SARA VINCO, ALESSIO BURRELLO, and DANIELE JAHIER PAGLIARI, Politecnico di Torino, Torino, Italy

Photoplethysmography-based Blood Pressure (BP) estimation is a challenging task, particularly on resource-constrained wearable devices. However, fully on-board processing is desirable to ensure user data confidentiality. Recent Deep Neural Networks (DNNs) have achieved high BP estimation accuracy by reconstructing BP waveforms or directly regressing BP values, but their large memory, computation, and energy requirements hinder deployment on wearables. This work introduces a fully automated DNN design pipeline that combines hardware-aware Neural Architecture Search, pruning, and Mixed-Precision Search to generate accurate yet compact BP prediction models optimized for ultra-low-power multi-core Systems-on-Chip (SoCs). Starting from state-of-the-art baseline models on four public datasets, our optimized networks achieve up to 7.99% lower error with a 7.5× parameter reduction, or up to 83× fewer parameters with negligible accuracy loss. All models fit within 512 kB of memory on our target SoC (GreenWaves' GAP8), requiring less than 55 kB and achieving an average inference latency of 142 ms and energy consumption of 7.25 mJ. Patient-specific fine-tuning further improves accuracy by up to 64%, enabling fully autonomous, low-cost BP monitoring on wearables.

CCS Concepts: • **Applied computing** → **Consumer health**; **Health informatics**; • **Computing methodologies** → **Neural networks**; **Supervised learning by regression**;

Additional Key Words and Phrases: PPG, Neural Architecture Search, Blood Pressure, DNN, On-board

ACM Reference format:
Francesco Carlucci, Giovanni Pollo, Xiaying Wang, Massimo Poncino, Enrico Macii, Luca Benini, Sara Vinco, Alessio Burrello, and Daniele Jahier Pagliari. 2026. End-to-end Automated Deep Neural Network Optimization for PPG-based Blood Pressure Estimation on Wearables. *ACM Trans. Comput. Healthcare* 7, 3, Article 50 (June 2026), 27 pages.
https://doi.org/10.1145/3809503

This work was supported under grant number DM117.
Authors' Contact Information: Francesco Carlucci, Politecnico di Torino, Torino, Italy; e-mail: francesco.carlucci@polito.it; Giovanni Pollo (corresponding author), Politecnico di Torino, Torino, Italy; e-mail: giovanni.pollo@polito.it; Xiaying Wang, ETH Zurich, Zurich, Switzerland; e-mail: xiaywang@iis.ee.ethz.ch; Massimo Poncino, Politecnico di Torino, Torino, Italy; e-mail: massimo.poncino@polito.it; Enrico Macii, Politecnico di Torino, Torino, Italy; e-mail: enrico.macii@polito.it; Luca Benini, Università di Bologna, Bologna, Italy and ETH Zurich, Zürich, Switzerland; e-mail: luca.benini@unibo.it; Sara Vinco, Politecnico di Torino, Torino, Italy; e-mail: sara.vinco@polito.it; Alessio Burrello, Politecnico di Torino, Torino, Italy; e-mail: alessio.burrello@polito.it; Daniele Jahier Pagliari, Politecnico di Torino, Torino, Italy; e-mail: daniele.jahier@polito.it.

ACM 2637-8051/2026/6-ART50
https://doi.org/10.1145/3809503

1 Introduction

Blood Pressure (BP) is a critical health parameter. It is one of the most significant risk factors for **Cardiovascular Diseases (CVDs)** [24], including High Blood Pressure, atrial fibrillation, myocardial infarction, and aortic rupture [47, 52]. These conditions contribute to a high number of deaths worldwide [32]. Consequently, periodic or continuous BP monitoring is highly desirable for early diagnosis and prevention [26].

BP monitoring techniques can be broadly classified into two categories: invasive and non-invasive methods [39, 49, 65]. Among invasive methods, arterial cannulation is the most accurate. This technique, which directly measures **Atrial Blood Pressure (ABP)**, involves inserting a thin catheter with a cannula directly into an artery. However, due to the necessity of specialized equipment and trained personnel, it is predominantly used in hospital settings for precise BP monitoring.

Among non-invasive methods, cuff-based sphygmomanometers are the most widely used and accurate. These devices consist of a pressure cuff wrapped around the upper arm, an air pump, and a manometer. By inflating the cuff to compress the artery and gradually releasing the pressure, BP is measured with high reliability. However, despite their accuracy and non-invasiveness, cuff-based methods are not suitable for continuous monitoring, as they require specialized equipment uncomfortable to wear during daily activities and a controlled environment.

A promising alternative that enables both non-invasive and continuous BP monitoring is **Photoplethysmography (PPG)**. PPG is an optical technique that measures blood volume changes in the skin using **Light-Emitting Diodes (LEDs)** and photodetectors. This method has gained significant popularity due to its integration into consumer wearable devices such as smartwatches and fitness bands. An LED sensor illuminates the skin, and a photodiode captures the reflected light, the intensity of which varies according to blood volume fluctuations caused by heart activity [35].

Other medically relevant parameters can be derived from PPG with high accuracy, such as **Heart Rate (HR)** [8, 14] and respiratory rate [9]. However, in this article, we focus on its usage for the estimation of **Systolic Blood Pressure (SBP)** and **Diastolic Blood Pressure (DBP)**, which is an actively explored field, with accuracies not reaching medical grade.

Given the PPG signal, estimating the SBP and DBP is non-trivial. Different techniques have already been explored, including **Machine Learning (ML)** methods that rely on hand-crafted features, such as Pulse Transit Time [51], as well as **Deep Learning (DL)** approaches that learn directly from raw or minimally processed signals [18, 22, 42]. Among ML methods, techniques such as a **Random Forest (RF)** [28] or an ensemble of **Support Vector Regression (SVR)** [23] trained with bagging, have been explored. Compared to traditional ML-based methods, **Deep Neural Networks (DNNs)** offer the advantage of bypassing the often costly feature extraction process and have demonstrated superior generalization capabilities for unseen data in biosignal processing tasks [13, 15, 70]. Different DNN architectures have been investigated for PPG-based BP estimation [4, 18, 22, 30], with recent research focusing on 1D **Convolutional Neural Networks (CNNs)** [70].

However, existing DL models for BP estimation have a large number of parameters and high computational complexity [6, 40, 63]. When pursuing continuous monitoring on resource-constrained, low-power devices such as wearables, those models either exceed the available memory or incur excessive latency [11]. As a result commercial wearable BP systems [5] adopt remote processing pipelines, where raw data is collected and processed on the cloud. This approach is non-real time and raises concerns on data privacy and security [19, 41].

This article, which extends the preliminary work presented in [11], aims at reducing the complexity of DNN-based methods while maintaining their high accuracy, through the use of a fully automated hardware-aware DNN compression pipeline. The starting points of our optimization are DNN architectures derived by a previous study [27], representing the current **State-of-the-Art (SotA)** for the task. The following are the main contributions of this work:

—We propose a fully automated DNN optimization toolchain for PPG-based BP estimation: first, we leverage a *gradient-based* **Neural Architecture Search (NAS)** to automatically select DNN layers from a pre-defined

pool and optimize network depth. Second, we apply *structured pruning* to eliminate redundant portions of the convolutional and fully connected layers. Lastly, we apply *mixed precision quantization*, obtaining integer deployable models that utilize `int2`, `int4`, `int8` data types for both weights and activation, further reducing memory occupation.

— On four different open-source datasets [2, 17, 37, 43], our integrated pipeline produces models with up to 7.99% lower error compared to the best DL SotA with a 7.5× parameter reduction, or with up to 83× fewer parameters with negligible accuracy loss.

— We deploy Pareto-optimal models, i.e., models that balance model size and estimation error optimally, among those found with our optimizations on a SotA multi-core RISC-V ultra-low-power **System-on-Chip (SoC)**, GAP8 [62], exploiting an open-source deployment tool, DORY [12]. On the BCG Dataset [17], i.e., the one with the most samples per patient among the four considered, all our deployed models occupy less than 55 kB, while reaching an average latency per inference of 142.14 ms (min: 52.87, max: 241.33) and a corresponding energy consumption of 7.25 mJ (min: 2.70, max: 12.31).

— As a further experiment, we investigate *patient-specific fine-tuning* to improve the accuracy of our DNNs. Using a consistent and leakage-free split strategy, fine-tuning yields accuracy improvements of up to 61.1% for SBP and 64.27% for DBP. Although the study cohort ($n = 40$) is smaller than the 85-subject minimum required by the AAMI protocol [66], a widely used validation standard for BP measurement devices that prescribes both a minimum cohort size and acceptance thresholds to assess agreement with reference measurements, our most accurate models satisfy its core accuracy criterion (mean error ≤ 5 mmHg and STD ≤ 8 mmHg). Notably, the best-performing of the three deployed models achieved a Mean Error of 1.39 mmHg with a STD of 2.36 mmHg on DBP.

2 Background and Related Works

The study of BP monitoring solutions based on wearable devices equipped with PPG sensors has attracted significant research interest from both academia and industry in recent years.

Figure 1 compares the PPG and ABP waveforms over a single cardiac cycle. In both signals, when the left ventricle contracts and pumps blood into the arteries (Systole), the waveforms rise and reach the SBP, corresponding to the maximum arterial pressure. After this push, the aortic valve closes and the heart relaxes; the pressure then slowly drops as blood keeps flowing through the arteries to the rest of the body. The lowest point before the next beat is the DBP. A small dip called the dicrotic notch can be observed around the moment the aortic valve closes, marking the transition from Systole to Diastole. This shared morphology helps explain why PPG can contain informative cues for BP estimation [48]. However, practical BP estimation from PPG in daily wearable use remains challenging, because the optical signal is sensitive to various types of artifacts due to noise, changes at the skin-sensor interface, and so on. [7].

Among the proposed solutions, two main modeling approaches have emerged: signal-to-label models and signal-to-signal models. The first directly estimates discrete SBP and DBP values from segments of the PPG signal. In contrast, signal-to-signal models reconstruct the continuous ABP waveform from the PPG signal. The values of systolic and DBPs are then measured from the peaks and valleys of such a reconstructed signal.

Researchers typically measure BP estimation accuracy using the **Mean Absolute Error (MAE)** for both SBP and DBP, calculated as $MAE_{SBP} = \mathbb{E}(|SBP_{true} - SBP_{pred}|)$ and $MAE_{DBP} = \mathbb{E}(|DBP_{true} - DBP_{pred}|)$ respectively, where $\mathbb{E}()$ represents the average over all predictions in the test set. Recognizing the importance of standardized validation, the IEEE established guidelines in 2014 for wearable cuff-less BP devices, which mandate reporting MAE as a key performance metric in validation results [1].

Tables 1 and 2 summarize the major recent approaches for PPG-based BP estimation, focusing on ML and DL methods. A key difficulty when analyzing SotA methods for this task lies in the fact that many of them use custom pre-processing and dataset preparation techniques (input filtering, data splitting, etc.), often unrealistic in practice

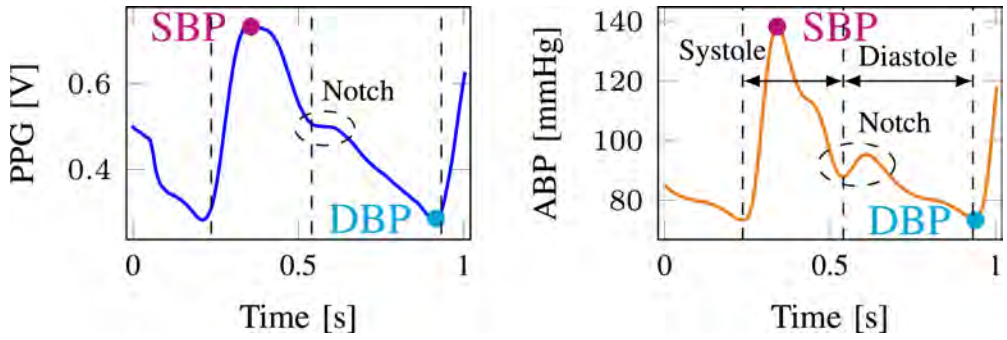

Fig. 1. SBP and DBP estimation from PPG (left) and ABP (right) signals.

(e.g., filtering out particularly noisy records without a well-grounded biological "threshold") or introducing various degrees of information leakage. This makes a direct numerical comparison between MAE results of little meaning. Therefore, we categorize related works into two main groups: (i) *Custom Pre-processing*, which includes works that employ specific, non-standard data pipelines; and (ii) *Benchmark Pre-processing*, which includes methods, such as ours, that follow the standardized and real-world plausible data preparation and splitting protocol proposed in [27]. The two categories are discussed in Sections 2.1 and 2.2, respectively. Another limitation of most of the available works is that they do not analyze the performance of their algorithms once deployed on wearable-class hardware, thus making their real-world usability without cloud support questionable. Some of the few related works that investigate this aspect are thus discussed in Section 2.3.

2.1 Custom Pre-processing

The authors of [38] addressed BP estimation using the publicly available MIMIC-II dataset [57]. After filtering out invalid records (e.g., short duration), they retained 3,663 segments from approximately 1,000 unique patients. Their pre-processing involved wavelet transformation and signal zeroing to mitigate noise and artifacts. The processed data were evaluated using classical ML models, with AdaBoost achieving the best performance, yielding MAEs of 5.35 and 11.17 for DBP and SBP, respectively. In 2021, [2] introduced a new dataset composed of recordings from 1,195 **Intensive Care Unit (ICU)** patients. Their pre-processing involved filtering techniques and min-max normalization. The authors proposed a DL architecture consisting of an encoder, a decoder, and attention modules. The model achieved MAEs of 6.57 (DBP) and 14.39 (SBP). Cheng et al. [18] proposed a novel neural network architecture inspired by Wave-U-Net [60], which processes the PPG signal along with its first and second derivatives (VPPG and APPG, respectively). Using the MIMIC-II dataset [57], cleaned by removing short and unreliable signals, they obtained 277,050 segments from approximately 1,620 patients. The model achieved MAEs of 3.73 (DBP) and 6.41 (SBP). In 2023, [21] conducted a comprehensive comparison of classical algorithms (e.g., XGBoost, LightGBM, and CatBoost) and DL models (e.g., Residual U-Net, ResNet-18, and ResNet-LSTM) on both MIMIC-II and MIMIC-III datasets [34, 57]. Pre-processing varied by dataset: for MIMIC-II, a fourth-order band-pass filter was applied to the PPG signal, whereas for MIMIC-III, pre-processing involved removing ABP values smaller than 20 mmHg and higher than 300 mmHg, along with the same band-pass filter. Classical models outperformed DL methods, with XGBoost achieving the lowest MAEs on MIMIC-III (DBP and SBP), and on MIMIC-II (DBP), while CatBoost performed best on MIMIC-II (SBP). More recently, [63] proposed the Parallel Convolution Transformer Network, which integrates convolutional and transformer layers to capture both local and global signal features. Evaluated on the MIMIC-III dataset, the model outperformed previous methods, achieving MAEs of 2.36 (DBP) and 4.44 (SBP), setting a new SotA on this dataset. However, this performance is conditioned by the fact that authors discard low-quality segments, therefore not being in the same testing condition of competitors.

Table 1. SotA Table of Methods with Custom Data Pre-processing

Work	Dataset	Architecture	Pre-processing	DBP MAE	SBP MAE
Kachuee et al. [38]	MIMIC II	Linear Regression Decision Tree SVR AdaBoost	Re-sampling 1 kHz, Wavelet decomposition, Zeroing [0, 0.25] Hz, Zeroing [250, 500] Hz, Wavelet denoising, Wavelet reconstruction	6.74 7.75 5.91 5.35	14.71 16.28 12.26 11.17
Aguirre et al. [2]	Sensors	Encoder, Decoder, and Attention	Null data and saturated points detection, Butterworth Filter [0.5, 8] Hz, Min-Max Normalization, band-pass Butterworth filter [0.5, 45] Hz	6.57	14.39
Cheng et al. [18]	MIMIC-II	ABP-Net	Removal of short and unreliable signals	3.73	6.41
Costa et al. [21]	MIMIC-II	XGBoost LightGBM CatBoost Residual U-Net ResNet 18 ResNet LSTM	Cleaning and removal of dirty samples, PPG passed through a fourth-order Chebyshev II band-pass filter from 0.5 to 10 Hz	7.14 7.21 7.38 8.18 8.42 7.96	16.67 16.49 16.43 19.1 19.85 17.71
	MIMIC-III	XGBoost LightGBM CatBoost Residual U-Net ResNet 18 ResNet LSTM	Cleaning of missing signals and setting of upper and lower bounds for the ABP (300 and 20 mmHg)	8.45 8.59 8.60 10.94 10.38 10.47	16.18 16.28 16.32 18.60 18.16 17.56
Tian et al. [63]	MIMIC-III	Parallel Convolutional Transformer Network	Cleaning unpaired PPG and ABP data, removal of data with poor signal quality, band-pass filter with frequency range of [0.5, 10] Hz	2.36[a]	4.44[a]

[a]Low-quality segments discarded and not tested.

However, due to the diversity in data processing and validation methods, comparing these results remains difficult. Moreover, many of these models are tested on only a single dataset, making their generalizability questionable.

2.2 Standardized Pre-processing

To address this issue, in 2023, the authors of [27] proposed a standardized pre-processing pipeline for four datasets (Sensors [2], UCI [37], BCG [17], and PPGBP [43]). This pipeline significantly reduces result variability caused by data leakage and inconsistent pre -processing. Their methodology includes:

—Alignment of PPG and ABP signals using maximum cross-correlation.
—Segmentation into non-overlapping 5-second windows.
—Removal of distorted ABP segments based on physiological plausibility (e.g., amplitudes between 30 and 220 mmHg, pulse pressure over 10 mmHg, resting HR between 35 and 140 BPM).
—Extraction of SBP and DBP labels from ABP for signal-to-value models via the median of systolic peaks and cardiac cycle boundaries (refer to Figure 1, right, for a visual representation).
—Removal of distorted PPG segments based on peak/valley STD and baseline correction via cubic spline interpolation.

Table 2. SotA Table of Methods with Standardized Pre-processing from [27]

Work	Dataset	Architecture	Pre-processing	DBP MAE	SBP MAE
Gonzalez et al. (Benchmark) [27]	Sensors	ResNet	Standard benchmark pre-processing (no fine-tuning)	8.33	17.46
	UCI			8.30	16.59
	BCG			7.76	12.20
	PPGBP			**8.61**	**13.62**
	Sensors	U-Net		7.66	15.64
	UCI			7.88	16.93
	BCG			7.98	12.30
	PPGBP			*n/a*	*n/a*
Lim et al. [44]	PPGBP	Conv-Transformer	Standard benchmark pre-processing (no fine-tuning)	9.17	14.82
This article, 2025	Sensors	ResNet	Standard benchmark pre-processing (no fine-tuning)	7.83	17.01
	UCI			**7.69**	17.09
	BCG			7.50	11.51
	PPGBP			8.68	13.88
	Sensors	U-Net		**7.51**	**15.51**
	UCI			7.81	**16.32**
	BCG			**7.26**	**11.11**
	PPGBP			*n/a*[a]	*n/a*[a]

The bold numbers represent the most accurate models on the four datasets on both DBP and SBP.
[a]U-Net, which is a signal-to-signal model, cannot be trained on PPGBP because the dataset just contains the scalar values of SBP and DBP.

In Table 2, we report the results of the two best-performing DL models (on average) from [27], which are those that we used as input for our optimization pipeline (i.e., ResNet and U-Net). In [44], similarly to [63], the authors introduced a hybrid architecture combining convolutional and transformer layers designed to extract both local and global features. They tested their model on PPGBP using the standardized benchmark pipeline. Despite the innovative architecture, their model underperformed compared to a standard ResNet of [27], achieving MAEs of 9.17 vs. 8.61 (DBP) and 14.82 vs. 13.62 (SBP), respectively.

To ensure fair and reproducible comparisons, our study adopts the standardized benchmark pre-processing introduced in [27]. This guarantees data leakage-free evaluation across subjects and aligns with best practices in the field, while providing a robust SotA comparison. In particular, rather than performing standard sample-level random splits (which could place measurements from the same subject in both training and test partitions), we enforce subject-wise partitioning so that all samples from a given subject are assigned to a single fold/set.

Even though several works in Table 1 report promising performance (e.g., [63]), we do not include them in our comparison for two reasons: (i) their custom pre-processing pipelines hinder a fair evaluation; in particular, the commonly adopted random train/test split can place samples from the same subject in both sets, leading to overly optimistic metrics; and (ii) they rely on operators that are too computationally demanding to be efficiently deployed on wearables (e.g., Transformers [64]). In contrast, our pipeline (architecture exploration, structured pruning, and integer-only quantization) systematically optimizes edge-friendly CNNs using a robust and leakage-free data pipeline. Within that experimental setup, it not only results in models with a smaller memory footprint with respect to previous DNN-based solutions, but also superior accuracy.

2.3 Embedded Solutions

While pre-processing and data splitting play a crucial role in ensuring a fair comparison across methods and measuring accuracy realistically, an equally important requirement for real-world applicability is deployment on wearable device-class hardware.

In [68], the authors provide a comprehensive analysis and comparison of approaches for embedded BP estimation. They review a broad range of deployment platforms, spanning commercial processors (CPUs and GPUs), general-purpose microcontrollers, and custom hardware solutions such as FPGAs [59] and ASICs [36, 67]. Although FPGA- and ASIC-based implementations can be extremely efficient, we do not consider them in this study because they are arguably less applicable to real-world products (unless with huge market volumes), due to their high design costs, especially for ASICs, and limited flexibility. In fact, those approaches usually implement a *fixed algorithm* or DNN structure, and require an external host for control, reconfiguration (if any), sensor data reading and transmission, and so on. In contrast, we focus our analysis on solutions deployed on *programmable* embedded devices, describing below some of the most notable previous works. While we also report their MAE results, besides deployment metrics, we remark that none of them uses the same data splitting and pre-processing setup of [27], thus error values are not directly comparable with ours.

In [45], the authors propose a simple two-layer ANN that takes the raw PPG waveform as input and directly estimates BP. The model is deployed on an ultra-low-power EFM32 Leopard Gecko microcontroller (ARM Cortex-M3, 32-bit). While the reported energy per inference is very low (about 2.1 mJ), the inference latency is still relatively high (about 42 ms) considering the simplicity of the network. The memory consumption is reported to be less than 25 kB. On MIMIC-III, they obtained an MAE of 3.42 and 1.92 for SBP and DBP, respectively. In [54], the authors introduce BP-Net, a neural architecture derived from U-Net [55]. They deploy the model on a Raspberry Pi 4 with 4 GB of RAM and report an inference time of 42.5 ms, comparable to [45] despite the substantially more complex architecture and much more capable hardware. Although memory and power are not reported, a conservative upper bound for the latter can be derived from the platform supply rating (≈15 W), which would result in a very high energy consumption per inference of ≈0.64 J. The new architecture was evaluated on MIMIC-II obtaining an MAE of 5.16 and 2.89 on SBP and DBP, respectively. In [61], the authors evaluate several deep networks (including a 0.30 MB ResNet) across multiple edge devices, such as the Raspberry Pi 3 Model B and the ESP32-WROVER-IE. For ResNet on the Raspberry Pi, they report an inference time of 186 ms. Regarding energy, the same considerations done for the previous paper apply (power values are not reported either). On a more energy-constrained platform, the ESP32, the inference time increases dramatically to 60.82 seconds, which effectively rules out real-time use. Evaluated on the MIMIC-IV dataset, their ResNet20 achieved an MAE of around 12.1 and around 5.9 for SBP and DBP, respectively. In [56], the authors deploy a Temporal Convolutional Network that uses PPG features as input on a Raspberry Pi Zero W. They report an inference time of about 2.5 seconds for a 3-second PPG window. Power and energy metrics are again not reported, while memory is reported to be around 32 kB. On MIMIC-II/III, the authors achieved an MAE of 2.38 (SBP) and 1.23 (DBP). Finally, in [69], the authors deploy a substantially more complex model (Sample Convolution and Interaction Network, and Gated Time Convolutional Network) on an NVIDIA Jetson Nano. The inference time is around 2 seconds, and no memory consumption is reported; although power is also not reported, the Jetson Nano is far more power-hungry than the ultra-low-power embedded targets considered in this work. On the MIMIC-III dataset, they obtained a MAE of 7.44 (SBP) and 5.70 (DBP).

3 Materials and Methods

The main goal of this article is to define a complete, multi-stage DNN optimization pipeline for PPG-based BP estimation, targeted at obtaining models that are not only accurate, but also compact, thus amenable for deployment on wearables. To this end, we extended the open-source DNN optimization library PLiNIO [33], which offers a user-friendly interface for implementing a variety of model selection and compression algorithms. While the individual algorithms used in this work were already included in the PLiNIO package, in this work we: (i) integrated them into a coherent pipeline and (ii) extended them to be applicable onto SotA DNN architectures for BP estimation, such as U-Net-like autoencoders. This required multiple modifications to the PLiNIO inner workings and the addition of support for new layer types.

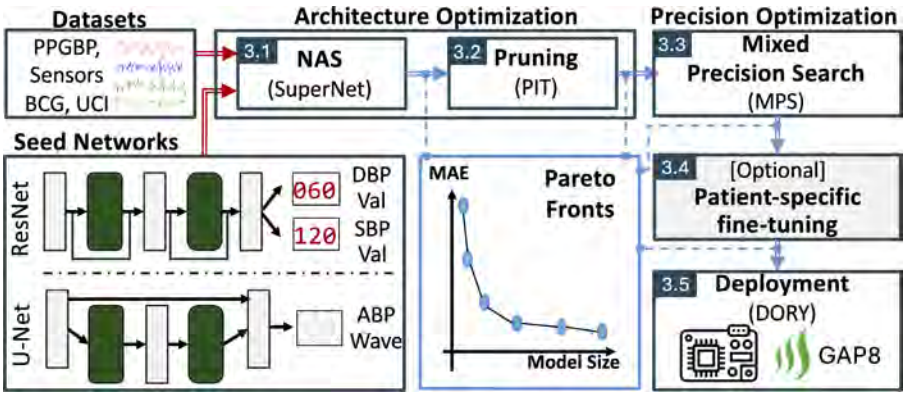

Fig. 2. Overview of the proposed automated DNN optimization flow. Starting from a dataset and a baseline model, the pipeline performs NAS, followed by structured pruning and mixed-precision search (quantization). Before deployment, selected candidate models can optionally undergo patient-specific fine-tuning using data from an individual subject to further improve accuracy.

Our pipeline, depicted in Figure 2, is composed of three model optimization steps: NAS (*SuperNet*), structured pruning (*Pruning-In-Time*), and **Mixed-Precision Search (MPS)**. We (optionally) apply a subject-specific fine-tuning step to further improve the accuracy of our optimized models, followed by an automatic Python-to-C compilation and deployment on the target SoC.

The optimization process takes as input a training dataset, pre-processed and segmented into windows as described in [27], alongside a *seed* network. This seed, representing the initial DNN architecture, acts as a template from which optimized models are derived. We adopt as seeds the two top-performing DNNs identified in [27]. Both are 1D CNNs; however, they differ in their design and target output. The first is a signal-to-value model inspired by ResNet [53], which directly predicts the average SBP and DBP values within each input window. The second model follows a U-Net-like [55] signal-to-signal structure, estimating the entire ABP waveform from which SBP and DBP values are later extracted. For more detailed information on these seed architectures, we refer the reader to [27].

Importantly, while these models were previously optimized in [27] for maximum predictive accuracy, our work introduces a new dimension: *hardware cost-aware* optimization. We demonstrate that this approach enables the discovery of models with significantly reduced size and computational demands. For each combination of seed network and dataset, our optimization pipeline follows three core stages, described in the next sections.

3.1 SuperNet NAS

The first step of our pipeline consists of applying a gradient-based NAS, whose working principle, shown in Figure 3, is inspired by DARTS [46]. Compared to the latter, SuperNet supports hardware-aware optimization, considering various cost models (e.g., model size, number of operations) and is able to generate multiple architectures, thereby exploring the accuracy-cost tradeoff more thoroughly. The NAS builds a so-called SuperNet, replacing every convolutional layer l_i in the seed architecture with a *set* S_i *of alternative layers* $l_{i,j}$, all of which receive the same input. Here, i indexes the position of the original layer in the seed network (i.e., the ith convolution being replaced), while j indexes a specific candidate operation among the alternatives available for that layer (i.e., the jth option in the set S_i). In our work, each standard convolution from the two seeds is replaced with a set of alternatives chosen specifically to reduce the models' footprint toward wearable deployment, including: a standard 1D convolution (C), an identity layer (ID), and a depthwise-separable convolutional module (DW). The latter is composed of a depthwise layer followed by a pointwise convolution, and it is commonly used in

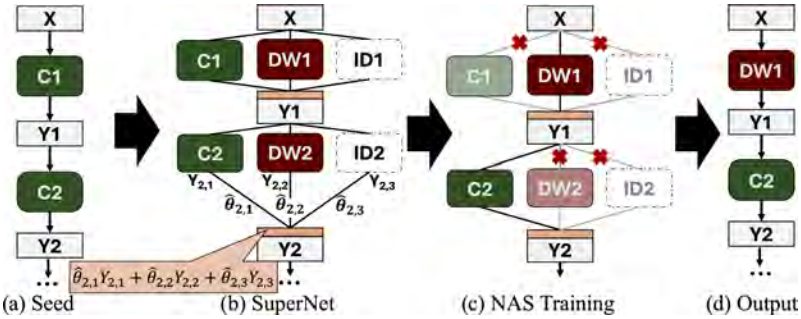

Fig. 3. SuperNet-based NAS. From left to right, the figure reports the initial condition of the network (a), the different layer options (b), the selection process during training (c), and finally the output model (d).

efficient CNN models due to its reduced parameters and FLOPs count w.r.t. standard convolution [29]. The Identity operation, instead, is added only when the input and output of the original layer have the same shape; it allows the NAS to skip some convolutions, thus optimizing the network depth. Each alternative $l_{i,j}$ in the set has a corresponding trainable parameter $\theta_{i,j}$. The output of the set is computed as a weighted sum of the various alternatives' outputs, where the weight associated with each alternative $l_{i,j}$ depends on a softmax-ed version of the corresponding $\theta_{i,j}$ as depicted in Figure 3(b).

Formally, let $S_i = \{l_{i,1}, l_{i,2}, \ldots, l_{i,n}\}$ be the set of alternative paths replacing a single original layer. Given the layer's input X_i, the output Y_i of the resulting SuperNet layer is computed as shown in Equation (1):

$$Y_i = \sum_{j=1}^{||S_i||} \hat{\theta}_{i,j} \cdot Y_{i,j} \qquad \text{with } \hat{\theta}_{i,j} = \sigma_j(\theta_{i,j}), \tag{1}$$

where $\sigma()$ is the softmax operation and $Y_{i,j} = l_{i,j}(X_i)$ is the output of layer $l_{i,j}$.

A NAS-optimized architecture is obtained, intuitively, by choosing for each layer a single path, i.e., setting one of the parameters $\hat{\theta}_i$ to 1 and the other to 0. SuperNet uses gradient descent to solve a continuous relaxation of this assignment problem, jointly optimizing both the weights W and the added θs in a standard training loop.

This training uses the modified loss function shown in Equation (2), where the standard task loss $\mathcal{L}$, computed on the model's prediction and on the ground truth BP values, is augmented by a cost-based regularization term $\mathcal{R}$. In our case $\mathcal{L}$ is the MSE of the SBP and DBP predictions for ResNet, or of the whole reconstructed signal for U-Net:

$$\mathcal{L}(W, \theta) = \mathcal{L}(W, \theta) + \lambda \mathcal{R}(\theta). \tag{2}$$

$\mathcal{R}$, defined in Equation (3), estimates the expected cost of the architecture by weighting the cost of each alternative layer:

$$\mathcal{R}(\theta) = \sum_{i} \sum_{j=1}^{||S_i||} \hat{\theta}_{i,j} \cdot Cost(l_{i,j}). \tag{3}$$

$\text{Cost}(l_{i,j})$ represents a pre-defined cost metric (e.g., the number of parameters associated with each alternative), and λ is a hyperparameter that balances accuracy and efficiency. In our setup, we use the parameter count as a cost metric to favor architectures suitable for deployment on memory-constrained wearable platforms. When the training is finished, the optimized architecture is obtained, keeping the path associated with the largest θ_i, discarding the others.

Algorithm 1 reports a high-level overview of the **Differential NAS (DNAS)** training procedure. It is composed of three phases: the SuperNet is initially pre-trained with all paths equally contributing to the output; then, the

Algorithm 1: Training-time Optimization Procedure

for each $l_i \in$ SuperNet **do**
  for $l_{i,j} \in S_i$ **do**
    Add $\theta_{i,j} \leftarrow 1/||S_i||$ ▷ Initialization
  end for
end for
Freeze all θ
for $i \leftarrow 1, \ldots,$ Epochs$_{pt}$ **do**
  Update W based on $\mathcal{L}(W)$ ▷ Pre-training (phase 1)
end for
Unfreeze all θ
while not converged **do**
  Update W, θ based on $\mathcal{L}(W, \theta) + \lambda\mathcal{R}(\theta)$ ▷ DNAS (phase 2)
end while
Extract selected path based on $\hat{\theta}$
for $i \leftarrow 1, \ldots,$ Epochs$_{ft}$ **do**
  Update W based on $\mathcal{L}(W)$ ▷ Fine-tuning (phase 3)
end for

actual DNAS optimization is started; lastly, the model can be fine-tuned after selecting the best single path. This general scheme also applies to the following optimizations (pruning and mixed precision quantization), which similarly employ a gradient-based approach. However, in later pipeline steps some of the phases may become redundant and can be skipped. For instance, an already fine-tuned model coming from the SuperNet optimization does not require additional pre-training (phase 1), and can be directly pruned (phase 2). The same happens when applying MPS to an already optimized network. For further details about the DNAS training procedure and all other algorithms employed in this work we refer the reader to [33].

3.2 Structured Pruning

Starting from the optimized network architectures found by the NAS, the second optimization step consists of applying structured pruning to further reduce their parameter count (and indirectly, their computational complexity) while preserving accuracy.

We employed the **Pruning-in-Time (PIT)** method from PLiNIO, which prunes convolutional and linear layers by removing slices of the weight tensors (and corresponding activations) over multiple axes, with the effect of: (i) reducing the number of output channels/neurons; (ii) reducing the filter size (i.e., the receptive field) or (iii) increasing the dilation. Thus, this step implements a finer-grain neural network optimization compared to the *NAS*, tuning each layer's geometrical hyperparameters rather than selecting among pre-defined alternatives, possibly improving the Pareto front found in the previous step. *PIT* works by adding trainable binary masks to a model, that selectively eliminate slices of weights and activation tensors. These masks are once again optimized jointly with the network weights via gradient descent, using the loss formulation of Equation (2). After training, pruned parts are permanently removed. Figure 4 shows an example of the pruning method when applied to *output channels*. As shown, a trainable vector of binary mask parameters θ_i is added to each layer, where each $\theta_{i,j}$ determines whether an output channel j is kept (setting $\theta_{i,j} = 1$) or removed ($\theta_{i,j} = 0$). The masks are applied to the weights W_i of layer l_i as follows:

$$W_i' = H(\theta_i) \circ W_i,$$

where $H(\theta_i)$ is the Heaviside step function and $\circ$ denotes the Hadamard product. Since the Heaviside function is non-differentiable, the **Straight-Through Estimator (STE)** is used to allow gradients to flow through θ_i during back-propagation.

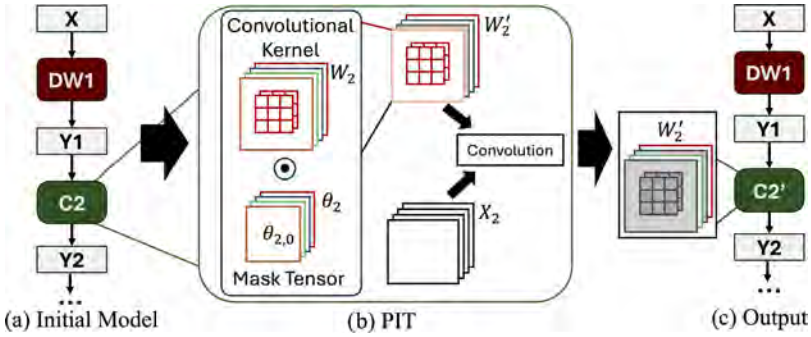

Fig. 4. PIT overview: starting from the initial layer sequence (a), PIT injects a trainable mask tensor θ_2 to gate slices of the convolutional kernel W_2 during training (b), yielding a pruned kernel W_2' and a reduced layer $C2'$ in the final network (c). PIT, Pruning-in-Time.

As anticipated in Section 3.1, PIT's optimization is carried out following a scheme similar to the one of Algorithm 1, albeit without the initial pre-training phase. Moreover, PLiNIO's original PIT algorithm had to be extended to support SotA BP prediction networks. The main changes were required to add support for pruning some previously unsupported layer types, such as **Parametric ReLUs (PReLUs)**, Instance Normalization, feature concatenation, and grouped convolutions. In the latter, pruning output channels independently as done normally by PIT would lead to an irregular result, with different groups including a different number of filters. Such a layer would be harder to accelerate, and incompatible with standard DNN frameworks (e.g., PyTorch). Therefore, we added an extra preparation step to ensure that *channels are pruned uniformly across groups*, by sharing the masks relative to corresponding channels within different groups (e.g., the first output channels generated by group 1, group 2, and so on, all share a single θ_i).

3.3 Quantization and MPS

After the network architectures have been optimized through NAS and further refined by PIT, the final stage before deployment is integer *quantization*. This step further reduces the memory footprint and computational requirements of the models by representing weights and activations as low-precision integer values, while also enabling efficient deployment on hardware platforms that do not include a **Floating Point Unit (FPU)**, such as our target, GAP8 [62], and many other low-power wearable-class devices. PLiNIO supports MPS, which assigns independent precision levels to weights (W) and activations (X) in Convolutional and Fully Connected layers. Inspired by [16], this method can be configured to select from a set of pre-defined bit-widths $p \in P$, e.g., $P = \{2, 4, 8\}$.

MPS uses so-called "fake-quantization" during the training/optimization phase, i.e., it uses floating point values internally to allow small gradient-based updates, but simulates an affine quantization process by means of scaling, offsets, and rounding [33]. The non-differentiable rounding function is approximated by means of STE during back-propagation. More specifically, as shown in Figure 5, the activations X_i and the weights W_i tensors of each layer, undergo fake-quantization for all bit-widths in the set P. Then, similarly to the various layer alternatives in *SuperNet*, the different quantized variants are linearly combined using trainable parameter vectors $\theta_{i,w}$ (for weights) and $\theta_{i,x}$, which have a length of $\|P\|$ and are normalized via a SoftMax function (σ). The resulting effective tensor is computed as:

$$\hat{T}_i = \sum_{p \in P} \hat{\theta}_{i,t,p} \cdot T_{i,p} \qquad \text{with } \hat{\theta}_{i,t,p} = \sigma_p(\theta_{i,t,p}), \tag{4}$$

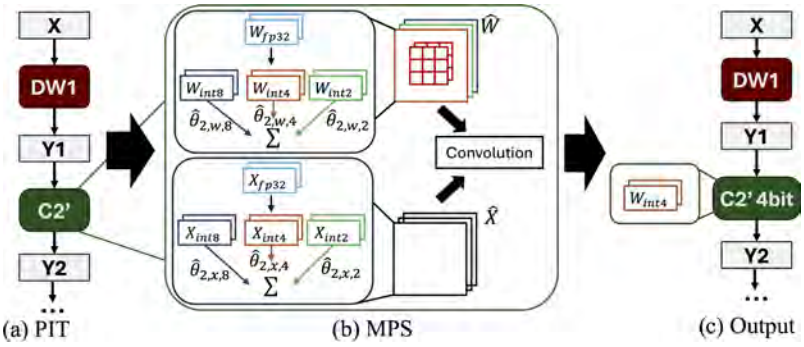

Fig. 5. MPS. The figure shows the selection of the bit-width of each convolutional layer, which leverages a SuperNet-like approach. Supported bit-width are currently 2, 4, and 8.

where T (and subscript t) are used to refer to a generic tensor which could be W (subscript w) or X (subscript x) and $T_{i,p}$ is the p-bit quantized version of T_i. A higher value of $\theta_{i,t,p}$ increases the contribution of the corresponding bit-width, making $\hat{T}_i$ more closely resemble the p-bit quantized tensor. The effective tensors $\hat{W}$ and $\hat{X}$ derived in this way are then used to compute the layer's output, e.g.:

$$Y = \text{Conv}(\hat{X}, \hat{W}). \tag{5}$$

A key advantage of this approach is that all fake-quantized versions originate from a single floating-point tensor, significantly reducing memory overhead during training.

As with other PLiNIO methods, this modified DNN is trained in a DNAS-like optimization loop, where both W and θ are updated jointly to minimize Equation (2).

In this work, precision search has been applied only to weights, fixing activations to eight bits, because our main objective was to compress the model weights so that they could fit the target platform (GAP8) on-board memory. Therefore, the optimization cost term ($\mathcal{R}$) in Equation (2) in this case is set to the total number of *bits* required by the network, as a function of each $\theta_{i,w}$.

3.4 Subject-specific Fine-tuning

Several previous studies have shown that, due to physiological differences between subjects, ML models trained on biosignals for BP estimation may perform better when exposed to data from the test subject during training [18, 21]. For this reason, it is common practice among commercial devices, especially those used for medical purposes to perform a calibration or fine-tuning operation tailored for each subject, e.g., using data from a sphygmomanometer as ground truth, which may be repeated periodically [3].

To this end, subject-specific fine-tuning has been integrated as the final pipeline stage to specifically address the challenges inherent to PPG-based BP estimation. Specifically, we propose a novel fine-tuning procedure that follows strict constraints to avoid data leakages. By accommodating individual cardiovascular characteristics, this fine-tuning procedure enables the resulting models to satisfy the stringent accuracy thresholds mandated for clinical-grade diagnostic devices.

Namely, let a dataset consist of N patients, each denoted by $p_i \in \mathcal{P}$, where $i = 1, \ldots, N$. Each patient p_i has M_{p_i} samples. In all our experiments, we initially train our models using a five-fold cross-validation strategy, consistent with the methodology used in [27], where the splitting unit is the patient. In each fold, we generate a training set $\mathcal{P}_{\text{train}}$ and a test set $\mathcal{P}_{\text{test}}$, such that $\mathcal{P}_{\text{train}} \cup \mathcal{P}_{\text{test}} = \mathcal{P}$ and $\mathcal{P}_{\text{train}} \cap \mathcal{P}_{\text{test}} = \emptyset$. Thus, models at this stage have not been trained on any sample from the subject under test.

Table 3. Subject-specific Fine-tuning Protocol

Step	Description
1	Train the model on all samples from patients in $\mathcal{P}_{\text{train}}$.
2	Evaluate on 20% of $\mathcal{P}_{\text{test}}$ to obtain baseline accuracy.
3	Fine-tune the model on the remaining samples of $\mathcal{P}_{\text{test}}$.
4	Evaluate again on 20% of $\mathcal{P}_{\text{test}}$ to assess the effect of fine-tuning.

When performing experiments that require a direct comparison with solutions trained following the protocol of [27], we simply evaluate trained models on all M_{p_i} samples from $\mathcal{P}_{\text{test}}$. Specifically, $|\mathcal{P}_{\text{train}}| = \frac{4N}{5}$ and $|\mathcal{P}_{\text{test}}| = \frac{N}{5}$, supposing a five-fold cross-validation. When assessing the benefits of subject-specific fine-tuning, the model is first evaluated on a subset of 20% of the test subjects' samples to establish baseline performance. The remaining samples are used to fine-tune the model. Then, a final evaluation is again conducted on the same 20% subset, allowing us to measure the improvement attributable to the fine-tuning step. Table 3 summarizes the steps of the fine-tuning protocol.

It is worth noting that our optimization pipeline (i.e., NAS, Pruning, Quantization) is applied to the SotA ResNet and U-Net backbones with the goal of obtaining more accurate and lightweight models that generalize across every subject, i.e., without requiring calibration. For this reason, the architectural parameters θ are selected on a validation set that is strictly disjoint from the subjects in $\mathcal{P}_{\text{test}}$, ensuring that model selection remains free of subject-specific leakage. Subject-specific fine-tuning is then studied separately, on a per-subject basis, starting from the model instances that achieve the best performance under the subject-independent evaluation protocol. This paradigm ensures that the generated optimized architectures possess sufficient flexibility for subject-specific fine-tuning on unseen data after they reached good performances across patients, thereby avoiding the risk of overfitting scarce data from individual subjects. In real-world scenarios, our pipeline enables BP monitoring device providers to maintain a single high-quality and efficient model that can be fine-tuned on limited user-specific data without requiring individual alterations to the architecture nor the deployed model code.

3.5 Hardware Target and Deployment

We export our optimized DNNs into a custom ONNX-compatible format, which can be directly processed by the open-source deployment framework DORY [12] for execution on multi-core RISC-V platforms. DORY automatically generates highly optimized C inference code, taking care of memory allocation, **Direct Memory Access (DMA)** transfer scheduling, and the invocation of optimized AI kernels. To maximize efficiency on our target platform, DORY applies graph-level optimizations, tiling strategies, and memory management techniques that exploit the hardware hierarchy, with the goal of minimizing latency and maximizing resource utilization.

For the implementation of DNN layers, we employ two distinct backend libraries of manually optimized DNN primitives in C for our target: PULP-NN [25] for fully `int8` quantized networks (a specific case explored in our MPS search), and PULP-NN-Mixed [10] for mixed-precision execution. The reason why we consider the full `int8` case separately is that this precision is natively supported by the four-way, **Multiply and Accumulate (MAC)-capable** Single Instruction Multiple Data Arithmetic Logic Units of our hardware target. Sub-byte operations, instead, require extra unpacking/packing operations onto/from 8-bit values [10].

Therefore, while MPS models including <8-bit tensors are surely beneficial in terms of memory compression, latency- and energy-wise their benefit is less certain. Indeed, smaller tensor sizes reduce the cost of memory transfer operations and allow fitting bigger portions of a layer in faster and closer memories, but this advantage can be reduced or nullified by the (un)packing overheads. Accordingly, we consider both the case in which MPS

is allowed to select any precision in $P = \{2, 4, 8\}$ and the one in which it is constrained to $P = \{8\}$, thus becoming effectively equivalent to a standard **Quantization-Aware Training (QAT)**.

3.5.1 Hardware Platform and Measurements. Our deployment target is the GreenWaves GAP8 SoC [62], a low-power RISC-V-based multi-core processor specifically designed for edge signal processing. GAP8 features a cluster of eight general-purpose cores for parallel execution of compute-intensive workloads, a two-level scratchpad memory with 512 kB of main memory for code and weight storage, and a 64 kB last-level cache with single-clock access latency for the cluster. Data transfers between memory levels are managed by an integrated DMA engine.

We use the GAPuino evaluation board as the deployment platform, connected to the Nordic Power Profiler Kit II [58] for accurate power and energy measurements. Execution latency is measured through the GAP8 internal performance counters, while the external profiler provides fine-grained insight into the energy consumption of the deployed models.

We note that, while GAP8 is a highly competitive ultra-low-power processor, featuring dedicated hardware acceleration capabilities for parallel, data-intensive workloads such as neural network inference, the methodology described in the previous sections is not tied to this specific target. The generated models are entirely architecture agnostic and could be easily deployed on alternative embedded architectures, albeit clearly with different results in terms of latency and energy consumption.

4 Results

4.1 Setup

Some architectural modifications were required to ensure compatibility of our seed DNNs with *full-integer* quantization and deployment, necessary to obtain good latency and energy performance on our target: Batch and Instance Normalization layers were repositioned immediately after convolutional layers, enabling their parameters to be folded into convolutional weights; similarly, PReLU activations were kept for SuperNet and PIT, but then replaced with standard ReLUs for the quantization step, to ensure compatibility with DORY [12]. Additionally, zero-padding shortcuts in ResNet-based seeds were replaced with learnable 1×1 convolutions with independent normalization. This increases the number of parameters of the seed but also allows PIT to explore a larger optimization space.

For the SuperNet NAS, we use two Adam optimizers: one for the network weights $\mathcal{W}$ (learning rate 0.001) and one for the architecture parameters θ (learning rate 0.01). Network weights were optimized on the training set, while θ on the validation set, following the DARTS approach [46]. For each dataset and for both seeds, we explored 18 values of the regularization parameter λ, logarithmically spaced between 10^{-11} and 10^{-7}, using the total parameter count as the optimization cost $\mathcal{R}$. NAS training consisted of 20 warm-up epochs, followed by 200 epochs of architecture search and an additional 200 epochs of fine-tuning. Early stopping with a patience of 40 epochs and checkpointing based on the minimum MSE loss were applied throughout.

From the NAS results, we selected for PIT-based structured pruning the models achieving the lowest SBP and DBP errors, as well as the corresponding seed models, even when they did not attain the lowest SBP/DBP errors. PIT training followed the same protocol as SuperNet with 200 epochs of pruning optimization, and 200 epochs of fine-tuning with early stopping. We also explored the same regularization strength values λ as for the NAS.

In the final stage, we applied QAT (at fixed `int8` precision for both weights and activations) and MPS (assigning layer-wise weights bit-widths between `int2`, `int4`, and `int8`). As for the previous stage, we took as input both the seeds and the models with the lowest errors from the combined SuperNet and PIT Pareto fronts. Additionally, only for the BCG dataset, we also applied QAT/MPS to the Pareto-optimal model with the fewest parameters. The reason for this difference is that this is the dataset on which we measure deployment results (see Section 4.8), thus we wanted to also include an example of a much smaller (yet slightly less accurate) model. We applied a standard min–max affine quantization scheme for weights and signed Parametrized Clipping Activation [20]

for activations, with 32-bit accumulations and biases, as supported by our target inference library [25]. MPS training employed the same protocol as the previous stages: a differentiable search phase where $\mathcal{W}$ and θ were optimized jointly (with $\hat{\theta}$ annealed via a SoftMax temperature τ initialized at 5 and decayed by $e^{-0.0045}$ per epoch [50]), and a final fine-tuning phase where θ was frozen and discrete precision selections were applied. The parameter τ controls the smoothness of the sampling distribution. The annealing schedule is designed so that the bit-width selection parameters sampling increasingly resemble an argmax operation. Such a procedure improves training stability and prevents degenerative conditions in which the θ parameters assume uniform values. We performed nine experiments per seed with logarithmically spaced values of λ to balance the tradeoff between model size and MSE loss. Strong values of λ can bring to aggressively quantized networks (i.e., employing many 2-bit and 4-bit weights). The output of such networks is a noisy reconstructed signal, which can hinder the peak localization algorithm necessary to estimate BP values. To mitigate this problem, a smoothing filter was added to the final layer before extracting systolic and DBP values from reconstructed arterial waveforms. This fifth-order Butterworth low-pass filter employs an adaptive cutoff frequency defined as a proportional coefficient of the absolute magnitude of the input signal. This proportionality coefficient and the filter order were determined based on empirical validation, selecting the configuration exhibiting maximum reliability across all folds. Importantly, the subsequent stages of the evaluation pipeline, including the peak-detection algorithm and the outlier-rejection controls, remained unmodified to ensure a fair and consistent comparison with all previous 32-bit floating-point models.

It is worth noting that the choice of the models produced by each optimization step, which are used as input for the subsequent ones, can be considered as an additional degree of freedom. We opted to use the most accurate points and the seeds, in order to primarily focus on obtaining accurate (yet compressed) DNNs; however, an alternative selection (e.g., taking multiple Pareto-optimal models from each curve) is also possible, with an obvious difference in the global optimization time.

4.2 Datasets

We employ the same four datasets used in the most recent and comprehensive benchmarking study on PPG-based BP estimation [27]. For details on each dataset and descriptive statistics, please refer to the original papers.

4.2.1 Sensors. The Sensors [2] dataset is a subset of the MIMIC-III database, containing records from 1,195 ICU patients. It includes demographic data along with PPG and ABP waveforms, collected using the Philips CareVue Clinical Information System and iMDsoft MetaVision ICU [31]. The dataset is the second largest in the set after UCI, with a total measurement duration of approximately 15 hours. Each record consists of two 15-second signal segments, spaced 5 minutes apart.

4.2.2 UCI. The UCI [37] dataset, also known as the Cuff-Less Blood Pressure Estimation Dataset, is a subset of the MIMIC-II Waveform Database. Although MIMIC-II and MIMIC-III originate from the same underlying measurements, collected under identical conditions, in the same hospitals, and using the same devices, the UCI and SENSORS datasets are distinct and should not share records. However, UCI does not include subject information, making it impossible to check for data leakage across subjects. Nevertheless, as the largest among the four, it is of particular interest for DL methods.

4.2.3 BCG. The BCG [17] dataset is a bed-based ballistocardiography dataset collected from 40 patients, four of whom had preexisting heart conditions, while the rest were healthy. Data collection was conducted under Kansas State University IRB protocol #9386, using the Finapres Medical Systems Finometer PRO for continuous brachial BP measurement and the GE Datex CardioCap 5 for PPG. The dataset is resampled from 1,000 Hz to 125 Hz, and BP signals are rescaled by a factor of 100 mmHg/Volt. With a total of approximately 4 hours of recorded measurements, BCG is smaller than the UCI and SENSORS datasets. However, it has a notably high ratio of segments per subject, though its limited number of participants results in lower overall data variability.

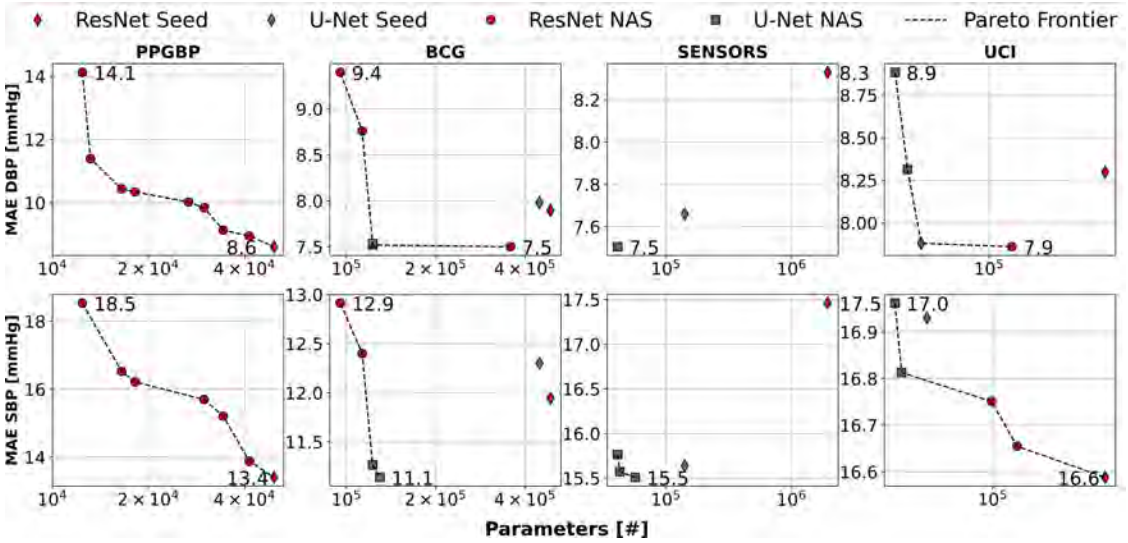

Fig. 6. NAS results on all datasets on DBP (top row) and SBP prediction (bottom row).

4.2.4 PPGBP. PPGBP [43] dataset is the smallest among the four, totaling less than an hour of recordings. However, it includes 219 subjects, all with various CVDs such as hypertension and diabetes. BP measurements were taken using the Omron HEM-7201 device, providing only SBP and DBP discrete values. Following a 10-minute rest period, each subject underwent a single BP measurement, followed by three 2.1-second PPG recordings using the SEP9AF-2 device. The original 1,000 Hz signals were resampled to 125 Hz. With only three segments per subject, PPGBP has a notably low segment-to-subject ratio but exhibits high data variability relative to its small size.

4.3 NAS Pareto Analysis

Figure 6 depicts the results of NAS on all four datasets for both DBP and SBP prediction models. All results are reported in a MAE vs. model size (number of parameters) plane. Red and green diamonds correspond to seed DNNs (ResNet and U-Net from [27], respectively). Correspondingly colored circles and squares are the Pareto-optimal architectures found with NAS (the non-Pareto-optimal ones are omitted for visual clarity).

All results at this stage use floating-point weights and activations. While all our models simultaneously predict DBP and SBP, we report separate Pareto fronts for the two quantities for easier visualization.

On all datasets, we obtain models that either dominate the seeds or are on the memory vs. error Pareto front. Namely, on PPGBP, we obtain a rich curve of Pareto architectures starting from the ResNet. We are able to reduce the seed size by 1.19×, with a small increase in the MAE of only 3.9% and 3.5% on DBP and SBP prediction, respectively. As mentioned earlier, U-Net-derived models are not applied to this dataset, since the full BP signal ground truth is not available.

On BCG, we Pareto-dominate both seed networks, improving both MAE and size. Our most accurate U-Net-derived model obtains 11.139 mmHg MAE on SBP prediction and 7.52 mmHg MAE on DBP, being 6.7%/4.7% better than the best seed (ResNet). Simultaneously, this network reduces the total number of parameters by 3.8×.

Similarly, on the two largest datasets, thanks to our NAS, we are again able to obtain Pareto-dominant solutions. On Sensors, our U-Net-derived architectures reduce the size of the most accurate seed (U-Net) by 2.5×, while achieving a similar or lower MAE of 7.51 mmHg/15.51 mmHg on DBP/SBP, respectively. The situation reverses in UCI, where ResNet-derived DNNs achieve the best performance. The most accurate NAS output networks require

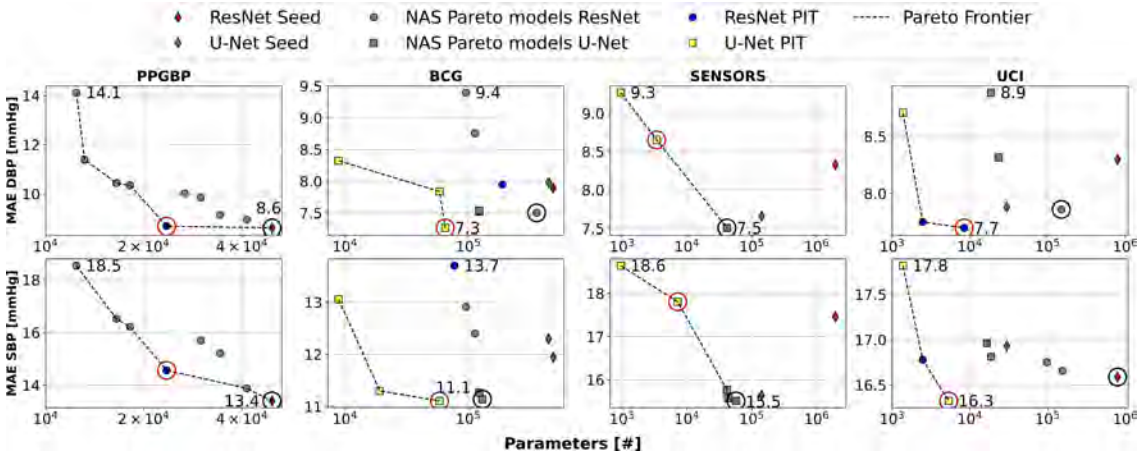

Fig. 7. Pruning results on all datasets on DBP (top row) and SBP (bottom row) prediction. Black circles are PIT seeds, red circles are the models discussed in Section 4.4.

only 149.8k/156.3k parameters to achieve a close-to-optimal MAE of 16.65 mmHg on SBP estimation, and the lowest overall MAE (7.86 mmHg) on DBP estimation. While the seed ResNet is able to achieve an even lower MAE on SBP, with its 792k parameters, it would be impossible to deploy on GAP8's internal memory of 512 KB, even when quantized.

Interestingly, on BCG and Sensors, U-Net-based architectures outperform ResNets. We attribute this behavior to the ability of this network topology to learn faster from a lower amount of data, thanks to the richer training signal provided by the full-time series reconstruction task.

Lastly, we note that, already after the NAS stage, most of the obtained models have memory footprints compatible with the constraints of our target platform (512 kB on-chip memory) [62]. As mentioned, this is the primary goal to ensure deployability. However, the following sections demonstrate how the combined application of pruning and quantization further improves the Pareto front, yielding smaller models of similar accuracy, which would not only fit in even tighter memory constraints, but also execute faster and more efficiently, thanks to the reduced data movement and FLOPs, and to the integer data format better suited to the FPU-less GAP8.

4.4 Pruning Pareto Analysis

Figure 7 shows that structured pruning further improves the Pareto front on all datasets, and generates new most-accurate models on two of them. Starting from the previously obtained NAS Pareto front, we apply PIT to the black circled models. Figure 7 includes all points already shown in Figure 6 (the seeds in red and green and the NAS Pareto-front in gray). Additionally, if Pareto-optimal, it also includes (in yellow and blue) the new results obtained by PIT. As before, intermediate (non-Pareto-optimal) pruning candidates are omitted for clarity. The models circled in red are the ones that will be discussed in detail in this section.

On PPGBP, PIT generates a new Pareto-optimal model starting from the seed ResNet. Notably, the latter still achieves the lowest MAE (8.61 mmHg vs. 8.68 mmHg), but at the cost of a 2.1× larger memory footprint with respect to the pruned model, outlined in red in the top-left chart of Figure 7.

On BCG, the new PIT models dominate all previous results, completely redefining the Pareto front. The most accurate models (red circles) achieve a 7.99% and 7.01% decrease in DBP and SBP MAE, respectively, with a 7.5× and 8.4× reduction in the number of parameters compared to the ResNet seed. Alternatively, a 30.3× (13.3×) decrease in model size is achieved while still improving the SBP (DBP) MAE by 1.31% (1.77%).

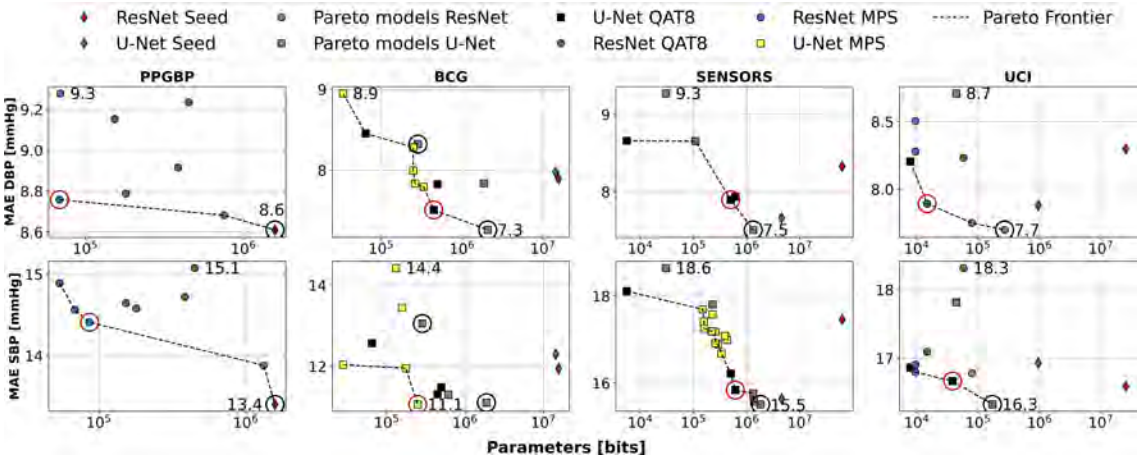

Fig. 8. Results of MPS on all datasets on DBP (top row) and SBP (bottom row) prediction. Black circles are MPS seeds, red circles are the models discussed in Section 4.5.

On Sensors, PIT created several models with drastically reduced memory footprint but also slightly larger errors. Namely, the most accurate PIT results, circled in red achieved an 88.3× (70.9×) parameters reduction with an increase in SBP (DBP) MAE of 1.73% (1.47%) compared to the ResNet seed and 14.53% (12.6%) with respect to the best NAS output.

On UCI, the biggest and most complex dataset, PIT achieved good accuracy, completely redefining the Pareto front and generating new most accurate models on both SBP and DBP. The most accurate PIT model marked in red in the lower rightmost square, derived from U-Net, achieved a 1.59% error reduction with 149× fewer weights on SBP compared to the previously most accurate ResNet seed. Similarly, on DBP, the most accurate PIT model (a ResNet) achieves a 2.3% error reduction with 94× fewer parameters compared to the most accurate NAS output.

4.5 Quantization and MPS Pareto Analysis

Figure 8 shows how the Pareto front further changes when applying the last step of our optimization chain, i.e., QAT/MPS. In this case, gray points represent the combined Pareto front of the NAS and pruning phases. As before, QAT/MPS have been applied to the models circled in black. Those circled in red, instead, are the ones discussed in the rest of this section in detail. For clearer visualization of the new results, some of the least accurate models generated by the previous steps (NAS and pruning) have been omitted from the figure. The x-axis reports the total number of *bits* required by each model instead of the number of parameters, to meaningfully compare networks using different data precision. Quantization further advances the Pareto Front, creating models with slightly higher errors but a way smaller memory footprint.

On PPGBP, the most accurate quantized models show a 6.18% and 8.25% higher errors for DBP and SBP, respectively, but requires 9.99× less memory bits, compared to the original ResNet. MPS models improve significantly the left part of the Pareto front, yielding better accuracies compared to previous NAS+pruning outputs of similar size. On BCG, our quantization algorithms were applied to U-Net models already optimized with both NAS and pruning. The most accurate DBP model, red-circled in the second top square of Figure 8, obtains a 6.2× parameter reduction at the cost of a 6.95% higher error compared to the most accurate previous network. Most notably, on SBP, MPS yields a model with both 0.39% lower error and a 7.5× smaller footprint than previous results. The lower error is probably due to the regularization effect of quantization on such a small architecture.

On SENSORS, two U-Nets generated by NAS and one generated by PIT underwent quantization. The most accurate results were obtained by 8-bit models (the black squares circled in red in the SENSORS plots), e.g., 5.2%

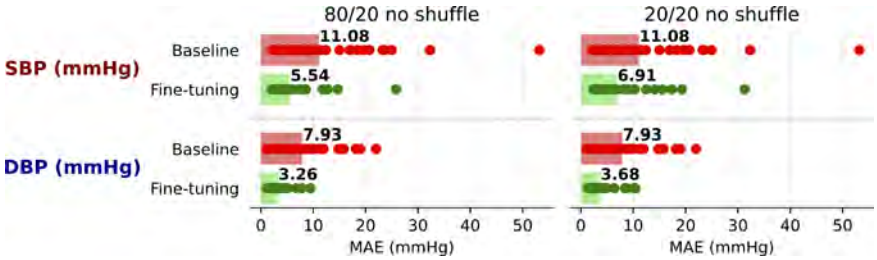

Fig. 9. MAE comparison pre- and post-fine-tuning, without data shuffling, using 80% of the samples as a training set and 20% test set (left), or 20% as training set and 20% as a test set (right).

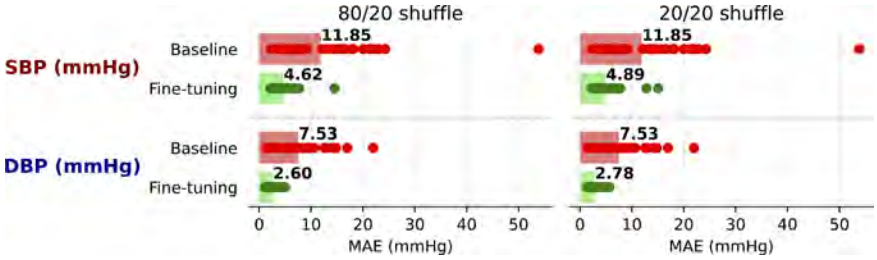

Fig. 10. MAE comparison pre- and post-fine-tuning, with data shuffling, using 80% of the samples as a training set and 20% test set (left), or 20% as training set and 20% as test set (right).

and 2.13% higher DBP and SBP MAE, respectively, in exchange for a 4× memory reduction compared to the previous best model.

On the biggest dataset, UCI, we quantized both ResNet and U-Net-based architectures, already optimized by the previous algorithms. The most accurate quantized model for DBP prediction is an 8-bit precision ResNet (red-circled green dot), which is 5.37× smaller and achieves a 2.52% higher error, compared to the most accurate model overall. With regard to SBP, instead, the most accurate model is U-Net-based, 8-bit quantized, and reaches a 2.1% higher MAE than the best float solution, while requiring 4.44× fewer bits.

In conclusion, although the quantized models expectedly did not achieve substantially lower errors compared to floating point models, they provide significant memory reductions, and consequently energy savings once deployed, often with a negligible compromise in accuracy.

4.6 Subject-specific Fine-tuning

The key dataset requirements to assess the benefits of subject-specific fine-tuning are the availability of subject-level annotations and a sufficiently large number of samples per patient to enable both fine-tuning and evaluation. Based on this observation, we perform our fine-tuning experiments on BCG, since UCI does not provide subject identity metadata, and PPGBP and Sensors have only three and nine samples per subject, respectively (versus approximately 75 in BCG).

We selected three quantized models optimized by our cross-subject pipeline (see previous sections), namely the best performing quantized models for SBP and DBP prediction, and the one with the least number of parameters. Figures 9 and 10 show the MAE distribution before and after fine-tuning for one of these models (the one with the lowest DBP error). The other configurations show similar results, and their global MAE improvements after fine-tuning are reported in Table 5. Each dot in the graphs represents the MAE on one subject, while bars are the global averages. The difference between the two graphs is that Figure 9 uses a *contiguous chunk* of the samples as a fine-tuning set (i.e., without shuffling P_{test}). We keep samples in order to maximize fairness, under the assumption

that samples are temporally ordered in the dataset. Therefore, this split mechanism corresponds to evaluating the fine-tuned model at a separate time with respect to its training. However, since the temporal ordering of samples is not clearly specified in the BCG metadata, we also experiment with shuffling P_{test} (Figure 10). In both figures, the leftmost graph refers to a setup that uses the entire set of samples for the subject under consideration, splitting them into 80% for training and 20% for testing (either temporally, in Figure 9, or randomly, in Figure 10). Additionally, to assess the effectiveness of fine-tuning in a small-data setup, we repeat the experiment keeping the same test samples, but reducing the training set to a (disjoint) set of the same size, i.e., 20% of the total, corresponding to 15 samples per subject on average. This experiment is useful to validate whether models could be effectively personalized through a reasonably short calibration session.

In all four cases, fine-tuning demonstrates a pronounced positive effect on accuracy, providing a consistent reduction in MAE for both SBP and DBP. As expected, the lowest error is achieved when fine-tuning on the whole 80% training set, both with and without shuffling. Compared to the baseline, fine-tuning without shuffling reduces the MAE by 50% for SBP and by 58.9% for DBP. More specifically, the MAE reduces for 36 out of the 40 subjects in both SBP and DBP. For the latter, the 36 improving subjects show an average MAE reduction of 5.51 mmHg, whereas the remaining four experience an average increase of just 1.16 mmHg. Similarly, for SBP, the average MAE improvement is 8.10 mmHg, while the average degradation is just 0.62 mmHg. Shuffling data prior to fine-tuning results in even greater improvements, with global MAE reductions of 61.01% for SBP and 64.27% for DBP.

Additionally, the right side of the two figures shows that even fine-tuning on a much smaller dataset yields consistent improvements. The SBP (DBP) MAE is reduced by 37.6% (53.6%) in the no-shuffling case, and by 58.7% (63.1%) in the shuffling case. This demonstrates that competitive performance can be achieved even in tightly data-constrained fine-tuning setups.

Lastly, we note that no statistically significant difference was observed in the performance gains achieved through fine-tuning between the cohort of 36 healthy patients and the subset of four patients with CVDs within the BCG dataset.

These findings support the importance of subject-specific fine-tuning in improving BP prediction accuracy.

4.7 Comparison with the SotA

Table 4 compares the results of our best performing models, without per-subject fine-tuning, against the best-performing solutions (either classical models or DNNs) from the SotA, limited to those using the standardized pre-processing introduced in [27] for fairness of comparison.

Results show that, on three out of four datasets, our automated DNN optimization pipeline yields a model with lower error than the SotA. The only exception is PPGBP, where classical ML models (SVR and RF) outperform our optimized DNNs in both performance and size. Namely, the most accurate of the two, i.e., the SVR, achieves an MAE of 8.04 mmHg and 13.15 mmHg in DBP and SBP estimation, respectively. In comparison, our most accurate DNNs obtain 8.61 and 13.40, i.e., a 7% and 2% increase. The SVRs are also more parameter-efficient, being 2.42× (2.93×) smaller than our most accurate DNNs for SBP (DBP). We impute this result to the small size of this dataset, which favors classical models over DNNs.

On BCG, classical models remain competitive, although our DNNs obtain slightly lower error, namely 3.7% (1.1%) for SBP (DBP), at the cost of a significant size increase for SBP (5.44×) and a moderate one for DBP (1.16×). However, it shall be remarked that [27] trains *separate* SVR and RF models for SBP and DBP prediction. Therefore, when considering a full ABP monitoring system that predicts both values, those approaches would require two separate models, which makes our DNNs both more accurate and more compact, as detailed in Section 4.8. Additionally, compared to the best performing DNNs for SBP (V-Net), our model achieves a slight accuracy improvement, with a huge parameter reduction of 7.5×.

Table 4. Comparison with SotA Methods from [27] and [44]

Model	SBP		DBP	
	MAE (mmHg)	Params (#)	MAE (mmHg)	Params (#)
PPGBP				
RF	13.17	20.3k	8.12	20.4k
SVR	**13.15**	29.0k	**8.04**	16.9k
Conv-Transformer [44]	14.82	-	9.17	-
Ours	13.40	49.3k	8.61	49.3k
BCG				
RF	12.88	0.21k	7.89	85.3k
SVR	11.45	10.7k	7.34	55.9k
V-Net	11.42	491k	8.01	491k
Ours	**11.07**	58.2k	**7.26**	64.8k
Sensors				
RF	15.86	64.0k	7.66	171k
SVR	15.60	775k	**7.50**	416k
PPGBIABP	16.45	92.5M	7.99	92.5M
Ours	**15.51**	56.8k	**7.50**	41.2k
UCI				
RF	16.85	21.3k	8.25	4.26k
SVR	17.45	18.5M	8.07	4.54M
PPGBIABP	17.06	296k	8.07	296k
Ours	**16.32**	5.32k	**7.69**	8.43k

The bold numbers represent the most accurate models on the four datasets on both DBP and SBP.

On Sensors, classical ML methods, that were shown to outperform the original U-Net and ResNet seeds in [27], are instead surpassed by our NAS-optimized DNNs. Namely, SVRs, which achieved the best results on both metrics, are now matched by our U-Net NAS models in terms of error (0.6% lower for SBP and similar for DBP), while our models also require 13.7× (10.1×) fewer parameters. Comparing our best model with one of the best DNNs (excluding ResNet and U-Net, which were previously considered), PPGBIAP, we can improve the performance in both metrics with an impressive 2,250× parameter reduction.

Lastly, on UCI, the dataset with the largest number of samples, our models strongly outperform both the SotA DNNs and classic methods, achieving 6.5% (4.7%) lower MAE on SBP (DBP) and reducing the required number of parameters by a striking 3,500× (540×). In fact, the higher complexity of this dataset causes the number of parameters of both the SVR (with RBF kernel) and the RF to increase exponentially, further advocating for a DNN-based approach.

4.8 Network Deployment

Figure 8 shows that Pareto-optimal (quantized) models for all datasets have a limited memory footprint, lower than the 512 kB available on-board our target platform (GAP8). Notably, this means that all of them could theoretically be deployed.

However, in this section, we focus on deployment results for models trained on BCG, i.e., the same dataset on which we experimented with subject-specific fine-tuning (see Section 4.6). Namely, we deploy three neural networks optimized with our proposed pipeline, as well as three models from the SotA [27]. We select the

Table 5. Deployment Results on GAP8

Network	MACs (M)	Cycles	MACs/Cycles	DBP MAE	SBP MAE	Latency (ms)	Energy (mJ)	Memory (KByte)
RF (DBP + SBP)	n.a.	366.6k + 1.2M	n.a.	7.89	12.88	3.7 + 12.2	0.19 + 0.62	85.3 + 0.21
SVR (DBP + SBP)	n.a.	764.6k + 1.4M	n.a.	**7.34**	11.45	7.64 + 13.89	0.39 + 0.71	55.9 + 10.7
VNet	64.4	6.7M	9.55	8.01	11.42	67.49	3.44	491
Best DBP	47.09	13.2M	3.56	7.51 (2.68)[a]	11.31 (4.62)[a]	132.24	6.74	54.35
Best SBP	48.59	24.1M	2.01	8.28 (2.80)[a]	**11.06** (4.70)[a]	241.33	12.31	30.26
Best Params	11.61	5.3 M	2.20	9.32 (7.72)[a]	12.25 (11.06)[a]	52.87	2.70	2.15

[a]Values in parentheses come from fine-tuning and are computed on a slightly different set of data. For further details on the fine-tuning protocol, please refer to Section 3.4.
The bold numbers represent the most accurate models on the BCG dataset on both DBP and SBP.

same (best) baselines reported in Table 4 for BCG, which include two classic ML algorithms, an RF and an SVR, and a signal-to-signal VNet DNN. For what concerns our results, we select the two models with the lowest SBP and DBP MAE (among quantized ones) and, in addition to that, the smallest Pareto-optimal model overall.

Table 5 reports the inference costs of these models, in particular: the memory footprint, the total number of MAC operations, and the latency and energy consumption per input window. Results refer to GAP8 operating at 100 MHz. Latency values are read from performance counters, while energy is estimated considering an average active power consumption of 51 mW. We also report the DBP and SBP MAEs before and after subject-specific fine-tuning. It's worth noting how the best SBP model before fine-tuning has a slightly higher SBP MAE than the best DBP model (4.70 against 4.62). That could be explained by the smaller size of the former, which makes it less capable of improvement with subject-specific fine-tuning. The "Best SBP" model is also a mixed precision one, while "Best DBP" uses all 8-bit weights, which explains why it is faster to execute despite a larger memory occupation.

As anticipated in Section 4.7, classical models (SVR and RF) are *separately trained* for SBP and DBP prediction. Thus, a complete ABP monitoring system based on these approaches would require the deployment of two separate models. Accordingly, we report in Table 5 the memory, energy, and latency results for both sub-models, which must be summed to obtain the totals (assuming sequential execution). Conversely, the VNet, like our models, can be used to predict both SBP and DBP.

The results show that our optimized models outperform all baselines in terms of memory occupation versus accuracy tradeoff. Even the largest one occupies merely 54.4 kB, and the smallest one remarkably only requires 2.15 kB of memory, while still achieving decent MAE results of 9.32 (7.72) for DBP and 12.25 (11.06) for SBP before and after fine-tuning. In comparison, the RF and SVR require a total of 66.6 kB and 85.5 kB, respectively. The VNet requires even more space (491 kB), exceeding the GAP8 L2's 512 kB (which also contains code segments, temporary data, etc.), and requiring the use of external memory, connected via SPI, for storing weights.

In terms of latency and energy consumption, classic ML models are superior to DNNs, as expected, given the nature and number of the involved operations per prediction (e.g., the RF is approximately 3.35× faster and more efficient than our smallest DNN). Our results are instead comparable to those of VNet, which achieves slightly lower latency with respect to our most accurate models thanks to the usage of normal convolutions, as opposed to harder to accelerate (but more parameter-efficient) depthwise layers. Importantly, however, all our models largely meet real-time performance constraints, as the inference latency ranges between 52.87 and 241.33 ms, thus being significantly lower than the time duration of a BCG input window, i.e., 5 seconds.

While not more efficient than all baselines, our networks still allow days of continuous monitoring on a wearable without recharging. In fact, considering their consumption, which ranges between 12.31 and 2.7 mJ per inference, and assuming a typical wearable battery with 500 mAh capacity @3.7V, our models would allow roughly

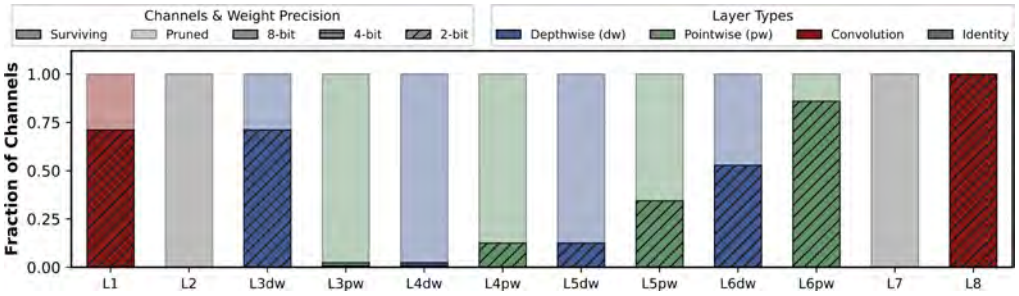

Fig. 11. Visualization of the choices made by the pipeline during the optimization of the "Best Params" deployed model.

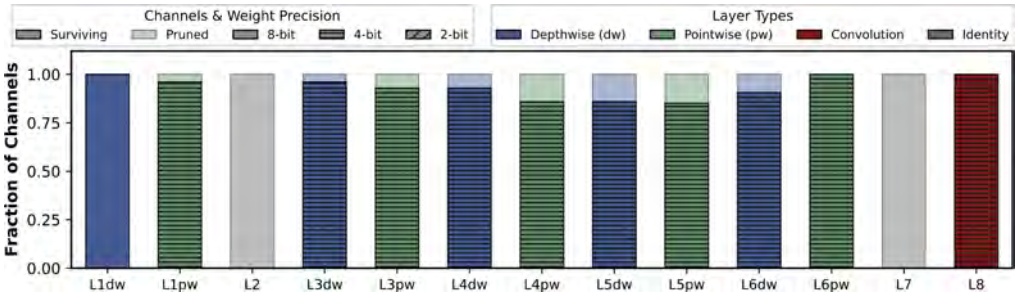

Fig. 12. Visualization of the choices made by the pipeline during the optimization of the "Best SBP" deployed model.

between 500k and 2.5M inferences on a single charge, translating to 30 and 150 days of continuous monitoring. These simple calculations only consider processing energy, which is clearly an approximation. However, note that our DNN's energy consumption results are lower than those reported in [15] where a full-system analysis is performed for a similar PPG-based application.

4.9 Architectural Analysis

Figures 11 and 12 provide a layer-wise view of two of the three deployed models, *Best Params* and *Best SBP* from Table 5. We omit *Best DBP* since it is a fully-8-bit DNN, thus not allowing an analysis of the MPS results. In each plot, every bar corresponds to one network layer: the *color* encodes the operator selected by SuperNet, the *height* indicates the fraction of channels retained after PIT pruning, and the *hatching* denotes the weight bit-width assigned by MPS. Across both deployed networks, SuperNet shows a strong preference for *depthwise-separable convolutions*, suggesting that this operator is particularly well matched to PPG processing within our search-space constraints, while not being present in the baseline architectures. Moreover, several layers near the input and output are replaced by *identity* mappings (i.e., fully bypassed), indicating that a shallower encoder/decoder is sufficient to capture and reconstruct the relevant information from PPG signals, and that additional depth in these regions would be largely redundant. The pruning behavior learned by PIT differs markedly between the two deployments, consistently reflecting the cost term controlled by λ. In *Best Params* (Figure 11), optimized with a larger λ to emphasize efficiency, PIT performs substantial channel reduction across most layers. In contrast, *Best SBP* (Figure 12), optimized to prioritize estimation accuracy, retains a considerably larger portion of channels overall. Beyond this global tradeoff, the pruning decisions reveal a consistent *layer-wise* pattern in both models: intermediate layers are pruned more aggressively, whereas early and late layers preserve higher channel counts. This suggests that (i) early layers are more critical for robust feature extraction from raw PPG and (ii) later layers are more sensitive due to their role in reconstruction/prediction, while mid-network representations contain

more removable redundancy. Finally, MPS assigns mixed-precision in a largely even manner across layers, while still differentiating the two operating points. The *Best Params* model is dominated by *2-bit* weight assignments, enabling more extreme compression under the efficiency-driven objective. Conversely, the *Best SBP* model uses *4-bit* precision in several layers, preserving accuracy while still providing significant quantization benefits.

5 Conclusion

Continuous BP monitoring is crucial for the early detection and prevention of CVDs. While most research in this field has focused on maximizing monitoring accuracy, an equally important but often overlooked aspect is the ability to deploy these models on ultra-low-power SoCs suitable for wearable applications, thus enabling low-cost continuous monitoring. In this work, we presented a fully automated pipeline for the optimization of DNNs for PPG-based BP prediction consisting of chained NAS, pruning, and mixed-precision-search steps. Each stage contributes to improving either the accuracy or the efficiency of the models, thus moving the Pareto front between performance and resource usage. From this process, we selected a subset of optimized models for deployment on the GAP8 ultra-low-power SoC, and evaluated them in terms of inference latency, energy consumption, and memory footprint, showing that our models enable multiple weeks of real-time continuous monitoring on battery-powered wearables. Additionally, we investigated patient-specific fine-tuning, an approach increasingly adopted by companies developing wearable devices, to further enhance model accuracy for individual users. Future work will explore adaptive on-device learning and multi-sensor data fusion to further improve long-term personalization and robustness in dynamic real-world conditions, for example, by tackling motion artifacts.

References

[1] IEEE Standard Association. 2014. IEEE standard for wearable cuffless blood pressure measuring devices. In *IEEE Std 1708-2014*, IEEE, 1–38. DOI: https://doi.org/10.1109/IEEESTD.2014.6882122

[2] Nicolas Aguirre, Edith Grall-Maës, Leandro J. Cymberknop, and Ricardo L. Armentano. 2021. Blood pressure morphology assessment from photoplethysmogram and demographic information using deep learning with attention mechanism. *Sensors* 21, 6 (2021), 2167. DOI: https://doi.org/10.3390/s21062167

[3] Jérémy Alexandre, Kevin Tan, Tiago P. Almeida, Josep Sola, Bruce S. Alpert, and Jay Shah. 2023. Validation of the Aktiia blood pressure cuff for clinical use according to the ANSI/AAMI/ISO 81060-2:2013 protocol. *Blood Pressure Monitoring* 28, 2 (2023), 109–102. DOI: https://doi.org/10.1097/MBP.0000000000000639

[4] Noor Faris Ali and Mohamed Atef. 2023. An efficient hybrid LSTM-ANN joint classification-regression model for PPG based blood pressure monitoring. *Biomedical Signal Processing and Control* 84, 7 (2023), 104782. DOI: https://doi.org/10.1016/J.BSPC.2023.104782

[5] Tiago P. Almeida, Meritxell Cortés, David Perruchoud, Jérémy Alexandre, Pascale Vermare, Josep Sola, Jay Shah, Luisa Marques, and Cyril Pellaton. 2023. Aktiia cuffless blood pressure monitor yields equivalent daytime blood pressure measurements compared to a 24-h ambulatory blood pressure monitor: Preliminary results from a prospective single-center study. *Hypertension Research* 46, 6 (Jun. 2023), 1456–1461. DOI: https://doi.org/10.1038/s41440-023-01258-2

[6] Amir Arjomand, Amin Boudesh, Farnoush Bayatmakou, Kenneth B. Kent, and Arash Mohammadi. 2025. TransfoRhythm A transformer architecture conductive to blood pressure estimation via solo PPG signal capturing. arXiv:2404.15352. Retrieved from https://arxiv.org/abs/2404.15352

[7] Win-Ken Beh, Yu-Chia Yang, Yi-Cheng Lo, Yun-Chieh Lee, and An-Yeu(Andy) Wu. 2023. Machine-aided PPG signal quality assessment (SQA) for multi-mode physiological signal monitoring. *ACM Transactions on Computing for Healthcare* 4, 2, Article 14 (Apr. 2023) 20 pages. DOI: https://doi.org/10.1145/3587256

[8] Luca Benfenati, Sofia Belloni, Alessio Burrello, Panagiotis Kasnesis, Xiaying Wang, Luca Benini, Massimo Poncino, Enrico Macii, and Daniele Jahier Pagliari. 2024. EnhancePPG: Improving PPG-based heart rate estimation with self-supervision and augmentation. arXiv:2412.17860. Retrieved from https://arxiv.org/abs/2412.17860

[9] Dayi Bian, Pooja Mehta, and Nandakumar Selvaraj. 2020. Respiratory rate estimation using PPG: A deep learning approach. In *Proceedings of the 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC '20)*, 5948–5952. DOI: https://doi.org/10.1109/EMBC44109.2020.9176231

[10] Nazareno Bruschi, Angelo Garofalo, Francesco Conti, Giuseppe Tagliavini, Davide Rossi. 2020. Enabling mixed-precision quantized neural networks in extreme-edge devices. In *Proceedings of the 17th ACM International Conference on Computing Frontiers (CF '20)*. ACM, New York, NY, 217–220. DOI: https://doi.org/10.1145/3387902.3394038

[11] Alessio Burrello, Francesco Carlucci, Giovanni Pollo, Xiaying Wang, Massimo Poncino, Enrico Macii, Luca Benini, and Daniele Jahier Pagliari. 2024. Optimization and deployment of deep neural networks for PPG-based blood pressure estimation targeting low-power wearables. In *Proceedings of the IEEE Biomedical Circuits and Systems Conference (BioCAS '24)*, 1–5. DOI: https://doi.org/10.1109/BioCAS61083.2024.10798404

[12] Alessio Burrello, Angelo Garofalo, Nazareno Bruschi, Giuseppe Tagliavini, Davide Rossi, and Francesco Conti. 2020. DORY: Automatic end-to-end deployment of real-world DNNs on low-cost IoT MCUs. arXiv:2008.07127. Retrieved from https://arxiv.org/abs/2008.07127

[13] Alessio Burrello, Francesco Bianco Morghet, Moritz Scherer, Simone Benatti, Luca Benini, Enrico Macii, Massimo Poncino, and Daniele Jahier Pagliari. 2022. Bioformers: Embedding transformers for ultra-low power sEMG-based gesture recognition. arXiv:2203.12932. Retrieved from https://arxiv.org/abs/2203.12932

[14] Alessio Burrello, Daniele Jahier Pagliari, Pierangelo Maria Rapa, Matilde Semilia, Matteo Risso, Tommaso Polonelli, Massimo Poncino, Luca Benini, and Simone Benatti. 2022. Embedding temporal convolutional networks for energy-efficient PPG-based heart rate monitoring. *ACM Transactions on Computing for Healthcare* 3, 2, Article 19 (Mar. 2022), 25 pages. DOI: https://doi.org/10.1145/3487910

[15] Alessio Burrello, Daniele Jahier Pagliari, Matteo Risso, Simone Benatti, Enrico Macii, Luca Benini, and Massimo Poncino. 2021. Q-PPG: Energy-efficient PPG-based heart rate monitoring on wearable devices. *IEEE Transactions on Biomedical Circuits and Systems* 15, 6 (2021), 1196–1209. DOI: https://doi.org/10.1109/TBCAS.2021.3122017

[16] Zhaowei Cai and Nuno Vasconcelos. 2020. Rethinking differentiable search for mixed-precision neural networks. arXiv:2004.05795. Retrieved from https://arxiv.org/abs/2004.05795

[17] Charles Carlson, Vanessa-Rose Turpin, Ahmad Suliman, Carl Ade, Steve Warren, and David E. Thompson. 2021. Bed-based ballistocardiography: Dataset and ability to track cardiovascular parameters. *Sensors* 21, 1 (2021), 156. DOI: https://doi.org/10.3390/s21010156

[18] Juan Cheng, Yufei Xu, Rencheng Song, Yu Liu, Chang Li, and Xun Chen. 2021. Prediction of arterial blood pressure waveforms from photoplethysmogram signals via fully convolutional neural networks. *Computers in Biology and Medicine* 138, 11 (2021), 104877. DOI: https://doi.org/10.1016/J.COMPBIOMED.2021.104877

[19] Ali Cherry, Aya Nasser, Wassim Salameh, Mohamad Abou Ali, and Mohamad Hajj-Hassan. 2025. Real-time PPG-based biometric identification: Advancing security with 2D gram matrices and deep learning models. *Sensors* 25, 1 (Jan. 2025), 40. DOI: https://doi.org/10.3390/s25010040

[20] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. 2018. PACT: Parameterized clipping activation for quantized neural networks. arXiv:1805.06085. Retrieved from http://arxiv.org/abs/1805.06085

[21] Thiago Bulhões Da Silva Costa, Felipe Meneguitti Dias, Diego Armando Cardona Cardenas, Marcelo Arruda Fiuza De Toledo, Daniel Mário De Lima, Jose Eduardo Krieger, and Marco Antonio Gutierrez. 2023. Blood pressure estimation from photoplethysmography by considering intra- and inter-subject variabilities: Guidelines for a fair assessment. *IEEE Access* 11 (2023), 57934–57950. DOI: https://doi.org/10.1109/ACCESS.2023.3284458

[22] Chadi El Hajj and Panayiotis A. Kyriacou. 2020. Cuffless and continuous blood pressure estimation from PPG signals using recurrent neural networks. In *Proceedings of the 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC '20)*, 4269–4272. DOI: https://doi.org/10.1109/EMBC44109.2020.9175699

[23] Mark Wong, Kei Fong, E. Y. K. Ng, Kenneth Er Zi Jian, and Tan Jen Hong. 2019. SVR ensemble-based continuous blood pressure prediction using multi-channel photoplethysmogram. *Computers in Biology and Medicine* 113, 10 (2019), 103392. DOI: https://doi.org/10.1016/J.COMPBIOMED.2019.103392

[24] Flávio D. Fuchs and Paul K. Whelton. 2020. High blood pressure and cardiovascular disease. *Hypertension* 75, 2 (2020), 285–292. DOI: https://doi.org/10.1161/HYPERTENSIONAHA.119.14240

[25] Angelo Garofalo, Manuele Rusci, Francesco Conti, Davide Rossi, and Luca Benini. 2019. PULP-NN: Accelerating quantized neural networks on parallel ultra-low-power RISC-V processors. arXiv:1908.11263. Retrieved from http://arxiv.org/abs/1908.11263

[26] Carlijn Geerse, Cher van Slobbe, Edda van Triet, and Lianne Simonse. 2019. Design of a care pathway for preventive blood pressure monitoring: Qualitative study. *JMIR Cardio* 3, 1 (Jan. 2019) e13048. DOI: https://doi.org/10.2196/13048

[27] Sergio González, Wan Ting Hsieh, and Trista Pei Chun Chen. 2023. A benchmark for machine-learning based non-invasive blood pressure estimation using photoplethysmogram. *Scientific Data* 10, 1 (Mar. 2023), 1–16. DOI: https://doi.org/10.1038/s41597-023-02020-6

[28] Rui He, Zhi-Pei Huang, Lian-Ying Ji, Jian-Kang Wu, Huihui Li, and Zhi-Qiang Zhang. 2016. Beat-to-beat ambulatory blood pressure estimation based on random forest. In *Proceedings of the IEEE 13th International Conference on Wearable and Implantable Body Sensor Networks (BSN '16)*, 194–198. DOI: https://doi.org/10.1109/BSN.2016.7516258

[29] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv:1704.04861. Retrieved from http://arxiv.org/abs/1704.04861

[30] Bin Huang, Weihai Chen, Chun Liang Lin, Chia Feng Juang, and Jianhua Wang. 2022. MLP-BP: A novel framework for cuffless blood pressure measurement with PPG and ECG signals based on MLP-mixer neural networks. *Biomedical Signal Processing and Control* 73, 3 (2022), 103404. DOI: https://doi.org/10.1016/J.BSPC.2021.103404

[31] iMDsoft. 2025. iMDsoft MetaVision ICU. Retrieved March 27, 2025 from https://imd-soft.com/metavision/icu/

[32] Ram Jagannathan, Shivani A. Patel, Mohammed K. Ali, and K. M. Venkat Narayan. 2019. Global updates on cardiovascular disease mortality trends and attribution of traditional risk factors. *Current Diabetes Reports* 19, 7 (Jun. 2019), 44. DOI: https://doi.org/10.1007/s11892-019-1161-2

[33] D. Jahier Pagliari, M. Risso, B. A. Motetti, and A. Burrello. 2023. PLiNIO: A user-friendly library of gradient-based methods for complexity-aware DNN optimization. arXiv:2307.09488. Retrieved from https://arxiv.org/abs/2307.09488

[34] Alistair E. W. Johnson, Tom J. Pollard, Lu Shen, Li Wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific Data* 3, 1 (May 2016), 1–9. DOI: https://doi.org/10.1038/SDATA.2016.35

[35] Greeshma Joseph, Almaria Joseph, Geevarghese Titus, Rintu Mariya Thomas, and Dency Jose. 2014. Photoplethysmogram (PPG) signal analysis and wavelet de-noising. In *Proceedings of the Annual International Conference on Emerging Research Areas: Magnetics, Machines and Drives (AICERA/iCMMD '14)*, 1–5. DOI: https://doi.org/10.1109/AICERA.2014.6908199

[36] Tresa Joseph and Bindiya T. S. 2024. Real-time blood pressure prediction on wearables with edge-based DNNs: A co-design approach. *ACM Transactions on Design Automation of Electronic Systems* 30, 1, Article 11 (Dec. 2024), 24 pages. DOI: https://doi.org/10.1145/3699512

[37] Mohamad Kachuee, Mohammad Mahdi Kiani, Hoda Mohammadzade, and Mahdi Shabany. 2015. Cuff-less high-accuracy calibration-free blood pressure estimation using pulse transit time. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '15)*, 1006–1009. DOI: https://doi.org/10.1109/ISCAS.2015.7168806

[38] Mohammad Kachuee, Mohammad Mahdi Kiani, Hoda Mohammadzade, and Mahdi Shabany. 2017. Cuffless blood pressure estimation algorithms for continuous health-care monitoring. *IEEE Transactions on Bio-Medical Engineering* 64, 4 (2017), 859–869. DOI: https://doi.org/10.1109/TBME.2016.2580904

[39] Tai Le, Floranne Ellington, Tao-Yi Lee, Khuong Vo, Michelle Khine, Sandeep Kumar Krishnan, Nikil Dutt, and Hung Cao. 2020. Continuous non-invasive blood pressure monitoring: A methodological review on measurement techniques. *IEEE Access* 8 (2020), 212478–212498. DOI: https://doi.org/10.1109/ACCESS.2020.3040257

[40] Nayoung Lee, Sang-Hyun Kim, Misoon Lee, and Jiyoung Woo. 2024. Advancing continuous blood pressure estimation with transformer on photoplethysmography in operation room. *IEEE Access* 12 (2024), 90486–90500. DOI: https://doi.org/10.1109/ACCESS.2024.3417940

[41] Lin Li, Chao Chen, Lei Pan, Leo Yu Zhang, Zhifeng Wang, Jun Zhang, and Yang Xiang. 2023. A survey of PPG's application in authentication. *Computers & Security* 135 (Dec. 2023), 103488. DOI: https://doi.org/10.1016/j.cose.2023.103488

[42] Yung-Hui Li, Latifa Nabila Harfiya, Kartika Purwandari, and Yue-Der Lin. 2020. Real-time cuffless continuous blood pressure estimation using deep learning model. *Sensors* 20, 19 (2020), 5606. DOI: https://doi.org/10.3390/s20195606

[43] Yongbo Liang, Zhencheng Chen, Guiyong Liu, and Mohamed Elgendi. 2018. A new, short-recorded photoplethysmogram dataset for blood pressure monitoring in China. *Scientific Data* 5, 2 (2018), 180020. DOI: https://doi.org/10.1038/sdata.2018.20

[44] Sungjun Lim, Taero Kim, Hyeonjeong Lee, Yewon Kim, Minhoi Park, Kwang Yong Kim, Minseong Kim, Kyu Hyung Kim, Jiyoung Jung, and Kyungwoo Song. 2025. Robust optimization for PPG-based blood pressure estimation. *Biomedical Signal Processing and Control* 105, 7 (2025), 107585. DOI: https://doi.org/10.1016/J.BSPC.2025.107585

[45] Wenrui Lin, Berken Utku Demirel, Mohammad Abdullah Al Faruque, and G. P. Li. 2021. Energy-efficient blood pressure monitoring based on single-site photoplethysmogram on wearable devices. In *Proceedings of the 43rd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC '21)*, 504–507. DOI: https://doi.org/10.1109/EMBC46164.2021.9630488

[46] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. DARTS: Differentiable architecture search. arXiv:1806.09055. Retrieved from https://arxiv.org/abs/1806.09055

[47] Russell V. Luepker, Donna K. Arnett, David R. Jacobs, Susan J. Duval, Aaron R. Folsom, Christopher Armstrong, and Henry Blackburn. 2006. Trends in blood pressure, hypertension control, and stroke mortality: The Minnesota heart survey. *The American Journal of Medicine* 119, 1 (Jan. 2006), 42–49.

[48] Gloria Martínez, Newton Howard, Derek Abbott, Kenneth Lim, Rabab Ward, and Mohamed Elgendi. 2018. Can photoplethysmography replace arterial blood pressure in the assessment of blood pressure? *Journal of Clinical Medicine* 7, 10 (2018), 316. DOI: https://doi.org/10.3390/jcm7100316

[49] Agnes S. Meidert and Bernd Saugel. 2017. Techniques for non-invasive monitoring of arterial blood pressure. *Frontiers in Medicine* 4 (Jan. 2017), 231. DOI: https://doi.org/10.3389/fmed.2017.00231

[50] Beatrice Alessandra Motetti, Matteo Risso, Alessio Burrello, Enrico Macii, Massimo Poncino, and Daniele Jahier Pagliari. 2024. Joint pruning and channel-wise mixed-precision quantization for efficient deep neural networks. *IEEE Transactions on Computers* 73, 11 (2024), 2619–2633. DOI: https://doi.org/10.1109/TC.2024.3449084

[51] Ramakrishna Mukkamala, Jin-Oh Hahn, Omer T. Inan, Lalit K. Mestha, Chang-Sei Kim, Hakan Töreyin, and Survi Kyal. 2015. Toward ubiquitous blood pressure monitoring via pulse transit time: Theory and practice. *IEEE Transactions on Bio-Medical Engineering* 62, 8 (2015), 1879–1901. DOI: https://doi.org/10.1109/TBME.2015.2441951

[52] Ayodele Odutayo, Christopher X. Wong, Allan J. Hsiao, Sally Hopewell, Douglas G. Altman, and Connor A. Emdin. 2016. Atrial fibrillation and risks of cardiovascular disease, renal disease, and death: Systematic review and meta-analysis. *BMJ* 354 (2016), i4482. DOI: https://doi.org/10.1136/bmj.i4482

[53] Caijie Qin, Yong Li, Chibiao Liu, and Xibo Ma. 2023. Cuff-Less blood pressure prediction based on photoplethysmography and modified ResNet. *Bioengineering* 10, 4 (2023), 400. DOI: https://doi.org/10.3390/bioengineering10040400

[54] K. Rishi Vardhan, S. Vedanth, G. Poojah, K. Abhishek, M. Nitish Kumar, and Vineeth Vijayaraghavan. 2021. BP-Net: Efficient deep learning for continuous arterial blood pressure estimation using photoplethysmogram. In *Proceedings of the 20th IEEE International Conference on Machine Learning and Applications (ICMLA '21)*, 1495–1500. DOI: https://doi.org/10.1109/ICMLA52953.2021.00241

[55] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. arXiv:1505.04597. Retrieved from https://arxiv.org/abs/1505.04597

[56] Monalisa Singha Roy, Rajarshi Gupta, and Kaushik Das Sharma. 2022. BePCon: A photoplethysmography-based quality-aware continuous beat-to-beat blood pressure measurement technique using deep learning. *IEEE Transactions on Instrumentation and Measurement* 71 (2022), 1–9. DOI: https://doi.org/10.1109/TIM.2022.3212750

[57] M. Saeed, C. Lieu, G. Raber, and R. G. Mark. 2002. MIMIC II: A massive temporal ICU patient database to support research in intelligent patient monitoring. In *Proceedings of the Computers in Cardiology*, 641–644. DOI: https://doi.org/10.1109/CIC.2002.1166854

[58] Nordic Semiconductor. 2024. Nordic II. Retrieved May 16, 2024 from https://www.nordicsemi.com/

[59] Mingda Sheng, Rui Xing, Youze Xin, Bing Zhang, Zhuoqi Guo, Zhongming Xue, and Li Geng. 2024. A 4.4μW cuffless blood pressure measurement processor based on event-driven and module-level asynchronous scheme. In *Proceedings of the IEEE Biomedical Circuits and Systems Conference (BioCAS '24)*, 1–5. DOI: https://doi.org/10.1109/BioCAS61083.2024.10798356

[60] Daniel Stoller, Sebastian Ewert, and Simon Dixon. 2018. Wave-U-Net: A multi-scale neural network for end-to-end audio source separation. arXiv:1806.03185. Retrieved from https://arxiv.org/abs/1806.03185

[61] Bailian Sun, Safin Bayes, Abdelrhman Mohamed Abotaleb, and Mohamed Hassan. 2023. The case for tinyML in healthcare: CNNs for real-time on-edge blood pressure estimation. In *Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (SAC '23)*. ACM, New York, NY, 629–638. DOI: https://doi.org/10.1145/3555776.3577747

[62] Greenwaves Technologies. 2024. Gap8. Retrieved May 16, 2025 from https://github.com/GreenWaves-Technologies/gap_sdk

[63] Zhonghe Tian, Aiping Liu, Guokang Zhu, and Xun Chen. 2025. A paralleled CNN and transformer network for PPG-based cuff-less blood pressure estimation. *Biomedical Signal Processing and Control* 99, 1 (2025), 106741. DOI: https://doi.org/10.1016/J.BSPC.2024.106741

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS '17)*. Curran Associates Inc., Red Hook, NY, 6000–6010.

[65] Matthew Ward and Jeremy A. Langton. 2007. Blood pressure measurement. *Continuing Education in Anaesthesia Critical Care & Pain* 7, 4 (2007), 122–126. DOI: https://doi.org/10.1093/bjaceaccp/mkm022

[66] William B. White, Alan S. Berson, Carroll Robbins, Michael J. Jamieson, L. Michael Prisant, Edward Roccella, and Sheldon G. Sheps. 1993. National standard for measurement of resting and ambulatory blood pressures with automated sphygmomanometers. *Hypertension* 21, 4 (1993), 504–509. DOI: https://doi.org/10.1161/01.HYP.21.4.504

[67] Jialong Zhang, Jianzheng Li, Yizhou Jiang, Kai Wang, Ran Guo, Yu Ma, and Yajie Qin. 2022. A hardware-based lightweight ANN for real-time wearable blood pressure estimation. In *Proceedings of the 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC '22)*, 4295–4298. DOI: https://doi.org/10.1109/EMBC48229.2022.9871215

[68] Yiming Zhang, Shirong Qiu, Kai Du, Shun Wu, Ting Xiang, Kenghao Zheng, Zijun Liu, Hanjie Chen, Nan Ji, Fa Wang, et al. 2026. Artificial intelligence-enhanced wearable blood pressure monitoring in resource-limited settings: A co-design of sensors, model, and deployment. *Nano-Micro Letters* 18 (2026) 164. DOI: https://doi.org/10.1007/s40820-025-02003-9

[69] Yiming Zhang, Congcong Zhou, Xianglin Ren, Qing Wang, Hongwei Wang, Ting Xiang, Shirong Qiu, Yuan-Ting Zhang, and Xuesong Ye. 2025. Personalized continuous blood pressure tracking through single channel PPG in wearable scenarios. *IEEE Journal of Biomedical and Health Informatics* 29, 6 (2025), 4109–4120. DOI: https://doi.org/10.1109/JBHI.2025.3535788

[70] Kai Zhou, Zhixiang Yin, Yu Peng, and Zhiliang Zeng. 2022. Methods for continuous blood pressure estimation using temporal convolutional neural networks and ensemble empirical mode decomposition. *Electronics* 11, 9 (2022), 1378. DOI: https://doi.org/10.3390/electronics11091378

Received 21 October 2025; revised 8 February 2026; accepted 3 April 2026