Expert Systems With Applications 238 (2024) 121523

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Wearable edge machine learning with synthetic photoplethysmograms ☆

Jukka-Pekka Sirkiä *, Tuukka Panula, Matti Kaisti

Department of Computing, University of Turku, Vesilinnantie 5, Turku, 20500, Finland

ARTICLE INFO

Dataset link: https://github.com/jpsirk/synthetic-photoplethysmograms

Keywords:
Edge machine learning
Neural network
Wearable
Photoplethysmography
Synthetic

ABSTRACT

Strict privacy regulations pose challenges to the development of machine learning (ML) in the field of health technology where data is particularly sensitive. Gathering and using robust, bias-free, and suitably anonymized datasets required by ML models is difficult, time-consuming, and thus expensive. Parametric synthetic data offers a solution by mimicking real-world processes with easily adjustable parameters that shape the information content of the data as desired. This article presents a system demonstrating how synthetic data can be used in conjunction with wearable edge devices. Importantly, the system preserves privacy as there is no risk of leaking sensitive information from the model or during the use of the wearable device. The system consists of (1) a synthetic photoplethysmogram (PPG) model, (2) convolutional neural network (CNN) models trained with the synthetic signals, (3) a wearable edge device that computes heart rate from real-time PPG signals using the developed CNN models, and (4) an accompanying mobile phone application receiving the results. The synthetic model produces realistic PPG signals together with labels that can be used in CNN model training. The quality of the synthetic data is sufficient to train even a tiny CNN model with only two convolutional layers and 28 parameters to detect PPG waveform feet. The developed wearable device is able to run the model smoothly and the performance of the model is on par with the more complex models and other foot detection algorithms.

1. Introduction

The use of machine learning (ML) methods in the field of health technology is typically hindered by privacy requirements, biased training data and insufficient amount of training data (Chen et al., 2021; Rajotte et al., 2022). Tightened privacy regulations, such as the General Data Protection Regulation (GDPR) 2016/679 of the European Union (EU), have led some to demand to remove the barriers to sharing health data to avoid potentially damaging effects on research and therefore ultimately on patients (Bentzen et al., 2021). For example, pseudonymized data are considered personal data under GDPR even though the identifiers have been replaced by codes (Bentzen et al., 2021). Bias in medical data can easily occur if the study participant backgrounds are homogeneous (e.g., socioeconomic factors, gender, age, and diseases) or the classes present in the data are imbalanced. Gathering data with only one type of equipment is also prone to producing data that might not generalize well in reality Chen et al. (2021). The lack of suitable existing data or access to them can also pose difficulties in obtaining large datasets. Data sharing practices are intended to facilitate the verification of published results and help build on existing data, but challenges arise from obeying these practices (Naudet et al., 2018).

These challenges can be addressed with synthetic data, which is data not originated directly from the observable world. Synthetic data can be generated with ML-based generative models (e.g. generative adversarial networks (GANs)), parametric models or physical simulations (Chen et al., 2021; Jordon et al., 2022). The models-based approach allows to efficiently generate data in large quantities, and the problem of bias is easier to tackle. More importantly, parametric and physical simulation models are inherently private, given that the data is generated with purely mathematical models. The privacy aspect can be transferred from the ML training phase to the deployment phase by securing the data from which predictions are to be made. However, also here tightening regulation, such as GDPR, can pose requirements for storing and transferring the data, for example outside the EU in the case of GDPR. Therefore, from a privacy point of view, the best approach is not to move the data at all.

With the increased computing power of microcontroller units (MCUs) and evolved software tools, running ML models close to the data source has become feasible. This is known as edge ML (Merenda et al., 2020; Murshed et al., 2021), and under the paradigm, ML models are executed directly on less-powerful MCUs (compared to, e.g. laptops, not to mention cloud-based ML-optimized tensor processing units),

☆ The works of J.-P. Sirkiä and T. Panula were partially funded by the University of Turku Graduate School.
* Corresponding author.
E-mail addresses: jpsirk@utu.fi (J.-P. Sirkiä), mkaist@utu.fi (M. Kaisti).

https://doi.org/10.1016/j.eswa.2023.121523
Received 5 May 2023; Received in revised form 25 August 2023; Accepted 7 September 2023
Available online 22 September 2023

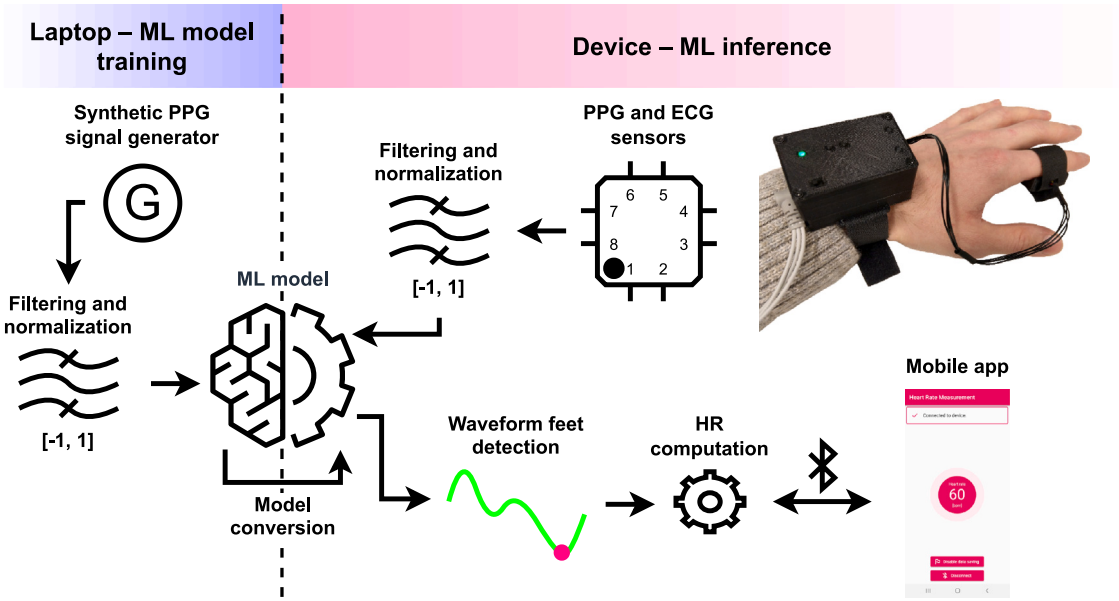

Fig. 1. Developed system. On the left: The generator is used to create a large number of synthetic PPG signals that are then bandpass-filtered and normalized to an interval of [−1, 1]. The processed signals are used as training data for the ML models. On the right: The PPG and ECG sensors of the device record signals. The green PPG signal is bandpass filtered and normalized to an interval of [−1, 1] before it is passed to the ML model. The output of the ML model (prediction array) is processed to detect the waveform feet that are used to compute heart rate (HR). The computed HR reading is transmitted to a connected mobile phone over a Bluetooth Low Energy (BLE) connection. The developed device is shown in the upper right corner.

skipping the transfer of data to servers where ML models are typically executed. Additional benefits of edge ML include minimized latency and bandwidth usage (Wu et al., 2019).

In this work, we present a system consisting of (1) a wearable device including photoplethysmography (PPG) and electrocardiogram (ECG) sensors, (2) a synthetic PPG generator, (3) a convolutional neural network (CNN) model trained with purely synthetic signals and running on the wearable (edge) device to detect PPG waveform feet in real time, and (4) a mobile phone application that receives computed heart rate (HR) values from the device. The system has been illustrated in Fig. 1. The synthetic PPG signal generator is shown to produce realistic signals that can be used to train neural network (NN) models. Usage of these models is demonstrated in HR computation at the edge.

2. Related work

2.1. Synthetic photoplethysmogram signals

Synthetic PPG signals have been generated using parametric models that sum Gaussian functions together (Martín-Martínez et al., 2013; Tang, Chen, Ward et al., 2020) or, alternatively, Gaussian functions with log-normal functions (Sološenko et al., 2017). In these models, the PPG pulse waveform is reconstructed by controlling the Gaussian function parameters so that the resulting set of bumps jointly form the desired waveform shape. More complex physiological models have also been developed that rely on simulation of cardiac function, including cardiac chambers, valves, and systemic circulation (Mazumder et al., 2022). Deep learning-based GAN models have generally become popular in synthetic data generation (Jordon et al., 2022) with synthetic PPG signals being no exception (Kiyasseh et al., 2020). In this article, the synthetic PPG generator falls into the first category and is inspired by the work of McSharry et al. (2003) where synthetic electrocardiogram (ECG) signals are generated with a set of differential equations that model the characteristic points of the ECG waveform and Kaisti et al. (2021) where the same generation principle is further simplified by eliminating the need to solve a set of differential equations. Unlike in Kaisti et al. (2021) where a parametric noise power spectral density (PSD) was used to transform synthetic signals into realistic noisy signals, here, we sample time-domain noise realizations from measured PSDs.

2.2. Neural networks and edge computing in health wearables

NN models have been widely used in classification and peak detection in biosignals, especially in electrocardiogram (ECG) signals (Acharya et al., 2017; Kaisti et al., 2021; Laitala et al., 2020; Sannino & De Pietro, 2018; Śmigiel et al., 2021) and PPG signals (Aschbacher et al., 2020; Kazemi et al., 2022), and recently in estimating blood pressure from PPG signals (Elgendi et al., 2019). Edge ML has been used, for example, for ECG beat classification (Hou et al., 2020) and detection of atrial fibrillation from ECG signals (Chen et al., 2023). In the development of edge ML models, the limitations of hardware in terms of computing capability and memory can quickly become a factor to consider in model development. For example, the long-short-term memory (LSTM)-based ECG R wave detection model presented in Laitala et al. (2020) and Kaisti et al. (2021) has 132,737 trainable parameters with a model size of approximately 541 kB, while the dilated convolutional neural network (CNN) model presented in Kazemi et al. (2022) to detect PPG peaks has only 3169 trainable parameters with a model size of approximately 17 kB. Although both models are relatively small, the CNN model with its significantly smaller memory requirement is more suitable for wearable devices and was thus selected as the reference NN model in this study. This reference model was further simplified into two smaller models.

3. Methods

3.1. Wearable device

The developed wearable device consists of a system on a chip (SoC) (ESP32-S3 by Espressif Systems, China), a PPG sensor (MAX30101 by Maxim Integrated, United States), an ECG module (AD8232 by Analog Devices, United States), and an analog-to-digital converter (ADC) (ADS1115 by Texas Instruments, United States). The components have been presented as a system block diagram in Fig. 2. The SoC has 512 KB of static random access memory (SRAM), an additional 8 MB of external pseudostatic random access memory (PSRAM), and 16 MB of flash storage. Together with the dual-core 240 MHz processor, the device has a lot of processing power, memory, and storage to execute

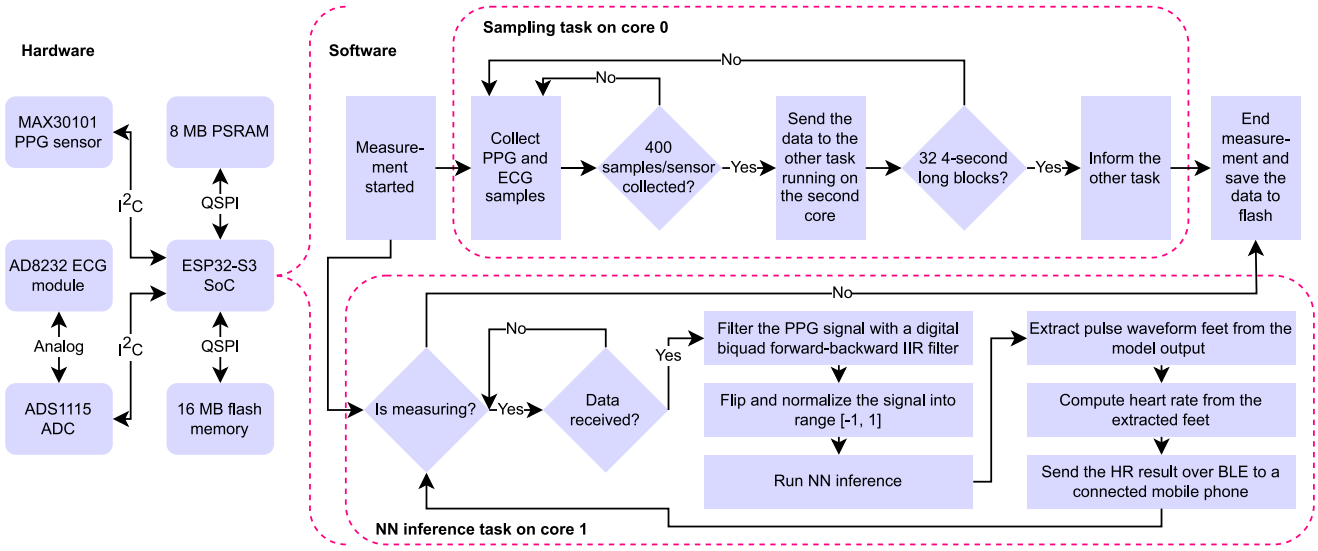

Fig. 2. A system block diagram of the hardware components and a flowchart of the two main software tasks responsible for sampling the sensors and performing NN inference & HR computation.

edge ML models and store raw sensor data together with the output of the ML model. The PPG sensor has three light-emitting diodes (LEDs): green (537 m), red (660 nm), and infrared (880 nm). The analog signal produced by the ECG module is converted to a digital signal with the 16 bit ADC. The system can be powered through the universal serial bus (USB) or with a rechargeable battery pack (950 mAh 3.7 V lithium polymer) to allow measurements on the move.

The PPG sensor was encapsulated in a custom-made casing with a Velcro strap that allows it to wrap around the finger so that the sensor is approximately in the middle of the proximal phalanx (where a ring is placed). Another larger casing was designed for the rest of the electronics. The case has holes for the PPG I^2C wires and the ECG cable, a power switch and buttons to reset the device and start/stop a measurement. The casing is wrapped around the wrist with a Velcro strap. The resulting device is shown in Fig. 1. All parts were designed with Autodesk Inventor Professional 2022 computer-aided design (CAD) software. The polylactic acid (PLA) parts were 3D printed using the fused filament fabrication (FFF) technique (CR-20 Pro by Creality, China).

3.1.1. Firmware

The SoC was programmed in C/C++ using Espressif's Internet of Things (IoT) development framework, known as ESP-IDF, built around the real-time operating system FreeRTOS. The software was programmed to utilize the two cores of the SoC to achieve maximal performance. The developed software has two main (FreeRTOS) tasks: sampling of the PPG and ECG sensors and running inference on the ML model. The inference task was assigned its own core because, together with the preprocessing and HR computation steps, it can take a few hundreds of milliseconds of time. The sampling task runs on the other core and pushes data to the inference task using queues. The main parts of these two main tasks have been illustrated in Fig. 2. An additional third task is run on the ML core to listen for commands over the serial port.

The SoC boot push button is programmed as a button to start and stop a measurement. In the idle state, a push of the button notifies the sampling and ML tasks of a request to start a new measurement. The sampling task polls the PPG and ECG sensors at a sampling frequency of 100 Hz (both for the sensors and for the three channels of the PPG sensor). The samples are temporarily stored into buffers and once four seconds (i.e., 400 samples per channel) of the signals have been gathered, the data is transferred to the ML task using queues. The ML task first runs the preprocessing steps for the selected PPG channel, as described in chapter 3.3.1. The Butterworth filter was implemented using the same digital biquad filter coefficients used in the Python analysis part. The ML inference is then executed and the raw data received together with the model output are temporarily stored in buffers. The measurement is automatically stopped after 128 s or alternatively with another press of the button. The temporarily stored data is stored on the flash storage. The serial task handles the reading of the stored data over the serial port. This can be done using the developed computer code.

The device firmware also allows performing real-time measurements without saving the data. In this mode, the device performs only ML inference and HR computation every four seconds. None of the data is stored on the flash but instead the computed HR readings are transmitted over Bluetooth Low Energy (BLE) to a developed mobile application.

3.1.2. Computer software

A simple computer program was developed to read the measurement data stored in the flash memory. The program can request the stored filenames and read the data files one by one. The data received are stored as a comma-separated value (CSV) file on the computer.

3.1.3. Mobile application

A mobile application was developed using the cross-platform Flutter software development kit (SDK). The application has a button to connect to the developed device and another one to enable or disable the device data saving mode. With data saving disabled, the developed device only makes predictions using the ML model and notifies the connected device of the computed HR values without saving any data. The HR values received are displayed to the application user as a number together with a ripple animation with a rhythm that matches the current HR. The main screen of the application is shown in Fig. 1.

3.2. Synthetic photoplethysmogram generation

The model used to generate synthetic PPG signals is inspired by the work of McSharry et al. (2003) and Kaisti et al. (2021) presenting the generation of synthetic ECG signals. The derivative of a single synthetic PPG pulse waveform is modeled as a sum of Gaussian function derivatives of the form:

$$s'_{synthetic} = \sum_{i=1}^{N} -\frac{\mathbf{x_i}}{c_i^2} f(\mathbf{x_i}), \tag{1}$$

where $\mathbf{x_i}$ is a phase signal for the ith Gaussian bump with values linearly spaced in the range $[-\pi, \pi]$ (i.e., a PPG pulse waveform is modeled as a rotation around a circle with values sampled at points $x_1, \ldots, x_n$, where $x_1 = -\pi$ and $x_n = \pi$) but with values shifted to the left or right of the center of the pulse waveform by d_i (in radians from the center point) depending on the location of the desired Gaussian bump (e.g., the Gaussian bump modeling the systolic peak of a PPG pulse waveform is shifted to the left), N is the number of Gaussian bumps (at minimum two, i.e., to model the systolic and diastolic parts of a PPG waveform), c_i is the width of the bump, and $f(\mathbf{x_i})$ is a Gaussian function:

$$f(\mathbf{x_i}) = a_i exp\left(\frac{-\mathbf{x_i}^2}{2c_i^2}\right), \tag{2}$$

where a_i is an amplitude modifier. The derivative, $s'_{synthetic}$, is then smoothed with a 2nd order Savitzky–Golay filter with a window length of 11. Smoothing is included in the algorithm to handle cases where a Gaussian bump is wide enough to cause discontinuity in the final summed derivative. In practice, the effect of a potential discontinuity is negligible, as shown in the supplementary information, and is not needed if the widths of the bumps are limited and the number of bumps is increased, as also shown in the supplementary information. The final signal, $s_{synthetic}$, is calculated with cumulative numerical integration using the trapezoidal rule with an initial value of zero and a time step of $1/f_s$ where f_s is the sampling frequency.

The model can be used to create a single PPG pulse waveform using a phase signal for a single cardiac cycle or a complete PPG signal by concatenating a series of $[-\pi, \pi]$. The number of samples in any given pulse waveform interval k (≥ 1) is computed using the equation adapted from Kaisti et al. (2021):

$$n_k = f_s[l + bsin(2\pi f_b \sum_{j=1}^{k-1} \frac{n_j}{f_s})], \tag{3}$$

where f_s is the sampling frequency (in Hz), l is the mean pulse waveform interval length (in seconds), b is the breathing coupling coefficient (assumed to be 0.1) and f_b is the breathing frequency (in Hz). Therefore, the calculated n_k defines the heart rate for the pulse waveform interval k. For example, $n_k = 80$ together with a sampling frequency of 100 Hz is equal to an HR of 75 ($= 60/(80/100)$) beats per minute (bpm).

Parametric waveform generation allows to find the waveform feet with simple rules. The first foot must be close to the beginning of the synthetic signal, whereas the rest of the feet must be close to the end points of the modeled pulse waveforms. Therefore, an array of potential foot locations was created by adding the pulse length n_k to the previous potential location, starting from zero, that is, $\{0, n_1, n_1 + n_2, \ldots\}$. A foot was defined as the minimum point within ± 100 ms of a potential location. The resulting array of feet was then converted into a label signal, where each foot corresponds to five subsequent ones, the rest of the signal being zero.

3.2.1. Waveform fitting and model parameters

The synthetic model was fitted to a real PPG pulse waveform normalized to the range [0, 1] to find suitable model parameters. The following objective function was used:

$$\min_{\mathbf{p}} \sum_{j=1}^{L} (s_{synthetic}(\mathbf{p})_j - s_{real_j})^2 \tag{4}$$

$$\text{s.t.} -\pi \leq d_1 \leq d_2 \leq \pi$$

$$0 \leq c_1 \leq c_2 \leq 3$$

$$0 \leq a_1, a_2 \leq 10,$$

where $\mathbf{p}$ is the synthetic model parameters $(d_1, c_1, a_1, \ldots)$ and L is the length (number of samples) of the real PPG waveform. The objective function was minimized using the differential evolution algorithm. In this study, the number of Gaussian functions was selected to be two to keep the model intuitive, one Gaussian function effectively fitting the systolic part of the pulse waveform and the other modeling the diastolic part of the waveform. However, as shown in the supplementary information, the number of Gaussian bumps could be increased to, for example, four while simultaneously making them narrower. The fitted pulse waveform, together with the real pulse waveform and the optimal model parameters, is presented in Fig. 3a. The parameters were then used to manually estimate parameter ranges that yield a rich variety of pulse waveforms, from waveforms with distinct diastolic peaks to more triangular waveforms with less pronounced characteristic points, when randomly sampled. These ranges have been listed in Table 1. Synthetic signals modeled using the lower and upper boundaries of the pulse waveform ranges along with their means have been presented in Fig. 3b and c.

Table 1
Parameter ranges for generating randomized synthetic PPG signals.

(a) PPG waveform parameter ranges.

Parameter	Waveform bump		Unit
	Systole	Diastole	
d (shift)	[−2.0, −1.4]	[0.4, 1.0]	rad
c (width)	[0.5, 0.9]	[1.7, 2.1]	arb. unit
a (amplitude)	[5.0, 10.0]	[5.0, 9.0]	arb. unit

(b) Ranges for the rest of the parameters.

Parameter	Range	Unit
l (mean waveform length)	[0.4, 1.3]	s
f_b (breathing frequency)	[0.15, 0.4]	Hz
a_{noise} (noise amplitude)	[0, 1.5]	arb. unit

3.2.2. Noise generation

The PPG signals generated with the model are unrealistically clean, which is why a noise generator was added to render the signals realistic, i.e. contaminated with noise arising from, e.g., the sampling process (sensor-related) and movement. The noise generator used is based on converting a PSD from the frequency domain to the time domain using the algorithm presented in Timmer and Koenig (1995). Three measurements (each a little more than two minutes long) were recorded while sitting, walking, and moving the hand with the device in random ways. These measurements were then divided into four-second blocks and PSDs were computed for each block using the Welch's method (Welch, 1967). Then, the computed PSDs were averaged, and an interpolation function was created. The mean PSD's extracted from the measurements have been presented in Fig. 4. The resulting three interpolation functions can then be passed to another function that converts them into time-domain noise. The generated noise is thus representative of the device- and measurement-protocol-specific noise without the need for separate generator processes for baseline wander and white noise.

The noise generator, along with all other code related to synthetic signals, was written in Python programming language. The code uses parallel computing to speed up the generation process. As an example, 200,000 4-s long signals can be generated in approximately two minutes using a laptop with an Intel i7-11850H central processing unit (CPU) having 16 logical processors and combined with 32 GB of random access memory (RAM).

3.3. Neural network model

The selected reference model presented in Kazemi et al. (2022) was used as a base for developing two additional CNN models: (1) a small model with four convolutional layers and (2) a tiny model with only two convolutional layers. Between the convolutional layers are normalization layers inspired by the work of Liu et al. (2022) where layer normalization was used for CNNs instead of perhaps more typical

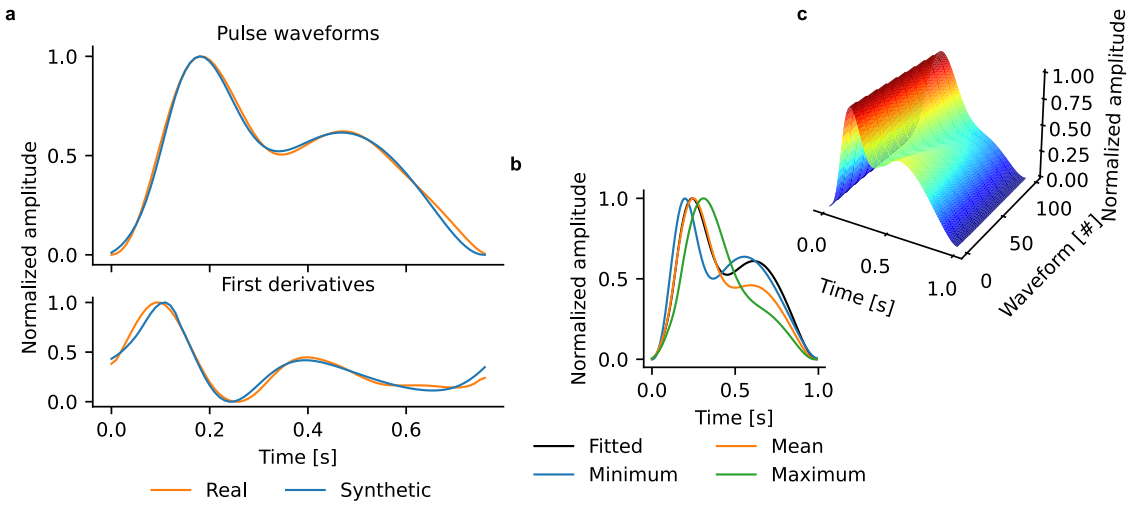

Fig. 3. Waveform fitting. a. A real PPG pulse waveform and its fitted synthetic waveform. The synthetic waveform was created with the optimized parameter values of: $d_1 = -1.81, c_1 = 0.68, a_1 = 8.35, d_2 = 0.82, c_2 = 1.89, a_2 = 9.69$. b. Generated synthetic PPG pulse waveforms with minimum, mean, and maximum parameter values along with the original fitted waveform presented in Fig. a. At the lower end of the range the dicrotic notches are clearly visible, whereas at the other end of the range the pulse waveforms are very triangular, reminding of a PPG signal typically obtained from elderly persons. c. Evolution of the waveform from a waveform generated with the minimum parameters to a waveform generated with the maximum parameters.

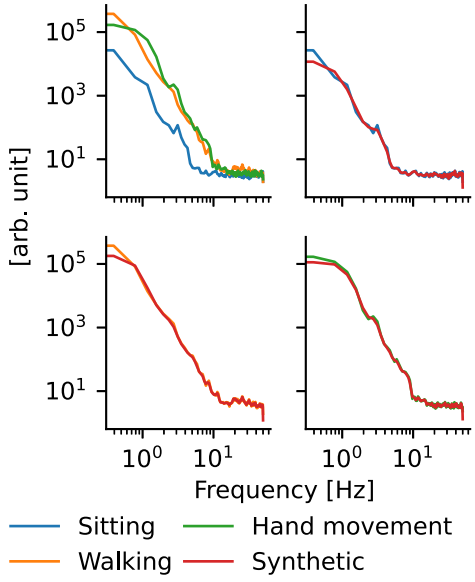

Fig. 4. PSDs used in noise generation. The figure on the upper left shows the mean PSDs computed for each type of measurement (sitting, walking, and hand movement). The rest of the figures show the same computed mean PSDs for each measurement type, but now together with their sampled (synthetic) PSDs. The synthetic PSDs overlap with the original ones, meaning that the noise generator works as intended.

batch normalization. The kernel size is five for each convolutional layer. The activation function is sigmoid for the last layer and swish otherwise. The input size is 400 samples long, i.e. equal to a 4-s signal with the used sampling rate of 100 Hz. The NN model architectures have been summarized in Table 2. The loss function is based on the Wasserstein distance considering the good performance results in Sun et al. (2022) where the systolic peaks were identified from remote PPG signals. The loss function is defined as:

$$l_w(\mathbf{p}, \mathbf{r}) = \sum_{j=1}^{L} |F_j(\mathbf{p}) - F_j(\mathbf{r})|, \tag{5}$$

where $\mathbf{p}$ is an array of predicted values and $\mathbf{r}$ is the array of the input label, F denotes the cumulative sum (that is, for example, $F(\mathbf{p})$ is the cumulative distribution of the predicted probability values) and L is the length (number of samples) of the arrays ($L = 400$ in this study).

Table 2
The NN models considered in this study. The layers are convolutional except for those labeled "Norm.", which are normalization layers. The parameters in the convolutional layers are ordered as follows: number of filters, dilation rate, and activation function. The layers have been ordered so that the first layer is on the top row and the last layer is on the bottom row. The output shape in each layer is the same as is the NN input layer shape, i.e., 400 samples.

	Reference NN model	NN model 1	NN model 2
Kernel size	3	5	5
Parameters	3,169	296	28
Trainable parameters	3,169	265	23
Model size [kB]	21.3	7.7	3.6
Layers	4, 1, elu	2, 1, swish	2, 2, swish
	8, 2, elu	Norm.	Norm.
	8, 4, elu	4, 2, swish	1, 4, sigmoid
	16, 8, elu	Norm.	–
	16, 16, elu	8, 4, swish	–
	32, 32, elu	Norm.	–
	1, 64, sigmoid	1, 8, sigmoid	–

The CNN model outputs are 400 samples long (i.e., the same length as the input array) probability label arrays. Each value ($\in [0, 1]$) in the array is the probability of a sample being predicted as one. Five subsequent ones mark the location of a foot in the input array, as detailed in chapter 3.2.

Model development was performed using TensorFlow 2.9.1 and the actual NN training was performed using NVIDIA T600 laptop graphics processing unit (GPU). All models were trained with 200 epochs. The batch size was 256 for all the models, and the generated synthetic signals were divided into training, validation, and test datasets using splits of 80%, 10%, and 10%, respectively. The optimizer was Adam with a learning rate of 0.001. The final trained models were converted to TensorFlow Lite models without any optimizations. These models were then further converted to C arrays that were compiled into the device firmware. Note that only one model was included in the device firmware at a time and that the final analysis was performed offline. It was verified that the NN model inputs (preprocessed PPG signals) and (inference) outputs computed using both online (in firmware) and offline (in Python) implementations matched each other.

3.3.1. Preprocessing

Real and synthetic PPG signals were filtered with a forward–backward (zero phase) 2nd order Butterworth bandpass filter with cutoff frequencies set at 0.5 Hz and 5.0 Hz to remove baseline wander and high-frequency noise components. The computed digital biquad filter coefficients were used in both online (the filter was implemented in the device firmware) and offline analysis. The initial filter conditions were calculated using the *sosfilt_zi* function of SciPy's signal module. The real signals were flipped, i.e. the systolic peaks point up. Both real and synthetic signals were normalized to the range [−1, 1]. The reference ECG signal was filtered with a zero-phase 4th order Butterworth bandpass filter with cutoff frequencies set at 2 Hz and 40 Hz. The relatively high high-pass cutoff frequency of 2 Hz was chosen to suppress motion-related baseline wandering and artifacts, whereas the low-pass cutoff frequency of 40 Hz was chosen to remove high-frequency noise while still meeting the Nyquist-Shannon sampling theorem, i.e., sampling rate (in this study 100 Hz) must be at least twice the bandwidth of the signal. These cutoff frequencies were considered reasonable for HR measurement purposes considering that the frequency content of the QRS complex of the ECG wave is generally between 8–50 Hz (Tereshchenko & Josephson, 2015).

The PPG feet labels of the synthetic signals were corrected after the above preprocessing steps to account for the fact that the locations of ones might not be centered anymore with the true local minima of the signal due to the added noise and subsequent filtering. The labels were corrected by finding the closest local minimum within 9 (90 ms) samples from the original center point and shifting the ones accordingly. This correction had a rather large impact on training and validation losses, as shown in the supplementary information.

3.3.2. Postprocessing

The probability label array returned by the NN model is processed by an algorithm to extract the waveform foot locations. Small values are set to zeros with a threshold empirically set to 0.1. A lower/higher threshold would improve/impair recall, but impair/improve precision. Each probability peak is then checked, and the maximum probability value within the peak (the starting/end point is the index at which the value is above/below the threshold) along with the location are recorded. The index corresponding to the minimum of the preprocessed signal within ±50 milliseconds of the maximum probability value is also stored in a list of possible feet. The probability locations and the potential feet are then sorted using the probabilities and looped through starting from the foot with the highest probability value. A potential foot is added to a list of final feet if the corresponding probability location is not within a 0.3 s window (which corresponds to a heart rate of 200 bpm) from any of the probability locations already considered earlier when including feet to the final list. Lastly, the final feet are sorted in ascending order based on the location indices. The time differences between subsequent feet are then computed, ignoring time differences greater than 1.5 s (HR of 40 bpm) and smaller than 0.3 s (HR of 200 bpm). HR during the 4-s interval is computed if there is at least one time difference, that is, at least two waveform feet, and it is computed by dividing 60 by the mean time difference (in seconds).

In order to evaluate the performances of the NN models, a set of peak detection algorithms was selected. Based on the benchmarked results in Charlton et al. (2022) automatic multiscale-based peak detection (AMPD) algorithm (Scholkmann et al., 2012) and its optimized version multi-scale peak and trough detection (MSPTD) (Bishop & Ercole, 2018) were selected together with the HeartPy algorithm presented in van Gent et al. (2019). The AMPD algorithm was implemented using the Python package pyampd, which also includes a modified version of the algorithm trying to overcome the original algorithm's problems in finding peaks close to the beginning and end of a signal. This modified version of the algorithm was also considered in this study. The MSPTD algorithm was implemented by converting the original Matlab code into Python code. The HeartPy algorithm was implemented using the Python package released by the authors of the algorithm. Threshold-based algorithms, such as Shin et al. (2009), were not considered because such algorithms typically do not work well with short signals. Regarding HR values, an additional simple HR algorithm based on autocorrelation (Das et al., 2016) was used along the foot detectors. Autocorrelation was calculated for each 4-s signal block, and the first peak location was considered as the length of the pulse waveform. The same postprocessing steps were applied before converting the value into HR.

The feet detected by the algorithms were compared with the ECG R peaks. A foot was considered to be valid if it occurred after the R peak and was not delayed by more than 300 ms with respect to the R peak. This value was selected after considering that the pulse arrival time (PAT) between the ECG R peak and the PPG waveform foot measured from the finger is typically at rest 180–260 ms (Mukkamala et al., 2015). The rather large window of 300 ms should cover the fact that motion-related noise falling within the pass band of the filter can distort the waveforms, and hence the waveform foot locations, in such a way that the rather tight ranges of variation measured at rest, such as the above 80 ms, would reject many of the feet.

3.4. Experiments

The developed device was tested on 20 volunteers (mean age 33.9 years ± 9.2 years, range [23, 65] years) of which 9 were females. Three different measurements were taken from each subject: (1) sitting comfortably on a chair, (2) walking, and (3) moving the hand with the device in random ways. Each measurement was two minutes and eight seconds long with approximately the first and last 20 s recorded with no movement at all. Despite that the device recorded three different PPG channels, only the green channel was used in this study, given the results in Lee et al. (2013) where green light was shown to be more suitable for tracking HR in normal daily life than shorter-wavelength blue light or longer-wavelength red light. Due to the poor quality of the ECG signal in five measurements, the total number of measurements was 55 (of which 20 were sitting, 17 walking, and 18 with hand movement).

4. Results and discussion

The output of the synthetic model is illustrated in Fig. 5a which shows the clean signal together with its more realistic version after adding a noise component to it. Fig. 5b shows longer snippets of synthetic signals generated in each of the three noise scenarios together with the clean signal on which all three noisy signals are based. The synthetic signals presented in the figures are smooth and continuous and also exhibit variations in amplitude and interval due to breathing modulation. After adding noise, the signals represent realistic raw PPG signals with low-frequency baseline wander and white noise. The synthetic model was used to generate 200,000 four seconds long realistic PPG signals by using uniform random variables and the parameter ranges presented in the Table 1. Thus, in approximately 33% of the generated synthetic signals the heart rate was lower than 60 bpm (mean waveform length > 1 s) and in 22% of the signals the heart rate was higher than 100 bpm (mean waveform length < 0.6 s). The noise type (sitting/walking/hand movement) for each generated signal was also selected using uniform random variables, meaning that the share of each noise type was approximately the same in the generated dataset. The noise amplitude parameter range (see Table 1) ensured that the generated dataset contained signals ranging from completely clean signals (i.e., signals recorded under perfect conditions) to very noisy signals. The assumed breathing frequency range of [0.15, 0.4] Hz translates into 9–24 breaths per minute, a range that approximately covers the assumed activities, from resting to walking at a comfortable pace.

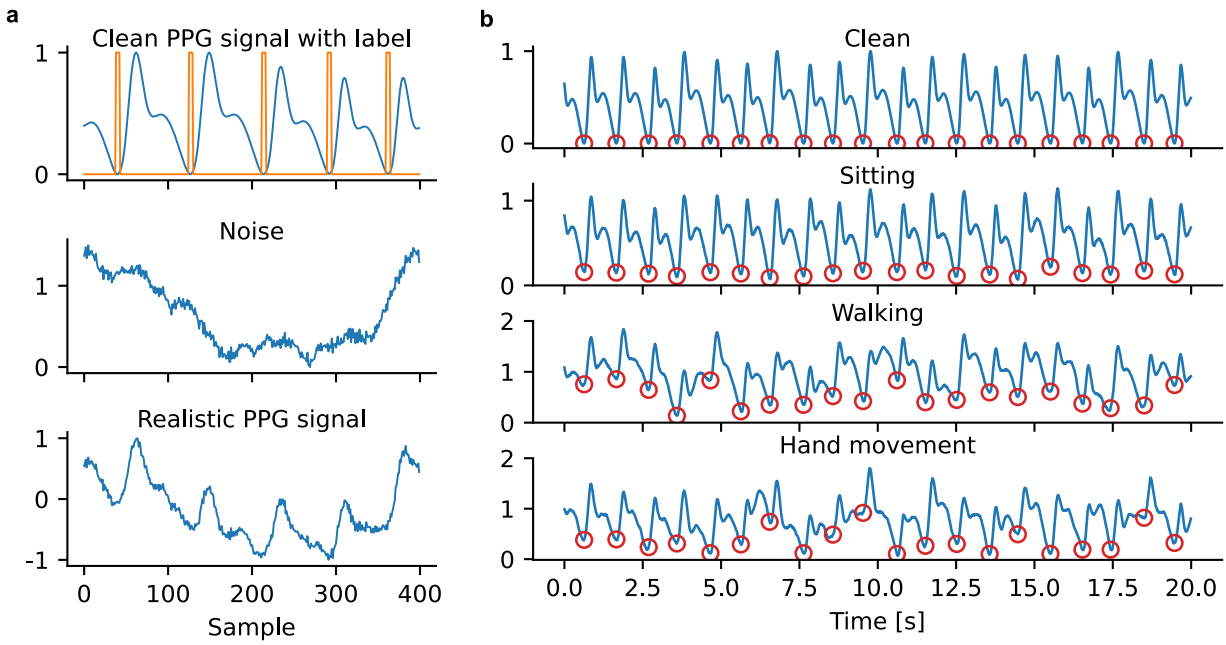

Fig. 5. Examples of synthetic PPG signals. a. The top figure shows a generated clean PPG signal together with the label signal denoting the foot locations. The middle figure shows the generated noise signal, and the bottom figure shows the final realistic PPG signal created by summing the clean PPG and noise signals together. The final signal has been normalized to the range [−1, 1]. b. Examples of generated longer (20 s) synthetic PPG signals with the different noise types used in the article. The underlying clean PPG signal is the same in each figure. The noise realizations have been generated with the same random seed to guarantee similar noise PSD sampling. The signal representing a measurement taken while sitting clearly has the lowest noise level, while the other two signals show significant motion-related noise.

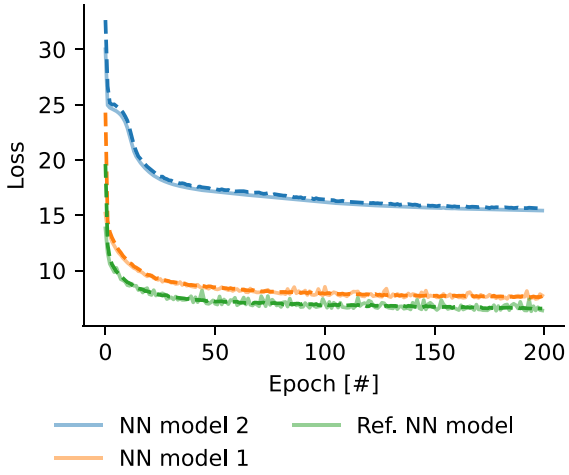

Fig. 6. Training and validation losses of the CNN models. Training values have been plotted with dashed lines while the validation values have been plotted with solid lines with alpha colorings.

After the preprocessing steps, the signals, together with the automatically created label arrays (where each waveform foot is marked by five subsequent ones, the rest being zeros), were used as input data in the NN model training. The training and validation losses for the NN models are presented in Fig. 6. Overall, the training and validation losses are very close to each other because the underlying synthetic data distributions are the same for the two datasets. This is because all the synthetic data were randomly generated in the same way. The NN model 1 is close to the reference model with slightly larger losses, while the tiny NN model 2 has significantly larger losses than the two larger models. The losses on the test datasets were 1.79, 4.16 and 1.33 for the NN model 1, the NN model 2 and the reference NN model, respectively.

The performance comparison between the NN models and the other foot detection algorithms has been presented in Table 3. The reference NN model is clearly the best foot detection method with the highest count of top values. Interestingly, though, the two smaller NN models do not perform significantly worse than the larger reference model. Furthermore, the tiny NN model 2 performs better than the larger NN model 1, and the model is almost as good as the reference NN model in terms of F1 score. The small differences in how well all the models perform on real data is quite unexpected based on the differences in validation and test losses. The performance of the other, non-ML, methods tends to be worse. The AMPD algorithm's problem in finding feet close to the end points of a signal is clearly visible in the recall values. The performance-improved MSPTD algorithm suffers from the same problem. However, the modified AMPD shows better recall values, meaning that it is able to better detect the feet. However, this seems to come at the cost of precision. The HeartPy algorithm is somewhere in between the ML-based and AMPD-based algorithms.

The performance of the methods with respect to HR calculation is quite in line with the foot detection results. The reference NN model is the best over all the measurements followed by the NN model 2 and Heartpy. The Heartpy algorithm performs surprisingly well presumably due to the algorithm's built-in feature to compute and evaluate instantaneous HR along with peak-peak interval rejection threshold. The ML-based methods are rather close to each other and in general they tend to be close to the top with the exception of the sitting measurements, where the more traditional methods are better. The very simple autocorrelation method is the worst, except for the sitting measurements. However, the differences between the models, excluding the worst, are surprisingly small. With regards the NN models, the results, interestingly, do not decrease significantly even if the signal length used in the calculations is decreased from four to three or even two seconds. The results of this experimentation are presented in the supplementary information.

Fig. 7 shows an example of a challenging signal from a measurement where the study participant was walking. The non-ML algorithms tend to have more false positives (noise interpreted as feet) and false negatives (missed feet) than the NN model-based foot detectors. The raw outputs of the NN model (probability arrays) are quite similar, with some differences in peak locations and amplitudes. The postprocessing

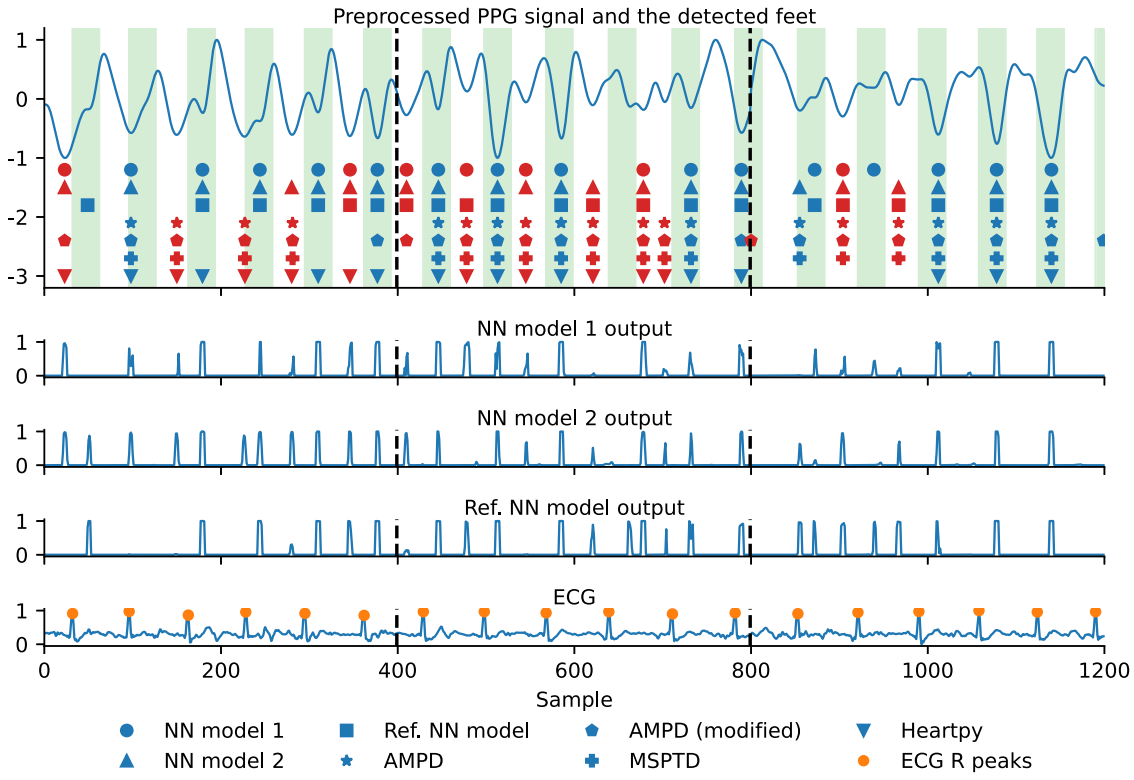

Fig. 7. An example of a challenging signal during a measurement where the test subject was moving. The blue marker color indicates a correctly detected foot, whereas the red marker color indicates a falsely detected foot. If a marker is missing from a green bar (which marks the period after an ECG R peak where a PPG foot is expected to occur), it means that the detection method has failed to identify a foot. The vertical dashed black lines mark the boundaries of the 4-s (400 samples) long periods.

Table 3
Summary of the results of the algorithm comparison. The table presents the mean precision, recall, F1 scores and HR mean absolute error (MAE) relative to the ECG for each foot detection algorithm and for each measurement type. The best and worst values in each column have been stylized with bold black and bold red fonts, respectively. The mean values have been calculated over the entire measurement length (i.e. about 2 min).

	Sitting				Walking				Hand movement				All measurements			
Algorithm	Precision	Recall	F1 score	HR MAE	Precision	Recall	F1 score	HR MAE	Precision	Recall	F1 score	HR MAE	Precision	Recall	F1 score	HR MAE
NN model 1	0.94	0.98	0.96	**3.1**	0.69	**0.70**	0.69	15.7	0.73	0.62	0.67	15.5	0.80	0.78	0.78	11.0
NN model 2	0.96	0.98	**0.97**	2.0	0.69	0.69	0.69	15.9	0.73	0.63	0.67	**15.4**	0.80	0.78	**0.79**	10.7
Ref. NN model	0.95	0.98	**0.97**	2.5	0.70	**0.72**	**0.71**	**13.7**	0.74	0.64	**0.68**	15.5	0.80	**0.79**	**0.79**	**10.2**
AMPD	**0.99**	**0.80**	**0.89**	1.3	**0.72**	**0.53**	**0.61**	17.9	**0.75**	**0.50**	**0.60**	16.6	**0.83**	**0.62**	**0.71**	11.4
AMPD (modified)	**0.91**	**0.99**	0.95	2.0	0.68	0.66	0.67	18.7	0.71	**0.65**	**0.68**	16.8	**0.77**	0.78	0.77	12.0
MSPTD	**0.99**	**0.80**	**0.89**	1.3	0.70	0.56	0.62	17.0	**0.75**	0.54	0.62	15.6	0.82	0.64	0.72	10.8
Heartpy	0.98	0.89	0.93	**1.2**	**0.65**	0.62	0.63	16.5	**0.67**	0.60	0.63	15.9	0.78	0.71	0.74	10.7
Autocorrelation	-	-	-	2.7	-	-	-	**20.2**	-	-	-	**19.0**	-	-	-	**13.4**

algorithm with probability peak amplitude and distance thresholds seems to equalize the NN models to some extent. Nevertheless, the relatively good performance of the NN models implies that the models trained with synthetic data work well even with real data. The fact that even the tiny NN model works so well suggests that there is a limited amount of information to learn from 1-dimensional, rather tightly filtered, and short PPG signals.

The developed device was easily able to run all the NN models due to their relatively small sizes. However, there were quite significant differences in model inference times. The large reference NN model takes approximately 109 ms to perform inference, whereas the smaller NN models 1 & 2 take only approximately 22 ms and 3 ms, respectively. Thus, the reference NN model takes significantly more computational time, approximately 36 times, than the tiny NN model 2 and, as seen earlier, without a significant performance difference in foot detection and HR calculation. In a battery-powered wearable device, the two smaller models could, hence, be more suitable to minimize current consumption. In its current state, without any power optimizations, the device draws approximately 185 mA of current while in the measurement state, with a single roughly 2-minute measurement consuming approximately 6.6 mAh.

The presented synthetic signal generator applied the principles of generating synthetic ECG signals to PPG signals along with a method to add realistic noise to the generated signals. Instead of summing directly two Gaussian functions, as in Tang, Chen, Ward et al. (2020), the model sums two or more Gaussian function derivatives together and then uses numerical integration to construct the final signal. This approach allows to create continuous and smooth signals without additional smoothing interpolation between separate waveforms, as in Tang, Chen, Allen et al. (2020). Multiple Gaussian functions have been used in PPG waveform decomposition analysis (single PPG waveform) (Wang et al., 2013) but the presented model also creates continuous signals.

Overall, synthetic data generators add a clear benefit to data-intensive ML, since realistic data with accurate labels can be generated quickly in large amounts for free (if the costs related to computation are ignored). The generated data can be used alone (as in this article) or augmented to a real dataset. In this article, we used synthetic data to train NN models for the relatively straightforward task of calculating

HR. However, the system could be extended to more complex health-related monitoring tasks such as predicting arrhythmia and estimating blood pressure and sleep quality. Of course, synthetic data is only as good as the underlying model, and thus different diseases or physiological conditions could prove difficult to model accurately without adjusting the model, for example, in the case of arrhythmia.

This study emphasized the privacy preservation aspect of health data, however, data security is also of significance importance in medical/healthcare IoT applications (Chanal & Kakkasageri, 2020). Without data security, data privacy cannot exist and therefore data security is a prerequisite for any system handling health-related data. Significant attack vectors in medical devices, such as firmware vulnerabilities, communication protocols (BLE in our device), and applications (the mobile application in our case) (Hernández-Álvarez et al., 2022), should be taken into account to reduce the risk of security breaches. The list of common security pitfalls to avoid in IoT systems maintained by The Open Worldwide Application Security Project (OWASP) (The Open Worldwide Application Security Project, 2018) should also be considered when building secure wearable devices and related systems. Future research should address these aspects to ensure the security of the system.

5. Conclusion

Motivated by difficulties in gathering robust real-world datasets, we demonstrated a system where synthetic data generator together with wearable edge machine learning is used to preserve privacy without the risk of leaking sensitive information. In our example, the system consisted of a PPG signal generator, CNN models trained with synthetic data, and a wearable device utilizing the CNN models to compute HR in real time. The results demonstrated that even a tiny CNN model trained with purely synthetic data performs well on unseen real-world data. Future studies could explore the use of the synthetic PPG model in different disease states together with more sophisticated NN models to detect diseases in a portable form factor.

CRediT authorship contribution statement

Jukka-Pekka Sirkiä: Designed the research, Developed the device, Software, Analyzed the data, Performed the research, Obtained the Funding, Writing – original draft. **Tuukka Panula:** Designed the research, Obtained the Funding, Writing – original draft. **Matti Kaisti:** Designed the research, Obtained the Funding, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The data that support the findings of this study are available on a reasonable request from the corresponding author. Python code is available on GitHub: https://github.com/jpsirk/synthetic-photoplethysmograms.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.eswa.2023.121523.

References

Acharya, U. R., Fujita, H., Oh, S. L., Hagiwara, Y., Tan, J. H., & Adam, M. (2017). Application of deep convolutional neural network for automated detection of myocardial infarction using ECG signals. *Information Sciences*, *415–416*, 190–198. http://dx.doi.org/10.1016/j.ins.2017.06.027.

Aschbacher, K., Yilmaz, D., Kerem, Y., Crawford, S., Benaron, D., Liu, J., Eaton, M., Tison, G. H., Olgin, J. E., Li, Y., & Marcus, G. M. (2020). Atrial fibrillation detection from raw photoplethysmography waveforms: A deep learning application. *Heart Rhythm O2*, *1*(1), 3–9. http://dx.doi.org/10.1016/j.hroo.2020.02.002.

Bentzen, H. B., Castro, R., Fears, R., Griffin, G., ter Meulen, V., & Ursin, G. (2021). Remove obstacles to sharing health data with researchers outside of the European union. *Nature Medicine*, *27*(8), 1329–1333. http://dx.doi.org/10.1038/s41591-021-01460-0.

Bishop, S. M., & Ercole, A. (2018). Multi-scale peak and trough detection optimised for periodic and quasi-periodic neuroscience data. In T. Heldt (Ed.), *Intracranial pressure & neuromonitoring XVI* (pp. 189–195). Cham: Springer International Publishing.

Chanal, P. M., & Kakkasageri, M. S. (2020). Security and privacy in IoT: A survey. *Wireless Personal Communications*, *115*(2), 1667–1693. http://dx.doi.org/10.1007/s11277-020-07649-9.

Charlton, P. H., Kotzen, K., Mejía-Mejía, E., Aston, P. J., Budidha, K., Mant, J., Pettit, C., Behar, J. A., & Kyriacou, P. A. (2022). Detecting beats in the photoplethysmogram: Benchmarking open-source algorithms. *Physiological Measurement*, *43*(8), Article 085007. http://dx.doi.org/10.1088/1361-6579/ac826d.

Chen, J., Jiang, M., Zhang, X., da Silva, D. S., de Albuquerque, V. H. C., & Wu, W. (2023). Implementing ultra-lightweight co-inference model in ubiquitous edge device for atrial fibrillation detection. *Expert Systems with Applications*, *216*, Article 119407. http://dx.doi.org/10.1016/j.eswa.2022.119407.

Chen, R. J., Lu, M. Y., Chen, T. Y., Williamson, D. F. K., & Mahmood, F. (2021). Synthetic data in machine learning for medicine and healthcare. *Nature Biomedical Engineering*, *5*(6), 493–497. http://dx.doi.org/10.1038/s41551-021-00751-8.

Das, S., Pal, S., & Mitra, M. (2016). Real time heart rate detection from PPG signal in noisy environment. In *2016 international conference on intelligent control power and instrumentation* (pp. 70–73). http://dx.doi.org/10.1109/ICICPI.2016.7859676.

Elgendi, M., Fletcher, R., Liang, Y., Howard, N., Lovell, N. H., Abbott, D., Lim, K., & Ward, R. (2019). The use of photoplethysmography for assessing hypertension. *npj Digital Medicine*, *2*(1), 60. http://dx.doi.org/10.1038/s41746-019-0136-7.

Hernández-Álvarez, L., Bullón Pérez, J. J., Batista, F. K., & Queiruga-Dios, A. (2022). Security threats and cryptographic protocols for medical wearables. *Mathematics*, *10*(6), 886. http://dx.doi.org/10.3390/math10060886.

Hou, D., Raymond Hou, M., & Hou, J. (2020). ECG beat classification on edge device. In *2020 IEEE international conference on consumer electronics* (pp. 1–4). http://dx.doi.org/10.1109/ICCE46568.2020.9043116.

Jordon, J., Szpruch, L., Houssiau, F., Bottarelli, M., Cherubin, G., Maple, C., Cohen, S. N., & Weller, A. (2022). Synthetic data – what, why and how? http://dx.doi.org/10.48550/ARXIV.2205.03257, arXiv.

Kaisti, M., Laitala, J., & Airola, A. (2021). Training neural networks with synthetic electrocardiograms. http://dx.doi.org/10.48550/ARXIV.2111.06175, arXiv.

Kazemi, K., Laitala, J., Azimi, I., Liljeberg, P., & Rahmani, A. M. (2022). Robust PPG peak detection using dilated convolutional neural networks. *Sensors*, *22*(16), http://dx.doi.org/10.3390/s22166054.

Kiyasseh, D., Tadesse, G. A., Nhan, L. N. T., Van Tan, L., Thwaites, L., Zhu, T., & Clifton, D. (2020). Plethaugment: GAN-based PPG augmentation for medical diagnosis in low-resource settings. *IEEE Journal of Biomedical and Health Informatics*, *24*(11), 3226–3235. http://dx.doi.org/10.1109/JBHI.2020.2979608.

Laitala, J., Jiang, M., Syrjälä, E., Naeini, E. K., Airola, A., Rahmani, A. M., Dutt, N. D., & Liljeberg, P. (2020). Robust ECG R-peak detection using LSTM. In *Proceedings of the 35th annual ACM symposium on applied computing* (pp. 1104–1111). New York, NY, USA: Association for Computing Machinery, http://dx.doi.org/10.1145/3341105.3373945.

Lee, J., Matsumura, K., Yamakoshi, K.-i., Rolfe, P., Tanaka, S., & Yamakoshi, T. (2013). Comparison between red, green and blue light reflection photoplethysmography for heart rate monitoring during motion. In *2013 35th annual international conference of the IEEE engineering in medicine and biology society* EMBC, (pp. 1724–1727). http://dx.doi.org/10.1109/EMBC.2013.6609852.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022). A ConvNet for the 2020s. http://dx.doi.org/10.48550/ARXIV.2201.03545, arXiv.

Martín-Martínez, D., Casaseca-de-la Higuera, P., Martín-Fernández, M., & Alberola-López, C. (2013). Stochastic modeling of the PPG signal: A synthesis-by-analysis approach with applications. *IEEE Transactions on Biomedical Engineering*, *60*(9), 2432–2441. http://dx.doi.org/10.1109/TBME.2013.2257770.

Mazumder, O., Banerjee, R., Roy, D., Bhattacharya, S., Ghose, A., & Sinha, A. (2022). Synthetic PPG signal generation to improve Coronary artery disease classification: Study with physical model of cardiovascular system. *IEEE Journal of Biomedical and Health Informatics*, *26*(5), 2136–2146. http://dx.doi.org/10.1109/JBHI.2022.3147383.

McSharry, P., Clifford, G., Tarassenko, L., & Smith, L. (2003). A dynamical model for generating synthetic electrocardiogram signals. *IEEE Transactions on Biomedical Engineering*, *50*(3), 289–294. http://dx.doi.org/10.1109/TBME.2003.808805.

Merenda, M., Porcaro, C., & Iero, D. (2020). Edge machine learning for AI-enabled IoT devices: A review. *Sensors*, *20*(9), http://dx.doi.org/10.3390/s20092533.

Mukkamala, R., Hahn, J.-O., Inan, O. T., Mestha, L. K., Kim, C.-S., Töreyin, H., & Kyal, S. (2015). Toward ubiquitous blood pressure monitoring via pulse transit time: Theory and practice. *IEEE Transactions on Biomedical Engineering*, *62*(8), 1879–1901. http://dx.doi.org/10.1109/TBME.2015.2441951.

Murshed, M. G. S., Murphy, C., Hou, D., Khan, N., Ananthanarayanan, G., & Hussain, F. (2021). Machine learning at the network edge: A survey. *ACM Computing Surveys*, *54*(8), http://dx.doi.org/10.1145/3469029.

Naudet, F., Sakarovitch, C., Janiaud, P., Cristea, I., Fanelli, D., Moher, D., & Ioannidis, J. P. A. (2018). Data sharing and reanalysis of randomized controlled trials in leading biomedical journals with a full data sharing policy: Survey of studies published in the BMJ and PLOS medicine. *BMJ*, *360*, http://dx.doi.org/10.1136/bmj.k400.

Rajotte, J.-F., Bergen, R., Buckeridge, D. L., El Emam, K., Ng, R., & Strome, E. (2022). Synthetic data as an enabler for machine learning applications in medicine. *iScience*, *25*(11), Article 105331. http://dx.doi.org/10.1016/j.isci.2022.105331.

Sannino, G., & De Pietro, G. (2018). A deep learning approach for ECG-based heartbeat classification for arrhythmia detection. *Future Generation Computer Systems*, *86*, 446–455. http://dx.doi.org/10.1016/j.future.2018.03.057, URL https://www.sciencedirect.com/science/article/pii/S0167739X17324548.

Scholkmann, F., Boss, J., & Wolf, M. (2012). An efficient algorithm for automatic peak detection in noisy periodic and quasi-periodic signals. *Algorithms*, *5*(4), 588–603. http://dx.doi.org/10.3390/a5040588.

Shin, H. S., Lee, C., & Lee, M. (2009). Adaptive threshold method for the peak detection of photoplethysmographic waveform. *Computers in Biology and Medicine*, *39*(12), 1145–1152. http://dx.doi.org/10.1016/j.compbiomed.2009.10.006.

Śmigiel, S., Pałczyński, K., & Ledziński, D. (2021). Deep learning techniques in the classification of ECG signals using R-peak detection based on the PTB-XL dataset. *Sensors*, *21*(24), http://dx.doi.org/10.3390/s21248174.

Sološenko, A., Petrėnas, A., Marozas, V., & Sörnmo, L. (2017). Modeling of the photoplethysmogram during atrial fibrillation. *Computers in Biology and Medicine*, *81*, 130–138. http://dx.doi.org/10.1016/j.compbiomed.2016.12.016.

Sun, Z., Junttila, J., Tulppo, M., Seppanen, T., & Li, X. (2022). Non-contact atrial fibrillation detection from face videos by learning systolic peaks. *IEEE Journal of Biomedical and Health Informatics*, *26*(9), 4587–4598. http://dx.doi.org/10.1109/jbhi.2022.3193117.

Tang, Q., Chen, Z., Allen, J., Alian, A., Menon, C., Ward, R., & Elgendi, M. (2020). PPGSynth: An innovative toolbox for synthesizing regular and irregular photoplethysmography waveforms. *Frontiers in Medicine*, *7*, http://dx.doi.org/10.3389/fmed.2020.597774.

Tang, Q., Chen, Z., Ward, R., & Elgendi, M. (2020). Synthetic photoplethysmogram generation using two Gaussian functions. *Scientific Reports*, *10*(1), 13883. http://dx.doi.org/10.1038/s41598-020-69076-x.

Tereshchenko, L. G., & Josephson, M. E. (2015). Frequency content and characteristics of ventricular conduction. *Journal of Electrocardiology*, *48*(6), 933–937. http://dx.doi.org/10.1016/j.jelectrocard.2015.08.034.

The Open Worldwide Application Security Project (2018). OWASP Internet of things | OWASP foundation. https://owasp.org/www-project-internet-of-things/. (Accessed: 17 August 2023).

Timmer, J., & Koenig, M. (1995). On generating power law noise. *Astronomy and Astrophysics*, *300*, 707–710.

van Gent, P., Farah, H., van Nes, N., & van Arem, B. (2019). HeartPy: A novel heart rate algorithm for the analysis of noisy signals. *Transportation Research Part F: Traffic Psychology and Behaviour*, *66*, 368–378. http://dx.doi.org/10.1016/j.trf.2019.09.015.

Wang, L., Xu, L., Feng, S., Meng, M. Q.-H., & Wang, K. (2013). Multi-Gaussian fitting for pulse waveform using weighted least squares and multi-criteria decision making method. *Computers in Biology and Medicine*, *43*(11), 1661–1672. http://dx.doi.org/10.1016/j.compbiomed.2013.08.004.

Welch, P. (1967). The use of fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms. *IEEE Transactions on Audio and Electroacoustics*, *15*(2), 70–73. http://dx.doi.org/10.1109/TAU.1967.1161901.

Wu, C.-J., Brooks, D., Chen, K., Chen, D., Choudhury, S., Dukhan, M., Hazelwood, K., Isaac, E., Jia, Y., Jia, B., Leyvand, T., Lu, H., Lu, Y., Qiao, L., Reagen, B., Spisak, J., Sun, F., Tulloch, A., Vajda, P., ... Zhang, P. (2019). Machine learning at Facebook: Understanding inference at the edge. In *2019 IEEE international symposium on high performance computer architecture* (pp. 331–344). http://dx.doi.org/10.1109/HPCA.2019.00048.

Further reading

Liang, Y., Chen, Z., Ward, R., & Elgendi, M. (2018). Photoplethysmography and deep learning: Enhancing hypertension risk stratification. *Biosensors*, *8*(4), http://dx.doi.org/10.3390/bios8040101.