POLITECNICO DI TORINO

Master's Degree in Data Science and Engineering

Master's Degree Thesis

Spiking Neural Networks in PPG-Based Blood Pressure Estimation

Supervisors

Prof. Alessio BURRELLO

Prof. Daniele JAHIER PAGLIARI

Prof. Vittorio FRA

Prof. Elisa DONATI

Francesco CARLUCCI

Candidate

Shakti RATHORE

March 2026

Abstract

Continuous and non-invasive blood pressure (BP) monitoring is a challenging problem, as typical methods are intermittent, not suitable for continuous use, and often require either clinical supervision or user interaction. In recent years, machine learning and deep learning approaches have been employed for this task, achieving promising results using photoplethysmography (PPG), but often at the cost of high model complexity and limited suitability for low-power or embedded deployment. Spiking Neural Networks (SNNs), inspired by biological neural processing, offer an interesting alternative due to their event-driven nature and potential for energy-efficient implementation on neuromorphic hardware. In this work, a lightweight SNN architecture for BP estimation is proposed and evaluated across four public datasets: BCG, Sensors, PPGBP, and UCI. The primary objective is **not to surpass state of the art models in accuracy**, but to investigate whether SNNs can achieve comparable performance while reaching lower complexity and improved energy efficiency. Multiple architectural variants, neuron models, and biologically inspired mechanisms are explored, including different types of spiking neurons, winner-take-all dynamics, and sparse connectivity. From an architectural perspective, several input–output configurations are also investigated, ranging from networks that take handcrafted features extracted from the PPG signal as input and use neuronal firing rates for inference, to models that directly process raw PPG signals and perform prediction based on the membrane potentials of the output neurons. Experimental results show that the proposed model achieves competitive accuracy compared to state-of-the-art machine learning and deep learning approaches, and ranks among the top-performing models in experiments conducted on the BCG dataset. Beyond prediction accuracy, the proposed SNN demonstrates substantial efficiency advantages. The model requires only 4,841 trainable parameters, which is orders of magnitude lower than the most accurate deep learning models, resulting in a low memory footprint. Neurobench evaluation further reveals an average activation sparsity exceeding 80%, highlighting the model's potential for energy-efficient inference. These results highlight the potential of SNNs for low-power BP estimation, particularly in scenarios where computational resources are limited. Finally, a preliminary deployment on neuromorphic hardware

(Dynapse) is investigated. While direct one-to-one mapping from simulation to hardware proved challenging due to weight quantization and parameter sensitivity, the study identifies key limitations and future research directions for bridging the simulation-to-hardware gap. Overall, this work demonstrates that spiking neural networks can provide a viable and efficient alternative to conventional deep learning models for non-invasive blood pressure estimation, particularly in energy-constrained and embedded applications.

Acknowledgements

For this work in particular, I am grateful to all my supervisors. They not only provided excellent guidance and direction, but were also consistently supportive, understanding, and remarkably quick in their responses. I am aware that this level of attention and dedication to students is not always something to be taken for granted.

I would like to thank my family for giving me the opportunity to pursue my studies to this level, especially considering the path they have taken in their own lives.

I would also like to thank my friends, those who are still here, and also those I met along the way who have taken different paths. They have definitely helped make everything more fun.

Finally, I would like to express gratitude for the fortune I have had. I believe it is always important to remember the privileges and opportunities we are given. Hard work certainly matters, but being born in the right place and at the right time in the world also makes a significant difference.

Table of Contents

List of Tables

List of Figures

Acronyms

BP
blood pressure

PPG
photoplethysmographic signal

ABP
arterial blood pressure

SP
systolic pressure

DP
diastolic pressure

ICU
intensive care unit

SNN
spiking neural network

AI
artificial intelligence

ML
machine learning

CNN
convolutional neural network

ReLU

rectified linear unit

LIF

leaky integrate-and-fire neuron

BPTT

backpropagation through time

OH

orthostatic hypotension

ABPM

ambulatory blood pressure monitoring

Chapter 1

Introduction

Measuring blood pressure (BP) can be a highly effective and important tool for assessing a person's health status and identifying potential future issues before they arise. For instance, consistently high blood pressure over time can significantly increase the risk of cardiovascular diseases. Traditionally, blood pressure is measured using cuff-based devices which are considered accurate but come with several drawbacks. These methods are typically intermittent, not suitable for continuous use, and often require either clinical supervision or user interaction. This makes them impractical for real-time, long-term health tracking in everyday life. In recent years, the growing availability and adoption of wearable devices, such as smartwatches, have opened new opportunities for non-invasive and continuous physiological monitoring. These devices, already equipped with multiple sensors are now being leveraged for more complex health metrics like blood pressure estimation. In parallel, the rapid advancement of artificial intelligence, particularly in the fields of machine learning and deep learning, has provided the tools necessary to model and interpret these physiological signals with increasing accuracy and robustness. For this reason, several non-invasive techniques have recently emerged. One of them is blood pressure inference through PPG signals. The photoplethysmographic (PPG) signal captures changes in blood volume within the microvascular bed of tissue by using a light-based optical sensor. It can be acquired non-invasively through commonly worn devices making it particularly suitable for everyday use. In addition to its convenience, the PPG signal allows for continuous, long-term monitoring without the involvement of medical personnel. However, the correlation between PPG and BP is not one-to-one. This is partly due to the limited accuracy of real-world measurements obtained from sensors in wearable devices. Therefore, the goal is to analyze and extract multiple features from the PPG signal and combine them to predict blood pressure values. The prediction focuses on the two blood pressure (BP) components: systolic pressure (SP) and diastolic pressure (DP), which together describe the full BP profile.

Several studies have explored this problem, evaluating the performance of different machine and deep learning models using various input features and output targets. This has helped establish a solid baseline for comparison, taking the work presented in [1] as a benchmark. In the study, four datasets were used, and the same datasets were adopted in this research to enable consistent comparison. Sensors, UCI, BCG, and PPGBP form standard benchmarking datasets in cuff-less BP research, with UCI and Sensors derived from the MIMIC IC waveform databases, BCG collected via Finometer-based acquisition in experimental settings, and PPGBP representing PPG with peripheral BP references. s The datasets were acquired using slightly different configurations. Some records are 5 seconds long with a sampling frequency of 125 Hz or 1000 Hz, while others are 2.1 seconds long sampled at 1000 Hz. Demographic information and the number of patients also vary between datasets, making them more representative of real-world scenarios. The data were subsequently preprocessed as described in [1], with the goal of cleaning the signals, aligning the PPG and ABP recordings, and extracting a set of features across multiple categories. These included point/time-based features, frequency-based features, and statistical features.

Standard neural networks, although effective, can face limitations in terms of computational power requirements. In contrast, the human brain, though less powerful than modern CPUs, can perform highly complex tasks with remarkable energy efficiency, consuming only around 12 to 20 W. This observation inspired the main contribution of this thesis. The brain has led to the development of Spiking Neural Networks (SNNs), a third generation of neural networks designed to mimic biological neurons. Unlike traditional networks that process continuous values, SNNs work by transmitting spikes and operate using reduced-precision synaptic weights, enabling significantly greater energy efficiency. This efficiency is crucial in applications like continuous BP monitoring, where algorithms must run on mobile and compact devices with limited battery life. Thus, the aim of this thesis was not to improve the accuracy of BP estimation via PPG per se, but rather to demonstrate that SNNs can achieve comparable performance to earlier generations of neural networks, while being more energy-efficient.

To achieve their full efficiency and effectiveness, Spiking Neural Networks (SNNs) must be executed on specialized hardware optimized for their unique computational model. At present, several experimental chips exist that support the implementation and efficient deployment of these networks. Therefore, as part of this thesis, the theoretical models were also tested on one such chip: the Dynapse-1, a neuromorphic processor developed by the Institute of Neuroinformatics [2]. This allowed for an evaluation of the practical feasibility of this SNN on hardware designed to emulate biological neural systems.

To conclude, the main contributions of this work can be summarized in two key points:

- Evaluation of different types of Spiking Neural Networks (SNNs): Various architectures were tested, differing in neuron models, connectivity patterns, and scale. Several input–output configurations were explored, including both raw signal and extracted features as inputs. Depending on the configuration, the focus was on different types of outputs, such as firing rates, membrane potentials, or spike sequences aiming to preserve also the temporal characteristics;
- Hardware implementation of the best-performing model: The most effective SNN identified during simulations was mapped onto a neuromorphic hardware chip to evaluate whether a direct one-to-one implementation, with partial fine-tuning of certain parameters, could already operate effectively on currently available chips;

The remainder of this thesis is structured as follows:

- Chapter 2, reviews the background on deep learning models across different generations, introduces the PPG signal, and provides an overview of blood pressure estimation methods;
- Chapter 3, presents related work, covering both the use of SNNs for biosignal processing and general applications of PPG, as well as an introduction to the neuromorphic chip used for deployment;
- Chapter 4, describes the methodology, introducing in detail the tested architectures and the strategies adopted for data handling and training;
- Chapter 5, discusses the experimental results obtained with the proposed methods.
- Chapter 6, presents final reflections on the findings and discusses possible directions for future research;

Chapter 2

Background

2.1 Deep Learning Networks Evolution

In recent years, there has been an explosion in applications related to artificial intelligence, driven by the ongoing digitalization of the world and the resulting availability of increasingly large and high-quality datasets. At the same time, advances in models and algorithms have significantly enhanced the capabilities of AI systems. Today, with the right data, it is possible to train machine learning algorithms to perform a wide range of tasks with high precision, such as regression, classification, or anomaly detection. However, traditional machine learning methods often rely on manual feature extraction and struggle to scale with complex, high-dimensional, or unstructured data. To address more complex problems, and to enable models to better capture the underlying structure within data, deep learning techniques are required. These approaches automatically learn multiple levels of abstraction, making them especially effective for domains such as computer vision, speech recognition, and natural language processing. Deep learning is a subfield of machine learning that focuses on models composed of multiple layers of interconnected computational units, called artificial neurons. These models differ significantly from traditional shallow architectures by their ability to extract hierarchical representations, such as low-level features like edges in early layers, and progressively more abstract concepts in deeper layers. This layered representation is what makes deep learning highly effective for unstructured data such as images, text, or audio. Furthermore, deep learning models have the capacity to generalize across tasks through transfer learning, and have seen widespread deployment in domains such as autonomous driving, medical diagnostics, and natural language generation.

The conceptual foundations of neural networks can be traced back to the work of McCulloch and Pitts (1943), who introduced the first formal model of an

artificial neuron. Their neuron computed a binary output by summing weighted binary inputs and applying a threshold: if the total exceeded a certain value, the neuron "fired" (output 1), otherwise, it remained inactive. Building upon this idea, Frank Rosenblatt (1958) [3] proposed the perceptron, which retained the threshold activation function and introduced a learning rule to adjust the connection weights based on classification error. This made the perceptron the first trainable neural network model. While still constrained to binary output and linear decision boundaries, Rosenblatt's model shifted the focus from static computation to adaptive learning. As stated in [4], such networks, given digital input and output, are theoretically capable of computing any Boolean function. The introduction of second-generation neural networks marked a fundamental evolution in the capabilities of artificial learning systems. Unlike first-generation models, such as the McCulloch-Pitts neuron or Rosenblatt's perceptron, which operated on binary input-output values using threshold-based activation, second-generation networks adopt continuous, differentiable activation functions like the sigmoid, tanh, and ReLU (Rectified Linear Unit). This shift enabled the use of gradient-based optimization techniques, most notably the backpropagation algorithm, allowing networks to be trained efficiently even when deep. As shown in [4], these models are universal approximators in the analog domain, capable of representing any continuous function to arbitrary precision given sufficient capacity.
To illustrate how backpropagation works in practice, we consider a Convolutional Neural Network (CNN), a widely adopted architecture for processing image data. CNNs consist of layers that perform discrete convolutions, typically followed by non-linear activation functions (e.g., ReLU), pooling layers, and fully connected layers. Given an input image $\mathbf{X} \in \mathbb{R}^{H \times W}$, a convolutional layer applies a kernel $\mathbf{K} \in \mathbb{R}^{k \times k}$, producing a feature map $\mathbf{Y}$ defined by:

$$\mathbf{Y}_{i,j} = \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} \mathbf{K}_{m,n} \cdot \mathbf{X}_{i+m,j+n} \tag{2.1}$$

The result is then passed through an activation function such as ReLU:

$$\text{ReLU}(x) = \max(0, x) \tag{2.2}$$

During training, the network's output $\hat{\mathbf{y}}$ is compared to the true label $\mathbf{y}$ using a loss function, such as categorical cross-entropy:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{2.3}$$

The backpropagation algorithm computes the gradient of the loss with respect to each parameter θ using the chain rule:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z} \cdot \frac{\partial z}{\partial \theta} \tag{2.4}$$

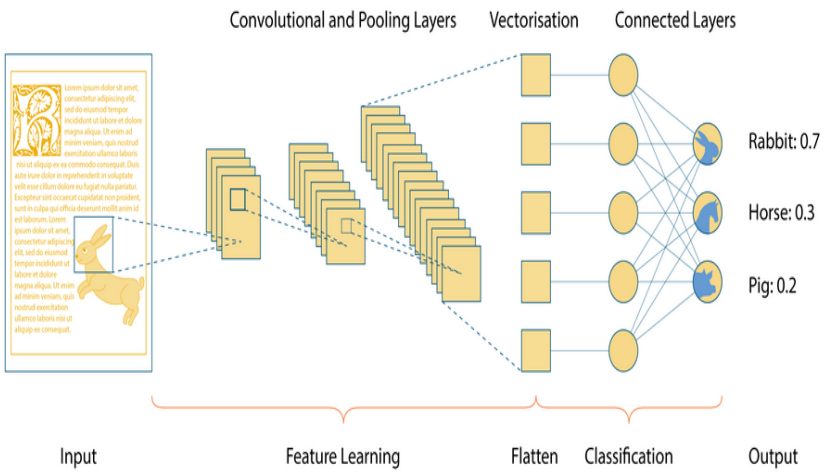


Figure 2.1: Graphical representation of a CNN processing an image
Source: Irisbox (2025), Wikimedia Commons, CC BY 4.0.

This example illustrates how second-generation neural networks, especially CNNs, make use of differentiable activations and gradient-based training to learn complex input-output mappings. Through repeated application of backpropagation and gradient descent, CNNs gradually adjust their parameters to optimize performance on tasks such as image classification. Neural networks based on this second generation have achieved remarkable performance in terms of error rates across a wide range of applications. However, as these architectures have grown increasingly complex, their number of parameters, and consequently their energy consumption, has also increased significantly. This makes their deployment challenging in certain scenarios, such as on mobile or edge devices that rely on battery power. In contrast, the human brain, despite being much slower and constrained in terms of raw computational capacity compared to modern CPUs, can perform highly complex tasks with remarkably low energy consumption, typically between 12 and 20 W. This biological efficiency has inspired the development of models that aim to more closely mimic the functioning of real neural systems. This gave rise to the third generation of neural networks, known as Spiking Neural Networks (SNNs). SNNs incorporate the concept of spatio-temporal coding, where information is not just conveyed by the activation level of a neuron, but also by the precise timing of its spikes. Unlike traditional units with continuous outputs, spiking neurons generate discrete events over time, a mechanism that mirrors how biological neurons communicate through

action potentials. As demonstrated by Maass [4], networks of spiking neurons are not only computationally more efficient, but also strictly more powerful in terms of function approximation and information encoding than traditional threshold or sigmoid units, even with fewer neurons. Furthermore, under certain constraints, SNNs can simulate the behavior of classical neural networks while also enabling computations that rely on temporal patterns. These properties make SNNs highly promising for applications requiring energy efficiency and low-latency response, such as neuromorphic hardware, robotics, and sensory processing.

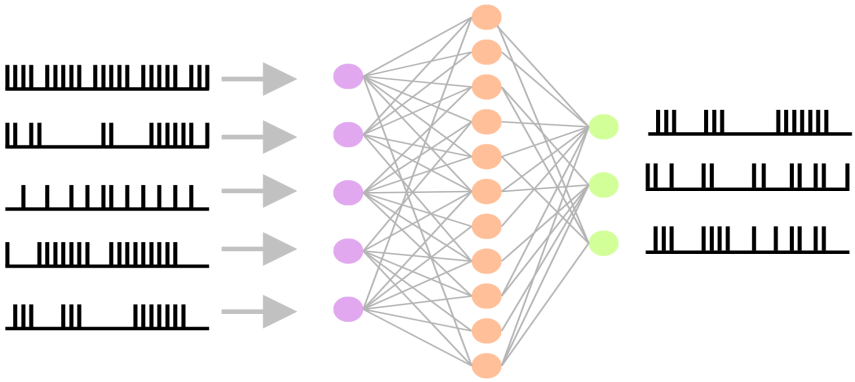


Figure 2.2: Illustration of a SNN architecture

One of the most widely used models in third-generation neural networks is the Leaky Integrate-and-Fire (LIF) neuron. The membrane potential $V(t)$ evolves over time according to the differential equation:

$$\tau_m \frac{dV(t)}{dt} = E_{\text{rest}} - V(t) + R \cdot I_{\text{inj}}(t) \tag{2.5}$$

where τ_m is the membrane time constant (with $\tau_m = R \cdot C$), E_{rest} is the resting potential, R is the membrane resistance, and $I_{\text{inj}}(t)$ is the input current at time t. When the membrane potential reaches a threshold V_{th}, the neuron emits a spike, after which the potential is reset to V_{reset} and the neuron enters a refractory period t_{ref}, during which it cannot fire. The LIF model captures temporal accumulation and leakage of input signals, making it ideal for modeling real-time decision-making processes. When driven by a stream of input spikes, the membrane potential integrates these inputs, but also decays exponentially toward a resting potential, emulating biological signal dissipation. This behavior is key for encoding time-dependent features in tasks such as dynamic gesture recognition or event-based

vision. Because the membrane potential is updated at each time step based on both the current input and its previous value, LIF neurons introduce a recursive, time-dependent computation. The output spike at time t depends not only on the instantaneous input but also on the neuron's past activity. As a result, training spiking neural networks requires propagating gradients through time. This is achieved through Backpropagation Through Time (BPTT), which computes the total gradient of a loss function $\mathcal{L}$ by accumulating its contribution over multiple time steps. For a synaptic weight w_{ij}, this yields:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \sum_t \frac{\partial \mathcal{L}}{\partial S_i(t)} \cdot \frac{\partial S_i(t)}{\partial V_i(t)} \cdot \frac{\partial V_i(t)}{\partial w_{ij}} \tag{2.6}$$

However, a fundamental challenge arises from the non-differentiability of the spike function. A typical spike output is defined as:

$$S[t] = \begin{cases} 1 & \text{if } V[t] > \vartheta \\ 0 & \text{otherwise} \end{cases} \tag{2.7}$$

The Heaviside step function used here is discontinuous, with its derivative being zero almost everywhere and undefined at the threshold. This makes direct application of gradient-based optimization impossible. To overcome this, surrogate gradient methods have been proposed [5]. These methods preserve the binary nature of the forward pass but approximate the non-differentiable function with a smooth surrogate during the backward pass. Various works have explored different surrogate gradients. In practice, Eshraghian et al. [5] recommend the following surrogate, which is also the default in `snnTorch`:

$$\frac{\partial \tilde{S}}{\partial V} = \frac{1}{\pi} \cdot \frac{1}{1 + (\pi(V - \vartheta))^2} \tag{2.8}$$

This function is referred to as the *arctangent surrogate gradient*, as it is the derivative of:

$$\tilde{S} = \frac{1}{\pi} \arctan(\pi(V - \vartheta))$$

Originally proposed by Yin et al. [6]. With this approximation in place, BPTT can be used to train spiking neural networks effectively, allowing them to perform tasks such as classification, temporal pattern recognition, and sensory processing with high energy efficiency.

2.2 Photoplethysmographic Signals

Photoplethysmography (PPG) is an optical, non-invasive technique that measures changes in blood volume in microvascular tissue by detecting variations in light absorption during each cardiac cycle. According to Allen [7], the PPG waveform consists of a pulsatile AC component, synchronous with heartbeat-induced arterial volume changes, superimposed on a slowly varying DC baseline due to venous blood, tissue absorption, respiration, and thermoregulatory activity. The pulsatile AC signal, typically in the 0.5-4 Hz range, reflects how arterial expansion during systole increases light absorption (reducing detector output), whereas diastole reduces absorption, resulting in periodic waveform peaks. These changes follow the modified Beer-Lambert law, linking optical density variation to blood volume fluctuations. The PPG waveform is not a direct measure of pressure but instead reflects the volumetric expansion of blood vessels caused by the pressure wave originating from the heart's systolic contraction. This pulse wave travels through the arterial system and modulates light absorption at peripheral sites such as the fingertip or wrist, providing a convenient yet indirect means of cardiovascular monitoring. Expanding on its clinical relevance, Almarshad et al. [8] highlight that PPG signals carry rich diagnostic features beyond traditional heart rate and oxygen saturation analysis. Morphological features such as pulse amplitude variability, inter-beat intervals, rise and fall times, and waveform shape indices have been associated with conditions including arterial stiffness, atrial fibrillation, autonomic dysfunction, and respiratory irregularities. These findings underscore the value of PPG as a non-invasive, low-cost screening tool across various domains. Sensor configuration plays a key role in signal acquisition. Allen [7] describes two common PPG modalities: transmission mode, where light passes through tissue (e.g., fingertip), and reflection mode, where backscattered light is detected from the same side (e.g., wrist or forehead). PPG is emerging the sensing principle behind many commercial devices, including pulse oximeters, fitness trackers, and smartwatches, which estimate heart rate, stress levels, or even sleep cycles.

However, as PPG is typically measured at superficial locations, it is highly sensitive to a range of environmental and physiological disturbances. Both Allen and Mahmoud note that PPG signals are particularly susceptible to noise, especially from motion artifacts, ambient light interference, poor sensor contact, and low perfusion. These artifacts can degrade signal quality and interfere with accurate physiological feature extraction. Since motion can introduce frequency components that overlap with the heart rate band (0.5–4 Hz), separating true physiological signals from noise requires sophisticated preprocessing. This motivates the application of techniques such as adaptive filtering, motion compensation algorithms, and signal quality indices to improve robustness in wearable scenarios. Addressing this challenge, Pollreisz and TaheriNejad [9] provide a systematic overview of motion artifacts

in wearable PPG signals and propose practical detection and removal strategies. They point out that motion artifacts often overlap spectrally with the physiological PPG signal making traditional filtering approaches ineffective.With proper signal quality control and artifact mitigation, PPG offers a powerful, compact tool for scalable and user-friendly health monitoring.

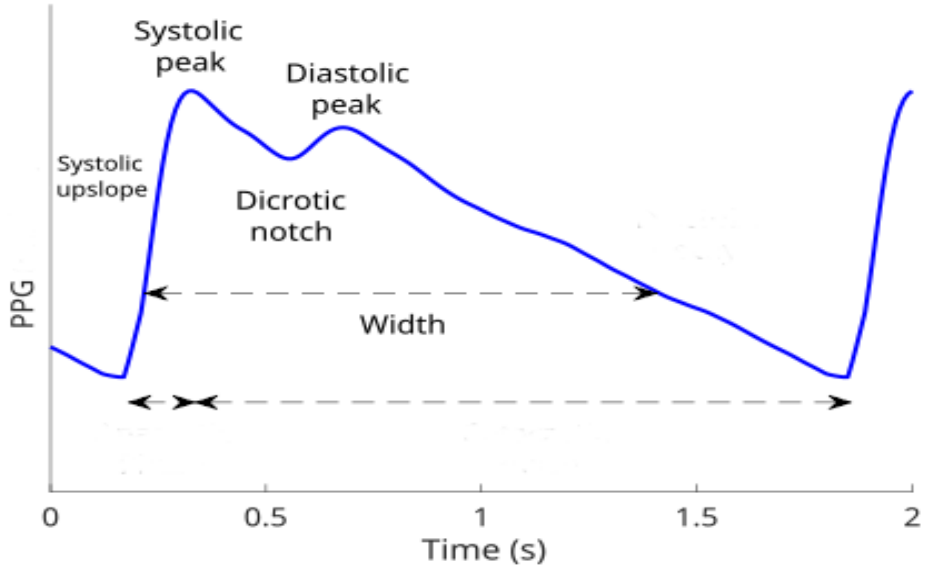


Figure 2.3: PPG Signal Waveform

Source: Peter H. Charlton (2021), Wikimedia Commons, CC BY 4.0.

2.3 Blood Pressure

Blood pressure (BP) is a fundamental physiological parameter and a key biomarker in cardiovascular health. Its proper regulation is essential for ensuring adequate perfusion of vital organs, and disturbances, either excessive or insufficient, can result in serious health complications. Among these, hypertension has become a global public health challenge of extraordinary scale. According to the World Health Organization, approximately 1.4 billion adults aged 30 to 79 suffer from hypertension worldwide, a number that has doubled since 1990 [10]. Despite advances in treatment and awareness campaigns, fewer than 50% of these individuals are aware of their condition, and less than 25% have it under control, with awareness and treatment rates even lower in low and middle-income countries [10]-[11]. This makes hypertension not only widespread but also dangerously silent in its progression, earning it the nickname "the silent killer" due to its often asymptomatic nature. It

is implicated in the development of heart failure, stroke, aortic aneurysms, kidney damage, and cognitive decline [12]. Epidemiological studies attribute nearly 11 million deaths globally each year to complications related to elevated blood pressure [10]. Among cardiovascular deaths alone, roughly 50% are linked to hypertension, making it the single most modifiable contributor to global mortality [11]. Beyond its role in end-organ damage, elevated blood pressure is increasingly being recognized as a continuous, rather than categorical, risk factor, meaning that even moderate increases in systolic pressure can lead to an incremental rise in mortality risk [12]. On the other end of the spectrum, low blood pressure, particularly orthostatic hypotension (OH), also deserves clinical attention. Although traditionally viewed as benign or even favorable in younger, healthy populations, hypotension can present significant challenges in elderly individuals or those with underlying cardiac, neurological, or endocrine conditions. OH, typically defined as a sustained drop in systolic or diastolic BP within a few minutes of standing, is common among older adults and has been associated with symptoms such as dizziness, blurred vision, fatigue, and even syncope. Recent studies estimate that up to 30% of older adults experience orthostatic hypotension, with prevalence increasing significantly in institutionalized or frail populations [13]. Beyond transient symptoms, it has been linked to higher risks of falls, cardiovascular events, and even all-cause mortality, including a approximately two-fold higher risk of death in middle-aged individuals [13].

These epidemiological findings emphasize the importance of accurate, accessible, and frequent blood pressure monitoring. Traditionally, BP is measured using a sphygmomanometer (manual or digital), either with auscultation (Korotkoff sounds) or oscillometric methods. Measurements are typically taken after 5 minutes of rest, using a cuff placed snugly on the upper arm. Guidelines recommend at least two measurements, spaced 1–2 minutes apart. Manual measurements rely on listening for Korotkoff sounds using a stethoscope, while automated devices calculate pressure based on oscillation patterns detected during cuff deflation. Despite being widely adopted, these techniques remain episodic and can be influenced by stress (white-coat effect), human error, or patient movement. For this reason, ambulatory blood pressure monitoring (ABPM) over 24 hours or home-based self-monitoring is increasingly recommended to obtain more representative data. ABPM devices typically record BP every 15 to 30 minutes throughout a 24-hour period, providing information on circadian patterns such as nighttime dipping, morning surge, or masked hypertension. These insights are clinically valuable and allow more accurate risk stratification than isolated clinic readings. Nevertheless, the discomfort caused by frequent cuff inflation often reduces patient compliance, highlighting the need for continuous, cuffless solutions. In light of the growing burden of both hypertension and hypotension, particularly in aging populations, there is a pressing need for novel, reliable, and comfortable BP monitoring solutions. Wearable and cuffless

systems, often leveraging signals such as PPG, pulse transit time, or machine learning-based models, are currently under investigation as alternatives that could enable continuous, non-invasive blood pressure estimation and early detection of adverse trends.

Chapter 3

Related works

The literature on SNNs and conventional methods for physiological signal analysis has expanded rapidly in recent years. Significant progress has been made in energy-efficient neuromorphic processing, as well as in the use of PPG signals for wearable and clinical monitoring applications.

3.1 Applications of SNNs in Biosignal Processing

Spiking Neural Networks (SNNs), as biologically inspired models capable of processing temporal information in an event-driven and energy-efficient manner, have emerged as promising tools for the analysis of physiological signals. Their ability to naturally encode and process time-dependent patterns makes them particularly well-suited to biosignals, which are inherently dynamic and often noisy. One physiological domain where SNNs have shown strong potential is electromyography (EMG), which records muscle activity through surface or intramuscular electrodes and is widely used for applications ranging from prosthetic control to clinical diagnosis. In one of the first neuromorphic implementations, Donati et al. [14] demonstrated the feasibility of using an SNN deployed on analog hardware to classify hand movements from EMG signals . Their approach leveraged the event-driven nature of SNNs to reduce energy requirements without sacrificing classification performance, establishing a baseline for real-time EMG interpretation on compact embedded systems. Similarly, Scrugli et al. [15] presented a fully FPGA-based architecture optimized for gesture recognition using surface EMG signals. Their system combined pre-processing modules with a spiking neural network that was able to discriminate a set of hand and finger gestures with high accuracy, while maintaining low-latency and ultra-low-power operation, making it suitable for wearable neuroprosthetic devices or battery-constrained applications. More recently, Bezugam et al. [16] introduced DEXAT, a recurrent spiking network

architecture designed to operate on Intel's Loihi neuromorphic chip. Their system was evaluated on EMG gesture recognition tasks and demonstrated performance comparable to deep learning models, while consuming significantly less power. In addition to showcasing the effectiveness of recurrent SNNs in capturing temporal dependencies in EMG patterns, this work highlights the compatibility of SNNs with modern neuromorphic hardware platforms.

Beyond EMG signals, Spiking Neural Networks have also gained traction in the analysis of electrocardiograms (ECG), which are crucial for diagnosing cardiac abnormalities. Yan et al. [17] proposed a two-stage ECG classification framework combining CNNs and SNNs to reduce power consumption while maintaining high diagnostic accuracy. In their approach, an initial CNN filters out normal heartbeats, while a secondary classifier handles arrhythmic cases. The resulting CNN is then converted to an SNN using rate encoding and threshold balancing, achieving 91% classification accuracy on the MIT-BIH arrhythmia database with a power draw as low as 0.077 W. Further exploring energy-efficient learning, Amirshahi and Hashemi [18] adopted a biologically inspired training paradigm using unsupervised spike-timing-dependent plasticity (STDP) and reward-modulated STDP (R-STDP) for ECG signal classification. Focusing specifically on the detection of ventricular ectopic beats (VEBs), their network, trained without conventional backpropagation, achieved over 97% accuracy. This work reinforces the potential of SNNs not only for efficient inference, but also for biologically plausible adaptive learning. Finally, Yan et al. [19] introduced SparrowSNN, a hardware/software co-design targeting ECG arrhythmia on wearable ASICs. Tested on MIT-BIH, it achieved a 98.29% classification accuracy at just 31.4 nJ per inference and 6.1 µW power consumption, demonstrating state-of-the-art accuracy with ultra-low energy use.

Beyond EMG and ECG analysis, electroencephalographic (EEG) signals represent another crucial domain where spiking neural networks have demonstrated promising capabilities, particularly in the context of neurological disorders such as epilepsy. EEG is inherently noisy, high-dimensional, and time-dependent, which aligns well with the spatiotemporal processing strengths of SNNs. A representative example is the work by Burelo et al. [20], who developed a neuromorphic spiking neural network designed to detect high-frequency oscillations (HFOs) in scalp EEG recordings from pediatric epilepsy patients. These events of interest, typically in the 80, 250 Hz range, are emerging as reliable non-invasive biomarkers for identifying epileptogenic brain regions. The proposed SNN architecture employed biologically realistic LIF neurons and a dual-stage detection pipeline, including a module for artifact suppression and another tuned for HFO identification. Notably, their system operated on energy-efficient neuromorphic hardware and achieved 80% accuracy in detecting active epilepsy, with HFO rates strongly correlating with seizure frequency. This line of research builds on earlier efforts by Sharifshazileh et al. [21], who implemented an SNN for real-time HFO detection in intracranial

EEG. Their neuromorphic approach enabled ultra-low-latency inference and laid the foundation for SNN-based epilepsy diagnostics in more invasive but controlled environments. By demonstrating real-time neuromorphic detection of HFOs in intracranial EEG, their work laid the foundation for SNN-based epilepsy diagnostics in controlled invasive settings. Additionally, most recently Yang et al. (2022) [22] designed a neuromorphic deep SNN featuring a novel Spiking ConvLSTM (SPCLU) layer to detect seizures from EEG signals. Tested on Freiburg, CHB-MIT, and EPILEPSIAE datasets, their architecture achieved AUC scores of 92.7%, 89.0%, and 81.1%, respectively, while reducing computational and energy demands.

Continuing the trend toward broader biosignal applicability, recent studies have explored the use of photoplethysmographic signals, a non-invasive optical method widely used in wearable devices to measure cardiovascular parameters such as heart rate and respiratory rate. PPG signals, while cost-effective and easy to acquire, are often susceptible to motion artifacts and low signal-to-noise ratios, making them ideal candidates for event-based, noise-resilient processing offered by SNNs. In this context, Yang et al. [23] introduced an end-to-end Spiking Neural Network for respiratory rate estimation from PPG. Their approach encodes raw PPG windows into sequential spike trains and processes them with a feedback-based integrate-and-fire network. Evaluated on the BIDMC respiratory dataset, the network achieved mean absolute errors as low as 1.15 bpm (for 64 s windows) and demonstrated significantly improved energy efficiency over traditional deep learning models. Complementing this, De Luca et al. [24] proposed a neuromorphic multi-scale framework for real-time heart rate monitoring and physiological state detection. Their system integrates multimodal biosignals, including PPG, ECG, and accelerometer data, which are processed through a bank of band-pass filters and encoded into spike trains using LIF neurons. A spiking neural state machine decodes instantaneous heart rate and identifies long-term state transitions, such as agitation, across multiple temporal scales. Implemented and validated on mixed-signal neuromorphic hardware, the architecture demonstrated robust operation despite analog circuit variability and ultra-low power consumption, highlighting its suitability for always-on wearable applications.

These developments in PPG analysis, alongside existing progress in EMG, ECG, and EEG applications, underscore the versatility and maturity of SNNs for real-world biosignal interpretation. Their unique blend of temporal precision, bio-plausibility, and low energy demands makes them a compelling choice for next-generation wearable and implantable monitoring systems.

3.2 Photoplethysmographic Signal Beyond SNNs

Although the application of Spiking Neural Networks to photoplethysmographic signals remains relatively recent, PPG is already a mature and widely used biosignal in both clinical and wearable health monitoring. PPG is a non-invasive optical method that detects blood volume changes in microvascular tissue by measuring light absorption at the skin surface. It typically uses two light sources, red (660nm) and infrared (940nm), and a photodetector to generate waveforms that reflect cardiac cycles. This low-cost, compact sensing architecture makes PPG ideal for continuous monitoring in wearable and mobile health systems. In [25], Shaw et al. proposed a real-time pulse oximeter system using analog circuits and a digital microcontroller. Tested on 10 individuals, their prototype achieved pulse rate measurements with a mean absolute error of 1.2–1.6bpm when compared to a clinical pulse oximeter, and SpO_2 values within $\pm 2\%$ of reference readings. Beyond traditional vital sign monitoring, PPG has been explored in affective computing. In [26], Lee et al. developed a 1D convolutional neural network to classify emotional states using only PPG signals. The model operated on single 1.1-second pulse segments and was evaluated on the DEAP dataset, a standard benchmark in emotion recognition research. The classifier achieved 75.3% accuracy in distinguishing between high and low valence states, and 76.2% for arousal states, showing that PPG encodes subtle changes in autonomic nervous system activity relevant to affective states. The significance of this work lies in demonstrating that even very short windows of PPG, without any additional sensor modalities, can support meaningful emotion classification, opening the door to lightweight affective monitoring in mobile and wearable settings. PPG has also demonstrated significant value in the cardiovascular domain. In [27], Liang et al. assessed hypertension classification using ECG and PPG signals from the MIMIC database. They extracted extensive morphological PPG features and pulse arrival time (PAT) features and evaluated three classification tasks: normotension vs. prehypertension, normotension vs. hypertension, and combined normotension/prehypertension vs. hypertension. PPG features outperformed PAT, and combining both modalities achieved the highest F1 score of 94.84%. These findings illustrate the standalone diagnostic power of PPG and its complementarity to ECG-derived features. Similarly, PPG has proven effective in large-scale arrhythmia screening efforts. The study by Yang et al. [28] proposes a PPG-based atrial fibrillation (AF) detection method using wearable wristband devices. The authors extract frequency domain features from 10 seconds PPG segments, with Haar wavelet decomposition achieving optimal performance. Their approach focuses on reducing computational complexity by selecting a minimal set of features that maintain high accuracy. The method achieves

classification accuracies exceeding 90% for individual patients using frequency-domain features. The algorithm's efficiency supports implementation on resource-constrained wearable devices, reducing data storage needs compared to raw PPG recordings. This makes it well-suited for long-term ambulatory monitoring and addresses challenges such as paroxysmal AF detection. Supporting these individual findings, a systematic review by Almarshad et al. [8] further reinforced PPG's diagnostic versatility. The authors examined a large number of studies spanning applications in cardiology, respiratory monitoring, neurology, and physical activity assessment. The review emphasized how features derived from PPG, such as waveform morphology, variability, and timing, can support the estimation of respiratory rate, assessment of vascular aging, pulse transit time analysis, and tracking of autonomic nervous system activity. Notably, the review highlighted growing evidence that PPG is not limited to heart rate and SpO_2 monitoring, but can serve as a standalone biosignal for broader health insights, particularly in wearable and remote monitoring contexts. However, the authors also underscored the importance of ensuring signal quality and robustness in real-world scenarios to fully realize PPG's clinical utility. Real-world recordings are susceptible to motion artifacts, baseline drift, skin tone variability, and ambient light interference. Allen [7] emphasized that even minimal movement or sensor tilt can introduce distortions in PPG waveform morphology, potentially compromising the estimation of systolic peaks or dicrotic notches. Although various adaptive filtering techniques and signal quality indices have been proposed, there remains a pressing need for real-time, artifact-resilient PPG analysis, particularly in mobile, ambulatory, and fitness-oriented settings.
Given the growing diagnostic importance of PPG and its sensitivity to noise, neuromorphic processing approaches such as Spiking Neural Networks offer a compelling path forward. Their event-driven operation, biological plausibility, and energy efficiency make them well-suited to handle temporally rich but noisy biosignals like PPG. As discussed earlier, early efforts to integrate SNNs for PPG-based tasks like respiratory and heart rate estimation have shown promising results.

3.3 Benchmark and datasets

A key point of reference for this thesis is the work presented in [1], which stands out as a comprehensive studies for blood pressure (BP) estimation using photoplethysmographic (PPG) signals. In this study, the authors used four of the most common public datasets used in BP estimation:

- **Sensors Dataset (from MIMIC-III Waveform Database)**: A subset of the MIMIC-III waveform database [29], containing synchronized fingertip PPG and invasive arterial BP waveforms from 1195 ICU patients. The

benchmark selected two non-overlapping 15-second PPG–ABP segments per patient (spaced 5 minutes apart) to ensure temporal independence [1]. This dataset therefore represents a large, heterogeneous ICU population spanning a broad range of ages and clinical conditions.

- **UCI Dataset (from MIMIC-II Waveform Database)**: Introduced by Kachuee et al.[30], this subset of the MIMIC-II waveform database contains 12,000 segments of simultaneous PPG, ECG, and ABP data from 942 ICU patients. Each segment is approximately 8.2 seconds long (1024 samples at 125Hz). The data were originally provided in four parts (3000 segments each) with no patient identifiers, so the benchmark employs a hold-one-set-out split to avoid subject overlap [1]. It is the largest dataset in the benchmark and has been widely used for cuffless BP estimation.
- **PPGBP Dataset (Liang et al., 2018)**: A community-based PPG–BP dataset with 219 adult subjects (many with hypertension or diabetes) [**liang2018**]. After 10 minutes of seated rest, each subject's SBP/DBP was measured with an Omron HEM-7201 cuff, followed by acquisition of three 2.1-second fingertip PPG segments (SEP9AF-2 sensor). The original recordings (1000Hz) were downsampled to 125Hz for consistency. In total 613 segments (3 per subject) were obtained, making this the smallest dataset by segment count.
- **BCG Dataset (Kansas State University)**: A bed-based ballistocardiography dataset collected by Carlson et al.[31], containing one recording session each from 40 adults (4 with known cardiac conditions) lying supine. Ballistocardiogram (BCG) signals were captured via EMFi and load-cell sensors, fingertip PPG via a GE Datex CardioCap5, and continuous brachial BP via a Finapres FinometerPRO. All signals were originally sampled at 1000Hz; for the benchmark they were downsampled to 125Hz [1]. Because of the small cohort, this dataset yields a high number of segments per subject.

The same datasets were used also for this project. A particularly important contribution of [1] is the preprocessing methodology. In addition to standard outlier removal and filtering of low-quality data, the paper emphasizes subject-wise data splitting. For each dataset, the authors designed specific fold-division strategies to preserve original distributions while ensuring that data from the same subject does not appear in both training and test sets. This precaution is crucial to avoid data leakage, which can lead to overly optimistic results due to the model learning patient-specific patterns. The authors show that not considering this aspect can increase performance by up to 30%. From the raw PPG signals, the authors extracted a rich set of features, related to points of interest (e.g., systolic peak, dicrotic notch), in time domain, frequency domain, statistical index, and energy-related. These features were used to train and compare a wide range ML and

DL models. The results they present serve as solid benchmarks for future research in this field. Table 3.1 summarizes the best-performing models, reporting results for both systolic and diastolic blood pressure estimation in terms of mean absolute error (MAE), mean error with standard deviation (ME ± SD), and (MASE%), a new metric introduced by the authors to have more comparable results across different datasets. This metric represent the proportion, in percentage, between the naive classifier and the results obtained by the model. The naive classifier in this context is the one that return always the mean value of the training set.

Table 3.1: Performance of benchmark models on BP estimation from PPG signals.

Dataset	**Model**	**Systolic BP**			**Diastolic BP**		
		MAE	ME ± SD	MASE%	MAE	ME ± SD	MASE%
Sensors	SVR	15.60	0.00 ± 19.68	88.62%	7.50	-1.45 ± 9.81	90.76%
UCI	U-Net	16.93	0.06 ± 20.92	96.04%	7.88	-2.46 ± 10.80	92.17%
BCG	SVR	11.45	-0.79 ± 15.56	93.07%	7.34	0.01 ± 9.88	92.75%
PPGBP	SVR	13.15	-0.64 ± 17.05	80.29%	8.04	-0.22 ± 10.14	90.90%

The previous table is a summary reporting, for each dataset, the model that achieved the lowest mean error computed as the average of the SBP and DBP MAEs. However, the other metrics should also be considered when evaluating model performance. While this work provides a valuable foundation for BP estimation, it does not account for the energy efficiency of the proposed models. Considering that PPG signals can be collected non-invasively via wearable battery-powered devices, energy consumption becomes a crucial design constraint. To address this, other works such as [32] have focused on optimizing performance and efficiency. In particular, the authors started from two well-performing deep learning models and applied a combination of neural architecture search (NAS) and quantization. Their final model was deployed on an ultra-low-power multicore System-on-Chip (SoC), achieving a minimum diastolic MAE of 8.08 mmHg with an energy consumption of just 0.37 mJ per inference.

3.4 DYNAPs

The Dynamic Neuromorphic Asynchronous Processor (DYNAPs) [2] is a mixed-signal neuromorphic computing platform designed to support real-time processing of spiking neural networks with high energy efficiency and architectural flexibility. Unlike conventional von Neumann processors, which suffer from memory access bottlenecks, DYNAPs co-locates memory and computation within its processing

elements, enabling in-memory computing directly at the neuron and synapse level. This minimizes latency and power consumption by eliminating the need to move data between separate memory and compute units. The architecture is composed of multiple cores, each containing 256 analog neurons and thousands of programmable synapses. Communication among neurons is event-driven and asynchronous, using the Address-Event Representation (AER) protocol. To enable efficient and scalable spike routing, DYNAPs implements a two-stage tag-based addressing scheme combined with a hierarchical-mesh routing fabric. This approach significantly reduces memory usage per neuron while preserving high fan-out connectivity and network flexibility. Each neuron and synapse is implemented using subthreshold analog circuits that replicate biophysically plausible dynamics. The behavior of these circuits is modulated via on-chip programmable bias generators, which expose a range of tunable parameters including synaptic time constants, excitability thresholds, and adaptation currents. Synapses are also configurable in terms of their dynamic behavior, and weights are represented by local SRAM-encoded states. Routing is managed by a three-level hierarchy of asynchronous routers (R1, R2, and R3), which efficiently handle local, inter-core, and inter-chip communication. Memory used for routing and configuration is distributed across the system, implemented using asynchronous SRAM and content-addressable memory (CAM) blocks. This design enables true in-memory computation, with each synapse storing both connection data and processing incoming spikes locally. Despite being fabricated in a 0.18 μm CMOS process, the DYNAPs processor achieves competitive energy and latency metrics. Event routing and spike generation consume power in the picojoule to nanojoule range, and broadcast and routing latencies remain under 30 ns in typical conditions.

Chapter 4

Methodology

This chapter presents an overview of the main strategies adopted to address the problem of blood pressure estimation from photoplethysmographic (PPG) signals using spiking neural networks (SNNs). The architectural design of the networks is examined, along with the training methodologies and tuning techniques employed. Before proceeding with the experimental analysis, a general overview of snnTorch is provided. This library, built on top of the PyTorch framework, was used to simulate SNNs. While it does not offer a fully faithful emulation of deployment on neuromorphic hardware, it enables effective simulations that offer valuable insights into the behavior and potential of SNN-based applications. The fundamental components of these networks are linear layers combined with spiking neuron models. snnTorch provides a range of neuron models, each with distinct dynamics and levels of biological plausibility. The main neuron types available in the library include:

- **`snn.Leaky`**: A discrete-time first-order LIF neuron. The update rule is:

  $$u[n+1] = \beta u[n](1 - S[n]) + I[n]$$

  where β is the decay constant, $S[n]$ is the spike at timestep n, and $I[n]$ is the input current.

- **`snn.RLeaky`**: A recurrent variant of the leaky model, adding feedback from previous spikes:

  $$u[n+1] = \beta u[n](1 - S[n]) + I[n] + RS[n-1]$$

  where R is a recurrent weight.

- **`snn.SLSTM`**: A spiking Long Short-Term Memory (LSTM) neuron model that extends classical LSTM cells to the spiking domain. It incorporates gating

mechanisms and internal state variables analogous to the hidden and cell states of standard LSTMs, while enforcing spike-based communication through thresholding and reset dynamics. This model enables long-term temporal dependency modeling within spiking neural networks.

Several of these neuron models were initially tested. However, subsequent experiments focused primarily on the Leaky Integrate-and-Fire (LIF) neuron model due to its balance between computational simplicity and biological plausibility, making it well-aligned with the fundamental principles of SNNs.
The networks evaluated in this work are generally similar in structure, but differ in terms of input and/or output configurations. The primary model, which takes engineered features as input and firing rates as reference output, demonstrated the best performance. As a result, it was selected as the main approach, and most of the effort in terms of optimization, loss analysis, and hyperparameter tuning was focused on this configuration. However, two additional variants are also presented to highlight alternative strategies, compare their performance, and demonstrate that different modeling choices remain viable and worth exploring.

4.1 Approach with features as input and firing rates as output

In this case, the network received 20 input signals, which were connected via linear layers to an equal number of neurons forming a hidden layer. These neurons were then connected to an output layer composed of 200 neurons, with 100 neurons dedicated to inferring systolic pressure (SP) and the remaining 100 assigned to diastolic pressure (DP).
Focusing on SP estimation, the prediction was carried out using the firing rates of the 100 neurons associated with SP. These neurons represent equidistant values between 70 and 190 mmHg, corresponding to the expected physiological range of systolic pressure. The final estimated value is obtained through a weighted sum, where each neuron's associated pressure value is multiplied by its normalized firing rate. The firing rates are normalized such that their sum equals 1, ensuring the output is a valid convex combination.
The estimation formula is defined as follows:

$$\hat{P}_{\mathrm{SP}} = \sum_{i=1}^{100} f_i \cdot v_i \tag{4.1}$$

where:

- $\hat{P}_{\mathrm{SP}}$ is the predicted systolic pressure,

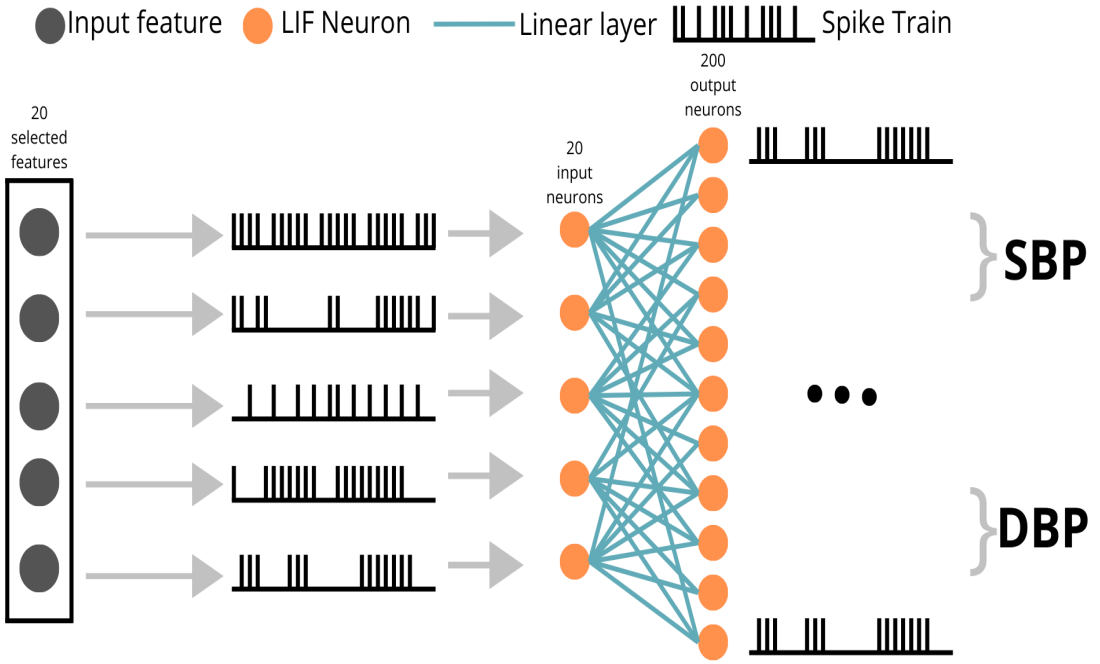


Figure 4.1: Network that receives features extracted from PPG as input and uses the firing rates of the second layer to compute the output

- f_i is the normalized firing rate of neuron i,
- v_i is the pressure value associated with neuron i (ranging from 70 to 190 mmHg).

A similar procedure is applied to the remaining 100 neurons to estimate the DP. In this case, a variant of the network was also implemented in which the raw PPG signal is directly provided as input, replacing the pre-extracted features. The overall architecture remains similar to the previous model, however, the entire signal is fed into the network, and the hidden layer is composed of 125 neurons, each fully connected to the input.

4.2 Approach with PPG as input and membranes potential as output

In this configuration, prediction is performed by directly adding two output neurons: one for SP and one for DP. The predicted value corresponds to the membrane

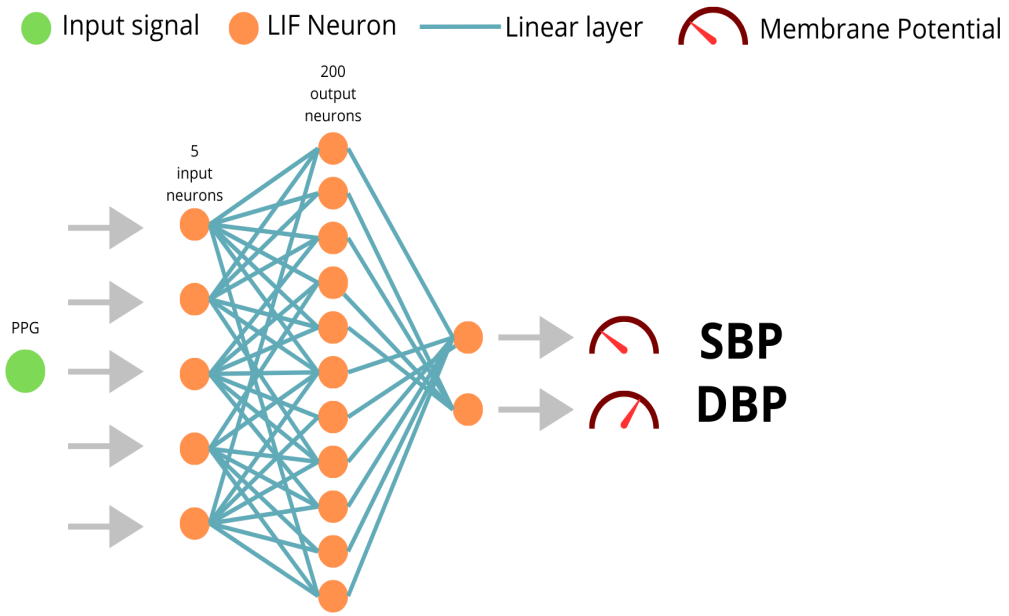


Figure 4.2: Network that receives PPG as input and uses the membrane potentials of the last layer to compute the output

potential of each respective neuron at the end of a predefined number of simulation steps. The network architecture in this case consists of a single input signal connected to a small hidden layer of five neurons. The second layer, preceding the two output neurons used for inference, is identical to the one used in the previous configuration and is composed of 200 neurons. One of the characteristics of Leaky Integrate-and-Fire (LIF) neurons is that, after emitting a spike, they reset either by returning to zero or by subtracting a fixed value from the membrane potential closely resembling the behavior observed in biological neurons.
When the membrane potential is used directly for prediction, the most coherent choice is to disable the reset mechanism in the output layer, in order to preserve the information accumulated throughout the simulation. This prevents the loss of information accumulated during the simulation, thus allowing for a more accurate and continuous representation of the signal in the final layer.

It is important to note that the number of neurons used in the networks was not selected arbitrarily. Instead, it was defined in accordance with the physical constraints of the neuromorphic hardware, specifically the DynapsE chip [2], on which the deployment was later performed. Each core of the chip includes 256 neurons

and supports a limited number of connections and input channels. Consequently, the network dimensions were chosen based on these hardware limitations, as well as on empirical guidelines provided by researchers with direct experience using the chip. These guidelines suggest typical proportions between input size and hidden layer dimensions that ensure efficient and feasible implementations.
Although each Dynapse chip comprises four cores, allowing for potential scaling of the network, the value of 256 neurons per core was adopted as a baseline. Furthermore, since the experimental results did not highlight significant advantages from marginal increases in the number of neurons, the initial proportions were largely preserved throughout the network design process.

4.3 Encoding

Third-generation neural networks operate using spikes, therefore, it is necessary to adapt the input signals accordingly. There are several methods for converting a numerical value into a spike train over a fixed number of time steps. Some of the most common encoding techniques are outlined below.

Rate Encoding

In this type of encoding a spike train of length n is generated from a single scalar input value. This value is first normalized to fall within the range $[0, 1]$ and then used as the firing probability across all time steps. If $v_{\text{norm}} \in [0,1]$ is the normalized input value, each element of the spike train $S \in \{0,1\}^n$ is drawn independently as:

$$S_t \sim \text{Bernoulli}(v_{\text{norm}}), \quad \text{for } t = 1, 2, \ldots, n$$

Over a sufficiently large number of steps n, the original normalized input value can be approximately reconstructed by computing the average firing rate:

$$\hat{v}_{\text{norm}} \approx \frac{1}{n} \sum_{t=1}^{n} S_t$$

Temporal Encoding

In temporal encoding, the normalized input value is used to determine the specific time step within the spike train at which a single spike will occur. This means exactly one spike is generated per input value, and its timing encodes the magnitude

of the input. Formally, if $v_{\text{norm}} \in [0,1]$ is the normalized input value and the spike train length is n, then the spike occurs at time step

$$t_s = \lceil (1 - v_{\text{norm}}) \times (n-1) \rceil$$

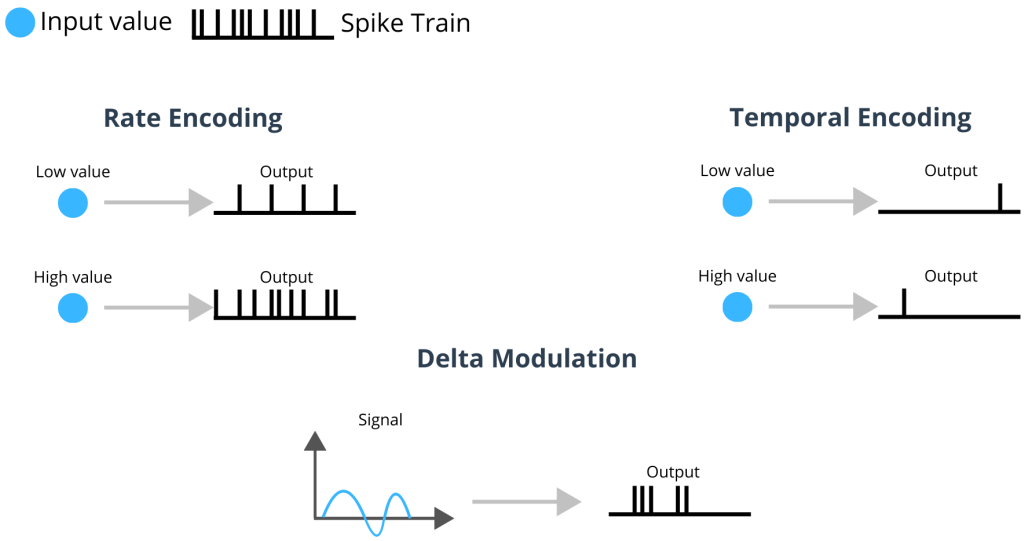
where t_s ranges from 0 (earliest spike for maximum input) to $n-1$ (latest spike for minimum input). One limitation of this encoding is its sensitivity to noise, as only a single spike represents the input, reducing robustness compared to rate encoding.

Delta Modulation

A commonly used strategy, especially for encoding continuous signals, is based on detecting significant changes in the input signal relative to predefined thresholds. Typically, two thresholds and two channels are employed, generating two spike trains. When the input signal increases by more than a positive threshold, a spike is emitted on the positive channel. Conversely, when the signal decreases by more than a negative threshold, a spike is emitted on the negative channel. This technique is advantageous because it produces only two spike trains for an entire continuous signal, rather than one train per individual sample. Moreover, from an energy perspective, it is efficient since no spikes are emitted when the signal remains stable. However, this approach is still sensitive to noise, which can cause unintended spike emissions. Formally, denoting the input signal at time t as x_t, and the thresholds as θ^+ and θ^-, spikes are emitted as follows:

$$S_t^+ = \begin{cases} 1 & \text{if } x_t - x_{t-1} > \theta^+ \\ 0 & \text{otherwise} \end{cases} \qquad S_t^- = \begin{cases} 1 & \text{if } x_{t-1} - x_t > \theta^- \\ 0 & \text{otherwise} \end{cases}$$

Figure 4.3: Encoding Types



4.4 Data and Training

Features

From the PPG signal a large amount of information can be extracted. The selected features span a wide range of descriptors designed to capture the rich temporal, morphological, and frequency-based characteristics of the PPG waveform and its derivatives. Several features are based on timing intervals between key fiducial points, for instance, `T_peak_e`, `T_peak_a`, and `TdiaToEnd` represent time intervals between the systolic peak and subsequent morphological landmarks in the PPG or its derivatives, these intervals provide indirect information about arterial stiffness and wave propagation dynamics. Features such as `DW25`, `DWdivSW25`, and `DWdivSW75` quantify the durations and relative proportions of the systolic and diastolic phases of the pulse cycle.
Another important subset includes features derived from the first derivative of the PPG (VPG) and its second derivative (APG), which emphasize rate of change information and acceleration phases. Histogram-based descriptors like `vpg_histogram_down` and `vpg_max_neighbor_mean` capture local activity and distributional properties of VPG segments during falling slopes, while features such as `apg_e` and `ratio_apg_b` are based on the amplitude and timing of characteristic

points labeled e to b in the APG signal. These APG points are physiologically relevant and often linked to arterial elasticity and vascular aging. Additional temporal descriptors such as `T_d`, `T_b`, `T_c`, and `T_e` reflect latency between specific wave components (e.g., systolic peak to dicrotic notch).

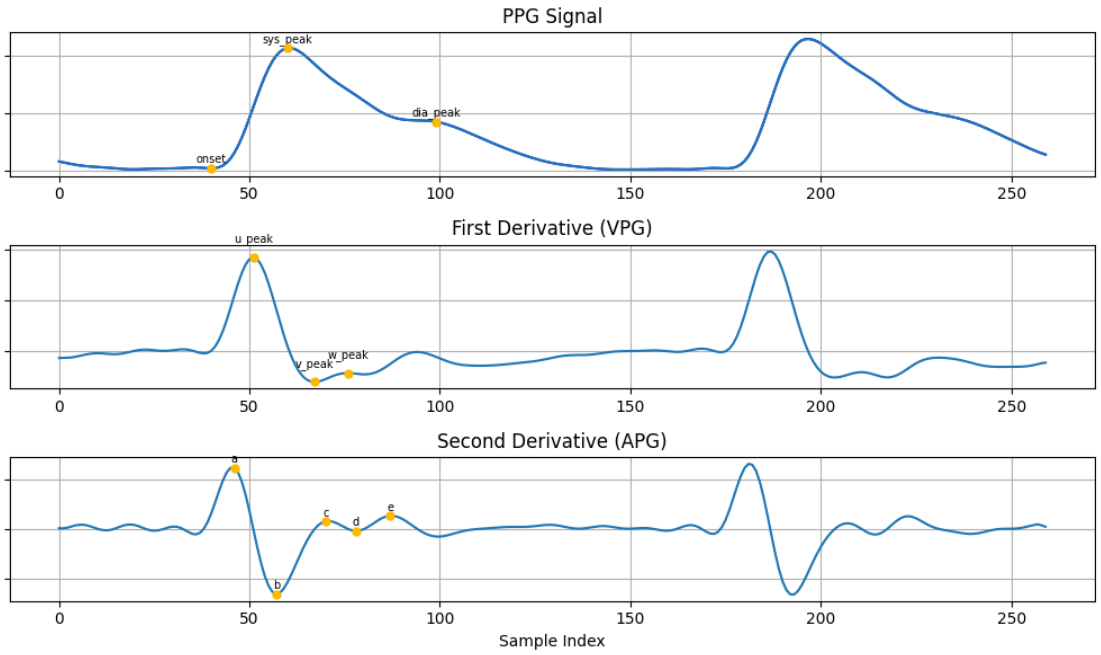


Figure 4.4: PPG representation along with its first and second derivatives

Frequency-domain features like `ppg_fft_peaks_heights` and `ppg_fft_peaks_neighbor_avgs` describe the amplitude and spectral neighborhood of the dominant harmonics extracted via the FFT of the PPG waveform. These features are helpful in capturing oscillatory behavior of the cardiovascular system. Other global features such as `AUCsys_norm` and `AUCdia_norm` measure the area under the curve during the systolic and diastolic phases Signal quality is quantified using metrics such as `SQI_kurtosis`, offering a measure of deviation from normal distribution, useful to assess noise or irregular morphology. Lastly, the features labeled as `S1` to `S4` represent the area under the curve (AUC) of the PPG waveform in four distinct time segments within a single cardiac cycle. The complete list of selected features and their formal definitions is reported in the Appendix 1.

Given the large number of extracted features, a selection step was performed to keep the 20 most relevant ones. This was done using a tree-based method, two separate RandomForestRegressor models were trained, one for each of the two

target variables. Each forest was composed of 100 estimators (trees), and both training and feature importance extraction were performed using the scikit-learn library. After training, the importance of each feature was evaluated based on the mean decrease in impurity, which reflects the total reduction in impurity a feature contributes across all trees in the forest. In the context of regression tasks, scikit-learn uses the squared error as the impurity measure. The importance $I(f)$ of a feature f is calculated as:

$$I(f) = \sum_{t \in T_f} \frac{N_t}{N} \cdot (\mathrm{SE}(t) - \mathrm{SE}(t_L) - \mathrm{SE}(t_R))$$

Where:

- T_f is the set of all internal nodes where feature f is used for splitting,
- N is the total number of training samples,
- N_t is the number of samples reaching node t,
- $\mathrm{SE}(t)$ is the sum of squared errors at node t,
- t_L and t_R are the left and right children of node t.

This method assigns higher importance to features that consistently produce large reductions in squared error across the ensemble.

Normalization

Since the extracted features are in different ranges of values, a normalization step was needed. In this case, a min-max normalization was applied to each feature, bringing the values to the $[0, 1]$ interval. The normalization is defined by the following formula:

$$x' = \frac{x - \min(X)}{\max(X) - \min(X)}$$

Where:

- x is the original feature value,
- $\min(X)$ and $\max(X)$ are the minimum and maximum values of the feature across the training set,
- x' is the normalized value.

Some preprocessing was also applied to the input signal. Since the signal itself acts as a single feature, normalization is not strictly necessary, especially in scenarios where no spike conversion occurs before feeding the data into the model. Nevertheless, various preprocessing strategies were tested. For the configuration where the membrane potential is used as output, the best-performing approach was to directly feed the raw signal into the network, one value at a time over its temporal length. However, because the original signal consisted of 625 values, a downsampling step was applied, reducing its length to 125 values to make the input more computationally manageable. A similar strategy was adopted for the case in which the raw signal serves as input, while the outputs are represented as firing rates. In this setting, we also evaluated an alternative where the full signal is treated as a single feature and passed entirely through the network over a fixed number of time steps. Specifically, the input layer comprised 125 neurons, each receiving the entire signal at every time step. Although the raw signal already lies within a relatively narrow range, its absolute values are low. This can lead to poor gradient flow or under-activation of certain neurons and connections during training. To mitigate this, it was also tested an approach where the entire signal was scaled by a factor of 10 before being fed into the model.

Sampling and augmentation

For training and evaluation, a cross-validation strategy was employed for the BCG, Sensors, and PPGB datasets, while for the UCI dataset, a hold-one-out (HOO) approach was preferred due to its large number of records. In all cases, data splitting was performed following the criteria described in [1], ensuring that no patient data leakage occurred across folds.
One of the main challenges in blood pressure estimation is that most of the recorded systolic and diastolic pressure values fall within a narrow physiological range. This makes it difficult for models to learn and generalize to BP values outside of these common intervals. As illustrated in 4.5, for example, approximately 70% of the SP values in the BCG dataset lie within the range $[100, 140]$, while approximately 80% of the DP values fall within $[50, 80]$. This not only limits the model's ability to predict rare or extreme BP values, but also complicates the identification of whether such outlier readings represent genuine physiological events or noisy measurements.

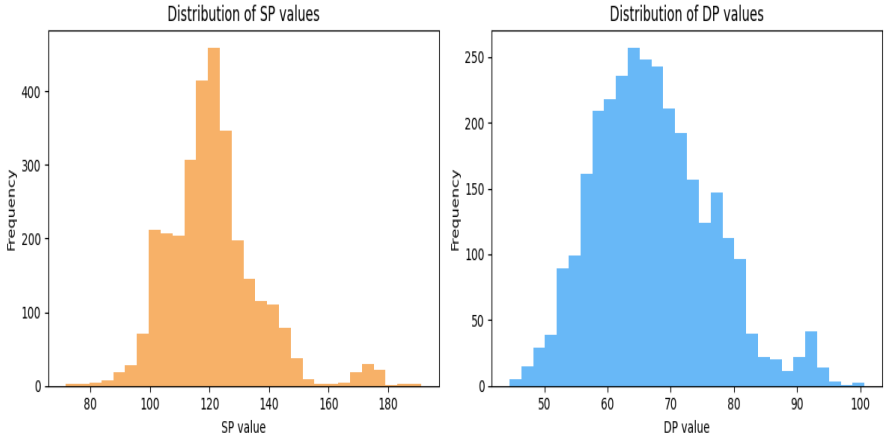


Figure 4.5: Labels distribution in BCG

To partially reduce the impact of the mentioned issues, a stratified sampling strategy was employed, involving a naive augmentation of underrepresented intervals. The SP values in the training set were used as a reference, and the range from, approximately, 70 to 190 mmHg was divided into 10 intervals.
A target value x, depending on the dataset size, was then imposed such that each interval would contain x records. For intervals with fewer than x records, existing samples were replicated until the threshold was met. Conversely, in central intervals where the number of available records exceeded x, a number of samples greater than x but not exceeding $1.5 \times x$ was selected. This allowed the model to benefit from richer data in well-represented ranges while preventing excessive overrepresentation with respect to other intervals. The target value was chosen to ensure a sufficiently large total training set, while avoiding excessive duplication in less central intervals. To avoid augmenting potentially unreliable outliers, no naive augmentation was applied to the two extreme intervals (i.e., the lowest and highest). In those ranges, only the original records were retained. The pseudocode for this procedure is reported below:

Listing 4.1: Stratified Sampling and Naive Augmentation

```
# Divide SP range into N intervals, from min around 70 to max around
    190 mmHg
intervals = create_equal_intervals(SP_train_values, N)
# Set target number of samples per interval
x = choose_target_count(intervals)
augmented_dataset = []
for interval in intervals:
    records = get_records_in_interval(interval)
    if is_extreme_interval(interval):
```

```
        # Avoid augmenting potential outliers
        augmented_dataset.extend(records)
    elif len(records) < x:
        # Replicate records until reaching x
        reps = repeat_until(records, x)
        augmented_dataset.extend(reps)
    elif len(records) > x:
        # Allow slight overrepresentation in central ranges
        max_allowed = int(1.5 * x)
        selected = select_random_subset(records, max_allowed)
        augmented_dataset.extend(selected)
    else:
        # Interval already contains x records
        augmented_dataset.extend(records)
return augmented_dataset
```

Training

SNNs framweork and main application, are often focused on classifications task, therefore also the loss function from libraries like SNNtorch are more close and optimized for this kind of application. But in this case it was necessary to perform a regression task, therefore several approaches have been considered.
One initial approach was to reformulate the regression problem as a classification task, where each output neuron represents a discrete target value and the model is trained to maximize the firing activity of the neuron associated with the value closest to the true label. For this purpose, the cross-entropy spike count loss *ce_count_loss* from **snnTorch** was evaluated. This loss accumulates the total number of spikes emitted by each output neuron over all time steps and applies a cross-entropy loss to the resulting spike counts. As an example, in the case of SP estimation, the output layer consist of 100 neurons representing discrete values in the range from 70 to 190 mmHg. The 10^{th} neuron in this setting corresponds to a value of:

$$70 + \left(\frac{190 - 70}{100}\right) \cdot 10 = 82 \text{ mmHg}.$$

During inference, the predicted value is obtained by selecting the neuron with the highest firing rate and mapping it back to the corresponding SP value. Despite being natively supported and optimized for SNNs, this classification-based approach yielded unsatisfactory results. After a few training steps, several neurons tended to stop firing altogether, resulting in degraded learning behavior. The model would often converge to predicting average values close to the mean of the training set, which is indicative of poor generalization.
To address the limitations of classification-based approaches, more conventional regression techniques from the standard PyTorch environment were explored. The

best-performing strategy for the main proposed model involved combining two complementary loss functions. The first was a standard Mean Squared Error (MSE) loss, computed between the target value and a weighted sum of the output neuron values, where the weights correspond to their normalized firing rates. Let r_i be the normalized firing rate of neuron i and v_i its associated value. The predicted output $\hat{y}$ is given by:

$$\hat{y} = \sum_{i=1}^{N} r_i \cdot v_i,$$

and the MSE loss is:

$$\mathcal{L}_{\text{MSE}} = (y - \hat{y})^2,$$

where y is the ground-truth target. This regression-oriented formulation improved overall prediction accuracy and generalization. However, it still led to reduced activity in neurons associated with less frequent or extreme values, limiting robustness across the full range of outputs.
To counteract this, a second loss was introduced based on the Kullback-Leibler Divergence (KLDiv), computed between the vector of actual firing rates $\mathbf{r}$ and a target distribution $\mathbf{p}$ shaped as a soft peak centered near the neuron associated with the target value:

$$\mathcal{L}_{\text{KLDiv}} = D_{\text{KL}}(\mathbf{p} \parallel \mathbf{r}) = \sum_{i=1}^{N} p_i \log\left(\frac{p_i}{r_i + \epsilon}\right)$$

This distribution $\mathbf{p}$ was designed such that neurons closer to the target value were assigned higher firing rates, with progressively lower targets for more distant neurons, ensuring gradient presence and avoiding full suppression of non-target neurons.

The total loss was then defined as a weighted combination:

$$\mathcal{L}_{\text{total}} = \lambda \cdot \mathcal{L}_{\text{MSE}} + \mathcal{L}_{\text{KLDiv}}$$

where λ is an hyperparameter controlling the balance between the regression accuracy and neuron activity distribution. This combination allowed the model not only to minimize prediction error but also to maintain diverse and meaningful spiking activity across output neurons.

4.5 Hyper-parameters Tuning

In Spiking Neural Networks (SNNs), hyperparameter tuning is particularly sensitive due to the presence of biologically inspired parameters such as the membrane

potential decay rate β and the neuron firing threshold ϑ. The parameter β controls how quickly the neuron's membrane potential decays over time, while ϑ defines the minimum membrane potential required for a neuron to emit a spike. Due to the impact of these parameters on network dynamics and learning performance, an effective tuning strategy is essential. In this work, we employed the `Neural Network Intelligence (NNI)` library [33], which provides a flexible and efficient framework for automated hyperparameter optimization. A recommended pipeline and structure for tuning SNNs are discussed in [34], highlighting the importance of carefully defining the search space and choosing an appropriate tuning strategy. The search space determines which hyperparameters are explored and their respective ranges. An example of a defined search space using NNI is shown below:

Listing 4.2: Definition of search space in NNI

```
search_space = {
    'threshold_hidden': {'_type': 'quniform', '_value': [0.05, 1, 0.05]},
    'threshold_output': {'_type': 'quniform', '_value': [0.05, 1, 0.05]},
    'beta_hidden': {'_type': 'quniform', '_value': [0.3, 1, 0.05]},
    'beta_output': {'_type': 'quniform', '_value': [0.3, 1, 0.05]},
    'alpha': {'_type': 'quniform', '_value': [0.1, 0.8, 0.1]}#weight that balances the two loss functions,
    'max_val': {'_type': 'choice', '_value': [0.8,0.9,1.0 ]}, #Max values for the target distribution, starting from this the values around start to decay
    'decay': {'_type': 'choice', '_value': [0.25, 0.5, 0.75, 1.0]}#Firing rate reduction constant,
    'taper': {'_type': 'choice', '_value': [ 0.15, 0.25,0.35]}#Firing rate reduction speed,

}
```

The core component of the tuning process is the tuner, which determines the sequence and selection of parameter sets to evaluate. Since the search space in SNNs can be quite large and experiments are computationally expensive and with a limited duration, it is crucial to efficiently explore promising regions of the parameter space. For this purpose, the Anneal tuner was selected. As described in the official NNI documentation [33], this method operates as follows:

> "This simple annealing algorithm begins by sampling from the prior, but tends over time to sample from points closer and closer to the best ones observed. This algorithm is a simple variation on random search that leverages smoothness in the response surface. "

This approach balances exploration and exploitation effectively, making it a practical choice when tuning spiking networks with limited resources.

4.6 Neuronbench

Neuromorphic and brain-inspired computing have been subjects of research for several decades, but SNNs, their applications, and the associated hardware platforms have only recently experienced rapid growth. A major challenge in this emerging field is the lack of standardized benchmarks for assessing performance across the diverse hardware architectures and network structures. Such metrics are essential, especially considering that one of the primary applications of SNNs is real-time processing of biosignals on portable and low-power devices. Having reliable and comparable performance indicators is therefore fundamental. Moreover, the variety of existing techniques and hardware solutions complicates the identification of what actually is a neuromorphic solution. To address these issues, Neurobench[35] has been developed as an open-source benchmarking framework.Neurobench provides a standardized set of tools and metrics designed to evaluate neuromorphic algorithms in both hardware-agnostic and hardware-dependent settings.
For this work, the Neurobench toolkit was employed to perform hardware-independent evaluations of the proposed SNN models. This allowed us to quantify the impact of the network on various performance dimensions and gain insight into their suitability for future deployment. Neurobench metrics are divided into two main categories: The main categories of metrics considered include:

- **Static metrics:** These capture intrinsic properties of the network and hardware configuration, such as memory footprint and static power consumption.
- **Workload metrics:** These characterize dynamic aspects like spike activity, computation load, and latency during typical operational scenarios.

These metrics provide a comprehensive overview of the efficiency and scalability of the proposed models in realistic application settings. A detailed overview of the specific metrics considered in this thesis will be provided in the Results section.

4.7 Hardware Implementation

One of the additional goals of this work was to explore the feasibility of deploying the proposed SNNs on neuromorphic hardware. While current simulation frameworks allow for extensive experimentation in software, fully leveraging the potential of spiking neural networks requires dedicated neuromorphic chips. However, standardization in this domain is still lacking, and various hardware platforms are currently used in research environments. In this case, the implementation was carried out on the DYNAPSE platform, described in 3.4. The objective was to assess whether an almost one-to-one mapping of the trained network, while accounting for hardware constraints, could still lead to satisfactory results. The

network selected for implementation was the main proposed architecture, which uses features as input and firing rates as output representation.
The Dynapse chip has four cores, each composed of 256 analog spiking neurons. A fixed number of input neurons, in this case 20, were selected to encode the feature vector of each data sample. The activity of these neurons was controlled via an FPGA board. The output of each input neuron is defined as a spike train, generated through a rate encoding scheme. For each data record, the scalar feature value is converted into a predefined number of spikes distributed across a fixed duration. Unlike simulations, however, the Dynapse does not operate in discrete time steps. Therefore, the spike sequence must be translated into precise spike times. This requires defining a minimum temporal distance between spikes, effectively determined by the input spike frequency. This frequency parameter becomes crucial in determining how rapidly each input neuron emits spikes in response to high feature values. Once the spike timings for each neuron are computed based on the current data record, they are preloaded into the FPGA, which then manages the precise timing of spike emission.
One of the main challenges in implementing a Spiking neural network on neuromorphic hardware is the management of synaptic weights and their precision. On the chip, it is not possible to use continuous valued weights for the connections between neurons, instead, weights must be discretized to match the hardware constraints. Each neuron can receive inputs from multiple neurons, with a maximum fan-in of 64 synapses. Since all synapses are implemented using identical circuits, the only way to control the strength of a connection is by adjusting the number of parallel synapses between the source and target neurons. To map the continuous weights obtained from simulations to this hardware-limited setup, the following approach was adopted. First, for each postsynaptic neuron, the sum of the absolute values of its input weights is computed. The highest such value across the network is identified and used to define a scaling factor that ensures no neuron exceeds the maximum allowed fan-in of 64 synapses. Specifically, this maximum is scaled so that the sum of scaled weights for any neuron does not surpass 64. Each weight in the network is then multiplied by this scaling factor, producing a new set of weights bounded between 1 and 64. These scaled weights are subsequently rounded to the nearest integer. The final integer value determines the number of synapses to be instantiated in hardware for each connection. To make this process clearer, let $S = \max_j \sum_i |w_{ij}|$ be the maximum total input weight across all neurons. We define a scaling factor $\alpha = 64/S$ so that the rescaled weights $\alpha \cdot w_{ij}$ lie approximately within the range $[-64, 64]$. Each rescaled weight is then rounded to the nearest integer to determine the number of synaptic connections:

$$\tilde{w}_{ij} = \text{round}(\alpha \cdot w_{ij}).$$

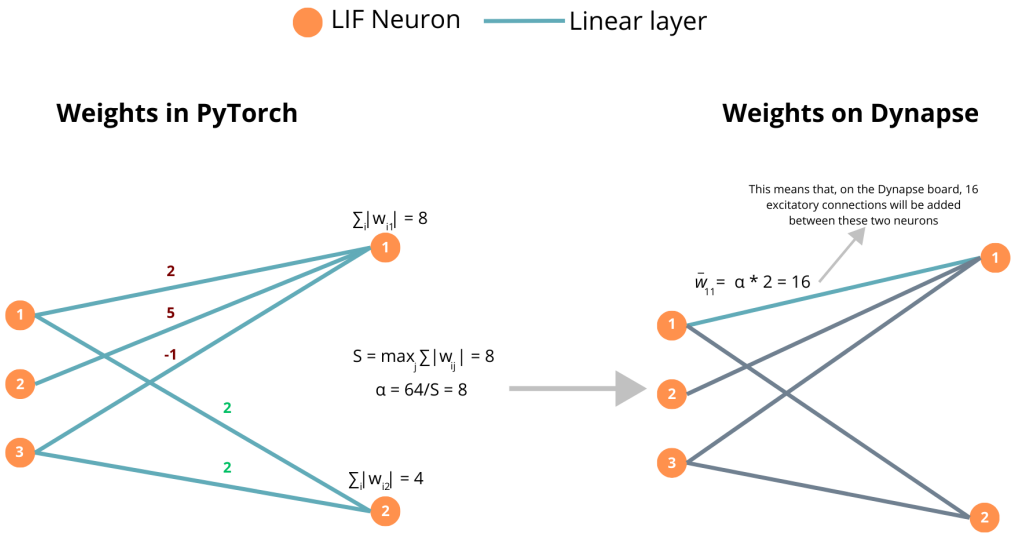


Figure 4.6: Example of weights transfer

Absolute values are used in the summation because a neuron can receive both positively and negatively weighted connections as input. On hardware, this distinction corresponds to the use of different types of synapses, excitatory and inhibitory. Excitatory synapses increase the likelihood of the post-synaptic neuron firing by contributing to the accumulation of input spikes, while inhibitory synapses reduce it. However, the total fan-in limit of a neuron on the chip is independent of the synapse type. Therefore, when mapping stronger negative weights, as well as stronger positive ones, multiple parallel synapses must be used to replicate the desired weight magnitude. As such, both excitatory and inhibitory connections must be included in the total fan-in budget, for this reason it is necessary the use of absolute values in the summation.
Another key aspects of implementing a SNN on neuromorphic hardware is the tuning of its configurable parameters. The Dynapse chip provides a wide range of tunable biases that directly influence the behavior of its circuits, including both neuronal and synaptic dynamics. These parameters are critical to align the hardware implementation with the behavior observed in simulation. Among the neuron-related parameters, `IF_RFR_N` sets the refractory period, determining the minimum time between two consecutive spikes, while `IF_TAU1_N` defines the main membrane time constant, affecting how quickly the neuron integrates input current. The parameter `IF_DC_P` corresponds to a DC bias current injected into the neuron, and `F_THR_N` adjusts the gain of the membrane potential (not the firing threshold itself), indirectly influencing the firing dynamics. Synaptic behavior is

governed by parameters such as NPDPIE_TAU_F_P and NPDPII_TAU_F_P, which set the time constants for fast excitatory (AMPA-like) and inhibitory (GABA-like) synapses, respectively. Their corresponding threshold parameters, NPDPIE_THR_F_P and NPDPII_THR_F_P, define the maximum current each synapse can produce. Additionally, PS_WEIGHT_EXC_F_N and PS_WEIGHT_INH_F_N control the effective strength of excitatory and inhibitory connections, while PULSE_PWLK_P specifies the duration of the synaptic pulse. All this parameters value are shared across the neurons and synapses of one core. Proper tuning of these parameters is essential, it is generally empirical and iterative, guided by prior knowledge of the network's target dynamics and the behavior observed in software simulations. It allows for precise control of spiking frequency, excitability, and temporal response.
The tuning pipeline adopted in this work began with a set of default empirical values provided in the official documentation [36]. These parameters are not mutually independent, thus, a set of constraint rules was maintained and referenced whenever a parameter was modified. For example, the value of IF_THR_N was typically set to 2–3 times the value of IF_TAU1_N, as suggested by prior documentation. To optimize the final configuration, two variants of a grid search were tested. In the first variant, the objective function to minimize was the mean absolute error (MAE) of the predicted label compared to ground truth values. In the second, the goal was to reduce the distance between the firing rates generated by the simulated network and those produced by the hardware implementation.
Regarding the output processing, since computation on the chip is not executed over a fixed number of time steps, as in software simulations, the adopted strategy was to monitor the output neurons over a defined observation window. This window was set as the sum of two components: the time required to transmit all input spikes, and an additional buffer period accounting for synaptic and neuronal dynamics, keeping track of the activity also slightly beyond the end of the stimulus. At the conclusion of this monitoring period, the timestamp of the last emitted spike was taken as a temporal reference. The entire interval was then discretized into a fixed number of bins, analogous to time steps in software simulations, and the spike count was accumulated within each bin. This bin-wise spike count served as a proxy for estimating the output firing rates of each neuron.

Chapter 5

Results

For training and testing, the `PyTorch` and `snnTorch` frameworks were employed, as previously described. Given the relatively small size of the SNN architecture, the number of training epochs was limited in comparison to other similar studies. In particular, five epochs were found to be sufficient to achieve good performance, allowing the network to learn effectively while avoiding overfitting. The `Adam` optimizer was used with a learning rate of 5×10^{-4}. Although the training pipeline followed the procedure detailed in Chapter 4, it is important to emphasize that the preliminary experiments were not conducted using the final loss configuration described in the Methodology. Instead, the initial focus was placed on identifying network configurations capable of reducing prediction error while maintaining low architectural complexity. Only at a later stage were broader evaluation aspects, such as robustness across datasets and distribution shifts, taken into account. The model that emerged from this process, referred to in previous sections as the *proposed model*, uses features as input and its firing rates are considered for the output. For this reason, some configurations shown in the Results section may exhibit lower error than the final proposed configuration, due to being optimized with different priorities and without considering some aspects.
Regarding the dataset, the four described in 3.3 has been used, BCG, Sensors, PPGBP, and UCI.

Accuracy Metrics

The metrics used to compare the proposed solutions, both internally and against state-of-the-art methods, are the same as those adopted in the benchmarking paper used as a reference [1]. These metrics are designed to provide a comprehensive and interpretable assessment of model performance.

Specifically, the following evaluation metrics were considered:

- **Mean Absolute Error (MAE)**: Measures the average magnitude of the absolute difference between predicted and true values. It is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i|$$

  where $\hat{y}_i$ and y_i are the predicted and true values, respectively.

- **Mean Error (ME)**: Computes the average signed difference between predicted and true values, providing a measure of prediction bias:

$$\text{ME} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)$$

- **Standard Deviation of the Error (SD)**: Quantifies the variability of the prediction errors, reflecting the consistency of the predictions:

$$\text{SD} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left((\hat{y}_i - y_i) - \text{ME}\right)^2}$$

- **Mean Absolute Scaled Error (MASE)**: A scale-independent error metric adapted to the task, introduced in [1]. It compares the MAE of the model to the MAE of a naive baseline that always predicts the mean value of the target from the training set:

$$\text{MASE} = \frac{\text{MAE}_{\text{model}}}{\text{MAE}_{\text{naive}}} \cdot 100$$

  This metric is particularly useful for comparing models across different datasets, where error magnitudes may vary significantly. In this work, due to the use of stratified sampling and naive oversampling in underrepresented ranges, the naive baseline calculated directly from the training set may not be reliable. Therefore, to enable fair comparison and avoid overly optimistic estimates, the reference values reported in [1] were used as the denominator for MASE, rather than those obtained from the modified training distribution.

Neuron Type

As described in the previous section, different types of neurons are available in the `snnTorch` environment. In this study, we tested three main neuron models:

the Leaky Integrate-and-Fire (LIF), the Recurrent LIF, and the LSTM-LIF. The LIF neuron simulates the decay of the membrane potential over time and emits a spike when a predefined threshold is reached, thereby capturing the basic dynamics of biological neurons. The Recurrent LIF extends this behavior by introducing recurrent connections, allowing neurons to retain information across time steps, making it more suitable for modeling temporal dependencies. The LSTM-LIF, in contrast, combines spiking dynamics with the gating mechanisms typical of Long Short-Term Memory (LSTM) networks, enabling the network to selectively store and propagate information through time. The objective in this case was to evaluate whether introducing additional complexity through recurrence and memory mechanisms could lead to performance improvements. The final results on the BCG dataset are summarized in Table 5.1. Although the use of more complex neuron models led to slightly improved performance in terms of mean absolute error (MAE), the gains were not significant enough to justify the increased computational overhead. One of the core advantages of SNNs is their lightweight nature, and in this context, the limited accuracy improvement did not outweigh the added complexity. Furthermore, introducing more complex structures in small networks such as this one can increase the risk of overfitting.

Table 5.1: Comparison of neuron models on the BCG dataset.

Neuron Type	MAE SP	MAE DP
LIF	11.96	7.26
Recurrent LIF	12.00	7.78
LSTM-LIF	11.52	7.64

Brain inspired techniques

Since SNNs aim to replicate various behaviors of the human brain, some biologically inspired testing mechanisms have also been explored. One such approach is the implementation of a mechanism similar to Winner-Take-All (WTA). This was achieved by enabling a specific parameter available in `snnTorch`, called `inhib`. When `inhib` is active, after a neuron fires in response to a stimulus, all other neurons in the same layer are temporarily inhibited from spiking. This resembles certain neural dynamics observed in the brain, where inhibitory mechanisms prevent widespread activation, allowing only the most strongly stimulated neuron (or group) to dominate the response.
In biological systems, WTA dynamics often occur in the context of population coding, for example, in classification tasks where multiple neurons represent each class. In such scenarios, when one neuron from a given population spikes, other

populations are suppressed. In this case, the setup differs slightly: each output neuron corresponds to a specific value rather than a class, and neurons operate on discrete time steps. At each step, the neuron with the highest membrane potential above the spiking threshold is allowed to fire, while the others are inhibited. This allows for a form of temporal competition where only the most strongly activated neuron spikes per time step.
Biological neural circuits are typically characterized by sparse rather than fully connected architectures. Inspired by this and considering hardware limitations in neuromorphic systems, a variant of the proposed model was tested, where sparse connectivity was introduced between layers. This was implemented using a custom `MaskedLinear` class, which extends the standard linear layer by applying a binary mask to the weights. The mask defines which synapses are active, and is fixed at initialization. In our architecture, different sparsity levels were applied: 30% connectivity from input to the first hidden layer, and 15% in the subsequent layers. Formally, the masked weights at each layer are computed as:

$$W_{\text{masked}} = W \odot M$$

where W is the weight matrix, M is the binary mask, and $\odot$ denotes element-wise multiplication. The rest of the training pipeline, including input encoding and prediction via weighted firing rate decoding, remains unchanged.

Table 5.2: Comparison of methodes on the BCG dataset.

Model Type	**MAE SP**	**MAE DP**
Refernce	11.96	7.26
WTA	11.81	7.59
Sparse	14.38	7.64

As shown in Table 5.2, the tested methods did not lead to any improvement in prediction error compared to the baseline model.

General discussion and benchmark comparison

In this section, the results of the specific task of blood pressure (BP) prediction are examined and compared against existing benchmarks. In [1], several machine learning (ML) and deep learning (DL) models were tested and evaluated on the same datasets used in this project, providing baseline results suitable for comparison. 5.2 and 5.3 reports the results obtained by the best-performing model in terms of generalization and error, as well as the benchmark models

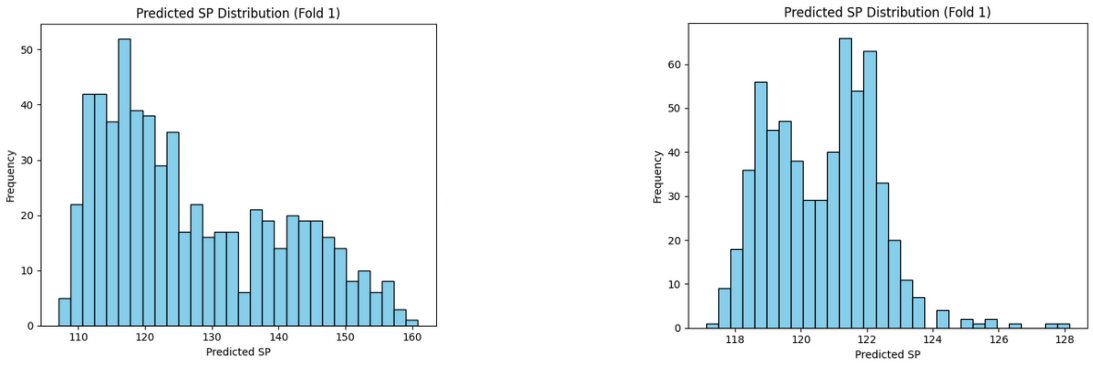
Before proceeding with the general comparison, it is useful to review the different models presented in Chapter 4 in order to justify the choice of the model proposed in this work. Among the three presented configurations, the model that uses handcrafted features as input and firing rates as output will be referred to as **Model P** (proposed). The model that uses raw signals as input and firing rates as output will be referred to as **Model F**, while the model using raw signals as input and the membrane potential of the last two neurons as output will be referred to as **Model M**. Table 5.3 summarizes the performance of these three models in terms of Mean Absolute Error (MAE), Mean Error (ME), and Standard Deviation (SD) of the error.

Table 5.3: Comparison of proposed models on both BCG and Sensor datasets. Metrics include MAE and ME ± SD for systolic (SP) and diastolic (DP) pressure predictions.

Dataset	**Model**	**MAE (SP)**	**MAE (DP)**	**ME ± SD (SP)**	**ME ± SD (DP)**
BCG	Model P (Features → FR)	12.97	7.33	0.43 ± 16.43	0.50 ± 8.12
	Model F (Raw → FR)	14.39	8.23	-0.12 ± 20.00	1.19 ± 10.52
	Model M (Raw → MemPot)	16.43	9.14	3.07 ± 20.67	-1.78 ± 11.38
Sensors	Model P (Features → FR)	16.60	7.76	-1.06 ± 20.72	0.34 ± 10.00
	Model F (Raw → FR)	14.93	8.58	-2.89 ± 20.37	1.30 ± 11.01
	Model M (Raw → MemPot)	12.63	7.65	2.52 ± 16.90	-0.72 ± 9.76

In Table 5.3, it can be observed that the proposed model **P** is not always the one with the lowest error. However, it was selected as the reference model for further improvements and for comparison with the benchmark. The reason for this choice lies in the nature of the task, predicting blood pressure (BP) and evaluating such predictions is inherently delicate. In this context, the MAE alone is not a sufficient performance indicator. Some models can achieve very low MAE values simply by consistently predicting values close to the mean of the test set. This leads to a network that produces nearly constant predictions within a narrow range (e.g., 2-3 units), which is undesirable. While the standard deviation (SD) of the predictions can be an indicator of this issue, it is often insufficient on its own. A more reliable approach is to analyse the distribution of the predictions made during testing. Naturally, these distributions are not uniform, and most predictions will cluster around certain physiological ranges. However, a robust model should also be able to generalize and produce values that are not concentrated only near the mean. This criterion was therefore included in our evaluation process, leading to select the proposed model **P**, which offers an acceptable error while demonstrating adequate generalization capabilities. Figure 5.1 illustrates this issue. On the left, predictions made by model **P** on the Sensors dataset are shown, the distribution is comparable to that of a classical neural network and does not collapse to a single

mean value. On the right, predictions made by model **F** on the same test fold of the Sensors dataset are shown, despite the very low error, the model in this case is less reliable due to limited prediction variability.



(a) P model predictions

(b) F model predictions

Figure 5.1: Comparison between generalization capabilities

The following part is dedicated to the comparison between proposed model and several classical neural network models reported in [1]. Specifically, first, the focus is on the MAE metric.

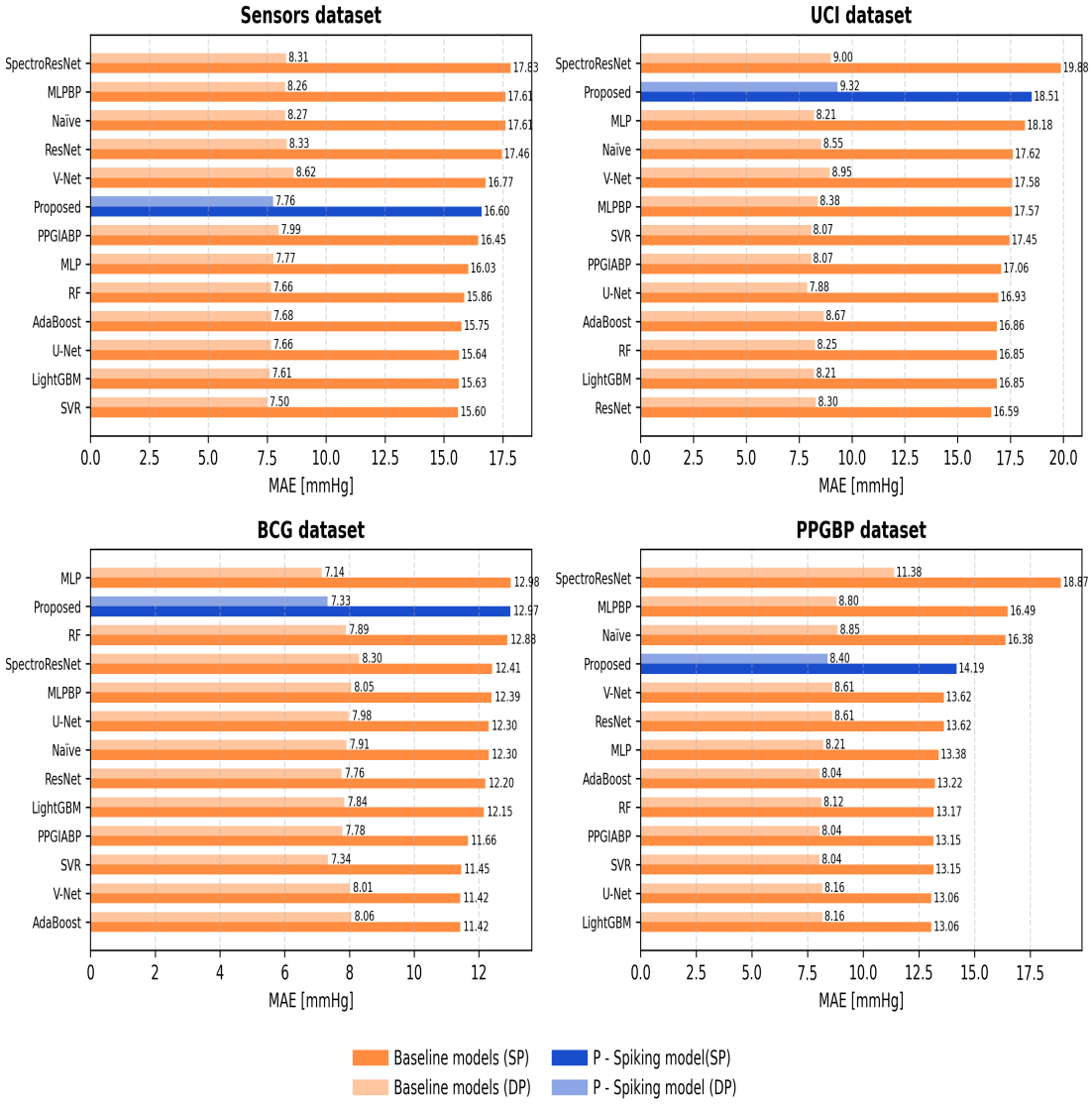


Figure 5.2: Mean Absolute Error(MAE) comparison

As can be observed, the proposed model **P**, despite not achieving the lowest errors, has a performance that is comparable to the other benchmark models. This demonstrates that, at this level, SNNs can compete with classical ML and DL architectures. In the case of the DP prediction on the BCG dataset, **P** ranks among the most accurate models, achieving the second best error.
Also, analysing the ME and SD in Figure 5.3, the proposed model performs well, maintaining values close to those of the benchmark models in most cases and, in some datasets, ranking among the top-performing approaches.

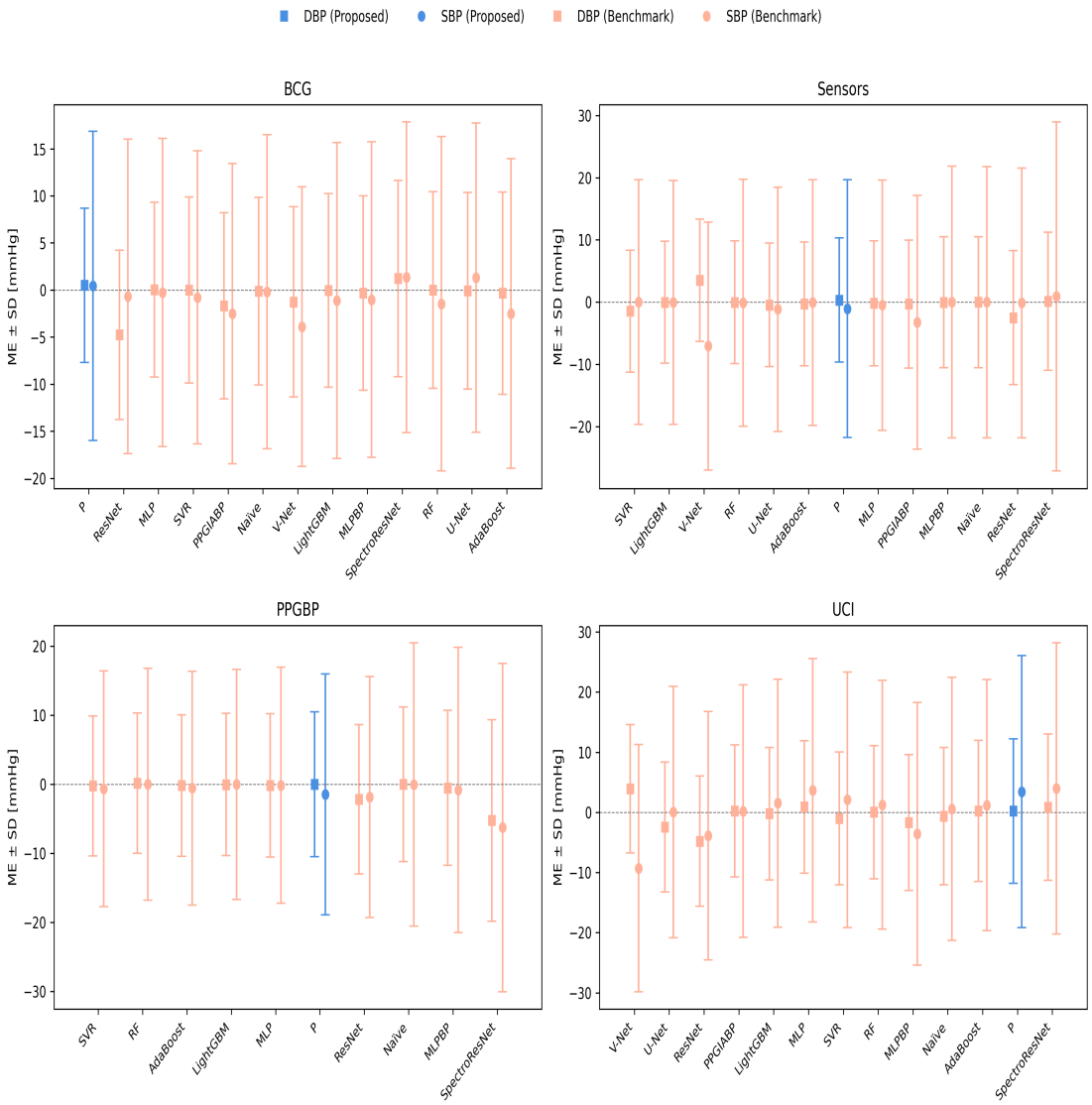


Figure 5.3: Comparison of Mean Error (ME) and Standard Deviation (SD) across datasets for benchmark models and the proposed approach.

Finally, to complete the comparison with the results reported in [1], Table 5.4 presents the obtained MASE values for each dataset. The table also reports the mean MASE computed across the various benchmark models included in the previous figures.

The previous results provided a general overview of the prediction errors and the position of SNNs in comparison to ML and DL architectures. Although SNNs achieved good results, they were not the best-performing models. However, their

Table 5.4: Comparison of MASE values for the proposed model and the mean MASE of the benchmark models from [1]. Lower values indicate better performance.

Dataset	P-MASE SP	MB-MASE SP	P-MASE DP	MB-MASE DP
Sensors	94.26	93.26 ± 4.67	93.83	96.10 ± 4.33
UCI	105.05	98.93 ± 5.05	109.00	97.82 ± 4.04
BCG	105.45	98.49 ± 4.35	92.67	99.02 ± 4.02
PPGBP	86.63	87.74 ± 12.25	94.92	97.97 ± 11.94

*MB-MASE stands for Mean Benchmark MASE

main advantage lies not only in prediction accuracy, but particularly in their lower complexity, which translates into reduced computational cost. In [32], the authors built upon the benchmark established in [1], presenting and analysing new models and their physical deployment while also considering model complexity. For this purpose, they evaluated not only prediction errors, such as MAE, but also the number of parameters.

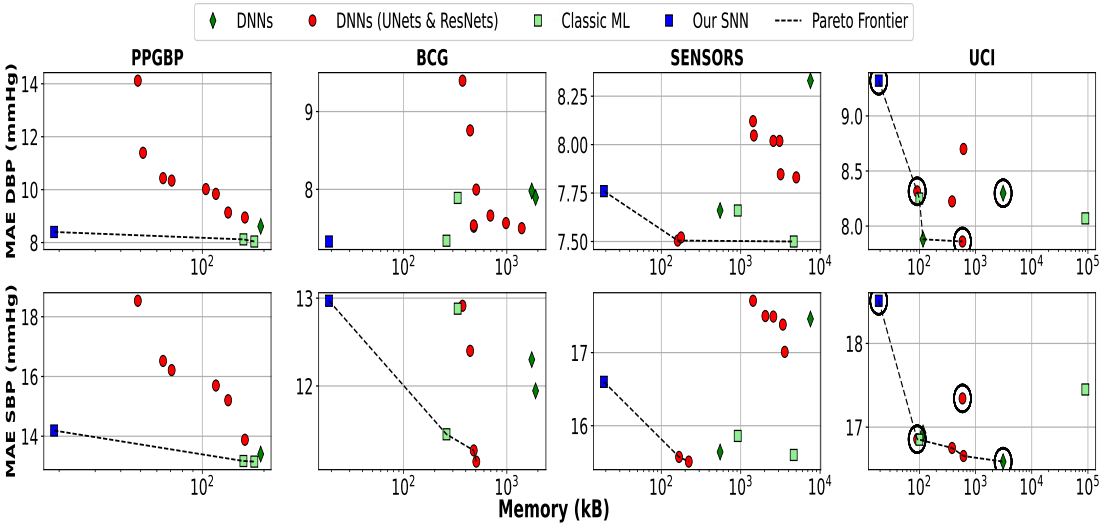


Figure 5.4: Parameters - MAE comparison

From Figure 5.4, the proposed model highlights the strength of SNNs, in addition to achieving good overall results, its number of parameters is orders of magnitude lower compared to the most accurate deep learning models.

Neurobench and Hyperparameters

Alongside the evaluation of the metrics related to BP estimation, an additional assessment of the proposed model was carried out using the Neurobench toolkit, as introduced in the previous chapter. Since different models were trained and tuned separately for each dataset, it was first necessary to report the hyperparameters of the models being evaluated.
For each dataset, a nested cross-validation approach was adopted using the NNI framework described previously, meaning that for each fold, a separate set of hyperparameters was obtained. The Neurobench tests were run independently on each of these configurations. In Table 5.5, the average values of the main hyperparameters for each dataset are reported. These values, the threshold(θ) and the decay(β) for each neuron layer, can strongly influence the behavior of the network, and are therefore important to highlight before analyzing the performance results.

Dataset	β_{first_layer}	β_{output_layer}	θ_{first_layer}	θ_{output_layer}
BCG	0.57	0.74	0.52	0.54
Sensors	0.59	0.71	0.51	0.53
PPGB	0.77	0.61	0.31	0.46
UCI	0.80	0.40	0.45	0.75

Table 5.5: Average hyperparameters

As described in Chapter 4, for the hardware-independent evaluation of the proposed SNN models, we adopted the *static* and *workload* metrics provided by the Neurobench toolkit. These metrics offer insight into the model's computational and structural complexity, independent of any specific neuromorphic hardware. The following metrics were considered:

- **Footprint**: Represents the total memory required to store the model at inference time. This includes both the model's trainable parameters and its internal buffers, which may store recurrent states, spiking dynamics, or temporary input representations;

- **Parameter Count**: Denotes the total number of trainable parameters in the model;

- **Connection Sparsity**: Measures the proportion of zero-valued weights in the connection matrices. A higher sparsity indicates a lower degree of connectivity;

- **Activation Sparsity**: Refers to the proportion of neurons that remain inactive during inference across time. This metric provides insight into the dynamic sparsity of the network and its energy efficiency;
- **Synaptic Operations**: Quantifies the number of operations carried out at the synaptic level during inference. It includes three components:
  - *Effective MACs*: Multiply and accumulate operations performed on non-zero, active synaptic connections;
  - *Effective ACs*: Accumulate only operations, where the no multiplication is done;
  - *Dense*: The total number of operations that would be performed under full connectivity, used as a baseline for comparison.
- **Membrane Updates**: Counts how often a neuron's membrane potential is updated during inference;
- **Neuron Operations**: Refers to the total number of operations performed within neuron units:
  - *Neuron MACs*: Multipl and accumulate operations related to internal state updates of neurons.
  - *Neuron Dense*: The total neuron related operations assuming full activity and no sparsity;

Table 5.6: Average Neurobench Metrics for the Proposed Model on the BCG Dataset

Metric	Average Value
Footprint (bytes)	19404
Parameter Count	4841
Connection Sparsity	0.0
Activation Sparsity	**0.87**
Effective MACs	0.0
Effective ACs	24201
Dense Synaptic Ops	115500
Membrane Updates	5525
Neuron MACs	11050
Neuron Dense Ops	11050

The activation sparsity clearly indicates that, on average, more than 85% of neurons remain inactive during inference across time and layers. This highlights the potential efficiency of the proposed model on neuromorphic-specific hardware. Regarding connection sparsity, although the current value is zero due to the nature of the training procedure, this metric could potentially be improved. Since weights are updated via gradient descent, they rarely reach an exact zero, even if their absolute value becomes very small. This explains the absence of true disconnections in the simulation. Another important metric is the one related to synaptic operations. As shown, only accumulate operations are performed at the synaptic level, with no multiplication involved. This is a positive aspect, as accumulate-only operations are significantly more efficient. particularly in neuromorphic systems that aim to reduce energy consumption and computational complexity.

Hardware Deployment

The implementation on the DynapSE followed the steps described in Section 4.7. The goal was to verify whether an approximately one-to-one mapping of the networks obtained in simulation could yield good results on this specific hardware. After multiple tests, using the various strategies described, it was not possible to obtain meaningful results. Following tuning aimed at reducing the gap between the theoretical and simulated firing rates, starting from the first neurons layer, on the BCG dataset, the error was significantly reduced. However, the resulting network was not usable, as it simply tended to predict values close to the average SP and DP values, leading to hardware that essentially produced fixed outputs.
One of the main reasons for this behavior lies in the substantial difference in weight precision. When moving to hardware, quantization is required, and the mapping is not always linear, as previously described. On-chip, every synaptic connection shares the same weight value. To make a connection stronger than another, the only available option is to add multiple parallel connections. However, as observed during experimentation, this does not always correspond to the expected proportional increase. For example, if an input neuron has two excitatory connections to a receiving neuron A and one to a neuron B, with a constant input over time, the firing rate of A is not double that of B. This nonlinearity is one of the key challenges to consider when mapping networks onto the hardware. Another difficulty lies in the large number of physical hyperparameters on which the chip's behavior depends. These parameters exceed those considered in simulation and also exhibit mutual dependencies, which further complicates the tuning process. Moreover, each parameter typically has a narrow, connection and network dependent range of effective operation. Finetuning directly on hardware is challenging because it requires real-time signal processing, a factor that is largely negligible in simulation.

As a result, exploring the entire parameter space becomes impractical, often leading to convergence to local minima. Therefore, specific strategies must be adopted to significantly reduce the search space.
Despite its limitations, neuromorphic hardware, and DynapSE in particular, has already demonstrated its effectiveness across a wide range of applications, achieving significant results in terms of both accuracy and efficiency. Also during preliminary studies, tests were conducted to investigate whether neuromorphic hardware could be used to extract meaningful information from PPG signals. For this purpose, basic configurations of the DynapSE were evaluated, with the spikes produced by the output neurons of the spiking network serving as input to classical ML and DL models. Some promising results were obtained, highlighting the potential of such chips. However, these results are not presented here, as they were limited in scope, not part of a larger study, and merely served as an initial exploratory analysis. Moreover, hybrid approaches combining SNNs with classical models were not within the scope of the present thesis work. This is mentioned to emphasize that the field of neuromorphic computing, particularly the application of SNNs to PPG signal analysis, offers significant opportunities for further research

Chapter 6

Conclusion

In this work, we presented an approach for estimating blood pressure from a PPG signal using Spiking Neural Networks, a third-generation neural architecture that enables more efficient computation compared to classical models. Such efficiency makes them potentially crucial for applications involving the prediction of biological parameters from signals acquired through wearable devices. This is a particularly important field, as non-invasive and continuous BP measurement could not only improve the remote monitoring of patients without the need for dedicated medical staff or equipment, but also provide healthy individuals with a continuous monitoring tool that may help prevent and predict complications before they arise, particularly in relation to cardiovascular conditions.
The primary objective of this thesis was not to surpass the accuracy of existing benchmarks, but rather to investigate whether SNNs can achieve results comparable to those of more computationally complex models, thereby demonstrating their potential as an efficient alternative on certain tasks.
Different model variants were tested, varying both the type of spiking neurons and the choice of input and output features. The final proposed model is a Spiking Neural Network based on Leaky Integrate-and-Fire (LIF) neurons, which receives as input handcrafted features that have been properly pre-processed. The prediction is obtained exploiting the firing rates of the neurons in the network's final layer. Although several types of encoding are possible, in this work a rate encoding scheme was adopted, in which the information is represented by mapping the input over a spike trains of a fixed number of steps or a fixed duration, and transforming the input value into a corresponding firing rate.
To evaluate the performance of the proposed models, the benchmark presented in [1] was taken as a reference. In that work, multiple classical ML and DL models were tested on the BP estimation task, providing a solid baseline for comparison. The datasets used for training and testing were BCG, Sensors, PPGBP, and UCI, which contain data from both healthy subjects and patients, the majority of whom were

in intensive care units or had underlying health conditions. This variety provides diverse physiological profiles under different conditions, enabling a comprehensive evaluation of model performance. To avoid overly optimistic or biased results, a nested cross-validation procedure was adopted for training and validation, using the folds defined in [1], which ensured that data from the same patient did not appear across different folds, thereby preventing data leakage.
The strategy adopted to oversample data in underrepresented ranges, combined with the use of two presented different loss functions, enabled the achievement of very good results in terms of both error and generalizability. The test errors across all metrics were in line with those reported in the benchmark, and in some cases, such as the prediction of DP on the BCG dataset, the proposed model ranked as the second-best performing.
Since SNNs are inherently designed for hardware implementation, we also carried out a preliminary evaluation from this perspective. Using NeuroBench metrics, the model achieved an activation sparsity above 85%, indicating promising energy-efficiency potential. Furthermore, taking into account the work presented in [32], where different models were evaluated not only for their accuracy but also for their complexity, it was performed a comparison using the data provided in that study. Their analysis plotted parameters against error, and in this space the proposed model achieved an average error compared to the models presented, while requiring at least one order of magnitude fewer parameters than the DL models included.
Part of this work involved transferring the simulated SNN onto hardware, using the DynapSE chip [2]. However, due to the challenges discussed in Section 5 including weight quantization effects, non-linear synaptic mapping, and the tuning of interdependent hardware parameters, the results obtained through the different mapping and tuning strategies were not conclusive. These findings highlight the complexity of deploying SNNs on neuromorphic hardware and leave room for further investigation.
This work lies within two rapidly evolving and potentially impactful research areas, the estimation of biological signals from data acquired through wearable devices, and the study and application of Spiking Neural Networks themselves. Naturally, even when focusing on a single task, it is not possible to explore every possible direction. Starting from this thesis, a few different directions for future work stand out:

- **Expanding the diversity of training data.** The under-representation of extreme blood pressure values is a limitation. Alternatively, new data augmentation techniques could be explored, aiming to better correlate PPG signals with BP values;
- **Including more data from healthy subjects.** Most current datasets are dominated by patients with cardiovascular issues, which may lead to

PPG signals with distortions. Training networks initially on large quantities of healthy subject data could provide a more stable baseline, reducing the influence of such distortions;

- **Exploring hybrid diagnostic–predictive models.** Before estimating BP, models could first detect potential cardiovascular conditions and then adjust the prediction accordingly. This could improve accuracy, given that patients with different conditions may have distinct cardiac dynamics. In this context, it would also be relevant to investigate whether starting from a general model and performing fine-tuning for individual patients wearing the device would be beneficial. As highlighted in [1], data leakage of the same patient across folds can lead to overly optimistic cross-validation results, while this warns us to take care in evaluating general-purpose methods, it also suggests that tailored per-patient applications could be an intersting direction;
- **Further investigation of hardware deployment on DynapSE.** Although the attempts made in this work did not yield positive results, this does not mean that slightly modified fine-tuning strategies could not lead to mappings with behaviour closely matching the simulations;
- **Testing on other neuromorphic chips.** While the DynapSE already enables remarkable experiments, for this specific task other hardware platforms may prove more suitable;

Appendix

Table 1: Description of the features selected and used for training

Category	Feature(s)	Signal	Definition and Physiological Interpretation
Cardiac Timing	T_c	PPG	Full cardiac cycle duration (valley-to-valley interval)
	T_d	PPG	Diastolic duration (systolic peak to end of cycle).
	T_b	APG	Timing of the b wave in the second derivative
	$T_{\text{peak},e}$	PPG–APG	Time delay between systolic peak and APG e wave (absolute and normalized).
	$T_{\text{SystoDiaRise}}$	PPG/VPG	Delay from systolic peak to diastolic rise.
	T_{diaRise}	VPG	Timing of diastolic positive slope after systolic decay.
	$T_{\text{NegSteepest}}$	VPG	Timing of maximum negative slope during diastole. Reflects blood deceleration phase.
Slope Amplitudes	SteepDiaRise	VPG	Amplitude of diastolic positive slope (blood re-acceleration).
APG Amplitudes and Ratios	apg_e	APG	Amplitude of e wave
	ratio_apg_e	APG	e/a amplitude ratio.

Continued on next page

Category	Feature(s)	Signal	Definition and Physiological Interpretation
Area Features	S_1, S_3, S_4	PPG	Segmented systolic/diastolic areas under the pulse waveform. Normalized versions divided by total AUC.
	AUC_{sys}, AUC_{dia}	PPG	Total systolic and diastolic area under the curve.
Width Ratios	DW25, DW/SW25	PPG	Diastolic width at 25% amplitude and diastolic-to-systolic width ratio.
Spectral / Neighbor Features	ppg_max_neighbor_mean_0		Mean amplitude around dominant peak of average cycle.
	vpg_max_neighbor_mean_0		Equivalent neighborhood amplitude in first derivative domain.
	ppg_fft_peaks_heights_0		Height of dominant frequency component.
	ppg_fft_peaks_neighbor_avgs_0		Mean spectral magnitude around dominant frequency.
Histogram Morphology	ppg_histogram_down_{2,4}		Selected bins of distribution density during diastolic decay.
	vpg_histogram_down_{0,1,2}		Histogram bins of first derivative during decay phase.
	vpg_histogram_up_2		Histogram bin of systolic upstroke slope distribution.
	apg_histogram_down_{2,6}		Histogram bins of second derivative during decay.
	ppg3_histogram_down_{6,7,8}		Morphological bins of higher-order(3^{rd}) derivative during diastole.
	ppg4_histogram_down_{3,4}		Higher-order(4^{th}) curvature descriptors of diastolic decay.
Deviation Curves	dsdc_{1,8,13}	PPG	DownSlope Deviation Curve samples describing deviation from ideal exponential decay.

Continued on next page

Category	**Feature(s)**	**Signal**	**Definition and Physiological Interpretation**
Signal Quality	SQI_kurtosis	PPG	Kurtosis of normalized pulse cycle. Detects abnormal morphology or artifacts.

Bibliography

[1] G. Sergio, H. Wan-Ting, and Trista. P. «Comprehensive Benchmark for Blood Pressure Estimation Using PPG». In: *Under Review* (2023) (cit. on pp. 2, 17, 18, 30, 39, 40, 42, 44, 46, 47, 52–54).

[2] S. Moradi, N. Qiao, F. Stefanini, and G. Indiveri. «A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)». In: *IEEE Transactions on Biomedical Circuits and Systems* 12.1 (2018), pp. 106–122 (cit. on pp. 2, 19, 24, 53).

[3] F. Rosenblatt. «The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain». In: *Psychological Review* 65.6 (1958), pp. 386–408 (cit. on p. 5).

[4] W. Maass. «Networks of Spiking Neurons: The Third Generation of Neural Network Models». In: *Neural Networks* 10.9 (1997), pp. 1659–1671 (cit. on pp. 5, 7).

[5] J K. et al. Eshraghian. «Training Spiking Neural Networks Using Lessons from Deep Learning». In: *Proceedings of the IEEE* 109.5 (2021), pp. 769–805 (cit. on p. 8).

[6] B. Yin, F. Corradi, and K. Boahen. «Accurate and Efficient Time-Domain Classification with Adaptive Spiking Recurrent Neural Networks». In: *Nature Machine Intelligence* 3.10 (2021), pp. 905–913 (cit. on p. 8).

[7] John Allen. «Photoplethysmography and Its Application in Clinical Physiological Measurement». In: *Physiological Measurement* 28 (2007), R1–R39. DOI: 10.1088/0967-3334/28/3/R01 (cit. on pp. 9, 17).

[8] M. A. Almarshad, S. Islam, S. Al-Ahmadi, and A.. BaHammam. «Diagnostic Features and Potential Applications of PPG Signal in Healthcare: A Systematic Review». In: *Healthcare* 10.3 (2022), p. 547 (cit. on pp. 9, 17).

[9] E. Pollreisz and M. TaheriNejad. «Detection and Removal of Motion Artifacts in PPG Signals». In: *Sensors* 19.3 (2019), p. 673 (cit. on p. 9).

[10] World Health Organization. *Hypertension Fact Sheet*. https://www.who.int/news-room/fact-sheets/detail/hypertension. 2023 (cit. on pp. 10, 11).

[11] NCD Risk Factor Collaboration (NCD-RisC). «Worldwide Trends in Hypertension Prevalence and Control from 1990 to 2019». In: *The Lancet* 398.10304 (2021), pp. 957–980 (cit. on pp. 10, 11).

[12] Paul K. Whelton et al. «2017 ACC/AHA Guideline for the Prevention, Detection, Evaluation, and Management of High Blood Pressure in Adults». In: *Hypertension* 71.6 (2018), e13–e115 (cit. on p. 11).

[13] F. Ricci et al. «Orthostatic Hypotension: Epidemiology, Prognosis, and Treatment». In: *Journal of the American College of Cardiology* 66.7 (2015), pp. 848–860 (cit. on p. 11).

[14] E. Donati, M. Payvand, N. Risi, and et al. «Discrimination of EMG Signals Using a Neuromorphic Implementation of a Spiking Neural Network». In: *IEEE Transactions on Biomedical Circuits and Systems* 13.5 (2019), pp. 795–803 (cit. on p. 13).

[15] M. A. Scrugli, G. Leone, P. Busia, and P. Meloni. «sEMG-Based Gesture Recognition with Spiking Neural Networks on Low-Power FPGA». In: *Design and Architectures for Signal and Image Processing (DASIP 2023/2024 Proceedings)*. Springer, 2024 (cit. on p. 13).

[16] S. S. Bezugam, A. Shaban, and M. Suri. «Low Power Neuromorphic EMG Gesture Classification». In: *arXiv* (2022). eprint: arXiv:2206.02061 (cit. on p. 13).

[17] Z Yan, J. Zhou, and W. Wong. «Energy Efficient ECG Classification with Spiking Neural Network». In: *Biomedical Signal Processing and Control* 63 (2021), p. 102170. DOI: 10.1016/j.bspc.2020.102170. URL: https://arxiv.org/abs/2006.12016 (cit. on p. 14).

[18] A. Amirshahi and M. Hashemi. «ECG Classification Algorithm Based on STDP and R-STDP Neural Networks for Real-Time Monitoring on Ultra Low-Power Personal Wearable Devices». In: *IEEE Transactions on Biomedical Circuits and Systems* 13.6 (2019), pp. 1483–1493. DOI: 10.1109/TBCAS.2019.2948920 (cit. on p. 14).

[19] Z. Yan, Z. Bai, T. Mitra, and W. Wong. «SparrowSNN: A Hardware/Software Co-Design for Energy Efficient ECG Classification Using Spiking Neural Networks». In: *arXiv preprint* (2024). eprint: arXiv:2406.06543 (cit. on p. 14).

[20] K. Burelo, G. Ramantani, G. Indiveri, and J. Sarnthein. «A Neuromorphic Spiking Neural Network Detects Epileptic High-Frequency Oscillations in the Scalp EEG». In: *Scientific Reports* 12 (2022), p. 1798 (cit. on p. 14).

[21] M. Sharifshazileh, K. Burelo, J. Sarnthein, and G. Indiveri. «An Electronic Neuromorphic System for Real-Time Detection of High-Frequency Oscillations (HFO) in Intracranial EEG». In: *Nature Communications* 12 (2021), p. 3094 (cit. on p. 14).

[22] Yikai Yang, Jason K. Eshraghian, Nhan D. Truong, Armin M. Nikpour, and Omid Kavehei. «Neuromorphic Deep Spiking Neural Networks for Seizure Detection». In: *Neuromorphic Computing and Engineering* (2023) (cit. on p. 15).

[23] K. Yang, H. Kang, P. Charlton, P. A. Kyriacou, J. Kim, Y. Li, and C. Park. «Energy-Efficient PPG-Based Respiratory Rate Estimation Using Spiking Neural Networks». In: *Sensors* 24.12 (2024), p. 3980 (cit. on p. 15).

[24] C. De Luca, M. Tincani, G. Indiveri, and E. Donati. «A Neuromorphic Multi-Scale Approach for Real-Time Heart Rate and State Detection». In: *npj Unconventional Computing* 2.1 (2025), p. 6 (cit. on p. 15).

[25] I. Shaw and et al. «A Real-Time Analysis of PPG Signal for Measurement of SpO2 and Pulse Rate». In: *International Journal of Engineering Trends and Technology* 24.3 (2015), pp. 120–124 (cit. on p. 16).

[26] M. Lee, Y. K. Lee, D. S. Pae, M. T. Lim, D. W. Kim, and T. K. Kang. «Fast Emotion Recognition Based on Single Pulse PPG Signal with Convolutional Neural Network». In: *Applied Sciences* 9.16 (2019), p. 3355 (cit. on p. 16).

[27] Y. Liang, Z. Chen, R. Ward, and M. Elgendi. «Hypertension Assessment via ECG and PPG Signals: An Evaluation Using MIMIC Database». In: *Diagnostics* 8.3 (2018), p. 65 (cit. on p. 16).

[28] C. M. Yang, C. Veiga, J. J. Rodriguez-Andina, J. Farina, A. Iniguez, and S. Yin. «Using PPG Signals and Wearable Devices for Atrial Fibrillation Screening». In: *IEEE Transactions on Industrial Electronics* 66.11 (2019) (cit. on p. 16).

[29] George B. Moody and Roger G. Mark. *MIMIC-III Waveform Database*. `https://physionet.org/content/mimic3wdb/1.0/`. 2020 (cit. on p. 17).

[30] Mehrdad Kachuee, Mohammadhassan M. Kiani, Hamed Mohammadzade, and Mohammad Shabany. «Cuff-less High-Accuracy Calibration-Free Blood Pressure Estimation Using Pulse Transit Time». In: *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2015, pp. 1006–1009. DOI: `10.1109/ISCAS.2015.7168806` (cit. on p. 18).

[31] Ryan Carlson, Alexander Greco, Andrew McComas, and Ankur Shah. «A Bed-Based Dataset for Blood Pressure Estimation Using Ballistocardiogram and Photoplethysmography». In: *Proceedings of the 43rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2021, pp. 1234–1237. DOI: 10.1109/EMBC46164.2021.9630192 (cit. on p. 18).

[32] A. Burrello, F. Carlucci, G. Pollo, X. Wang, M. Poncino, E. Macii, L. Benini, and D. Pagliari. «Optimization and Deployment of Deep Neural Networks for PPG-based Blood Pressure Estimation Targeting Low-power Wearables». In: *IEEE Internet of Things Journal* (2024). DOI: 10.1109/JIOT.2024.3453834 (cit. on pp. 19, 47, 53).

[33] *Neural Network Intelligence (NNI)*. https://nni.readthedocs.io (cit. on p. 34).

[34] V. Fra. «Application-oriented automatic hyperparameter optimization for spiking neural network prototyping». In: *arXiv preprint arXiv:2502.12172* (2025) (cit. on p. 34).

[35] J. Yik et al. «The neurobench framework for benchmarking neuromorphic computing algorithms and systems». In: *Nature communications* 16.1 (2025), p. 1545 (cit. on p. 35).

[36] Institute of Neuroinformatics. *Dynapse Chip repository*. https://gitlab.com/neuroinf/monotonic_nsm (cit. on p. 38).