Neuromorphic Signal Processing for Wearable Devices

Yuqi Ding

Submitted in fulfilment of the requirements for the
Degree of Doctor of Philosophy

James Watt School of Engineering
College of Science and Engineering
University of Glasgow

June 2025

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Yuqi Ding
June 2025

Abstract

Wearable health devices have a strong demand in real-time biomedical signal processing. Over the past few decades, advances in Artificial Intelligence (AI), and particularly the development of neural networks, have significantly impacted the field of signal processing, enabling more efficient and accurate analysis of complex biomedical signals. However, continuous monitoring could generate massive amounts of data, resulting in an information bottleneck that challenges data transfer and subsequent post-processing. Neuromorphic computing, an emerging brain-inspired computational architecture, has garnered significant research attention in recent years. In contrast to the conventional von Neumann architecture, which relies on a separation between memory and processing units, neuromorphic systems integrate computation and data storage within a unified physical framework. This design emulates the synaptic dynamics observed in biological neural networks. Such methods can process data proximate to the sensor with reduced power consumption, latency and bandwidth, providing new solutions to signal processing for wearable devices. Among neuromorphic systems, Physical Reservoir Computing (PRC) has emerged as a compelling solution. PRC harnesses the intrinsic dynamics of physical systems to accelerate and reduce the energy consumption of machine learning computations.

This thesis focuses on modelling and implementing PRC frameworks in signal processing applications. In the initial stage, the potential of PRC as a predictor for biomedical applications was explored. The model successfully maps the Magnetomyography (MMG) signal to Electromyography (EMG) with an acceptable normalized root mean square error (NRMSE) of 0.3894. In addition, an average NRMSE of 0.3690 was obtained for predicting Electrocardiography (ECG) to Phonocardiography (PCG).

However, practical applications of signal processing present more complex challenges. While the neuromorphic signal processing underperforms compared to state-of-the-art Deep Learning (DL) algorithms, and relatively few studies have investigated its application in classification tasks, the research is extended to examine the use of PRC in a complex biometric identification task. An overall classification accuracy of 89.03% in identifying twelve testing subjects was achieved during the intermediate stage.

Nevertheless, a significant concern emerged with respect to maintaining precision when im-

plementing high-resolution signals via electronic circuits in PRC-based systems, particularly due to limitations in electronic components. This challenge motivated the research to the next stage that explores a hybrid approach of combining PRC and Spiking Neural Network (SNN), which allows for the conversion of signals from the high-precision analogue domain into a low-precision discrete spiking domain, thereby reducing hardware and storage costs. Consequently, an event-driven PRC framework was proposed and validated in the final stages to address this concern. An average classification accuracy of 80.3% was obtained in classifying 50 gestures which outperforms current SNN-based methods. The results pipeline a new insight into processing real-time signals at the edge for wearable devices, promising compact and ultra-low power electronic systems for temporal signal processing in wearable devices.

Acknowledgements

Completing this PhD journey has been one of the most challenging yet rewarding experiences of my life. This work would not have been possible without the support, guidance, and encouragement of many individuals and institutions.

First and foremost, I would like to express my deepest gratitude to my supervisor, Prof. Hadi Heidari, for offering an opportunity to pursue my PhD as a member of Microelectronics Lab (meLAB). His patience, support, and encouragement to think critically and independently have been a source of motivation, especially during the most challenging moments of my PhD journey. I sincerely appreciate his help, not only in guiding me through my research but also in shaping my approach to scientific inquiry and professional development. I would also like to express my deep appreciation to Dr. Xiangpeng Liang and Dr. Haobo Li, for their invaluable guidance through my research journey. Their expertise and mentorship have greatly benefited me in establishing the knowledge I need in the field of reservoir computing and biomedical signal processing, respectively.

I would also like to express my sincere gratitude to the Institute of Neuroinformatics (INI) for generously offering me a two-month visiting opportunity at the University of Zurich and ETH Zurich. A special thanks to Dr. Elisa Donati for hosting and supervising me during my visit at INI. Her deep understanding of electromyography (EMG) and spiking neural networks (SNNs) has had a profound impact on my research. The access to resources, expert insights, and professional support provided by INI and its members was invaluable in advancing my work.

In addition, I would like to acknowledge the MEG-Center Tübingen at the University of Tübingen and the Extreme Light Group at the University of Glasgow for generously providing the datasets used in this thesis.

I wish to thank all the meLAB members with special appreciation for those who helped in my PhD research: Dr. Bhavani Yalagala, Dr. Hannah Thompson, Dr. Jungang (Judy) Zhang, Dr. Mohammed Waqas Mughal, Dr. Siming Zuo, Antonia Pavlidou, Changhao Ge, Huxi Wang, and Yuanxi Cheng. A special thanks to Jiaoran Wang, with whom I shared daily lunches and countless conversations. Also, her guidance in scientific figure plotting has been invaluable to

me. I would also like to express my heartfelt gratitude to my roommate in Glasgow, Xinmiao Liu, for her support and encouragement during difficult moments, as well as for generously sharing her delicious homemade pastries and sweets.

Lastly, I would like to express my deepest appreciation for the unconditional love, endless support of my family throughout my life. My deepest gratitude goes to my father Guoqiang Ding; my mother Guozhen Zhang; cousins Yuhan Ding, Weihao Zhang and Yuchan Zhang; and all my other family members, whose encouragement has given me the strength and courage to face challenges. I would also like to thank my beloved dog, Buding. Your cute and fluffy presence brought me comfort during difficult times and accompanied me through my journey from junior school to university. Though it is a pity that you could not witness my doctoral graduation, your sincerity and enthusiasm have always made me optimistic, and for that, I will always be grateful.

Abbreviations

- Acc - Accuracy
- ADC - Analog-to-Digital Converter
- AI - Artificial Intelligence
- ALR - Auto-Labelling-Refining
- ALU - Arithmetic Logic Unit
- ANN - Artificial Neural Network
- ASIC - Application-Specific Integrated Circuit
- Bi-LSTM - Bidirectional Long Short-Term Memory
- BCI - Brain-Computer Interface
- BioPatRec - Biological Pattern Recognition
- BPTT - Backpropagation-Through-Time
- C2C - Cycle-to-Cycle
- CE - Cross-Entropy
- CMOS - Complementary Metal-Oxide-Semiconductor
- CNN - Convolutional Neural Network
- CVD - Cardiovascular Disease
- D2D - Device-to-Device
- DAC - Digital-to-Analog Converter
- DL - Deep Learning
- DLR - Delay Line Reservoir

- DNN - Deep Neural Network
- DOE - Diffractive Optical Element
- ECG - Electrocardiography
- EEG - Electroencephalography
- EMG - Electromyography
- eRNR - electronic Rotating Neuron Reservoir
- ESN - Echo State Network
- FN - False Negative
- FP - False Positive
- FPGA - Field-Programmable Gate Array
- GUI - Graphical User Interface
- GPU - Graphics Processing Unit
- HIST - Histogram
- HYSER - High-densitY Surface Electromyogram Recordings
- IoT - Internet of Things
- k-NN - k-Nearest Neighbours
- LDA - Linear Discriminant Analysis
- LIF - Leaky Integrate-and-Fire
- LSM - Liquid State Machine
- LSTM - Long Short-Term Memory
- MC - Memory Capacity
- MCG - Magnetocardiography
- mDWT - marginal Discrete Wavelet Transform
- MMG - Magnetomyography
- NinaPro - Non-Invasive Adaptive Prosthetics
- NRMSE - Normalized Root Mean Square Error

- NVM - Non-Volatile Memory
- ODE - Ordinary Differential Equation
- PCA - Principal Component Analysis
- PCB - Printed Circuit Board
- PCG - Phonocardiography
- PP - Positive Predictivity
- PPG - Photoplethysmography
- PRC - Physical Reservoir Computing
- RBF - Radial Basis Function
- RC - Reservoir Computing
- ReLU - Rectified Linear Unit
- RF - Random Forest
- RMS - Root Mean Square
- RNN - Recurrent Neural Network
- RNR - Rotating Neuron Reservoir
- ROI - Region of Interest
- SCR - Simple Cycle Reservoir
- Se - Sensitivity
- sEMG - surface Electromyography
- sFCN - spiking Fully Connected Layer
- SNN - Spiking Neural Network
- SNR - Signal-to-Noise Ratio
- SOA - Semiconductor Optical Amplifier
- Sp - Specification
- sRNR - spiking Rotating Neuron Reservoir
- STDP - Spike-Timing-Dependent Plasticity

- STM - Short Term Memory
- SVM - Support Vector Machine
- TD - Time-Domain
- TN - Ture Negative
- TP - True Positive
- t-SNE - t-Distributed Stochastic Neighbor Embedding
- VLSI - Very Large Scale Integration
- VMM - Vector–Matrix Multiplication
- WTA - Winner-Take-All

Symbols

- α - Learning rate
- β - Regularization parameter
- γ - Scaling parameter in RNR
- τ - Time constant for the dynamic neuron
- τ_r - Rate of rotation in
- σ - Standard deviation
- b - Vectors of biases
- f - Activation function
- $f_{sampling}$ - Sampling frequency
- Δw - Change in the synaptic weight
- Δt - time difference between the postsynaptic and presynaptic spikes
- A^+ and A^- - Scaling constants for potentiation and depression
- C - Capacitance
- M - Number of parallel reservoirs
- N - Network size
- I - Identity matrix
- R - Resistance
- W - Weight matrix for the reservoir layer
- $s(n)$ - State vector at time step n
- $u(n)$ - Input at time n

- $y(n)$ - Output at time n
- $\hat{y}$ - Predicted output
- I_{in} - Injected current
- S_{out} - Output spike trains
- V_{mem} - Membrane potential
- V_{thr} - Threshold voltage
- V_{reset} - Reset voltage
- V_{rest} - baseline Membrane potential when the neuron is inactive
- W_{in} - Weight matrix for the input layer
- W_{out} - Weight matrix for the output layer
- $Spk(t)$ - Input spike trains
- $Spk_M(t)$ - Masked input spike trains
- $P(y = i)$ - Output probability for the i-th class
- cov - Covariance

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter introduces the neuromorphic implementation for processing signals at the edge of wearable devices, beginning with a general overview of edge computing for wearable devices. The applications of neuromorphic algorithms and architectures for processing biomedical signals from wearable sensors and devices are stated as motivations for this study. In addition, the aims and objectives are put forward, followed by the contributions to the field of neuromorphic signal processing.

1.1 Process signals at the edge

In the context of signal processing for wearable sensors, edge computing plays an important role in enabling real-time, efficient data analysis. It allows for processing data locally on the wearable device or nearby, rather than sending data to a centralized server for subsequent post-processing, thereby saving computational overhead and decreasing bandwidth usage. Moreover, local data processing mitigates the risk of potential privacy breaches, which is particularly critical in applications involving sensitive patient information [1]. Traditional machine learning approaches, such as, support vector machine (SVM), k-nearest neighbours (k-NN), linear discriminant analysis (LDA), and random forest (RF) require feature extractions from both the time domain and frequency domain. In contrast, deep learning (DL) models have demonstrated superior performance by automatically learning features. Specialized hardware for accelerating computation has been introduced and extensively utilized in recent years, especially with the advent of graphics processing units (GPUs), which has driven DL algorithms to the predominant techniques in machine learning. However, this comes at the cost of increased power consumption and memory requirements. While GPUs typically operate at around 200 Watts for training deep neural networks (DNNs) [2], their integration into lightweight and portable wearable devices is highly impractical. Ensuring efficient processing with minimal computa-

tional resources in edge devices remains a critical challenge, necessitating innovations in both software and hardware design.

First, from the software aspect, it is essential to apply learning algorithms implementable on chips with low training costs to ensure efficient operations within resource-constrained environments. Backpropagation is the core algorithm to train DNNs, enabling them to learn from data by adjusting internal weights to minimize the error and using the chain rule to propagate the error backwards [3]. However, backpropagation could be computationally expensive to converge to a local minimum. Even worse, for large datasets or complex models, gradients can become extremely small (vanishing) or large (exploding), causing slow learning or numerical instability during the training phase. But, could the backpropagation be avoided?

Second, from the hardware aspect, it is meaningful to reference computing systems that mimic the architecture of information processing in the brain. Although the Von Neumann computer architecture has made significant contributions to science and technology for decades, its performance is inherently limited by inefficiencies arising from the separation of the storage and processing units. This separation results in relatively slow and energy-intensive data movement, leading to suboptimal computational efficiency [4]. But, is it possible to realize simultaneous storage and computation in signal processing by comprising dedicated electronic circuits to implement neuron and synapse circuits?

1.2 Neuromorphic computing

Inspired by how the brain functions, a fundamentally different way to construct information processing was originally proposed by Carver Mead in 1990s, known as neuromorphic or brain-inspired computing [5]. It refers to mixed signal very large scale integration (VLSI) computing systems that take inspiration from the intrinsic built-in physical capabilities of the human brain to learn and deal with complex data.

Neuromorphic systems are characterized by their highly interconnected and parallel structures, low-power consumption, and the integration of memory and processing units. While the design is inherently compelling, the importance has grown due to several emerging challenges: the expected end of Moore's law, the rising power demands linked to the breakdown of Dennard scaling, and the limitations of low data transfer rates between processing units and memory units, commonly known as the von Neumann bottleneck [6, 7]. The neuromorphic approach prioritizes low latency and energy efficiency, making it a promising solution for processing temporal signals at the edge.

Neuromorphic computing, as a broad concept, encompasses multiple domains, including algorithms, circuits, materials and devices. This thesis mainly focused on the neuromorphic

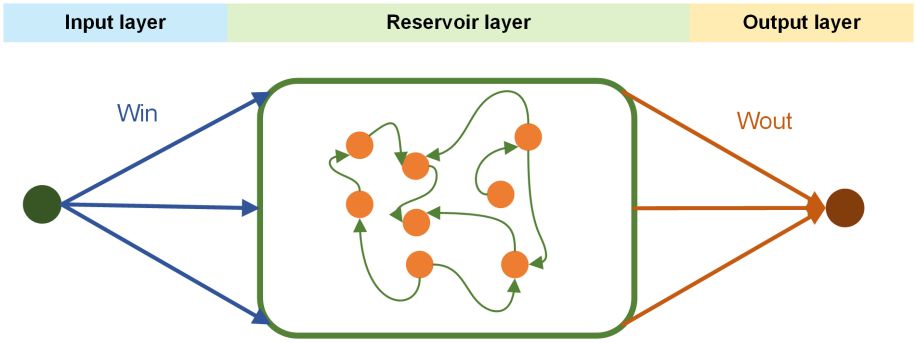

Figure 1.1: A classical RC network.

software-hardware co-design and the applications in biomedical signal processing tasks.

1.2.1 Reservoir computing

The reservoir computing (RC) is a feedforward computational framework derived from recurrent neural networks (RNNs) that leverages the high-dimensional dynamic behaviours of complex systems for efficient information processing. It was proposed in 2000s by incorporating the concepts of echo state network (ESN) and liquid state machine (LSM) [8, 9, 10]. Within the range of RNNs, it is particularly well-suited for tasks involving temporal data and systems, as the reservoir's recurrent connections naturally capture and transform these dynamics into a higher-dimensional feature space to be linearly separable. The key feature of reservoir computing is that only the weights of the output layer are trained, while the internal reservoir's connections remain static, allowing for fewer parameters monitoring and lower training costs compared to traditional RNNs that require backpropagation-through-time (BPTT).

A classical RC network is shown in Fig. 1.1, it typically consists of three key elements: an input layer, a fixed "reservoir" layer of recurrently connected neurons, and a readout layer. Given an input $u(n)$, an output $y(n)$, and a reservoir size of N (N neurons), a conventional routine of an RC network is introduced by the following:

- *Input layer:* The input layer incorporates an input masking procedure that interfaces the inputs with the reservoir layer by an input mask, denoted by W_{in}.
- *Reservoir layer:* The reservoir layer delivers transient responses for the inputs, denoted by $s(n)$, which are also known as the states/dynamics of the reservoir. In this case, the inputs are mapped to the high-dimensional feature space for linear separability with recurrent connections.
- *Readout layer:* While the two layers mentioned above remain fixed, only the readout layer requires training.

1.2.2 Physical reservoir computing

The concept of physical reservoir computing (PRC) was proposed to exploit the complex dynamics of physical systems, being capable of processing information for edge devices in a decentralized manner to reduce adaption delay caused by data transmission [11, 12, 13]. In the classical reservoir topology, the weights and connections among neurons are, although fixed, both randomly generated as introduced in Section 1.2.1, making it hard to implement by hardware components directly. Nevertheless, a simplified cyclic topology was proposed to minimize the complexity of the reservoir without degrading the performance [14]. According to the structure shown in Fig. 1.2(a), the units are organized in a ring topology, and nonzero elements in reservoir weight matrix W are on the $W_{i+1,i} = 1$ and $W_{1,N} = 1$, described by the matrix in equation (1.1):

$$\begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 1 \\ 1 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \ddots & 1 & 0 & 0 \\ 0 & 0 & \ddots & 0 & 1 & 0 \end{bmatrix} \tag{1.1}$$

This simplified topology was later realized by electronic circuits using rotating elements (analogue multiplexers) and known as rotating neuron reservoir (RNR) [15]. The connections of the input-to-reservoir layer and reservoir-to-output layer are circularly shifted at each time step, as demonstrated in Fig. 1.2(b). The circular shifts of connections among layers and neurons are realized by analogue multiplexers while the dynamic neurons are realized by Leaky-integrate rectified linear unit (ReLU) circuit, as represented in Fig. 1.2. The weights of the input mask in this case are randomly selected from {-1,1}, denoted by negative/positive signal sources.

The RNR framework was proposed and verified by several simple benchmark datasets in [15], however, the potential of its ability was not fully exploited in complex tasks. Therefore, during the initial stages of my research, I applied it to two prediction tasks and one complex and high-resolution classification task.

While results demonstrated the feasibility of RNR in processing practical biomedical signals, significant problems occur in terms of solving classification tasks using RC-related networks. Typically, a prevalent way for classification tasks relies on a linear model trained by a convex optimization technique like ridge regression, and using a winner-take-all (WTA) algorithm to determine the corresponding target class that exceeds a certain threshold [16, 17, 18]. This method is effective when the number of classes to be classified is fewer than ten, and brings a

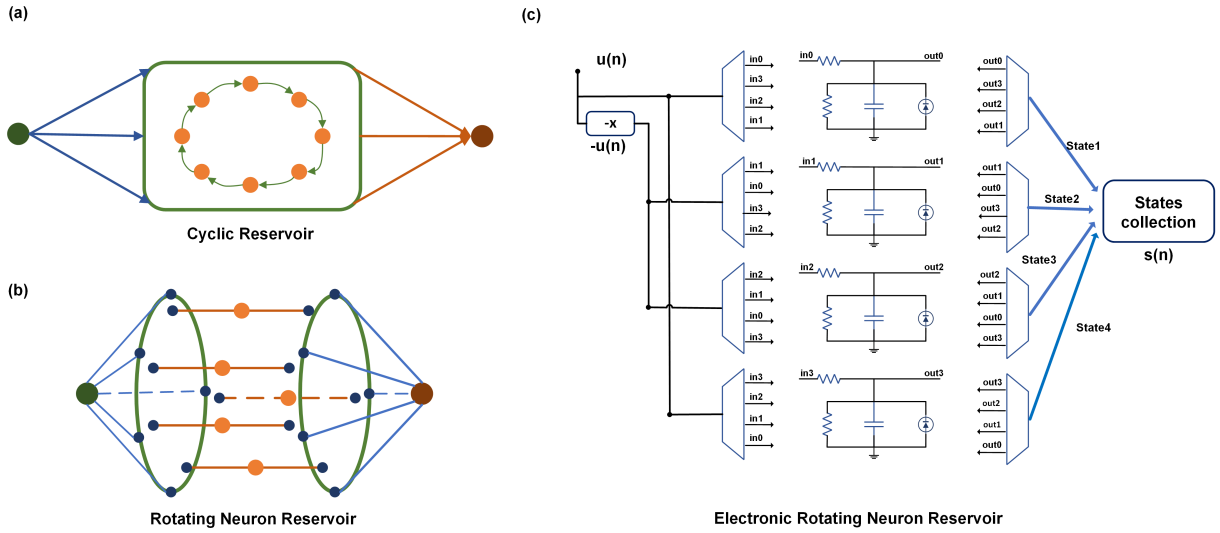

Figure 1.2: The evolved RNR topology. (a) A simplified cyclic reservoir topology. (b) A 3D representation of an RNR topology. (c) The electronic implementation of a 4-neuron RNR topology.

larger network size which is costly in hardware. With the increase of target classes, nonlinear classifiers could demonstrate superior performance and decrease the network size. However, the high-dimension and high-precision reservoir states prevent the use of standard nonlinear classifiers being applied on the readout layer while some dimension reduction procedures on the reservoir states were proposed to apply classifiers on the readout layer [19]. Another possible solution for solving this challenge could be implementing the reservoir in the low-precision spiking domain, which refers to the next phase of my research.

1.2.3 Spiking neural network

The major difference between an spiking neural network (SNN) and an artificial neural network (ANN) lies in the nonlinear neuron. While ANNs usually receive continuous analogue information and passes the information through sigmoid or ReLU neurons for nonlinear activation, SNNs applies biologically plausible spiking neurons to receive spike-based information, which drives the inner membrane potential and generates outputs also in terms of spikes. Fig. 1.3 provides insights into the circuit for the two dynamic neuron models.

In this research, for the first time, we proposed the SNN-based RNR framework. This incorporation offers three key benefits: (i) A spike is a single-bit event, either a '1' or a '0', which is more hardware-friendly than floating-point values. In comparison, the electronic rotating neuron reservoir (eRNR) where signals are processed in the analogue domain. However, floating point values' precision in analogue circuits is costly in terms of complexity and power consumption [20], hence, processing information in the form of low-precision spike trains could be a possible solution. (ii) Event-driven processing allows for energy efficiency and low latency as

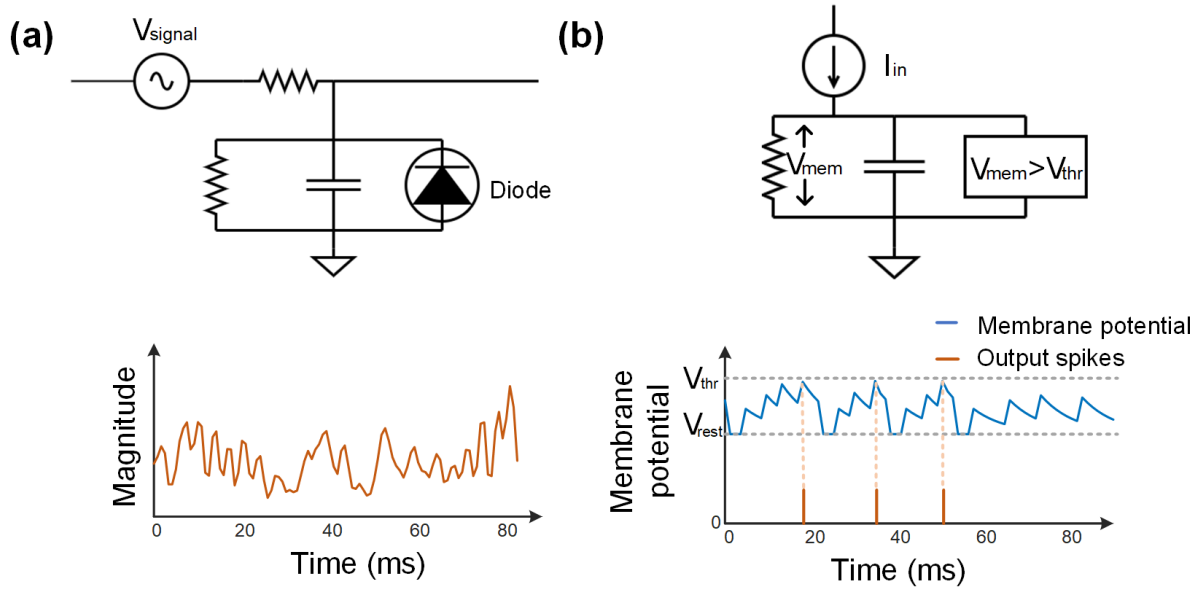

Figure 1.3: The description of dynamic neurons. (a) The nonlinear integration-ReLU-Leakage neuron model and the generated dynamics in terms of continuous output states. (b) The leaky integrate-and-fire neuron model. The neuron will fire a spike when the membrane potential V_{mem} exceeds the threshold V_{thr}.

the neurons only respond when an event occurs, leading to sparse vectors/tensors that are cheap to store and low-power to move [21]. (iii) This fusion still retains the advantages of RNR's simple and hardware-friendly structure, with only the readout layer requiring training. At the same time, the low-precision reservoir states enable the use of standard classifiers for complex classification tasks to further improve classification performance.

1.3 Research summary

The primary objective of this thesis is to implement neuromorphic algorithms and frameworks in biomedical signal processing for wearable devices, facilitating both prediction and classification tasks. Notable advancements have been achieved in developing hardware-compatible algorithms in a more computationally efficient form while maintaining performance levels comparable to DL-based strategies.

Consequently, to facilitate the development of hardware-compatible algorithms, this study focuses on three primary research objectives:

- The RNR network for biomedical prediction tasks — This initial phase serves to validate the network's efficacy while providing foundational insights for parameter optimization.
- The RNR network using linear regression for complex biomedical classification tasks – In this intermediate research phase, the RNR network's performance on classification tasks

was evaluated in the analogue domain. Key challenges concerning network scalability and computational efficiency were identified, which subsequently guided the research direction toward further development.

- Integration of RNR and SNN for complex classification Tasks – In this final phase, the network architecture was adapted to the spiking neural domain, achieving a substantial reduction in network size while maintaining high classification accuracy and computational efficiency.

An outline of the thesis is presented below:

Chapter 2 presents a comprehensive literature review of PRC in biomedical applications. Specifically, it examines various implementation paradigms of PRC systems, categorizing them based on different approaches and their applications in biomedical signal processing. In addition to analyzing reservoir implementation platforms, this chapter reviews training methodologies for diverse biomedical signal processing tasks, as these methods significantly influence the performance of neuromorphic systems.

Chapter 3 presents two prediction tasks utilizing the RNR architecture, accompanied by a comprehensive investigation into parameter tuning and validation. The results demonstrate that the prediction errors achieved using RNR remain minimal and are comparable to those obtained with the software-based cyclic reservoir algorithm. This successful implementation serves as a foundation for the subsequent stage of research, which focuses on applying RNR to a more complex classification task.

Chapter 4 introduces an optical stethoscope-based laser-camera system for biometric identification, utilizing RNR as the processing core. The heart sound signals collected from the optical stethoscope are employed as unique biometric identifiers, with RNR demonstrating exceptional accuracy in distinguishing among 12 subjects. However, the current RNR implementation requires a large network size of 3000 neurons, and the readout classification methods remain suboptimal due to the high dimensionality and resolution of reservoir states. Consequently, the next phase of research focuses on transitioning RNR to the SNN domain to address these challenges.

Chapter 5 introduces a network that integrates a PRC framework, specifically RNR, within a SNN architecture and adopts an event-based encoding scheme to transform electromyography (EMG) signals into spike trains. This approach outperforms existing SNN-based methods in classification accuracy for gesture recognition tasks while maintaining competitiveness with DL models in a more computationally efficient manner. Notably, the network size is significantly reduced to 480 neurons for the classification of 50 gestures. This advancement contributes to the development of next-generation lightweight wearable systems with ultra-low latency and embedded intelligence.

Chapter 6 provides a comprehensive summary of the key contributions presented in this thesis. Additionally, it critically discusses the emerging opportunities and existing challenges related to the practical industrial deployment of PRC systems.

1.4 List of publications

1.4.1 Journal publications

[1] **Y. Ding**, H. Li, X. Liang, M. Vaskeviciute, D. Faccio and H. Heidari, "Physical Reservoir Computing for optical stethoscope-based Heart Sound Biometric Identification", IEEE Transactions on Artificial Intelligence, 2024. **(Under review, major revision)**

[2] **Y. Ding**, E. Donati, H. Li, H. Heidari, "Event-Driven Implementation of a Physical Reservoir Computing Framework for Superficial EMG-based Gesture Recognition", IEEE Transactions on Artificial Intelligence, 2025. **(Under review, major revision)**
arXiv preprint version available: arXiv:2503.13492

[3] **Y. Ding**, B. Yalagala, H. Li, M. Mughal, "Physical Reservoir Computing for Biomedical Applications", Neuromorphic Computing and Engineering, 2025. **(Under review, major revision)**

[4] M. Mughal, B. Yalagala, A. Pavlidou, **Y. Ding**, H. Heidari, "Fully Hardware Readout Layer for The Neuromorphic Reservoir Computing (RC) Using Memristors", IEEE Transactions on Nanotechnology. **(Under review)**

1.4.2 Conference proceedings

[1] **Y. Ding**, X. Liang, T. Middelmann, J. Marquetand, and H. Heidari. (2022) MMG/EMG Mapping with Reservoir Computing. In: 2022 IEEE International Conference on Electronics, Circuits and Systems (ICECS), Glasgow, United Kingdom, 24-26 October 2022. (doi: 10.1109/ICECS202256217.2022.9971109) **(Oral presentation)**

[2] **Y. Ding**, H. Li, X. Liang, M. Vaskeviociute, D. Faccio, H. Heidari. (2024) A Physical Reservoir Computing Processor for ECG-to-PCG Signals Prediction. In: 2024 IEEE International Symposium on Circuits and Systems (ISCAS), Singapore, 19-22 May 2024. (doi: 10.1109/ISCAS58744.2024.10557860) **(Oral presentation)**

[3] J. Wang, J. Zhang, **Y. Ding**, M. Kirimi, N. Mirzai, J. Mercer, H. Heidari. (2025) Intelligent Rapid Antenna Design with Integrated Impedance Matching Network for Wireless Communication System. (2025) IEEE International Symposium on Circuits and Systems (ISCAS), London, United Kingdom, 25-28 May 2025. **(Accepted for publication)**

Chapter 2

Literature Review

A wide range of interdisciplinary research has been undertaken in recent years to fully enhance the capabilities of RC, especially with the advent of PRC. PRC has demonstrated efficacy in applications for biomedical edge devices with advantages in power consumption, latency, bandwidth and privacy. This chapter provides a comprehensive review of PRC implementation paradigms in different categories and their applications in biomedical signal processing, including the training methods.

2.1 Introduction

The neuromorphic approach where memory and processing units are not separated, distinct from traditional Von Neumann architectures, prioritizes low latency and energy efficiency, making it a promising solution for processing temporal signals at the edge. Under the term neuromorphic, RC occurs as a compelling algorithm which is derived from RNNs. As a feedforward computational network, RC leverages the high-dimensional dynamic behaviors of complex systems for efficient information processing. Within the range of RNN, it is particularly well-suited for tasks involving temporal data and systems, as the reservoir's recurrent connections naturally capture and transform these dynamics into a higher dimensional feature space to be linearly separable. The key feature of reservoir computing is that only the weights of the output layer are trained, while the internal reservoir's connections remain static, allowing for fewer parameters monitoring and lower training costs compared to traditional RNNs that require BPTT [22, 23]. Based on the simple structure of RC algorithms, the concept of PRC was proposed to exploit the dynamics of physical systems, being capable of processing information for edge devices in a decentralized manner to reduce power consumption and adaption delay [11, 13]. As an emerging computational paradigm, PRC is compatible with various physical systems, such as electronics, photonics, and mechanical systems. By harnessing the inherent nonlinear charac-

teristics in physical systems, PRC bridges the gap between hardware and algorithmic efficiency, offering a compelling solution to traditional computational methods for tasks requiring real-time processing and adaptability.

The integration of PRC into biomedical signal processing offers innovative frameworks for classification, anomaly detection, and prediction tasks, paving the way for faster, more accurate, and energy-efficient healthcare solutions [24, 25, 26, 27, 28]. Despite the spring-up research of PRC in low-power computing, a thorough summary of the recent state-of-the-art is meaningful to provide experience in selecting and designing compatible solutions for biomedical-related applications in future research. Therefore, a systematic review of the recent advances of PRC and its applications in biomedical signal processing is covered in this chapter. This chapter explores the implementation paradigms of PRC as the hardware design of PRC plays a critical role in the performance effects. Additionally, the primary biomedical signals, where PRC is applied as the processing core are identified. It is also worth mentioning the traditional and emerging training methods for RC systems. Finally, a discussion of the competitiveness of PRC systems and the existing challenges for practical industrial applications is provided.

2.2 Reservoir computing

The core of RC is the reservoir, which serves as a temporal kernel to project the input data to a high-dimensional feature space through the random recurrent connections, and this high-dimensional projection enhances the linear separability of the data. Notably, both the input-to-reservoir layer and the reservoir dynamics remain fixed, while only the reservoir-to-output layer requires training, significantly reducing computational complexity and training requirements.

A classical RC network is shown in Fig. 2.1(a), it typically consists of three key elements: an input layer, a fixed "reservoir" layer of recurrently connected neurons, and a readout layer.

Nevertheless, the randomly weighted and interconnected neurons within the reservoir exhibit significant complexity while the complexity of the reservoir could be minimized by introducing simplified topologies without degrading the performance [14]. The proposed simplified reservoir topologies, such as delay line reservoir (DLR) and simple cycle reservoir (SCR), have demonstrated broad applicability in constructing PRC systems [15, 29, 30, 31].

In this review, RC is classified using various criteria, according to structural frameworks, modeling neurons and physical realizations, as shown in Fig. 2.1(b)-(d).

2.2.1 Modelling neurons

The modelling neuron serves as the fundamental computational unit of a neural network. Variations in the characteristics of modeling neurons lead to distinct types of neural networks. Two

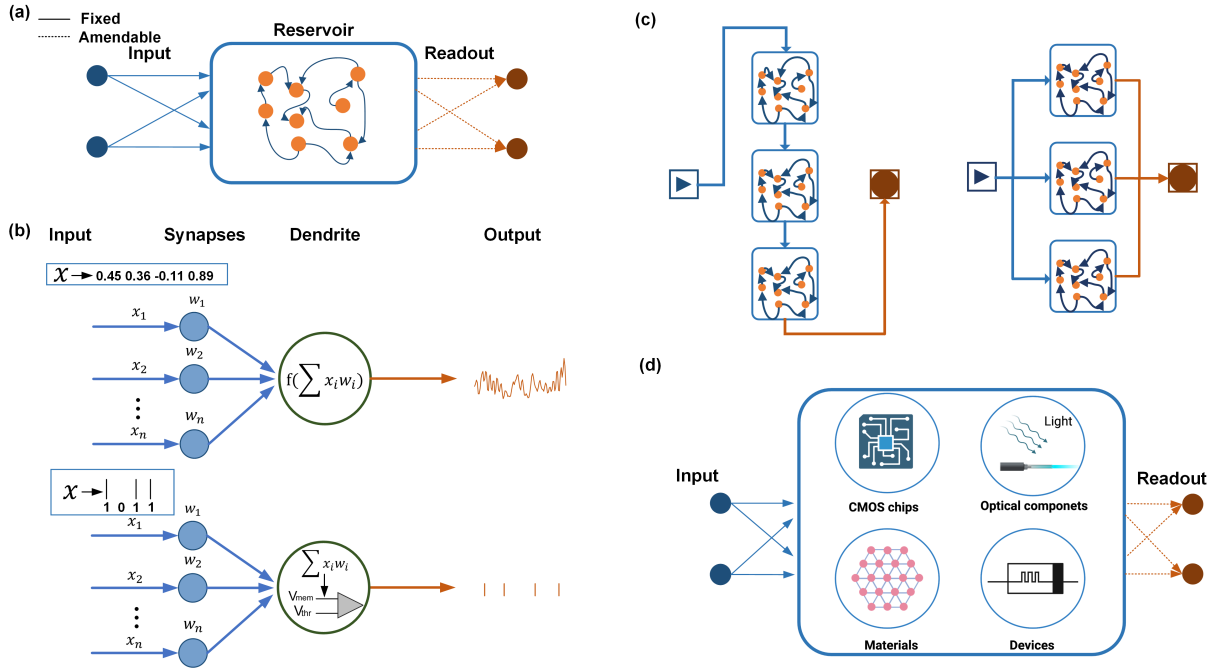

Figure 2.1: (a) The classical reservoir topologies. (b) Two different modelling neurons. Upper: artificial neurons. Lower: spiking neurons. (c) The hierarchical frameworks. Left: deep reservoirs. Right: wide reservoirs. (d) Physical reservoirs.

primary categories of neural networks are ANNs and SNNs, consisting of artificial neurons and spiking neurons, respectively. While the artificial neurons use sigmoid units or ReLUs for continuous nonlinear activation, the spiking neurons are distinct from artificial neurons by presenting and exchanging information in an event-driven manner in the spiking domain for sparse representation beneficial to storage costs and energy consumption [32]. The fundamental neuron model for SNNs mimics the behaviour of biological neurons by transmitting information through discrete electrical pulses called spikes [33]. In contrast, the artificial neuron dominated the ANN scenes for decades [34]. This neuron model processes multiple inputs by computing their weighted sum, incorporating an additional bias term. The resulting value is then passed through an activation function to generate the neuron's output.

The reservoir network proposed in the field of the SNN was configured as LSM [9] while the one proposed in the field of ANN was configured as ESN [35]. Both networks were unified and defined as reservoir computing in the subsequent years [10, 22, 36].

Liquid state machine (LSM)

In LSMs, the simplest spiking neuron model is the leaky integrate-and-fire (LIF) neuron. The dynamics in a LIF neuron can be quantified by an ordinary differential equation (ODE) given in (2.1) which provides a discrete and recurrent representation. The neuron will fire a spike ($S_{out}(t) = 1$) when the membrane potential $V_{mem}(t)$ exceeds the threshold V_{thr} and it is reset to

V_{reset}. Otherwise, the reset term will not be applied ($S_{out}(t) = 0$). This process can be denoted by equations (2.1) and (2.2):

$$\tau \frac{dV_{mem}(t)}{dt} = -V_{mem}(t) + RI_{in} \tag{2.1}$$

$$V(t) = V_{reset} \Leftarrow S_{out}(t) = 1 \tag{2.2}$$

This model coarsely represents a low-pass filter circuit of a resistor R and a capacitor C. In addition, τ is the time constant for the membrane relaxation time, V_{mem} is the membrane potential, and I_{in} is the injected current that drives the membrane potential. The neuron receives spikes as inputs and also generates dynamics in terms of spikes.

Echo state network (ESN)

The ESN adopts discrete-time artificial neurons as the basic units to generate high-dimensional states. Given a k-dimensional input $u(n)$ and a reservoir size of N (N neurons), the update equation for the states is defined in equation (2.3):

$$s(n) = f(W_{in}^{N\times k} u(n) + W^{N\times N} s(n-1) + b) \tag{2.3}$$

The component b stands for the vectors of biases, and the function f represents the activation function, among which the sigmoid units and ReLU are basic neuron activation units.

2.2.2 Structural frameworks

In addition to reservoir topology designs, the structural designs involving stacked deep reservoirs and parallel reservoirs are also worth mentioning. The structural designs of RC systems could be roughly categorized into deep reservoirs and wide reservoirs [37, 38]. The reservoirs are configured either in a cascaded series arrangement or in parallel to enhance performance on complex tasks compared to traditional, single-reservoir models. While some physical realizations of RC systems have adopted the wide reservoir structure, the deep reservoir structure is still in the software implementation level. In this section, a brief review of the deep reservoir and wide reservoir frameworks is covered.

Deep reservoir framework

The deep reservoir framework was innovated from deep learning topologies. The hidden layer is represented by stacked sub-reservoir components and the outputs of each reservoir are fed into the input for the next reservoir [39], as depicted in Fig. 2.1(b). Although the shallow reservoir

already provides a rich pool of dynamics, the deep stacked form makes it possible to achieve temporal data representations across multiple layers, thereby enhancing the network's capacity for processing complex temporal patterns [40]. This idea was extended to deep fuzzy RC using reinforce learning to strengthen the feature extraction ability of the reservoir [41]. Strategies such as adding additional layers between reservoirs to achieve better short-term memory property and state richness also gained attention in the following years [42, 43].

Wide reservoir framework

In contrast, the wide reservoir framework processes inputs simultaneously by parallel connected sub-reservoirs. This architecture has been particularly effective in enabling applications within hardware implementations. When using optical amplifiers to implement the RC system, the parallel structure offers a faster and power-efficient mode in terms of photonics integration [44]. Similarly, volatile memristors-formed RC systems also adopt the parallel framework, while each device serves as a processing core to receive the input. Consequently, the device-to-device (D2D) variability of volatile devices could increase the state richness and thus improve the performance of RC systems [45, 46]. RC performed by electronic circuits also favors parallel configurations due to their compatibility with more straightforward integration strategies [15].

2.2.3 Physical reservoir computing (PRC)

RC can be analogized to the physical process of dropping a stone into a still body of water, where the impact of the stone generates ripples that propagate outward. The key idea is that the characteristics of the stone, such as its weight and velocity, can be inferred by observing the resulting ripples in the water. In this analogy, the input corresponds to the initial disturbance, for example, the act of dropping the stone, while the dynamics represent the subsequent ripples that evolve over time. The term "reservoir" is used to describe this system, as it reflects a dynamic environment that captures and processes the input. Based on the concept, one of the earliest physical realizations of the reservoir systems was implemented by electric motors to generate ripples and a bucket of water to reflect ripple patterns [47].

However, such a system lacks portability and precise controllability, limiting its practical implementation. While Lepri *et al.* proposed that a time-delayed dynamical system can generate high-dimensional patterns [48, 49], Appeltant *et al.* took inspiration from the delayed system and applied it to construct an RC framework by using a single nonlinear node with delay line [29]. Incorporating delay lines to create a delay-coupled reservoir significantly reduces the number of nonlinear neurons to one. This simplification enables efficient hardware implementation, supporting high-speed and low-power computing.

In more recent years, various dynamical models and physical systems have been explored as

Table 2.1: Comparison of the power for PRC implementation platforms.

Implementation platform	# of neurons	Network type	Computation speed (Hz)	Power	Reference
Analog circuits	64	ANN	10^7	4.7 mW	[15]
65 nm COMS IC	10	ANN	4×10^5	$\approx 4.4mW$	[55]
FPGA	48	ANN	10^6	1.5W	[56]
DynapSE	192	SNN	-	0.05mW	[57]
Loihi	1250	SNN	-	$\leq 0.45W$	[58]
SpiNNaker	1471	SNN	-	1-4 W (1W per chip)	[59]
Memristors system	24	ANN	-	22 μW	[18]
Photonics	388	ANN	1.3×10^7	150 W	[60]

potential reservoirs [11, 13, 50, 51]. The system was constructed by different combinations of reservoir topologies, structural frameworks, modelling neurons and physical substrates. In the following section, we review the implementation paradigms for PRC systematically.

2.3 Implementation paradigms

The implementation of PRC spans a wide range of platforms, each leveraging different physical properties to create efficient reservoirs. Under the term neuromorphic, PRC also depend on specialized hardware to fully realize its potential, just like deep learning with GPUs [52, 53, 54]. In this review, these platforms are broadly categorized into three key areas: complementary metal-oxide-semiconductor (CMOS)-based implementations, emerging devices, and photonic systems. A comparison of representative implementation platforms chosen from the three areas is provided in Table 2.1. A relatively fair comparison is attempted by considering the number of neurons, the network type and the computation speed (if provided) to evaluate the power of each platform. Note that some literature delivers the power consumption in Joules, they are transformed into Watts according to the information provided in the article for intuitive comparisons.

2.3.1 Complementary metal-oxide-semiconductor (CMOS) technology

The CMOS technology is a widely used semiconductor fabrication process that forms the basis of modern digital and analog circuits. It remains the most dominant and fundamental computation platform for neural networks by offering scalability and compatibility in VLSI circuits. In this section, we mainly review the reservoirs implemented at a chip level, achieved by application-specific integrated circuits (ASICs), field-programmable gate arrays (FPGAs) and dedicated neuromorphic chips.

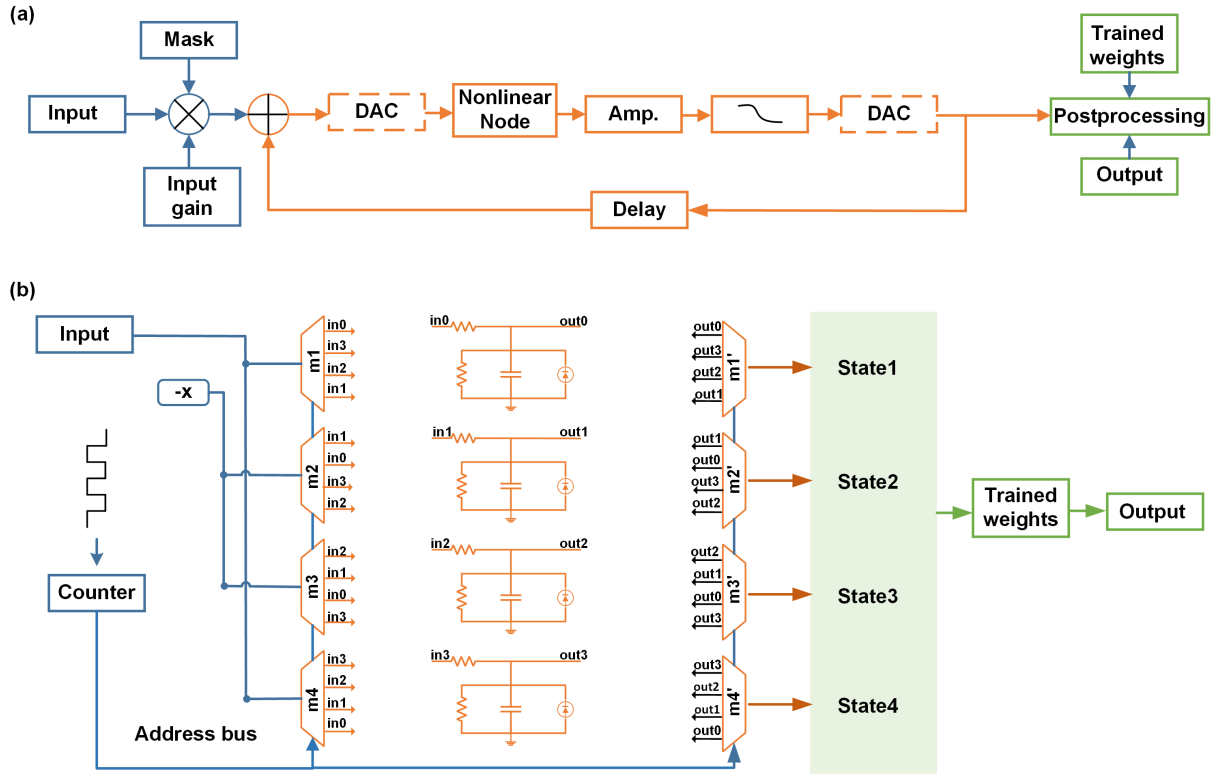

Figure 2.2: (a) The fundamental structure for a delay-based reservoir, adapted from [29] (CC BY-NC-SA 3.0). The nonlinear node is implemented through a Mackey-Glass type nonlinear node as in reference [61]. The delay loop is implemented by ADCs and DACs. (b) The fundamental structure of a cyclic rotating reservoir, adapted from [15] (CC BY 4.0). The input-to-reservoir and reservoir-to-output connections are cyclically rotated at each time step controlled by analog multiplexers and bit counters.

ASICs

The ASIC is a custom-designed chip built for a specific application. Once manufactured, post-manufacturing modifications are impossible. As mentioned in Section 2.2, the complexity of the reservoir can be minimized to adapt to hardware implementation by introducing the delay node [29, 62]. A structure of the framework and a fundamental neuron circuit for the delay node are shown in Fig. 2.2(a). Several investigations of ASIC implemented delay-node reservoirs were conducted in the following years, either incorporating asynchronous pulses [63] or spikes [64, 65]. In addition, a 65-nm based CMOS IC that performs computations of the reservoir layer in the mixed signal domain was proposed also using the delay-based architecture [55]. A more recently proposed cyclic reservoir node based on rotating elements is also worth mentioning; it evolved from the SCR reservoir topology mentioned in Section 2.2 [46]. A schematic of building blocks for the cyclic reservoir is demonstrated in Fig. 2.2(b). However, currently, the analog circuits are implemented at a printed circuit board (PCB) level with the readout layer implemented by a memristor crossbar array. The integration of this architecture into an ASIC chip level is promising for future studies as the cyclic architecture could reduce hardware costs

compared with the delay node architecture, attributed to the parallel realizations and lack of memory units and analog-to-digital converters (ADCs) units.

FPGAs

In contrast, the FPGA is a reconfigurable chip that can be reprogrammed multiple times after manufacturing. It has gained significant attention as a hardware acceleration platform for neural networks by offering reconfigurability [66, 67]. In the context of RC, various FPGA implementations have been explored for reservoir construction. An FPGA processor allows for real-time signal processing and online training with signal recorded from the board and transferred to the computer via USB [68]. Similar to the situation explained in Section 2.3.1, with the simplified delay node design, the network implementation on FPGA boards can be optimized with reduced resource utilization [69, 70, 71, 72]. Furthermore, the simplified cyclic reservoir was also investigated through FPGA implementation to save circuit area and power [56]. In addition to the reservoir node design, efforts have been made to execute the readout learning layer with weight quantization techniques for memory storage efficiency recently [73].

Neuromorphic chips

Apart from ASICs and FPGAs, significant advancements have been achieved in the development of dedicated neuromorphic chips. These platforms are more compatible and flexible with neuromorphic-based algorithms, especially SNNs. The SpiNNaker is a massively parallel computer system for simulating spiking neurons by using ARM cores. Both the architecture and chip design were developed by the University of Manchester in 2014 [74]. The system does not support online training, the readout layer of RC requires pre-training offline and then the extracted weights are used as synaptic weights in SpiNNaker chip [59, 75, 76]. The TrueNorth, designed by IBM in 2015, is a milestone of large-scale digital neuromorphic chips [77]. It also pre-trains RC networks offline and then deploys the weights on the chip for inference only. Both the SpiNNaker and TrueNorth rely entirely on digital circuits, making them suitable for scalability and computational flexibility, though less biologically plausible in certain behaviors compared to mixed-signal systems. The Loihi, designed by Intel in 2017, processes information asynchronously and provides programmable synaptic on-chip learning rules [78]. Various RC-solved tasks were then implemented on Loihi in the following years [58, 79, 80]. In the same year, the Institute of Neuroinformatics proposed DYNAP [81]. It is a mixed-signal processor that developed to combine the benefits of analog for neuron membrane potentials and synaptic weights, and digital for control and communication. This configuration offers improved power efficiency and flexibility for custom-designed neural networks, making it an excellent platform for RC systems [24, 82]. The subsequent version, DYNAP-SE2, was released in 2024 with improvements in interfacing with natural signals in closed-loop applications [83].

2.3.2 Emerging devices

In recent years, the continuous scaling of transistors, denoted as Moore's law, has slowed down as silicon-based transistors are approaching their physical limit [6]. The emerging devices are being explored as potential alternatives for 'beyond CMOS' technologies [54]. Recently, various emerging devices based on diverse technological approaches have been proposed for physical reservoir computing [12]. Among all, efficient memory devices are critical for the hardware implementation of physical reservoir computing, with both volatile devices for dynamic properties and non-volatile devices for mimicking the synaptic behaviours.

Memristors are the devices that usually exhibit the pinched hysteresis characteristics between the applied voltage and current. The change in the resistance of the device is proportional to the history of the current flows through the device and it remembers the amount of charge that flows through it, thus making them ideal candidates for non-volatile memory applications. Furthermore, there are numerous advantages of memristor devices such as the nanoscale device dimensions, low-power consumption, higher switching speeds, CMOS compatibility, scalability, analog behaviours. A pioneering study of applying nanoscale memristor devices as synapses in neuromorphic systems was conducted and demonstrated the feasibility of the non-volatile characteristics of memristors [84]. In addition to the non-volatile memory characteristic, the volatile memory property of dynamic memristors was also explored to serve as the dynamic neuron to project features from the temporal inputs to a high-dimensional feature space [85, 86]. In more recent years, the fully analogue RC system was studies based on both volatile memristors for dynamic neurons and non-volatile memristors for artificial synapses [18].

2.3.3 Photonics

Other than electrons, photons can also serve as carriers of information, providing faster, and higher bandwidth solutions for RC systems. In addition, photonic nodes can exhibit more complex internal dynamics compared to sigmoidal nodes used in software, potentially enhancing processing capabilities. The concept of photonic reservoir computing was proposed in 2008 by employing semiconductor optical amplifiers (SOAs) as the basic building blocks [87], configured as a spatially distributed RC. Later in 2011, the introduction of delay-based systems significantly reduced hardware costs. Subsequent research in this area further accelerated advancements in photonic RC in the following years [88, 89]. In this section, we provide a brief review of photonic RC based on the two categories mentioned above.

Spatially distributed design

In a spatially distributed RC system, the reservoir consists of an optical node array made by SOAs to boost optical signals. The node array is organized in a 4×4 matrix, with each node

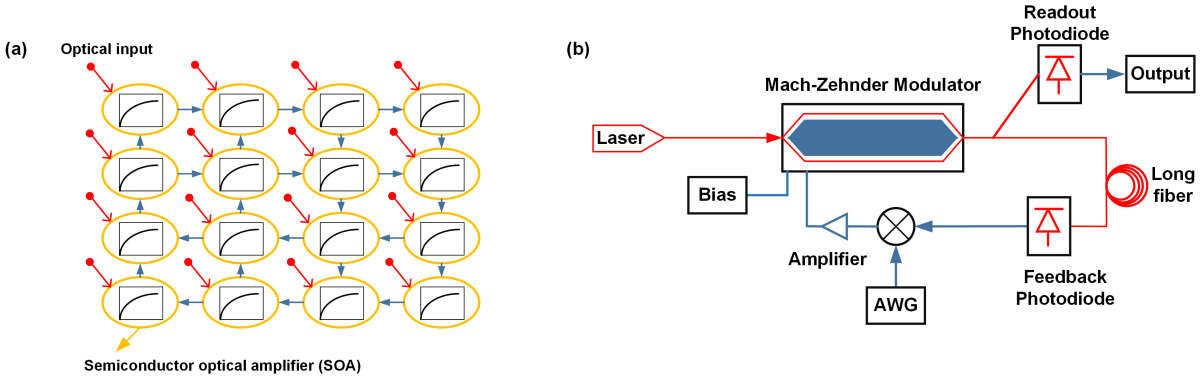

Figure 2.3: (a) A 4×4 swirl topology for spatially distributed photonic RC using SOA, adapted from [13] (CC BY 4.0). (b)The building blocks for an optoelectronic implementation of delay-based RC, adapted from [90] (CC BY-NC-SA 3.0). The delay is generated by a long fiber.

connected to the nearest 4 neighbours in a swirl topology, as demonstrated in Fig. 2.3(a). This work was a preliminary simulation to validate the feasibility of photonic RC systems [87]. A further investigation, including structure design, noise effects and parameter fine-tuning, was followed up [44]. Finally, this prototype was experimentally performed on a silicon-photonics chip [91]. However, the actual readout was trained offline and implemented electronically. Instead of SOAs, the optical node can also be implemented by diffractive optical elements (DOEs). For example, Daniel *et al.* applied the semiconductor laser based on diffractive optical coupling to form a 8×8 optical array [92]. While the above two implementations fall within the domain of classical ANNs, where the optical nodes serve as the artificial neurons that sum up the weighted current and provide nonlinearity, a different approach to exploit the spiking behaviours of excitable photonic was investigated using laser systems [93].

Delay-based design

While the time-delayed dynamical system was introduced in 2011 to reduce the complexity of the physical reservoir [29], following-up research focus not only on the electronics-based implementations as introduced in Section 2.3.1, but also in the field of photonics. It is worth mentioning that delayed coupling inherently comes from unwanted reflections, which leads to oscillations and chaos in optics. This property was utilized to develop time-delayed optical RC systems. A general building block of this system is shown in Fig. 2.3(b). Two of the earliest research in this area both used an optoelectronic oscillator to generate delay feedback [90, 94]. Light is continuously emitted from a laser source and modulated, while the delayed feedback loop is implemented by a long fiber, an amplifier and a filter. Besides, photodiodes are applied to detect and collect output states, saved for training in the post procedures. Despite the success of optoelectronic reservoirs, significant efforts have been directed toward the implementation of all-optical RC due to its potential for integration into photonic chips. Electronics in the feedback loop (filter and amplifier) were replaced by all-optical components like SOAs, fiber

Table 2.2: Various biomedical signals and their representative datasets.

Signal	Dataset	sensor	position	Applications
ECG	MIT-BIH AR [101]	ECG electrodes	Chest, arms, and legs	Cardiovascular arrhythmias detection
PCG	PhysioNet [102, 103]	Digital stethoscope	Chest	Abnormal heartbeats and heart murmurs detection
	-	Optical stethoscope	Remote, laser pointed at neck	Biometric identification
PPG	PhysioNet [104]	Photosensors	Wrist and finger	Heart rate monitoring
sEMG	NinaPro [105, 106, 107, 108, 109, 110]	EMG electrodes	Forearm surface	Gesture recognition and estimation
	BioPatRec [111]			
	HYSER [112]			
MMG	-	magnetic sensors	Muscle surface	human-computer interaction
EEG	Bonn [113]	EEG electrodes	Scalp	Epilepsy seizure detection
	CHB-MIT [114]			Epilepsy seizure detection
	SEED [115]			Emotion Recognition

coupler and fiber amplifiers [95]. Brunner *et al.* reached a milestone of applying a network size of 388 neurons to solve computationally hard tasks at data rates beyond 1 Gbyte/s, which bridges the gap between photonic RC and cognitive information science [96]. With the theory foundation and feasibility verification in terms of delay-based photonic RC, following research endeavour in refining input layer time-masking [60, 97], analog readout layer [98], reducing hyperparameters [99], and on-chip implementations [100].

2.4 Biomedical signal applications

Biomedical signals are generated by biological systems, such as the human heart, muscle, and brain, which can be measured and analyzed to provide valuable information useful in medical diagnoses, monitoring, and treatment. While intelligent wearable sensor systems have revolutionized healthcare by enabling continuous monitoring of biomedical signals with the advancement of artificial intelligence techniques, the vast amount of real-time biomedical data generated by these devices requires efficient processing solutions to ensure timely analysis and response. Neuromorphic approaches for edge computing contexts provide a platform for processing data locally on the wearable device or nearby, rather than sending data to a centralized server for subsequent post-processing, thereby saving computational overhead and decreasing bandwidth usage [1]. In this section, we review various types of biomedical signals that have been widely used by RC methods and their publicly available benchmarks. Table 2.2 shows the summary of biomedical signals and Fig. 2.4 shows the representative biomedical signals and their applications.

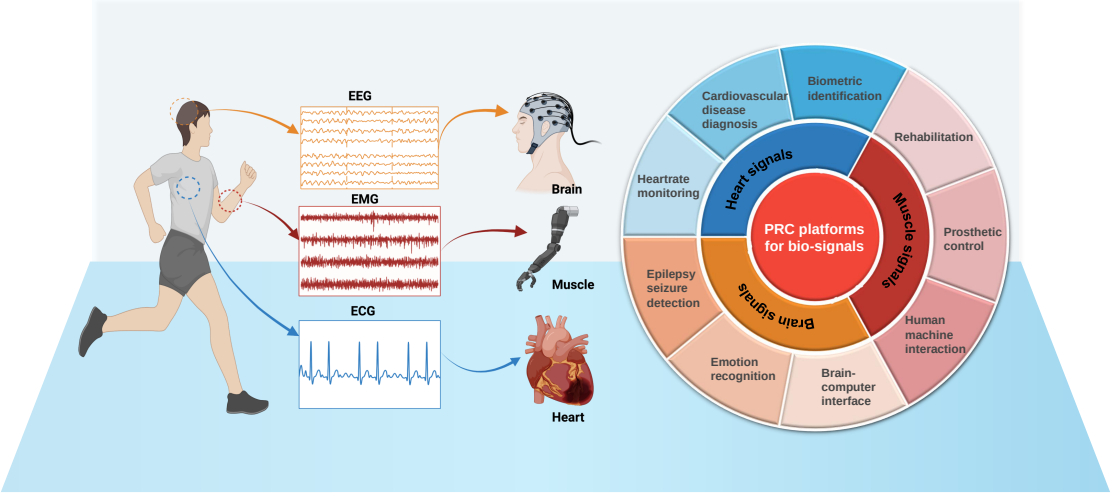

Figure 2.4: Left: Three representative signals from brain, muscle, and heart, respectively. Right: The applications of biomedical signals in a sector chart.

2.4.1 Heart signals

Electrocardiography (ECG)

The electrocardiography (ECG) is a non-invasive technique used to measure the electrical activity of the heart over time. It records electrical impulses generated by heart through electrodes placed on the skin (usually on the chest, arms, and legs), providing valuable information about heart rhythm, rate, and potential abnormalities such as arrhythmias and other cardiovascular disorders. This technology is widely used in medical diagnostics, continuous health monitoring for wearable cardiac devices. The MIT-BIH arrhythmia (MIT-BIH AR) database is one of the most internationally authorised large-scale ECG datasets, including 48 ECG records, each lasting 30 minutes [101]. All the heartbeat annotation labels are classified into five heartbeat types by independent experts: N (normal beats), S (supraventricular ectopic beats), V (ventricular ectopic beats), F (fusion beats), and Q (unclassifiable beats). In the term of RC algorithm, a software-based RC classifier was used for arrhythmia detection based on a classical (random-connected) reservoir topology and a ring (cyclic) topology [116]. The delay-based PRC topology was also applied for ventricular heartbeat detection with the delay node simulated by a circuit [16]. Regarding the experimental implementation of PRC for ECG-related applications in hardware, various approaches have been explored. Specifically, a random-connected reservoir was deployed on a neuromorphic chip in the SNN domain [24]. Furthermore, a delay-based topology was implemented on FPGA and ASIC chips [55, 72]. In addition, a fully analog, memristors-based PRC system has been implemented for the task of ventricular heartbeat detection [18].

Phonocardiography (PCG)

The phonocardiography (PCG) is a non-invasive technique used to record the sound waves produced by the structural and hemodynamic changes consisting of a complex of heart muscle, valves, and blood flow. It involves the use of a digital stethoscope consisting of microphone-based acoustic sensors placed on the chest to capture vibrations produced by the heart during the cardiac cycle. Normally, the first (S1) and the second (S2) heart sounds are the two most easily recorded. PCG signals are useful in detecting cardiovascular diseases (CVDs), such as aortic stenosis, mitral regurgitation, and heart murmurs. Numerous PCG heart sound recordings can be accessed in PhysioNet, for purposes of abnormal heartbeat detection and heart murmurs detection [102, 103]. PCG signals serve as a complementary tool of ECG in medical applications, a PRC based framework was proposed to predict ECG to PCG by regression methods through a custom-developed dataset, aiming at decreasing clinical measurements for patient while providing distinct insights into operations of the heart for doctors [28]. Despite advancements in PRC systems, their applications in heart murmur detection are limited. Future research in this area could further demonstrate the ability of PRC systems in CVD applications. Instead of CVDs detection, heart sound can also be a biometric information for identification [117]. A custom-developed optical stethoscope-based PCG dataset was utilized for biometric identification, while the optical stethoscope is composed of a laser-camera system with laser pointing at the neck remotely [118].

Photoplethysmography (PPG)

The photoplethysmography (PPG) is a non-invasive optical technique used to measure blood flow changes generated by the heartbeat. It relies on a light emitter and a photosensor placed on the wrist or finger to detect variations in light absorption, which correspond to pulsatile blood flow. PPG is widely used in heart rate monitoring, blood oxygen saturation measurement, and vascular health assessment. Compared with ECG and PCG, although the PPG delivers limited heart-related information, it is low-cost in measurement, making it a popular intermedia in heart rate measurement for wearable devices. In terms of datasets, PPG is usually recorded as a supplementary indicator for ECG and other bio-signals. The PhysioNet provides accessible PPG dataset [104], and an optoelectronic-based delay node PRC system was applied to detect atrial fibrillation based on this dataset [119].

2.4.2 Muscle signals

Surface electromyography (sEMG)

The surface electromyography (sEMG) is a non-invasive method used to capture the electrical activity of the skeletal muscles, offering valuable information on muscle function for diagnosis,

rehabilitation, and various medical applications. This technology supports applications in wearable devices, particularly in fields such as human-computer interaction, prosthetic control, and robotics [120]. Great efforts have been made in establishing sEMG datasets, several datasets that provide sEMG signals for a variety of tasks are open-access, enabling the development of algorithms and models for signal analysis. Non-Invasive Adaptive Prosthetics (NinaPro) databases (from DB1 to DB9) are the most widely utilized dataset, comprising EMG signals collected from multiple subjects performing a variety of hand and finger gestures [105, 106, 107, 108, 109, 110]. This dataset is extensively applied in hand gesture classification, as well as in the estimation of hand kinematics and dynamics. Similarly, Biological Pattern Recognition (BioPatRec) datasets also contain multiple databases, capturing various hand movements such as finger gestures and wrist movements [111]. High-densitY Surface Electromyogram Recordings (HYSER) is a more recently released high-density sEMG dataset, particularly useful for muscle force estimation with high-density sEMG channels [112]. Among them, a PRC system was successfully applied in gesture recognition by applying the NinaPro DB2 in gesture classification tasks in the domain of SNNs [82]. In addition to the widely used large datasets, some custom-developed sEMG datasets were also applied in the performance evaluation of PRC systems [57]. Successful demonstrations highlight the feasibility of PRC systems in sEMG signal processing. However, further investigations are required to explore various implementation platforms of PRC in the context of sEMG, particularly in large-scale datasets and more complex tasks.

Magnetomyography (MMG)

The magnetomyography (MMG) is a technique for measuring the magnetic fields generated by human muscle activity, serving as an informative bio-signal. In recent years, MMG has gained significant research interests due to its potential applications in biomedical and neuromuscular studies. While EMG and MMG signals are both biomedical signals resulting from muscle activities and play essential roles in health monitoring and human-machine interaction, EMG is the measurement of the electrical currents generated from muscle activities and MMG signal is from the magnetic field generated by the electronic currents [121, 122]. Although MMG signals are subject to various sources of background magnetic noise from the surrounding environment, which pose challenges for noise reduction, they offer the advantages of being measured in a contactless manner and providing superior spatial resolution compared to other techniques. Since MMG is an emerging area, it is at the infancy stage of focusing on magnetic sensors development and measurement setup [123]. A few large-scale and open-access datasets could be found in the literature. Instead, attempts of applying PRC for MMG to EMG mapping were studied to extract desirable information from the noisy MMG signals through a custom dataset [27].

2.4.3 Brain signals

Electroencephalography (EEG)

The electroencephalography (EEG) is a non-invasive technique used to record electrical activity in the brain. It captures voltage fluctuations resulting from synaptic activity among neurons using electrodes placed at specific scalp locations to capture activity from different brain regions. EEG has a wide range of applications in medical diagnostics by providing real-time monitoring of brain activity, such as epilepsy, sleep disorders, and brain injuries. In neuroscience research, EEG is used to study cognitive functions, attention, memory, and emotional responses. In addition, EEG plays a crucial role in brain-computer interfaces (BCIs), an emerging research area that enables real-time direct control between the brain and external devices, which benefits individuals with disabilities. A substantial number of EEG datasets have been developed, each designed for specific research applications. In the domain of epilepsy detection, the Bonn EEG dataset, constructed by the University of Bonn [113], and CHB-MIT EEG dataset, collected at the Children's Hospital Boston [114], are widely recognized as benchmark datasets for evaluating classification models. Memristive PRC systems were applied for epileptic seizure detection based on the Bonn dataset [124, 125]. Moreover, EEG-based emotion recognition datasets have witnessed significant progress in recent years, with several open-access datasets emerging. Among them, The SJTU Emotion EEG Dataset (SEED) dataset is one of the most widely used datasets for affective computing research [115]. Based on the contribution of previous studies, PRC systems implemented across various frameworks, algorithms, and platforms demonstrate significant potential in advancing EEG-related applications.

2.5 Training the readout layer

While the majority of RC-related literature focuses on the implementation of the reservoir layer, the training process, despite its simplicity, still requires careful consideration, particularly for complex classification tasks in biomedical applications. A wide range of strategies are applied to train the readout layer. In this section, we provide a summary of mainstream training methods for RC readout layer in the field of ANN and SNN, separately. A design toolbox useful for selecting training methods is introduced in Fig. 2.5. A summary of the representative training methods for various biomedical tasks can be referenced in Table 2.3.

2.5.1 ANN-based training methods

A traditional way to train the readout layer is to apply linear regression on the high-dimensional reservoir states by solving an optimization problem to determine the optimal output weight matrix W_{out}, and a prediction is simply made by a vector–matrix multiplication (VMM) as described in equation (2.4). Two common linear regression techniques are defined as lasso regression

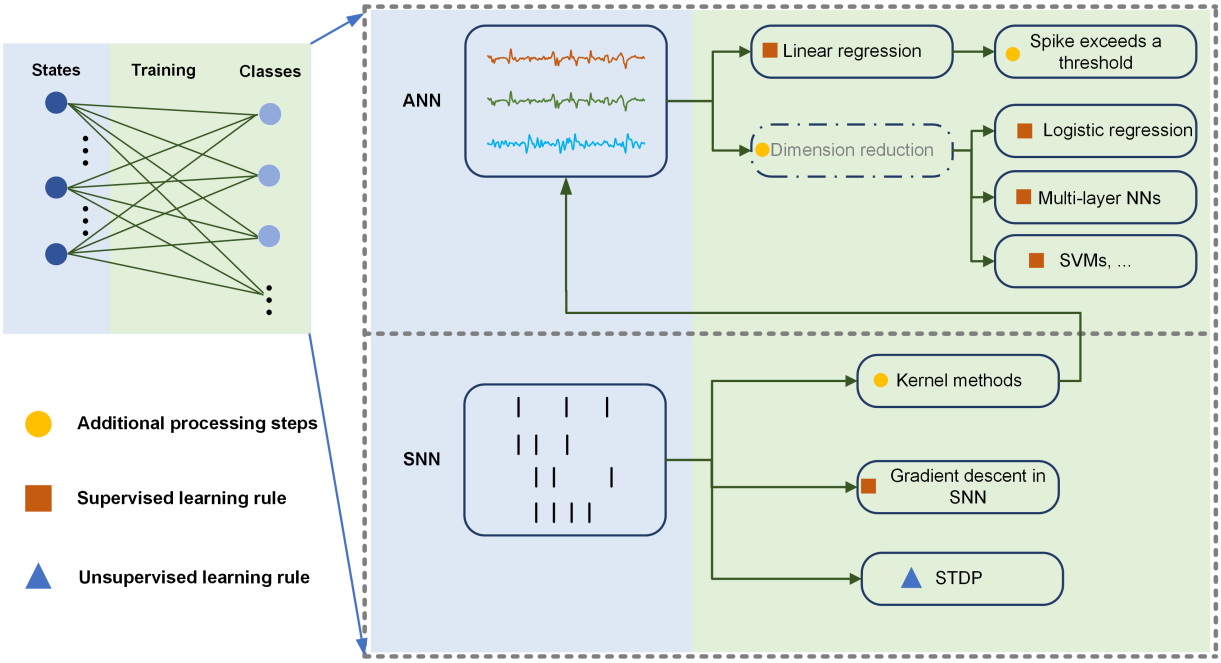

Figure 2.5: A design toolbox for training the readout layer.

(L1 regularization) and ridge regression (L2 regularization). The Lasso regression applies L1 penalty to enforce sparsity as shown in equation (2.5), while the ridge regression applies L2 to minimize the cost function in equation (2.6). β is the regularization parameter in both cases. While ridge regression is usually preferred in training RC readout by providing a closed-form solution shown in equation (2.7) for lower computation cost compared with lasso regression which involves iterative methods, lasso regression may perform well in certain tasks with sparse representation [16].

$$\mathbf{y}(\mathbf{n}) = \mathbf{W_{out}} \cdot \mathbf{x}(\mathbf{n}) \tag{2.4}$$

$$\mathbf{W}_{out} = \arg\min_{\mathbf{W}} \sum_{t} \|\mathbf{W}\mathbf{x}(t) - \mathbf{y}(t)\|^2 + \beta \sum |w_i| \tag{2.5}$$

$$\mathbf{W}_{out} = \arg\min_{\mathbf{W}} \sum_{t} \|\mathbf{W}\mathbf{x}(t) - \mathbf{y}(t)\|^2 + \beta \|\mathbf{W}\|^2 \tag{2.6}$$

$$\mathbf{W}_{out} = (\mathbf{X}^T\mathbf{X} + \beta I)^{-1}\mathbf{X}^T\mathbf{Y} \tag{2.7}$$

However, while linear regression methods are simple and commonly employed for time-series prediction tasks, classification tasks necessitate additional considerations. One solution could be still applying linear regression methods, wherein the correct prediction in the output vectors

Table 2.3: A summary of the training methods and accuracy for reservoir computing algorithms in biomedical applications.

Reference	Signal	Network type	Platform	# of classes	Training method	Accuracy (%)
Liang et al [15]	ECG	ANN	Simulated circuit	2	Lasso regression+detect spikes exceed threshold	98
Bauer et al [24]	ECG	SNN	Neuromorphic chip	5	Kernel+least mean squares+detect spikes exceed threshold	92
Chandrasekaran [55]	ECG	ANN	65nm CMOS chip	2	Logistic regression	87
					SVM	87
					2-layer neural network	86
Sharma et al [119]	PPG	ANN	Photonics	2	2-layer neural network	83
Ma et al [82]	sEMG	SNN	Neuromorphic chip	3	STDP	83
					SVM	75
				8	STDP	57
					SVM	35
Donati et al [57]	sEMG	SNN	Neuromorphic chip	3	Logistic regression	81
					SVM	84
					kernel+delta rule	74
Garg et al [126]	sEMG	SNN	Software	3	SVM	88
					LDA	83
				5	SVM	70
					LDA	62
Fourati et al [127]	EEG	ANN	Software	8	Intrinsic plasticity rule	70
Merkel et al [124]	EEG	ANN	Memristors	2	Linear regression+detect spikes exceed threshold	85
Kudithipudi et al [125]	EEG	ANN	Memristors	2	Least mean square through memrsitor crossbar+detect spikes exceed threshold	90

will appear as a spike, while the values corresponding to other classes remain flat. In this case, an algorithm can be utilized to identify the spike that exceeds a predefined threshold or to determine the maximum value within a specified interval, also known as a WTA method [16, 17, 45]. Another solution is to apply nonlinear methods, such as logistic regression, multi-layer neural networks, and SVM classifiers, which are better for classification tasks. However, for complex tasks, reservoir states are typically high-dimensional and high-resolution, leading to increased computational complexity. To address this, some dimensionality reduction techniques are employed. A common approach involves selecting either the last reservoir state or the mean of all reservoir states, though this often results in a reduction in classification accuracy. Alternatively, methods such as principal component analysis (PCA) and other dimensionality reduction strategies can be applied to mitigate computational demands while preserving performance [19].

2.5.2 SNN-based training methods

Different from RC in the ANN domain, the spiking RC generates outputs in terms of spikes. Although SNN brings the merits of sparse and low-precision representation beneficial to storage and computation costs, it is tricky to find a suitable learning rule due to the non-differentiability of spikes. One traditional way involves applying kernel methods to transform spike trains into continuous series and natural numbers and then using common techniques in signal processing for further analysis [24, 57]. A comprehensive review of kernel methods is presented in [128].

In recent years, driven by the significant success of DNNs, there has been growing research interest in applying gradient descent methods to SNNs. In the training of a single-layer network, a simple weight update rule, such as the delta rule, is applied to minimize the loss and converge to a local minimum. However, complex tasks typically require multi-layer networks for effective loss convergence, necessitating the use of gradient descent methods to propagate errors backwards. Due to the non-differentiability of spikes, traditional gradient descent methods cannot be directly applied to SNNs. Efforts have been directed toward identifying approximate derivatives of spike activity to address the non-differentiability issue inherent in SNNs [129, 130]. A thorough tutorial of applying lessons learned from DNNs to train SNN is presented in [131].

Distinct from classical ANN-based learning rules, the spike-timing-dependent plasticity (STDP), a more biologically plausible learning rule, was proposed as an unsupervised learning rule, being more suitable for SNNs [82, 132, 133]. The fundamental weight update rule is expressed in equation (2.8):

$$\Delta w = \begin{cases} A^+ \cdot \exp\left(\frac{\Delta t}{\tau^+}\right), & \text{if } \Delta t > 0, \\ -A^- \cdot \exp\left(\frac{-\Delta t}{\tau^-}\right), & \text{if } \Delta t < 0. \end{cases} \tag{2.8}$$

Where Δw is the change in the synaptic weight, $\Delta t = t_{\text{post}} - t_{\text{pre}}$ is the time difference between the postsynaptic and presynaptic spikes, A^+ and A^- are the scaling constants for potentiation and depression, respectively, τ^+ and τ^- are the time constants that govern the time scales for potentiation and depression. It adjusts synaptic weights based on the precise timing of spikes between pre- and post-synaptic neurons. The implementation of the local synaptic plasticity learning rule is promising to be implemented physically by circuits and to build efficient on-chip learning in neuromorphic processing systems [134].

2.5.3 Summary of AI techniques across chapters

Based on the training methods mentioned above, several supervised training methods were applied to solve certain tasks in the following chapters, ranging from ANN-based methods to SNN-based methods. A detailed breakdown is introduced as follows:

- Linear regression method is used for the prediction tasks described in Chapter 3.
- Linear regression method and an algorithm to detect where the peak occurs are used for the biometric identification task described in Chapter 4.
- Chapter 5, a kernel method is employed to transform spike-based event data into numerical representations for an event-based gesture recognition task. An SVM serves as the baseline model to optimize parameters during spike encoding and network construction. Furthermore, a delta learning rule paired with a Softmax classifier is applied to the same task, as the delta rule's computational simplicity makes it amenable to hardware implementation, potentially enabling a fully neuromorphic system in future work.
- The normalized root mean square error (NRMSE) is used for evaluating the performance for predictions and a performance matrix including accuracy (Acc), specification (Sp), sensitivity (Se), precision and F1 score is used to evaluate the classification tasks. In addition, for classification tasks, the confusion matrix that compares the predictions against actual (true) values is also applied as a performance evaluation technique in classification tasks.
- Network size and power consumption are applied as factors for measuring computational efficiency.

2.6 Conclusion and Discussion

This chapter reviews a wide range of implementation paradigms for PRC systems based on different categories. In addition, a variety of training methods for the readout layer were discussed and provided in the review. In conclusion, PRC holds significant promise in pioneering innovative solutions for wearable biomedical devices in the context of edge computing, by enabling computationally efficient yet effective algorithms in resource-constrained environments.

In practical hardware implementations, CMOS-based circuits have gained a large variety of applications, achieving low power consumption. Although certain large-scale neuromorphic systems offer a platform with reconfigurable network structures for PRC and have demonstrated feasibility in biomedical applications, simplified hardware architectures, as illustrated in Fig. 2.2, present a promising opportunity for integration into CMOS chip-level implementations with reduced hardware costs. Beyond conventional CMOS technology, emerging memory devices hold significant potential, enabling in-memory computing solutions with ultra-low power consumption. However, challenges related to device reliability, including D2D and cycle-to-cycle (C2C) variability, must be addressed to ensure their practical deployment in biomedical applications [12]. Photonics provide a faster computation speed, the energy consumption is not optimal compared with other solutions, and a critical research question is the alignment of

timescales between the task and the reservoirs [135].

PRC currently exhibits limited real-world applicability compared to established artificial intelligence (AI)-methods, primarily due to its nascent developmental stage. Further research is warranted to advance PRC through improved algorithmic frameworks and optimized hardware implementations, which could enhance its practical utility in future applications.

While the earlier proposed delay-based PRC model has been widely studied, the RNR architecture is chosen as the main objective in this thesis due to several advantages: (1) The absence of ADC and digital-to-analog converter (DAC) modules reduce hardware cost. (2) Balance the memory capacity (MC) and state richness. (3) Parallel computing owing to the unnecessity of time-multiplexing. The improvement on RNR network and the applications in various tasks are introduced in the following chapters.

Chapter 3

PRC as Predictors

Under the term neuromorphic, PRC processes the raw signals in the analogue domain with in-memory computing, thus reducing massive power consumption and adaption delay. At the same time, RC is a special type of RNN that is suitable for time-dependent signal processing. The biomedical signals generated from the terminal of wearable devices could be applied by using PRC in prediction tasks under medical contexts, and specific applications in muscle signals (MMG and EMG) and heart signals (PCG and ECG) are covered in this chapter.

3.1 Introduction

3.1.1 MMG/EMG mapping

Measuring muscle movement has been a challenging area over the past decades. EMG and MMG signals are both biomedical signals resulting from muscle activities and play essential roles in health monitoring and human-machine interaction [121, 122]. EMG is the measurement of the electrical current generated from muscle activities, and the MMG signal is from the magnetic field generated by the electronic current. While EMG provides low spatial resolution, other equivalent methods such as MMG can address this issue [136]. However, MMG suffers from various noises, including $1/f$ noise and ambient noise. Spatial resolution in measuring muscle activity can be increased by using needles inserted under the skin to collect the signals, which is painful and inconvenient [137]. The advantage of MMG signals is that they can be measured in a contactless way and more easily collected. However, MMG signals contain various background magnetic noises from surrounding equipment that are challenging to remove. Similar problems occur in magnetocardiography (MCG) signals. Noise-removal methods by mapping the MCG to ECG signals by applying machine learning methods including deep learning and software-based RC have been implemented [26]. RC has demonstrated a fast computation ability and a small error in MCG/ECG mapping tasks. In principle, similar methods could be applied to MMG for

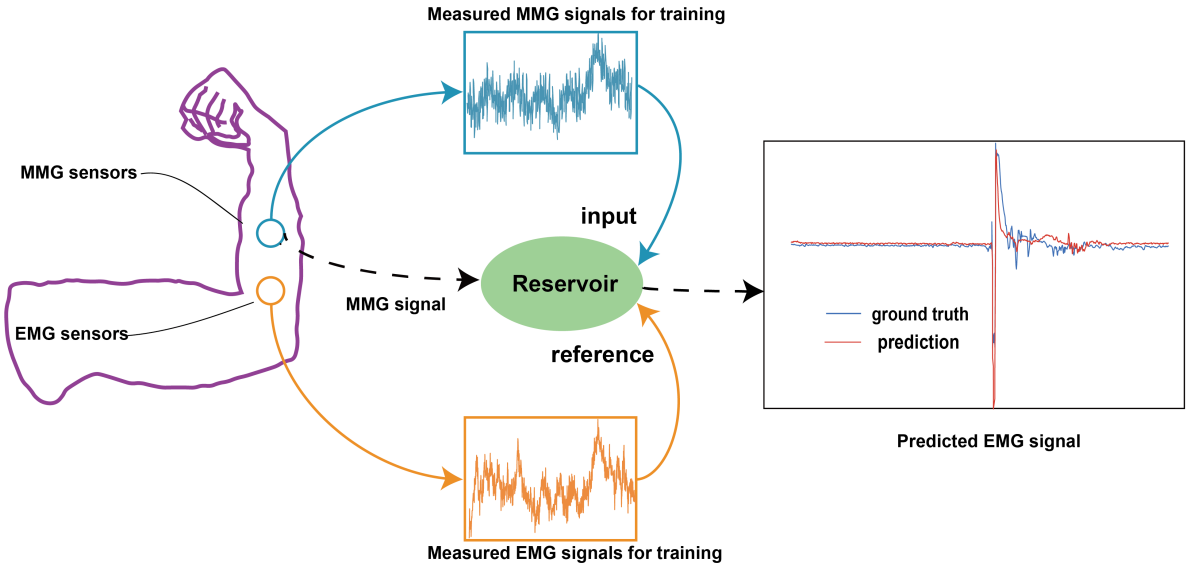

Figure 3.1: The conceptual picture of the EMG-MMG mapping process

noise-removal purposes. In our work, both the MMG and EMG datasets are real-world data collected from humans.

3.1.2 ECG-to-PCG signals prediction

The ECG is a skin-surface measurement of the potential changes that result from the electrical polarisation changes of the heart. The magnitude of the signal recorded at the cutaneous level depends on the mass of the activated heart muscle. The recorded waves, therefore, relate to the activation state of the heart. In particular, a normal ECG recording usually contains three distinct features - the QRS complex, and the P and T waves- that arise from different activation states. The P wave indicates the onset of atrial contraction, the QRS complex arises from the activation of the ventricles, and the T wave is produced by the electrical repolarisation of the ventricles. The PCG, on the other hand, is a recording of the sound waves produced by the structural and hemodynamic changes that consist of a complex of the heart muscle, valves, and blood flow. As the heart repeats its contraction-relaxation (systole-diastole) cycle, pressure waves emanate to the chest wall and this is then set into vibration. Therefore, a PCG recording is mainly produced by those cardio-hemic effects that are energetic enough to travel to the skin surface without dissipating. Normally, the first (S1) and the second (S2) heart sounds are the two most easily recorded. Unsurprisingly, these are related to the most energetic heart muscle movements - the onset of the systole and diastole, respectively.

PCG and ECG are both diagnostic tools used in cardiology, but they provide different types of information about the heart and its functioning. Since ECG provides information about the electrical activity of the heart, it can diagnose arrhythmias, ischemia (inadequate blood flow to

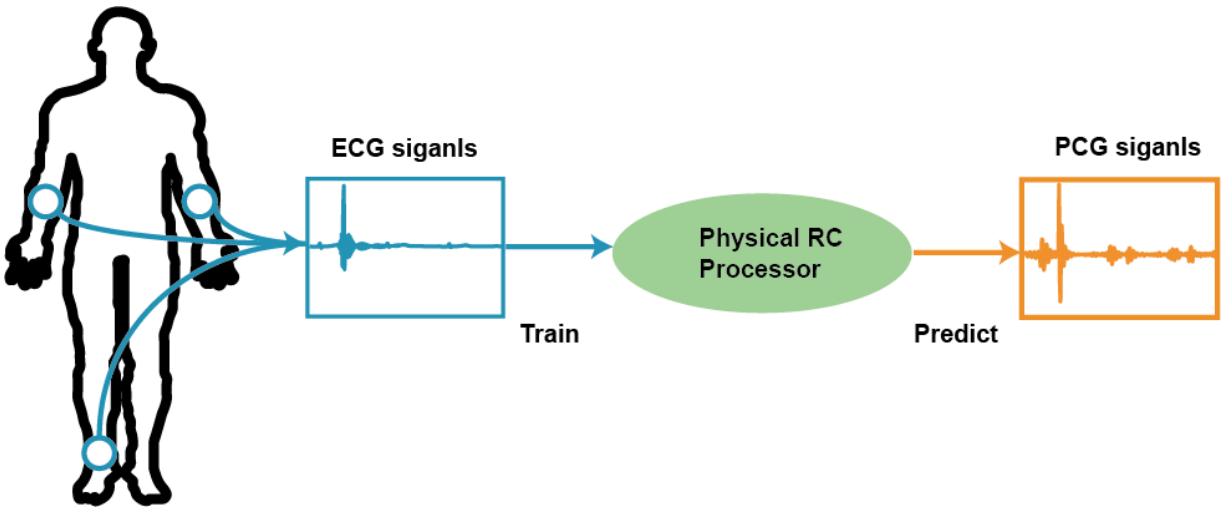

Figure 3.2: The conceptual figure of ECG-to-PCG prediction using a physical reservoir processor.

the heart), and conduction system abnormalities. It's particularly useful for identifying irregular heart rhythms and ischemic events. PCG provides information about the mechanical aspects of the heart's functioning. It is often the first screening for cardiac problems and can help identify abnormalities in heart sounds, such as murmurs or extra heart sounds, which can be indicative of valvular problems, chamber abnormalities, or other structural heart issues. From a clinical standpoint, obtaining ECG measurements is more convenient for healthcare professionals because the traditional medical stethoscope cannot deliver PCG data in a digital format. Therefore, the ability to predict PCG information from recorded ECG signals is highly valuable, particularly considering that these two signals offer distinct insights into the heart's operation. Machine learning algorithms can be applied to predict the PCG based on collected ECG signals to avoid repetitive measurement for patients while providing valuable insights into heart operations to assist in medical diagnosis in clinical environments. Machine learning algorithms can be applied to predict the PCG based on collected ECG signals. In this work, we applied RNR to perform the ECG-to-PCG prediction task for 10 subjects, with data collected in an office environment at the University of Glasgow.

3.2 Methodology

3.2.1 Network description

RC is a special type of RNN, where the current network responses are dependent not only on the current inputs but also on the historical inputs. The network can be sensitive to small changes in the time series, which endows it with an excellent ability to process time-dependent sequences. Some also think of RC as an analogy to SVM, and consider the reservoir as a temporal kernel [22]. Both techniques apply a trick to map the input to a high-dimensional

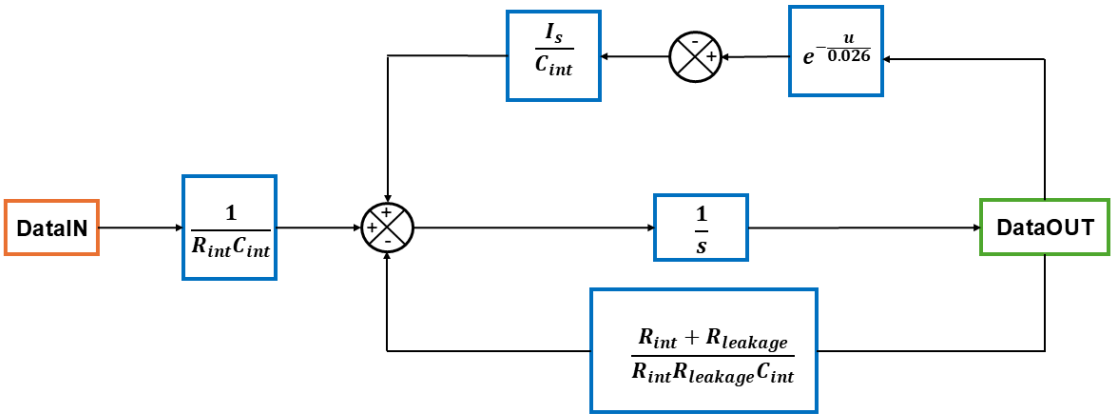

Figure 3.3: The block diagram of the circuit inside the dynamic neuron simulated in Simulink.

feature space to allow for linear separation. The difference is that RC has a recurrent nature owing to its structure.

A reservoir network contains three layers: an input layer, a reservoir layer, and an output layer. The injected input $u(n)$ is multiplied by an input weight matrix W_{in}, and then this product goes to the reservoir layer. The reservoir generates transient responses $s(n)$ for the input injection. The process can be represented by the state update equation as shown in equation (2.3) in Section 2.2.1 of literature review chapter.

The responses, also known as the states of the reservoir, are collected for training. In our work, we used a simple linear regression called ridge regression for the prediction task. The output weight matrix W_{out} is calculated by the equation discussed in the equation (2.7) in Section 2.5.1. Finally, in the output layer, the prediction at time n is achieved by multiplying the output weight matrix W_{out} and the state $s(n)$ generated for the input at time n, denoted by equation (2.4) in Section 2.5.1.

The RNR architecture is the electronic circuit implementation of the SCR topology as discussed in Section 1.2.2. Analogue multiplexers are used to shift the connections between input-to-reservoir and reservoir-to-output and deliver the masked input at each time step. The dynamic neurons are modelled by Leaky-Integrate circuits, with a diode to act as the ReLU activation function. The dynamic responses of each neuron at the whole training phase are collected for training by linear regression. The dynamic neuron is simulated in Simulink, MATLAB, and the building blocks inside the dynamic neuron is presented in Fig. 3.3.

3.2.2 Dataset description for MMG and EMG

The MMG and EMG datasets used here were collected at the BMSR-2 (Berlin Magnetically Shielded Room 2), located in the national metrology institute in Berlin, Germany utilizing a

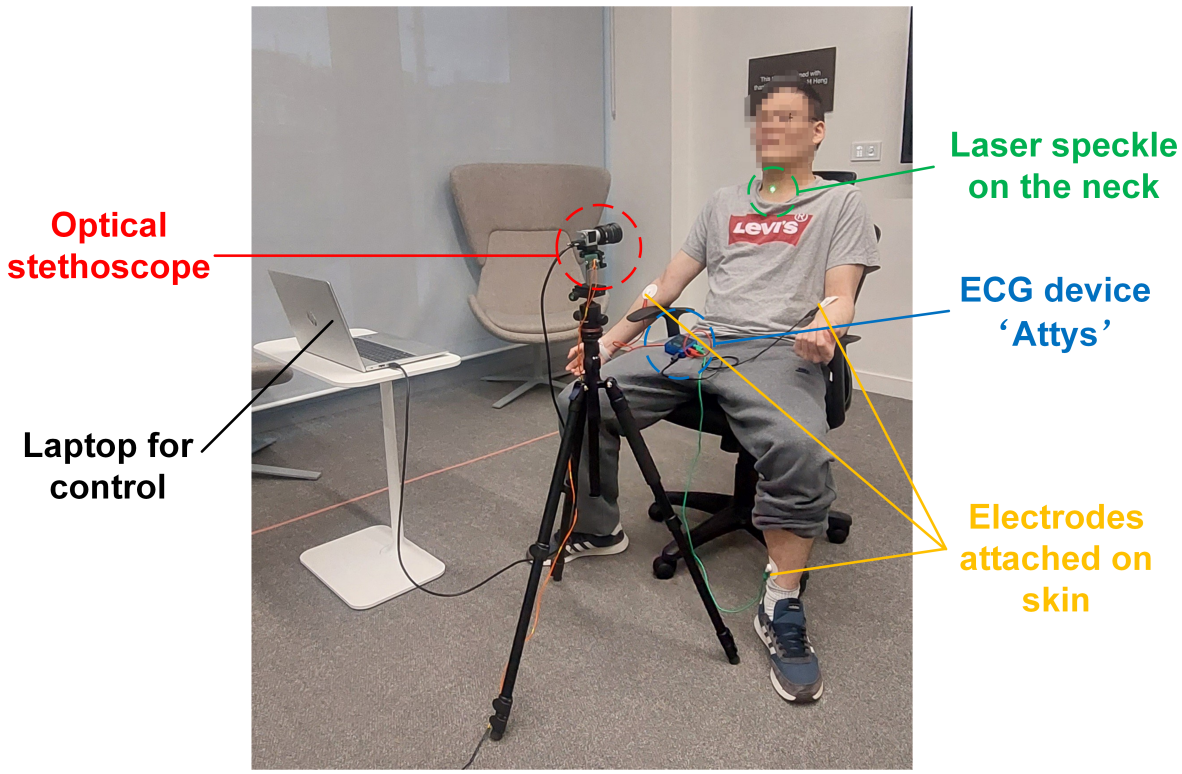

Figure 3.4: Experimental setup of ECG and PCG measurement

SQUID sensors array. The measurement system consists of 92 channels of MMG signals, 4 channels of surface EMG signals and 1 channel of needle EMG. The experiments were conducted according to the standards of the World Medical Association (World Medical Association, 2001). The subject of this study gave his consent for his data to be published online. MMG and EMG signals were collected synchronously at an 8000 Hz sampling rate.

Due to the limited memory capacity in RC system, the network cannot remember a long-term signal. Therefore, downsampling is required to shorten the length of signals, so the information within a same time duration can be less and easier to be remembered by the network. The original sampling rate for both MMG and EMG signals are 8000 Hz, which was down-sampled to 500 Hz. The raw data was injected into the network without preprocessing or feature extraction.

According to the dataset, there are 92 MMG channels in total and 4 EMG channels. However, the last two EMG channels were skipped because of their poor signal quality. 70% of the MMG and EMG datasets were treated as training data and the remaining were treated as testing data.

3.2.3 Dataset description for ECG and PCG

ECG and PCG were simultaneously gathered in an office setting at the University of Glasgow, as depicted in Fig. 3.4. The data was acquired using the 'Attys' device, a creation of the Extreme Light Group at the University of Glasgow. This ECG device is linked to three identical electrodes

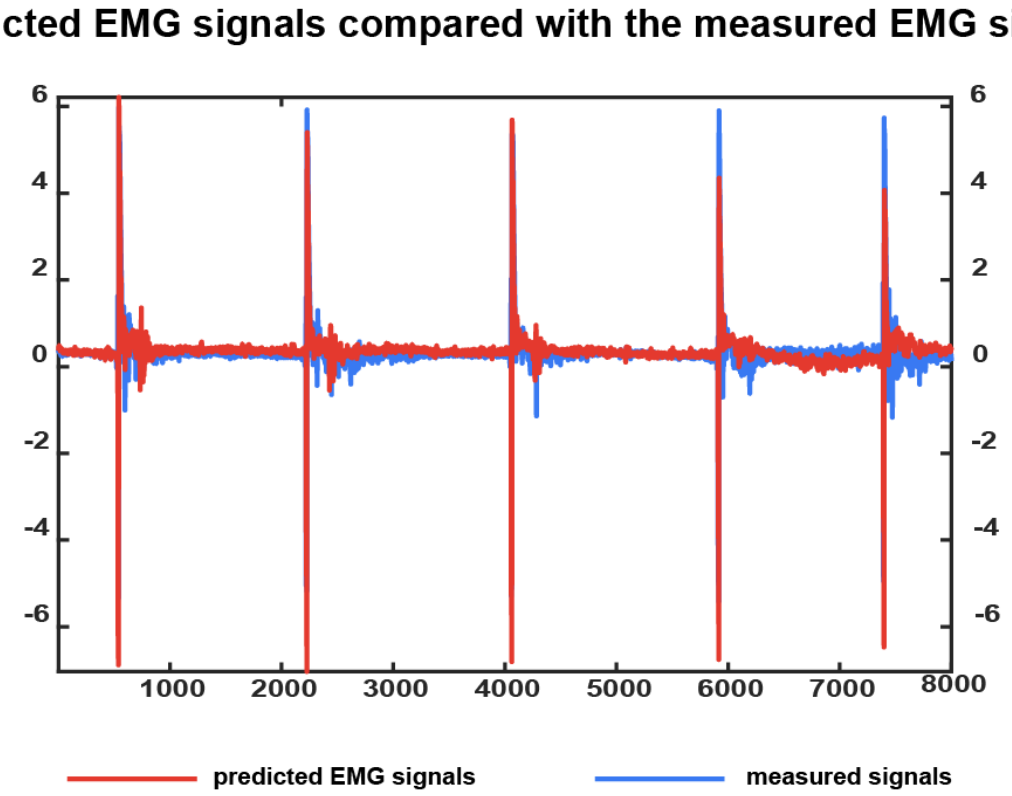

Figure 3.5: The results of the prediction using RNR while $\gamma = 0.7, \tau = 3s$.

for recording electrical signals from the left and right elbow, as well as the inner side of the ankle. Subsequently, the collected data is transmitted to a Bluetooth connection. As for the PCG data, it was obtained using an optical stethoscope, positioned approximately 1 meter in front of the test subject. This optical stethoscope comprises a 532nm laser diode and a high-speed industrial camera. The laser diode emits a beam towards the subject's neck area, while the camera captures the reflected speckle pattern from the neck. Due to that the laser speckle movement is highly correlated with the blood flow movement in the neck blood vessel, a Farneback optical flow algorithm [138], is then utilized to transform the laser speckle vibrations into the audio PCG signal.

The ECG and PCG signals were downsampled from 500Hz and 1470Hz to 150Hz respectively owing to the limitation of memory capacity in RC. After the downsampling, the signals are injected into the network directly without any segmentation and feature extraction, which is one of the merits of RC.

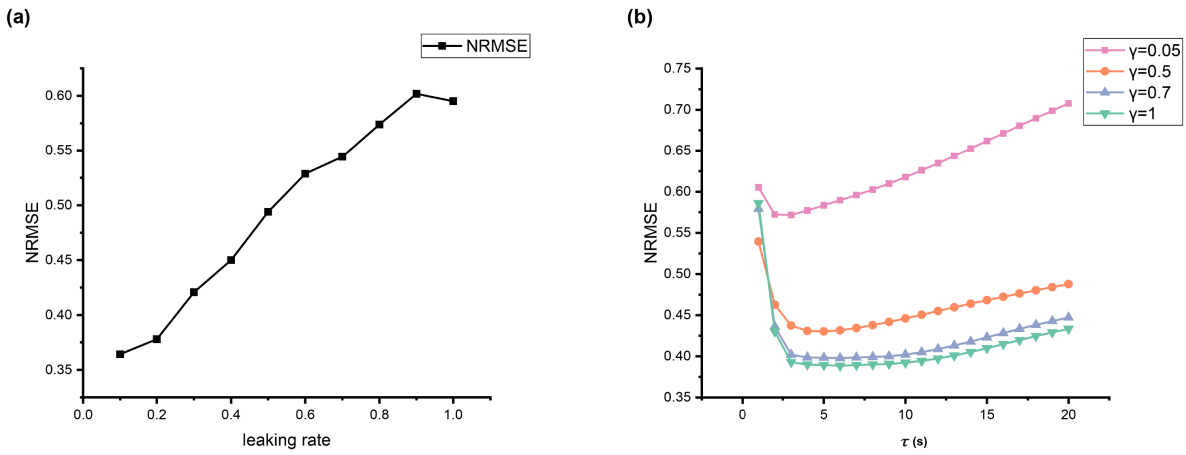

Figure 3.6: (a) The relationship between NRMSE and leaking rate for SCR network (b)The relationship between NRMSE and input scaling parameter γ and time constant τ for eRNR network. The reservoir size is fixed to 400.

3.3 Results and analysis

3.3.1 Prediction error for MMG/EMG mapping

The result that compares the estimated EMG signals and the measured EMG signals is shown in Fig. 3.5. For the EMG signals that only contain spikes, the network could accurately capture where the spike occurs and the peaks of the spikes. Parameters in both SCR and eRNR structures could be adjusted to achieve better performance.

For reservoir size in both structures, the NRMSE does not decrease as the reservoir size increases in this task. The reservoir size of both structures is fixed to 400 for simplicity. For the SCR network, the parameter that affects NRMSE hugely and is worth adjusting is the leaking rate. The leaking rate is the parameter that affects how much percent of the states in the previous time step are involved in updating the current state of the reservoir. The leaking rate for optimal performance varies in different tasks. In this task, the best leaking rate occurs when it is equal to 0.1, as shown in Fig. 3.6(a), the NEMSE at this point is 0.3641. For the eRNR network, the involved parameters are different from the ones in SCR. There are two parameters that affect the NRMSE: input scaling parameter γ and the time constant τ ($\tau = R_{int}C_{int}$). As illustrated in Fig. 3.6(b), the NRMSE will decrease as γ increases. However, NRMSE increase slightly as γ is over 0.7. From observation, the simulation time will increase as γ increases. Considering the cost of computation, the best γ lies in the range of (0.7,1). The time constant τ should lie in the range 2s to 5s for a smaller NRMSE, according to the results in Fig. 3.6(b). The smallest NRMSE is 0.3894 when γ is 1 and τ is 5s.

For both SCR and eRNR methods, the smallest NRMSE achieved by optimizing the parameters in each network respectively are very close. This is because the eRNR network is the hard-

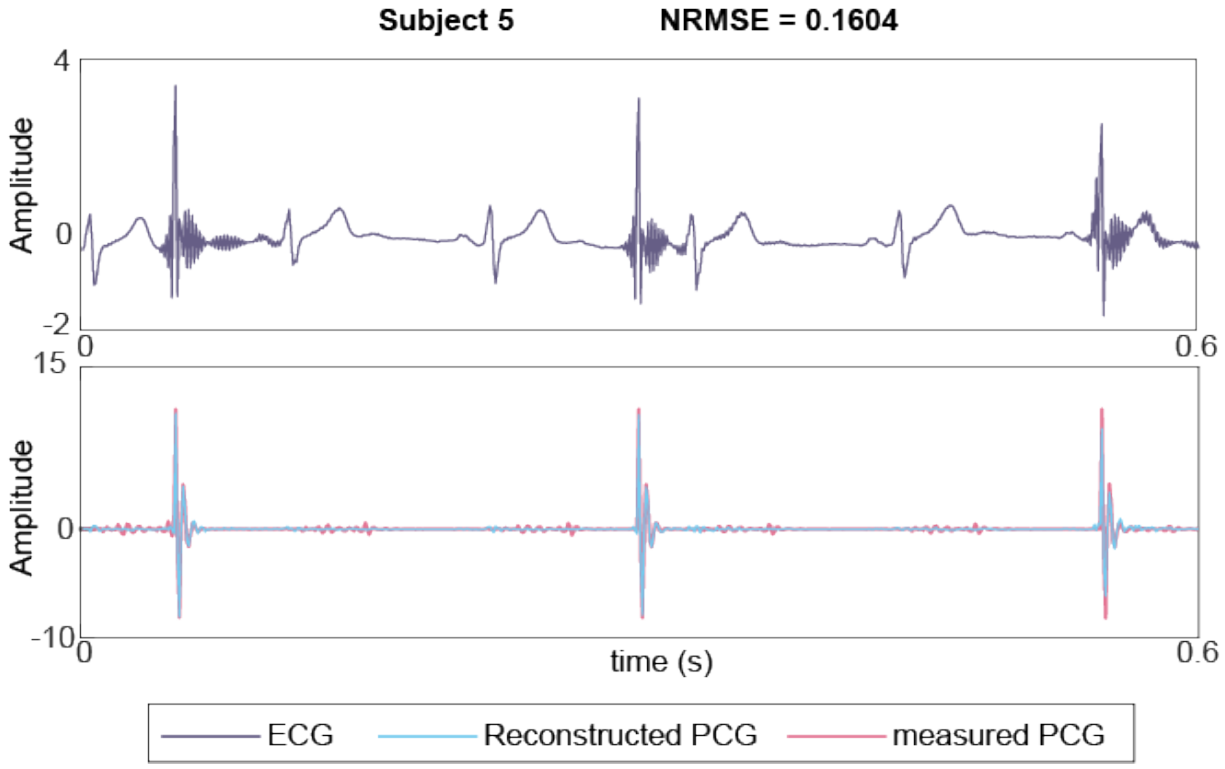

Figure 3.7: An example of prediction results. (Top) The input ECG for a time duration of 6s. (Bottom) The measured PCG and the reconstructed PCG data.

ware implementation of the SCR network. Although the parameters might change when it is implemented by hardware, the performance is similar.

3.3.2 Prediction error for ECG/PPG mapping

Included in the dataset, there are 9 repetitions for each of the 10 subjects. 6 of them were used for training and the remaining were used for testing. It was simulated in MATLAB with an electronic circuit built by Simulink. In the simulation, there are 400 neurons in total and other parameters are kept in accordance to the empirical value as proposed in [15]. The prediction error was also quantified by NRMSE. An example of the prediction results for subject a representative subject is shown in Fig. 3.7. The NRMSE for subject 5 is 0.1604 and the network is able to predict the reconstructed PCG from the input ECG data in an excellent way.

A histogram illustrating the distribution of NRMSE for each subject is shown in Fig. 3.8. The NRMSE for half of the subjects lies in the interval [0,0.3] and the other half distributes in [0.4, 0.7]. This could be a result of the collection of the dataset. The difference among the NRMSE for different subjects occurs because of the recording of the dataset was not recorded under the exact same conditions. In the future, the impact of varying data recording conditions for different subjects can be studied to eliminate the inter-subject variability. Overall, the network can predict spikes of PCG signals accurately for each subject. The NRMSE for each subject

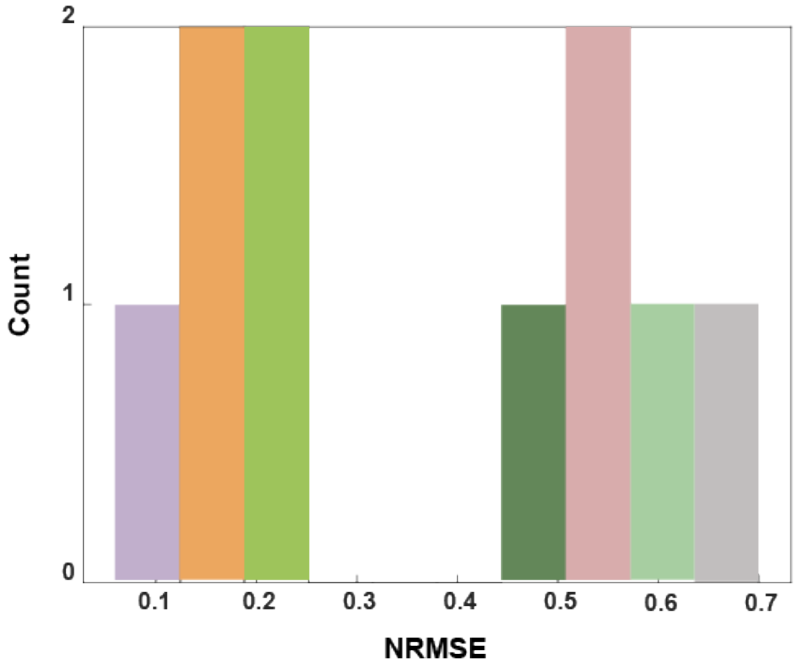

Figure 3.8: Histogram for the distribution of NRMSE for each subject.

Table 3.1: Experimental results for each subject.

Subject No.	1	2	3	4	5	6	7	8	9	10	Average
NRMSE	0.6927	0.6006	0.4704	0.1676	0.1604	0.2120	0.2023	0.5719	0.0647	0.5478	0.3690

varies, the highest NRMSE is 0.6927 for subject 1, and the lowest NRMSE is 0.0647 for subject 9, and their average is 0.3690. The experimental results for each subject are listed in Table 3.1.

3.4 Discussion and conclusion

This chapter presents two tasks using PRC in biomedical signal forecasting tasks, and the prediction errors are evaluated by NRMSE. It is notable that biomedical signals were collected and evaluated in an intra-subject format since the biomedical signals from individuals are inherently different. Results demonstrate the ability of RNR as a predictor compared with other deep learning methods. In addition, the lightweight algorithm of RC holds the potential for PRC to be embedded as a processing core for wearable devices in medical applications. As an initial trial in the field of biomedical signal processing, the predictive modelling tasks serve to advance the research conducted during my PhD to the next stage, enabling the exploration of more complex classification challenges.

Chapter 4

PRC for Heart Sound-based Biometric Identification

Heart sound signal has emerged as a promising solution to biometric identification. In this chapter, an optical stethoscope-based laser-camera system for biometric identification using PRC was introduced. As a bio-inspired algorithm, RC has attracted growing research interests in recent years. Unlike conventional machine learning classifiers, RNR is a hardware-based neuromorphic model that preserves the majority of computing in the analogue domain, holding the promise of a next-generation machine learning accelerator. The proposed system is verified by an experimentally collected heart sound dataset by laser-camera system achieving an overall accuracy of 89.03% in identifying twelve testing subjects. Additionally, the elevated heart sounds from 8 subjects have been blended with their normal heart sounds to assess the robustness of the proposed system. The classification accuracy reaches over 83% in this mixed test. The successful demonstration promises a novel application of physical RC for future biometric identification.

4.1 Introduction

4.1.1 PRC for heart sound biometric identification

The importance of human identification has grown over the past decades as more services have expanded into the digital domain. Currently, identification is commonly found in our daily lives, e.g., accessing private facilities or proving identity to third parties when accessing online services or physical locations. The shift from knowledge- (e.g., passwords) and ownership- (e.g., tokens) based authentication methods towards biological recognition technology has increasingly become desirable, considering the growing security risks tied to password reuse and security fatigue.

Biological recognition technology offers an alternative identification approach that relies on either physiological or behavioural features [139]. Over the past two decades, heart sounds have emerged as a novel physiological biometric capable of sidestepping issues arising in knowledge- and ownership-based authentication methods. Conventionally, heart sounds refer to cardiac sounds acquired using a PCG method. PCG is a record of acoustic vibrations of the heart, usually recorded in the chest area using a microphone [140]. This recording qualifies as a potential identity factor due to its universality, comparative uniqueness and acceptability [139].

An alternative method for recording cardiovascular data has been proposed [141] and applied to biometric identification [142]. It relies on an interference pattern, called speckle, that arises when coherent light interacts with a diffuse medium. As this medium moves, its changes are reflected in the changes of the speckle pattern; these, in turn, can be tracked to collect information about the vibrations of the material [143]. This ability to track vibrations forms the basis of using coherent light illumination to extract the heart sound information. The device, consisting of a camera and a laser, points at a patch of skin that vibrates as the heart contracts during its cycle. When a suitable location is chosen, most of the vibrations recorded by the device are due to cardiovascular activity.

Considering the uniqueness of individual hearts and the different skin conditions, such a signal possesses a uniqueness level that makes it suitable for biometric identification [144]. It compares favourably with other biometric identification technologies in several areas. Unlike radar-based systems that rely on single-subject environments [145], it can function in comparatively busy ones as long as its view of the subject is unobstructed. Similarly, it works in environments where noise levels make voice-based recognition impossible. It requires no contact with a sensor, unlike the systems based on hand recognition [146]. These advantages make the laser-camera system particularly suitable when contactless and continuous identity monitoring is required. For example, embedding the laser module in a smartphone and collecting data through the built-in camera.

Neural networks, as bio-inspired algorithms, have been widely applied for information processing tasks. For software-based neural networks relying on Von Neumann architecture, the analogue input information is converted into digital bits and calculations are performed in a digital system. In addition, in this system, processing and memory units are separated. Additional power is consumed for data transmission between these two blocks [147]. Besides, the scaling of transistors in the conventional computer, as noted by Moore's Law, is approaching its end and thus hinders the further development of Von Neumann-based computers [6, 20]. Neuromorphic computing, as a bio-inspired computing paradigm, is expected to break the Von Neumann bottleneck and solve machine learning problems with low-power, collocating memory and processing [148].

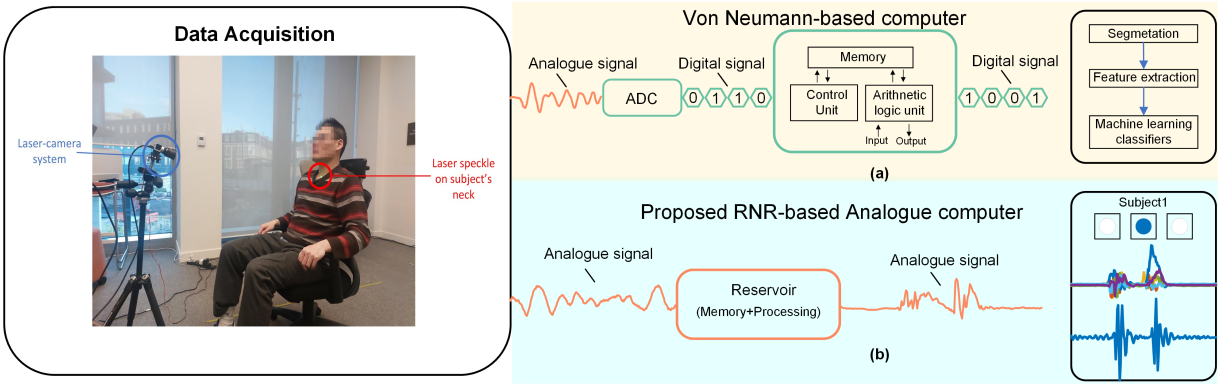

Figure 4.1: The experimental setup of heart sound data collection using laser and the comparison of the conventional Von Neumann-based machine learning classifier method and the hardware RC classification method. (a) In a Von Neumann-based computer, the processing and memory units are separated in the Von Neumann architecture, and additional power is consumed for data transmission. In terms of the classification algorithm, the signals experience data segmentation and feature extraction before they are sent into a classifier. (b) An electronic components-built reservoir in the hardware RC method realizes both memory and processing. The signals are kept analogue all the time. Therefore, for classification, a program that detects where the peak occurs can be applied to obtain the classification results.

RC is an RNN-based framework suitable for sequential signal processing owing to the inner connections between the current input with the past network states. Furthermore, this algorithm employs a reservoir layer to project the input data into a high-dimensional feature space, thereby enhancing linear separability. Consequently, only a single layer necessitates training, significantly reducing computational demands compared to DL-based approaches. Under the term neuromorphic, physical realizations of RC are promising to drive the next-generation machine learning hardware devices [11, 13, 149]. RNR was proposed recently as a potential hardware implementation of RC. Using rotating elements for implementing cyclic reservoirs, RNR fulfills a hardware-friendly analogue system [15]. Compared with the previously proposed peer work [29], which utilized a delay line to construct a delay-coupled reservoir, the reservoir layer of RNR is advantageous from several perspectives: 1) The absence of ADC and DAC modules reduces hardware cost. 2) Balance the MC and state richness. 3) Parallel computing owing to the unnecessity for time-multiplexing.

In this work, we introduced a fast PRC recognition processor called RNR constructed from optical stethoscope-based heart sound biometrics. Contrasting this processor with a conventional Von Neumann-oriented computer, a conceptual figure is illustrated in Fig. 4.1.

4.1.2 Impact statement

Biometric identification plays an important role in daily life. However, existing methods like fingerprints and facial biometrics are susceptible to being acquired and duplicated, resulting

in privacy breaches and online fraud. In contrast, heart sounds as biometrics are expected to be more secure as heart sounds are hard to imitate and abuse. While traditional neural networks require data sent to a workstation or cloud for training and analysis, this work proposes a PRC-based biometric identification system by applying the optical stethoscope retrieved heart sound signal. The PRC architecture, under the term neuromorphic computing, provides a fast hardware-based machine learning processor where computation is performed on devices at the edge and thus minimizes energy consumption. The proposed identification system was evaluated by experimentally collected datasets. It is promising that this system can be implemented in modern devices like smartphones and smartwatches through embedded laser modules to collect and retrieve the heart sound data and an RNR processor to function as the processing core.

The merits of this work are stated as follows:

- Heart sounds as biometrics are expected to be more secure as heart sounds are hard for imitation and information abuse. It raises fewer ethical and private concerns than the systems requiring recognisable images capturing [150]. Fingerprints and facial images are susceptible to being acquired and duplicated [117]. Especially, facial biometrics are prone to privacy breaches and online fraud due to the inherent risk of information leakage. Heart sound data for individuals is unique and hard to imitate and abuse.

- Laser-camera system for remote heart sound data collection through an optical stethoscope. We point an eye-safe laser at the neck of testing subjects while a CMOS camera is utilized to capture the reflected speckle patterns. The reflected speckle patterns are converted into video frames and the human heart sound is retrieved by calculating the optical flow from the video data. Compared to conventional stethoscope-based technology, the laser-camera-based method we proposed avoids skin contact. It can even be deployed in photo booths, allowing users to monitor their heart health at any time. Our proposed method also allows free movement of the limbs of testing participants during data recording. Moreover, it is very robust to the interference caused by other moving human targets because of the focused ability of laser sensing. In addition, an optical stethoscope has a much higher signal-to-noise ratio (SNR) than a traditional digital stethoscope.

- Collocating memory and processing units. As shown in Fig. 4.1(a), a conventional software-based neural network approach usually relies on a Von Neumann computer. Signals are first digitized into bit streams and then stored in a memory block and waiting for a calling from the control block to perform calculations in the arithmetic logic unit (ALU). In contrast, a physical RC processor is not simply executing instructions. The learning procedure takes place inside the reservoir, where memory and processing units are combined, and signals remain analogue, as illustrated in Fig. 4.1(b). Data is processed near the device and reduces the need for data transmission, thereby lowering bandwidth requirements and

improving response times.

- Reduction in computing complexity but with a satisfactory classification result. Machine learning classifiers require a complex classification training algorithm as well as segmentation and feature extraction techniques, where massive computational power is consumed. For an RC algorithm, the training is a simple linear regression. In addition, raw signals are directly fed into the reservoir, and no segmentation and feature extraction methods are needed.
- Combination of the optical stethoscope and the RNR algorithm. Optical stethoscope requires subsequent signal processing and machine learning, while RNR as a neuromorphic model can perform local data processing and machine learning, greatly reducing power consumption and enhancing data security, as there is no need to transmit to the cloud or server.

4.2 Experiment setup and data pre-processing

4.2.1 Dataset description

The experimental setup is illustrated in Fig. 4.1, where the laser-camera joint system is fixed on a 1.2 m tripod, and the camera is connected to a laptop via a USB cable for powering and data transferring. The laser diode (DJ532-40, Thorlabs) with a wavelength equal to 532nm is pointed to the neck of the participants, producing an illumination spot of approximately 5 mm diameter, whereas the CMOS camera (acA640-750um, Basler Optics) captures the reflected speckle pattern with an FPS equal to 1.47 kHz. The green laser diode has a distance of around 1 m from the participants, and the laser power exposed on human skin is below an eye-safe level ($\tilde{0}$.5 mW, safe for long-term eye/skin exposure). The focal length and f-stop of the camera objective are set as 25 mm and 0.95, respectively, allowing the camera system to detect the laser speckle from a very close (0.1 m) to a relatively far range (up to 3 m). Additionally, the size of region of interest (ROI) window is chosen as 128x128 pixels, and the camera exposure time is set as 600 µs.

The data was collected with 12 people (8 males, 4 females) aged from 22 to 32 in an office environment (5 m x 4 m office area) by the Extreme Light Group, School of Physics and Astronomy, at the University of Glasgow. The participants were asked to sit naturally on an office chair in front of the laser-camera system while the camera recorded the dynamic speckle patterns from their neck skin. For each of the 12 participants, we measured 10s heart sound 10 times using a Python-based graphical user interface (GUI). The GUI incorporates a data processing feature and is capable of offering real-time visualization of human heart sounds.

4.2.2 Signal pre-processing

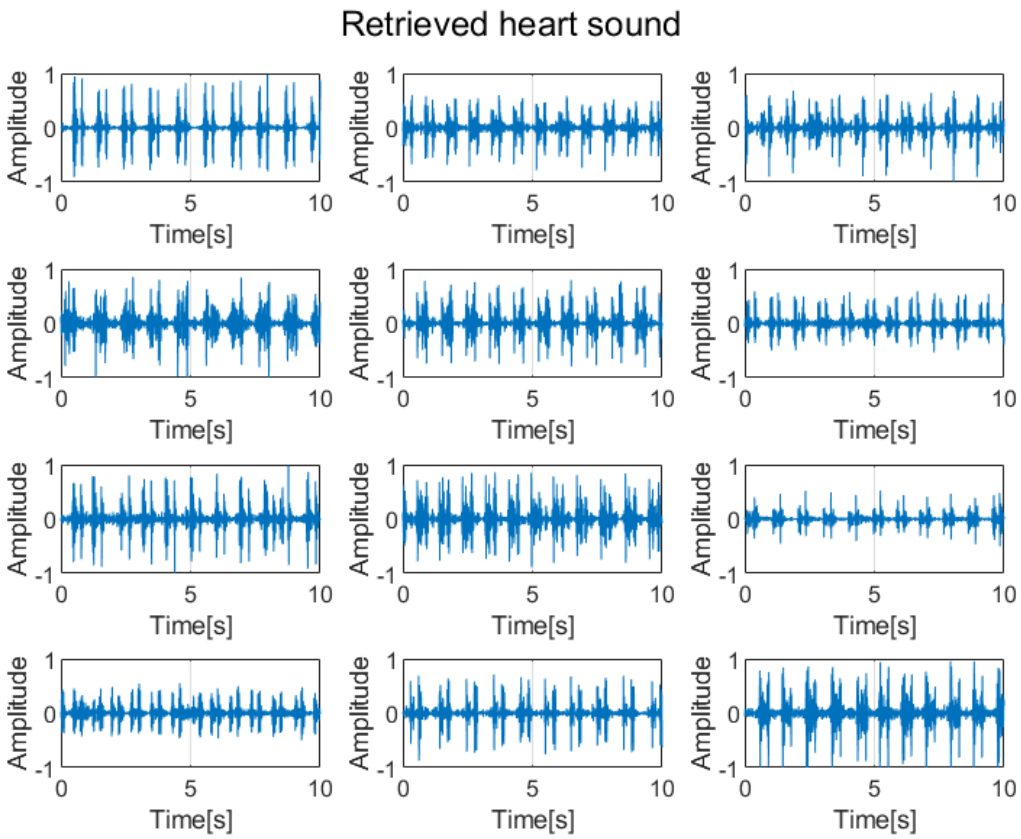

Figure 4.2: The retrieved heart sound of each testing subject.

.

The raw data is in the format of a video frame, for a 10s duration, the video frame size is 128x128x14700. In order to retrieve the heart sound from video data, an optical flow-based algorithm called Farenback [138, 141, 151] is utilized for estimating the directions and speed of reflected laser speckles. However, subtle movements during the measurement, such as tilting of the neck skin, will influence the flow of the interference pattern [141]. Therefore, after the computation of optical flow, a bandpass filter with a cut-off frequency equal to 20 and 700 Hz is applied to remove the unwanted frequency components due to skin or neck movement.

The filtered signal of each participant is illustrated in Fig. 4.2. It is clear that every participant has two peaks within one cycle and they are separated in a good manner. The first peak is known as S1 and the second peak is known as S2. S1 is generated by the closure of the mitral and tricuspid at the beginning of heart systole, whereas S2 is generated by the closure of aortic and pulmonary valves and it represents an end of heart systole.

The original dataset should be resampled by the resampling rate of 147 Hz. This is because of the memory capacity limits in RNR. Due to the short-term memory property, the network

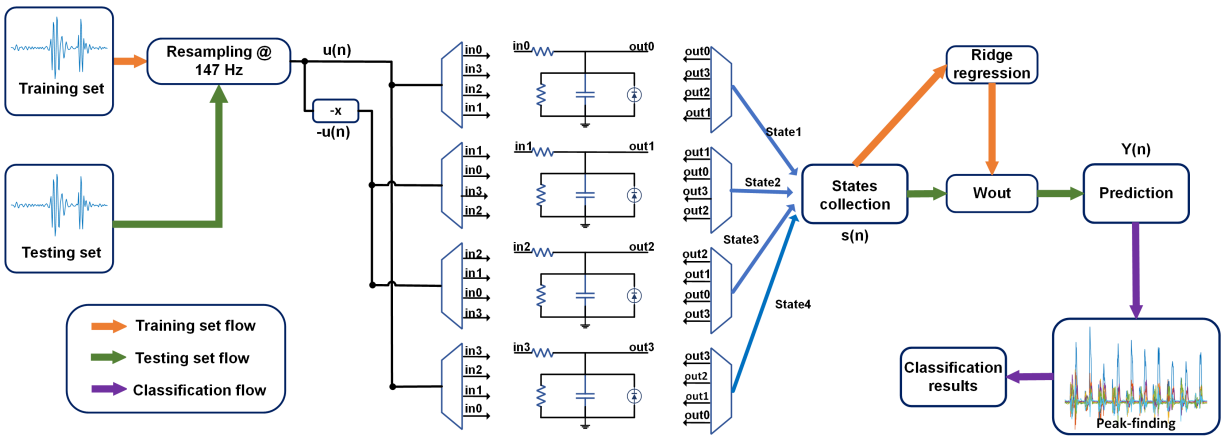

Figure 4.3: (a) The network description of the processing procedures with an example of a 4-neuron reservoir. In practice, the number of neurons can be increased. The original data is resampled at 147Hz and goes into an input weight matrix implemented by randomly chosen positive and negative signal sources. For training, the states are collected by applying ridge regression to calculate the output weight matrix W_{out}. For testing, predictions are obtained by the multiplication of the states s(n) and W_{out}. A peak-finding method is used to find the predicted label.

cannot remember a long-term signal. Downsampling the signal can shorten the length of the signal, which means data points inside the same time duration are fewer, thus the network can remember the information of an entire heartbeat event more easily.

4.3 Network design

4.3.1 Parallel reservoirs

In this work, RNR is used to fulfil biometric identification by using heart sound signals. As a hardware architecture, RNR is different from traditional Von Neumann-based architectures where the processing unit and storage unit are separated. In RNR, the information is processed immediately without being stored. Therefore, a large amount of power consumed for transmission between the process and storage units under conventional digital computers can be saved. Furthermore, the data flow for RNR does not contain any feature extraction techniques that bring additional costs in signal processing. The raw signal directly goes into the RNR processor and performs the prediction, and the majority of computing occurs in the analogue domain. Such a hardware RNR system can handle biometric identification tasks faster and more power-efficient.

We hereby compare RC with two related machine learning algorithms, SVM and long short-term memory (LSTM). RC shares a similar computing method with SVM, both apply a trick to map the input to a high-dimensional feature space to allow for linear separability. The merit of RC compared with SVM in processing temporal data is that it adds the temporal feature to the projection through its state-updating equation. While the state-updating equation endows

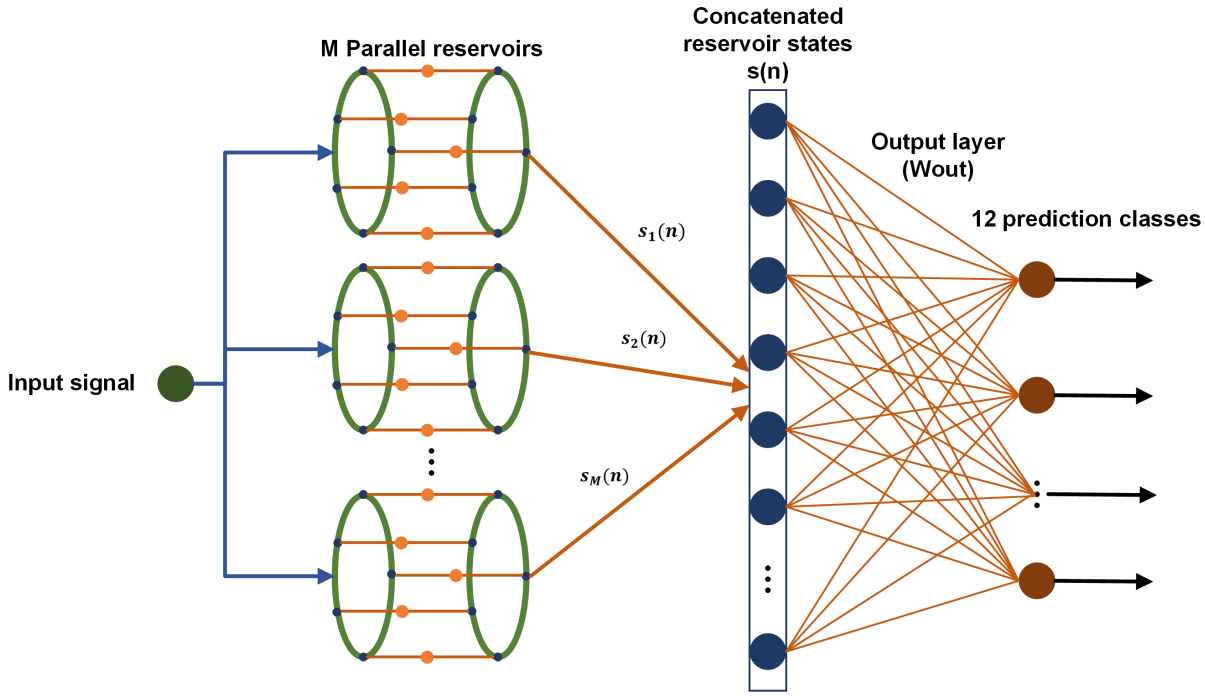

Figure 4.4: An example of M parallel reservoir topology with 12 prediction classes.

RC with a recurrent nature, RC does not need to deal with complex gradient problems like other RNNs do. In this case, RC could stand out and bring high efficiency in temporal signal processing. Therefore, reservoir computing is expected to show excellent performance in terms of distinguishing different people according to their unique heart sound signals effectively.

As discussed in Section 2.2, a classical reservoir network contains three layers: an input layer, a reservoir layer and an output layer. The circuit of the network as shown in Fig. 4.3 is simulated in MATLAB R2023a and Simulink. The neuron circuit was modelled in Simulink by building blocks and modules. The discrete states generated in Simulink were subsequently transferred to the MATLAB workspace for further analysis. Additionally, the pre- and post-neuron rotors were modelled by continuously shifting $W_{in}u(k)$ and collected states from Simulink.

However, a 12-people identification task relies on a large network size to provide rich reservoir states. Increasing network size will lead to extensive simulation time. As one of the primitives in neuromorphic computing, parallel processing is a crucial element that enables the brain to rapidly perform computations [20]. To speed up computation time and provide rich state dynamics [15, 44] at the same time, we adopted a parallel reservoir computing topology with multiple reservoirs (M) in parallel and each reservoir size (N) of 500. The topology of the parallel reservoir is shown in Fig. 4.4.

4.3.2 Training and Regression

Different from the traditional classifiers that introduce a segmentation and feature extraction method, RC processes the complete raw signal. Therefore, the labelling of the dataset should be treated differently from the traditional ones. The labelling of the dataset was done manually, which means there will be artificial errors in labelling because one cannot tell where each heartbeat exactly ends, while in traditional classifiers, this can be avoided by limiting the signal in a certain segmentation. Therefore, the dataset should be carefully labelled by certain techniques. In our work, the dataset is manually labelled in a continuous and point-by-point fashion, which is similar to prior works [16, 18, 22]. Usually, at the end of each heartbeat, there is a '1' to declare the corresponding subject and a '0' for the rest of the label set. However, for the purpose of facilitating the classification and avoiding artificial errors, the number of '1's is expanded to their multiple neighbouring positions as shown in Fig. 4.5 and 4.6. The reason for this is that the longer length of the label could make the network less sensitive to the location where the heartbeat ends, because the end of each heartbeat can only be roughly estimated and labelled which is less likely to be identical.

The reservoir states of each parallel reservoir will be concatenated to build $s(n)$ which will be collected for training the output layer. In RC, the most common regression method is called ridge regression. The output weights can be calculated by equation (2.7). Finally, the prediction of the classification is obtained by the multiplication of the output weight matrix W_{out} and the state of the reservoir at time n, written as equation (2.4).

In recent studies, the output readout layer can be expected to be realized by a memristor crossbar array to keep the signals analogue [18, 84]. Hence, the output prediction is in the form of a continuous analogue signal as shown in Fig. 4.5 and Fig. 4.6.

4.3.3 Classification

There are two steps for classification: training and testing. 70% of the dataset constructs the training set and the rest is for the testing set. A sample of classification results is shown in Fig. 4.5 and Fig. 4.6. In the method presented in our work, the decision of which subject the heart sound signal belongs to is made during each interval in the testing procedure by a peak-finding operation [15, 18]. The input channel is a one-dimensional heart sound signal while the output contains 12 channels that represent the possibilities for the 12 subjects as shown in Fig. 4.4. Ideally, the other 11 channels remain zero while the corresponding correct prediction appears as one. Therefore, the target class can be easily decided according to where the highest peak occurs.

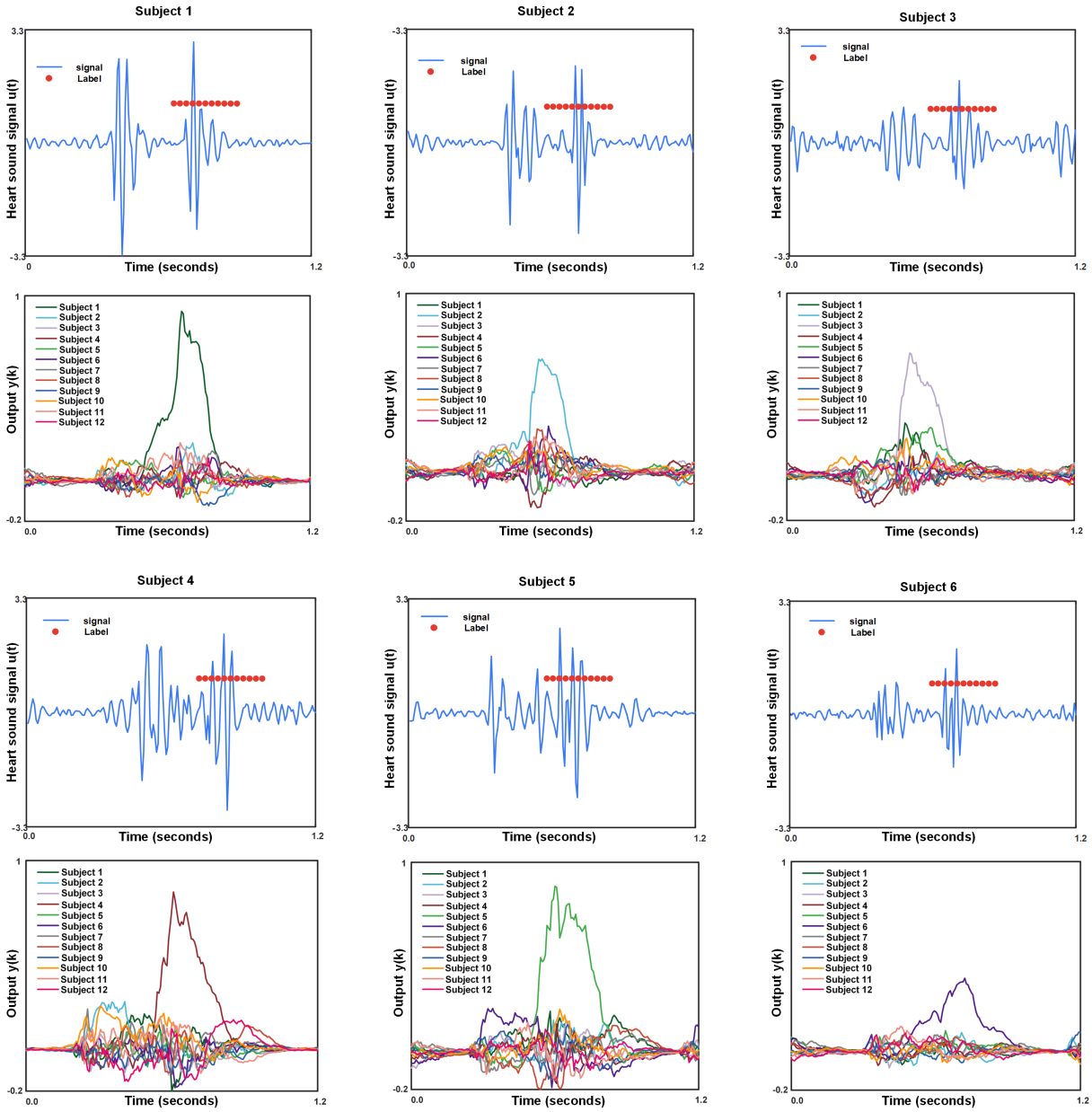

Figure 4.5: Examples of labelling and output prediction of each heart sound for subject 1 to subject 6. Continued on Fig. 4.6 in the next page.

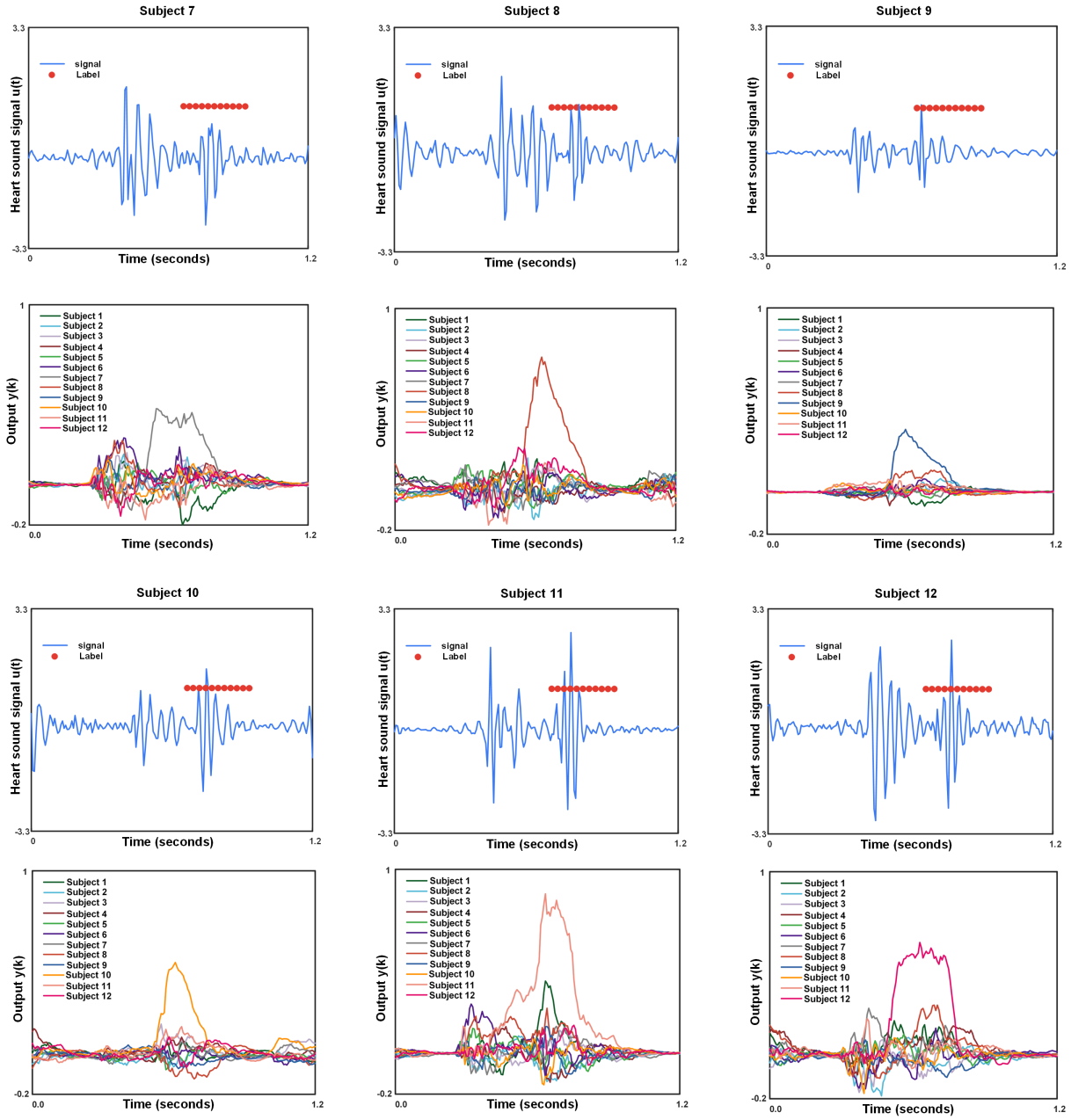

Figure 4.6: Examples of labelling and output prediction of each heart sound data for subject 7 to subject 12. The magnitude for each label is about 0.27 seconds. The data is manually labelled in a continuous and point-by-point fashion. At the end of each heartbeat, a set of '1's, which is represented by the red dots in the figure, is used to declare the corresponding subject, while the rest of the label set is kept '0'. In the output, there will be 12 channels representing the prediction results. The predicted label decision is made where the peak occurs.

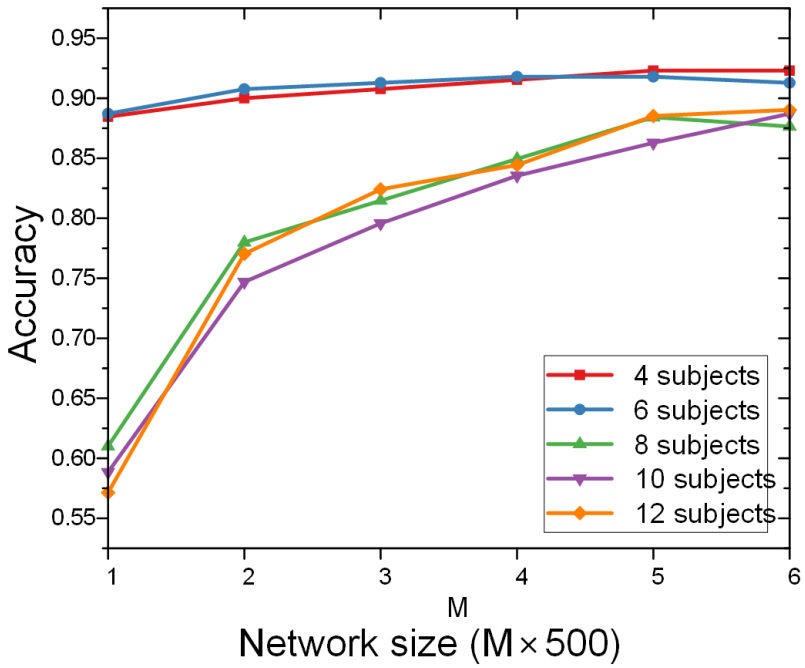

Figure 4.7: The relationship between the accuracy and the reservoir size for different numbers of subjects involved.

4.4 Performance evaluation

4.4.1 Parameter optimization

The parameters in RNR can affect classification accuracy significantly. The three dominant parameters are the time constant τ, reservoir size M×N and the input range.

The time constant τ is a parameter that affects the performance of the dynamic neuron. As a neuromorphic computing system, the time constant for RNR is biologically realistic and the time constant is usually greater than the millisecond scale. In RNR, the dynamic neuron is realized by a leaky-integration-ReLU circuit. The circuit defines τ by the following equation:

$$\tau = R_{int} C_{int} \tag{4.1}$$

where R_{int} is the integral resistance and C_{int} is the integral capacitance. τ_r is the rate of rotation, which should be consistent with multiplexers. For example, $\tau_r = \tau/8$ for 8-switches multiplexers and $\tau_r = \tau/4$ for 4-switches multiplexers. The choice of the value for τ and τ_r should be matched to make the system function properly. In this work, we choose the empirical value of $\tau = 1s$ and $\tau_r = \tau/8$ fixed for the following parameter optimization.

The reservoir size is made up of M parallel reservoirs, and each reservoir size of N. It is a parameter that defines how many neurons are included in the reservoir. A small reservoir size could

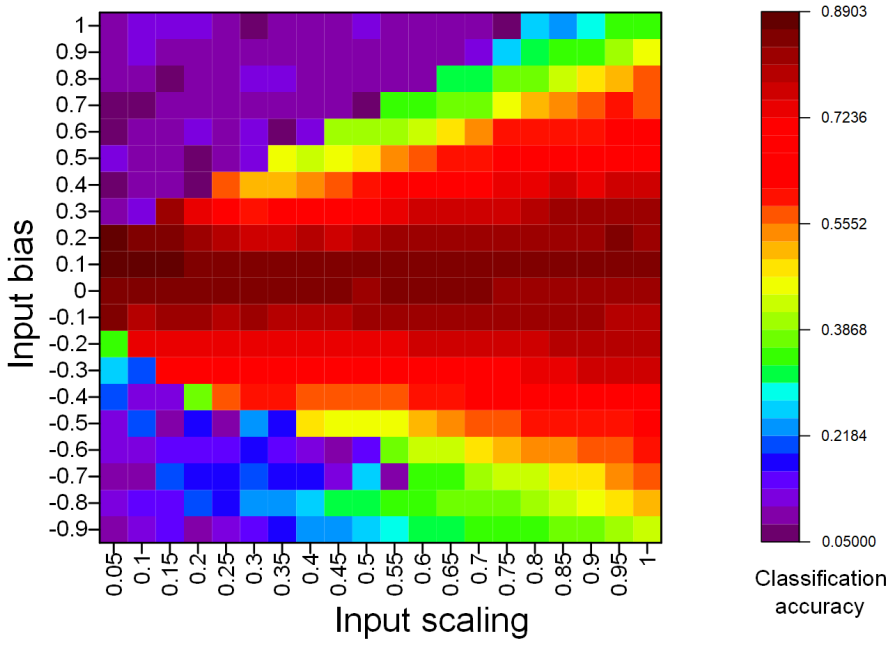

Figure 4.8: The heatmap for parameter adjustment. The two parameters, input scaling and input bias define the input range of the injected signals.

not be sufficient for solving a complex task while a large reservoir size could bring complexity to the hardware and not necessarily increase the network performance. We provided an analysis of the comparison of the accuracy when the number of people is increased, as illustrated in Fig. 4.7. For a 4-people or 6-people classification task, a network size of 1 $\times$500 is enough to obtain an accuracy of around 90%. However, the accuracy will dramatically decrease when the number of people exceeds 8. Adding multiple parallel reservoirs could efficiently enhance identification performance. When the network size is approaching 5$\times$500, the accuracy for 12-people identification is over 88%. Therefore, considering the cost of the hardware, we finally chose the reservoir size to be 6$\times$500.

The parameter that defines the input range is the input scaling parameter γ and the bias. Restricted to the circuit, the input range cannot exceed the supply voltage, for example, 3.3V. Therefore, the range of the parameter γ we consider is in the interval (0,1). In this case, the adjustment of the data input range is within (-3.3V, 3.3V). From the heatmap, the best classification result is 89.03% and it is obtained when γ is 0.05 and the bias is 0.1, as demonstrated on the heatmap in Fig. 4.8. At this time, the data input range is (-0.21, 0.19).

A confusion matrix to see the accuracy for each class is shown in Fig. 4.9. The error mainly occurred when predicting the 3rd subject. The 3rd subject is occasionally recognized as the 8th subject and the 10th subject. The possible reasons for the failure cases could lie in the manual labelling of the dataset. Although we extended label length to make the network less sensitive to where a heartbeat ends as discussed in Section 4.3.2, artificial errors still exist. The information

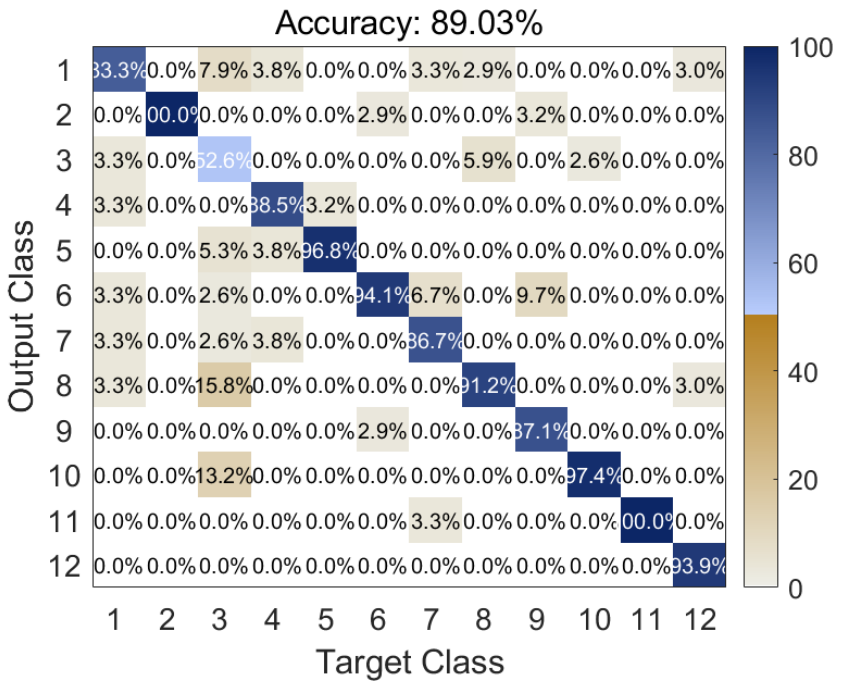

Figure 4.9: The confusion matrix for the accuracy of each subject.

of one signal remembered by the network may be similar to another different signal remembered by the network due to the label position change brought by manual labelling, resulting in indistinguishable classes attributed to higher acceptance rates. One solution to address this issue in practical applications could involve utilizing a longer analysis window encompassing multiple heartbeats. By making an overall prediction of the class that appears most frequently, as discussed in Section 4.5, the accuracy and robustness of the classification could potentially be improved.

4.4.2 Noise analysis

In addition to analysing the tuning of the reservoir's internal parameters, we conducted a comprehensive evaluation of potential noise factors encountered during signal acquisition and the hardware implementation of the network, to provide a more thorough understanding of the method's robustness and practical applicability.

In terms of signal acquisition, two primary sources of noise were identified: (1) subtle movements of participants' skin and neck, and (2) noise originating from the acquisition equipment. The first source of noise, due to minor movements, can be mitigated using a bandpass filter between 20 Hz and 700 Hz, effectively removing unwanted frequencies as discussed in Section 4.2.1. The second source of noise stems from the laser and signal acquisition equipment, with noise levels influenced by the laser's power and the CMOS camera's performance, both of which were standardized across all measurements in our study. However, to assess the po-

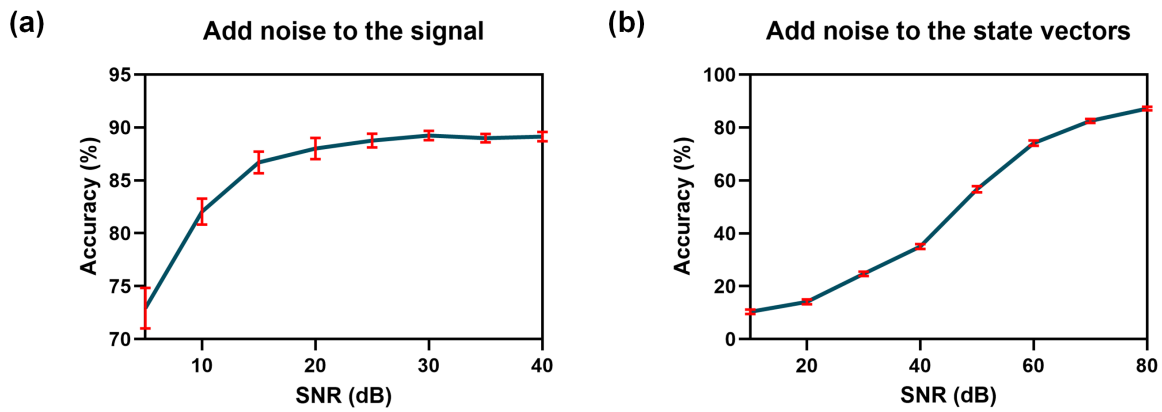

Figure 4.10: The effects of different noise levels on the classification accuracy by artificially adding Gaussian white noise to (a) the collected heart sound signals (b) the state vectors.

tential impact of noise for future applications that may use different acquisition equipment, we conducted a noise analysis by artificially introducing Gaussian white noise to the collected heart sound signals. This approach allowed us to evaluate how varying noise levels affect classification accuracy, thereby offering insights into the robustness of the method under different noise conditions.

As shown in the simulation results in Fig. 4.10(a), the system maintains high classification accuracy with SNR above 20 dB, which is achievable with commercial laser and data acquisition equipment. This possible reason is likely due to the neural network's inherent robustness to moderate noise levels. These findings suggest that the system can effectively tolerate noise incurred during signal acquisition, thereby supporting its potential reliability in practical applications.

Another source of noise incurred during the hardware implementation of the RNR system. A typical challenging problem in analogue computing is the D2D and C2C variability and noise for hardware implementations. The design of the system in our study was based on the use of commercially available components, such as resistors, capacitors, diodes, and multiplexers. These components are characterized by excellent D2D and C2C consistency, along with minimal noise levels. While recent research in analog computing has focused extensively on novel, emerging electronic devices that leverage intrinsic memory properties and nonlinear dynamics to generate transient states, D2D and C2C variability of these novel devices should be carefully considered. Nevertheless, the variability in D2D could enhance the state richness and thus improve the performance of the network as revealed by some research, while the C2C variability should be avoided [12, 86].

To assess the impact of noise on classification accuracy, Gaussian white noise was introduced to the collected state vectors to evaluate the SNR required for a realistic hardware implementation of the RNR system. As illustrated in Fig. 4.10(b), the network requires an SNR of at least 70 dB to attain a classification accuracy exceeding 80%, indicating a careful hardware design

in constructing neuron circuits and collecting state vectors. Enhancing classification accuracy could be achieved by increasing the SNR within the circuit or expanding the network size.

4.4.3 Memory capacity

The short term memory (STM) denotes the ability of the network to remember past information and MC is a quantitative concept that measures this ability. To define the MC, Jaeger[152] proposed the equation below:

$$M(k) = \frac{cov^2(u(t-k), y(t))}{\sigma^2(u(t)\sigma^2(y(t))} \tag{4.2}$$

This equation indicates the k-delay STM capacity, which means how much information from k time steps ago can be recovered by the reservoir[153]. cov is the covariance and σ is the standard deviation. Usually, $M(k)$ is in the scope [0,1] and it will decrease as the reservoir needs to reconstruct the information from farther past. MC is the summation of all the $M(k)$ from $k = 1$ to infinity, as denoted by the equation below:

$$MC = \sum_{k=1}^{\infty} M(k) \tag{4.3}$$

MC is affected by many factors: reservoir size and parameters τ and γ in the reservoir. For CR and RNR, as they are theoretically equivalent, the MC for them are almost the same when the parameters are matched [15].

A higher MC does not promise better performance. Redundant information might be remembered and affect the network's decision of the correct result. In our work, the MC is 68.91 when the best classification result is obtained with parameter $\tau = 1s$, $\gamma = 0.05$, $bias = 0.1$.

4.4.4 Comparison with consistent software network

To observe the consistency between RNR and the software RC, including the classical reservoir and CR, a performance matrix is used for comparison. The comparison can be seen in Fig. 4.11. The performance matrix is composed of Acc, Se, Sp, precision and F1 score. The calculation of these values involves true positive (TP), ture negative (TN), false positive (FP) and false negative (FN), denoted by the following equations:

$$Acc = \frac{TP + TN}{TP + TN + FN + FP} \tag{4.4}$$

$$Se = \frac{TP}{TP+FN} \tag{4.5}$$

$$Sp = \frac{TN}{TN+FP} \tag{4.6}$$

$$Precision = \frac{TP}{TP+FP} \tag{4.7}$$

$$F1 = \frac{2T}{2TP+FP+FN} \tag{4.8}$$

For RNR, the input matrix W_{in} is realized by using a switch to randomly select a positive or negative signal source. Therefore, the corresponding W_{in} in CR should be binary and composed of randomly chosen -1 and 1. However, from our observation, CR with a matrix of randomly chosen -1 and 1 has a decline in performance compared with any other networks. It is surprising that the RNR can obtain a higher performance than the corresponding software network CR. The possible reasons for RNR and classical reservoir showing a better performance could lie in two perspectives: 1) From the masking type perspective. Compared with the classical reservoir whose weights in W_{in} are randomly chosen numbers from the uniform distribution [-1,1], the binary W_{in} in CR could not provide complex dynamics of the response and result in a performance reduction [154]. 2) From the dynamic neuron perspective. Although the topologies for CR and RNR are mathematically equivalent, the realizations of the neurons for the two are slightly different. RNR applies a Leaky-Integrate circuit and a rectifying diode to implement the neuron. The Leaky-integrate circuit could result in parameter mismatches. In addition, the activation ReLU function for RNR and CR are different. When the optimal parameter for RNR is obtained after fine-tuning, the input data fed into the neuron is in the nonlinear region of the diode, which is different from the ideal ON/OFF form of software-implemented ReLU. This could enhance the state representation for RNR [15] and hence improve the performance.

Compared with the optimal software RC networks, RNR only shows a slight reduction in the performance with $Acc = 89.03\%$, $Se = 89.3\%$, $Sp = 99\%$, $Precision = 89.31\%$, $F1score = 88.87\%$. Compared with the state-of-the-art models, specifically an SVM classifier employing 16 time-domain features with a polynomial kernel and a bidirectional long short-term memory (Bi-LSTM) network with 120 hidden neurons and learning rate set to 0.01, batch size set to 16, which achieved a classification accuracy of 85.42% for SVM and 90.56% for LSTM after 200 epochs of training respectively. The RNR model demonstrates a slight decrease in accuracy compared with the Bi-LSTM network. This result shows the feasibility of applying a fully hardware RNR processor for heart sound biometric identification.

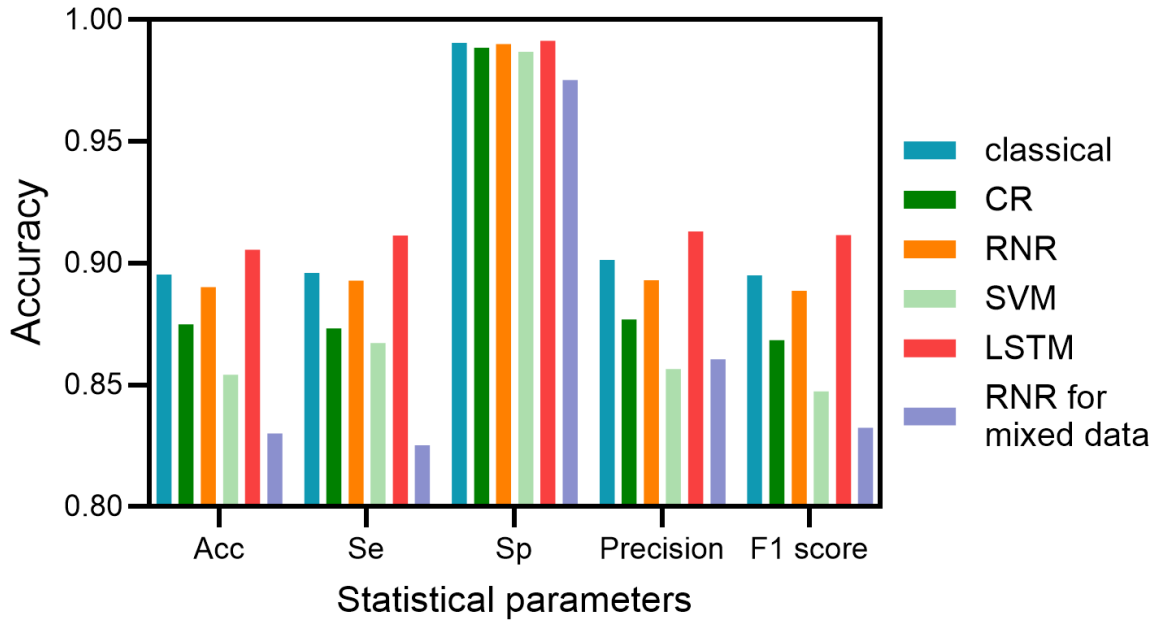

Figure 4.11: The bar chart of the comparison in terms of performance matrix for the five networks and RNR for the mixed elevated and normal dataset.

4.4.5 Power analysis

Although this study is based on simulation, the power consumption can be estimated by referencing the work conducted by Liang et al. [15], where the system is implemented through PCB circuits and memristor crossbar arrays.

The total power consumption P can be expressed by equation (4.9):

$$P = P_c + (P_s + P_t + \frac{E_c^{dyn} + E_t^{dyn}}{\tau_r}) \times M + \frac{E_m^{dyn}}{\tau_r} \tag{4.9}$$

where P_s is the driving power source, P_c and P_t are the static power of the counter and transmission dates, respectively. E_c^{dyn} and E_t^{dyn} represent the dynamic energy dissipated in the transition region driven by the rate of rotation τ_r. E_m^{dyn} is the energy consumed in the output layer (memristor array) for one inference. According to [15], $P_s = 3.27\mu W$, $P_c = 0.93\mu W$, and $P_t = 0.70\mu W$, regardless of how fast the rotors are operating. $E_c^{dyn} = 0.31pJ$ and $E_t^{dyn} = 0.07pJ$ are related the the rotor speed. The number of neurons is scaled from 8×8 to 500×6, therefore, a scaling parameter of 46.88 was multiplied by M. The dynamic energy of the memristor array is $E_m^{dyn} = 463.36pJ/class$. In addition, in the experiment conducted in this thesis, the rotor speed τ_r is 0.125s. Therefore, a power consumption of $230.60\mu W$ can be estimated for the overall system based on the parameter settings in this experiment. In comparison to a baseline DL system running on a low-cost device, such as a Jetson Nano that has a power around $5W$ to $10W$, the

power for the RNR is much lower.

4.5 Evaluation of the identification performance under after-exercise condition

In a practical situation, a subject may encounter different scenarios leading to changes in heart sounds. For example, a subject may use the identification system after running. Therefore, to evaluate the performance of the identification system under the after-excise situation, we collect heart sound data from eight individuals (4 males and 4 females) following a two-minute jumping jack exercise. An example of the comparison between the elevated heart sound signals and normal heart sound signals is depicted in Fig. 4.12(a).

It is evident that the main difference between an elevated heart sound and a normal heart sound is the heart rate. In this case, we could not train all the samples together as the normal heart sound signals and elevated heart sound signals are naturally different. Therefore, to identify the mixed elevated and normal heart sound signals, we trained two reservoir systems, one for normal heart sound and one for elevated heart sound. The testing data will first be confirmed to be normal or elevated according to the heart rate (heart rate over 100 to be elevated and heart rate below 100 to be normal), and then it will be delivered to the corresponding reservoir system for prediction. The flow chart of the procedures is illustrated in Fig. 4.12(c). The accuracy of the identification for 8 people is 83.01% with a reservoir size of 1000 (M=2, N =500). A confusion matrix for the result is shown in Fig. 4.12(b). The accuracy decreases compared with a normal-condition-only system. The possible reason behind this is that the differences between each elevated heartbeat are large as our identification system is based on only one heartbeat identification. In practical application, as the cardiac rhythm is continuous, enhanced accuracy can be attained by employing the most frequently occurring prediction within a given temporal window. As illustrated in Fig. 4.12(d), we employ a 4-second window with 7 heartbeats included. In this case, the heart rate is 105 which belongs to the elevated category. 6 of the heartbeats are predicted to belong to subject 4 and 1 of the heartbeats is predicted to be subject 5. In this case, an overall prediction of subject 4 is made as it appears the most frequently.

4.6 Discussion and Conclusion

In this chapter, we introduced a novel biometric identification method by retrieving optical stethoscope-based human heart sound signal through a laser camera system and processing it based on a PRC architecture called RNR for the first time. Through parameters optimization, the classification accuracy for a 12-subject identification can reach 89.03%. We also evaluated the identification accuracy of people after exercise and achieved an accuracy of 83.01% for

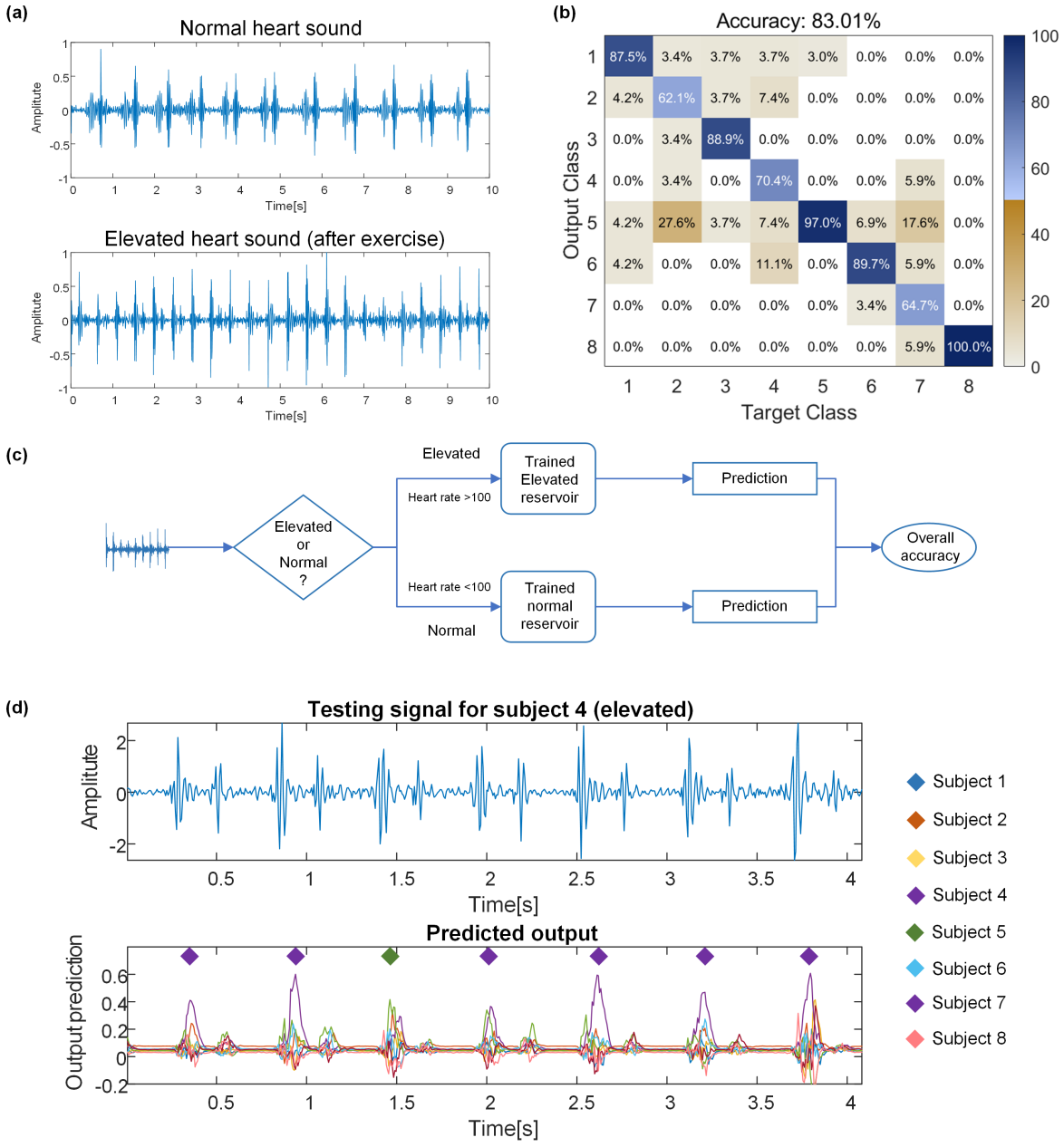

Figure 4.12: (a) An example of normal heart sound and elevated heart sound. (b) A confusion matrix for biometric identification under mixed normal heart sound and elevated heart sound dataset. (c) A flow chart for identifying mixed elevated and normal heart sound data. (d) An example of employing a 4-second window and making an overall prediction of subject 4 as it appears most frequently. The testing signal belongs to the elevated category as the heart rate of the testing signal is 105.

mixed elevated and normal heart sound datasets. It is noticeable that in our work, the accuracy of the prediction results is for one single heartbeat. However, the cardiac rhythm is continuous. In practical industrial scenarios, real-time identification can process a sequence of heartbeats and repeat the identification procedures. Enhanced accuracy can be attained by employing the most frequently occurring prediction within a given temporal window. Under the concept of neuromorphic, this model processes the raw heart sound signals in the analogue domain with in-memory computing, thus reducing massive power consumption. So far, our work takes the first insight into the feasibility of a PRC system in terms of heart sound biometric identification. It is promising that this technology could be implemented in modern devices like smartphones and smartwatches through embedded laser modules to collect and retrieve the heart sound data and RNR processor to function as the processing core.

Full development of such an identification system remains a topic for future exploration. First, there is only a small size of samples included in this experiment, a larger and more functional public dataset is expected to be constructed and analyzed to further refine the system applicable under various conditions and potential outliers. Second, with a network size of 3000 neurons, integration of the system into chip level is desired, which requires interdisciplinary research with other fields. Evaluation could be enhanced with long-term/longitudinal test data and robustness checks (e.g. posture variation, background noise).

Chapter 5

Event-Driven RNR for sEMG-based Gesture Recognition

Wearable health devices have a strong demand in real-time biomedical signal processing. However, traditional methods often require data transmission to a centralized processing unit with substantial computational resources after collecting it from edge devices. Neuromorphic computing is an emerging field that seeks to design specialized hardware for computing systems inspired by the structure, function, and dynamics of the human brain, offering significant advantages in latency and power consumption. This chapter explores a novel neuromorphic implementation approach for gesture recognition by extracting spatiotemporal spiking information from sEMG data in an event-driven manner. At the same time, the network was designed by implementing a simple-structured and hardware-friendly PRC framework called RNR within the domain of SNN. The spiking rotating neuron reservoir (sRNR) is promising to pipeline an innovative solution to compact embedded wearable systems, enabling low-latency, real-time processing directly at the sensor level. The proposed system was validated by an open-access large-scale sEMG database and achieved an average classification accuracy of 74.6% and 80.3% using a classical machine learning classifier and a delta learning rule algorithm, respectively. While the delta learning rule could be fully spiking and implementable on neuromorphic chips, the proposed gesture recognition system demonstrates the potential for near-sensor low-latency processing.

5.1 Introduction

5.1.1 Event-driven Implementation for sEMG-based Gesture Recognition

The sEMG is a non-invasive technique that reflects the electrical activities of skeletal muscle movements, providing insights into muscle cells essential for diagnosis, rehabilitation and other

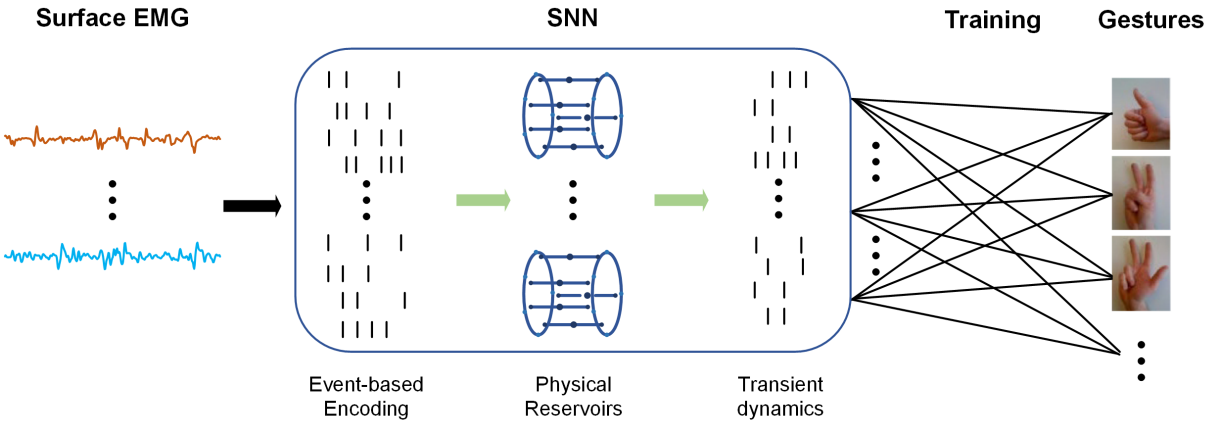

Figure 5.1: Architecture of the proposed classification system. The raw sEMG signals are encoded into SNN-compatible spike trains by an event-based encoding scheme. An SNN consisting of physical reservoirs is used to generate transient responses to a higher dimensional feature space. The collected dynamical states are trained in the readout layer only by machine learning algorithms for classifying gestures.

medical applications. sEMG-based gesture classification is a specialized application that focuses on recognizing motor intentions by analyzing the electrical activity of muscles. This field enables applications involving wearable devices in areas such as human-computer interaction, prosthetics control and robotics [57, 120]. Processing data locally on the wearable device or nearby, rather than sending data to a centralized server is crucial to perform real-time analysis, save bandwidth, improve privacy and save battery. To further improve local processing, it is crucial to identify lightweight gesture recognition algorithms. Traditional machine learning approaches, such as, SVM, k-NN, LDA, and RF require feature extractions from both the time domain and frequency domain [105]. Deep learning models have recently achieved higher classification accuracy on raw sEMG signals by automatically learning features [155, 156]. However, this comes at the cost of increased power consumption and memory requirements.

Neuromorphic computing, which prioritizes low latency and energy efficiency, is another promising approach for processing temporal signals at the edge [1, 7, 157, 158]. Among neuromorphic systems, PRC has emerged as a compelling solution. PRC harnesses the intrinsic dynamics of physical systems with collocated memory and processing units, distinct from traditional Von Neumann architecture, to accelerate and reduce the memory requirements of machine learning computations. This approach aligns well with the demands of edge computing, where information processing is performed closer to sensors, minimizing the latency associated with data transmission [11, 12, 159]. At the same time, RC is a form of RNNs used primarily for processing time-series data, but what makes it different is the use of a fixed, untrained network called a "reservoir" that projects input data into a high-dimensional space. In this high-dimensional space, complex temporal patterns can be more easily processed with training readout layer only to avoid costly operations for backpropogation associated DNNs.

RNR is a novel PRC paradigm characterized by its simple, hardware-compatible architecture, realized through an analog electronic circuit [15]. This architecture has demonstrated its efficacy in various temporal signal forecasting applications [27, 28]. For the first time we propose an SNN based RNR framework, more biologically plausible than traditional ANNs based on sigmoid units or ReLU [32, 160, 161]. This incorporation offers three key benefits: (i) A spike is a single-bit event, either a '1' or a '0', which is more hardware-friendly than floating point values. In comparison, the eRNR [15] where signals are processed in the analog domain. However, floating point values precision in analog circuits is costly in terms of complexity and hardware costs [20], hence, processing information in the form of low-precision spike trains could be a possible solution. (ii) event-driven processing allows for energy efficiency and low latency as the neurons only respond when an event occurs, leading to sparse vectors/tensors that are cheap to store and low-power to move [162]. (iii) This fusion still retains the advantages of RNR's simple and hardware-friendly structure with training readout layer only. At the same time, the low-precision reservoir states enables the use of trainable classifiers to further improve classification performance.

Encoding sEMG signals into spatiotemporal spiking information plays an important role in executing SNNs. A common signal-to-spike encoding technique is the delta-modulator analog-to-digital converter [57, 82, 163]. However, the classification performance by using this strategy lags behind the state-of-the-art, especially compared with deep learning methods. In our work, we adopted an event-based encoding fashion for sEMG signals which was validated by a regression task. This method is inspired by how mammalian cochlea processes auditory signals and has been applied for feature extraction of neural and audio signals [164, 165]. It extracted enough informative features from sEMG signals and performed well in a force estimation regression task [166]. The integration of this encoding scheme and our proposed sRNR network escalated the SNN-based gesture recognition accuracy to a new level.

5.1.2 Impact Statement

While deep learning dominates sEMG-based gesture recognition, its high computational cost limits real-time, low-power applications. Neuromorphic computing offers an energy-efficient alternative, making it ideal for edge computing in resource-constrained environments. This work is the first to integrate a PRC framework, specifically the RNR, within an SNN architecture and introduce a novel event-based encoding scheme to convert superficial EMG signals into spike trains. Our approach surpasses existing SNN-based methods in classification accuracy while remaining competitive with deep learning models in a more lightweight form. Crucially, the use of recurrent reservoirs addresses a fundamental challenge in neuromorphic systems—the absence of built-in memory. By generating memory at the network level, this method enables robust, real-time processing of dynamic signals, which is essential for real-world biomedical

applications. This advancement paves the way for next-generation wearable systems with ultra-low latency and embedded intelligence.

The proposed gesture recognition system was verified by an open-access database. While the use of SNNs for sEMG-based gesture recognition has been explored previously, this research offers substantial advancements in this field, as detailed below:

- We incorporated a PRC framework called RNR with an SNN scheme for the first time to achieve sparse and low-precision data representation beneficial to memory requirements and low latency. While PRC generates dynamics directly through physical systems beneficial to resource-efficient information processing, the RNR paradigm additionally outperforms other PRC paradigms by offering a simplified hardware design that is more easily interpretable by algorithms. This design minimizes the need for modules such as ADC, buffer and memory, thereby reducing overall system complexity.
- The proposed network has a fixed and simplified topology, which is hardware-friendly and capable of using trainable nonlinear classifiers. Additionally, only the readout layer requires training, resulting in fewer parameters monitoring and reduced operations compared with backpropagation-aided DNNs.
- The proposed work obtained state-of-the-art performance among SNNs and demonstrated competitiveness with deep learning methods.

5.2 Dataset description and pre-processing

We evaluated our method using a publicly available sEMG dataset. Encoding signal to spikes is a crucial step in event-based signal processing [167]. In this work, the signals were converted into spike trains using an event-based encoding scheme inspired by the mammalian cochlea [166, 168] to feed into a novel RNR, implemented in an SNN using SNNtorch [131] to classify hand gestures.

5.2.1 sEMG dataset and pre-processing

NinaPro databases are open-access EMG datasets that are commonly used in hand gesture classification tasks [105]. In our project, the NinaPro DB2 was chosen to validate the proposed approach. This database is composed of 40 intact subjects (28 males, 12 females) performing 50 gestures with 6 repetitions for each. The sEMG signals were collected by 12 Delsys Trigno Wireless electrodes placed on the forearm and sampled at a rate of 2 kHz. Each movement repetition lasts 5 s and then returns to the rest state for 3 s to remove any residual muscular activation. The 50 gestures include exercise B, C, D and rest position: exercise B comprises 8 isometric

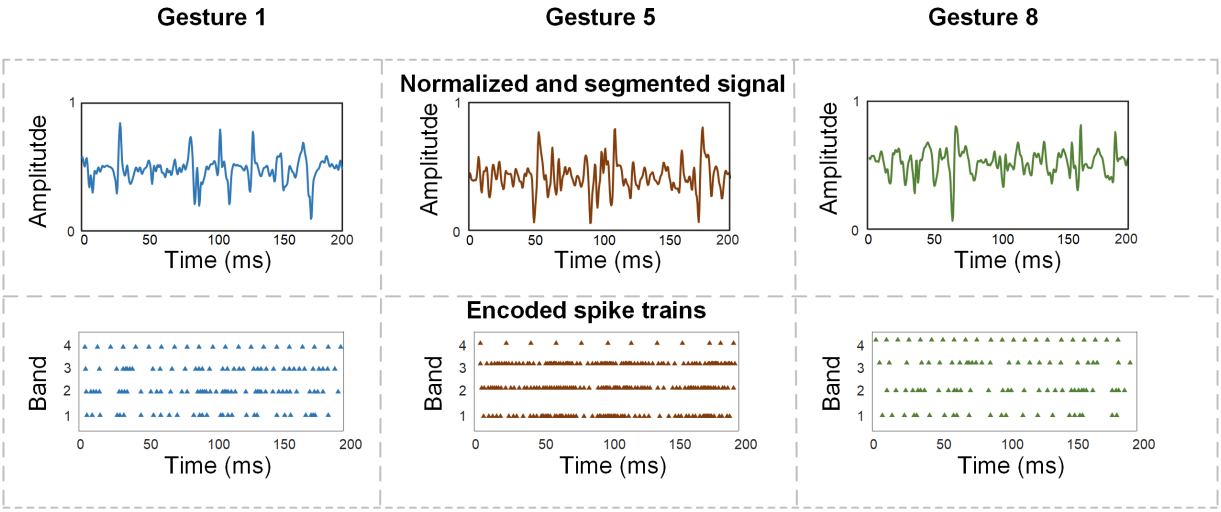

Figure 5.2: Examples of one channel of normalized sEMG signals and encoded spike trains for Gesture 1, Gesture 5 and Gesture 8, respectively.

and isotonic hand configurations and 9 wrist movements; exercise C comprises 23 grasping movements which are common daily-life actions; exercise D includes 9 force patterns [105].

Before converting the signal to spike trains that are compatible with an SNN architecture, a pre-processing procedure including normalization and segmentation is required to construct the training and testing datasets, as described in the following steps:

1. *Normalization:* To make it easier to compare sEMG signals from different repetitions and subjects and ensure that all signals have the same data distributions, a normalization procedure is applied to scale the signals to the range of (0,1) by equation (5.1).

$$signal_{normalized} = \frac{signal - min(signal)}{max(signal) - min(signal)} \tag{5.1}$$

2. *Segmentation and Windowing:* In biomedical recognition tasks, the acquisition of precisely labelled datasets is essential. As some participants may start the movement earlier than the actual motion and some may start later, we only investigated the steady state of the movements. This work omitted the initial and final 600 ms of each repetition, which may include gesture transition states, similar to the work done in [82] for intuitive comparison. Besides, an optimal window length of 150-250 ms is preferred regarding classification performance latency in prosthetic control [169]. Given the constraints, we constructed a dataset with an equal distribution of gestures, employing a window length of 200 ms for each segment. The examples of preprocessed sEMG signals with a window length of 200 ms can be viewed in Fig. 5.2.

3. *Training and Testing Sets:* We focused on evaluating the intra-subject gesture classifica-

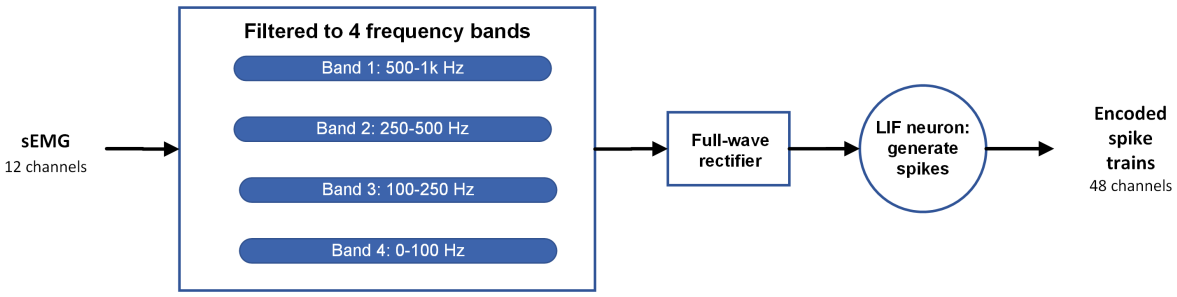

Figure 5.3: The process of spike encoding. After encoding, the original 12 channels of sEMG signals are encoded to 48 channels of spike trains.

tion performance. For each subject, the training and testing sets were shuffled and split by a 4:1 ratio. The classification accuracy and standard deviation were obtained by averaging over all 40 subjects included in Ninapro DB2.

5.2.2 Spike encoding

Encoding the raw sEMG into spike trains is a crucial step in terms of performing spike-based gesture classification. The spike sequences should convey as much intrinsic information about muscle activity included in the raw sEMG signals as possible during the encoding process. Inspired by how mammalian cochlea process auditory signals, we adopted an event-based encoding scheme in our project that exploits the raw sEMG signals to four different frequency bands and encodes the extracted signals in each frequency band to spike trains by LIF neurons based on the energy of the frequencies [166, 168]. Since in this work we performed a classification task, the spike encoding is slightly different. The detailed encoding strategy is explained below and a flow chart of this process is illustrated in Fig. 5.3.

1. *Bandpass Filtering:* Each channel of the raw sEMG signals is filtered by a four 4th-order Butterworth bandpass filter. According to the Nyquist-Shannon Sampling Theorem, the sampling frequency should be twice the highest frequency component of the signal to avoid aliasing as defined in equation (5.2):

$$f_{sampling} \geq 2 f_{max} \tag{5.2}$$

As the sampling frequency for NinaPro DB2 is $f_{sampling} = 2kHz$, the max cutoff frequency for the frequency bands should be $f_{max} = 1kHz$. Besides, the energy of sEMG signals is mainly concentrated within the frequency range of 10 to 500Hz [170]. In this case, we split the (0,1 kHz) band into 4 frequency bands, denoted by the logarithmic-distributed cutoff frequencies presented in equation (5.3).

$$f_{cutoff} = \{0, 100, 250, 500, 1000\} Hz \tag{5.3}$$

The 12 channels of raw sEMG signals will be expanded to 48 channels accordingly.

2. *Full-wave rectifying:* The injected currents in LIF neurons are required to be positive, therefore, a full-wave rectifier is applied to keep all components positive.

3. *Genereating spikes through LIF neurons:* The LIF neuron is a simplified model used to describe the electrical characteristics of a biological neuron. The dynamics of the LIF neuron is governed by the following differential equation (5.4):

$$\frac{dV_{mem}(t)}{dt} = \frac{-(V_{mem}(t) - V_{rest})}{\tau} + \frac{I_{in}(t)}{C_m} \tag{5.4}$$

where τ is the time constant for membrane relaxation time, $V_{mem}(t)$ is the membrane potential at time t, V_{rest} is the baseline membrane potential when the neuron is inactive, $I_{in}(t)$ is the injected current that drives the membrane potential and C_m is the membrane capacitance. $S_{out}(t) \in \{0, 1\}$ is the output spike train generated by the neuron. The neuron will fire a spike ($S_{out}(t) = 1$) when the membrane potential $V_{mem}(t)$ exceeds the threshold V_{thr} and it is reset to V_{reset}. Otherwise, the reset term will not be applied ($S_{out}(t) = 0$). This process can be denoted by equations (5.5) and (5.6):

$$S_{out}(t) = \begin{cases} 1, V(t) \geq V_{thr} \\ 0, otherwise \end{cases} \tag{5.5}$$

$$V(t) = V_{reset}, when\, S_{out}(t) = 1 \tag{5.6}$$

In this work, a set of 48 LIF neurons is employed to convert the 48 channels of processed sEMG signals to 48 channels of spike trains. Examples of encoded spike trains for 3 gestures all from electrode 1 are demonstrated in Fig. 5.2. The parameters in each of the LIF neurons were fine-tuned to keep the spike firing rate of each channel under 300 Hz. As the spike trains include precise timing of spikes, higher firing rate could result in saturated neurons which may blur the distinction between significant temporal patterns and lose valuable information. The selection of 300 Hz as the maximum spike firing rate was considered to prevent the network from reaching saturation and causing performance degradation. Each subject shares the same parameters in terms of spike encoding.

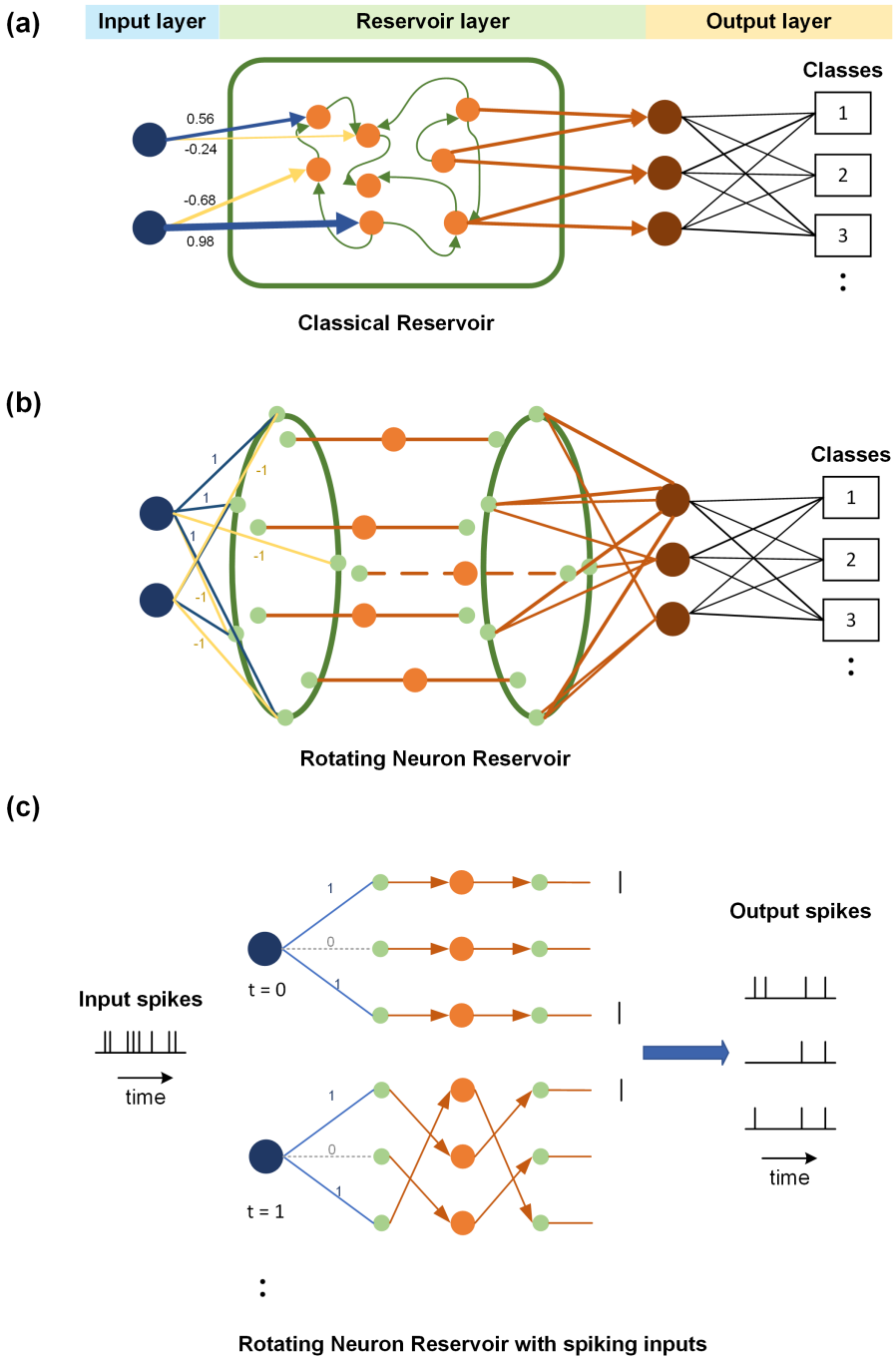

Figure 5.4: The description of reservoir topologies. The weights of input masks is represented by different colors, dash types and line thicknesses. (a) A classical reservoir topology. The input weight mask W_{in} is fixed and random from a uniform distribution [-1,1]. In addition, the connections among neurons are also fixed and random. (b)&(c) An RNR topology. A 3D description in (b) and a 2D sketch expanded by time in (c). The connections of the input-to-reservoir layer and reservoir-to-output layer are circularly shifted at each time step. (b) exhibits a binary input mask, randomly selected numbers from {-1,1} in conventional eRNR. (c) demonstrates a binary input mask {0,1} in a three-neuron sRNR topology proposed in our work that incorporates spike trains as inputs. The resulting outputs are also spike trains.

5.3 Network design

5.3.1 Network description

A reservoir computing network, as a subset of RNNs, is particularly suitable for temporal signal processing due to the recurrent connections among its neurons [22, 36]. The reservoir projects the inputs to a higher dimension by a set of randomly inter-connected recurrent neurons and generates rich dynamics in the high-dimensional feature space that facilitates linear separability. The values of weights and connections among neurons in the reservoir are fixed, and only the output layer requires training. Compared with other deep RNNs which require weight updates in every layer with careful selection of learning parameters, the training cost for reservoir computing is lower.

As wearable devices have a strong demand for low-power time series processing, the PRC framework was proposed to meet edge computing requirements that incorporate information processing near sensors or into sensors and reduce adaption delay caused by data transmission [11]. In our study, we conducted a gesture recognition task, with a preference for processing data proximate to the terminal device that generates it to minimize the need for data transmission to a centralized server, thereby saving computational overhead and decreasing bandwidth usage. Moreover, local data processing mitigates the risk of potential privacy breaches, which is particularly critical in applications involving sensitive patient information. Consequently, we selected a PRC architecture called RNR to execute the gesture classification task.

In the classical reservoir topology, the weights and connections among neurons are, although fixed, both randomly generated as shown in Fig. 5.4(a), making it hard to implement by hardware components directly. Nevertheless, a simplified cyclic topology was proposed to minimize the complexity of the reservoir without degrading the performance [14]. In this structure, the units are organized in a ring topology, and nonzeros elements in reservoir weight matrix W are on the $W_{i,i+1} = 1$ and $W_{1,N} = 1$, described by the matrix in equation (1.1).

This simplified topology was later realized by electronic circuits using rotating elements (analog multiplexers) and known as eRNR [15]. The topology of theRNR is demonstrated in Fig. 5.4(b). The connections of the input-to-reservoir layer and reservoir-to-output layer are circularly shifted at each time step. A sketch of the working principle of this circular shift procedure expanded by time is demonstrated in Fig. 5.4(c). The impact of this rotating procedure on the inputs is mathematically the same as the weight matrix W in equation (1.1) [15]. The authors conducted comparisons between the RNR structure and the single node with delayed feedback structure [29], the latter being another widely utilized topology in PRC, and claim advantages mainly in terms of (i) lower hardware components costs without ADCs, (ii) parallel operation decrease system complexity and latency while the delayed feedback line operates in a

serial manner, (iii) capable for large-scale integration.

Based on the description of RNR above, we implement it in an SNN architecture, and a sketch of the working principle of this sRNR is illustrated in Fig. 5.3(c).

There are three layers in a typical reservoir computing paradigm: an input layer, a reservoir layer, and a readout layer. As mentioned above, the structure of the input is fixed and the reservoir layer is predetermined to rotate periodically at the initialization stage, only the readout layer requires training. Typically, a prevalent way for solving classification tasks using RC-related network relies on linear model trained by convex optimization technique like ridge regression, and using a WTA algorithm to determine the corresponding target class that exceeds a certain threshold [15, 16, 18]. Despite their exceptional training speed, multivariate time series classifiers utilizing a standard RC architecture with linear regression fail to attain the accuracy levels achieved by fully trainable neural networks. With the increase of target classes, such as the gesture recognition task in our work that involves 50 distinct gestures, nonlinear methods could demonstrate superior performance. However, the high-dimension and high-precision reservoir states prevent the use of standard nonlinear classifiers being applied on the readout layer, while some dimension reduction procedures on the reservoir states were proposed to apply nonlinear classifiers on the readout layer [19]. In contrast, we implemented the reservoir in the low-precision spiking domain in this work, solving the challenge of using nonlinear classifiers. Two distinct classifiers were utilized: the SVM classifier and a delta learning rule with Softmax activation.

The detailed design of the input and reservoir layers will be introduced as follows and the training of the readout layer will be covered in Section 5.3.2.

1. *Input layer:* The input layer incorporates an input masking procedure that interfaces the input spike trains with the reservoir layer given by equation (5.7), targeting at increasing state richness [12]:

$$Spk_M^{N\times L}(t) = W_{in}^{N\times 1} \otimes Spk(t)^{1\times L} \tag{5.7}$$

where $Spk(t)$ denotes the input spike trains, W_{in} denotes the input mask matrix, $Spk_M(t)$ denotes the masked spike trains, N is the reservoir size (N neurons in the reservoir), and L is the length of the spike train. The masking matrix W_{in} usually contains randomly selected numbers from a uniform distribution of [-1,1] or randomly chosen binary weights {-1,1} in classical RC and eRNR frameworks respectively as indicated in Fig. 5.4(a) and 5.3(b). And the 1 and -1 denote the positive/negative signal source in eRNR. However, as a spike-based framework proposed in our work, these choices may not be compatible with the network, as negative and floating point numbers should be avoided to add additional hardware costs. At the same time, a sparse connection to establish connection probabilities

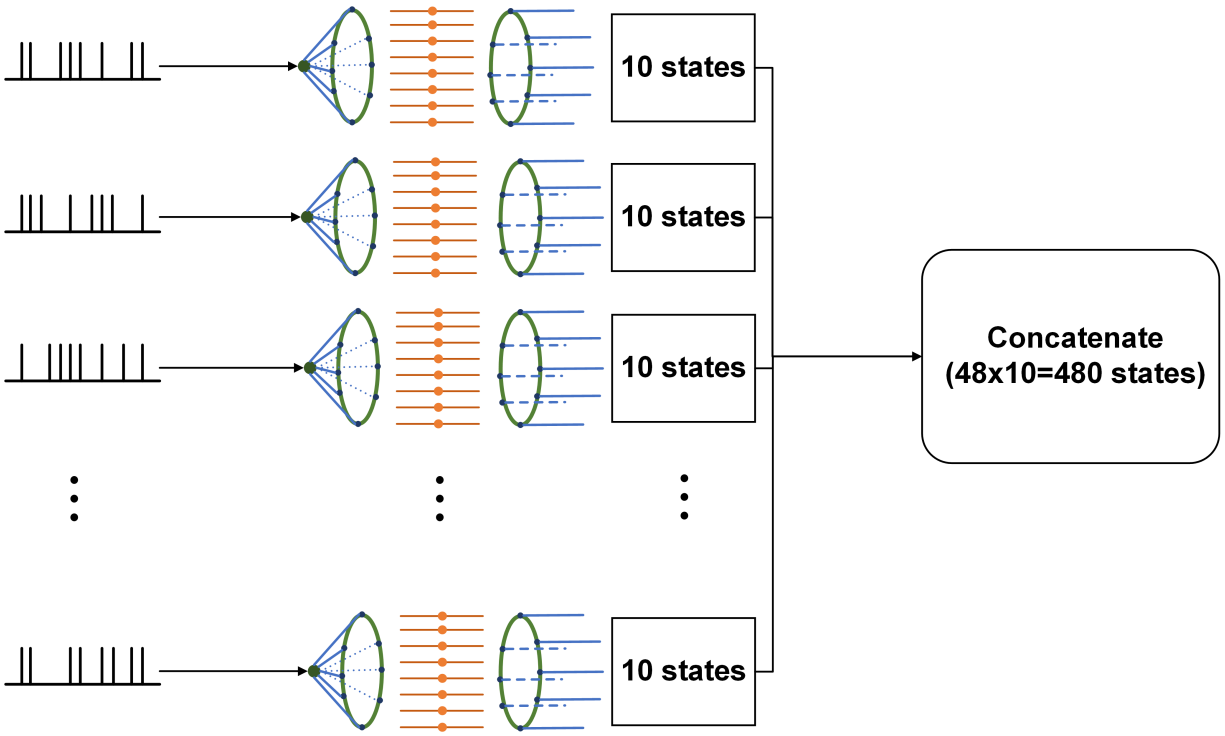

Figure 5.5: Each 10-neuron reservoir processes an input spike train. The parallel reservoirs project the input spike trains to a higher dimension (from 48 to 480).

might be useful in SNN [171]. Taking the two factors into account, we finally chose the binary mask {0,1} as shown in Fig. 5.4(c). When 0 is applied, it indicates an absence of connection between the input and the LIF neuron. Otherwise, it exerts no effect on the connection between the input and LIF neuron at that time step.

2. *Reservoir layer:* The major difference between sRNR and eRNR lies in the dynamic neurons in the reservoirs. Fig. 1.3 provides insights into the circuit for the two neuron models. In eRNR, the neuron is a nonlinear integration-ReLU-Leakage circuit that performs two important functions: nonlinearity and dynamics. While the nonlinearity is realized by diodes which have similar characteristics with the ReLU activation function and dynamics are provided by the leaky and integrate circuit. However, in sRNR, the dynamic neuron is a biologically plausible LIF model. It also has integration and leakage characteristics, but different from a nonlinear activation for a continuous analog signal, the neuron receives spikes as inputs and also generates dynamics in the term of spikes. The neuron model we used in our project is called the Lapicque LIF neuron model provided by the simulator SNNtorch [33, 131]. The dynamics in a LIF neuron can be quantified by an ODE given in equation (2.1) which provides a discrete and recurrent representation.

   This model coarsely represents a low-pass filter circuit of a resistor R and a capacitor C. The parameter setting is presented in Table 5.1. The working principle is the same as discussed in Section 5.2.2. The neuron will fire a spike when the membrane potential V_{mem}

Table 5.1: Parameter settings for LIF neuron.

Resistance (R)	**Capacitance** (C)	**Time step**	**Threshold**
5	3×10^{-3}	1×10^{-3}	$0.5V$

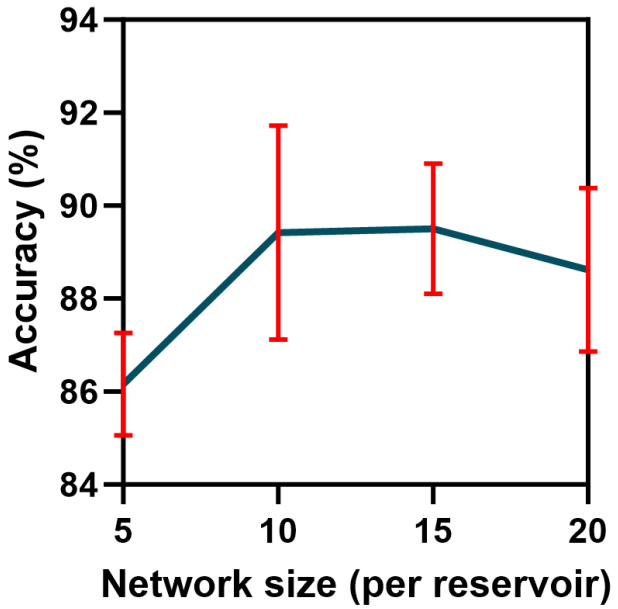

Figure 5.6: The effect of network size on the classification accuracy of a representative subject by performing exercise B (17 gestures).

exceeds the threshold V_{thr}. Instead of collecting the continuous internal states as shown in Fig. 1.3(a), we collected the output spikes for training sRNR.

The spike trains are injected into the reservoir layer to generate transient dynamics that will be collected for training. There are 48 spike trains and each of the spike trains is injected into a 10-neuron reservoir to form a parallel structure as shown in Fig. 5.5. The reason for this is that the simulation time for large reservoirs will be extensive and long, a parallel structure allows for neurons states to be computed simultaneously and speeding up the computation times [44]. The size of 10 was chosen by conducting experiments on a subset of the gestures, as shown in Fig. 5.6. The accuracy does not necessarily increase with a larger network size, therefore, a size of 10 was chosen by considering the tradeoff between accuracy and computation cost.

According to the reservoir structure, output spikes have a dimension of 480. An example of input spike trains and output spike trains is shown in Fig. 5.7.

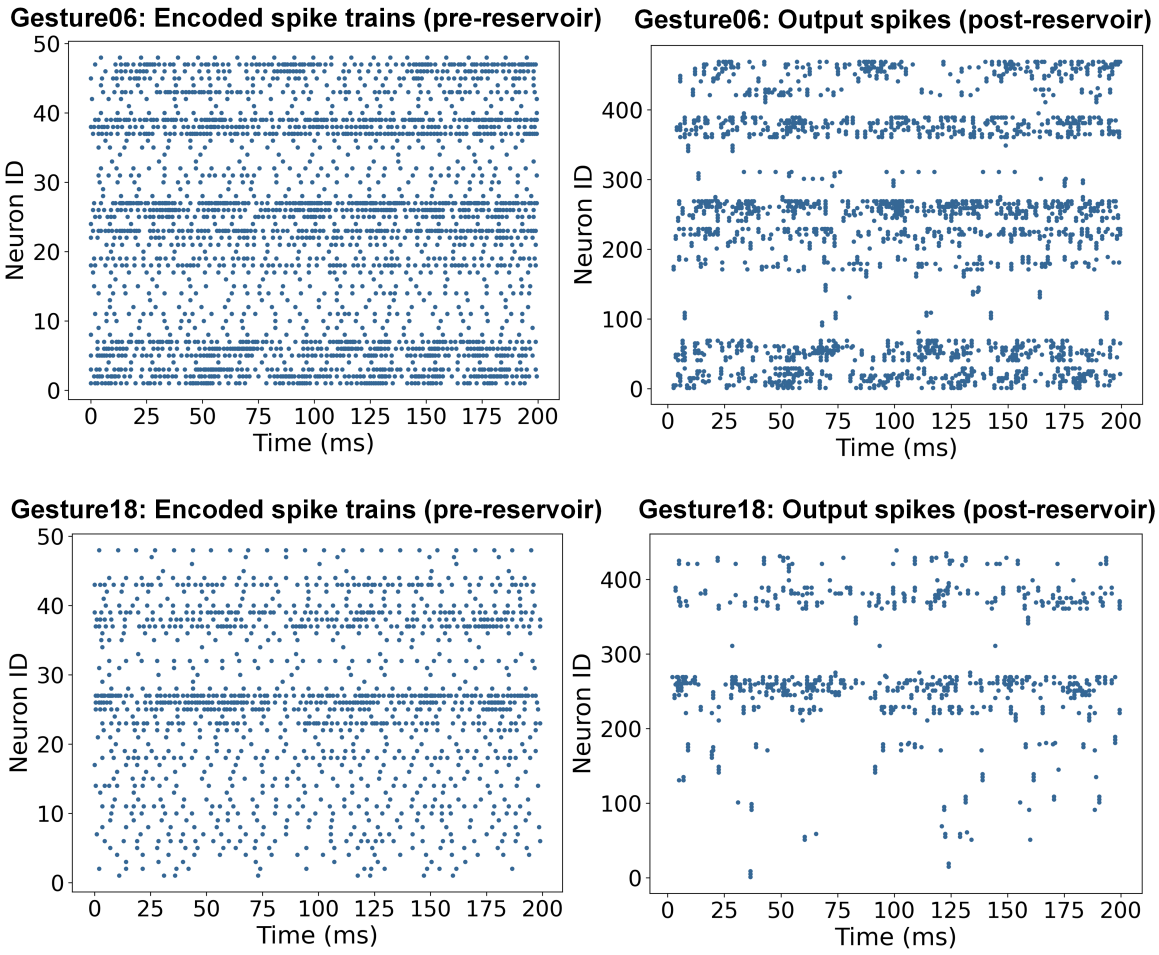

Figure 5.7: Two examples of input spike patterns (left) and output spike patterns (right).

5.3.2 Readout and classification

After the generated output spikes are collected, they will undergo training for classification. As discussed in Section 5.3.1, the training takes place in the readout layer only. In this chapter, two different supervised classification algorithms were adopted to train the readout layer: a SVM classifier and a delta rule as the learning rule for the single-layer neural network. The SVM classifier is utilized as a baseline to assess the performance of the network and optimize parameters related to spike-encoding and the dynamics generated by reservoirs. Subsequently, a delta rule is applied as it could be fully spiking, offering the potential for implementation on neuromorphic chips which has been investigated by several research [57, 172, 173].

The feature vectors sent to classification consisted of spike counts in a time duration, which refers to a count and binned kernel [128]. In this case, the spiking information is transformed into natural numbers.

SVM is a supervised machine learning algorithm used for classification tasks which works by finding hyperplanes that best separate the dataset. In the SVM algorithm, we split the training

and testing sets in a 4:1 ratio, and the samples for training were shuffled. In addition, the results were cross-validated over 5 different combinations of training and testing sets to evaluate the performance of classification.

Delta rule is a foundational learning rule typically applied in the training of a single-layer neural network. The rule adjusts the weights (w) of the inputs feature (x) based on the difference between the expected output (y) and the actual output ($\hat{y}$) to reduce the error in predictions:

$$\Delta w = \alpha(y - \hat{y})x \tag{5.8}$$

where Δw is the weight change and α is the learning rate.

Since it is a multi-class classification task, we applied Softmax activation in the output layer. It converts the linear outputs into probabilities that sum up to 1. The prediction is made by finding the maximum softmax scores (probability). The process is denoted by the following equations:

$$P(y = i) = \frac{e^{z_i}}{\sum_{j=0}^{k-1} e^{z_j}}, i \in \{0, ..., k-1\} \tag{5.9}$$

$$\hat{y} = max(P(y = i), i = 0, ..., k-1) \tag{5.10}$$

where $P(y = i)$ is the output probability for the i-th class, z_i is the i-th element in the input vector z and $\hat{y}$ is the predicted output class. The cross-entropy (CE) loss is computed at each epoch as the cost function to evaluate how well the model's predictions align with the actual target values.

5.4 Analysis and results

In this section, we presented a t-distributed stochastic neighbor embedding (t-SNE) technique which assists in revealing clusters and patterns in the data. Also, we compared the performance by applying different training algorithms. Furthermore, we compared our method with state-of-the-art gesture recognition using sEMG signals from Ninapro databases.

5.4.1 t-SNE

As introduced in previous sections, the reservoir projects the inputs to a higher dimensional feature space to facilitate linear separability. Therefore, to visualize the linear separability, we performed a t-SNE analysis on the input layer (pre-reservoir) and output layer (post-reservoir) to observe clusters and patterns inside the data. The t-SNE was applied separately to exercises B, C, and D described in Ninapro DB2, as visualizing all gestures in the same plot would result in

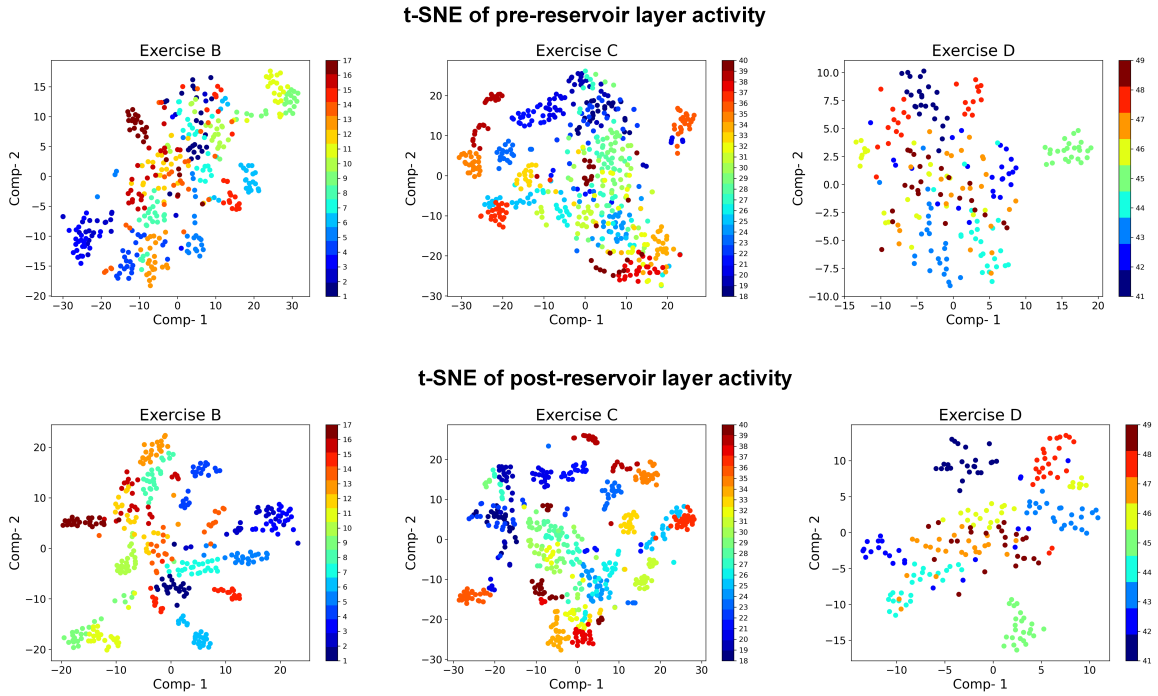

Figure 5.8: Two-dimensional t-SNE projections applied on the input layer (pre-reservoir) and output layer (post-reservoir) for Exercises B, C, and D separately in a representative subject. Linear separability is enhanced and observed by more closely clustered patterns following the reservoir layer, where the data are projected into a higher-dimensional feature space.

indistinguishable colour discrimination. According to the demonstration in Fig. 5.8, projecting the data to a higher-dimensional space facilitates separability since data points belonging to the same class form more closely clustered groups.

5.4.2 SVM for classification

SVMs apply kernel functions to map the data into a higher-dimensional space where data are linearly separable. In this work, we applied both a linear kernel and an radial basis function (RBF) kernel to distinguish all 50 gestures included in Ninapro DB2 and obtained accuracy of $74.6\% \pm 6.3\%$ and $65.2\% \pm 7.4\%$ on the testing sets respectively. Applying a linear kernel achieved around 10% higher accuracy. This improvement can be attributed to the fact that a linear kernel does not perform any transformation on the data, making it well-suited for data that is already linearly separable. In contrast, the RBF kernel, as a nonlinear kernel, projects the data to an infinite-dimensional space which is often used for nonlinear problems. Since the reservoir layer in our proposed method already performs the higher-dimensional mapping which facilitates linear separability, a linear kernel could demonstrate superior classification performance.

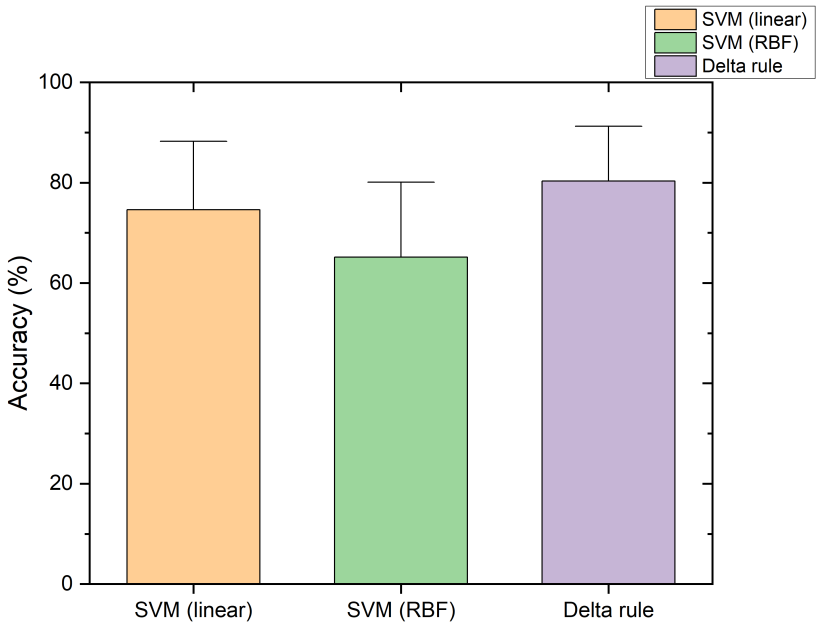

Figure 5.9: The accuracies for testing sets using SVM (both linear kernel and RBF kernel) and delta learning rule with softmax classifier, respectively. Results were averaged over all the subjects with average and standard deviation reported in the bar chart.

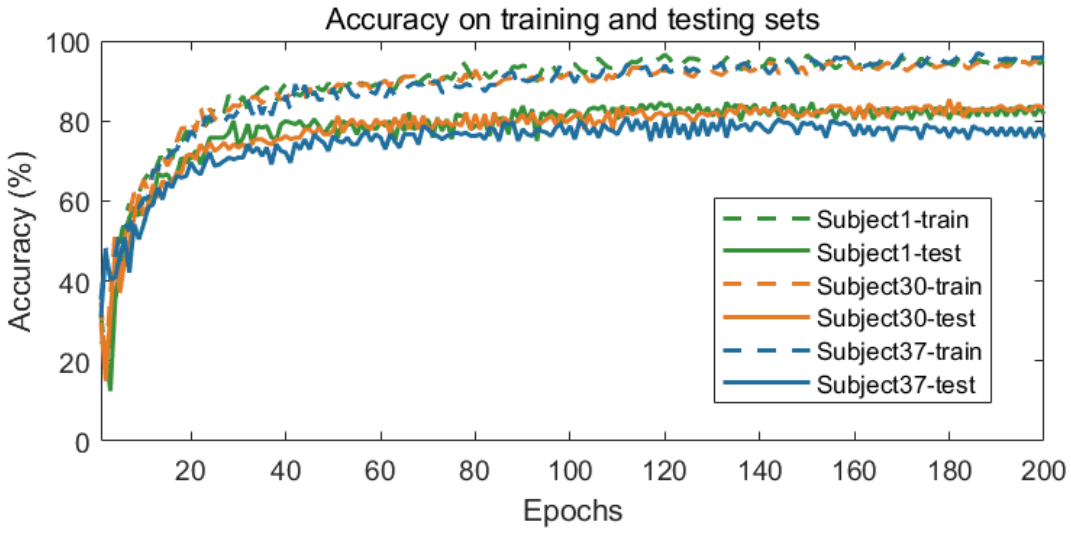

Figure 5.10: Observed classification accuracies on training sets and testing sets for subject 1, subject 30 and subject 37 through 200 epochs of training.

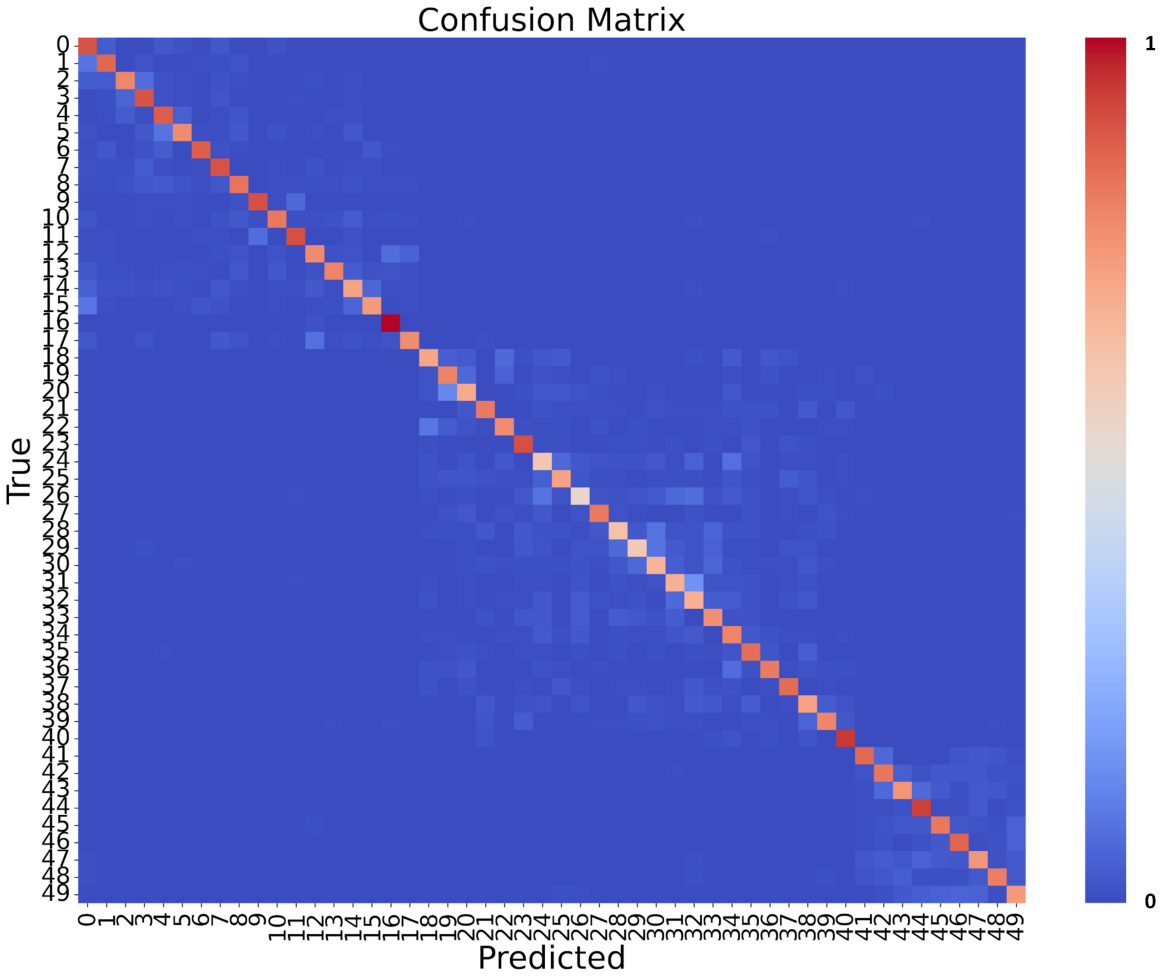

Figure 5.11: Confusion matrix for classifying 50 gestures on the testing set for all subjects.

5.4.3 Delta learning rule for classification

Using the learning rule explained in Section 5.3.2, we applied it to classify all 50 gestures by setting the learning rate to 0.005 and batch size to 1. The loss for each subject converges to below 0.2 after 50 epochs of training and achieves a maximum classification accuracy of 94.6% on the training set and 80.3% on the testing set after 200 epochs of training by averaging over the 40 subjects. Taking the obtained classification accuracy by SVM classifiers as a baseline, we compared the performance of different algorithms as reported in Fig. 5.9. Delta learning rule by applying softmax classifier shows a slightly higher classification accuracy. In addition, Fig. 5.10 illustrates the observed accuracies on training and testing sets through training for three representative subjects and a confusion matrix to illustrate the performance of classifying 50 gestures by averaging all subjects on the testing sets using the delta rule.

In addition to the evaluation of classification accuracy, we also assess the model performance based on Positive Predictivity (PP), Sp, Se and F1 score. These metrics can be calculated by using the values of TP, TN, FP and FN. Se, also known as recall, is the ability to correctly identify all positive instances of a class, defined by equation (4.5).

Sp presents the ability of a classifier to correctly identify all negative instances of a class, defined by equation (4.6).

PP, also known as precision, is the proportion of predicted positives that are actually positive, defined by equation (4.7).

F1 score is the combined assessment of precision and sensitivity, defined by equation (4.8).

The detailed statistical results of the metrics in percentage mentioned above for each subject are shown in Table 5.2, with an overall accuracy of 80.3%, PP of 80.1%, Sp of 99.6%, Se of 77.2% and F1 score of 78.9%.

5.4.4 Comparison with the state-of-the-art

In recent years, significant advancements have been made in the field of sEMG-based gesture recognition, particularly with the development of state-of-the-art methods such as deep learning algorithms. These approaches have demonstrated notable improvements in classification accuracy. Before deep learning, conventional machine learning algorithms relied on feature extraction in the pre-processing stage. A common machine learning technique selected five feature sets consisting of root mean square (RMS), Time-Domain (TD) statistics, histogram (HIST), marginal Discrete Wavelet Transform (mDWT) and the normalized combination of all above, and achieved an overall classification accuracy of 75.3% using RF to classify all 50 gestures in the Ninapro DB2 dataset [105]. In contrast, DNNs can automatically learn from data to extract features while maintaining a high classification accuracy. For example, a multi-view convolutional neural network (CNN) method achieved an overall accuracy of 83.7% in classifying all 50 gestures included in Ninapro DB2 while another Auto-Labelling-Refining (ALR) CNN method achieved an overall accuracy of 87.9% in classifying 41 gestures in Ninapro DB2 (all gestures included in exercise B and C) [155, 156]. However, DNN-related methods usually bring increased computational costs, making them unsuitable for the lightweight algorithm requirement of edge devices. In addition, continuous monitoring the muscle activities through wearable devices could generate massive sEMG data, resulting in an information bottleneck that challenges data transfer and subsequent post-precessing.

Neuromorphic-based approaches were studied to solve these challenges by bringing analogue and continuous sEMG signals to the discrete spiking domain in an event-driven mode. Such methods can process data on the sensor side with reduced computational overhead and latency,

Table 5.2: The results of statistical analysis.

Subject No.	Acc (%)	PP (%)	Sp (%)	Se (%)	F1 (%)
1	84.6	85.3	99.6	80.1	81.5
2	80.8	79.3	99.6	77.5	80.0
3	85.8	84.3	99.7	81.5	80.0
4	80.4	80.5	99.6	77.7	79.1
5	83.7	85.7	99.6	82.2	83.8
6	82.1	82.4	99.7	81.0	82.1
7	80.4	77.4	99.5	75.3	77.0
8	79.6	80.8	99.6	78.4	80.1
9	87.1	83.8	99.7	81.5	83.2
10	81.7	77.7	99.6	76.4	79.5
11	80.4	77.0	99.6	74.8	77.4
12	63.8	62.1	99.2	57.7	59.3
13	79.2	81.3	99.6	78.9	79.7
14	73.8	74.2	99.4	70.8	72.4
15	85.4	87.2	99.7	85.0	86.0
16	68.3	63.0	99.3	61.9	64.8
17	87.9	87.6	99.7	83.6	85.3
18	72.9	73.0	99.4	69.8	72.3
19	89.2	88.6	99.8	87.8	89.1
20	71.6	74.6	99.4	72.3	73.3
21	71.7	75.5	99.5	72.5	75.9
22	84.2	83.9	99.7	83.3	84.9
23	91.3	91.3	99.8	89.8	90.8
24	83.8	82.8	99.7	81.4	82.3
25	82.1	77.1	99.5	74.6	76.6
26	82.1	83.1	99.6	79.6	80.9
27	65.8	62.0	99.2	60.1	64.1
28	78.8	77.6	99.5	74.5	76.8
29	77.9	81.4	99.5	76.9	79.0
30	85.4	82.9	99.7	81.6	83.2
31	72.1	76.6	99.4	70.1	71.2
32	86.7	87.4	99.7	84.0	84.3
33	88.3	89.3	99.7	86.8	87.7
34	81.7	78.2	99.6	76.1	78.5
35	86.7	86.5	99.7	83.5	84.1
36	77.1	81.3	99.5	77.4	78.5
37	80.8	79.9	99.5	75.8	77.9
38	82.5	79.3	99.5	76.4	79.2
39	77.9	79.7	99.5	76.9	78.9
40	78.3	81.3	99.5	73.0	74.6
overall	80.3	80.1	99.6	77.2	78.9

providing a new solution to wearable devices. Nevertheless, a significant degradation in classification performance arises in SNN-based methods. An accuracy of only 57.2% was obtained in classifying a subset of 8 selected gestures in Ninapro DB2 by applying a spiking RNN with a STDP learning rule [82]. In addition, the same network was deployed on a configurable neuromorphic chip and achieved a classification accuracy of 55.9%, only a small reduction compared with the software simulation result. An improvement to 74.0% was made by applying a spiking fully connected layer (sFCN) to classify a subset of 13 gestures in Ninapro DB5, also implemented on a neuromorphic chip [163]. However, the results remain less competitive than state-of-the-art machine learning algorithms. In our work, we adopted a novel event-based spike encoding scheme for sEMG signals and proposed an sRNR which attempted the fusion of PRC and SNN for the first time and achieved an overall accuracy of 80.3% by applying a delta learning rule and softmax classifier in classifying all 50 gestures in Ninapro DB2, with fewer parameters monitoring in training a single-layer readout. This marks a significant improvement over the existing state-of-the-art in terms of sEMG-based gesture recognition in the SNN domain. Although this work is not implemented on a similar neuromorphic chip since the changing connections, as described in Section 5.3.1, are not compatible with neuromorphic chips which require initialization of network topologies during the startup phase, it still paves way to the integration into chip level with dedicated hardware design for future research. A summary of the comparison of the state-of-the-art is demonstrated in Table 5.3.

5.5 Discussion and Conclusion

In this chapter, a novel sEMG-based gesture recognition framework by implementing a PRC topology within an SNN architecture was introduced, alongside an innovative event-based spiking encoding strategy for the sEMG signals. We performed a thorough investigation into the proposed approach by a t-SNE visualization of internal network activity and two different classifiers to evaluate the classification performance. The results indicate that our approach significantly outperforms existing SNN-based methods for the large-scale public dataset with higher classification accuracy, and at the same time, has a lower training cost compared with deep learning algorithms. Although this work focuses on sEMG signals, the modular and event-driven nature of the proposed sRNR architecture is compatible with other bio-signals such as EEG and ECG, provided a suitable spike encoding layer. The reservoir can be reused with minimal changes, enabling cross-domain generalization. Statistical evaluations demonstrate that our proposed solution pipelines a new insight into processing real-time signals at the edge for wearable devices, promising compact and lightweight electronic systems for temporal signal processing in wearable devices. While this experiment is based on collected and processed data, a real-time latency evaluation could be retained for future real-world applications.

Table 5.3: Comparison of recent research found in the literature using sEMG-based gesture recognition.

Work	**Approach**	**Feature extraction**	**Demonstration**	**DNN**	**Database**	**No. of Gestures to be classified**	**Accuracy(%)**
Atzori *et al.* [105]	RF	Yes	Software	No	Ninapro DB2	50	75.3
Fatayer *et al.* [156]	ALR-CNN	No	Software	Yes	Ninapro DB2	41	87.9
Wei *et al.* [155]	Multi-view CNN	No	Software	Yes	Ninapro DB2	50	83.7
Vitale *et al.* [163]	Spiking FCN	No	Hardware	Yes	Ninapro DB5	13	74.0
Ma *et al.* [82]	Spiking RNN + STDP	No	Software	No	Ninapro DB2	8	57.2
			Hardware				55.9
This work	Spiking RNR	No	Software	No	Ninapro DB2	50	80.3

Chapter 6

Conclusions and Future Perspectives

6.1 Conclusions of this thesis

This thesis presents the vision of PRC for processing biomedical signals at the edge of wearable devices, emphasizing its advantages in enabling real-time and low-latency applications. While there is a growing interest in existing PRC systems for biomedical applications, the primary focus of this thesis is the RNR framework and its applicability to various real-world biomedical signal processing tasks. Current research predominantly centers on hardware implementation platforms and is often validated through relatively simple tasks; however, the potential of PRC for handling complex tasks within the domain of edge computing remains underexplored. This thesis addresses this gap by introducing advancements in RNR design that bridge the gap between hardware and algorithmic efficiency, offering a compelling solution to traditional computational methods for tasks requiring real-time and adaptive signal processing.

Firstly, we systematically review a wide range of implementation paradigms of PRC systems, classifying them according to distinct methodologies and their applications in biomedical signal processing. Beyond examining reservoir implementation platforms, we also evaluate training methodologies for various biomedical signal processing tasks, highlighting their crucial impact on the performance of neuromorphic systems.

To comprehensively assess the effectiveness of RNR, the main objective of this thesis, in biomedical signal processing tasks, the initial phase of this research investigated the feasibility of utilizing PRC as a predictive model for biomedical applications, along with parameter optimization techniques. The findings, as detailed in Chapter 3, demonstrated low prediction errors, highlighting the potential of PRC in this domain.

However, real-world signal processing tasks present significantly greater complexities. While neuromorphic signal processing currently underperforms compared to state-of-the-art DL algo-

rithms and has been relatively underexplored in classification tasks, this study was extended to investigate the application of PRC in a more complex biometric identification task based on heart sounds collected from optical stethoscope. The intermediate phase of the research yielded promising results, with a classification accuracy of 89% in identifying 12 subjects as reported in Chapter 4.

A significant challenge emerged concerning the precision of high-resolution signal processing when implemented in electronic circuits within PRC-based systems. This limitation not only led to increased hardware costs associated with electronic components but also introduced scalability issues due to the large network size. This challenge motivated the next phase of research, which focused on developing a hybrid approach that integrates PRC with SNN. This combination facilitates the conversion of high-precision analog signals into a low-precision discrete spiking domain, thereby reducing both hardware complexity and storage requirements. To address this issue, an event-driven PRC framework was proposed and subsequently validated in the final research phase. The proposed algorithm reports an overall classification accuracy of 80% in recognizing 50 sEMG-based gestures, outperforming the existing SNN-based approaches. Statistical evaluations demonstrate that the proposed solution offers a novel approach to real-time signal processing at the edge for wearable devices, paving the way for compact, low-power and low-latency electronic systems capable of handling temporal signal processing in wearable technologies.

6.2 Future perspectives

As an emerging area, PRC holds significant promise in pioneering innovative solutions for wearable biomedical devices in the context of edge computing, by enabling computationally efficient yet effective algorithms in resource-constrained environments. However, the practical application of PRC systems in real-time biomedical signal processing presents a three-fold challenge, requiring careful considerations of both software and hardware aspects, alongside biomedical signal collections.

6.2.1 Algorithm

An efficient algorithm plays a crucial role in determining the overall performance of PRC systems. Regarding reservoir architecture, the deep reservoirs and wide reservoirs can be applied to enhance state richness. While wide reservoirs, characterized by a parallel architecture, have been utilized in PRC systems, further exploration of deep reservoir architectures presents a promising direction for future research. Besides, the significance of training algorithms for the readout layer is often underestimated in the current research. Traditionally, the readout layer employs linear regression for training to evaluate the performance of PRC systems with simple bench-

marks; however, when it comes to real-world biomedical signals, this approach is suboptimal for handling complex classification tasks [19]. Further research into RC algorithms for effectively training the readout layer in PRC-implemented biomedical applications is urgently required. Leveraging insights from DNNs or incorporating biological learning rules from the domain of SNN may provide novel approaches to enhance the performance of these systems [131, 147].

6.2.2 Hardware implementation

Hardware serves as a crucial part for the implementation paradigms of PRC. Although certain large-scale neuromorphic systems offer a platform with reconfigurable network structures for PRC and have demonstrated feasibility in biomedical applications, simplified hardware architectures, as illustrated in Fig. 2.2, present a promising opportunity for integration into CMOS chip-level implementations with reduced hardware costs. Beyond conventional CMOS technology, emerging memory devices hold significant potential, enabling in-memory computing solutions with ultra-low power consumption. However, challenges related to device reliability, including D2D and C2C variability, must be addressed to ensure their practical deployment in biomedical applications [12]. Photonics provide a faster computation speed, the energy consumption is not optimal compared with other solutions, and a critical research question is the alignment of timescales between the task and the reservoirs [135]. In addition to the implementation of the reservoir layer, the development of trainable circuits for the readout layer warrants further research, as it provides support for on-chip learning [73, 134, 174].

6.2.3 Biomedical signals

Biomedical signals provide valuable insights into the physiological and pathological states of the human body, however, they often exhibit noise, artifacts and subject-specific variability. It is crucial to develop robust preprocessing and denoising techniques at the level of wearable sensors to enhance signal quality and reliability [175]. Furthermore, in the context of SNNs-based biomedical signal processing, signal-to-spike conversion methods play a fundamental role in the representation of spiking information. The spike encoding scheme must effectively capture and convey essential information from the original signals to ensure high accuracy and robustness while maintaining compatibility with wearable sensor systems [167]. Traditional spike encoding methods, such as rate encoding and temporal encoding, are widely utilized in SNNs, while the delta modulation method exhibits good performance in biomedical circuits and systems [57, 82, 163]. In addition, a multi-frequency band spike conversion method inspired by cochlear acoustic signal processing also achieved good performance in both biomedical signal regression tasks and classification tasks in recent research [166, 168, 176]. It is promising to apply different encoding schemes to multiple biomedical signals to evaluate the performance in future research.

References

[1] Erika Covi et al. “Adaptive extreme edge computing for wearable devices”. In: *Frontiers in Neuroscience* 15 (2021), p. 611300.

[2] Albert Reuther et al. “AI accelerator survey and trends”. In: *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE. 2021, pp. 1–9.

[3] Paul J Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[4] Dennis V Christensen et al. “2022 roadmap on neuromorphic computing and engineering”. In: *Neuromorphic Computing and Engineering* 2.2 (2022), p. 022501.

[5] Carver Mead. “Neuromorphic electronic systems”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1629–1636.

[6] M Mitchell Waldrop. “The chips are down for Moore’s law”. In: *Nature News* 530.7589 (2016), p. 144.

[7] Catherine D Schuman et al. “A survey of neuromorphic computing and neural networks in hardware”. In: *arXiv preprint arXiv: 1705.06963* (2017).

[8] Herbert Jaeger. *Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach*. Vol. 5. 1. Citeseer, 2002.

[9] Wolfgang Maass, Thomas Natschläger, and Henry Markram. “Real-time computing without stable states: A new framework for neural computation based on perturbations”. In: *Neural Computation* 14.11 (2002), pp. 2531–2560.

[10] David Verstraeten et al. “An experimental unification of reservoir computing methods”. In: *Neural Networks* 20.3 (2007), pp. 391–403.

[11] Kohei Nakajima. “Physical reservoir computing—an introductory perspective”. In: *Japanese Journal of Applied Physics* 59.6 (2020), p. 060501.

[12] Xiangpeng Liang et al. “Physical reservoir computing with emerging electronics”. In: *Nature Electronics* 7.3 (2024), pp. 193–206.

[13] Gouhei Tanaka et al. “Recent advances in physical reservoir computing: A review”. In: *Neural Networks* 115 (2019), pp. 100–123.

[14] Ali Rodan and Peter Tino. “Minimum complexity echo state network”. In: *IEEE Transactions on Neural Networks* 22.1 (2010), pp. 131–144.

[15] Xiangpeng Liang et al. “Rotating neurons for all-analog implementation of cyclic reservoir computing”. In: *Nature Communications* 13.1 (2022), p. 1549.

[16] Xiangpeng Liang et al. “A neuromorphic model with delay-based reservoir for continuous ventricular heartbeat detection”. In: *IEEE Transactions on Biomedical Engineering* 69.6 (2021), pp. 1837–1849.

[17] Xingxing Feng et al. “Human recognition with the optoelectronic reservoir-computing-based micro-Doppler radar signal processing”. In: *Applied Optics* 61.19 (2022), pp. 5782–5789.

[18] Yanan Zhong et al. “A memristor-based analogue reservoir computing system for real-time and power-efficient signal processing”. In: *Nature Electronics* 5.10 (2022), pp. 672–681.

[19] Filippo Maria Bianchi et al. “Reservoir computing approaches for representation and classification of multivariate time series”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.5 (2020), pp. 2169–2179.

[20] Jack D Kendall and Suhas Kumar. “The building blocks of a brain-inspired computer”. In: *Applied Physics Reviews* 7.1 (2020).

[21] Amirhossein Tavanaei et al. “Deep learning in spiking neural networks”. In: *Neural Networks* 111 (2019), pp. 47–63.

[22] Mantas Lukoševičius and Herbert Jaeger. “Reservoir computing approaches to recurrent neural network training”. In: *Computer Science Review* 3.3 (2009), pp. 127–149.

[23] Min Yan et al. “Emerging opportunities and challenges for the future of reservoir computing”. In: *Nature Communications* 15.1 (2024), p. 2056.

[24] Felix Christian Bauer, Dylan Richard Muir, and Giacomo Indiveri. “Real-time ultra-low power ECG anomaly detection using an event-driven neuromorphic processor”. In: *IEEE Transactions on Biomedical Circuits and Systems* 13.6 (2019), pp. 1575–1582.

[25] Silvia Ortín et al. “Automated real-time method for ventricular heartbeat classification”. In: *Computer Methods and Programs in Biomedicine* 169 (2019), pp. 1–8.

[26] Sadman Sakib et al. “Noise-removal from spectrally-similar signals using reservoir computing for MCG monitoring”. In: *ICC 2021-IEEE International Conference on Communications*. IEEE. 2021, pp. 1–6.

[27] Yuqi Ding et al. “MMG/EMG mapping with reservoir computing”. In: *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE. 2022, pp. 1–4.

[28] Yuqi Ding et al. “A Physical Reservoir Computing Processor for ECG-to-PCG Signals Prediction”. In: *2024 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2024, pp. 1–5.

[29] Lennert Appeltant et al. “Information processing using a single dynamical node as complex system”. In: *Nature Communications* 2.1 (2011), p. 468.

[30] Lennert Appeltant et al. “Constructing optimized binary masks for reservoir computing with delay systems”. In: *Scientific Reports* 4.1 (2014), p. 3629.

[31] Heshan Wang and Xuefeng Yan. “Improved simple deterministically constructed cycle reservoir network with sensitive iterative pruning algorithm”. In: *Neurocomputing* 145 (2014), pp. 353–362.

[32] Kaushik Roy, Akhilesh Jaiswal, and Priyadarshini Panda. “Towards spike-based machine intelligence with neuromorphic computing”. In: *Nature* 575.7784 (2019), pp. 607–617.

[33] Louis Édouard Lapicque. “Louis lapicque”. In: *J. physiol* 9 (1907), pp. 620–635.

[34] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The Bulletin of Mathematical Biophysics* 5 (1943), pp. 115–133.

[35] Herbert Jaeger. “The “echo state” approach to analysing and training recurrent neural networks-with an erratum note”. In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148.34 (2001), p. 13.

[36] Mantas Lukoševičius, Herbert Jaeger, and Benjamin Schrauwen. “Reservoir computing trends”. In: *KI-Künstliche Intelligenz* 26 (2012), pp. 365–371.

[37] John Moon, Yuting Wu, and Wei D Lu. “Hierarchical architectures in reservoir computing systems”. In: *Neuromorphic Computing and Engineering* 1.1 (2021), p. 014006.

[38] Chenxi Sun et al. “A review of designs and applications of echo state networks”. In: *arXiv preprint arXiv: 2012.02974* (2020).

[39] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. “Deep reservoir computing: A critical experimental analysis”. In: *Neurocomputing* 268 (2017), pp. 87–99.

[40] Claudio Gallicchio and Alessio Micheli. “Why layering in recurrent neural networks? a DeepESN survey”. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2018, pp. 1–8.

[41] Shaohui Zhang et al. “Deep fuzzy echo state networks for machinery fault diagnosis”. In: *IEEE Transactions on Fuzzy Systems* 28.7 (2019), pp. 1205–1218.

[42] Paolo Arena, Luca Patanè, and Angelo Giuseppe Spinosa. “Robust modelling of binary decisions in Laplacian Eigenmaps-based Echo State Networks”. In: *Engineering Applications of Artificial Intelligence* 95 (2020), p. 103828.

[43] Ying-Chun Bo, Ping Wang, and Xin Zhang. “An asynchronously deep reservoir computing for predicting chaotic time series”. In: *Applied Soft Computing* 95 (2020), p. 106530.

[44] Kristof Vandoorne et al. “Parallel reservoir computing using optical amplifiers”. In: *IEEE Transactions on Neural Networks* 22.9 (2011), pp. 1469–1481.

[45] Yanan Zhong et al. “Dynamic memristor-based reservoir computing for high-efficiency temporal signal processing”. In: *Nature Communications* 12.1 (2021), p. 408.

[46] Xiangpeng Liang et al. “A Physical Reservoir Computing Model Based on Volatile Memristor for Temporal Signal Processing”. In: *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE. 2022, pp. 1–4.

[47] Chrisantha Fernando and Sampsa Sojakka. “Pattern recognition in a bucket”. In: *European Conference on Artificial Life*. Springer. 2003, pp. 588–597.

[48] Kensuke Ikeda and Kenji Matsumoto. “High-dimensional chaotic behavior in systems with time-delayed feedback”. In: *Physica D: Nonlinear Phenomena* 29.1-2 (1987), pp. 223–235.

[49] S Lepri et al. “High-dimensional chaos in delayed dynamical systems”. In: *Physica D: Nonlinear Phenomena* 70.3 (1994), pp. 235–249.

[50] Dan A Allwood et al. “A perspective on physical reservoir computing with nanomagnetic devices”. In: *Applied Physics Letters* 122.4 (2023), p. 040501.

[51] Susan Stepney. “Physical reservoir computing: a tutorial”. In: *Natural Computing* (2024), pp. 1–21.

[52] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems* 25 (2012).

[53] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *Nature* 521.7553 (2015), pp. 436–444.

[54] Dhireesha Kudithipudi et al. “Neuromorphic computing at scale”. In: *Nature* 637.8047 (2025), pp. 801–812.

[55] Sanjeev Tannirkulam Chandrasekaran et al. “Toward real-time, at-home patient health monitoring using reservoir computing CMOS IC”. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 11.4 (2021), pp. 829–839.

[56] Miquel L Alomar et al. “Efficient parallel implementation of reservoir computing systems”. In: *Neural Computing and Applications* 32 (2020), pp. 2299–2313.

[57] Elisa Donati 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.

[58] Ramashish Gaurav, Terrence C Stewart, and Yang Cindy Yi. “Spiking reservoir computing for temporal edge intelligence on loihi”. In: *2022 IEEE/ACM 7th Symposium on Edge Computing (SEC)*. IEEE. 2022, pp. 526–530.

[59] Jan Behrenbeck et al. “Classification and regression of spatio-temporal signals using NeuCube and its realization on SpiNNaker neuromorphic hardware”. In: *Journal of Neural Engineering* 16.2 (2019), p. 026014.

[60] Daniel Brunner, Miguel Cornelles Soriano, and Ingo Fischer. “High-speed optical vector and matrix operations using a semiconductor laser”. In: *IEEE Photonics Technology Letters* 25.17 (2013), pp. 1680–1683.

[61] A Namajūnas, K Pyragas, and A Tamaševičius. “An electronic analog of the Mackey-Glass system”. In: *Physics Letters A* 201.1 (1995), pp. 42–46.

[62] Miguel C Soriano et al. “Delay-based reservoir computing: noise effects in a combined analog and digital implementation”. In: *IEEE Transactions on Neural Networks and Learning Systems* 26.2 (2014), pp. 388–393.

[63] Peter Petre and Jose Cruz-Albrecht. “Neuromorphic mixed-signal circuitry for asynchronous pulse processing”. In: *2016 IEEE International Conference on Rebooting Computing (ICRC)*. IEEE. 2016, pp. 1–4.

[64] Chenyuan Zhao et al. “Novel spike based reservoir node design with high performance spike delay loop”. In: *Proceedings of the 3rd ACM International Conference on Nanoscale Computing and Communication*. 2016, pp. 1–5.

[65] Jialing Li et al. “Analog hardware implementation of spike-based delayed feedback reservoir computing system”. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2017, pp. 3439–3446.

[66] David Verstraeten, Benjamin Schrauwen, and Dirk Stroobandt. “Reservoir computing with stochastic bitstream neurons”. In: *Proceedings of the 16th Annual Prorisc Workshop*. Citeseer. 2005, pp. 454–459.

[67] Amos R Omondi. *FPGA implementations of neural networks*. Springer, 2006.

[68] Piotr Antonik et al. “FPGA implementation of reservoir computing with online learning”. In: *24th Belgian-Dutch Conference on Machine Learning*. 2015, p. 99.

[69] Miquel L Alomar et al. “Digital implementation of a single dynamical node reservoir computer”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 62.10 (2015), pp. 977–981.

[70] Bogdan Penkovsky, Laurent Larger, and Daniel Brunner. “Efficient design of hardware-enabled reservoir computing in FPGAs”. In: *Journal of Applied Physics* 124.16 (2018), p. 162101.

[71] Chunxiao Lin, Yibin Liang, and Yang Yi. “Fpga-based reservoir computing with optimized reservoir node architecture”. In: *2022 23rd International Symposium on Quality Electronic Design (ISQED)*. IEEE. 2022, pp. 1–6.

[72] Aya N Elbedwehy et al. “FPGA-based reservoir computing system for ECG denoising”. In: *Microprocessors and Microsystems* 91 (2022), p. 104549.

[73] Cong Shi et al. “Ghost Reservoir: A Memory-Efficient Low-Power and Real-Time Neuromorphic Processor of Liquid State Machine With On-Chip Learning”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 71.10 (2024), pp. 4526–4530.

[74] Steve B Furber et al. “The spinnaker project”. In: *Proceedings of the IEEE* 102.5 (2014), pp. 652–665.

[75] Alberto Patiño-Saucedo et al. “Liquid state machine on spinnaker for spatio-temporal classification tasks”. In: *Frontiers in Neuroscience* 16 (2022), p. 819063.

[76] Oscar I Alvarez-Canchila et al. “Optimizing Reservoir Separability in Liquid State Machines for Spatio-Temporal Classification in Neuromorphic Hardware”. In: *Journal of Low Power Electronics and Applications* 15.1 (2025), p. 4.

[77] Filipp Akopyan et al. “Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip”. In: *IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems* 34.10 (2015), pp. 1537–1557.

[78] Mike Davies et al. “Loihi: A neuromorphic manycore processor with on-chip learning”. In: *IEEE Micro* 38.1 (2018), pp. 82–99.

[79] Ramashish Gaurav, Terrence C Stewart, and Yang Yi. “Reservoir based spiking models for univariate Time Series Classification”. In: *Frontiers in Computational Neuroscience* 17 (2023), p. 1148284.

[80] Denis Kleyko et al. “Principled neuromorphic reservoir computing”. In: *Nature Communications* 16.1 (2025), p. 640.

[81] Saber Moradi et al. “A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs)”. In: *IEEE Transactions on Biomedical Circuits and Systems* 12.1 (2017), pp. 106–122.

[82] Yongqiang Ma et al. “EMG-based gestures classification using a mixed-signal neuromorphic processing system”. In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 10.4 (2020), pp. 578–587.

[83] Ole Richter et al. “DYNAP-SE2: a scalable multi-core dynamic neuromorphic asynchronous spiking neural network processor”. In: *Neuromorphic Computing and Engineering* 4.1 (2024), p. 014003.

[84] Sung Hyun Jo et al. “Nanoscale memristor device as synapse in neuromorphic systems”. In: *Nano Letters* 10.4 (2010), pp. 1297–1301.

[85] Chao Du et al. “Reservoir computing using dynamic memristors for temporal information processing”. In: *Nature Communications* 8.1 (2017), p. 2204.

[86] John Moon et al. “Temporal data classification and forecasting using a memristor-based reservoir computing system”. In: *Nature Electronics* 2.10 (2019), pp. 480–487.

[87] Kristof Vandoorne et al. “Toward optical signal processing using photonic reservoir computing”. In: *Optics Express* 16.15 (2008), pp. 11182–11192.

[88] Guy Van der Sande, Daniel Brunner, and Miguel C Soriano. “Advances in photonic reservoir computing”. In: *Nanophotonics* 6.3 (2017), pp. 561–576.

[89] S Abreu et al. “A photonics perspective on computing with physical substrates”. In: *Reviews in Physics* (2024), p. 100093.

[90] Yvan Paquot et al. “Optoelectronic reservoir computing”. In: *Scientific Reports* 2.1 (2012), p. 287.

[91] Kristof Vandoorne et al. “Experimental demonstration of reservoir computing on a silicon photonics chip”. In: *Nature Communications* 5.1 (2014), p. 3541.

[92] Daniel Brunner and Ingo Fischer. “Reconfigurable semiconductor laser networks based on diffractive coupling”. In: *Optics Letters* 40.16 (2015), pp. 3854–3857.

[93] A Hurtado et al. “Investigation of vertical cavity surface emitting laser dynamics for neuromorphic photonic systems”. In: *Applied Physics Letters* 100.10 (2012), p. 103703.

[94] Laurent Larger et al. “Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing”. In: *Optics Express* 20.3 (2012), pp. 3241–3249.

[95] François Duport et al. “All-optical reservoir computing”. In: *Optics Express* 20.20 (2012), pp. 22783–22795.

[96] Daniel Brunner et al. “Parallel photonic information processing at gigabyte per second data rates using transient states”. In: *Nature Communications* 4.1 (2013), p. 1364.

[97] Ian Bauwens et al. “Use of a Simple Passive Hardware Mask to Replace the Digital Masking Procedure in Photonic Delay-Based Reservoir Computing”. In: *IEEE Journal of Selected Topics in Quantum Electronics* (2024), pp. 1–13.

[98] François Duport et al. “Fully analogue photonic reservoir computer”. In: *Scientific Reports* 6.1 (2016), p. 22381.

[99] Lina Jaurigue and Kathy Lüdge. “Reducing reservoir computer hyperparameter dependence by external timescale tailoring”. In: *Neuromorphic Computing and Engineering* 4.1 (2024), p. 014001.

[100] Kosuke Takano et al. “Compact reservoir computing with a photonic integrated circuit”. In: *Optics Express* 26.22 (2018), pp. 29424–29439.

[101] George B Moody and Roger G Mark. “The impact of the MIT-BIH arrhythmia database”. In: *IEEE Engineering in Medicine and Biology Magazine* 20.3 (2001), pp. 45–50.

[102] Gari D Clifford et al. “Classification of normal/abnormal heart sound recordings: The PhysioNet/Computing in Cardiology Challenge 2016”. In: *2016 Computing in Cardiology Conference (CinC)*. IEEE. 2016, pp. 609–612.

[103] Matthew A Reyna et al. “Heart murmur detection from phonocardiogram recordings: The george b. moody physionet challenge 2022”. In: *PLOS Digital Health* 2.9 (2023), e0000324.

[104] B Moody et al. *MIMIC-III Waveform Database Matched Subset (version1. 0). PhysioNet*. 2020.

[105] Manfredo Atzori et al. “Electromyography data for non-invasive naturally-controlled robotic hand prostheses”. In: *Scientific Data* 1.1 (2014), pp. 1–13.

[106] Stefano Pizzolato et al. “Comparison of six electromyography acquisition setups on hand movement classification tasks”. In: *PloS One* 12.10 (2017), e0186132.

[107] Francesca Palermo et al. “Repeatability of grasp recognition for robotic hand prosthesis control based on sEMG data”. In: *2017 International Conference on Rehabilitation Robotics (ICORR)*. IEEE. 2017, pp. 1154–1159.

[108] Agamemnon Krasoulis et al. “Improved prosthetic hand control with concurrent use of myoelectric and inertial measurements”. In: *Journal of Neuroengineering and Rehabilitation* 14 (2017), pp. 1–14.

[109] Agamemnon Krasoulis, Sethu Vijayakumar, and Kianoush Nazarpour. “Effect of user practice on prosthetic finger control with an intuitive myoelectric decoder”. In: *Frontiers in Neuroscience* 13 (2019), p. 891.

[110] Néstor J Jarque-Bou, Manfredo Atzori, and Henning Müller. “A large calibrated database of hand movements and grasps kinematics”. In: *Scientific Data* 7.1 (2020), p. 12.

[111] Max Ortiz-Catalan, Rickard Brånemark, and Bo Håkansson. “BioPatRec: A modular research platform for the control of artificial limbs based on pattern recognition algorithms”. In: *Source Code for Biology and Medicine* 8 (2013), pp. 1–18.

[112] Xinyu Jiang et al. “Open access dataset, toolbox and benchmark processing results of high-density surface electromyogram recordings”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 29 (2021), pp. 1035–1046.

[113] Ralph G Andrzejak et al. “Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state”. In: *Physical Review E* 64.6 (2001), p. 061907.

[114] John Guttag. *CHB-MIT Scalp EEG Database*. PhysioNet. 2010. URL: https://physionet.org/content/chbmit/1.0.0/.

[115] Wei-Long Zheng and Bao-Liang Lu. “Investigating critical frequency bands and channels for EEG-based emotion recognition with deep neural networks”. In: *IEEE Transactions on Autonomous Mental Development* 7.3 (2015), pp. 162–175.

[116] Miquel Alfaras, Miguel C Soriano, and Silvia Ortín. “A fast machine learning model for ECG-based heartbeat classification and arrhythmia detection”. In: *Frontiers in Physics* 7 (2019), p. 103.

[117] Koksoon Phua et al. “Heart sound as a biometric”. In: *Pattern Recognition* 41.3 (2008), pp. 906–919.

[118] Haobo Li et al. “Remote optical sensing of heart sounds for biometric identification using information fusion”. In: *IEEE Transactions on Instrumentation and Measurement* 73 (2024), pp. 1–12.

[119] Divyam Sharma et al. “Halide perovskite photovoltaics for in-sensor reservoir computing”. In: *Nano Energy* 129 (2024), p. 109949.

[120] Lin Guo, Zongxing Lu, and Ligang Yao. “Human-machine interaction sensing technology based on hand gesture recognition: A review”. In: *IEEE Transactions on Human-Machine Systems* 51.4 (2021), pp. 300–309.

[121] Siming Zuo et al. “Ultrasensitive magnetoelectric sensing system for pico-tesla magnetomyography”. In: *IEEE Transactions on Biomedical Circuits and Systems* 14.5 (2020), pp. 971–984.

[122] Siming Zuo et al. “Miniaturized magnetic sensors for implantable magnetomyography”. In: *Advanced Materials Technologies* 5.6 (2020), p. 2000185.

[123] Negin Ghahremani Arekhloo et al. “Alignment of magnetic sensing and clinical magnetomyography”. In: *Frontiers in Neuroscience* 17 (2023), p. 1154572.

[124] Cory Merkel et al. “Memristive reservoir computing architecture for epileptic seizure detection”. In: *Procedia Computer Science* 41 (2014), pp. 249–254.

[125] Dhireesha Kudithipudi et al. “Design and analysis of a neuromemristive reservoir computing architecture for biosignal processing”. In: *Frontiers in Neuroscience* 9 (2016), p. 502.

[126] Nikhil Garg et al. “Signals to spikes for neuromorphic regulated reservoir computing and EMG hand gesture recognition”. In: *International Conference on Neuromorphic Systems 2021*. 2021, pp. 1–8.

[127] Rahma Fourati et al. “Unsupervised learning in reservoir computing for EEG-based emotion recognition”. In: *IEEE Transactions on Affective Computing* 13.2 (2020), pp. 972–984.

[128] Il Memming Park et al. “Kernel methods on spike train space for neuroscience: a tutorial”. In: *IEEE Signal Processing Magazine* 30.4 (2013), pp. 149–160.

[129] Yujie Wu et al. “Spatio-temporal backpropagation for training high-performance spiking neural networks”. In: *Frontiers in Neuroscience* 12 (2018), p. 331.

[130] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. “Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks”. In: *IEEE Signal Processing Magazine* 36.6 (2019), pp. 51–63.

[131] Jason K Eshraghian et al. “Training spiking neural networks using lessons from deep learning”. In: *Proceedings of the IEEE* 111.9 (2023), pp. 1016–1054.

[132] Yang Dan and Mu-ming Poo. “Spike timing-dependent plasticity of neural circuits”. In: *Neuron* 44.1 (2004), pp. 23–30.

[133] Peter U Diehl and Matthew Cook. “Unsupervised learning of digit recognition using spike-timing-dependent plasticity”. In: *Frontiers in Computational Neuroscience* 9 (2015), p. 99.

[134] Lyes Khacef et al. “Spike-based local synaptic plasticity: A survey of computational models and neuromorphic circuits”. In: *Neuromorphic Computing and Engineering* 3.4 (2023), p. 042001.

[135] Daniel Brunner et al. “Roadmap on Neuromorphic Photonics”. In: *arXiv preprint arXiv: 2501.07917* (2025).

[136] Thomas Klotz, Leonardo Gizzi, and Oliver Röhrle. “Investigating the spatial resolution of EMG and MMG based on a systemic multi-scale model”. In: *Biomechanics and Modeling in Mechanobiology* 21.3 (2022), pp. 983–997.

[137] Justus Marquetand et al. “Optically pumped magnetometers reveal fasciculations non-invasively”. In: *Clinical Neurophysiology* 132.10 (2021), pp. 2681–2684.

[138] Gunnar Farnebäck. “Two-frame motion estimation based on polynomial expansion”. In: *Image Analysis: 13th Scandinavian Conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 Proceedings 13*. Springer. 2003, pp. 363–370.

[139] Mohammed Abo-Zahhad, Sabah M Ahmed, and Sherif N Abbas. “Biometric authentication based on PCG and ECG signals: present status and future directions”. In: *Signal, Image and Video Processing* 8 (2014), pp. 739–751.

[140] Aubrey Leatham. “Phonocardiography”. In: *British medical bulletin* 8.4 (1952), pp. 333–342.

[141] Zeev Zalevsky et al. “Simultaneous remote extraction of multiple speech sources and heart beats from secondary speckles pattern”. In: *Optics Express* 17.24 (2009), pp. 21566–21580.

[142] Lucrezia Cester et al. “Remote laser-speckle sensing of heart sounds for health assessment and biometric identification”. In: *Biomedical Optics Express* 13.7 (2022), pp. 3743–3750.

[143] Silvio Bianchi. “Vibration detection by observation of speckle patterns”. In: *Applied Optics* 53.5 (2014), pp. 931–936.

[144] Takhellambam Gautam Meitei, Sinam Ajitkumar Singh, and Swanirbhar Majumder. “PCG-Based Biometrics”. In: *Handbook of Research on Information Security in Biomedical Signal Processing*. IGI Global, 2018, pp. 1–25.

[145] Shekh MM Islam and Victor M Lubecke. “BreathID: Radar’s New Role in Biometrics”. In: *IEEE Aerospace and Electronic Systems Magazine* 36.12 (2021), pp. 16–23.

[146] CO Folorunso, OS Asaolu, and OP Popoola. “A review of voice-base person identification: State-of-the-art”. In: *Covenant Journal of Engineering Technology* (2019).

[147] Catherine D Schuman et al. “Opportunities for neuromorphic computing algorithms and applications”. In: *Nature Computational Science* 2.1 (2022), pp. 10–19.

[148] Adnan Mehonic and Anthony J Kenyon. “Brain-inspired computing needs a master plan”. In: *Nature* 604.7905 (2022), pp. 255–260.

[149] Linfeng Sun et al. “In-sensor reservoir computing for language learning via two-dimensional memristors”. In: *Science Advances* 7.20 (2021), eabg1455.

[150] Lixiang Li et al. “A review of face recognition technology”. In: *IEEE Access* 8 (2020), pp. 139110–139120.

[151] Nan Wu and S Haruyama. “Real-time audio detection and regeneration of moving sound source based on optical flow algorithm of laser speckle images”. In: *Optics Express* 28.4 (2020), pp. 4475–4488.

[152] Herbert Jaeger. “Short term memory in echo state networks”. In: (2001).

[153] Qianli Ma et al. “Direct model of memory properties and the linear reservoir topologies in echo state networks”. In: *Applied Soft Computing* 22 (2014), pp. 622–628.

[154] Yoma Kuriki et al. “Impact of input mask signals on delay-based photonic reservoir computing with semiconductor lasers”. In: *Optics Express* 26.5 (2018), pp. 5777–5788.

[155] Wentao Wei et al. “Surface-electromyography-based gesture recognition by multi-view deep learning”. In: *IEEE Transactions on Biomedical Engineering* 66.10 (2019), pp. 2964–2973.

[156] Akram Fatayer, Wenpeng Gao, and Yili Fu. “sEMG-based gesture recognition using deep learning from noisy labels”. In: *IEEE Journal of Biomedical and Health Informatics* 26.9 (2022), pp. 4462–4473.

[157] Mostafa Rahimi Azghadi et al. “Hardware implementation of deep network accelerators towards healthcare and biomedical applications”. In: *IEEE Transactions on Biomedical Circuits and Systems* 14.6 (2020), pp. 1138–1159.

[158] Elisa Donati and Giacomo Indiveri. “Neuromorphic bioelectronic medicine for nervous system interfaces: from neural computational primitives to medical applications”. In: *Progress in Biomedical Engineering* 5.1 (2023), p. 013002.

[159] Logan G Wright et al. “Deep physical neural networks trained with backpropagation”. In: *Nature* 601.7894 (2022), pp. 549–555.

[160] Kashu Yamazaki et al. “Spiking neural networks and their applications: A review”. In: *Brain Sciences* 12.7 (2022), p. 863.

[161] Wolfgang Maass. “Networks of spiking neurons: the third generation of neural network models”. In: *Neural Networks* 10.9 (1997), pp. 1659–1671.

[162] Amirhossein Tavanaei et al. “Deep learning in spiking neural networks”. In: *Neural Networks* 111 (2019), pp. 47–63.

[163] Antonio Vitale et al. “Neuromorphic edge computing for biomedical applications: Gesture classification using emg signals”. In: *IEEE Sensors Journal* 22.20 (2022), pp. 19490–19499.

[164] Federico Corradi and Giacomo Indiveri. “A neuromorphic event-based neural recording system for smart brain-machine-interfaces”. In: *IEEE Transactions on Biomedical Circuits and Systems* 9.5 (2015), pp. 699–709.

[165] Minhao Yang et al. “A 0.5 V 55 μW 64 $\times$ 2 Channel Binaural Silicon Cochlea for Event-Driven Stereo-Audio Sensing”. In: *IEEE Journal of Solid-State Circuits* 51.11 (2016), pp. 2554–2569.

[166] Marcello Zanghieri et al. “Event-based Estimation of Hand Forces from High-Density Surface EMG on a Parallel Ultra-Low-Power Microcontroller”. In: *IEEE Sensors Journal* 25.5 (2024), pp. 7771–7780.

[167] Sizhen Bian, Elisa Donati, and Michele Magno. "Evaluation of Encoding Schemes on Ubiquitous Sensor Signal for Spiking Neural Network". In: *IEEE Sensors Journal* (2024).

[168] Marcello Zanghieri et al. "Event-based low-power and low-latency regression method for hand kinematics from surface EMG". In: *2023 9th International Workshop on Advances in Sensors and Interfaces (IWASI)*. IEEE. 2023, pp. 293–298.

[169] Lauren H. Smith et al. "Determining the Optimal Window Length for Pattern Recognition-Based Myoelectric Control: Balancing the Competing Effects of Classification Error and Controller Delay". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 19.2 (2011), pp. 186–192. DOI: 10.1109/TNSRE.2010.2100828.

[170] Kexiang Li et al. "A review of the key technologies for sEMG-based human-robot interaction systems". In: *Biomedical Signal Processing and Control* 62 (2020), p. 102074.

[171] Ryan Pyle and Robert Rosenbaum. "Spatiotemporal dynamics and reliable computations in recurrent spiking neural networks". In: *Physical Review Letters* 118.1 (2017), p. 018103.

[172] Elisabetta Chicca et al. "Neuromorphic electronic circuits for building autonomous cognitive systems". In: *Proceedings of the IEEE* 102.9 (2014), pp. 1367–1388.

[173] Tobias Delbrueck and C Mead. "Bump circuits". In: *Proceedings of International Joint Conference on Neural Networks*. Vol. 1. Citeseer. 1993, pp. 475–479.

[174] Yi Zhong et al. "An efficient neuromorphic implementation of temporal coding-based on-chip STDP learning". In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 70.11 (2023), pp. 4241–4245.

[175] Metin Akay. *Biomedical signal processing*. Academic Press, 2012.

[176] Yuqi Ding et al. "Event-Driven Implementation of a Physical Reservoir Computing Framework for superficial EMG-based Gesture Recognition". In: *arXiv preprint arXiv: 2503.13492* (2025).

Appendices

A PRC as predictors (Python)

The main Python codes for the SCR network, referring to contents elaborated in Section 3.1.1.

```
''' Setting reservoir parameters '''
inSize = 92
outSize = 4
resSize = 1000
a = 0.8  # leaking rate
trainLen = 40000
testLen = 10000
initLen = 200

''' Random generated Win '''
np.random.seed(42)
Win = (np.random.rand(resSize, inSize) - 0.5) * 1

''' Fixed value Wres - using Simple Cylic Reservoir (SCR) Structure
    '''
r = 0.5 # weight value
W = np.zeros((resSize, resSize))
for i in range(0,resSize-1):
    W[i+1,i] = r
W[0, resSize - 1] = r
rhoW = max(abs(linalg.eig(W)[0]))

''' Allocated memory for the design (collected states) matrix '''
X = np.zeros((inSize + resSize, trainLen - initLen))
# set the corresponding target matrix directly
Yt = data_EMG[:, initLen :trainLen ]

''' Run the reservoir with the data and collect X '''
```

```
x = np.zeros((resSize, 1))
for t in range(trainLen):
    u = data_MMG[:,t]
    u = u.reshape(92,1)
    x = (1 - a) * x + a * np.tanh(np.dot(Win, u) + np.dot(W, x))
    if t >= initLen:
        X[:, t - initLen] = np.vstack(( u, x))[:, 0]

''' Calculate Wout '''
reg = 1e-10
Wout = linalg.solve(np.dot(X, X.T) + reg * np.eye(inSize + resSize),
                    np.dot(X, Yt.T)).T

''' Use the trained reservoir to obtain predicted output based on
   test data input '''
Y = np.zeros((outSize, testLen))
X = np.zeros((inSize + resSize, testLen))
for t in range(testLen):
    u = data_MMG[:,t+trainLen]
    u = u.reshape(92,1)
    x = (1 - a) * x + a * np.tanh(np.dot(Win, u) + np.dot(W, x))
    X[:, t] = np.vstack((u, x))[:, 0]

''' Use Wout and collected states in test data to predict the output
   value Y '''
Y = np.dot(Wout,X)

''' The function to calculate the mse and NRMSE in each channel, also
    plot the camparison figure in each channel '''
def error_calculate(data_EMG,Y,trainLen,errorLen):
    for i in range(np.size(data_EMG,0)):
        mse = sum(np.square(data_EMG[i,trainLen :trainLen + errorLen]
            -
                    Y[i, 0:errorLen])) / errorLen
        NRMSE =  np.sqrt(np.divide(
        np.mean(np.square(data_EMG[i,trainLen : trainLen + errorLen]-
           Y[i, 0:errorLen])),
        np.var(data_EMG[i,trainLen : trainLen + errorLen])))
        print('EMG channel '+str(i+1)+'\n mse:'+ str(mse) + '\n NRMSE
           :' + str(NRMSE) + '\n')
        plt.figure(i).clear()
```

```
        plt.plot(data_EMG[i,trainLen:trainLen + testLen - 500], 'g')
        plt.plot(Y[i,0:testLen - 500].T, 'b')
        plt.title('channel ' + str(i+1) + ' Target and generated
            signals $y(n)$ starting at $n=0$' + '\n NRMSE = ' + str(
            round(NRMSE,4)))
        plt.legend(['Target signal', 'Free-running predicted signal'
            ])

''' Calculate prediction errors '''
errorLen = 1000
error_calculate(data_EMG,Y,trainLen,errorLen)
```

Listing 1: Python codes for SCR network (EMG/MMG mapping)

B PRC as predictors (MATLAB)

The main MATLAB codes for RNR network, referring to contents elaborated in Section 3.1.2.

```
% Setting reservoir parameters
N = 400;
rand( 'seed', 42 );
Win = (zeros(N,1)+1) .* (round(rand(N,1))*2 - 1);
tau_n = 1;
tau_r = tau_n/8;
C   = 10e-6;
R1 = tau_n/C;
R2 = 1000e3;
Is = 25e-9;
gama = 0.05;
offset= 0;
Len_train = size(traindata,1)*size(traindata,2);
Len_test = size(testdata,1)*size(testdata,2);
Len_init = 100;

% Simulink module setttings
SimulationTime = tau_r*(Len_train+Len_test);
x = (0:tau_r:tau_r*((Len_train+Len_test)-1))';
dataMask = [traindata;testdata]'.*Win*gama + offset;
for i = 1:length(dataMask) %% pre-neuron rotation
   dataShifted(:,(i-1)+1:(i-1)+1) = circshift(dataMask(:,i),i-1)-0.1;
end
```

```
theta=tau_r;
dataIN = dataShifted';
sim('Neuron.slx'); %% dynamic neuron
ak = dataOUT(2:end,:)';

% Collected states after rotation
sk = zeros(N,(Len_train+Len_test));
for t = 1:(Len_train+Len_test) %% post-neuron rotation
    sk(:,t) = circshift(ak(:,t),-t+1);
end

% Calculate output weight matrix Wout
ahead = 0;
reg = 1e-10;  % regularization coefficient
target = traintarget(Len_init+1+ahead:Len_train+ahead)';
trainingState = sk(:,Len_init+1:Len_train);
Wout = (target*trainingState' / (trainingState*trainingState' + reg*
   eye(N)))';

% Calculate the predicted outputs
testTarget = testtarget(1+ahead:Len_test+ahead);
testingStates = sk(:,Len_train+1:Len_test+Len_train);
output = testingStates'*Wout;

% Calculate the prediction errors
NRMSE = sqrt(mean((output(Len_init+1:end)-testTarget(Len_init+1:end))
   .^2)./var(testTarget(Len_init+1:end)));
disp(['NRMSE = ' num2str(NRMSE)])
```

Listing 2: MATLAB codes for RNR network (ECG-to-PCG prediction)

C PRC for biometric identification (MATLAB)

The main MATLAB codes for RNR network in a biometric identification task based on human heart sounds, referring to contents elaborated in Chapter 4.1.

```
% Parameter settings
N = 500;
Nres = 2; % number of parallel reservoirs -> refer to M in the thesis
tau_n = 1;
tau_r = tau_n/8;
```

```
theta = tau_r;
C  = 10e-6;
R1 = tau_n/C;
R2 = 1000e3;
Is = 25e-9;

% Define input weight matrix Win
rand( 'seed', 42 );
for i  = 1:Nres
    signal = round(rand(N,1))*2 - 1;
    Win(:,i) = (zeros(N,1)+1) .* signal;
end

% Parallel reservoirs
% The function for parallel RNR
function [dataIN,SimulationTime,x,Len_train,Len_test,Len_init] = RNR(
   traindata, testdata,tau_n,Win)
tau_r = tau_n/8;
gama = 0.05;
offset= 0.1;
Len_train = size(traindata,1)*size(traindata,2);
Len_test = size(testdata,1)*size(testdata,2);
Len_init = 100;
SimulationTime = tau_r*(Len_train+Len_test);
x = (0:tau_r:tau_r*((Len_train+Len_test)-1))';
dataMask = [traindata;testdata]'.*Win*gama + offset;
for i = 1:length(dataMask) %% pre-neuron rotation
   dataShifted(:,(i-1)+1:(i-1)+1) = circshift(dataMask(:,i),i-1)-0.1;
end
dataIN = dataShifted';
end

% Apply parallel reservoirs on the inputs
[dataIN,SimulationTime,x,Len_train,Len_test,Len_init] = RNR(traindata
   , testdata,tau_n,Win(:,1));
sim('Neuron.slx'); % dynamic neuron
ak1 = dataOUT(2:end,:)';
sk1 = zeros(N,(Len_train+Len_test));
for t = 1:(Len_train+Len_test) %% post-neuron rotation
    sk1(:,t) = circshift(ak1(:,t),-t+1);
end
```

```
45
46 [dataIN,SimulationTime,x,Len_train,Len_test,Len_init] = RNR(traindata
      , testdata,tau_n,Win(:,2));
47 sim('Neuron.slx'); % dynamic neuron
48 ak2 = dataOUT(2:end,:)';
49 sk2 = zeros(N,(Len_train+Len_test));
50 for t = 1:(Len_train+Len_test) %% post-neuron rotation
51     sk2(:,t) = circshift(ak2(:,t),-t+1);
52 end
53
54 % Stack the states from each reservoir
55 sk = [sk1;sk2];
56
57 % Calculate the output weight matrix Wout
58 ahead = 0;
59 reg = 1e-8;  % regularization coefficient
60 target = trainlabel(:,Len_init+1+ahead:Len_train+ahead);
61 trainingState = sk(:,Len_init+1:Len_train);
62 Wout = (target*trainingState' / (trainingState*trainingState' + reg*
      eye(N*Nres)))';
63
64 % Testing
65 testTarget = testlabel(:,1+ahead:Len_test+ahead);
66 testingStates = sk(:,Len_train+1:Len_test+Len_train);
67 output = testingStates'*Wout;
68
69 % Find the peak and calculate the acc
70 [m,n] = find(testlabel_old==1);
71 num_sample = length(m);
72 real = zeros(1,num_sample);
73 label = zeros(1,num_sample);
74 shift = 10;
75 for i = 1:num_sample
76     yloc = n(i)-shift;
77     xloc = n(i)-shift-70;
78     seg_signal = output(xloc:yloc,:);
79     max_signal = max(seg_signal,[],1);
80     [~,id] = max(max_signal);
81     real(i) = id;
82     label(i) = m(i);
83 end
```

```
acc = sum(real == label);
acc = acc/num_sample;
```

Listing 3: MATLAB codes for RNR network (biometric identification)

D Spike encoding for sEMG data (MATLAB)

The main MATLAB codes for spike encoding of sEMG signals, refering to contents elaborated in Section 5.2.

```
% Setting window length
fs = 2e3
window_len = 0.25*fs; % 200ms

% Setting 4 encoded spike trains for each of the 12 channels of sEMG
   signals
input = emg(:,1);
L = size(input,1);
spike_train = zeros(L+1, num_chs*4);

% Nomalization of original signals
for k = 1:num_chs
input = emg(:,k);
min_val = min(input);
max_val = max(input);
normalized_input = (input - min_val) / (max_val - min_val);

% Setting cutoff frequencies
cutoff1 = 999/ratio;
cutoff2 = 500/ratio;
cutoff3 = 250/ratio;
cutoff4 = 100/ratio;

order = 4; % The order of butterworth filter

% The function for highpass butterworth filter
function filtered_signal = lopass_butterworth(inputsignal,cutoff_freq
   ,Fs,order)
Wn = 2*cutoff_freq/Fs; % non-dimensional frequency
[filtb,filta] = butter(order,Wn,'high'); % construct the filter
filtered_signal = filter(filtb,filta,inputsignal); % filter the data
```

```
    with zero phase
end

% The function for bandpass butterworth filter
function filtered_signal = bandpass_butterworth(inputsignal,
    cutoff_freqs,Fs,order)
Wn = 2*cutoff_freqs/Fs; % non-dimensional frequency
[filtb,filta] = butter(order,Wn,'bandpass'); % construct the filter
filtered_signal = filter(filtb,filta,inputsignal); % filter the data
    with zero phase
end

% The function for lowpass butterworth filter
function filtered_signal = lopass_butterworth(inputsignal,cutoff_freq
    ,Fs,order)
Wn = 2*cutoff_freq/Fs; % non-dimensional frequency
[filtb,filta] = butter(order,Wn,'low'); % construct the filter
filtered_signal = filter(filtb,filta,inputsignal); % filter the data
    with zero phase
end

% Filter each channel of the sEMG signasl to different freqency bands
y_filt1 = hipass_butterworth(normalized_input,cutoff2,fs,order);
y_filt2 = bandpass_butterworth(normalized_input,[cutoff3 cutoff2],fs,
    order);
y_filt3 = bandpass_butterworth(normalized_input,[cutoff4 cutoff3],fs,
    order);
y_filt4 = lopass_butterworth(normalized_input,cutoff4,fs,order);

% The function for full-wave rectifier
function rectified_sig = full_rectifier(input_sig)
rectified_sig = abs(input_sig);
end

% Full-rectify the filtered signals
t = (1:length(normalized_input))/fs; % time
y_rectif1 = full_rectifier(y_filt1);
y_rectif2 = full_rectifier(y_filt2);
y_rectif3 = full_rectifier(y_filt3);
y_rectif4 = full_rectifier(y_filt4);

```

```
% Spike coding through LIF neuron
v_th = [8e-6 3e-5 8e-5 5e-3 ...
    8e-6 3e-5 8e-5 6e-3 ...
    3e-6 1e-5 2e-5 5e-3 ...
    5e-6 1e-5 4e-5 7e-3 ...
    1e-5 2e-5 6e-5 7e-3 ...
    1e-5 3e-5 5e-5 4e-3 ...
    5e-6 2e-5 5e-5 4e-3 ...
    2e-5 5e-5 1e-4 4e-3 ...
    1e-5 5e-5 5e-5 5e-3 ...
    4e-6 3e-5 7e-5 3e-3 ...
    25e-6 5e-5 7e-5 3e-3 ...
    1e-5 5e-5 7e-5 3e-3]; % voltage threshold, which is fine-tuned

v0 = 0; % reset voltage
dt = 1/fs;  % sampling time
v_rest = 0; % reset voltage
tau = 1; % time constant

% The function for LIF ODE
function [v,spk] = LIF_ODE(v_th,v_rest,v0, dt, I, tau)
v = v0 + (dt / tau) * (-(v0 - v_rest) + I);
spk = false;
if v >= v_th
    v = v_rest;
    spk = true;
end
end

% The function for generating spike trains through LIF neurons
function [spike_train,T] = spikeLIFcoding(v_th, I_ext, v0, dt, v_rest
   , tau)
n_tSteps = length(I_ext) + 1;
V = zeros(n_tSteps,1);
V(1) = v0;
T = zeros(n_tSteps,1);
spike_train = zeros(n_tSteps,1);
for i=1:n_tSteps -1
    [v,spk] = LIF_ODE(v_th,v_rest,v0, dt, I_ext(i),tau);
    T(i+1) = T(i)+dt;
    V(i+1) = v;
```

```
    if spk == true
        spike_train(i+1) = 1;
    end
    v0 = v;
end

% generating spike trains through LIF neurons
[spike_train1, T] = spikeLIFcoding(v_th(4*k-3), y_rectif1, v0, dt,
   v_rest, tau);
[spike_train2, T] = spikeLIFcoding(v_th(4*k-2), y_rectif2, v0, dt,
   v_rest, tau);
[spike_train3, T] = spikeLIFcoding(v_th(4*k-1), y_rectif3, v0, dt,
   v_rest, tau);
[spike_train4, T] = spikeLIFcoding(v_th(4*k), y_rectif4, v0, dt,
   v_rest, tau);

% Construct the spike trains
spike_train(:,4*k-3) = spike_train1;
spike_train(:,4*k-2) = spike_train2;
spike_train(:,4*k-1) = spike_train3;
spike_train(:,4*k) = spike_train4;
end
```

Listing 4: MATLAB codes for the spike encoding of sEMG data

E Spiking RNR for gesture recognition (Python)

The main Python codes for spiking RNR in gesture recognition, referring to contents elaborated in Section 5.3 and Section 5.4.

```
''' Initialize the parameters '''
num_inputs = 48
resSize = 10
num_outspk = num_inputs*resSize
num_outputs = 17 # refer to 17 gestures in exercise B, change to 23
   for exercise A and 9 for exercise C, respectively
tau_r1 = 1e-3
R1 = 5
C1 = 3e-3
trh1 = 0.5
tau_r2 = 2e-3
```

```
R2 = 5
C2 = 5e-3
trh2 = 0.3

''' Initialize LIF neurons '''
lif1 = snn.Lapicque(R=R1, C=C1, time_step=tau_r1, threshold = trh1)
lif2 = snn.Lapicque(R=R2, C=C2, time_step=tau_r2, threshold = trh2)
lif = [lif1, lif2]
np.random.seed(42)
lif_index = np.random.randint(0,2,size = num_inputs)

''' Define the sRNR network '''
def sRNR(spiketrain,resSize,lif):
    spiketrain = np.reshape(spiketrain,(len(spiketrain),-1))

    # input masking by randomly chosen {0,1}
    np.random.seed(42)
    Win  = np.random.randint(0,2,size = (1,resSize))
    dataMask = np.dot(spiketrain,Win)
    dataMask = np.transpose(dataMask)
    num_steps = dataMask.shape[1]
    dataShifted = np.zeros((resSize, num_steps))
    for i in range(num_steps):
        dataShifted[:,i] = np.roll(dataMask[:,i], i)
    cur_in = torch.tensor(dataShifted[:,:])
    mem = torch.zeros((1,resSize))
    spk_out = torch.zeros((1,resSize))
    mem_rec = [mem]
    spk_rec = [spk_out]
    for step in range(num_steps):
        spk_out, mem = lif(cur_in[:,step], mem)
        # Store recordings
        mem_rec.append(mem)
        spk_rec.append(spk_out)

    # convert the list of tensors into one tensor
    mem_rec = torch.stack(mem_rec)
    spk_rec = torch.stack(spk_rec)
    mem_rec = mem_rec.numpy()
    spk_rec = spk_rec.numpy()
    mem_rec =  mem_rec.squeeze()
```

```
52      spk_rec = spk_rec.squeeze()
53      spk_rec = spk_rec.T
54      sk = np.zeros((resSize,num_steps))
55      for t in range(num_steps):
56          sk[:,t] = np.roll(spk_rec[:,t],-t)
57      sk_T = sk.T
58      sk_T = torch.from_numpy(sk_T)
59      mem_rec = torch.from_numpy(mem_rec)
60      return sk_T,mem_rec
61
62  ''' Spikes injected to the reservoir and receive the genereted output
        spikes '''
63  num_sample = num_outputs*24
64  data_spk = np.zeros((num_sample,400,num_outspk))
65  data_mem = np.zeros((num_sample,401,num_outspk))
66  for j in range(len(data)):
67      spike = data[j,0]
68      spk = torch.zeros(0)
69      mem = torch.zeros(0)
70      for k in range(num_inputs):
71          spiketrain = spike[:,k]
72          spk_rec, mem_rec = sRNR(spiketrain,resSize,lif[lif_index[
                num_inputs-1]])
73          spk = torch.cat((spk,spk_rec), dim=1)
74          mem = torch.cat((mem,mem_rec), dim=1)
75      spk = spk.numpy()
76      mem = mem.numpy()
77      data_spk[j,:,:] = spk
78      data_mem[j,:,:] = mem
79
80  ''' Apply a window to bin the signal (count and bin kernel) '''
81  def bin_signal(window_size, signal,step_size):
82      bins = []
83      for i in range(0, len(signal) - window_size + 1,step_size):
84          window = signal[i:i + window_size,:]
85          val = np.sum(window, axis=0)
86          bins.append(val)
87      bins = np.array(bins)
88      return bins
89  window_size = 400
90  step_size = 400
```

```
fea_len = int(400/window_size)

''' Bin the output spikes (400 features)'''
feature_map = np.zeros((num_sample,fea_len,in_fea))
for i in range(num_sample):
    x = bin_signal(window_size,spike[i,:,:],step_size)
    feature_map[i,:,:] = x

# Reshape the features
X = feature_map
Y = new_label
X = X.squeeze()
Y = Y.squeeze()
Y = Y.astype(int)

''' Classification algorithms in the readout layer '''

''' Delta learning rule '''
# Construct training and testing sets and shuffle the dataset
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled,Y,
   test_size=.2, random_state=42)
y_train_one_hot = np.eye(num_classes)[y_train]
num_features = out_fea  # output features
weights = np.random.randn(num_features, num_classes) * 0.01
biases = np.zeros(num_classes)
learning_rate = 0.005
epochs = 200

# Softmax activation function for classification
def softmax(z):
    exp_z = np.exp(z - np.max(z, axis=1, keepdims=True))
    return exp_z / exp_z.sum(axis=1, keepdims=True)

# Deifne the function to pridict the output
def predict(X, weights, biases):
    linear_output = np.dot(X, weights) + biases
    probabilities = softmax(linear_output)
    return np.argmax(probabilities, axis=1)

```

```
131 # Training loop
132 epoch_lst = []
133 loss_lst = []
134 acc_test_lst = []
135 acc_train_lst = []
136 for epoch in range(epochs):
137     # Forward pass
138     linear_output = np.dot(X_train, weights) + biases
139     output = softmax(linear_output)
140 
141     # Compute the error
142     error = y_train_one_hot - output
143 
144     # Weight and bias updates
145     weights += learning_rate * np.dot(X_train.T, error)
146     biases += learning_rate * error.sum(axis=0)
147 
148     # Compute the loss (cross-entropy)
149     loss = -np.mean(np.sum(y_train_one_hot * np.log(output + 1e-9),
        axis=1))
150 
151     # Print the loss every epoch
152     test_predictions = predict(X_test, weights, biases)
153     train_predictions = predict(X_train, weights, biases)
154     accuracy_test = (test_predictions == y_test).sum() / y_test.shape
        [0]
155     accuracy_train = (train_predictions == y_train).sum() / y_train.
        shape[0]
156     epoch_lst.append(epoch)
157     loss_lst.append(loss)
158     acc_test_lst.append(accuracy_test)
159     acc_train_lst.append(accuracy_train)
160     if (epoch + 1) % 10 == 0:
161         print(f'Epoch {epoch+1}/{epochs}, Loss: {loss:.4f}')
162 
163 # Testing loop (predict and evaluate)
164 y_pred = test_predictions
165 report = classification_report(y_test, y_pred)
166 print(report)
167 
168 ''' SVM classifier '''
```

```
# Construct datasets, change random_state for cross-validation
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_scaled,Y,
    test_size=.2, random_state=42)

# Apply RBF kernel, change to kernel='linear' for linear kernel
classifier = SVC(kernel='rbf', random_state=42,degree=1)
classifier.fit(X_train, y_train)

# Predict and evaluate
y_pred = classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
print(report)

''' Statistical analysis '''
sensitivity = np.zeros(num_classes)
specificity = np.zeros(num_classes)
precision = np.zeros(num_classes)
f1_score = np.zeros(num_classes)
for i in range(num_classes):
    TP = cm[i, i]  # True Positives for class i
    FP = cm[:, i].sum() - TP  # False Positives for class i
    FN = cm[i, :].sum() - TP  # False Negatives for class i
    TN = cm.sum() - (FP + FN + TP)  # True Negatives for class i

    sensitivity[i] = TP / (TP + FN) if (TP + FN) != 0 else 0
    specificity[i] = TN / (TN + FP) if (TN + FP) != 0 else 0
    precision[i] = TP / (TP + FP) if (TP + FP) != 0 else 0
    f1_score[i] = 2 * precision[i] * sensitivity[i] / (precision[i] +
        sensitivity[i]) if (precision[i] + sensitivity[i]) != 0 else
       0
overall_sensitivity_macro = np.mean(sensitivity)
overall_specificity_macro = np.mean(specificity)
overall_precision_macro = np.mean(precision)
overall_f1_macro = np.mean(f1_score)

''' Plot the confusion matrix '''
y_pred = test_predictions
cm = confusion_matrix(y_test, y_pred)
```

```
cm_percentage = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
plt.figure(figsize=(20, 15))
sns.heatmap(cm_percentage, annot=False, fmt='.1f', cmap='Blues',
    xticklabels=[i for i in range(0,50)], yticklabels=[i for i in
    range(0,50)])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()

'''t-SNE visualization - an example for gestures in exercise B from
    input data'''
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# t-SNE settings
tsne = TSNE(n_components=2, random_state=42)
X_tsne = tsne.fit_transform(X_scaled)
p1 = 0
p2 = 1

# Plot the embedded data
plt.figure(figsize=(8,6), dpi=300)
plt.scatter(X_tsne[:, p1], X_tsne[:, p2],c=label, cmap=plt.cm.
    get_cmap("jet", 17)) # 17 refers to 17 gestures
plt.colorbar(ticks=range(18))
plt.title('Exercise B',fontsize = 22)
plt.xlabel('Comp- ' + str(p1+1),fontsize = 18)
plt.ylabel('Comp- '+ str(p2+1),fontsize = 18)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.show()
```

Listing 5: Python codes for the spiking RNR network (gesture recognition)