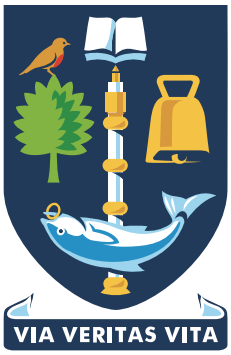
Physical Reservoir Computing with Dynamical Electronics



Xiangpeng Liang

James Watt School of Engineering

University of Glassgow

This dissertation is submitted for the degree of

Doctor of Philosophy

September 2022

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.

Xiangpeng Liang

September 2022

Acknowledgements

First and foremost, I especially appreciate the endless support throughout my life from my father Xuguang Liang, mother Xuefang Zou, brothers Xiangke Liang and Xianghao Liang, and other family members.

I would like to express my gratitude to my supervisor Prof. Hadi Heidari. I met Hadi five years ago when I was a master student. I can still remember the joy I felt when I received my PhD offer and research assistant offer from him. During the past five years, he has given me patient guidance, invaluable advice as well as warm encouragement. His enthusiasm in research always inspires me to fulfill a high-quality work. Next, I would like to thanks my co-supervisors, Dr. Rami Ghannam, Dr. Aleksandra Vuckovic and Dr. Francesco Fioranelli, who are always available whenever I need. They have provided their insightful suggestions to my research. Particularly, at the early stage of my PhD, they checked my manuscripts word by word and captured every technical detail. I cannot get on the track quickly without their helps.

I acknowledge Prof. Themis Prodromakis, Prof. Qammer Abbasi and Dr. Ahmed Taha for recognizing my works in my PhD viva and helping me to improve my PhD thesis.

I would also like to thank all of the members of meLAB (Microelectronics Lab) of UofG, especially Jinwei Zhao, Kaung Oo Htet, Asfand Tanwear, Rupam Das, Zehao Zhang, Yongdian Sun, Weipeng Wang, Yuqi Ding, Antonia Pavlidou, Elie Gautreau, and my good friends in UofG: Yihong Liu, Bowen Yang, Fei Deng, Tao Lyu and Yingke Wang. I am so

thankful for their friendship and those enjoyable moments. Special thanks to my best friends, Adnan Zahid, Mengyao Yuan and Haobo Li, for helping me to get through the ups and downs throughout my PhD.

I have experienced an unusual PhD due to the outbreak of COVID-19. Thankfully, I got a visiting position in LEMON (Laboratory of Emerging MemOry and Novel computing) group of Tsinghua University. I am deeply indebted to Prof. Huaqiang Wu and Prof. Chuhan Zhang, who accepted me as a visiting student. Without their help, I might have to work from home for the second half of my PhD. Instead, I spent one and a half years in Tsinghua, during which Prof. Jianshi Tang was my supervisor. I am extremely grateful to his invaluable supports in all respects when I had difficulties. During my hard time, he supported me and my research in the same way as supporting his own students. His deep understanding in neuromorphic computing impacted me a lot and shaped my way of thinking. In addition, I would definitely thank Dr. Yanan Zhong, who mentored and inspired me significantly. We had lunches and dinners every day, during which we discussed our research, solved technical problems as well as talking about our life. Making our ideas come true was the most enjoyable section of my PhD. Eventually, Prof. Tang and Dr. Yanan Zhong's efforts leveraged my works to a much higher level that I never expected before. Here I would also say thank you to those who have kindly contributed to my project: Zhengwu Liu, Keyang Sun, Dr. Xinyi Li, Dr. Peng Yao, Dr. Qingtian Zhang and Prof. Bin Gao. My works cannot go so smoothly without their generosity and expertise. Besides, as a guest, I received lots of kindness from my Tsinghua friends: Qi Dang, Jian Yu, Zhikai Wang, Dr. Qi Hu, Dr. Heyi Huang, Guofang Yu, Yixin Guan and so many others. I will never forget the time we spent together.

Finally, I would like to thank my loving girlfriend and future wife, Yuchi Liu, for everything she has done for me.

Abstract

Since the advent of data-driven society, mass information generated from human activity and the natural environment has been collected, stored, processed, and then dispersed under conventional von Neumann architecture. However, further scaling the computing capability in terms of speed and power efficiency has been significantly slowed down in recent years due to the fundamental limits of transistors. To meet the increasingly demanding requirement for data-intensive computation, neuromorphic computing is a promising field taking the inspiration from the human brain, an extremely efficient biological computer, to develop unconventional computing paradigms for artificial intelligence.

Reservoir computing, a recurrent neural network algorithm invented two decades ago, has received wide attention in the field of neuromorphic computing because of its unique recurrent dynamics and hardware-friendly implementation schemes. Under the concept of reservoir computing, hardware's intrinsic physical behaviours can be explored as computing resources to keep the machine learning within the physical domain to improve processing efficiency, which is also known as physical reservoir computing.

This thesis focuses on modelling and implementing physical reservoir computing based on dynamical electronics, along with its applications with sensory signals. First, the fundamental of the reservoir computing algorithm is introduced. Second, based on the reservoir algorithm and its functionalities, two different architectures for physically implementing reservoir computing, delay-based reservoir and parallel devices, are investigated to perform temporal

signal processing. Thirdly, an efficient implementation architecture, namely rotating neurons reservoir, is developed. This novel architecture is evaluated in both theoretical analysis and experiments. An electrical prototype of the rotating neurons reservoir exhibits unique advantages such as resource-efficient implementation and low power consumption. More importantly, the theory of rotating neurons reservoir is highly universal, indicating that a rotational object embedded with dynamical elements can act as a reservoir computer.

Table of contents

List of figures

List of tables

Nomenclature

Roman Symbols

τ	Sampling interval of input signal
τ_{Neuron}	Time constant of dynamic neuron circuit
τ_{rotor}	Rotation speed of eRNR
θ	Sampling interval of masked signal
a	Input scaling of a software-based reservoir computing algorithm
a'	Input scaling of a hardware-based RNR
b	Recurrent scaling of a software-based reservoir computing algorithm
b'	Recurrent scaling of a hardware-based RNR
C_{int}	Capacitance of the integrator
E_c^{dyn}	Dynamic energy (per operation) for counter in eRNR
E_t^{dyn}	Dynamic energy (per operation) for transmission gate in eRNR
f	Nonlinear function of reservoir algorithm
f_r	Nonlinear transformation of physical neurons in a hardware-based RNR

Nomenclature

G	Conductance of a memristor
$J(k)$	Masked input signal at k step
M	Number of parallel reservoirs
N	Number of Neurons
N_{delay}	Number of delay lines in delay-based reservoir computing
P	Total power consumption
P_c	Static power consumption for counter in eRNR
P_t	Static power consumption for transmission gate in eRNR
$q(n)$	Response of neurons collected at time step n
R_{int}	Resistance of the integrator
$s(n)$	State vector at time step n
$s_r(n)$	State vector at time step n in a hardware-based RNR
u_n	Discrete input at time step n
$y'(n)$	Predicted output of RC at step n in testing phase
$y(n)$	Target output of RC at step n in training phase, or original input signal
$\mathbf{W_{in}}$	Input layer weights of reservoir
$\mathbf{W_{out}}$	Output layer weights of reservoir
$\mathbf{W_{res}}$	Reservoir layer weights of reservoir

Acronyms / Abbreviations

1T1R One-Transistor-One-Resistor

AAMI Association for the Advancement of Medical Instrumentation

ADC Analogue-to-digital Converter

AI Artificial Intelligence

ANN Artificial Neural Network

ASIC Application-specific Integrated Circuits

C2C Cycle-to-Cycle

CA Computing ability

CMOS Complementary Metal-Oxide-Semiconductor Transistor

CNN Convolutional Neural Network

CR Cyclic Reservoir

CRJ Cycle Reservoir with Jumps

D2D Device-to-Device

DBN Deep Belief Network

DRC Delay-based Reservoir Computing

ECG Electrocardiogram

eRNR electrical Rotating Neurons Reservoir

ESN Echo State Network

FN False Negative

Nomenclature

FP False Positive

GP Generalization Rank

GPU Graphics Processing Units

KQ Kernel Quality

LSM Liquid State Machine

LSTM Long Short-term Memory

MC Memory Capacity

MEMS Micro-electromechanical Systems

MSE Mean-Squared Error

NARMA Nonlinear Autoregressive Moving Average

NRMSE Normalized Root Mean Square Error

NVM Non-volatile Memory

PCA Principal Component Analysis

PC Principal Component

RC Reservoir Computing

ReLU Rectified Linear Unit

RNN Recurrent Neural Network

RNR Rotating Neurons Reservoir

RRAM Resistive Random Access Memory

SMOTE Synthetic Minority Over-sampling TEchnique

SRAM Static Random Access Memory

SVM Support Vector Machine

TN True Negative

TP True Positive

UART Universal Asynchronous Receiver/Transmitter

VEB Ventricular Ectopic Beat

VMM Vector-matrix Multiplication

Chapter 1

Introduction

1.1 Computing resources from electronics

Since the 1950s, we have witnessed explosive growth in the electronics industry, enabling the technological revolution worldwide. In particular, the advances in transistors define how the digital computer automatically stores and processes the information generated by the natural environment and human activities. The semiconductor industry commits to shrinking the size of transistors to decrease the power and cost per device. Over the past half-century, the computing capability of processors has been significantly enhanced as the transistor density increases. The computing tasks that the computer could not handle ten years ago can now run smoothly on our cell phones [1–4]. Furthermore, the scaling of the transistor also empowers the widespread application of artificial intelligence, which is a typical computationally-intensive field. As a result, in recent years, people started feeling that their hardware has become 'smarter' in recognizing faces, understanding voices, driving cars, etc..

Introduction

In 1965, Gordon Moore, the co-founder of Intel, predicted that "the number of transistors on a microchip doubled about every two years" [5, 6]. We could expect that the scaling of transistors would continuously boost the emerging AI applications. However, recent research suggested that the scaling confronts the end due to the fundamental limits: the gate dielectric length approaches several atomic spacings or silicon lattice constant (0.54 nm) [1, 7–10]. The leakage current significantly increases as the gate length gets small. Simply shrinking the transistor may end up short-circuiting the chip. In addition, there is another famous prediction named Denard scaling, which states that increasing the transistor density can also improve the maximum clock frequency and energy efficiency. However, Denard scaling failed around 2006 because of the heat-removal problem. The computational capability of a single-core has barely improved afterwards [7, 11]. Thus, future computationally-intensive AI applications may be limited by the development of transistors.

The computing resource of modern computers stems from the use of transistors. Complementary Metal-Oxide-Semiconductor Transistor (CMOS)-based digital circuit can efficiently operate 0' and 1' bits for accurate computing under the Turing-von Neumann paradigm. Due to the difficulty discussed above, researchers began to explore alternative approaches to continue advancing computers. Apart from transistors, there exist other properties of electronic devices that could be developed as computing resources. For example, a simple diode can physically perform a nonlinear transformation similar to the Rectified Linear Unit (ReLU) function commonly used as an activation function in machine learning. Besides, a simple passive low pass filter composed of a resistor and a capacitor can serve as an integrator over the continuous input stimulation [12]. Furthermore, recently emerging devices and materials exhibit interesting characteristics that could be used for computing. For instance, non-volatile memristors offer continuously tunable conductance, which can act as variables in a physical computing system. Using memristor array, also known as Resistive Random Access Memory (RRAM), for high-efficient vector-matrix multiplication (VMM) has received considerable

attention [13]. Given the wide varieties of electronics, the next question is how to fully utilize their properties to compute?

1.2 Neuromorphic computing

The rapid development of both neuroscience and computer reveals the fact that the human brain is a super-powerful computer with low energy consumption and excellent cognitive capability. The human brain consists of about 10^{11} neurons and 10^{15} synapses, which are extremely complex and yet to be fully studied [3, 4, 10]. In 2009, IBM's Blue Gene Supercomputer with 147,546 processors and 144 terabytes of memory was used to simulate a cat's cerebral cortex. The simulator was about 83 times slower than a real cat and consumed much more energy, indicating computing potential and the complexity of the biological nervous system [14].

Despite the complexity of analysing a real brain, we can still take inspiration from its working mechanism to engineer a computing system, referred to as "neuromorphic". As a bio-inspired learning algorithm, Artificial Neural Networks (ANNs) provide computational algorithms that mimic the activities of simplified neurons and synapses, which have yielded tremendous successes in the software domain and have become the mainstream in AI applications [9, 15, 16]. However, running a neural network under a software-based system brings the entire information processing to a digital system where the data experiences storing, processing and communication using the bits of 1's and 0's. Software-based Artificial Intelligence (AI) faces challenges in further reducing computational cost and miniaturisation in the post-Moore era [10, 17]. To keep up with the computing needs o, neuromorphic computing, originated in 1990 by Carver Mead, paves a new way to develop bio-inspired neural networks in the physical domain [18, 19]. In general, neuromorphic systems fully explore hardware's intrinsic physical behaviours as computing resources to build intercon-

nected neurons and synapses. Such bio-inspired architecture is fundamentally different from conventional Turing-von Neumann architecture, holding the promises of parallel, analogue , adaptable, high efficient, noise-tolerant and low-power computing for solving machine learning tasks [2, 9, 20–22]. It has been predicted that after the graphics processing units (GPU)-driven and application-specific integrated circuits (ASIC)-driven machine learning in the past decades, developing an analogue neuromorphic computer will become the next dominance of AI research after the 2020s [22].

1.3 Physical reservoir computing

1.3.1 Reservoir computing algorithms

Reservoir computing (RC) is a bio-inspired machine learning paradigm invented two decades ago. Inspired by human brain, Prof. Jaeger invented Echo State Network in 2001 [23, 24]. Meanwhile, Prof. Maass independently proposed Liquid State Machine (LSM) in 2022 [25]. Afterware, Verstraeten *et al.* experimentally proved the similarity of ESN and LSM and unified them as RC [26]. The randomly and recurrently connected nonlinear nodes in the reservoir layer offer an efficient implementation of Recurrent Neural Network (RNN) with low training costs. The complex dynamic generated by the reservoir layer nonlinearly maps the input data into spatiotemporal state patterns in a higher dimensional feature space where the state vectors of different classes can be easily separated [23, 27]. Furthermore, RC is particularly powerful in studying temporal input data owing to the recurrent connections that create the dependency between the current and the past neurons' dynamic, which is also known as short-term memory or a fading memory [24, 28]. Because of its nonlinear, dynamical and memristive properties, RC has demonstrated superior performance in complex time series prediction and classification tasks. For example, by training the output weights

using linear regression, a reservoir can adaptively couple its nonlinear dynamic with natural chaos [27, 29].

A conventional reservoir computing network consists of an input layer, a reservoir and an output layer. For a reservoir network with d-dimensional input, l-dimensional output and N neurons, only the coefficients between the output and reservoir ($W_{out} \in \mathbb{R}^{l \times N}$) need to be trained by a linear regression method, while the input coefficients ($W_{in} \in \mathbb{R}^{N \times d}$) and reservoir coefficients ($W_{res} \in \mathbb{R}^{N \times N}$) are randomly generated [23, 24]. The complex dynamic and nonlinear transformation in the reservoir would map the input data onto higher dimensional space for classification or prediction. With the internal feedback, the past neuron states can be preserved in the fading memory to affect the computation at the current state [30, 31]. At each time step n, the states in a standard software-based reservoir are subjected to:

$$s(n+1) = f[a\mathbf{W_{in}}u(n+1) + b\mathbf{W_{res}}s(n)] \tag{1.1}$$

where *s(n)* denotes the reservoir states with n^{th} input, *s(n)* denotes the input at time step n, f represents the activation function, a and b are the input and feedback scaling factors, respectively. Under this dynamic, the interaction between neurons generates the high-dimensional recurrent states denoted by *s(n)*. Afterwards, only the output layer $\mathbf{W_{out}}$ of the reservoir needs to train by linear regression using collected *s(n)* and target output. The output of an RC $y(n)$ can be easily obtained:

$$y(n) = \mathbf{W_{out}}s(n) \tag{1.2}$$

The reservoir has been considered one of the effective methods to construct an RNN because of its training complexity. Compared with other neural network models, it shows superior performance in the prediction and classification tasks with time-dependent data input.

The overview of RC strategies is illustrated in Fig. 1.1(a). In particular, the reservoir layer defined by $\mathbf{W_{res}}$ had been comprehensively studied, resulting in a number of varieties [23, 27, 28, 32–34]. Originally, the $\mathbf{W_{res}}$ of a typical RC is a randomly generated matrix, as illustrated in Fig. 1.1(b). For example, by fine-tuning and scaling the $\mathbf{W_{res}}$, an RC could exhibit echo state property, which is discussed in the studies of ESN [23, 27]. Furthermore, researchers also try to design a reservoir in a deterministic manner, rather than randomly generated and scaled. A cyclic structure of $\mathbf{W_{res}}$, which is more simplified and sparse than a random one, was found that it can perform the RC functionalities in time series processing tasks without performance degradation, namely Cyclic Reservoir (CR), as shown in Fig. 1.1(c) [28]. Based on CR, the performance could be further improved by adding regular jumps on the cycle, namely Cycle Reservoir with Jumps (CRJ) [32, 35]. In addition, LSM comes from computational neuroscience background and represents spike-based RC with integrate-and-fire neurons [25, 36]. In network scale, deep RC (multiple reservoirs in series) and parallel RC (multiple reservoirs in parallel) are also of high interest for more demanding computing scenario [37–40]. Finally, RC has been successfully used in various applications, such as temporal signal processing/forecasting [23, 28, 41], pattern classification [31, 42], system approximation [29, 43, 44] and bio-signal processing [45–48].

As a machine learning algorithm, the high-dimensional mapping strategy of RC is similar to the concept of a support vector machine, while the recurrent connections allow the state vectors contain the spatiotemporal information [49, 50]. The state vector generated at time step n represents the high-dimensional features of not only the input at time step n, but also a fading amount of previous steps $(n-1, n-2, n-3, ...)$. The temporal input that cannot be directly distinguished could be linearly separable after the nonlinear mapping. Then, the output layer $\mathbf{W_{out}}$ acts like a hyperplane to separate different input classes in pattern classification. In prediction or system approximation tasks, the state vector could adapt to the target output by multiplying with trained $\mathbf{W_{out}}$. In this regard, RC algorithms

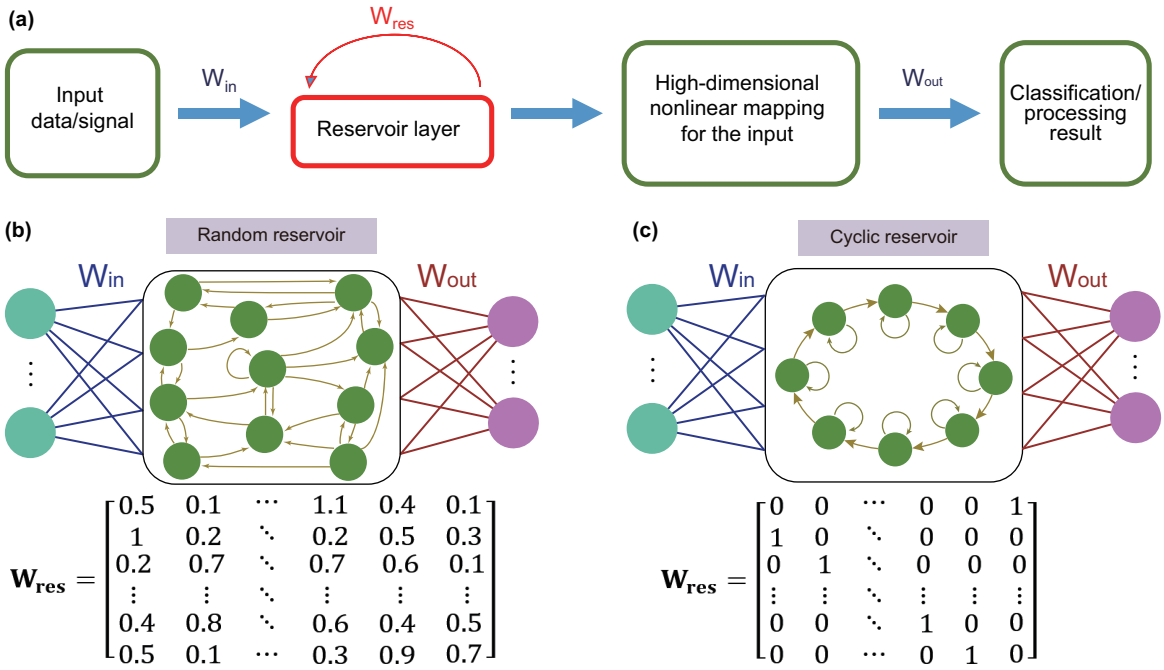


Fig. 1.1 An overview of RC algorithm (a) and two examples of reservoir network structures, including (b) The classical reservoir with randomly generated $\mathbf{W_{res}}$, and (c) A simplified reservoir with cyclic structure, which can be designed in a deterministic manner rather than random generation.

can also be considered a trainable dynamical system. In principle, a wide range of the existing dynamical discrete-time system, including dynamical controllers, nonlinear filters and even chaotic systems, could be approximated by properly configuring an RC [51]. This interesting property provides new insights into the physical implementation of RC, that is, using dynamical systems to approximate the nonlinear high-dimensional mapping function of RC algorithm to move the computing into the physical domain, which has received considerable attention in neuromorphic computing.

1.3.2 Physical implementation

Given the potential and unique property of RC, exploring physical dynamics as computational resources of reservoirs for high-efficient information processing has received considerable attention in recent years. Generally. the software-based reservoir layer is employed to nonlinearly map the current input and a fading amount of previous input into a high-dimensional space. In the context of physical RC, we are committed to seeking physical objects or systems that can effectively perform similar high-dimensional mapping as the software-based reservoirs do, which is becoming one of the major branches of neuromorphic computing [16, 52–55].

The exploration of physical RC is still at an early stage, which mainly involves two aspects: architecture and processing core (Fig. 1.2). In recent years, most works on physical RC investigated different processing cores with a chosen architecture. The processing core that has been proposed for physical RC includes conventional electronics [56–60], spintronic devices [17, 61, 62], memristive devices [48, 63–70], optical devices [54, 71–76], nanowires [77–82] and Micro-electromechanical Systems (MEMS) [83–85]. There are also interesting processing cores such as soft robot [86], biological tissue [87] and swarms [88]. In most cases, the processing core should provide nonlinear transform and integration simultaneously.

While the processing core has been widely studied, architectural innovation of physical RC is rarely found, but usually can significantly impact the field [16]. The effective architectures that can make the utmost of hardware resources as physical RC are discussed as follows.

Delay-based reservoir

At the early stage, the concept of delay line was introduced to the RC algorithm to describe the state update [28]. In 2011, a pioneer study introduced a delay-based reservoir and the concept of virtual nodes into a physical implementation of a CR, which is known as Delay-based Reservoir (DRC) [56]. It is worth mentioning that this work clearly discussed the role of dynamical neurons in physical RC, and consequently inspired other approaches, making it an attractive candidate in the field of neuromorphic computing.

By introducing physical delay lines and time-multiplexing operation to form a delay-coupled reservoir, the DRC dramatically reduces the number of nonlinear neurons to one, which facilitates its hardware implementation using analogue and optical components for high-speed and low-power computing [72, 76]. The virtual nodes can only be generated by properly tuning the ratio between the time constant of the processing core and the intervals of time-multiplexing. Similar to the traditional reservoir, the virtual nodes on the delay lines create a complex reservoir dynamic to map the input to high dimensional space for classification. The delayed feedback loop plays an important role in preserving previous information within the network, that is, Memory Capacity (MC). In this regard, DRC is particularly of high interest in combining with optical or optoelectrical devices since the delay line can be directly implemented by a long optical fibre subject to the transmitting velocity of light [54, 71–73].

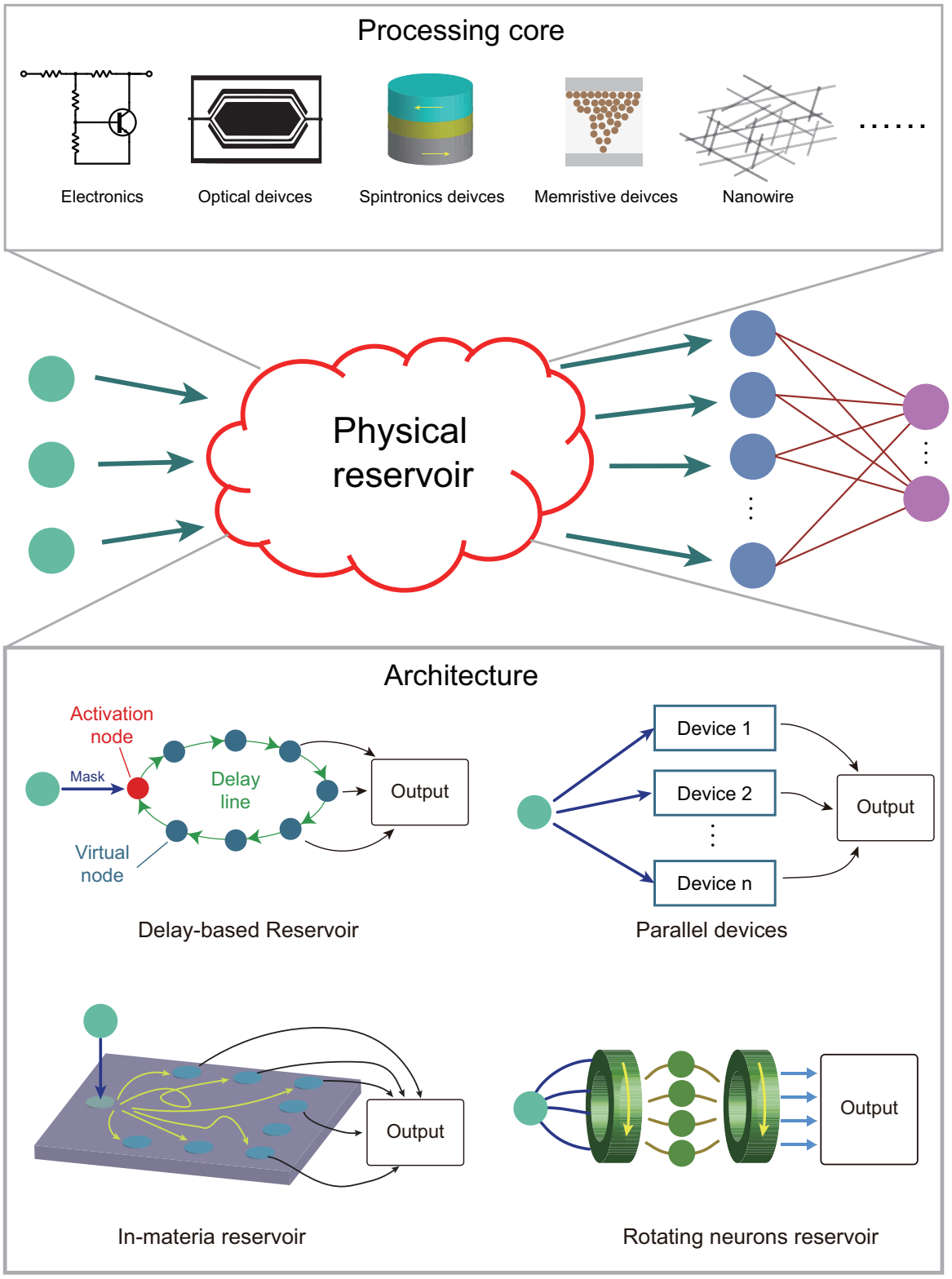


Fig. 1.2 An overview of physical RC development.

Parallel devices

Parallel devices is an attractive method to demonstrate the computing potentials of emerging devices. The architecture of parallel devices is simply using multiple dynamic devices in parallel to receive a common input and then output state vector. One important conclusion is that the usually unwanted device-to-device (D2D) variation is crucial to improve the state richness under the parallel devices architecture [64], whereas cycle-to-cycle (C2C) variation should be minimized. The state vector will be much less expressive if the D2D variation is low. Meanwhile, the MC is provided by the intrinsic property of the device, which is relatively lower compared with other approaches. Therefore, the parallel devices approach desires processing cores with memristive property, such as volatile memristor [63–68, 70, 89]. In particular, time-multiplexing can be used to increase state richness under parallel deices architecture. However, a long time-multiplexing matrix may reduce the MC and disable the RC functionalities [63].

In-materia

The in-materia is a designless method to implement a physical RC [80], which is based on the assumption that the inner dynamics of a chosen material are complex enough for high-dimensional mapping. Given a conductive surface or material, injecting signal at a specific position could result in different responses at other positions where the collected output could be considered the state vector. The signal experiences uncontrollable and unsymmetrical transformation within the material for high-dimensional mapping. Such in-materia method is normally implemented by using electrical input signal and conductive materials, and electrodes are placed on the surface for input and output [62, 90–93]. Similar to other approaches, memristive materials are preferable in in-materia approach to avoid the shortage in MC [48, 78, 80, 82]. Notably, nanowire and nanotube are frequently used as

processing cores in in-materia approach since they can create diverse interconnected patterns [48, 77–82, 91]. In addition, it is interesting that unconventional computing substrates, such as swarms (simulation) [88], was also used to demonstrate in-materia reservoir computing.

Rotating neurons reservoir

Rotating Neurons Reservoir (RNR), originally proposed by the author in [12], is the latest implementation paradigm of RC to date. The author found that the state vector update in conventional CR is analogous to a physically rotating object embedded with dynamical neurons, which can be proved by mathematical derivation. The RNR theory is adaptable to various rotational objects. Differ from other methods, the equivalence between software CR and hardware RNR allows interpretable hardware design. The rotation and signal-driven neurons can be implemented using resource-efficient devices and materials without the assisted peripherals such as Analogue-to-digital Converter (ADC) and data buffer. Specifically, the RNR can implement reservoir functionalities in the physical domain. Furthermore, an electrical RNR (eRNR) was simulated and then developed as a prototype to demonstrate the physical RC, where the rotation was implemented by efficient logic switches. The results confirmed the eRNR's advantages in resource-efficient edge computing.

1.3.3 Input layer

An input layer needs to be specifically designed according to the reservoir architecture and processing core. In DRC, the main operation in the input layer is time-multiplexing at which the input signal is multiplied by a randomly generated matrix $\mathbf{W_{in}}$. In fact, the time-multiplexing operation is adaptable to most existing physical RC for the purpose of increasing the state richness. In RNR, the input layer is physically configured by a switch array to select negative or positive input for each channel, which is equivalent to multiply

by a $\mathbf{W_{in}}$ consisting of -1, 1. $\mathbf{W_{in}}$ is not always required in parallel devices and in-materia architectures whose input can be a raw signal [65, 93]. Also, the raw signal can be encoded in a spike sequence as the input signal, which is commonly used with memristive processing core [64–66, 68, 80, 93, 94]. For 2D input data like images, usually, segmentation and rearrangement are needed before encoding into spikes [64, 65, 80, 94].

1.3.4 Output layer

The output layer of reservoir computing is normally a simple fully connected network, based on the assumption that the state vector after the reservoir layer can be processed by a linear readout. In the majority of existing RC works, the innovations focused on reservoir layers, whereas output layers are accomplished by using digital computers. From the viewpoint of hardware, the fully connected layer is a typical VMM operation, which can be achieved by a memristor array whose conductance is adjustable by an electrical signal [80, 94, 95]. The final output can be obtained by measuring the current output of a memristor array subject to Kirchhoff's Current Law [13, 96–99].

1.3.5 Discussion

The above-mentioned four architectures are the fundamentals of most existing physical RCs in the literature. The representative works of electrical RC are listed in 1.1 sorted by publication year. Apart from the architecture and processing core, this table also compares their network details, tasks and results. The number of cores means how many processing cores have been employed in the system. Next, the widespread use of a time-multiplexing mask allows every core to generate more neuron states so that number of neurons per core is considered.

Table 1.1 State-of-the-art electrical RCs

Works	Architecture	Processing core	# of neurons per core	# of cores	Output layer	Task	Result
2011 [56]	DRC	Nonlinear circuit	400	1	400×1	*NARMA10	NRMSE=0.15
					400×10	Spoken digit recognition	WER=0.2%
2013 [76]	DRC	Semiconductor laser diode	388	1	388×10	Spoken digit recognition	Classification error = 0.014%
					388×1	Santa Fe prediction	Prediction error = 10.6% (1.3×10^7 p/s)
2017 [17]	Parallel devices + Mask	Spintronic oscillators	400	1	400×10	Spoken digit recognition	Recognition rates =99.6%
					400×1	Waveform classification	RMSE = 1%
2017 [65]	Parallel devices	WOx memristors	1	5	5×10	Digit picture recognition	Successful
			1	88	176×10	Handwritten digit classification	Accuracy = 91.1%
			1	90	90×1	Solving a second-order nonlinear dynamic	NMSE = 3.61×10^{-3}

Works	Architecture	Processing core	# of neurons per core	# of cores	Output layer	Task	Result
2019 [64]	Parallel devices	WOx memristors	1	50	400 × 10	Spoken digit recognition	Recognition rates=99.2%
			1	20	1000 × 1	Mackey–Glass time series prediction	Successful
2020 [48]	Parallel devices	Volatile memristor	30	1	CNN + 5 × 5 perceptron	Neural activity analysis†	Accuracy = 87.0%
2021 [66]	Parallel devices	SnS-based memristor	1	14	71 × 5	Korean sentences recognition†	Accuracy = 91%
2021 [68]	Parallel devices	WO3 memristors	1	16	161×51 ×4	Artificial olfactory inference†	Accuracy = 95%
2021 [63]	Parallel devices	Volatile memristor	10	40	400 × 10	Spoken-digit recognition	Recognition rate = 99.6%
			4	25	100 × 1	Hénon map prediction	NRMSE = 0.046
			4	10	40 × 1	Waveform classification	NRMSE = 0.14
2021 [80]	In-materia	Nanowire networks	4	1	4 × 4‡	4 × 4 pattern recognition	Accuracy = 90%
			196	1	196 × 10	Handwritten digit classification	Accuracy = 90.4%
			98	1	N/A	Mackey–Glass prediction*	N/A

Works	Architecture	Processing core	# of neurons per core	# of cores	Output layer	Task	Result
2021 [93]	In-materia + DRC	Organic electro-chemical networks	N/A	1	N/A	MIT-BIH dataset	Accuracy = 88%
			N/A	1	N/A	Flower classification (Iris dataset)	Accuracy = 96%
2021 [92]	In-materia	Sulfonated Polya-nilineNetwork	15	1	15 × 1	Waveform generation	Accuracy = 88%-99%
			12	1	12 × 10	Spoken-digit recognition	Accuracy = 60%
2021 [92]	In-materia	Nanowire networks	261	1	261 × 1	Waveform generation	Accuracy=98% for sine 81% for square, 56% for phase shifted
2021 [85]	In-materia	Micromechanical resonator	440	1	440 × 10	Handwritten digit classification	Accuracy = 93%
			N/A	1	N/A	NARMA1	MNSE = 0.051
			N/A	1	N/A	Gesture classification†	Accuracy = 97.17±1%
2021 [95]	Parallel devices	Ferroelectric tun-neling junctions	1	28	196×64 ‡ ×256×10	Handwritten digit classification	Accuracy = 92.3%

Works	Architecture	Processing core	# of neurons per core	# of cores	Output layer	Task	Result
2022 [94]	Parallel devices	Optoelectronics	1	28	28×4	Image classification*	Accuracy = 99.97%
			1	25	25×4	Vehicle flow detection*	Accuracy = 100%
2022 [12]	RNR	Nonlinear circuit	400	1	400 × 1	NARMA10*	NRMSE=0.078
			1	64	64 × 1	Mackey–Glass time series prediction	NRMSE=0.03
			1	64	64 × 1‡	Handwirtten vowel recognition†	Accuracy = 94.0±0.8%

*Simulation result

†Custom task

‡Memristor-based output layer

The following is a brief comparison of the strengths and weaknesses of the four architectures:

- Compared with other approaches, the unique advantage of DRC is that fewer neurons are required even with a large network size (number of virtual neurons). Meanwhile, multiple delayed feedback lines can significantly increase the MC. However, the use of time-multiplexing and virtual nodes introduce serial operations at both input and output, which against the design primitives of parallel computing in neuromorphic computing. In addition, adding a delayed feedback line is not a straightforward design, especially in an electrical system where ADC and memory are required [16]. Because of this, DRC is preferable in optical systems where the delay line can be achieved by optical components, as discussed above. It is noteworthy that the virtual node concept is also useful for other architectures to improve the state richness.

- Parallel devices architecture is especially compatible with various emerging devices, and makes full use of their nonideality such as D2D variation. The state quality highly depends on a proper D2D variation and a low C2C variation. It is advantageous in terms of parallelism and analogue computing in the absence of a delay line. Its weakness involves the limited MC provided by the device itself. Therefore, memristive devices are more suitable for this architecture [63].

- In-materia method employs a standalone material or substance for physical RC. It is highly parallel and analogue in the process of generating state output. It is fundamentally different from other architectures in terms of interpretability. While the dynamic behaviours in other architectures are measurable and sometimes adjustable based on device characterisation and physical connections, the signal in in-materia usually experiences unclear transformation and only the state output channels are accessible.

- RNR presents advantages in system complexity, interpretability, low power consumption and all-analogue computing [12]. However, it is proposed recently and yet to be fully studied.

1.4 Physical reservoir computing with sensory input

Sensors act as the information collector of a machine or a system that can respond to its physical ambient environment. They are able to translate a specific type of information from a physical environment such as the human body to an electrical signal. Sensor technologies enable mass ambient data collection from human activities and the surrounding environment, which require miniaturized, flexible, and highly sensitive sensors to capture clear information [15, 100, 101]. However, from processing aspect and to make a signal meaningful towards personalized devices, further development is still needed [2]. Since the sensing signal is relatively weak and noisy, a readout circuit (normally composed of an amplifier, a conditioning circuit and an analogue signal processing unit) is necessary to make the signal readable for a system [102]. The subsequent high-level system processes the data and sends commands to actuators for a closed-loop control or interaction [103, 104]. For various applications ranging from human-machine interfaces to health monitoring, different combinations of sensors and systems have been developed over the past decade. The use of machine learning empowers sensors to build a novel smart application [2].

Recently, the field of artificial intelligence further boosts the possibility of smart sensory systems. The emerging intelligent applications and high-performance systems require more complexity and demand sensory units accurately describe the physical object. The decision-making unit or algorithm can therefore output a more reliable result [100, 105–107]. The novel applications using multiple sensors and high learning ability usually require more energy in the computing unit [108]. This weakness limits the further development of smart

sensory systems [108]. The existing solution is to wirelessly transfer the raw data onto a cloud where the computationally intensive algorithm is implemented. However, this solution is not ideal considering (i) the complexity of using a wireless module, (ii) the non-negligible power consumption, (iii) the amount of data, (iv) the space limitation due to the range of wireless transmission, (v) privacy issues due to the broadcast of signals, (vi) non-negligible time latency due to communication channel. These technological drawbacks strongly limit the application of smart sensors.

Implementation of ANN in von Neumann architectures results in a non-optimized distribution of the energy consumption. Compared to conventional approaches based on a binary digital system, brain-inspired neuromorphic hardware has yet to be advanced in the contexts of data storage and removal as well as their transmission between different units [2, 109]. In this perspective, a neuromorphic chip with a built-in intelligent algorithm can act as a front-end processor next to sensors. Physical RC, as discussed above, is a resource-efficient implementation of machine learning, which could be preferable to deploy at the edge [110].

1.5 Research summary

This thesis aims to investigate the mechanism and application of physical RC with two focuses: architecture and electronics implementation. The reasons are:

- As aforementioned in the literature review, prior studies have noted the importance of architectural innovation in the development of physical RC. Ideally, an excellent architecture could fully harvest the hardware dynamics as a computing resource with minimum system complexity and cost. A novel architecture could inspire more physical RC systems assembled with different processing cores.

- RC is an attractive computing paradigm that has been adopted by multiple fields, including optical devices, quantum devices and mechanical structures. However, electronics is still the mainstream substrate for computing and analogue signal processing. Particularly, the recently emerging novel devices, such as memristor, can provide a wide range of hardware dynamics to construct reservoir computers. Additionally, an electrical RC could be more compatible with modern computing and signal processing systems.

Thus, three architectures with their electrical implementations are studied in the following sections: delay-based RC, parallel devices RC and RNR. The in-materia RC is not covered in this thesis since this approach highly depends on the internal structure of the materials while in lack of interpretability, as previously discussed. The architectures and processing cores are implemented by simulation or hardware with for collecting experimental data. Their network characteristics, signal flows and performances on benchmark temporal signal processing are investigated. Meanwhile, each implementation is combined with sensory signal to evaluate its performance on practical applications. The main contributions are:

- In Chapter 1, the state-of-the-art reservoir computers are reviewed. For most existing physical RC, this thesis divide their architectures into four categories according to their implementation of the reservoir layer (or middle layer), including delay-based reservoir, parallel devices, in-materia and rotating neurons reservoir. Their main features, building blocks, signal flows and representative works in the literature are introduced independently, followed by their comparison and discussion. In addition to reservoir layer, input and output layer are also worth discussing for implementing a complete reservoir computing system. Input layer of physical RC mainly depends on the reservoir layer, acting like a pre-processing or encoding step for adapting the input analogue signal to be receivable by the reservoir layer. Meanwhile, output layer for most RC is a typical VMM operation, which can be implemented by memristor

array. During the development of physical RC, architectural innovations played an more significant role. An effective and ingenious architecture, such as delay-based RC proposed in 2011, can efficiently harness electronic properties for computing and inspire following works.

- In Chapter 2, the delay-based RC is investigated and tested by a arrhythmia detection task. A delay-based RC model with a Mackey-Glass typed nonlinear circuit and multiple delay lines was developed to receive filtered ECG signal collected by MLII configuration. The database used to test this model is the commonly-used MIT-BIH database, and the evaluation follows the inter-patient paradigm and standard dataset splits (DS1 and DS2). In this work, the proposed model, acting like a dynamic system, directly receives raw and continuous ECG signal in the absence of signal segmentation and feature extraction, which is the prime difference compared with the existing software-based algorithms. The analysis of memory capacity verifies the idea that the node state collected at the end of every single heartbeat contains the fading information of the whole heartbeat, so that the MC is fully utilized to store ECG information rather than using external memory. Also, the important parameters, such as input and feedback strength, are analysed in details and then fully optimized. Finally, the model yielded acceptable sensitivity and accuracy for the VEB detection task, while minimizing the memory required to run the system.

- In Chapter 3, another architecture, parallel devices, is studied, and TiO_x-based volatile memristor was used as processing core. Initially, the fabricated volatile memristors were characterized and then modelled using the discrete behavioural model. In order to study how to empower memristive deice with computing capability, a reservoir system was simulated based on this discrete model under parallel devices architecture, where time-multiplexing operation was also employed to increase state richness. Afterwards, this model was evaluated by three benchmark tasks: waveform classification, Hénon

map chaotic signal prediction and human activity recognition. For the first two tasks, the model demonstrated its capabilities in temporal signal processing in principle. The signals were successfully classified and predicted. Increasing the number of memristor and fine-tuning the parameters can improve the processing results. The 5-class human activity recognition is a more demanding task. The system obtained acceptable results but additional operations, such as delayed feedback and SMOTE, are required.

- Studies of physical RC show the importance of architectural innovations. Chapter 4 introduces the theory of the novel architecture, namely RNR, that was recently proposed by the author in [12]. RNR was developed by linking the CR algorithm and rotating behaviours in hardware. It was found that when rotating a specifically designed neuron array that processes nonlinearity and dynamical behaviours, the input and output of the hardware system can be roughly equivalent to a CR algorithm, which can be proved mathematically. Such network-level equivalence is rare to found in other architectures. Based on the RNR theory, an eRNR was designed. In the simulation, the equivalence between hardware and software model are validated by analysing the task-independent network properties such as MC. Additionally, it was surprisingly found that the eRNR model obtained the record-low error in the NARMA10 system approximation task. Those results emphasized the fact that, on one hand, the RNR hardware and software are roughly equivalent since their similar network behaviours were observed; on the other hand, software and hardware are not ideally equal and the hardware provides rich dynamics within a certain range that can be explored to enhance the performance.

- Chapter 5 discusses the experimental results of the eRNR. A proof-of-concept prototype of eRNR was implemented by commercial components and PCB, along with its measurement system and user interface. The first demonstration is real-time Mackey-Glass time series prediction, where the input is the Mackey-Glass time series and

the output is the one or multistep ahead prediction. The second demonstration is real-time handwriting vowel recognition. In this system, the eRNR directly received horizontal and vertical handwriting trace signals of a resistive touch screen without intermediate block for signal buffer or digital processing, indicating its near-sensor computing ability. Another experiment involves using memristor array as output layer for the VMM operation to form end-to-end all-analogue computing. Finally, the power analysis suggested that the all-analogue eRNR system consumed three order of magnitude lower power than the existing physical RC systems. This is attribute to the RNR architecture that is equivalent to CR on network level, enabling the coherent, straightforward and elegant hardware implementation.

- Finally, the conclusion of this thesis and further perspective about physical RC as well as its application are provided.

In addition to the physical RC, the author was also involved in several side research related to sensory system, signal processing and machine learning, which have mostly been published in journal articles and conference proceedings. These side works are summarised as follows:

- Based on the idea that hand gesture could be captured by tracking tendon movement around the wrist, a wrist-worn gesture sensing system was proposed in [100, 107]. In this work, an array of PDMS-encapsulated capacitive pressure sensors is attached to the user to capture the pressure distribution. The pressure signals are then processed by support vector machine to reconstruct gestures. Furthermore, different sensors and algorithms were tested for wrist-worn gesture, which were conducted by the author and collaborators [31, 111, 112].

- Two sensor fusion approaches were proposed to combine static sensor (pressure senor) and dynamic sensor (radar) [15, 113], which is a continuation of the wrist-worn gesture

sensing system. The two sensors that focus on different information can be a enhancer of each other to improve the final recognition rate.

- Spintronic sensor can detect magnetic field variation with high sensitive. When attaching spintronic sensor on eyeglass, eye movement can be captured by embedding a small magnet on contact lens for human-machine interaction [101].

1.6 List of publication

1.6.1 Journal publication

First author publication

[1] **X. Liang** *et al.*, "Rotating neurons for all-analog implementation of cyclic reservoir computing," *Nature Communications*, vol. 13, no. 1, p. 1549, 2022. [Selected for **Editors' Highlights**]

[2] **X. Liang** *et al.*, A Neuromorphic Model with Delay-based Reservoir for Continuous Ventricular Heartbeat Detection. *IEEE Transactions on Biomedical Engineering*, vol. 69, no. 6, p. 1837 2021.

[3] **X. Liang** *et al.*, "Fusion of Wearable and Contactless Sensors for Intelligent Gesture Recognition," *Advanced Intelligent Systems*, vol. 1, no. 7, p. 1 900 088, 2019.

[4] **X. Liang**, R. Ghannam, and H. Heidari, "Wrist-Worn Gesture Sensing with Wearable Intelligence," *IEEE Sensors Journal*, vol. 19, no. 3, pp. 1082–1090, 2019.

Co-first author publication

[5] E. Covi, E. Donati, **X. Liang** *et al.*, "Adaptive Extreme Edge Computing for Wearable Devices," *Frontiers in Neuroscience*, vol. 15, no. May, pp. 1–27, 2021.

[6] A. Tanwear, **X. Liang** *et al.*, "Spintronic Sensors Based on Magnetic Tunnel Junctions for Wireless Eye Movement Gesture Control," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 6, pp. 1299–1310, 2020.

Co-author publication

[7] Y. Zhong, J. Tang, X. Li, **X. Liang** *et al.*, "Memristor-based fully analog reservoir computing system for power-efficient real-time spatiotemporal signal processing," *Nature Electronics*, 2022, Accepted

[8] Y. Liu, **X. Liang** *et al.*, "Ultra-light smart patch with reduced sensing array based on rGO for hand gesture recognition," *Advanced Intelligent Systems*, 2022, Accepted

[9] A. Tanwear, **X. Liang** *et al.*, "Spintronic Eyeblink Gesture Sensor with Wearable Interface System," *IEEE Transactions on Biomedical Circuits and Systems*, 2022, Accepted

[10] H. Li, **X. Liang** *et al.*, "Hierarchical Sensor Fusion for Micro-Gesture Recognition with Pressure Sensor Array and Radar," *IEEE Journal of Electromagnetics, RF and Microwaves in Medicine and Biology*, vol. 4, no. 3, pp. 225–232, 2020.

1.6.2 Conference proceedings

[11] **X. Liang** *et al.*, "A Physical Reservoir Computing Model Based on Volatile Memristor for Temporal Signal Processing," 2022, Accepted.

[12] **X. Liang** *et al.*, "A Delay-Based Neuromorphic Processor for Arrhythmias Detection," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[13] **X. Liang** *et al.*, "Live demonstration: Gaze following system for noninvasively testing electronic contact lens," *27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, vol. 1, no. c, pp. 1–2, 2020.

[14] **X. Liang**, H. Heidari, and R. Dahiya, "Wearable capacitive-based wrist-worn gesture sensing system," in *1st New Generation of CAS (NGCAS)*, 2017.

[15] Y. Ding, **X. Liang** *et al.*, "MMG/EMG Mapping with Reservoir Computing," *29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2022, Accepted.

[16] Y. Liu, S. Zuo, **X. Liang** *et al.*, "Gesture recognition wristband device with optimised piezoelectric energy harvesters," *27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 2020–2023, 2020

[17] K. Lenard, **X. Liang** *et al.*, "Eye tracking simulation for a magnetic-based contact lens system," *27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 31–32, 2020.

[18] W. Wang, **X. Liang** *et al.*, "Wearable wristworn gesture recognition using echo state network," *26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 875–878, 2019.

[19] Y. Sun, **X. Liang** *et al.*, "Visual Hand Tracking on Depth Image using 2-D Matched Filter," *UK/China Emerging Technologies (UCET)*, pp. 14–17, 2019.

[20] Y. Chen, **X. Liang** *et al.*, "Wearable Resistive-based Gesture-Sensing Interface Bracelet," *UK/China Emerging Technologies (UCET)*, UCET 2019, pp. 14–17, 2019.

[21] C. Nunez, W. Taube, **X. Liang**, and R. Dahiya, "Live demonstration: Energy autonomous electronic skin for robotics," in *Proceedings of IEEE Sensors*, 2017.

Chapter 2

Delay-based Reservoir Computing for Continuous Signal Processing

This chapter presents a DRC model for intelligently processing continuous electrocardiogram (ECG) signal. This model aims to develop a hardware-based signal processing model and avoid employing digitally intensive operations, such as signal segmentation and feature extraction, which are not desired in an analogue neuromorphic system. A DRC is used as the information processing unit, along with a novel training and labelling method. Different from the conventional ECG classification techniques, this computation model is a end-to-end dynamic system that mimics the real-time signal flow in neuromorphic hardware. The input is the raw ECG stream, while the amplitude of the output represents the risk factor of a ventricular ectopic heartbeat. The intrinsic memristive property of the reservoir empowers the system to retain the historical ECG information for high-dimensional mapping. This model was evaluated with the MIT-BIH database under the inter-patient paradigm and yields 81% sensitivity and 98% accuracy. Under this architecture, the minimum size of memory required in the inference process can be as low as 3.1 MegaByte(MB) because the majority of the computation takes place in the analogue domain. Such computational modelling boosts

memory efficiency by simplifying the computing procedure and minimizing the required memory for future wearable devices.

2.1 Introduction

2.1.1 Machine learning for arrhythmias detection

Cardiovascular diseases are the major sources of global mortality, which led to 17.9 million deaths in 2016 (WHO) [114]. ECG is a clinical tool that records the electrical rhythm, rate and activity of the heart and provides a detailed analysis for the diagnosis and the treatment of abnormal heartbeats [115]. Early research focused on manual comparative ECG classification and diagnosis by the cardiologist. In recent decades, the rapid development of machine intelligence opens a novel pathway for automatic arrhythmias analysis and detection [75, 115]. In the future these systems have the potential to be integrated into remote patient devices to stratify clinical need by providing a more personalised healthcare system [115].

The methodologies of automatic ECG classification have been widely explored in recent years. Using the open-access MIT-BIH arrhythmia databases [116], previous researches were done on automatically classifying different types of ECG by the intelligent algorithms such as support vector machine [117–119], echo state network [46, 47], decision tree [120], and neural network [105, 121]. Specifically, artificial neural networks have been widely explored and yield over 90% accuracy in the literature [105, 121, 122]. However, running a neural network under a software-based system brings the entire information processing to a digital system where the data experiences storing, processing and communication using the bits of 1's and 0's.To overcome this challenge, neuromorphic engineering paves a new way to develop a physical neural network to keep up with the computing needs [16, 123].

Also, portable ECG signal acquisition devices have been developed in recent years, such as the commercial product Alivecor that can detect atrial fibrillation [124]. Despite good results obtained in previous studies, they require heavy digital processing of the analogue signals acquired from sensors. A prospective wearable device expects that the intelligent computing can also be carried out at the local edge [2, 22, 125]. Under such circumstances, a neuromorphic analogue processor based on the above-mentioned DRC architecture is well-suited to act as a direct interface to an ECG electrode with less memory requirement.

2.1.2 Physical DRC for arrhythmias detection

DRC has been proposed as a candidate for physical implementation among all the topologies of neural network [56]. It is categorized under RC which is derived from RNN. By introducing delay lines to form a delay-coupled reservoir, the DRC dramatically reduces the number of nonlinear neurons to one, which facilitates its hardware implementation using analogue and optical components for high-speed and low-power computing [63, 64, 72, 73, 75, 76]. The DRC architecture is chosen as the main ECG processing core in this work because it is well-suited to process time-dependent signal, and feasible to implement as neuromorphic hardware. To validate the performance of our proposed DRC model, this model is applied to a Ventricular Ectopic Beat (VEB) detection task. Frequent VEB could be a sign for coronary heart disease, rheumatic heart disease and even acute myocardial infarction.

There are fundamental differences between the neuromorphic model with DRC architecture proposed in this chapter and conventional automatic ECG detection. Conventional software-based ECG classification methods can be divided into five steps: *1)* ECG signal preprocessing, *2)* ADC, *3)* heartbeat segmentation, *4)* feature extraction and *5)* learning/classification [115]. These widely used procedures, such as detection of the QRS complex, signal segmentation and feature extraction, critically rely on digital operations which are not

desirable in neuromorphic hardware [4, 10, 22]. For example, the prime step for the majority of automatic ECG classification algorithm is segmenting the entire ECG recording into individual heartbeat according to the detection algorithm of the QRS complex, followed by a feature extraction step where various features are obtained from each individual heartbeat to improve the classification accuracy. These operations are easy to achieve within a digital system by accessing the memory unit and running algorithms in the processor [9, 10, 17].

However, the digital operations and mass memory bring several constrains like latency, throughput and power, hindering the further development of computing performance in conventional architecture [20, 22]. Meanwhile, a prospective wearable device expects that the intelligent computing can also be carried out at the local edge [2, 22, 125]. Under such circumstances, a neuromorphic analogue processor based on the above-mentioned DRC architecture is well-suited to act as a direct interface to an ECG electrode with less memory requirement. Fig. 2.1(a) illustrates the block diagram of the proposed system in comparison with the conventional method. The proposed model preserves the hardware-achievable operations and deposes the procedures involving intensive digital components. Fig. 2.1(b) is a conceptual figure of the input and output of an analogue neuromorphic computer for ECG. Ideally, the indicator can be directly driven by the neuromorphic output that reflects the ECG type. The dynamic system aims to receive a continuous ECG stream and output the abnormal ECG diagnosis result. The differences mentioned above greatly reduce the memory needed to deploy such detection algorithm. Compared with the conventional software-based implementations, the proposed neuromorphic system is expected to offer advantages on energy efficiency and computing power for machine learning workloads. The proposed model is validated by the MIT-BIH database [116]. The evaluation protocol follows the inter-patient paradigm which was presented by [126]. The result shows that this model obtained high accuracy and sensitivity and has a great potential to facilitate the future development of a pure analogue ECG processing system. The full development of the dynamic system is not

covered in this work and remains a topic of future research. The contribution of this chapter is to provide the first fundamental analysis model of such a neuromorphic dynamic system using a DRC architecture specifically designed for ECG signal.

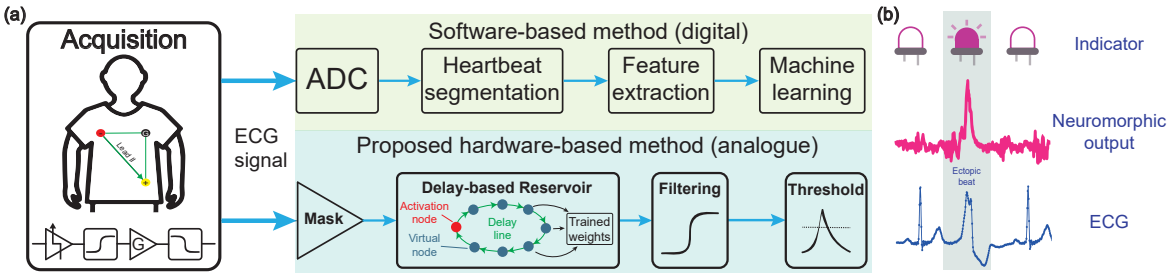


Fig. 2.1 (a) The comparison between the conventional software-based ECG classification and the proposed hardware-based method. The two standard procedures in literature, heartbeat segmentation and feature extraction were not adopted since they are difficult to achieve by analogue system. Instead, the DRC, along with its readout and pre-processing blocks is used as the information processing core. (b) The conceptual figure of the neuromorphic input ECG and output. The proposed hardware algorithm can receive continuous signal and perform point-by-point abnormal ECG detection. The output is an indication of the type of input ECG. A spike can be observed at the output when an ectopic ECG is received.

2.2 Delay-based reservoir design for ECG

The development of proposed ECG processing model mainly includes four key aspects: *1)* construction of the time-multiplexing reservoir, *2)* Lasso regression and shifting labelling, *3)* MC analysis, *4)* post-processing. The simulation was carried out in MATLAB/Simulink software.

2.2.1 Database

The performance of proposed method is evaluated by the MIT-BIH arrhythmia database which is a well-known benchmark task recommended by Association for the Advancement of Medical Instrumentation (AAMI). The database includes 48 two-lead ECG recordings

at 360 Hz sampling rate [127].The AAMI suggested that the heartbeats in the database can be divided into five classes: normal, ventricular, supraventricular, fusion of normal and ventricular and unknown beats. The abnormalities for each type of ECG are clearly labelled in the data stream by at least two cardiologists. In this chapter, the goal is to detect VEB type heartbeat which is highly correlated with coronary heart disease and cardiomyopathy. Furthermore, this work follows the inter-patient paradigm suggesting that the data for training and testing should come from different patients. Another study [126] suggested that the database can be divided into two groups: DS1 (101, 106, 108, 109, 112, 114, 115, 116, 118, 119, 122, 124, 201, 203, 205, 207, 208, 209, 215, 220, 223, 230, where the numbers indicate the recording label) and DS2 (100, 103, 105, 111, 113, 117, 121, 123, 200, 202, 210, 212, 213, 214, 219, 221, 222, 228, 231, 232, 233, 234). The DS1 is used to train the model while DS2 is for testing. The DS1 and DS2 split for inter-patient evaluation is in line with the use of MIT-BIH database in most ECG studies, which makes our results comparable with the state-of-the-art works. This paradigm is considered to be closer to a realistic scenario where the classifier can be directly used on the ECG signal from unknown patients without calibration. In addition, a single lead can satisfy the model requirement and the modified lead II (MLII) that placing electrodes on the chest was selected since it is an informative and commonly used configuration.

2.2.2 Delay-based reservoir computing

A conventional RC network consists of an input layer, a reservoir and an output layer. For a reservoir network with d-dimensional input, l-dimensional output and N neurons, only the coefficients between the output and reservoir ($\mathbf{W_{out}} \in \mathbb{R}^{l \times N}$) need to be trained by a linear regression method, while the input coefficients ($\mathbf{W_{in}} \in \mathbb{R}^{N \times d}$) and reservoir coefficients ($\mathbf{W_{res}} \in \mathbb{R}^{N \times N}$) are randomly generated [23, 24]. As described by Eq. 1.1, the complex dynamic and nonlinear transformation in the reservoir would map the input data onto higher

dimensional space for classification or prediction. With the internal feedback, the past neuron states can be preserved in the fading memory to affect the computation at the current state [30, 31]. In previous studies, RC has exhibited interesting network properties and excellent performance in temporal signal processing [16]. Meanwhile, exploring reservoir's network typology is of high interest in both software and hardware engineering [23, 28]

In recent years, the reservoir has been developed that can be implemented by only one nonlinear neuron with time-multiplexing and a delayed feedback [56]. The randomly connected middle layer in traditional RC is replaced by a single neuron and virtual nodes created by a delay lines, namely DRC [56]. This substitution facilitates the development of a physical reservoir, which is considered to be a candidate of the next-generation neuromorphic signal processors [16]. The 'Delay-based Reservoir' box in Fig. 2.1(a) briefly illustrates the network typology. In this chapter, the DRC is designed to process ECG signal as illustrated in Fig. 2.2(a). Also, the processing core, a nonlinear dynamical neuron, is shown in Fig. 2.2(b). These modelling and design will be explained as follows:

Pre-processing

Before sending the signal to the delay-loop, a pre-processing step is required to convert the raw ECG data to a shape specifically created for our DRC system. The filtering step follows the standard procedure: a 2^{nd} order Butterworth high-pass filter with a cut-off frequency at 0.5 Hz and a 12^{th} order finite impulse response filter with a cut-off frequency at 35 Hz, since the bandwidth in the range of 0.5-35 Hz contains contains most relevant ECG information [47]. Following the filtering, the ECG data was resampled by the sampling rate of 180Hz to reduce the number of samples and accelerate the modelling.

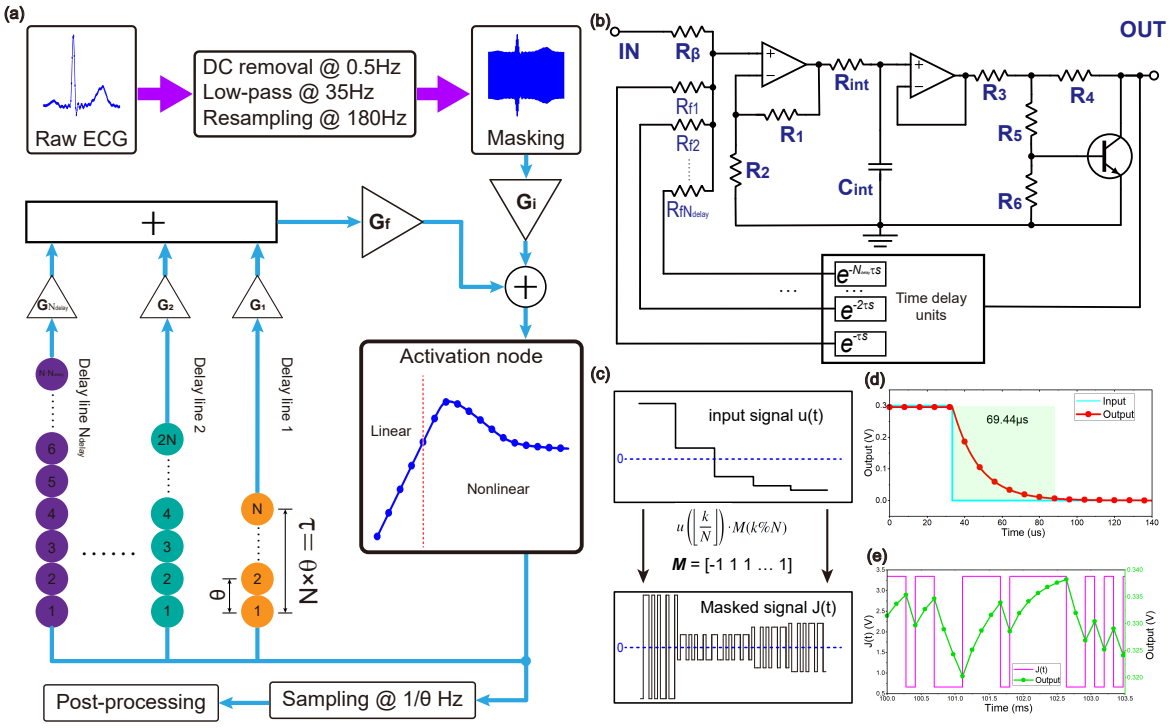


Fig. 2.2 (a) The system modelling of the DRC. After pre-processing and masking for the continuous raw ECG signal, the time-multiplexing signal will be fed into a activation node subjected to delayed feedback lines. Next, the result can be obtained through post-processing step. (b) The circuit design of the activation node and delay unit, which is the core of the RC unit. According to the working principle of DRC, the circuit should be able to exhibit nonlinearity (provided by bipolar junction transistor and $R_3 - R_6$) and integration (provided by R_{int} and C_{int}) properties. (c) An example of the masked signal generated by multiplying the original input with the mask matrix. (d) The settling period of the activation node allows each virtual node states in DRC to connect to historical states, which creates a similar dynamic in conventional reservoir by using fewer physical components. (e) A segment of input masked signal $J(t)$ and output signal of the DRC mode. The green dots are the node states sampled at $1/\theta$ Hz. The value of each green dot is related to historical several values, which implies the connection between the neighbouring virtual nodes.

Masked signal

After pre-processing, a mask step is essential to create a reservoir dynamic in the activation node. Each data point of ECG signal should be multiplied by a binary matrix *M* (consisting of randomly uniform distribution of 1 and -1) with length equal to *N,* which is the number of virtual nodes in the reservoir [74, 128]. Assuming that τ is the sampling interval of the ECG signal, the time interval between every two points in the mask is defined as $\theta = \tau/N$ to facilitate the MC quantification in this chapter, which will be discussed in the following section. Given the input ECG data is *u* and the masked data is *J(k)*, the masking algorithm can be described as:

$$J(k) = u\left(\left\lfloor \frac{k}{N} \right\rfloor\right) \cdot \frac{M(k\%N)}{s} + b,\ for\, k = 1, 2, \ldots, LN \tag{2.1}$$

where $\lfloor\ \rfloor$ is the floor function, *L* is the total length of input, *u*, *s* and *b* denote the scaling and bias factors respectively for adjusting the input range according to the linear and nonlinear ranges of activation node, % is the modulus operation. Afterwards, the resulting *J(k)* is sampled and held to generate a continuous signal *J(t)*. The signal before and after masking are plotted in Fig. 2.2(c). In this chapter, $\tau = 5.56ms$ is the reciprocal of the sampling rate of 180Hz. Also, the network size *N* = 400 and therefore $\theta = \tau/N = 13.89\mu s$ are configured. For the hardware implementation, the analogue masked signal can be obtained by periodically switching the signal between the original signal and its negative counterpart according to the mask matrix M, which remains future development.

Fig 2.3 introduces two approaches to implement the mask circuit using integrated circuit. Both solutions discussed above can generate masked signal using low-cost CMOS circuit and avoid digitalizing the signal with ADC.

In the first solution (Fig 2.3(a)), given an analogue signal source which is defined as a positive signal, an amplifier circuit is employed to generate a negative signal. Next,

the positive and negative signals are connected to a N channels multiplexer composed of transmission gates and inverters. The N channels of the multiplexer are randomly connected to the positive or negative signal, which corresponds to the randomly generated M. In addition, the N-bits counter, driven by pulse with a frequency of $1/\theta$ Hz, provides the address to the channel selection ports of multiplexer. In this solution, the M matrix is defined by the connection of each channel (positive or negative) of multiplexer. In the present of pulse, the counter periodically outputs binary address from 0 to (N-1) and then reset to 0, driving the multiplexer to poll every input channel. Therefore, the output is the masked signal with time-multiplexing. In the second solution (Fig 2.3(b)), similarly, a negative signal is generated by operational amplifier circuit. Then, both signals are fed to a two channels multiplexer. A Static Random Access Memory (SRAM) driven by a counter determines the output of the multiplexer (positive or negative). As the address periodically increases from 0 to (N-1) with interval q, each element of M will be generated continuously, and therefore switching the multiplexer output between positive and negative. In this case, the M matrix is stored in the SRAM requiring N-bits space. For our VEB detection setup where N=400, this solution will additionally require 400 bits memory, approximately 0.05 KB.

Delay-coupled activation node

The *J(t)* is sent to the activation node after pre-processing and masking. The design of the activation node is illustrated in Fig. 2.2(b). The circuit with bipolar junction transistor and resistors forms a Mackey-Glass nonlinear function for the input signal. A passive low pass filter is also connected to prevent the signal from rapidly reaching a plateau. Fig. 2.2(d) shows the step response of the circuit, the settling time τ_{neuron} is obtained from the empirical configuration, where $\tau_{neuron} = 5 \times \theta$ [123], thus it equates to 69.44 μs in our case. This operation plays a crucial role in connecting a number of neighbouring virtual nodes. The effect of this connection can be observed from the input and output in Fig. 2.2(e). The green

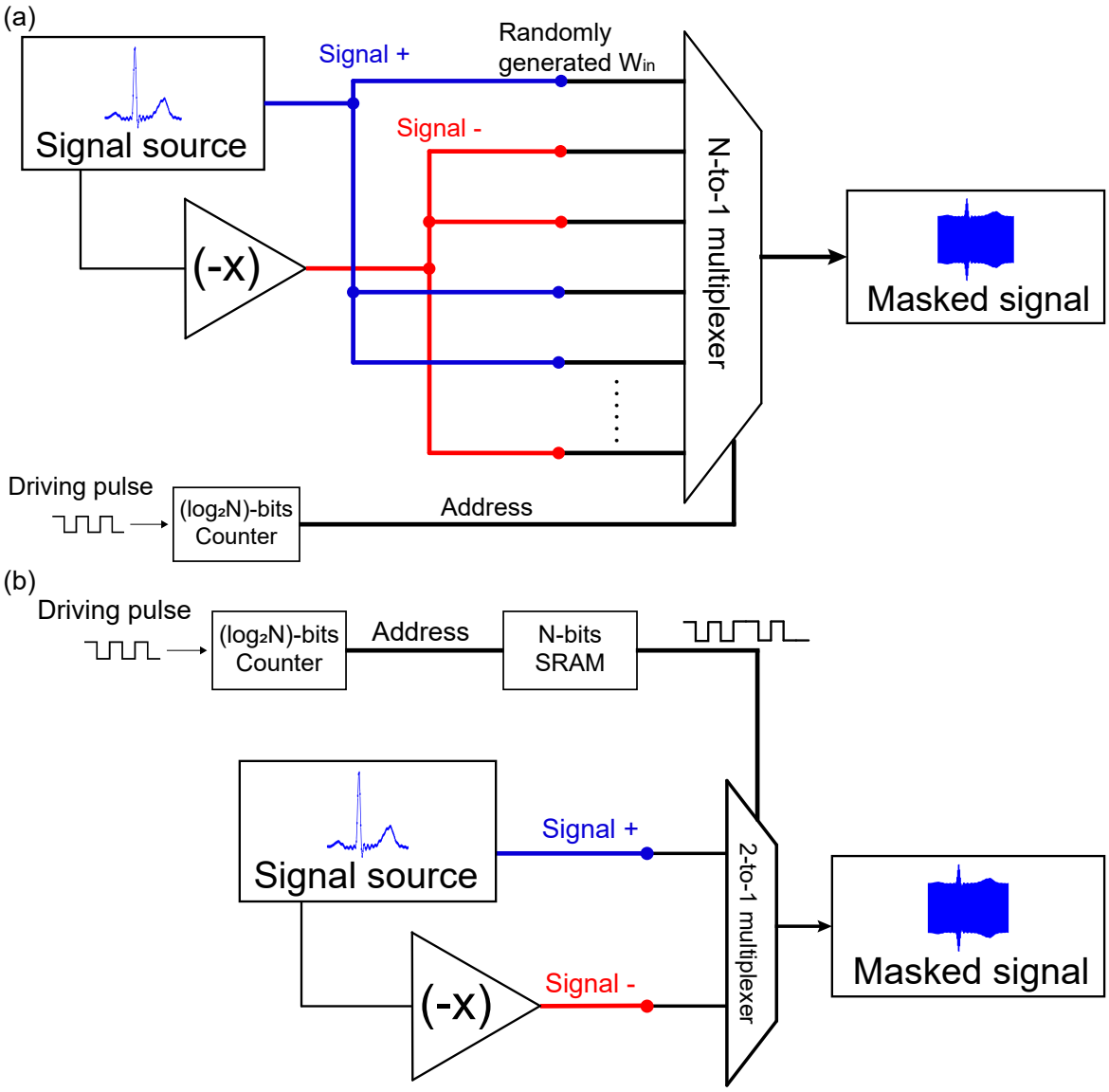


Fig. 2.3 Two schematics for implementing mask circuit. (a) The $\mathbf{W_{in}}$ is configured by the connection of N-to-1 multiplexer. (b) The $\mathbf{W_{in}}$ is stored in N-bits SRAM driven by a counter.

dot line represents the discrete virtual node state **Q** and the pink signal is the masked input *J(t)*. Both of the sampling intervals and mask separation are θ. Given the settling time $\tau_{neuron} = 5 \times \theta$, what stands out in this figure is the correlation between current node state and its historical node states, resulting in the connections between neighbouring virtual nodes. Basically, such connections are provided by the integration property of activation node, and they can exhibits reservoir dynamic to map the temporal input onto high-dimensional node states. Furthermore, if τ_{neuron} is smaller than θ, each node state will reach a plateau rapidly before the arrival of the next value. In this case, each state is irrelevant to historical states and the connections between nodes no longer exist.

To establish a recurrent connection, N_{delay} delayed feedback lines are coupled to the output of the activation node. Each delay line delays the output for a certain time length and then feeding it back to the input with another scaling factor G_f. This delay-coupled reservoir dynamic is subject to a delay differential equation below:

$$\begin{aligned} \dot{q}(t) = &- q(t) + f(G_1 q(t-\tau), G_2 q(t-2\tau), \\ &\ldots G_{N_{delay}} q(t - N_{delay}\tau), J(t)) \end{aligned} \tag{2.2}$$

where *f(x)* is the nonlinear function of the activation node (activation function) formed by the transistor and its surrounding resistors, $q(t)$ denotes the node state at time t, G_1, $G_2 \ldots G_{N_{delay}}$ are the strengths of each delay line. The delay lines, which can be easily implemented by optical fiber in analogue domain [72, 74, 75], keep the information of historical data points within the loop as a fading memory. Differ from the software-based reservoir (Eq. (1.1)), the Eq. (2.2) describes the analogue signal in hardware reservoir over continuous time t. When a data point is sent to the delay-coupled node, the output contains not only the information of the current point, but also a certain portion of knowledge from historical inputs. In the

absence of nonlinear function, the system can be considered as a positive feedback system and its transfer function is:

$$Q(s) = \frac{\frac{1}{R_{int}C_{int}s+1}}{1-\frac{1}{R_{int}C_{int}s+1}D(s)}J(s) \tag{2.3}$$

$$D(s) = \sum_{n=1}^{N_{delay}} G_n e^{-n\tau s} = G_1 e^{-\tau s} + \ldots + G_{N_{delay}} e^{-N_{delay}\tau s} \tag{2.4}$$

where $D(s)$ approximates the delay unit, R_{int} and C_{int} form the time constant τ_{neuron} of the activation node as shown in Fig. 2.2(b) and (d), $Q(s)$ and $J(s)$ denotes state output and masked input in Laplace domain, respectively. In this work, Eq. (2.3) was solved in MATLAB/Simulink together with the nonlinear function extracted from the activation node to obtain the resulting $q(t)$. The configuration of this delay-coupled activation node determines the volume of historical information that can be preserved in the loop, which is known as 'memory capacity'. Furthermore, the fading memory is capable of preserving the information of the ECG morphology with only one input channel.

2.2.3 Lasso regression

Sampling the continuous node state $q(t)$ obtained by solving Eq. (2.3), and then concatenating every N discrete elements as one matrix can obtain **S**, which is a high-dimensional mapping for the each input data *u(n)* and the fading memory of historical data (*u(n-1), u(n-2)...*)

$$\begin{aligned}\mathbf{S} &= \begin{bmatrix} s(1) \\ s(2) \\ \vdots \\ s(n) \end{bmatrix} = \begin{bmatrix} q_1(1) & q_2(1) & \cdots & q_N(1) \\ q_1(2) & q_2(2) & \cdots & q_N(2) \\ \vdots & \vdots & \ddots & \vdots \\ q_1(n) & q_2(n) & \cdots & q_N(n) \end{bmatrix} \\ &= \begin{bmatrix} q(\theta) & q(2\theta) & \cdots & q(N\theta) \\ q(\tau+\theta) & q(\tau+2\theta) & \cdots & q(\tau+N\theta) \\ \vdots & \vdots & \ddots & \vdots \\ q(n\tau+\theta) & q(n\tau+2\theta) & \cdots & q(n\tau+N\theta) \end{bmatrix}\end{aligned} \tag{2.5}$$

where $q_i(j)$ denotes the state of i^{th} virtual node after j^{th} input, and $Q(j)$ is the state matrix for input $u(j)$. The elements in the last matrix ($q(\theta)$...) can be obtained by solving Eq. (2.3). Thus, output weight $\mathbf{W_{out}}$ of the DRC can be calculated through a linear regression of $\mathbf{Q}$ and a desired output Y. Lasso regression is chosen as the regression method. Proposed by [129], Lasso provides a sparse linear regression by adding L_1-norm regularisation to ordinary least squares regression for preventing overfitting. After sampling the node states of training data, the output weights can be obtained by minimising the loss function of the Lasso regression:

$$\mathbf{W_{out}} = \underset{W}{argmin}\{\sum_{i=1}^{L}(y_i - \sum_{j=1}^{N} q_i(j)W_j)^2 + \lambda \sum_{j=1}^{N}|W_j|\} \tag{2.6}$$

where L is the length of training data, y_i is the training label that will be discussed in the next section and λ represents the regularisation parameter that determines the strength of the L_1 penalty [129, 130]. Finally, the predicted output Y' can be written as:

$$\mathbf{Y'} = \mathbf{SW_{out}}, or\, y'(n) = s(n)\mathbf{W_{out}} \tag{2.7}$$

Different from L_2-norm regularisation (Ridge regression), minimising the absolute value of coefficients will result in a sparse output connection by automatically eliminating redundant coefficients ($W_j = 0$), and that behaves like a coefficient selector. L_1 norm is advantageous and suitable for this application because sparse model reduces the number of components used to build a post-processing circuit in the next stage. In addition, the node states present a certain level of multicollinearity in our simulation. The Lasso regression is well-suited to minimise the effect of multicollinearity.

2.2.4 Training data

The construction of training data and labels is shown in Fig. 2.4(a). The blue line is the

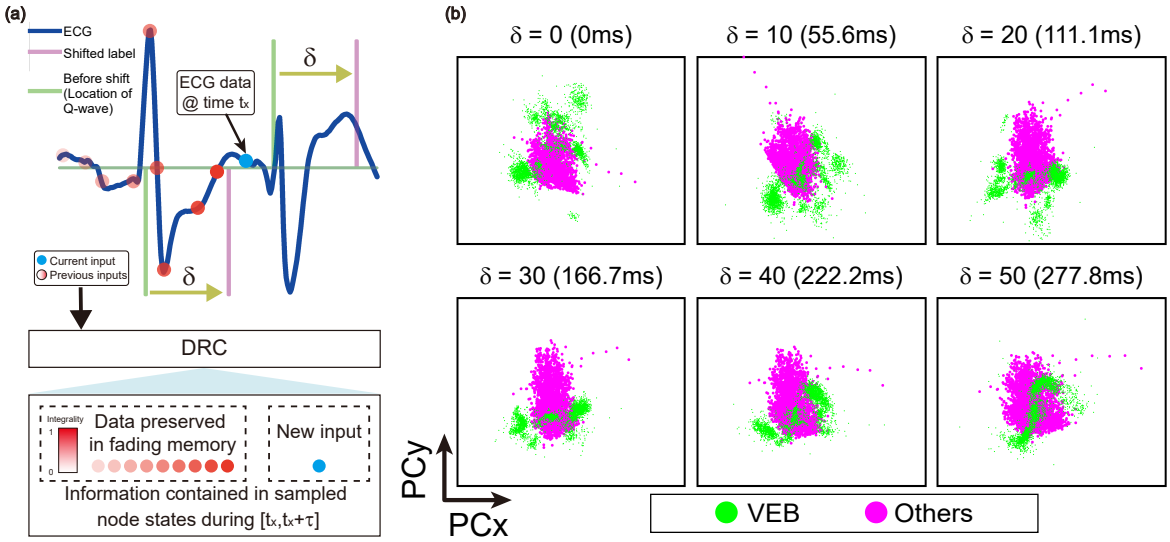


Fig. 2.4 (a) Schematic of the training data construction and the effect of fading memory. When the blue point is sent to the DRC, the information retained in the node state includes not only the current data (blue), but also the historical data (red). However, the historical data is not intact since the memory is fading. The top graph also illustrates the construction of training labels. The green line highlights the location of VEB (positive value) and other types of heartbeat (negative value). The pink line is a shifted version of green line. (b) Visualization of the node state distribution in high-dimensional feature space at δ points shifted away from the label location from database. PCA was used to reduce the state dimension from 400 to 2 for visualization.

training ECG data with the round dots representing each sample points. As mentioned above, the end-to-end setup receives the input data points one by one. The blue dot indicates the point injected to the system at the time stamp t_x while the red dots are the historical inputs at $t_x - \tau$, $t_x - 2\tau$ ..... The node states $q(t)$ during [t_x, $t_x+\tau$] contain high-dimensional information of not only the blue dot, but also of the historical inputs (red dots). This is because the information of previous points is preserved in the fading memory created by the delay-coupled loop. However, one data point cannot be maintained permanently and will be attenuated after each cycle. This feature is crucial for constructing training data. The MIT-BIH database has labelled the location of Q-wave (the central spike for MLII lead) in each heartbeat. There are three steps for defining training label Y:

- The label *y(n)* is based on the ECG type from the database and has the same length as input data. For the purpose of detecting VEB, the locations of VEB are set to 1, the locations of other types of ECG is set to -1 and the rest of the labels are 0 (the green line in Fig. 2.4)).
- Right shift the *y(n)* by δ points (δ /180ms). The label 1 and -1 are moved from the location of Q-wave to the end of a heartbeat (pink line in Fig. 3). The reason is that the node state *q(n)* at the end of each heartbeat includes the fading memory of the entire heartbeat, which is analysed with MC in the next subsection.
- The number of VEB (1) and other types of heartbeat (-1) are unbalanced in terms of training labels, leading to the hyperplane that separates the two classes of data in high-dimensional space getting closer to the majority. Therefore, the desired output *y(n)* should use $\frac{n_1+n_2}{n_1}$ and $-\frac{n_1+n_2}{n_2}$ instead of 1 and -1, where n_1 and n_2 are the number of VEB and other types of heartbeat respectively in the training data set.

Determining the shifting distance δ of labels should consider the state $\mathbf{Q}$ distribution and dispersion between different classes. With the ECG input, the state *Q(n)* implies the mapping

of $(u(n), u(n-1), u(n-2)\ldots)$ into a N-dimensional feature space. Generally, the fewer overlaps between different classes of ECG in the feature space, the better discrimination can be obtained by the regression. $\mathbf{Q}$ over different δ is visualised and plotted in Fig. 2.4(b). At the first step, $\mathbf{Q}$ was collected by feeding the ECG data to the DRC model. The green dots of each graph are the states at the locations of the shifted label which is δ points away from the Q-wave. For example, when $\delta = 0$, the states of green dot were collected at the location of Q-wave. In order to observe the dispersion between the extracted VEB states and normal ECG, the states of red dots were randomly collected within the range of other heartbeat classes including normal beat, left bundle branch block beat and right bundle branch block beat. After collection, the N-dimensional data of red dots are consistent for all six graphs. Here total 20395 heartbeat data points, including 15% VEB (green), were illustrated for each graph. For visualising the high-dimensional data, Principal Component Analysis (PCA) was used to map the data into two dimensions. As shown in Fig. 2.4(b), the two axes represent the first and second Principal Component (PC). In each graph, the result of PCA also shows that the first two PCs can explain 88.19% (std 1.87%) on average of all variances. When $\delta = 40$, the two classes of data points present less overlap. Therefore, the label was right-shifted for 40 samples (222.2ms) for training.

2.2.5 Memory capacity

Fig. 2.4(a) shows that the DRC can retain historical inputs in the fading memory. MC is a key criterion that indicates the volume of information from the previous inputs that can be retained in the network. MC depends on the parameters and structure of the reservoir. The crucial variables that affect the MC are the nonlinearity of activation node, the strength of each delay line $(G_1, G_2 \ldots G_{N_{delay}})$, the ratio between the strength of feedback and input ($\beta = G_f/G_i$) and the ratio between each delay line ($\gamma = G_2/G_1$) [122, 131]. Here, the nonlinear activation region leads to a rather low MC and lots of delay lines are needed to compensate

for the loss of MC. Therefore, the linear region with two delay lines ($G_3 \ldots G_{N_{delay}} = 0$) was chosen since our single input channel required high MC and too many delay lines are not optimal in hardware design. In order to keep the system stable, two requirements should be fulfilled: $G_f + G_i = 1$ and $G_1 + G_2 = 1$. Two tasks were employed to analyse the MC of the DRC model: *1)* the task of binary sequence reconstruction is a standard method to quantify the MC; *2)* a 'look back' ECG reconstruction task for validating the ECG morphology preserved in a fading memory.

Binary sequence reconstruction

A binary input $p(n)$ = -1 or 1 was randomly generated to evaluate the MC of the proposed reservoir. In this task, the training output $y_i(n)$ is the matrix of $p(n)$ shifted by i steps for i = 1, 2 $\ldots \infty$, which means that each state at the input $p(n)$ will be used to train and reconstruct the historical points of a square wave. The MC can be calculated as the sum of a linear correlation between reconstruction result $x_i(n)$ and the actual shifted sequence $y_i(n)$:

$$y_i(n) = p(n-i) \tag{2.8}$$

$$\begin{aligned} MC = \sum_{i=1}^{\infty} m(i) &= \sum_{i=1}^{\infty} \rho\left(x_i(n), y_i(n)\right) \\ &= \sum_{i=1}^{\infty} \frac{< y_i(n)\, x_i(n) >_n^2}{\sigma^2\left(p(n)\right)\sigma^2\left(x_i(n)\right)} \end{aligned} \tag{2.9}$$

where ρ and σ denotes the correlation and variance, respectively [122, 123]. Theoretically, the summation of $m(i)$ should be taken from i = 1 to ∞, which is incalculable. Thus, $i \in$ [1, 600] was adopted. The $p(n)$ was randomly generated with the length of 4000 (75% for training and 25% for testing) and consistent throughout the test. Also, the Ridge regression, instead of Lasso, was applied to the square wave reconstruction because a sparse $\mathbf{W_{out}}$ trained by Lasso may cause information loss. This means the result cannot accurately reconstruct

the square wave and fully reflect the amount of information retained in node states. During training, the shifted binary sequence $y_i(n)$ was used as the label, which means the node state collected at the input $p(n)$ is employed and trained to reconstruct the input at i steps prior to ($p(n-i)$). This task tests the capacity of the DRC to retain the historical samples due to the fact that the memory is fading. Reconstructing the sample becomes increasingly demanding as i becomes higher. Therefore, the MC can be quantified by analysing the correlation between $y_i'(n)$ and $y_i(n)$ in Eq. (2.9).

In this task, the two ratios, β and γ, are investigated. The MC is varied by different β, while γ was implemented to control the shape of $m(i)$. Fig. 2.5(a) shows a reconstructed sample for lower MC when β=9.1 and γ=0.1. When the shifting steps are small (i=7), the result (green dots) can capture most of the original points. As i increases, the reconstructed points depart from the targeted value with the decline of $m(i)$. In Fig. 2.5(b), raising γ to 10, which means that the percentage of 2$^{\text{nd}}$ delay line is increased, results in a small improvement in MC. As can be seen from the graph, most of the reconstructed points are close to the original data $p(n)$, whereas a small fluctuation was also found when i=19. The effect of MC can be observed through the comparison of the two graphs. A higher MC will enhance the reservoir's ability to retain historical inputs in the current node state. Subsequently, a higher γ will improve the MC because the proportion of the 2$^{\text{nd}}$ delay line (length = 2τ) is increased. The $m(i)$ can be computed based on the correlation between reconstructed data and original data over i, according to Eq. (2.9). Firstly, β is fixed and γ is varied from 0.01 to 100. The resulting $m(i)$ is plotted in Fig. 2.5(c). When γ=100 (the 2$^{\text{nd}}$ delay line is highly dominant), the unbalanced distribution of $m(i)$ for odd and even i was found. The reason for this issue is that the 2$^{\text{nd}}$ delay line facilitates the reconstruction of even i step shifting. In addition, the rest of the lines show that the $m(i)$ is slightly reduced as γ declines, and the distribution of $m(i)$ can also be tuned by γ. Secondly, γ is fixed and β is varied. The Fig. 2.5(d) shows that β has a much stronger effect on MC compared to γ. In the next step, the MC can be

quantified by taking the summation of the correlation $m(i)$, which is visualised in Fig. 2.5(e). In conclusion, the MC varies more significantly when β changes. The higher the β and γ, the stronger MC can be achieved. More specifically, MC mostly depends on the β, while γ can slightly change the distribution of $m(i)$.

'Look back' ECG reconstruction

The 'look back' ECG reconstruction is a multistep backward signal reconstruction task. The purpose of this task it to numerically proof the fading amount of ECG information retained in the delayed feedback loop. If the state matrix collected at the end of each heartbeat can reconstruct the entire past heartbeat episode, this state matrix should include a certain amount of historical ECG information and thus can be used to classify different heartbeat types. In this task, the node states at the end of each ECG beats were used for training to reconstruct the historical n_1 points. For example, the node state at time t_x in Fig. 2.4(a) that was extracted as one of the $Q_M(n)$ and $y_M(n)$ is a matrix of ECG data segments from $t_x - \tau n_1$ to t_x, where $Q_M(n)$ and the $y_M(n)$ are the node state and the training label for this task. In total, n_2 groups of node state and their corresponding ECG slots were collected to calculate the output weight W_M for training. During the testing, the node states at the end of a heartbeat were collected to reconstruct ECG using Eq. (2.7). Assuming that the t^E is an array recording the location

of each heartbeat in an ECG dataset, the elements in $S_M(n)$ and $y_M(n)$ can be written as:

$$\begin{aligned}\mathbf{S_M} &= \begin{bmatrix} s_M(1) \\ s_M(2) \\ \vdots \\ s_M(n_2) \end{bmatrix} \\ &= \begin{bmatrix} q\left(t_1^E+\theta\right) & q\left(t_1^E+2\theta\right) & \cdots & q\left(t_1^E+N\theta\right) \\ q\left(t_2^E+\theta\right) & q\left(t_2^E+2\theta\right) & \cdots & q\left(t_2^E+N\theta\right) \\ \vdots & \vdots & \ddots & \vdots \\ q\left(t_{n_2}^E+\theta\right) & q\left(t_{n_2}^E+2\theta\right) & \cdots & q\left(t_{n_2}^E+N\theta\right) \end{bmatrix}\end{aligned} \tag{2.10}$$

$$\begin{bmatrix} y_M(1) \\ y_M(2) \\ \vdots \\ y_M(n_2) \end{bmatrix} = \begin{bmatrix} u\left(t_1^E\right) & u\left(t_1^E-\tau\right) & \cdots & u\left(t_1^E-n_1\tau\right) \\ u\left(t_2^E\right) & u\left(t_2^E-\tau\right) & \cdots & u\left(t_2^E-n_1\tau\right) \\ \vdots & \vdots & \ddots & \vdots \\ u\left(t_{n_2}^E\right) & u\left(t_{n_2}^E-\tau\right) & \cdots & u\left(t_{n_2}^E-n_1\tau\right) \end{bmatrix} \tag{2.11}$$

where n_1 = 400 (~2.2s) and n_2 = 2000 were used in this task. Using the $\mathbf{s_M}$ and Y_M, the weights $\mathbf{W_{out}}^M$ and the reconstructed output can be calculated by ridge regression and Eq. (2.7). The reconstruction results at MC equal to 120, 70 and 20 along with the original waveform are plotted in Fig. 2.6(a). The upper graph illustrates the reconstruction of two normal beats and the lower one shows a ventricular beat followed by a normal beat. When MC=120, the output can perfectly copy the later beat and tend to follow most of the earlier beat. In contrast, the low MC (20) can only retain the rough shape of the later beat and has almost no information about the earlier beat. This phenomenon can be quantified by taking Mean-Squared Error (MSE) between the reference and reconstructed data. A more comprehensive experimental result of ECG reconstruction using 1000 heartbeat episodes from the database is shown in Fig. 2.6(b). It reveals that when MC is increased, there is a continuous decrease in MSE as well as the interquartile range. Also, a normal beat is easier

to reconstruct as compared to a ventricular beat. The result of this task further proves that the node state generated at the end of one ECG beat contains the information of the whole ECG under the DRC model reported in previous sections. The accuracy of reconstructing the ECG and the length of rebuildable data based on the MC value is verified in the last task. On the other hand, the earlier data are harder to retain in the network. Moreover, it is worth mentioning that, for up-sampled or analogue ECG signal, the value of $\theta \times N$ should not decrease as τ in order to maintain enough MC in time domain.

Memory capacity for up-sampled ECG

The original sampling rate of the ECG data is 360Hz. In our simulation, it is down-sampled 180Hz to speed up the simulation and optimization. This section will discuss how to maintain sufficient MC for ECG processing in up-sampled case.

The memristive behaviour of the DRC stems from the integration elements of the neuron circuit and the delayed feedback. The MC implies the length of signal that can be retained in the network. However, this length depends on the network configuration including the neuron and delayed feedback regardless of the sampling rate of the input signal. In most of the cases, the sampling interval of ECG τ equals to $\theta \times N$ to facilitate the parameter analysis, so that the MC can be represented by the number of data points. The MC shortage would happen if the equivalence between $\theta \times N$ and sampling interval is maintained as the sampling rate increases. However, in analogue or up-sampled case, it is needed to redefine and separate the two parameters since they are no longer consistent: τ' is the sampling interval of ECG while τ (equals to $\theta \times N$) is the DRC operation period. A lower sampling interval τ' will be obtained if the $\tau = \theta \times N$ is unchanged ($\tau > \tau'$). Fig. 2.7 illustrates how to maintain sufficient MC for an up-sampled case (original sampling rate of the database) in comparison with the setup discussed in this work. In an up-sampled case, the masked signal during every τ was

divided into two parts because $\tau = 2 \times \tau'$. In analogue case (τ' close to 0), the masked signal will be a continuous pulse with smoothly changing amplitude.

Under this setup, the network can still remember the same time length of signal since the memristive properties of the hardware remains unchanged. This conclusion is endorsed by an experiment comparing the ECG reconstruction performance under different τ' (Fig. 2.8). This experiment is similar to the task discussed in last section. The MSE of ECG construction using the states collected at the end of the ECG fragment can directly reflect how much ECG information is retained in up-sampled cases. Note that the ECG reconstruction target is the 180Hz data, in order the make the two cases comparable. The result shows that the MSE is slightly lower in up-sampled case, which means that the memory capacity is still enough to process the ECG signal. The minor improvement could own to the amplitude change during τ, bringing extra state richness for high-dimensional mapping.

In conclusion, the up-sampled or analog signal will not compress the MC in time domain. In contrast, the up-sampled or analogue signal can be beneficial to the network performance by improving state richness. The down-sampled operation in this chapter is due to the following reasons: 1) lower sampling rate can demonstrate the system ability in tolerating lower quality signals; 2) down-sampling can reduce the number of data fed into the model and speed up the simulation; 3) lower sampling rate will relieve the burden of hardware development and cost.

2.2.6 Post-processing

After obtaining the output $\mathbf{Y}'$ in Eq. (2.7), ideally, a spike would appear when a VEB is sent to the model, whereas the output should keep flat for other types of ECG. This is because a spike is set at the end of every VEB in the training shifted label. However, fluctuation always happens throughout the output signal since some of the components for other types

of ECG are similar to the VEB components. The Fig. 2.4(b) also demonstrates this issue as there always exists a certain amount of overlap. Thankfully, most of the spikes of VEB are higher than the unwanted noise. Therefore, a thresholding approach was adopted to capture the spikes of interest. Before the threshold, another two filters, which are the same as the high-pass and low-pass filters in the pre-processing step, were applied to the output $y'(n)$ to support the thresholding. In this step, the location of the possible VEB is highlighted by spikes after thresholding.

2.3 Performance measures and results

2.3.1 Performance matrix for VEB detection

The performance matrix for the VEB detection is recommended by AAMI. The metrics include sensitivity (Se), positive predictivity (PP), specificity (Sp), and accuracy (Acc), which are the standard statistic tools for evaluating the ECG classification on MIT-BIH database. In addition, F1 score, the harmonic mean of Se and PP is also used to optimise the parameters. These values can be calculated using the values of True Positive (TP), True negative (TN), False Positive (FP) and False Negative (FN) [127]:

$$Se = TP/(TP+FN) \tag{2.12}$$

$$PP = TP/(TP+FP) \tag{2.13}$$

$$Sp = TN/(TN+FP) \tag{2.14}$$

$$Acc = (TP+TN)/(TP+TN+FN+FP) \tag{2.15}$$

$$F1 = 2(Se \cdot PP)/(Se+PP) \tag{2.16}$$

The design goal is to minimise the FN and FP and maximise the TP and TN. It is worth mentioning that, in the absence of heartbeat segmentation, the performance quantification in this work is slightly different from the beat-by-beat comparison reported in the literature. Instead, the model output is a point-by-point indication because the raw ECG signal is continuously fed into the model which acts like a dynamical nonlinear system. To count the number, a threshold was applied to the output $y'(n)$ generated by ECG input. The threshold can highlight the spike of the output signal which may suggest the location of a VEB. If the location of one spike matches the VEB annotated in the database, this VEB will be counted as a TP. Similar approaches were also used to count TN, FP and FN.

2.3.2 Optimisation

The MC allows the model to load the ECG morphology to the recurrent network using only one-dimensional input without signal segmentation and feature extraction. However, a high MC will cause redundant information, such as previous multiple beats, preserved in the loop. Prior to calculating the final result, the F1 score was chosen as a standard to optimise the performance. In order to speed up the optimisation process, a descriptive dataset including 1138 normal beats and 420 VEBs randomly collected from DS1 was used to optimise the parameters. The F1 over different MC was firstly simulated. Fig. 2.9(a)

provides the relationship between MC and F1. The green line presents a curve fitting using 8^{th} order polynomial. The highest point in this simulation has been amplified and plotted. Before the maximum point, F1 keeps rising as MC increased. This is because the gain of MC enhances the model's ability to keep the heartbeat information. Afterwards, the F1 is reduced with the growth of MC because the redundant information makes the VEB detection more challenging. The highest MC achieved by current double delay model is less than 130. The highest F1 occurs when MC is around 75. Based on this value and the results

in Fig. 2.5(e), the β and γ generating MC approximately from 70 to 100 are evaluated by F1 (Fig. 2.9(b)). The highest F1 was obtained by β = 13.8 and γ = 3.01 in the condition that MC = 91.8.

The ratios discussed above had been used to deploy the model in which all ECG data in DS1 were tested. The output threshold versus the performance matrix are plotted in Fig. 2.9(c) and (d). At a low threshold value, the number of spikes including errors and noises were captured, resulting in a high FP and TP. In this case, unwanted noises were incorrectly detected as VEBs spikes. In contrast, a high threshold leads to the large number of missing VEBs (high FN and FP). Based on Eq. (2.12)-(2.16), the performance matrix is plotted in Fig. 2.9(d). The optimal F1 can be obtained when threshold is 0.3.

2.3.3 Results

The result of MC test and optimisation have been discussed in the previous sections. After optimisation, the entire testing dataset DS2 was fed to the model. In addition, four examples of the output signal together with the optimised threshold value, input signal and ground truth (the shifted labels of VEB in DS2) are shown in Fig. 2.10. One pair of matched dot and rhombus is counted as a TP, while FP is counted when no spike is found in the range of other types of heartbeat. As can be seen from the graph, most of the VEBs can be detected by the output spikes. At the same time, the values outside the VEB were kept low. In every TP, short displacements between the ground truth and result always existed because of the continuous point-by-point detection. However, the Record 233 contains the highest amount of multiform VEBs, which can only be partially detected due to its sharper waves. The unreadable artefact of the data such as few episodes in Record 105, which has also been reported in the database description, resulted in high TN and FN. The final performance matrix was calculated by the

gross TP, TN, FP and FN, which is listed in Table I. The proposed hardware-based model yielded Se = 80.9%, PP = 87.5%, Sp = 99.2% and Acc = 98.0%.

Table 2.1 The result of VEB detection

Record	Se (%)	PP (%)	Sp (%)	Acc (%)	F1 (%)	# of VEB
100	100.0	33.3	99.9	99.9	50.0	1
103	-	0.0	99.8	99.8	-	0
105	12.2	4.1	95.4	94.1	6.1	41
111	100.0	8.3	99.5	99.5	15.4	1
113	-	-	100.0	100.0	-	0
117	-	0.0	99.7	99.7	-	0
121	100.0	4.2	98.8	98.8	8.0	1
123	100.0	50.0	99.8	99.8	66.7	3
200	91.8	100.0	100.0	97.4	95.7	826
202	5.3	14.3	99.7	98.9	7.7	19
210	39.7	96.3	99.9	95.4	56.2	194
212	-	0.0	99.7	99.7	-	0
213	96.8	74.5	97.3	97.3	84.2	220
214	60.9	100.0	100.0	95.5	75.7	256
219	82.8	98.1	100.0	99.5	89.8	64
221	97.7	100.0	100.0	99.6	98.9	396
222	-	0.0	97.2	97.2	-	0
228	98.1	96.2	99.2	99.0	97.1	362
231	100.0	28.6	99.8	99.8	44.4	2
232	-	0.0	98.8	98.8	-	0
233	71.1	99.7	99.9	90.8	83.0	831
234	0.0	0.0	99.9	99.8	-	3
Overall	80.9	87.5	99.2	98.0	84.0	3220

2.3.4 Minimum memory needed for inference

In wearable devices, the trained VEB detection model can be deployed in a edge device where the hardware cost needs to be considered. Minimizing the memory that needs to be accessed for the detection is a crucial approach to reduce the hardware cost. While the state-of-the-art systems require massive memory for storing network parameters and performing nonlinear calculation, the memory needed for deploying the proposed model is

significantly lower since the major proportion of nonlinear computing occurs in the analogue domain. The absence of signal segmentation and feature extraction reduces the memory requirement to zero for the data before the input player. Furthermore, the fixed physical reservoir layer significantly saves the number of parameters for the network activities. In the proposed model, only the output of every time-multiplexing step needs to be stored and multiplied by the corresponding weight for the 'multiply and accumulate' operation. Thus, the total number of parameters in the model is 401 (400 weights and 1 network output at the current time step). Assuming that the data type is double precision floating point (8 bytes, as the simulation setup), the minimum size of memory for inference is 3.13 MegaByte(MB). In addition,under the proposed architecture, the memory is proportional to the number of virtual nodes. It means that the memory requirement would not exponentially increase if the network size is expanded for detecting more heartbeat types or higher performance matrix.

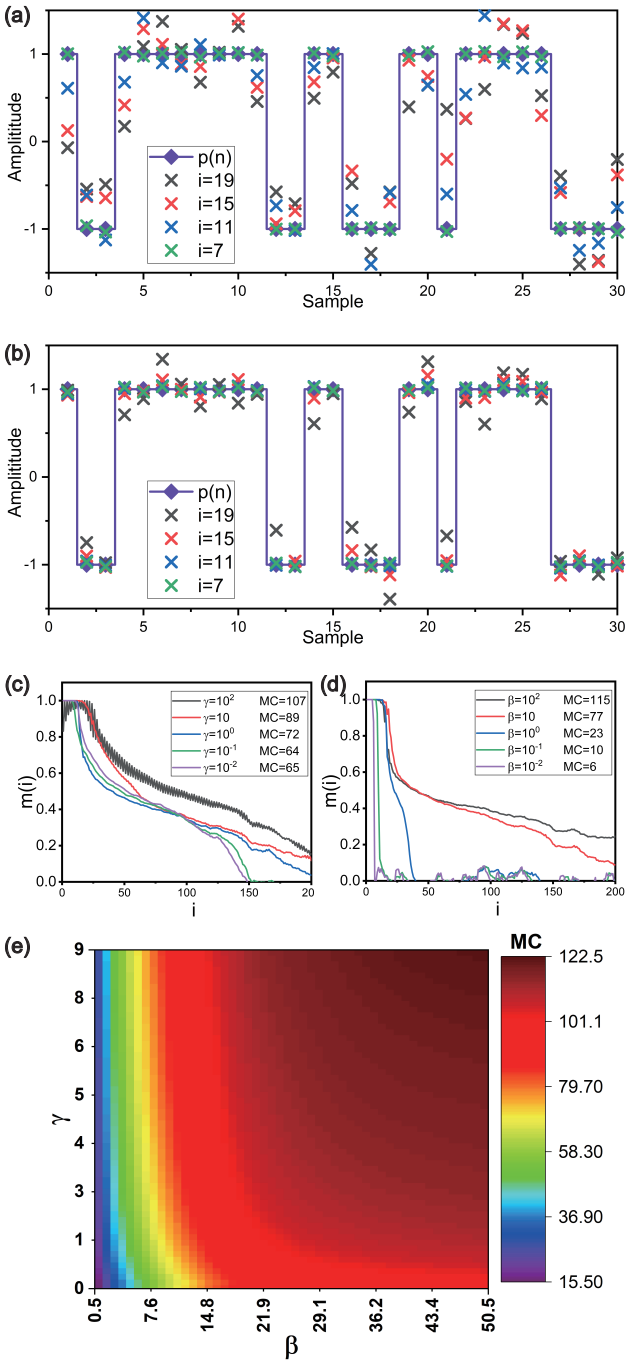


Fig. 2.5 Result of memory capacity testing using binary sequence reconstruction at (a) lower MC ($\beta = 9.1$ and $\gamma = 0.1$ and (b) higher MC ($\beta = 9.1$ and $\gamma = 10$. Based on the reconstruction results over different i, the correlation graph $m(i)$ can be computed. (c) The $m(i)$ curve with fixed β and (d) The $m(i)$ curve with fixed γ. The corresponding MC values are also provided. (e) The MC value as a function of the two ratios, β and γ.

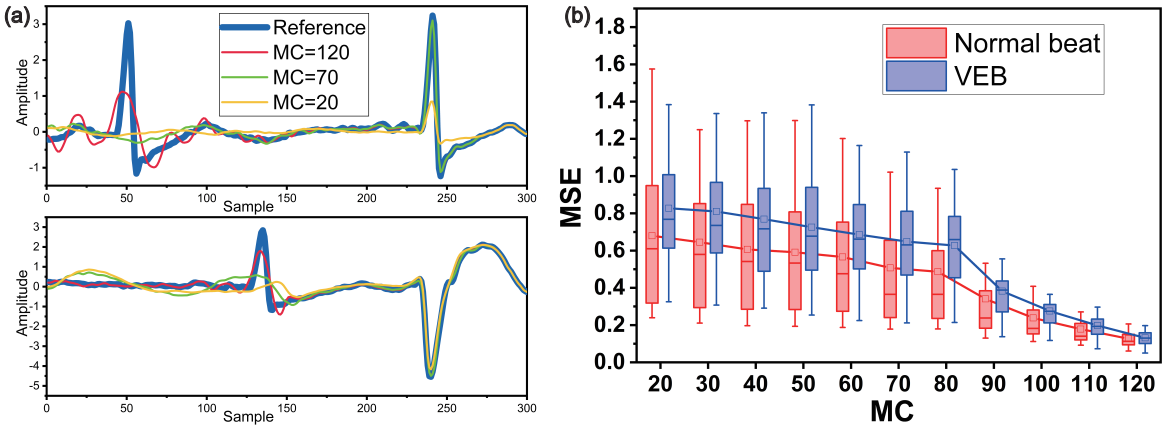


Fig. 2.6 (a) The result of the 'look back' ECG reconstruction task under different MC values. Under DRC models with different MC, the state matrix collected at the end of a heartbeat was used to reconstruct the past 400 ECG points (approximately two continuous heartbeats). The top graph shows the reconstruction of two normal beats and the bottom graph shows one VEB and one normal beat. The blue line is the reconstruction target. (b) The MSE between the reconstructed line and the reference data over MC value. The values of normal beat and VEB are plotted separately.The reconstruction results demonstrate the memristive property of the DRC model that preserving the historical information within the network.

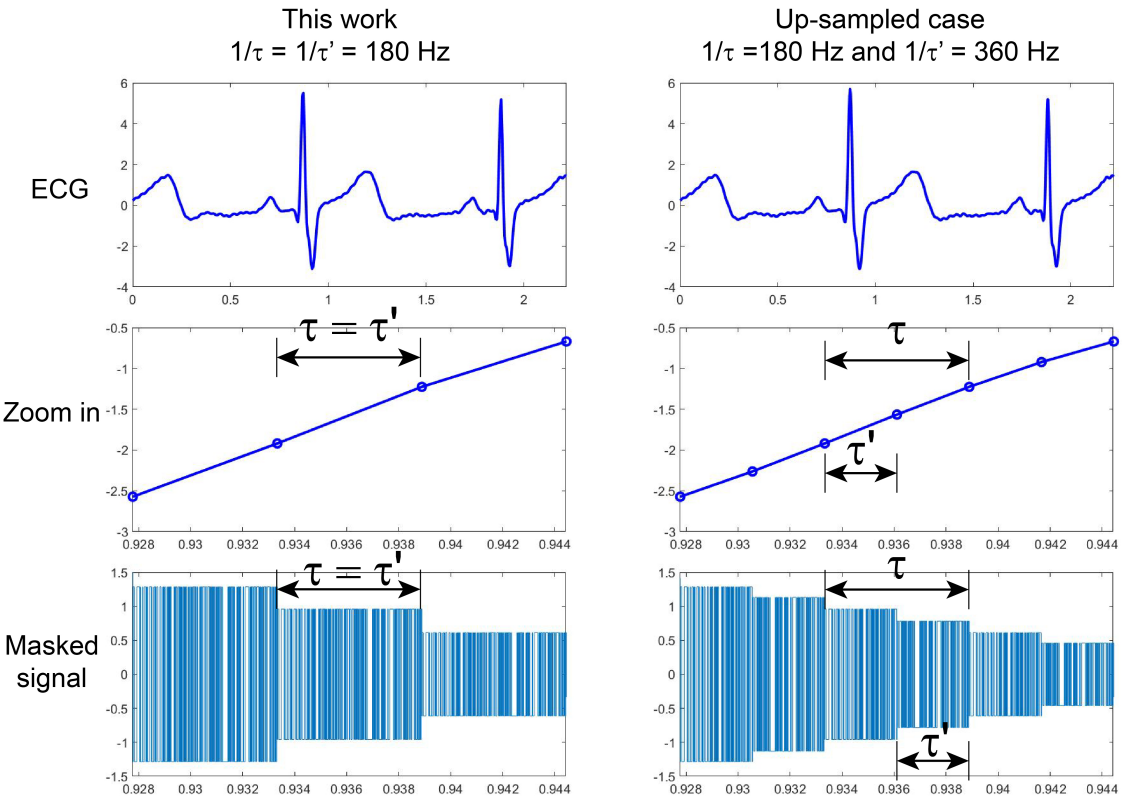


Fig. 2.7 ECG signal and Masked signal under different setup of τ and θ'.

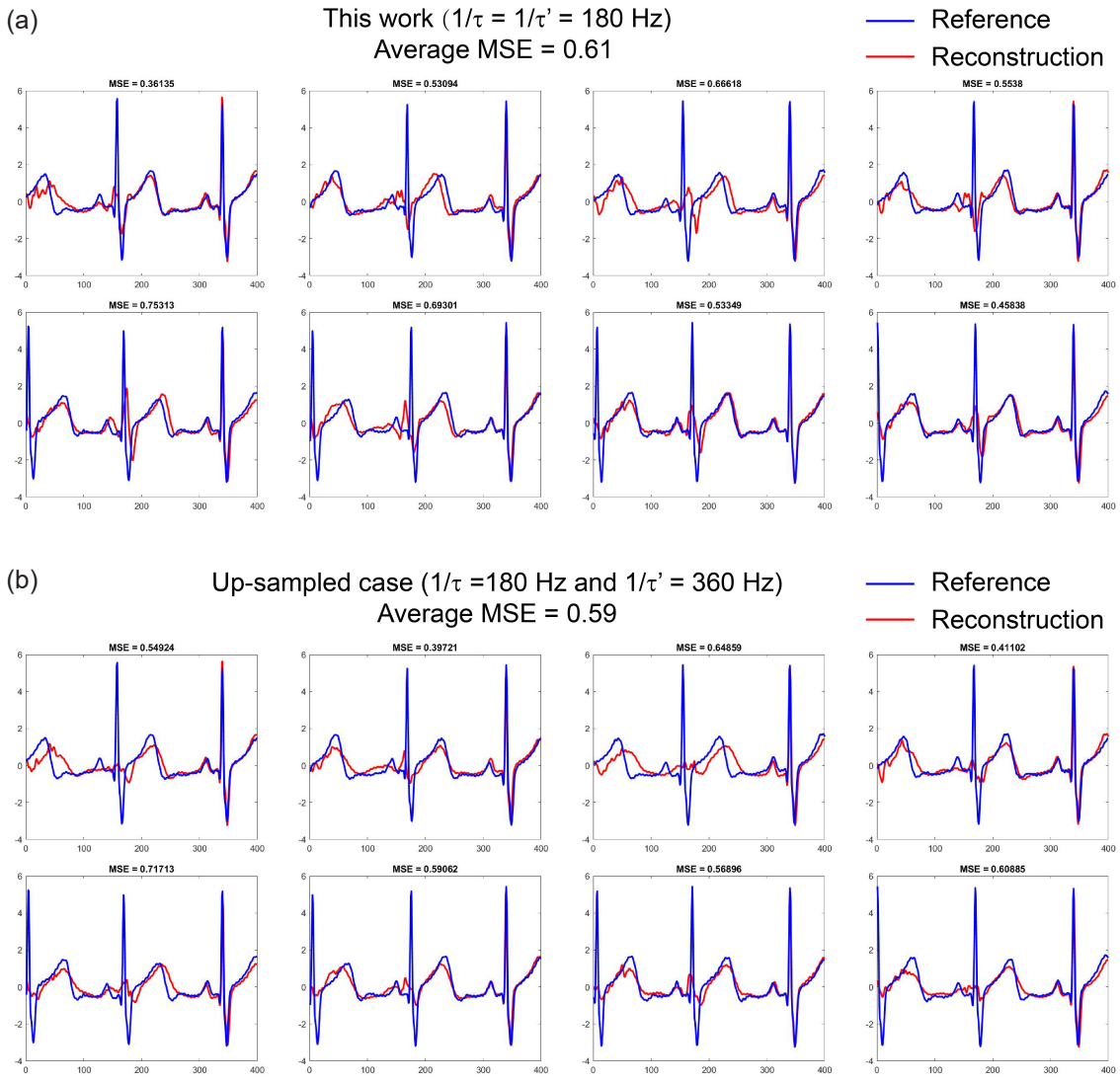


Fig. 2.8 ECG construction task under different τ'.

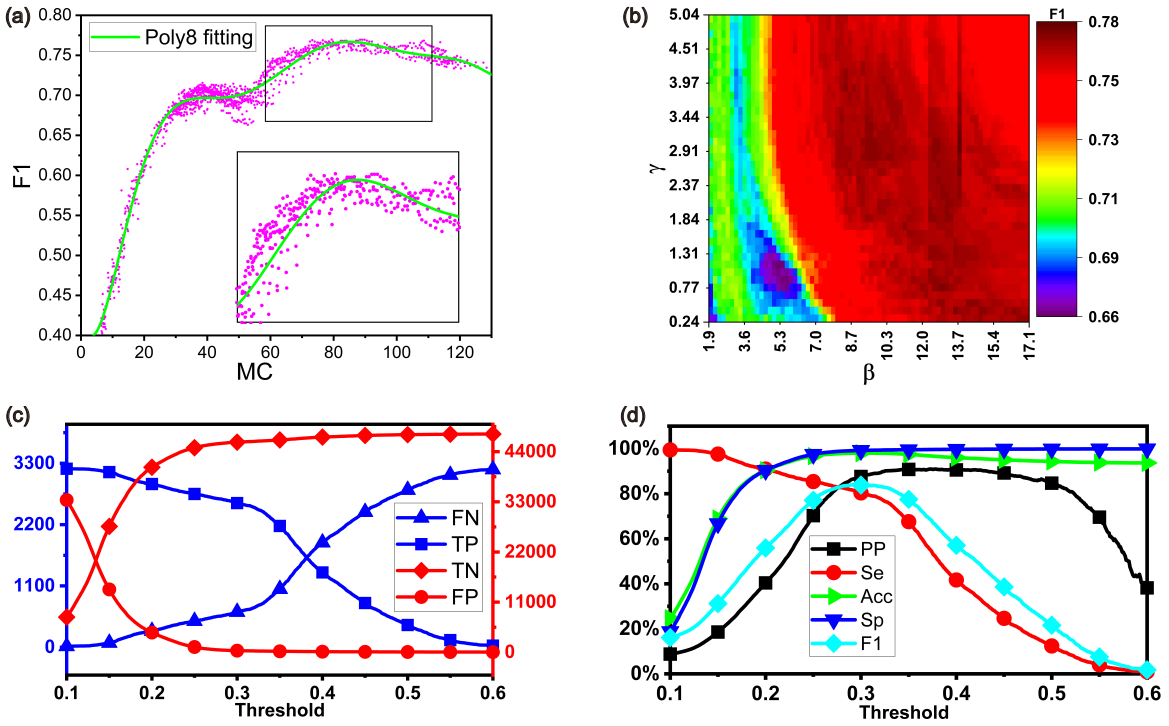


Fig. 2.9 Result of the parameter optimization and performance matrix. (a) The F1 score over MC. The green line stands for the curve fitting of the pink dots using an 8th order polynomial. The data was obtained from the simulation of MC over the two ratios and its resulting F1 scores. (b) The F1 score as a function of β and γ. The ranges of β and γ are the values producing MC from approximately 70 to 80, which is the range of the optimal F1 was computed in (a). The threshold evaluation in terms of (c) TP, TN, FP, FN and (d) performance matrix.

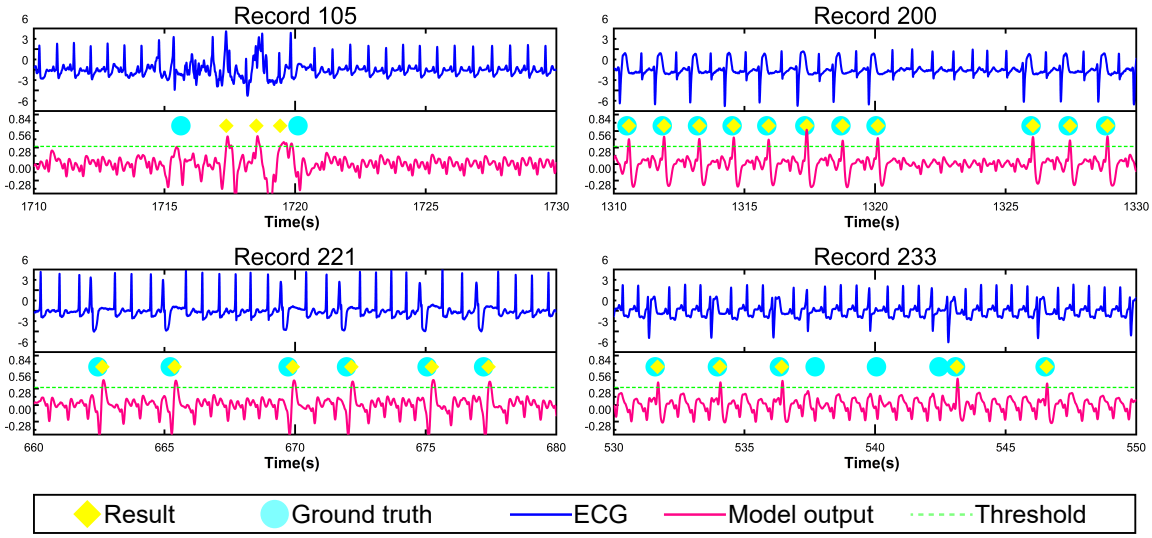


Fig. 2.10 Four episodes of the ECG and their processing output. The top raw shows the input ECG signal. The bottom row shows the output of DRC model. The red dots denote the locations of VEB from database. The yellow rhombuses are the VEB detection results. For example, the artifact in Record 105 led to several FPs, and the multiform VEB in 233 resulted in FPs. Meanwhile, most VEB can be successfully detected in Record 200 and 221.

Table 2.2 Comparison table of recent researches using intra-patient paradigm and MIT-BIH database

Work	Hardware -based?	Digital operation		Processing core	Resampling rate	Feature set	Effectiveness for VEB				Minimum memory per class (MB)
		Segmentation	Feature extraction				Se (%)	Acc (%)	Pp (%)	Sp (%)	
[105]	No	Yes	Yes	DBN	360	Intervals, morphology (with PCA)	85.8	97.9	-	98.8	>1522
[117]	No	Yes	Yes	PSO +SVM	360	Raw data, wavelets, TVCG	87.3	92.4	59.4	4.1	-
[118]	No	Yes	Yes	SVM	360	Intervals, SST, etc.	77.5	82.7	79.1	-	-
[47]	No	Yes	Yes	ESN	250	Intervals, std, average, etc	88.7	98.3	89.2	99.1	>8810
[46]	No	Yes	Yes	ESN	250	Raw ECG, RR-interval	92.7	-	95.7	-	>508
This work	Yes	No	No	DRC	180	Raw ECG	80.9	98	87.5	99.2	3.1

2.3.5 Comparison with the state-of-the-art

The methodologies of automatic ECG classification have been widely explored in recent years. Using the open-access MIT-BIH arrhythmia databases [116], previous researches were done on automatically classifying different types of ECG by the intelligent algorithms such as support vector machine [117–119], echo state network [46, 47], decision tree [120], and neural network [105, 121]. As summarised in Table II, the selected publications are the state-of-the-art studies with the following criteria: *1)* published in the past five years, *2)* evaluated by MIT-BIH arrhythmia database, *3)* inter-patient evaluation paradigm, *4)* different types of processing core algorithms and *5)* yielded good results. As can be seen from the table, the prior ECG signal processing studies mainly focused on software-based methods. The minimum size of memory was calculated using the method reported in the last section. The data type was double precision floating point unless a specific data type or fixed-point operation was used. For example, it has been reported that the ring ESN was designed to detect arrhythmia in [46]. First, the segmented heartbeat was 60 samples (240ms). Second, total 63-dimensional vector including raw data and features were sent to the input channels. Next, given the reservoir size is 1000, the size of input weight is 63×1000=63000. In the reservoir layer, the ring ECG hugely reduces the number of weights to 1000 in comparison of 1000×1000 in normal ESN. Combining with another 1000 data points for storing node states, the total number of variables in this network is 60+63+63×1000+1000=65123, and the corresponding memory size divided by the number of heartbeat types under detection is 508.8MB. This estimated number indicates the minimum number after fully optimizing the system, and the actual system should require larger memory. The Support Vector Machine (SVM)-based classifier involves complex operation using nonlinear kernel function to map the data or features to higher dimensional space. These operations will increase the burden of both processor and memory [117, 118] . The parameter reported is not sufficient to estimate the minimum memory. The algorithms with large network size such as Deep Belief Network

(DBN) [117] and 1500-neuron ESN [47] demands high memory and intensive nonlinear computing, which may not be suitable to deploy in a wearable edge device.

Compared to the state-of-the-art automatic ECG processing study, an important difference is that the proposed model simulates a dynamic neuromorphic system receiving continuous ECG signal, rather than designing an offline machine learning framework. It is advantageous in a number of ways: *1)* the two digital operations, signal segmentation and feature extraction, are removed from the processing model. This difference facilitates the implementation of a pure analogue neuromorphic processor; *2)* it allows raw ECG signal from single lead flow into the model for detecting VEB and meanwhile obtains the acceptable result of effectiveness, which can be also considered as near-sensor computing; *3)* it simulates a neuromorphic dynamic system, rather than a pure algorithm, which implies the processing and VEB detection happens in real-time; *4)* the relatively lower sampling rate reduce the overall system frequency for real-time processing ;*5)* inherited from RC, the training of this model is relatively easy and fast. These advantages can be measured by the minimum memory size. The proposed system requires significantly lower memory, only 3.1MB, for running the detection algorithm when the fully trained model is deployed.

2.4 Discussion and conclusions

In this chapter, an ECG signal processing model based on DRC for hardware implementation has been proposed for the first time. This model was evaluated by an abnormal heartbeat detection task with the MIT-BIH arrhythmia database. Considering the notion of neuromorphic engineering, the model design refrained from using digital signal processing components. The novelty of the proposed model can be summarised as follows:

(i) Conventionally, heartbeat segmentation and feature extraction are two routines of automatic ECG classification in the previous studies. They were sidestepped in this model

since these operations are much more friendly for software implementation rather than hardware. Instead, the method proposed in this chapter is to analytically test the MC and to preserve the desired amount of ECG morphology in the recurrent loop, which has been tested by two MC validation tasks. Accurate modelling of MC is crucial for designing the DRC model for processing a specific type of signal.

(ii) This model is a end-to-end dynamic system: the input is raw ECG signal and output is a point-by-point indication of signal class. As discussed Table II, the performance matrix in this work is comparable to the software-based methods in the literature considering that the hardware-based method is not as flexible as a software-based method.

(iii) The main signal processing takes place in the analogue domain. It avoids suffering the intensive data transmission and processing in memory and processor, and therefore the memory needed for executing detection algorithm can be greatly reduced compared with previous studies. This advantage would potentially be useful to form a low-power neuromorphic computer [2, 20, 22].

Given the merits discussed above, the main weakness is the limited computing ability and task performance. In the absence of segmentation and feature extraction, the heartbeat information is preserved by the inherent memristive property of the network. These differences raise the task difficulty compared with the software-based algorithms where mass memory and accurate digital computing were used, resulting in a non-ideal performance. The higher computing performance could be obtained by scaling the network size in actual hardware, which remains future challenge.

In conclusion, this model provides a fundamental analysis of using DRC as computing architecture for ECG processing. To the best of our knowledge, this work is the first hardware-based model acting like a dynamic system which can highlight VEB from continuous ECG input. The model and the analysis of its dynamic property (such as MC) will facilitate the

future development of a neuromorphic wearable device, for instance, an analogue end-to-end abnormal ECG detector with built-in RC algorithm, to form a next-generation long-term ECG monitoring device at edge. Full development of such system remains a topic in future exploration. To achieve this, further efforts should involve: *1)* developing pre- and post-processing front-end circuit; *2)* analogue delayed feedback circuit; *3)* power consumption evaluation and optimisation; *4)* ASIC implementation.

Chapter 3

Volatile Memristor-based Reservoir Computing

3.1 Memristive devices for reservoir computing

Memristive devices have emerged as a crucial element in implementing physical neural networks. In 1971, prof. Leon Chua predicted the existence of memristor by theoretically analysing the behaviour of the basic electrical components [132]. Later on, HP Lab fabricated the first memristor in 2008 [133, 134]. One memristor can exhibit multiple conductance states, which can be modulated by external signals. In principle, the historical stimulation redistributes the migration and diffusion of oxygen ions of the devices, and thus changes the conductance [135–138]. In applications, memristors is widely employed in various physical networks or systems. It has been considered a strong candidate to implement artificial synapses. To adapt a physical neural network to different computing tasks, the plasticity of the neural network can be established by changing the weight values stored in the memristors. In this usage, memristors are expected to accurately store the weights for a long period even without receiving a signal or power, which is known as non-volatile memory

(NVM). Particularly, an NVM memristor array can achieve VMM at an extremely high efficiency subject to Kirchhoff's Current Law in hardware circuits [13, 96, 97]. Therefore, it is frequently used to accelerate the VMM operations in physical neural networks, including ANN [96, 98], spiking neural network [99, 138, 139], CNN [13] and the output layer of RC [80, 95, 140], as briefly discussed in Section 1. Additionally, non-volatile memristor can also be used to implement low-power logic units in digital circuit [141–143].

Recent research also revealed that memristive devices can also exhibit short-term memory (i.e. volatile memory) and nonlinearity. The effects of historical simulations accumulate if the continued input is received at short intervals. If no signal or relatively low amplitude is injected, the conductance state will gradually decay toward the initial state. These processes are subjected to a certain range of input-output nonlinearity. This type of memristor exhibits interesting dynamic behaviours in a single device, namely dynamic memristor or volatile memristor [63–65, 89]. Volatile memristor is well-suited to be the processing core of physical RC because of its miniaturised size, nonlinearity and intrinsic memristive properties. In the previous studies, volatile memristor based on WO_x [64, 65, 68], GeS [70], SnS [66], $W/HfO_2/TiN$[144], $Ti/TiO_2/Si$ [67], $Au/P(VDF-TrFE)/Cs_2AgBiBr_6/ITO$ [94] and TiO_x/TaO_y [63] were experimentally and numerically proved their effectiveness as processing cores in physical RC. Those volatile memristor-based RC have been successfully applied in tasks including chaotic series prediction, nonlinear system approximation, waveform classification, handwritten recognition, vehicle flow detection, face image classification, artificial olfactory etc..

In this section, a volatile memristor based on TiO_x and cross-point structure is used to implement physical RC under hybrid delayed feedback and parallel devices architecture. First of all, the device is characterised and modelled. Secondly, the architecture of memristor-based RC will be introduced along with its simulator in Python 3.6. Thirdly, the network

performance with different device and architecture setups will be discussed. Finally, such a system will be used in human activity recognition tasks to demonstrate its effectiveness.

3.2 Device characterisation

The TiO_x-based volatile memristor is fabricated by vertically staking $Ti/TiO_x/Pd$ (110 nm / 80 nm / 50 nm), as shown in Fig. 3.1(a). (The fabrication is done by the LEMON (Laboratory of Emerging MemOry and Novel computing) group of Tsinghua University. The fabrication process is not covered in this thesis.) Its dynamic behaviours over time is given by the Schottky barrier at the Pb/TiO_x interface [145]. Next, the measurement was carried out by a semiconductor parameter analyzer (Agilent 1500) and a probe station at room temperature. The Pd is the positive electrode. First of all, the volatile memristor's I-V curve is characterised by sweeping the input voltage in the range of -2V to 4V (Fig. 3.1(b)). A hysteresis curve with strong nonlinearity was observed. Then, the same input was injected 20 times, and it can be seen that the resulting curve is highly consistent, which indicates excellent repeatability and stability. Compared with the previous work based on the stake of $Ti/TiO_x/TaO_y/Pt$ (50 nm/16 nm/30 nm/50 nm) [63], this device is more stable, easy to fabricate and exhibits similar behaviours.

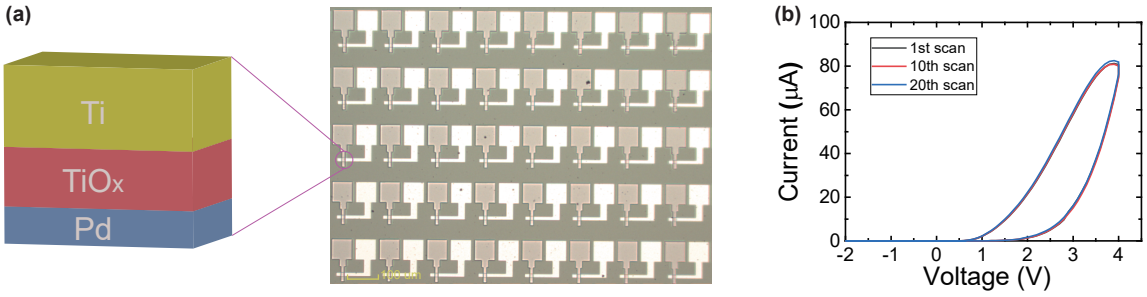


Fig. 3.1 Volatile memristor array and characterisation. (a) The fabricated volatile memristor array with the stake of $Ti/TiO_x/Pd$. (b) The I-V curve was measured by sweeping the voltage in the range of -2V to 4V, which was repeated by 20 times.

The next experiment further studies the dynamical behaviours under continuous read and write stimulation. The periodical input signal is programmed as a write pulse with an amplitude of 3V and width of 1ms, followed by 200 read pulse with an amplitude of 2V and width of $20\mu s$. The input and resulting output are shown in Fig. 3.2. The write pulse with a large voltage can gradually set the memristor to a higher conductance state. During the continuous read pulse, a decay can be observed after the write pulse. The conductance state of the volatile memristor depends on both the current input and historical input and state. These results demonstrate the memristive properties of the volatile memristor. Overall, the TiO_x-based memristor exhibits strong nonlinearity and short-term memory, which is well-suited to be used as the processing core of physical RC.

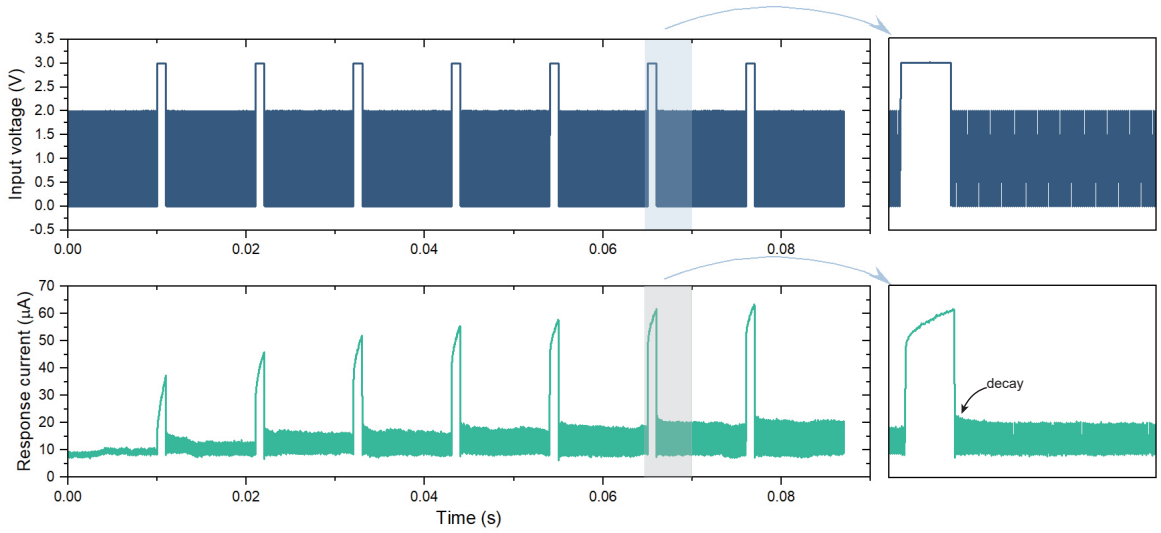


Fig. 3.2 Response of volatile memristor under continuous pulse input. The input is a periodical stimulation including a write pulse (3V, 1ms) and 200 read pulses (2v, $20\mu s$). The responding current signals show a dependency between the current output and historical output.

Furthermore, the I-V measurement was repeated 100 times to collect sufficient data for fitting the parameters of the discrete model of the volatile memristor. The equation of the discrete model is:

$$I_M = KGV^3 \tag{3.1}$$

$$G = G_0 + r(G' - G_0) + \frac{\alpha|V|}{\alpha|V|+1}(G_{th} - G') \tag{3.2}$$

$$V \geq 0 \begin{cases} K = K_p \\ \alpha = \alpha_p \\ r = r_p \\ G_{th} = 1 \end{cases} \qquad V < 0 \begin{cases} K = K_n \\ \alpha = \alpha_n \\ r = r_n \\ G_{th} = 0 \end{cases} \tag{3.3}$$

where I_M is the output current of the volatile memristor, V is the input voltage, K, α, r and G_{th} are the parameters varied with the sign of V (Eq. 3.3), G and G' denote the current conductance and the conductance at previous time step respectively. $G_0, r_p, r_n, \alpha_p, \alpha_n, K_p$ and K_n are the parameters to approximate the experimental data. This discrete model is an improved version of the previous work[63]. The measurement and simulation results are shown in Fig. 3.3, and the parameters of the discrete model for generating the simulation curve are provided in Table 3.1. As can be seen from the figure, the simulation model can capture the major characteristics of volatile memristors. This simulation model will be used as the processing core in the hybrid delayed feedback and parallel reservoir model.

Table 3.1 Parameters for the discrete model of volatile memristor

Parameter	Value	Parameter	Value
G_0	0.1	r_n	1.005
r_p	1.025	α_n	0.05
α_p	0.007	K_n	0.05
K_p	2.5		

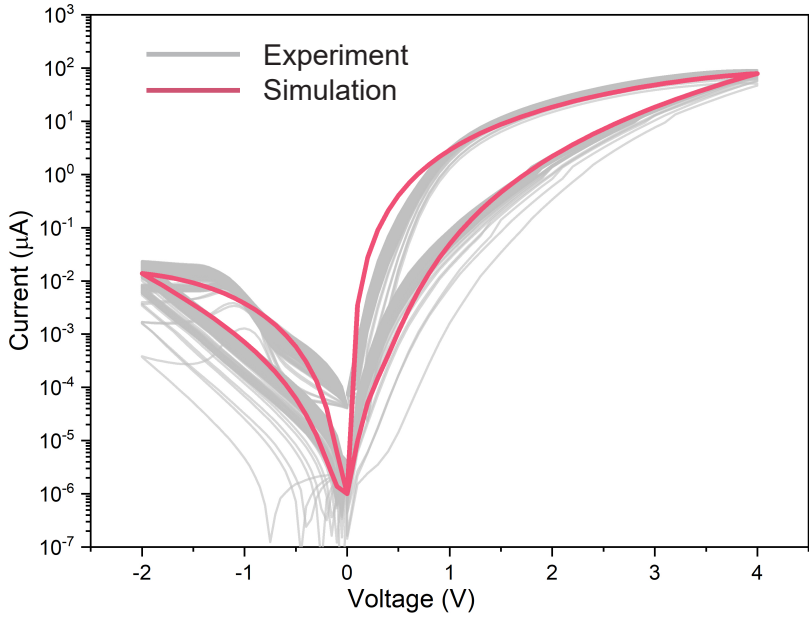


Fig. 3.3 I-V measurement and simulation results. The grey lines are the 100 repetitions of the I-V sweep in the range of -2 V to 4 V. The measurement results are used to fit the parameters of the simulation model. The red line is the fitting result.

3.3 Parallel memristors reservoir with mask

Given the rich dynamics offered by volatile memristor, an architecture should be properly designed to make full use of the dynamics for computing. In this section, a hybrid delayed feedback and parallel reservoir architecture are introduced. As illustrated in Fig. 3.4, multiple volatile memristors that act as processing cores receive the masked signal from a common input signal. The preprocessing procedure is similar to the mask process discussed in Section 2. Note that different mask matrices should be used for different volatile memristors in order to increase the state richness. The effect of using a common mask matrix for all volatile memristors will be discussed in the waveform classification task. Because of the short-term memory, the virtual nodes are nonlinearly coupled to each other, resulting in the temporal high-dimensional mapping of the input signal. The devices' responding output to every masked signal point should be collected as virtual node state $s(n)$ at n^{th} step. The node states should be put into the state matrix in a fixed order to multiple with the $\mathbf{W_{out}}$ value.

For example, if the first element in the state matrix is the response of the first masked point of volatile memristor 1, then the response of the first masked point of volatile memristor 1 should always be the first element for every masked period in both training and testing. The output layer is consistent with the routine of RC: in the training phase, the state matrix $s(n)$ is used to calculate the output weights $\mathbf{W_{out}}$ by using Ridge regression towards the target output $y(n)$; in the testing phase, the state matrix $s(n)$ is multiplied with $\mathbf{W_{out}}$ to obtain the predicted output $y'(n)$.

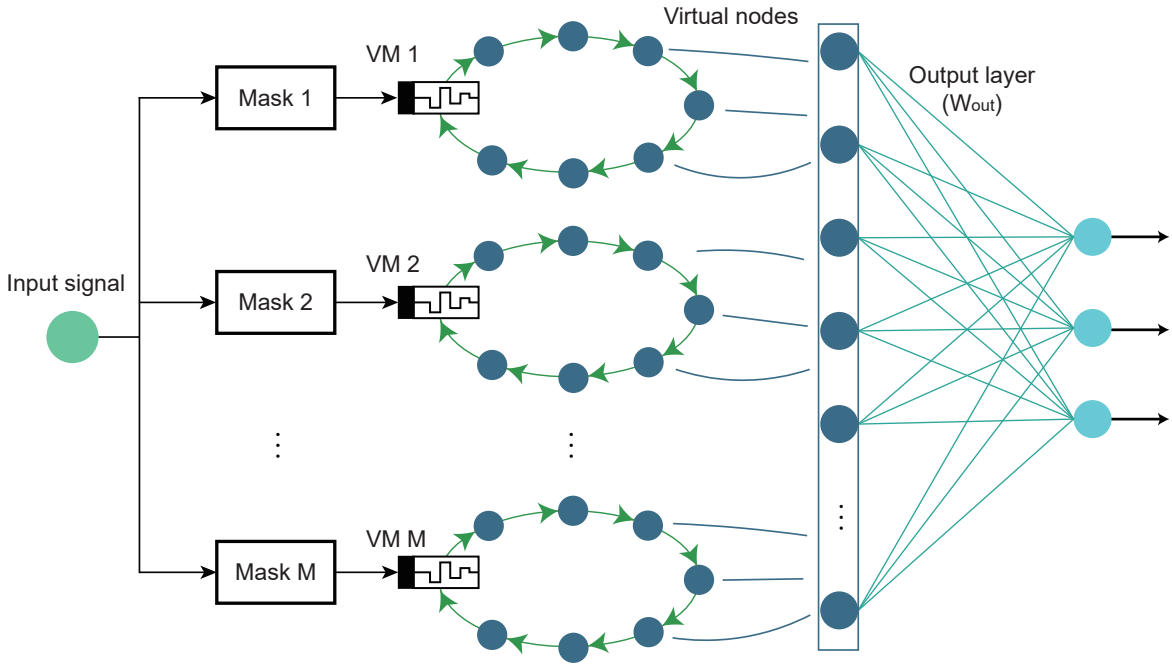


Fig. 3.4 Parallel reservoir computing architecture based on volatile memristor.

3.4 Waveform classification

3.4.1 Method

Waveform classification is a relatively simple task to evaluate the time-series processing performance. In this task, the input signal $u(n)$ consists of the randomly distributed sinusoid

and square wave with the same amplitude and frequency. The target output $y(n)$ is a binary sequence of 0 and 1 representing the waveform types of sinusoid and square wave, respectively. In this task, a total of 4000 data points were generated as the input signal, half of which were used to train the $\mathbf{W_{out}}$ and the rest for testing the classification results. The input values were mapped to the range of [-2V, 4V] as the characterisation of the volatile memristor suggested. According to the empirical results [63], the mask length cannot be high. Thus it is set to 5 throughout the test. In order to increase the task difficulty, only the volatile memristors are used to receive the masked input in the absence of delayed feedback. Also, a Gaussian white noise with an amplitude of 10^{-5} was added to the normalized conductance variable G on every time step. The signal flows and system diagram are shown in Fig. 3.5. The number of volatile memristors (M) is a variable that would be evaluated. For each volatile memristor, the masked signal can bring out temporal dynamics in its current response. For multiple volatile memristors, their different mask matrix results in distinguishable outputs that will be collected in the virtual nodes matrix as a reservoir state. Using the training and testing procedures introduced above, the predicted result $y'(n)$ can be obtained. Finally, the consistency between the ground truth $y(n)$ and predicted output $y'(n)$ will be quantified by normalized root mean square error (NRMSE). The definition of NRMSE is:

$$NRMSE = \sqrt{\frac{1}{m}\frac{\sum_{n=1}^{m}(y'(n)-y(n))^2}{\sigma^2(y(n))}} \tag{3.4}$$

where m is the total data length in the $y'(n)$ and $y(n)$, σ^2 is the output variance.

3.4.2 Results

The waveform classification results for the number of volatile memristors M = 1, 5 and 10 are plotted in Fig. 3.6. It can be seen from the first figure that a single volatile memristor is hard to distinguish the different waveforms and the NRMSE is 0.67±0.15. For M = 5, the model

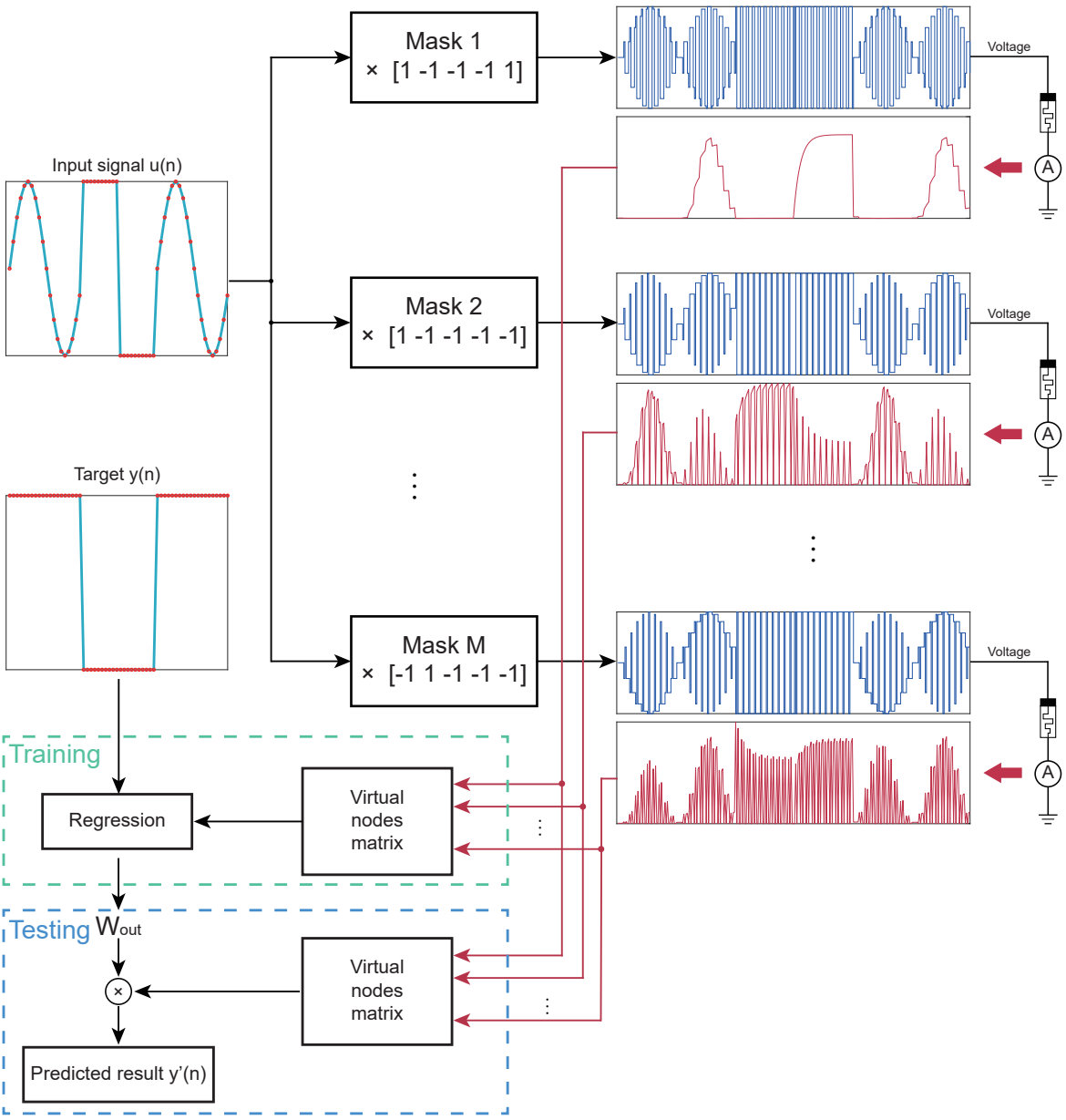


Fig. 3.5 Signal flows in the waveform classification task.

successfully captures the category of the input waveform with NRMSE = 0.28±0.05. When M is increased to 10, the NRMSE and standard deviation are further reduced to 0.15 and 0.02. Finally, the NRMSE over the different numbers of the volatile memristors is plotted in Fig. 3.7. The NRMSE and its standard deviation decreased as the number of volatile memristors increased. However, if every volatile memristor receives the masked signal with a common mask matrix, the performance cannot be improved since every volatile memristor produces similar outputs.

These results reveal that 1) the volatile memristor-based network model can successfully perform waveform classification task, which primarily proves its capabilities in temporal signal processing; 2) increasing the number of volatile memristors can improve the task performance when different mask matrices are applied to different devices. In the actual measurement, a certain level of D2D variation could also enhance the performance as the M increased[64], even though using a common mask matrix. However, using different masks is still an effective method to increase state richness.

3.5 Hénon map chaotic series prediction

3.5.1 Method

Introduced by Michel Hénon, the Hénon map is a simplified version of the Lorenz model and also a classical discrete chaotic time series, which can be used to test the prediction performance of the proposed model [63, 146]. A typical setup of the Hénon map time series is defined as follows:

$$\begin{aligned} x(n+1) &= y(n) - 1.4x(n)^2 \\ y(n+1) &= 0.3x(n) + w(n) \end{aligned} \tag{3.5}$$

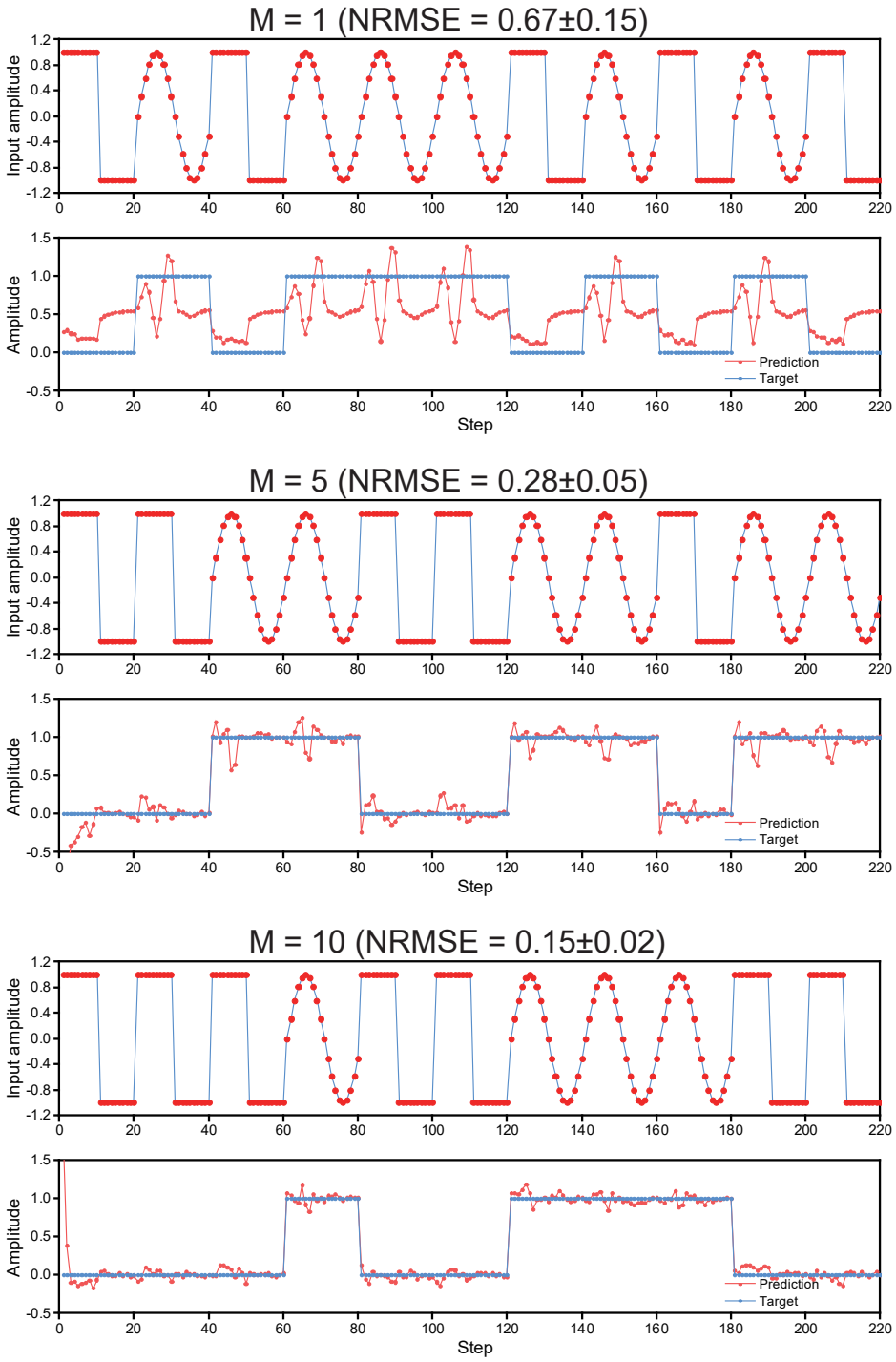


Fig. 3.6 Result of waveform classification for the number of volatile memristors M = 1, 5 and 10 respectively. Each NRMSE result was repeated 30 times to obtain the mean value and standard deviation.

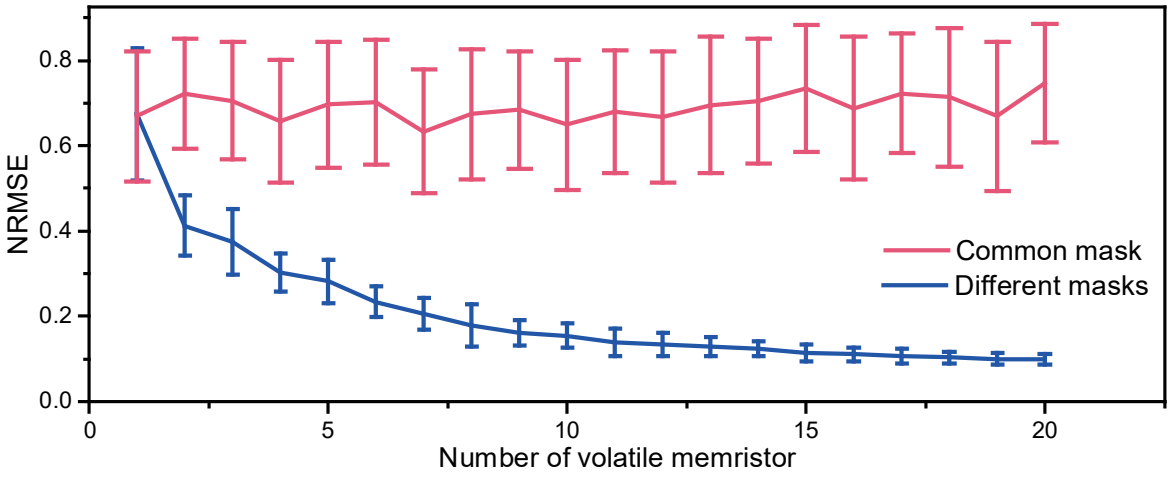


Fig. 3.7 Results of waveform classification over a different number of volatile memristors. The blue line indicates the cases that every volatile memristor uses different mask matrices, while the red line is the result of using a common mask matrix for all volatile memristors. Each NRMSE result was repeated 30 times to obtain the mean value and standard deviation.

where $x(n)$ and $y(n)$ denote the coordinates of the discrete data point on a 2D panel at n^{th} step, $w(n)$ is the Gaussian white noise randomly generated at every time step with a mean value of 0 and standard deviation of 0.05. The equations describe that given the value of $y(n)$ and $x(n)$, the iterative values of $y(n+1)$ and $x(n+1)$ can be calculated together with a Gaussian white noise value. An example of the Hénon map time series generated by Eq. 3.5 is shown in Fig. 3.8, where a chaotic attractor can be found.

In this task, a total of 2000 steps of Hénon map data (1000 for training and 1000 for testing) were generated. In the model, the input is $x(n)$ and the target output is $x(n+1)$, while another axis $y(n)$ can be easily calculated by Eq. 3.5. The signal flows and the resulting output is similar to what has been presented in Fig. 3.5. Also, the number of volatile memristor M is evaluated in this task.

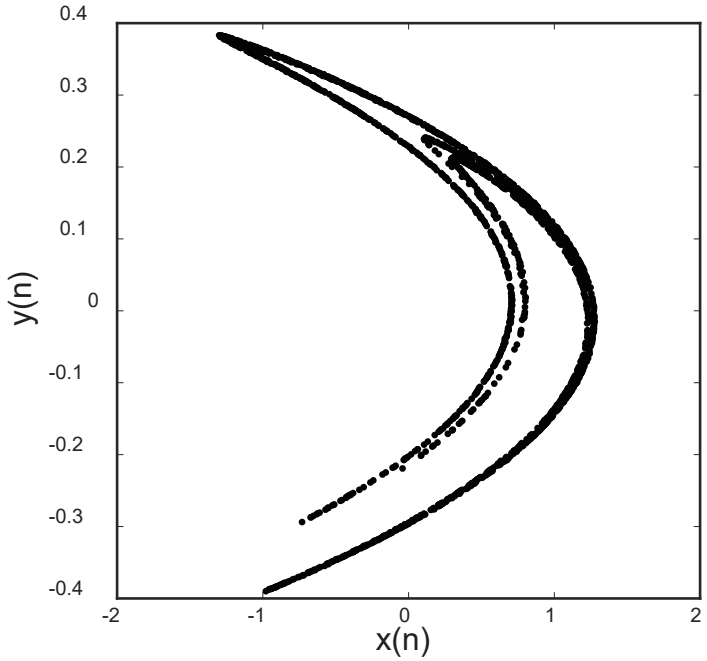


Fig. 3.8 An example of the Hénon map time series generated by Eq. 3.5.

3.5.2 Result

The results indicate that the proposed model can successfully predict the Hénon map time series with relatively large NRMSE (0.17±0.02) using only 5 volatile memristors, as shown in Fig. 3.9. When increasing the number of volatile memristors to 100, the prediction result can be significantly improved to NRMSE = 0.06±0.01. The 2D map can better illustrate the distribution of outputs in comparison with the target output. Furthermore, Fig. 3.10 indicates that the NRMSE converge at around 0.06 as the number of volatile memristors increases.

Generally, this task is more demanding than the waveform classification task in terms of the signal complexity and chaotic properties, therefore resulting in a higher requirement for the computing and approximation abilities of the model. In the waveform classification task, the optimal performance can be obtained using less than 20 volatile memristors, whereas the Hénon map prediction needs more than 40 volatile memristors. The above two benchmark tasks primarily test the computing performances of temporal signal processing.

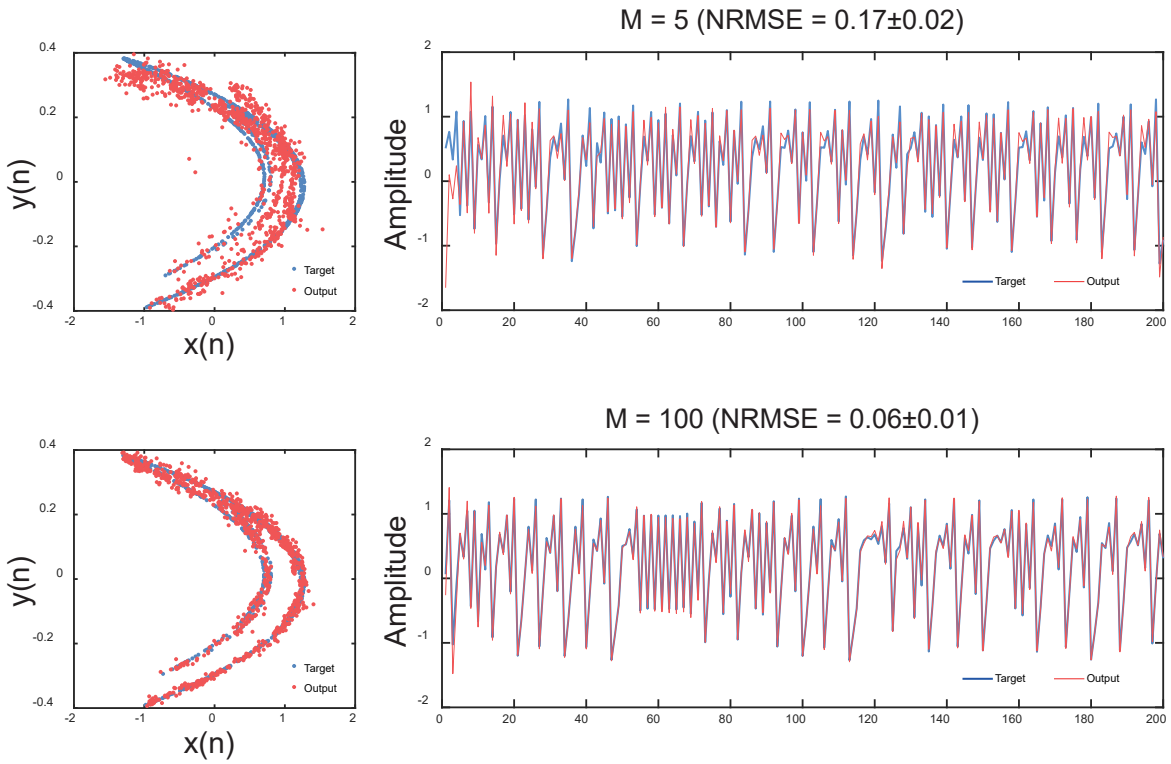


Fig. 3.9 Result of the Hénon map chaotic series prediction for the number of volatile memristors M = 5 and 100 respectively. Each NRMSE result was repeated 30 times to obtain the mean value and standard deviation.

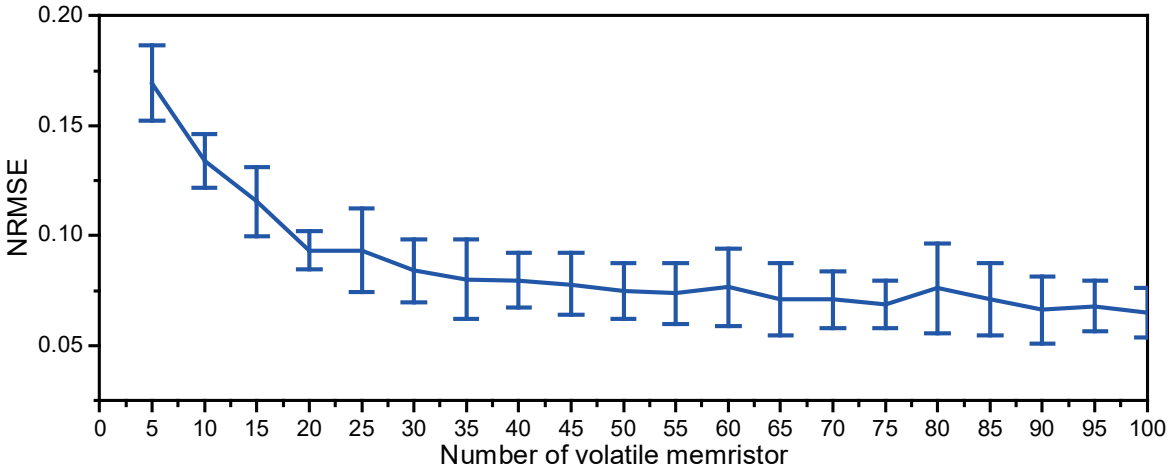


Fig. 3.10 Results of Hénon map chaotic series prediction over a different number of volatile memristors. Each NRMSE result was repeated 30 times to obtain the mean value and standard deviation.

3.6 Human activity recognition

In this task, the proposed volatile memristor-based RC is applied to human activity recognition which is a more practical task. An online available dataset collected by Wireless Sensor Data Mining Lab is used to test the system[147]. Overall, the volatile memristor-based RC model receives the raw 3-axis acceleration signals while the output is a classification result, as illustrated in Fig. 3.11.

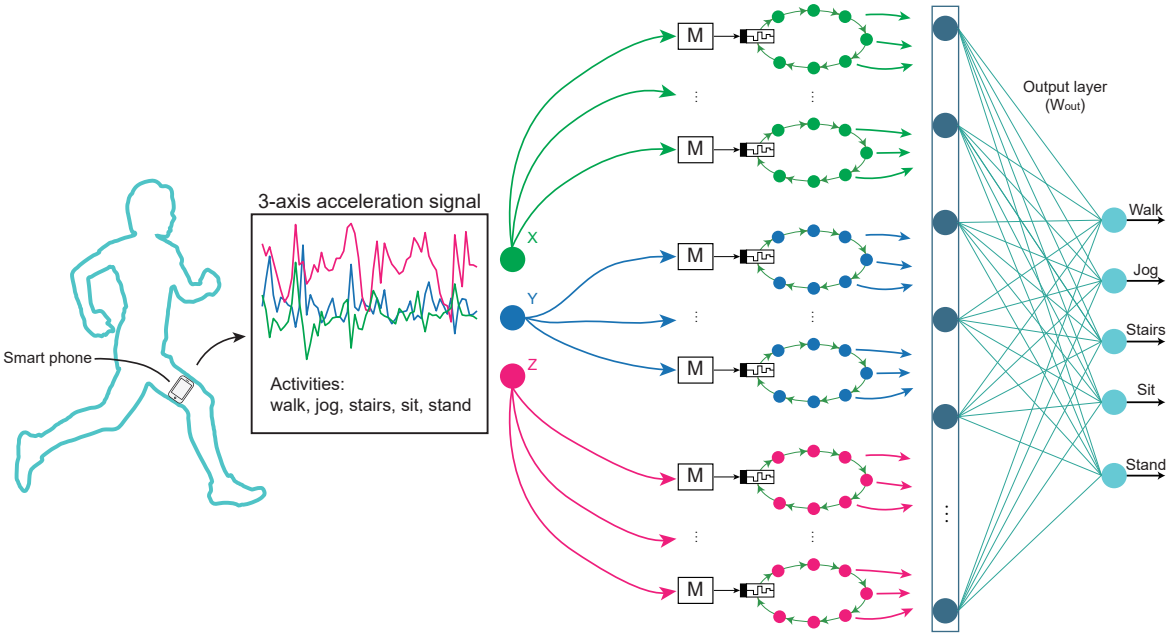


Fig. 3.11 System overview of the proposed volatile memristor-based RC for human activity recognition, where M denotes mask process. The signal source is an open dataset containing the 3-axis acceleration data over time for human activities including walking, jogging, standing, sitting and ascending/descending stairs, which are the typical temporal signals generated by human activities. The masked signals are sent to volatile memristor-based reservoir computers with delayed feedback. The reservoir output can be collected to calculate the final classification output via a linear readout layer.

3.6.1 Datasets

The 3-axis acceleration data of human activity used in this task is an open dataset collected by Wireless Sensor Data Mining Lab is used to test the system, namely the WISDM dataset [147].

In this dataset, 36 participants were required to carry a mobile phone with an Android system in their front pants leg pocket. Then, they were asked to walk, jog, ascend stairs, descend stairs, sit and stand for specific periods of time, which are most people's daily activities. The accelerometer embedded in the mobile phone can record the 3-dimensional acceleration data stream with a 20Hz sampling frequency. Meanwhile, the data was properly labelled in the dataset according to the activity that has been performed. Previously, this dataset has been used to test various human activity recognition using software-based algorithms and yielded over 80% accuracy, such as multilayer perceptron, logistic regression, decision tree [147] and Convolutional Neural Network (CNN) [148].

In this task, the WISDM dataset is tested by the proposed hardware-based reservoir model. First of all, the entire dataset is segmented using a sliding window with a fixed length of 180 time steps and total of 13000 segments were used, 70% (9100 segments) and 30% (3900) of which were used in training and testing, respectively. Here, the stairs data is in combined mode (ascending stairs and descending stairs are combined as stairs).

3.6.2 System framework

Unlike the software-based methods that usually employ feature extraction, the input in the proposed method is the raw 3-axis acceleration signal collected from the different human activities. The proposed reservoir system receives the signal for each axis using 50 volatile memristors. Different from the above two tasks, delayed feedback is added to the reservoir loop to enhance the performance. Note that different mask metrics should be used for different volatile memristors as discussed in previous sections. Since the mask length is 5, the total state size is $50 \times 5 \times 180 \times 3 = 135,000$ which takes up large memory during simulation. In order to speed up the simulation, only the state vector of 5 points was collected (collect once for every 36 points) instead of all 180 points, since the reduced state vector

could be representative of the rest to some extent. Thus, the actual state vector size is $50 \times 5 \times 5 \times 3 = 3750$. For training, ridge regression was employed to calculate $\mathbf{W_{out}}$ by using the state matrix (9100×3750) and the target matrix (9100×5). Finally, the output in the testing set can be calculated using $\mathbf{W_{out}}$ and the state matrix. Examples of input and output signals can be found in Fig. 3.12. The output amplitude represents the possibility for each activity. The activity corresponding to the highest value is considered the predicted result.

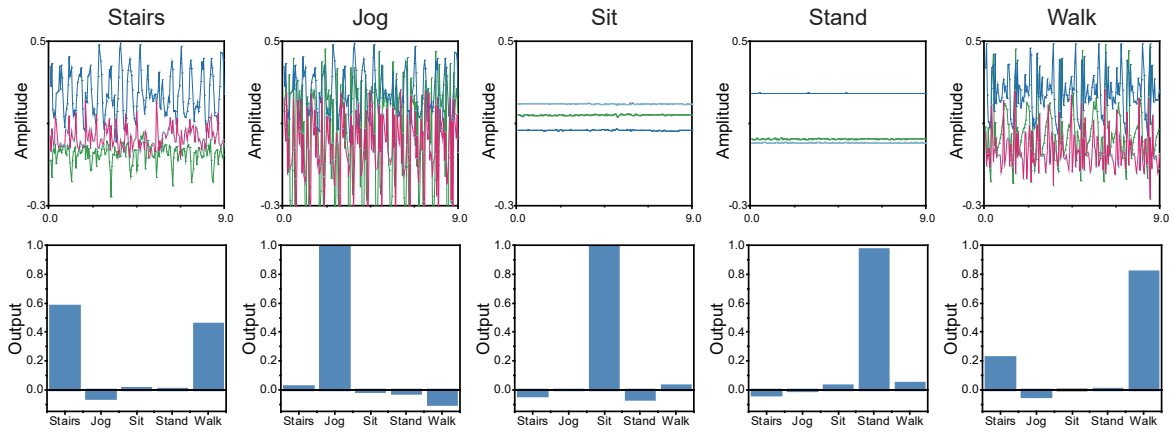


Fig. 3.12 Examples of input (top) and output (bottom) for the volatile memristor-based RC in human activity recognition tasks.

3.6.3 Optimization and results

Unbalanced dataset

The WISDM dataset is a highly unbalanced dataset. The training set contains 1868 stairs data, 2865 Jogging data, 481 Sitting data, 388 Standing data and 3498 walking data, where walking data significantly outnumbers other classes. In the multiclass classification problem, different amounts of the classes would result in the separation plane shifting closer to the classes with more samples, which is also known oversampling problem. The performance of the model trained by original dataset is shown in Fig. 3.13(a). Due to the highly unbalanced

dataset, the ‘Stairs’, whose signal is resemble walking data, is easily misclassified into ’Walk’ and yielded only 40.5% accuracy, while the overall accuracy is 81.4%. Such low accuracy disables the classification of ’Stairs’. To solve this problem, Synthetic Minority Over-sampling TEchnique (SMOTE) is used to create more state vectors for the minority classes. In the high-dimensional state space, the SMOTE algorithm generates more samples between the original samples, so that it creates additional state vectors without overlapping with the original real state vectors. Using the SMOTE, the number of state vectors of the classes except ’Walk’ were increased to be equal to ’Walk’, thus minimizing the effect of the unbalanced dataset. Fig. 3.13(b) is the result using the balanced dataset created by SMOTE for training. Although the overall accuracy (80.4%) slightly declined, the result of ’Stairs’ increased to an acceptable level and therefore enabling the all 5-class classification.

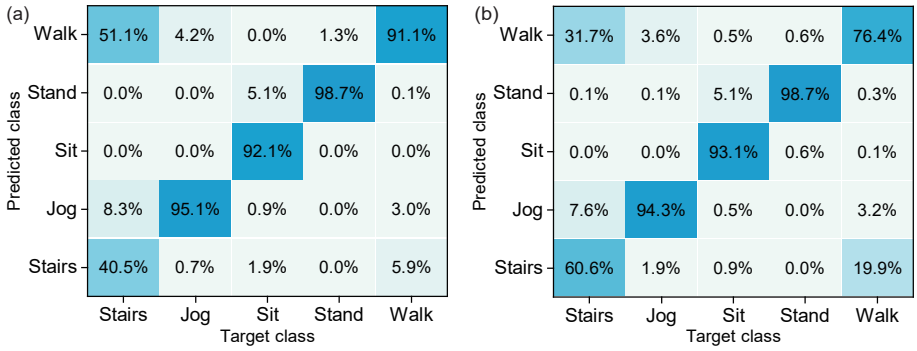


Fig. 3.13 Results of human activity recognition using (a) the original unbalanced dataset and (b) the dataset generated by SMOTE for training. Their overall accuracies are 81.4% and 80.4% respectively.

Delayed feedback

In the previous two tasks, the volatile memristors directly receive the masked input. This approach should be in line with the architecture of parallel devices. However, using only the parallel devices results in insufficient network capability to obtain an acceptable accuracy in the human activity recognition task. As discussed in Section 1, the different architectures are not completely exclusive. A hybrid reservoir system combining different architectures could

exhibit excellent performance. In the existing architectures, delay-based architecture can provide higher MC because the external memory unit is used to store the delayed feedback signal. In the volatile memristor-based reservoir system, adding delayed feedback can enhance the classification result since the MC is enhanced by the external feedback unit. The final result after adding the delayed feedback is shown in Fig 3.14.

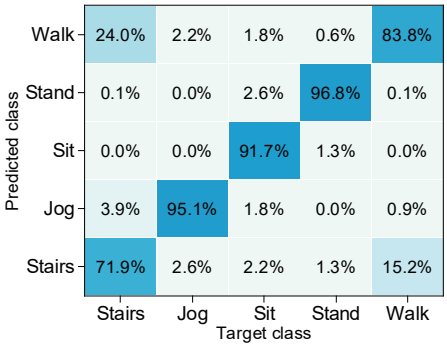


Fig. 3.14 Results of human activity recognition after the optimization using SMOTE and delayed feedback. The overall accuracy is 85.8%.

3.7 Conclusions

In this chapter, the volatile memristor ($Ti/TiO_x/Pd$) was measured and then modelled by the discrete behavioural model. The discrete model can approximate the I-V characteristics of the volatile memristor after fine-tuning the parameters. Next, this model was used to assemble a volatile memristor-based RC model with the time-multiplexing operation and the architecture of the parallel device that has been introduced in previous chapters. To evaluate the volatile memristor-based reservoir, firstly, it has been applied to a waveform classification task, in which the model is trained to distinguish between sinusoid and square waves. Here, the system performance over important factors including the number of volatile memristors and mask matrix is tested, The results proved that the proposed system can successfully classify the two types of the waveform when using more than 5 volatile memristors with different mask matrices. Secondly, the Hénon map chaotic series can be used to test the system's

performance in terms of chaotic system approximation, which is a more demanding task compared with the waveform classification. Similarly, the proposed model can successfully predict the Hénon map signal with acceptable error and the error can be reduced by increasing the number of the volatile memristors. Thirdly, a 5-class human activity recognition task using the WISDM database is tested. After optimising the system by using the techniques like SMOTE and delayed feedback, the accuracy is 85.8%.

In summary, this chapter analyses the volatile memristor-based reservoir computer and its performance in the benchmark tasks. The simulation results prove the feasibility of using the short-term memory of a volatile memristor to construct a reservoir computer. Its performance is not ideal but acceptable considering the limited network size and the difficulty of hardware implementation. Also, analogue computing is not as flexible as the digital computing in which various preprocessing and feature extraction algorithms are available, resulting in a lower accuracy in the human activity recognition benchmarks.

Overall, the parallel devices architecture used in this work successfully explores the intrinsic physical properties as computing resource. parallel devices provide limited computing capabilities, which can be enhanced by adding delayed feedback to form a hybrid architecture. However, these techniques usually require additional cost in the actual hardware implementation. Thus, an effective architecture of RC with minimum hardware complexity is still of high interest in the field of neuromorphic computing, and remains discussion in the next chapter.

Chapter 4

Rotating Neurons Reservoir: Theory and Simulation

Hardware implementations of RC demand an elegant architecture with minimum system complexity. However, an end-to-end reservoir architecture has yet to be developed. This chapter presents an architectural innovation for implementing cyclic reservoirs using rotating elements integrated with signal-driven dynamic neurons, whose equivalence to standard cyclic reservoir algorithm is mathematically proven. Simulations show that the rotating neuron reservoir achieves record-low errors in a nonlinear system approximation benchmark.

4.1 Motivation

As discussed in Chapter 1, the existing physical RC studies mainly adopted three kinds of architecture: DRC, in-materia and parallel devices. The main drawbacks associated with the use of delayed feedback and time-multiplexing are as follows:

(i) Delayed feedback is costly for hardware implementations using conventional CMOS technology or optical approaches, which require additional digital components [56, 123], such as ADCs and random-access memory, or bulky optical fibers [71–73, 76], respectively.

(ii) In the absence of a delayed feedback line, a RC system cannot simultaneously maintain an appropriate MC or satisfactory state richness. For example, previous research revealed that shortening the step size in time multiplexing could improve the MC but at the cost of reducing the state richness, or vice versa [63]. In this case, the architecture is similar to parallel devices architecture.

(iii) The serial operations in time multiplexing increase system complexity and latency for both input and readout, whereas parallel computing, which enhances the throughput, is more desirable in neuromorphic computing [22].

The in-materia and parallel devices architectures can compute in highly parallel and analogue manner. However, they suffer from similar weaknesses:

(i) The outputs are not explainable. The signal experience unclear transformation in materials and devices. This problem is particularly serious in the in-materia approach in which only the state outputs are measurable.

(ii) The relatively lower MC in the network. The MC in both approaches are provided by the memristive property of the devices or materials. To process temporal signal, usually external memory units are needed to enhance the MC.

These obstacles hinder further reductions in power and size when the cost for an entire reservoir computer, from the signal input to the computing output, is considered; thus, a knowledge gap associated with massive deployment in practical applications remains. There is an urgent need to develop a new architecture involving hardware-based reservoir computers

of miniature size with low power consumption and high capability for large-scale integration [2, 16].

In this work, a rotating neuron-based architecture is introduced for physically performing RC in a more intuitive way, namely rotating neurons reservoir, whose rotation behavior matches with the neurons update in a CR, as rigorously proven through mathematical derivations. Compared with the existing implementations in RC [64, 72, 86, 90, 123], the RNR is hardware-friendly, resource-efficient, fully parallel and equivalent to standard CR. To verify the feasibility and potential of the RNR, an electrical RNR (eRNR) design based on CMOS circuits is introduced together with a simulator. Furthermore, a prototype eRNR composed of eight parallel reservoir circuits is built to perform analogue near-sensor computing, and real-time Mackey-Glass time series prediction and real-time handwriting recognition are successfully performed in hardware experiments. To realize an all-analogue RC system, the eRNR is further integrated with an analogue memristor array that implements the fully connected output layer. Through the proposed noise-aware training method, the conductance variation of the memristor array is accommodated, and a high classification accuracy of 94.0% is achieved for a handwritten vowel recognition task. Finally, a CMOS circuit simulation based on standard 65nm technology indicated that the eRNR system is projected to consume as little as 32.7 μW of system power in the handwriting recognition task; this total would be more than three orders of magnitude lower than that achieved by literature-reported reservoir systems. These results highlight the tremendous potential of the proposed RNR, offering a promising paradigm for resource-efficient reservoir computers.

4.2 Physical CR with rotating neurons

For a typical RC with an m-dimensional input, an n-dimensional output and N neurons (Fig. 4.1(a)), the input coefficients $\mathbf{W_{in}}$ and reservoir weights $\mathbf{W_{res}}$ are randomly generated

[23]. The complex dynamics stemming from the massive and random connections in the reservoir layer aid in nonlinearly mapping the m-dimensional input to the N-dimensional feature space where different input classes can be linearly separated. For n output classes, only the output weights $\mathbf{W_{out}}$ need to be trained by using linear regression, which is relatively efficient compared to other recurrent neural network methods [23, 24, 31]. Note that linear ridge regression is used for training throughout this work. The neuron dynamics in the reservoir layer play an important role in signal mapping according to the Eq. 1.1. In RC, the reservoir layer $\mathbf{W_{res}}$ can be designed in a deterministic manner rather than being based on random connections [28]. In this case, the $\mathbf{W_{res}}$ becomes a shifted identity matrix $\mathbf{R}$:

$$\mathbf{W}_{\text{res}} = \mathbf{R} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 1 \\ 1 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 1 & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \ddots & 1 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 \end{bmatrix} \tag{4.1}$$

As a result, $\mathbf{W_{res}}$ is significantly simplified, and the network topology becomes CR, as shown in Fig. 4.1(b). Previous research concluded that CR could achieve comparable results to those of conventional RC [28]. Then, the matrix $\mathbf{R}$ corresponds to one-time shifting in a ring structure, and $\mathbf{R^n}$ indicates a n-time cyclic shift analogous to physically rotating an object. As illustrated in Fig. 4.1(d), it is assumed that (i) the post- and pre-neuron rotors are described by $\mathbf{R}$ and its transpose matrix $\mathbf{R^T}$, respectively; (ii) $\mathbf{q}(\mathbf{n})$ is the dynamic neuron output at the n^{th} step; and (iii) $\mathbf{s_r}(\mathbf{n})$ is the state matrix of the RNR at the n^{th} step measured at the end of each rotor's channel (before the output weights). Considering the rotation of the

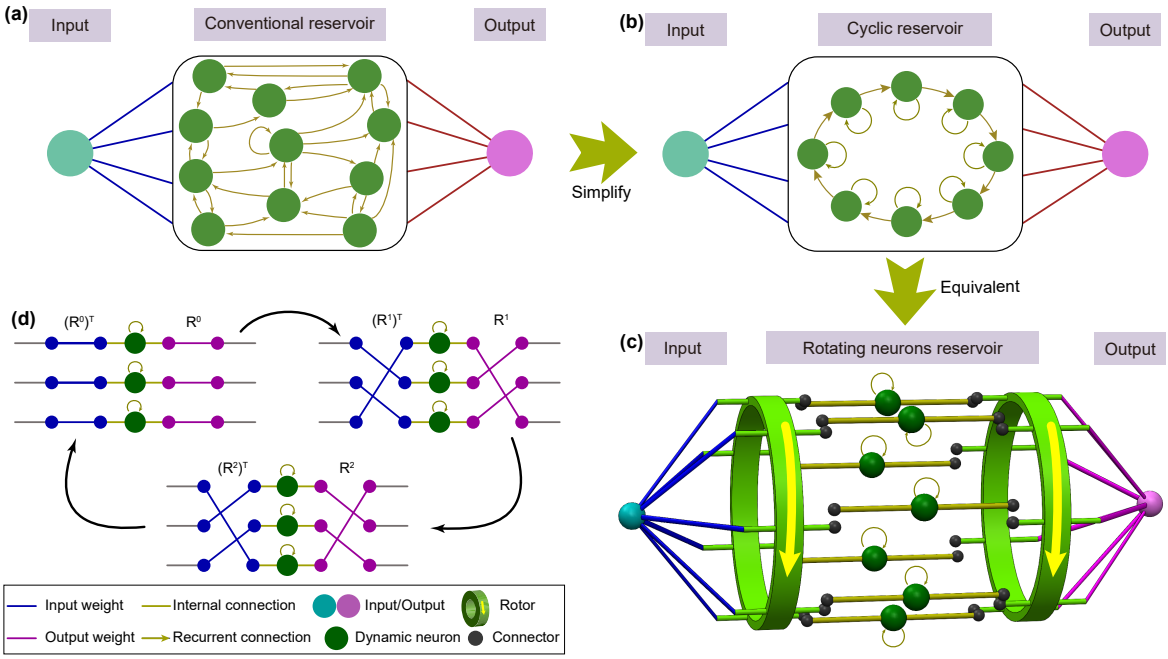


Fig. 4.1 RC architectures. (a) A conventional RC architecture with random connections. (b) A simplified version of a reservoir, also known as a cyclic reservoir. The randomly connected neurons are replaced with a ring structure. (c) Illustration of the working principle of the proposed rotating neuron reservoir (RNR) that can be physically implemented. The input weights are uniformly distributed in the range of [-1, 1], and a pre-neuron rotor sends the signal to different neuron channels at different time steps. After flowing through the dynamic neurons, the signal is sent to different state channels via another post-neuron rotor, and the final states are read out through a fully connected layer and used in training. (d) Sketch of the working principle for the case of three neurons, where **R** denotes the rotation matrix. The legend for all subfigures is provided at the bottom.

neuron output, the state $\mathbf{s_r(n)}$ updating formula can be written as:

$$\mathbf{R^{n-1}q(n)} = \mathbf{s_r(n)} \tag{4.2}$$

which indicates that, at the n^{th} step, the state matrix $\mathbf{s_r(n)}$ is obtained by rotating the neuron output $\mathbf{q(n)}$ for $(n-1)$ times. Furthermore, the output of dynamic neurons is determined based on both an input shift and the previous states:

$$\mathbf{q(n+1)} = f_r[a'(\mathbf{R^n})^T\mathbf{W_{in}u(n+1)} + b'\mathbf{q(n)}] \tag{4.3}$$

where b' denotes the decay factor resulting from the dynamic property of the neuron, a' is the scaling factor for the input, and $f_r(x)$ is the nonlinear transform implemented by the dynamic neurons. Eq. 4.3 describes the signal flow through the neurons. Given an input $u(n)$, it is first multiplied by the input weights $\mathbf{W_{in}}$. After n reverse rotations of the input connections, the signal is fed into the dynamic nonlinear neurons, which output $q(n+1)$. If both sides of Eq. 4.2 are multiplied by $\mathbf{R^n}$, thus:

$$\mathbf{R^n q(n+1)} = \mathbf{R^n}f_r[a'(\mathbf{R^n})^\mathbf{T}\mathbf{W_{in}u(n+1)} + d\mathbf{q(n)}] \tag{4.4}$$

Using Eq. 4.2, Eq. 4.4 can be simplified as:

$$\mathbf{s_r(n+1)} = f_r[a'\mathbf{W_{in}u(n+1)} + b'\mathbf{Rs_r(n)}] \tag{4.5}$$

Here, the excellent consistency between Eq. 1.1 and Eq. 4.5 reveals that the proposed physical RNR architecture (Fig. 4.1(c)) is equivalent to a software CR. Thus, a rotating object with dynamic neurons can act as a reservoir computer without using extra control units,

ADC or memory, which remarkably reduces the system complexity and power consumption compared with those in conventional hardware implementation.

The rotation couples the physical RNR and software CR. The mathematical derivation of the RNR proves that a rotating neuron array is equivalent to a CR model (Fig. 4.1(b)). Fig. 4.1(c) illustrates the operation principle of the rotation-based reservoir: if the neuron array is fixed, the pre- and post-neuron rotors rotate in the same direction to periodically shift the connections, which is equivalent to rotating the neuron while fixing the pre- and post-neuron rotors. Fig. 4.1(d) shows an example of a 3-neuron RNR. The rotors shift the connections before and after the neurons. The channels on the right side output the analogue computing results equivalent to the neuron states in a CR model with the same input. It can be emphasized that the fundamental of RNR is widely applicable to various rotating components, not limited to CMOS implementations, that can be developed as a reservoir by embedding dynamic neurons.

4.3 Hardware architecture

the main challenge of implementing a hardware RNR is the construction of the physical rotors and dynamic neurons based on the above approaches. Fig. 4.2(a) illustrates a schematic of an N-neuron eRNR designed using CMOS circuits. The implementation of the input layer using binary weights is important because it allows the system to directly interface with analogue sensory signals. $\mathbf{W_{in}}$ is taken to be a matrix consisting of a randomly generated uniform distribution of -1 and 1 values, which have been proven to be effective as multilevel weights [74]. Assuming that the signal source is $u(t)$, for each neuron, the driving signal should be $a'u(t)$ or $-a'u(t)$ during one time step. $\mathbf{W_{in}}$ can be configured by changing the switches (S1 to SN). Note that the $\mathbf{W_{in}}$ should remain unchanged while the RNR is operating so that the switches can be replaced with fixed connections.

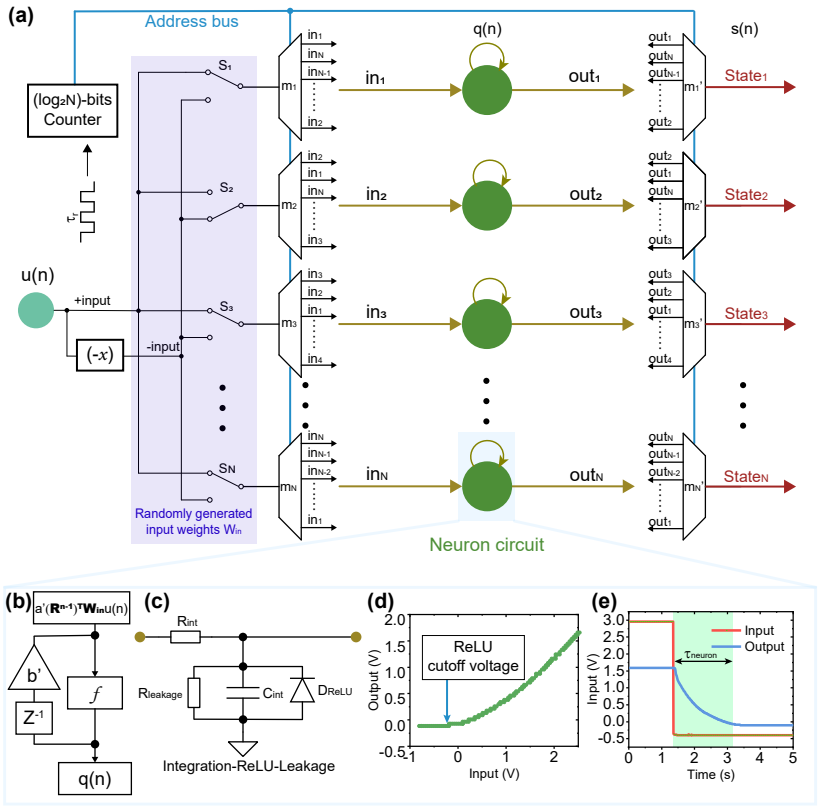


Fig. 4.2 Implementation of the eRNR. (a) Schematic of an N-neuron eRNR. Given an input $u(n)$, first, an operational amplifier generates another signal source $-u(n)$ or negative input. The switch array S_1 to S_N determines the input weights $\mathbf{W_{in}}$ by selecting a positive or negative source for each multiplexer. The multiplexers m_1 to m_N and m_1' to m_N' are involved in the electrical implementation of pre- and post-neuron rotors, respectively. The log_2N-bits counter outputs an address signal to sequentially activate the channels of each multiplexer at switch intervals τ_{rotor}. Based on the distinct sequence of neuron connections (in_1 to in_N for the input and out_1 to out_N for the output), the behavior of the multiplexer array is equivalent to that of a rotor cyclically shifting connections between neurons and input/output channels. The sequence for output channels is a mirror version of that of for input channels, which complies with the common-directional rotation principle in RNR theory. (b) General schematic of the dynamic properties required for a neuron in an RNR. When a neuron input $(\mathbf{R^{n-1}})^{\mathbf{T}}\mathbf{W_{in}}u(n)$ that has been processed by a pre-neuron rotor and input weights are provided, the neuron performs nonlinear transform f, integration (feedback line), and leakage (decay factor b') operations on the signal. $q(n)$ is the neuron output at the n^{th} step. (c) A dynamic neuron in the eRNR. C_{int} and R_{int} serve as integrators. The rectifying diode D_{ReLU} provides an activation function similar to a nonlinear ReLU function. Finally, high resistance $R_{leakage}$ is added to control the current leakage rate, that is, the decay factor b' in Eq. 4.5. (d-e). The nonlinear properties (d) and dynamic integration (e) of the neuron for $R_{int} = 10k\Omega$, $C_{int} = 1\mu F$, and $R_{leakage} = 100k\Omega$. D_{ReLU} is a germanium diode with a forward voltage of approximately 0.3 V.

Next, the pre-neuron rotor is implemented using N N-channel multiplexers composed of transmission gates. All multiplexers share a common address line from a $log_2 N$ (for N = 2, 4, 8, 16 ...) bit counter but different channel sequences for neuron connections, as illustrated in Fig. 4.2(a). A driving clock with a period of τ_{rotor} is used to sequentially increase the counter address from 0 to N-1 and then reset it to 0. This address is used to control the activated channels of all the multiplexers. Because the sequence of neuron connections is inconsistent, every multiplexer is connected to a different neuron during one τ_{rotor}. Such a configuration ensures that every input channel transmitting $a'u(t)$ or $-a'u(t)$ continues to poll every neuron during every rotation cycle $\tau_{rotor} \times N$, which corresponds to the transformation $a'(R^{n-1})^T W_{in} u(n)$, where $\mathbf{R^{n-1}}$ denotes (n-1)-time shifting. Upon receiving the neuron input $a'(R^{n-1})^T W_{in} u(n)$ and adding to its current value, the resulting neuron output $\mathbf{q}(\mathbf{n})$ is represented by the voltage level measured at the right side of the neuron circuit. The final step is to employ another post-neuron rotor at the output to convert $\mathbf{q}(\mathbf{n})$ to a state vector $\mathbf{s}(\mathbf{n})$. The post-neuron rotor performs an operation that is a mirror of that implemented by the input multiplexer array to obtain the forward rotation $\mathbf{R^n}$.

Multiple parallel RNRs can simultaneously connect to a common input signal but use different $\mathbf{W_{in}}$ configurations to increase the state richness. Fig. 4.3 illustrates a complete eRNR computing architecture that includes M parallel N-neuron eRNRs. The output weights are obtained through training and mapped in a memristor array to calculate the final results.

4.4 Design and modeling of dynamic neurons

In addition to the rotors, dynamic neurons are also crucial elements in nonlinear computing. Based on the fundamental RNR characteristics described in the last section, a neuron in the RNR should possess three important characteristics:provide a nonlinear activation function $f(x)$; support integration ability for the summation between the current input and previous

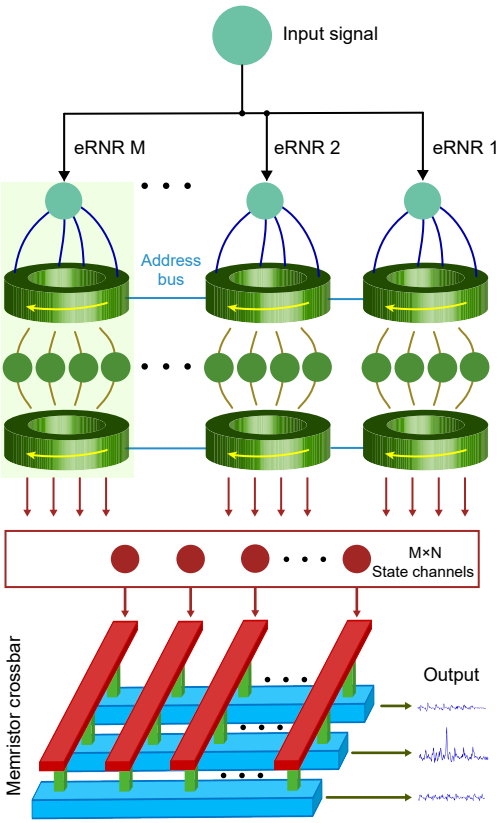


Fig. 4.3 Schematic of a complete eRNR system that includes M parallel N-neuron RNRs. The total size of the state matrix is $M \times N$. The voltage signal of each state channel is multiplied by the trained output weights stored in a memristor array to yield the final computing result.

state $q(n-1)$; and support leakage, as related to the decay factor b', to avoid saturation caused by the integration process.Any passive element that exhibits these three characteristics could essentially be used as a dynamic neuron in the RNR architecture by fine-tuning the time constants of neuron and rotors. A dynamic node working in a physical reservoir may suffer from device variation issues, which impact system performance. Previous studies have revealed that a certain degree of device variation may be beneficial to system performance by enhancing state richness [63, 64], but determining how to precisely control device variability warrants future explorations. In implementations using standard electronics (Fig. 4.2(c)), a ReLU-type nonlinear transform can be provided by a diode, and the resistor R_{int} and capacitor C_{int} can act as integrators. Leakage can be considered by a connecting the system to the ground via a large resistance $R_{leakage}$. In the simulation, this neuron can be modeled as follows:

$$\dot{V_o}(t) = \frac{1}{R_{\text{int}}\, C_{\text{int}}} V_i(t) - \frac{R_{\text{int}} + R_{\text{leakage}}}{R_{\text{int}}\, R_{\text{leakage}}\, C_{\text{int}}} V_o(t) + \frac{1}{C_{int}} I_s \left(e^{-\frac{V_o(t)}{V_T}} - 1 \right) \tag{4.6}$$

where $V_o(t)$ and $V_i(t)$ denote the input and output voltages, respectively. The saturation current I_s and thermal voltage V_T stem from the Shockley diode equation $I = I_s(e^{V_D/V_T} - 1)$. The typical values for germanium diodes $I_s = 25 \times 10^{-9} A$ and $V_T = 0.026V$ were used in the simulation. In the case of linear neurons, the last term $I_s(e^{V_D/V_T} - 1)/C_{int}$ should be removed from Eq. 4.6. In our simulation, Eq. 4.6 was solved in MATLAB/Simulink, as shown in Fig. 4.4. The discrete neuron output becomes $s(n) = V_o(n\tau_{rotor})$. The pre- and post-neuron rotors can be modeled by continuously shifting $W_{in}u(n)$ and the neuron output $q(n)$. Since $R_{leakage}$ is a large resistance, the time constant associated with this neuron is mainly determined by the integrator $\tau_{neuron} = R_{int}C_{int}$. For the rate of rotation τ_{rotor}, an empirical value of $\tau_{rotor} = \tau_{neuron}/8$ is used.

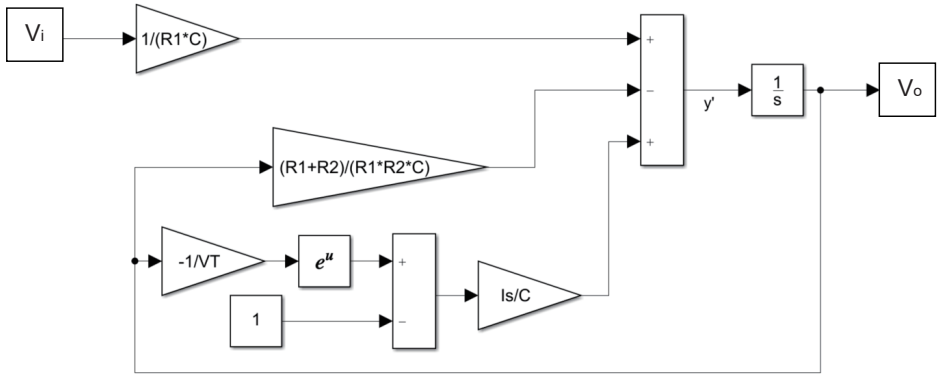


Fig. 4.4 Neuron model

Fig. 4.2(d) and (e) plot the nonlinearity (a rectified linear unit (ReLU) that can be implemented with a diode) and integration characteristics (with a time constant $\tau_{neuron} = R_{int} \times C_{int}$ for the neuron), respectively. In the absence of the diode, the activation function becomes linear. Most of the recently reported devices and materials for physical RC could also be used as the neuron in the RNR architecture [17, 63, 64]. Finally, an eRNR can be built by combining rotors and neurons.

4.5 Simulation: parameters matching and system modelling

4.5.1 Parameters matching method

A noise-free simulator was developed to evaluate the performance of the eRNR under different configurations and demonstrate its equivalence to a CR (as proven analytically in above sections). The first simulation was designed to confirm the consistency between the RNR and the CR and emphasize the role of rotation in the RNR. The key network characteristics based on different parameters, nonlinearities and rotation directions were investigated. Before comparing the network characteristics of the software CR and the hardware RNR, a numerical

method was developed to calculate the software CR parameters, such as the input scaling factor a and recurrent strength b, from the RNR behaviors to find the CR counterpart for a hardware RNR. Given a properly configured RNR, its CR counterpart should exist and exhibit similar network characteristics. Parameter matching provides a numerical method to determine the CR counterpart. The main difference between a hardware RNR and a software CR is associated with nonideal dynamic neurons, which result in different amplitude ranges for integration and nonlinearity. Therefore, the objective is to find the appropriate scaling coefficients for the software activation function to approximate the hardware neuron output under the same input $W_{in}u(n)$. An arbitrary $u(n)$ was generated as an input to the RNR, and the neuron output $q(n)$ was obtained. Assuming that this $q(n)$ is generated by a software CR, a comparative neuron update vector can be defined:

$$q_p\left(n+\mathbf{1},a,b,V_c\right)=\mathrm{ReLU}\left(bq(n)+a\mathbf{W}_{\mathrm{in}}\,u(n),V_c\right) \tag{4.7}$$

The prime task-independent network characteristic for a reservoir is the MC, which indicates its capability to retain the fading memory of the previous input[16, 149], and plays a critical role in the reservoir's performance in temporal signal processing. The standard MC measurement is introduced in Section 2. In addition to MC, the other three important network characteristics are computing ability (CA), kernel quality (KQ) and generalization rank (GR)[50]. They are calculated as follows:

- The MC can be quantified by a binary sequence reconstruction task. In this task, a binary sequence is randomly generated: $u(n)=1 or -1$, and $x_i(n)$ is the shifted version of $u(n)$ by i steps: $x_i(n)=u(n-i)$, for $i=1, 2, ...\infty$. In our simulation, 4000 data points of $u(n)$ were generated, in which 3000 points were used in training and 1000 points were used in testing. The $u(n)$ was injected into the reservoir, resulting in the state output $s(n)$. The training state matrix was used to calculated $\mathbf{W_{out}}$ using Ridge

regression and the target was $x_i(n)$, for $n \in 1,3000]$. In the testing phase, the testing state matrix ($s(n)$ for $n \in 3001,4000]$) was multiplied by the trained $\mathbf{W_{out}}$ to obtain $y_i(n)$, which indicated the reconstruction result. If the resulting $y_i(n)$ can properly match the testing $x_i(n)$, the reservoir can retain the information i steps ahead, which implies its memory property. Then, the MC can be quantified by taking the sum of the linear correlation between reconstruction $y_i(n)$ and actual shifted sequence $x_i(n)$ for i going from 1 to infinity according to Eq. 2.9.

- KQ evaluates the reservoir's performance on a high-dimensional nonlinear mapping. An ideal reservoir should be able to generate linearly separable node states in a higher dimensional space for different inputs. For an N-neuron reservoir, N different input sequences $U = [u_1, u_2 ... u_N]$ were generated and every u_i contains k random values. The N sequences will be injected into the reservoir one by one. For every sequence, the state matrix generated by n^{th} data is of interest, while the first $n-1$ data are employed to initialize the reservoir dynamic. Collecting all n^{th} state matrix forms an $N \times N$ matrix. Ideally, the states generated by different input sequences should be linearly independent, which means the rank of the $N \times N$ matrix should be equal to N. Therefore, this normalized rank can measure the high-dimensional mapping quality, also known as KQ.

- GR tests the reservoir's response for similar inputs. The reservoir should react to the coming temporal and deliver their representative states onto their classes regardless the effect of the previous inputs and minor fluctuation. Otherwise, the reservoir would collapse into an unwanted chaotic system that is highly sensitive to the initial condition and noise. To test the GR, a random sequence u_x with a length of l was generated to connect to the end of every u_i used in the KQ to form a new sequence $U' = [[u_1 u_x], [u_2 u_x] ... [u_N u_x]]$. It means that, for every reservoir under tested, it is first fed by different sequence with random values. Followed by all random values, the

same sequence ux with length l is fed into all reservoirs. Again, the states collected at the end or $(n+l)^{th}$ input form the $N \times N$ matrix to calculate the rank with 0.01 tolerance. In contrast to the KQ, a low rank means that the reservoir can quickly shift its attention to the latest l inputs, which is more desirable. Note that the choice of l should be determined by the temporal signal in actual application. For example, in the NARMA10 task, each point of NARMA10 is highly relevant to the previous 10 points. Therefore, a reservoir that gains a low GR when $l = 10$ can better couple with the NARMA10 system for prediction.

- CA is simply a combination of KQ and GR. It has been defined that a reservoir with good CA should be able to quickly focus on the certain length of previous inputs (low GR) and map it onto a linearly separable space (high KQ). Thus, CA can be calculated by the difference between them: CA = KQ - GR.

4.5.2 Results

Fig. 4.5(a) plots the MC as a function of reservoir size N in different scenarios. An excellent agreement in the MC between the eRNR and its CR counterpart for both ReLU and linear activation functions can be observed. The ReLU neurons yielded a lower MC because the nonlinearity suppressed the fading information for previous inputs, as also observed in earlier studies [131, 149]. For the RNR, the effect of the rotating direction is investigated to validate the design of the two rotors. The four lines at the bottom of Fig. 4.5(a) show the MC when the two rotors stopped or co-directionally rotated. The near-zero MC suggests that in cases with no rotation and counter-directional rotation, the RNR failed to implement RC functionalities since there was no MC for processing the temporal signal.

The CA, KQ and GR were analyzed by varying the time constant of neurons τ_{neuron}, which also changed the parameter matching result for the CR counterpart. As shown in Fig. 4.5(b),

the network characteristics of the physical eRNR again matched that of its CR counterpart. Here, the minor difference may be attributed to the imperfect diode characteristics as a ReLU function. The results presented in Fig. 4.5(a) and (b) corroborate the finding that a properly configured RNR (rotation in a common direction) is equivalent to a software-based CR and hence can be used for implementing physical RC.

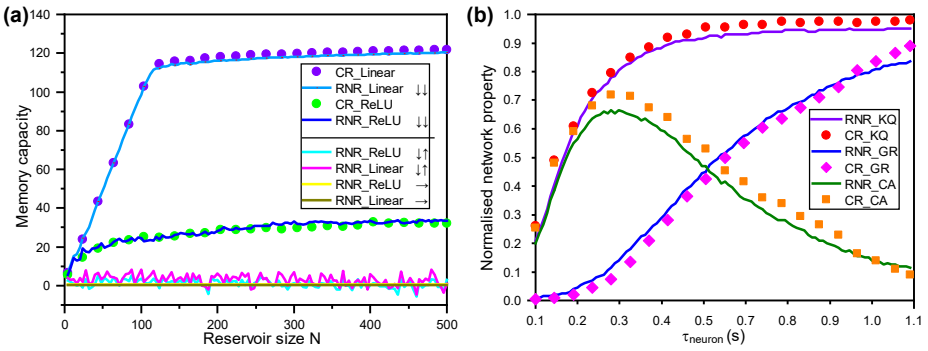


Fig. 4.5 eRNR simulation results for network characteristics and nonlinear system approximation. (a) MC versus the reservoir size N for different scenarios. The two blue lines plot the MC of the eRNR using dynamic linear and ReLU neurons, respectively. The purple and green dots are obtained from the parameter-matched CR counterparts. The remaining four lines show the MCs of dysfunctional RNRs (counter-directional rotation and no rotation). The symbols '$\downarrow\downarrow$', '$\downarrow\uparrow$' and '$\rightarrow$' indicate that the pre- and post-neuron rotors perform common-direction rotation, counter-directional rotation and no rotation, respectively. The parameters are $\tau_{neuron} = 1s, \tau_{rotor} = 0.125s, a' = 0.5$ and $M = 1$. (b) The CA, GR and KQ as a function of τ_{neuron} for the dynamic neurons. For every τ_{neuron} value, the properties of the RNR are first calculated. Then, the CR counterpart is calculated through the parameter matching method, and the results are analyzed. The obtained parameters are $\tau_{rotor} = 0.125s, a' = 0.5, N = 200$, and $M = 1$, and nonlinearity is provided by the diode.

4.6 Benchmark: System approximation of NARMA10

As an implementation of RC, the eRNR should be able to approximate a nonlinear system, for which a nonlinear autoregressive moving average system (NARMA) is a widely recognized benchmark for testing RC performance. A standard 10^{th}-order NARMA system can be express by the following formula:

$$y(n+1) = 0.3y(n) + 0.05y(n)\sum_{i=0}^{9} y(n-i) + 1.5u(n)u(n-9) + 0.1 \tag{4.8}$$

where $u(n)$ is a randomly generated white noise input in the range of [0, 0.5] and $y(n+1)$ is the target number. As can be observed in Eq. 4.8, the recursive configuration demands both nonlinear fitting and MC for the prediction model. In this task, an eRNR model was used to receive the $u(n)$ input and then predict the $y(n+1)$ output after training. In total, 4000 data samples ($u(n)$ and $y(n)$) for NARMA10 were generated to train (3000 samples) and test (1000 samples) the eRNR model. Given the same $u(n)$, the NRMSE (see Eq. 3.4 in Chapter 3) of the predicted result $y'(n)$ versus $y(n)$ calculated with the NARMA10 model based on Eq. 4.8 was used to quantify modelling performance. In the first trial, two key parameters of the eRNR, the input scaling factor a' and time constant of dynamic neurons τ_{neuron}, were assessed while other parameters were fixed to obtain the optimal NRMSE for a single 400-neuron eRNR. The input scaling factor changes the effective range of nonlinearity, and the time constant affects the decay factor b'. The noise-free simulation result is plotted in Fig. 4.6(b), where the optimal value (NRMSE = 0.078) was found at $a' = 0.061$ and $\tau_{neuron} = 1.1s$. It is worth mentioning that in a neuromorphic computing system, the electronic devices directly interacting with the environment and natural signals could exhibit a much longer time constant (e.g., >millisecond scale) compared with that of typical digital systems [20]. A fast time constant could result in an insufficient MC for retaining historical information. Such biologically realistic time constant values (τ_{neuron} and τ_{rotor}, from milliseconds to seconds scale) were used throughout the explored hardware implementation and simulation processes. The performance can be further improved by increasing the number of parallel reservoirs M with different input weights $\mathbf{W_{in}}$ as illustrated in Fig.4.3. As shown in Fig. 4.6(a), the resulting NRMSE can be clearly reduced by increasing M or N. The minimum NRMSE achieved in this experiment is 0.055 at $N = 388$ and $M = 50$. Fig. 4.6(e) shows an instance of the predicted value $y'(t)$ in comparison with the ground truth $y(t)$ when NRMSE = 0.055.

To the best of our knowledge, the NRMSE values for both the single eRNR (0.078) and parallel eRNRs (0.055) are lower than those reported in previous studies [23, 56] in the field of RC. Notably, the exponential form of nonlinearity in the transition region of the diode (different from the ideal ON/OFF form in the ReLU function used by software) enhances the state representation of the NARMA10 system. This result demonstrates the tremendous potential of the eRNR in high-order nonlinear system approximation due to the rich physical dynamics of electronics devices.

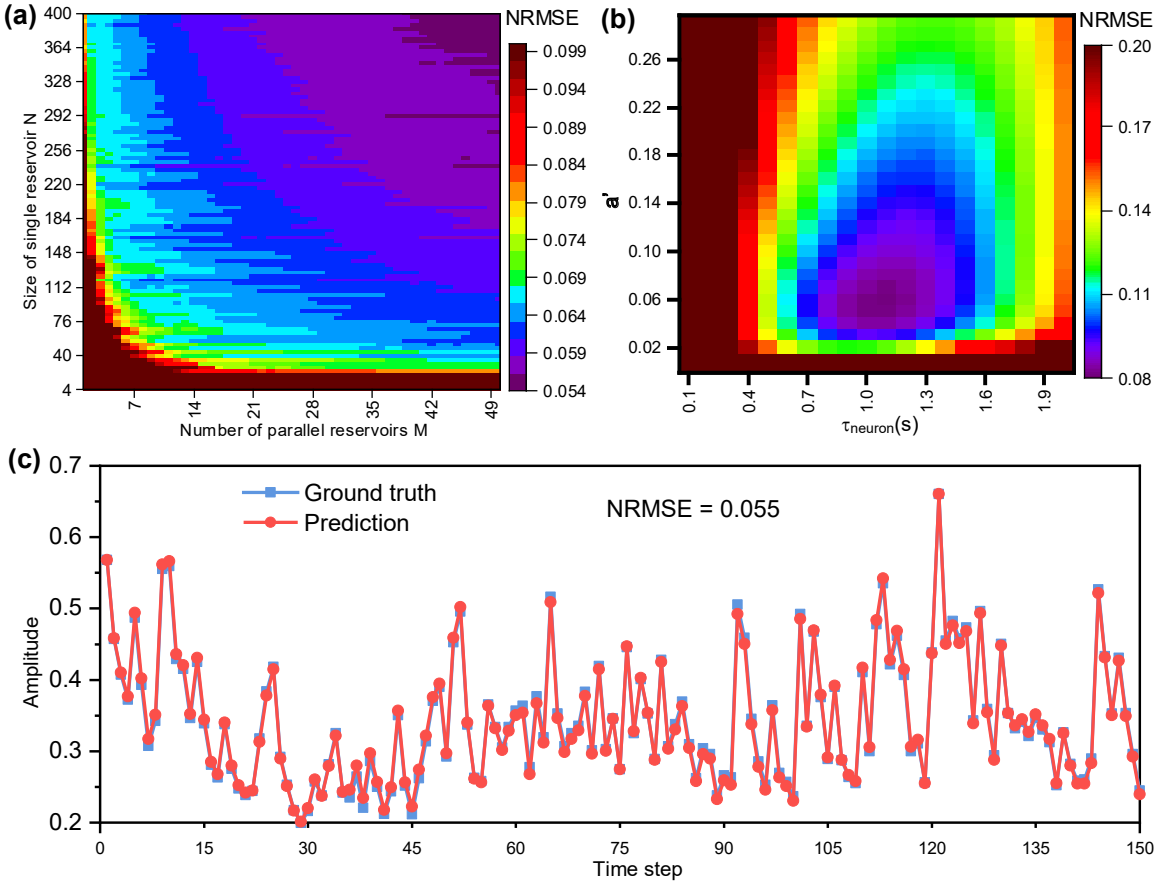


Fig. 4.6 eRNR simulation results for NARMA10. (a) NRMSE result for the NARMA10 system approximation task based on the two key parameters: the time constant τ_{neuron} and input scaling factor a'. The other parameters are $N = 400$ and $M = 1$. (b) NRMSE result for the NARMA10 modeling task when varying the reservoir size N and the number of parallel reservoirs M. The parameters are τ_{neuron} = 1 s, $\tau_{rotor} = 0.125s$ and $a' = 0.05$. (c) An example prediction result $y'(n)$ and the ground truth $y(n)$ when NRMSE=0.055, that is, for the best result obtained in (b). The parameters are $\tau_{neuron} = 1s$, $\tau_{rotor} = 0.125s, a' = 0.05, N = 388$, and $M = 50$ in this case, and a diode with a ReLU function is used.

Chapter 5

Rotating Neurons Reservoir: Implementation and Experiments

In this chapter, a hardware prototype was developed for near-sensor computing, chaotic time-series prediction and handwriting classification. By integrating a memristor array as a fully-connected output layer, the all-analogue RC system achieves 94.0% accuracy, while simulation shows $> 1000\times$ lower system-level power than prior works. Therefore, our work demonstrates an elegant rotation-based architecture that explores hardware physics as computational resources for high-performance RC.

5.1 Implementation

The schematic of eRNR is shown in Fig. 4.2. The network size of our prototype (Fig. 5.1) is N = 8 and M = 8, which means the single eRNR consists of 8 neurons and there are 8 parallel eRNRs. Both pre- and post-neuron rotors were implemented by eight CD4051B which is an 8-channel analogue multiplexer from Texas Instrument. The three signal selection ports were connected to a 3-bit binary counter consisted of a 4-bit counter (74LS161) and an inverter

(74HC04). The input mask was implemented by 8 switches to select positive or negative signals. In order to improve the state richness, each eRNR circuit should use a different input mask configuration.

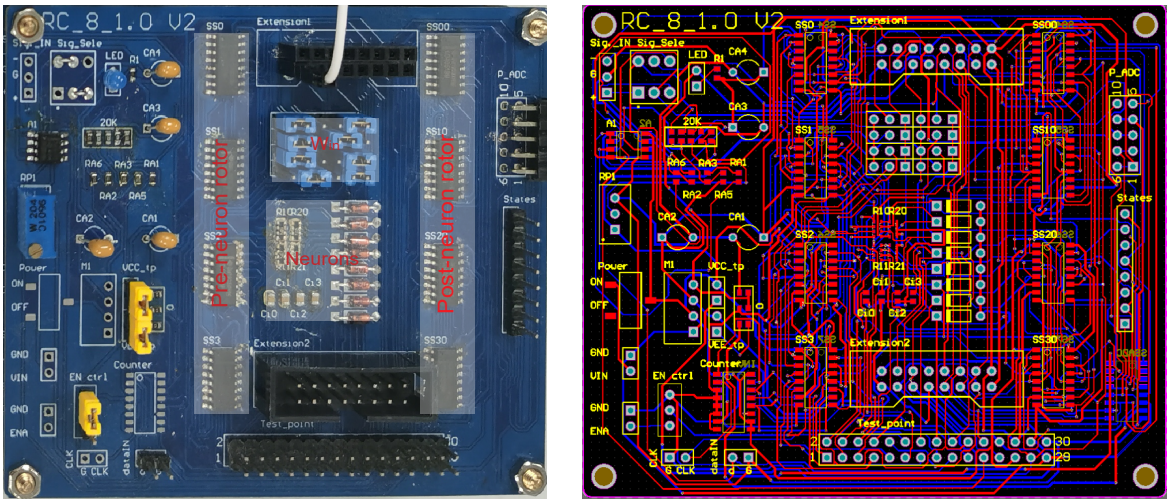


Fig. 5.1 An 8-neuron eRNR PCB board for real-time demonstration and data collection (left) and its design (right).

The customized demonstration and data acquisition platform serves to interface with the eRNR hardware, collect experimental data, and perform real-time demonstrations but regardless of low-power design. The 64 state channels on the eRNRs were connected to 8 12-bit ADC channels on a STM32 microcontroller via 8 multiplexers. There are additional 2 ADC channels on STM32 for collecting sensory signals. The total 66 channels data were sent to PC via universal asynchronous receiver/transmitter (UART) communication. The user interface software developed on LabVIEW received the data packages and plotted them in real-time. The software can also store the data in a file for further processing. Here, a proof-of-concept prototype with $\tau_{neuron} = 1s, N = 8$ and $M = 8$ was developed. The eight parallel eRNRs shared common power, counter, positive input and negative input characteristics. The input weight $\mathbf{W_{in}}$ varied for every eRNR to create diverse neuron dynamics and increase the state richness. The picture of software and hardware are shown in Fig. 5.2.

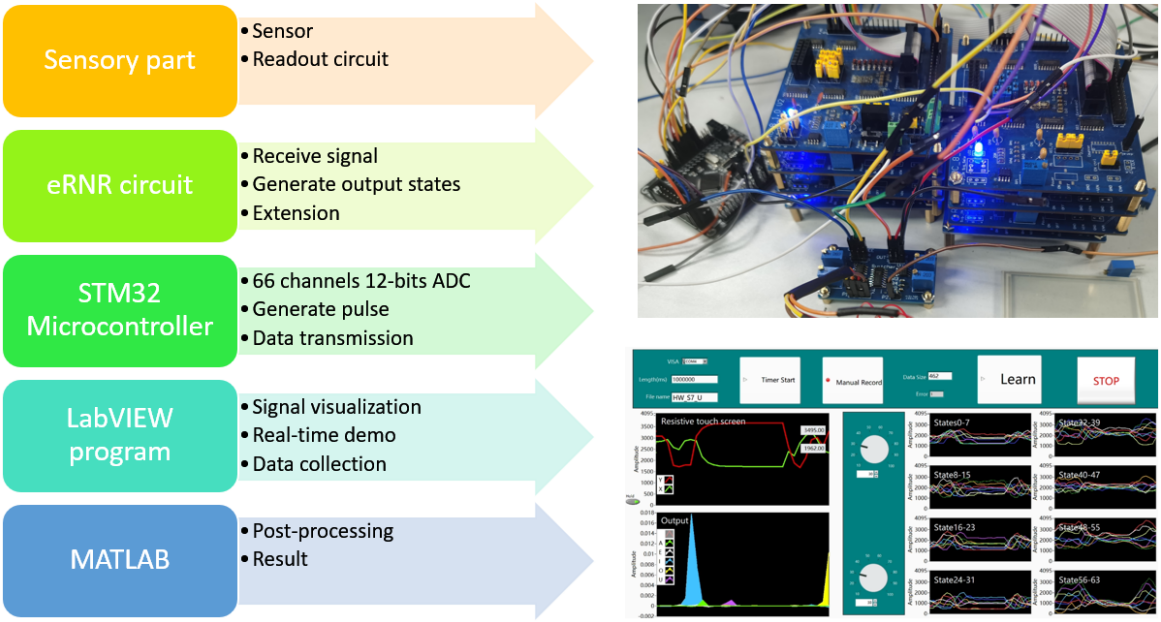


Fig. 5.2 An 8×8 eRNR system prototype.

In the real-time Mackey-Glass signal prediction demonstration (see the following sections), the software can also execute a MATLAB script where the Ridge regression was performed to calculate the $\mathbf{W_{out}}$. Then, the $\mathbf{W_{out}}$ was multiplied by the 64 state values at every time step to output the prediction result that was plotted in both the time-amplitude window and the phase window. In the handwriting recognition task (see the following sections), the platform collected the handwriting and state data for every participant. The $\mathbf{W_{out}}$ was trained in MATLAB by post-processing all the data from 8 participants. After training, the software can read the trained weights and calculate the 5 channels result indicating the probabilities of the 5 vowel classes.

5.2 Mackey-Glass chaotic time series prediction

To evaluate the state generation performance, the first experiment with the 8×8 eRNR system was a multistep ahead prediction for Mackey-Glass chaotic system, which has been

used in various RC studies as a benchmark task [23, 48, 64]. The Mackey-Glass system is defined by:

$$\frac{dy}{dt} = \beta \frac{y(t-\tau_{MG})}{1+y(t-\tau)^n} - \gamma y(t) \tag{5.1}$$

where the system parameters γ, β, and n were set to the widely used values 0.1, 0.2, and 10, respectively. Additionally, the system is chaotic when $\tau_{MG} > 16.8$, and predictions become correspondingly more difficult. In this experiment, $\tau_{MG} = 17$ and the initial value $y(0) = 1.2$ were set following previous works. The samples generated based on the Mackey-Glass system were input into the 8 × 8 eRNR system with a sampling rate of 8 Hz. This sampling rate should be the same as the driving frequency of the counter to ensure that every sample point is captured; that is, $\tau_{rotor} = 0.125s$. Based on this configuration, the 64 parallel output channels produce state values of the measured voltage for postprocessing. With our customized demonstration platform, the Mackey-Glass chaotic signal $y(n)$ was continuously fed into the eRNR system. The training state matrix $s(n)$ with a length of 64 based on $y(n)$ was used for output weight $\mathbf{W_{out}}$ training through linear regression, and the target value was input into the Mackey-Glass dataset shifted by i steps $(y(n+i))$. Here, the number of shifted steps i depended on how many steps ahead of $y(n)$ the system could predict. The system continuously received $y(n)$ without any preprocessing and produced 64 state outputs, which were multiplied by $\mathbf{W_{out}}$ to predict the value $y'(n+i)$. This process was performed in real time with the demonstration platform, and all the data, including $y(n), y'(n+i)$ and $s(n)$.

To better understand how the number of parallel RNRs (i.e., M) affected the prediction performance of the system, the states within 360 s (2880×64 samples, half for training and half for testing) were collected with the platform. Again, the NRMSE was used to quantify the difference between the actual values $y(n+i)$ and the predicted values $y'(n+i)$. The result is shown in Fig. 5.3(b). As i increased, the time series became increasingly difficult to

predict, resulting in a higher NRMSE; however, this NRMSE increase can be alleviated by using additional parallel reservoirs to enhance computational performance. Two examples of one-step-ahead prediction using one reservoir (NRMSE = 0.17) and eight parallel reservoirs (NRMSE = 0.03) are plotted in Fig. 5.3(c) and (d), respectively. The traces of $y(n+i)$ and $y'(n+i)$ in the phase space were also examined (Fig. 5.3(e) and (f)). The traces of eight eRNRs exhibited excellent consistency with the true values compared with the traces for the one-reservoir system. These experimental results suggest that the 8×8 eRNR prototype can be used to make accurate predictions of variables in the Mackey-Glass chaotic system after training. Even with the inevitable noise introduced by the analogue circuits, the eRNR can successfully emulate the chaotic system, with a low NRMSE of 0.03.

Moreover, our experiment revealed that the eRNR prototype can properly predict one-step-ahead for more chaotic signals ($\tau_{MG} > 17$) (Fig. 5.4(a-f)). In comparison, the system performance could degrade as τ_{MG} increases in multistep-ahead prediction (Fig. 5.4(g).

5.3 Demonstration of near-sensor computing: handwriting recognition

In the literature, some previously reported RC demonstrations achieved relatively low power consumption for certain parts inside systems using emerging devices and materials [17, 63, 64]. However, the operations for entire systems are usually overlooked. An interface between a sensory signal and the reservoir input is usually necessary, and assistive techniques, such as converting between digital and analogue data, memory buffering, preprocessing and feature extraction, are also often required [17, 56, 64]. These sophisticated operations increase system complexity and power consumption but are necessary in conventional physical RC and remain a key challenge for practical deployment [16]. In this work, a prime advantage

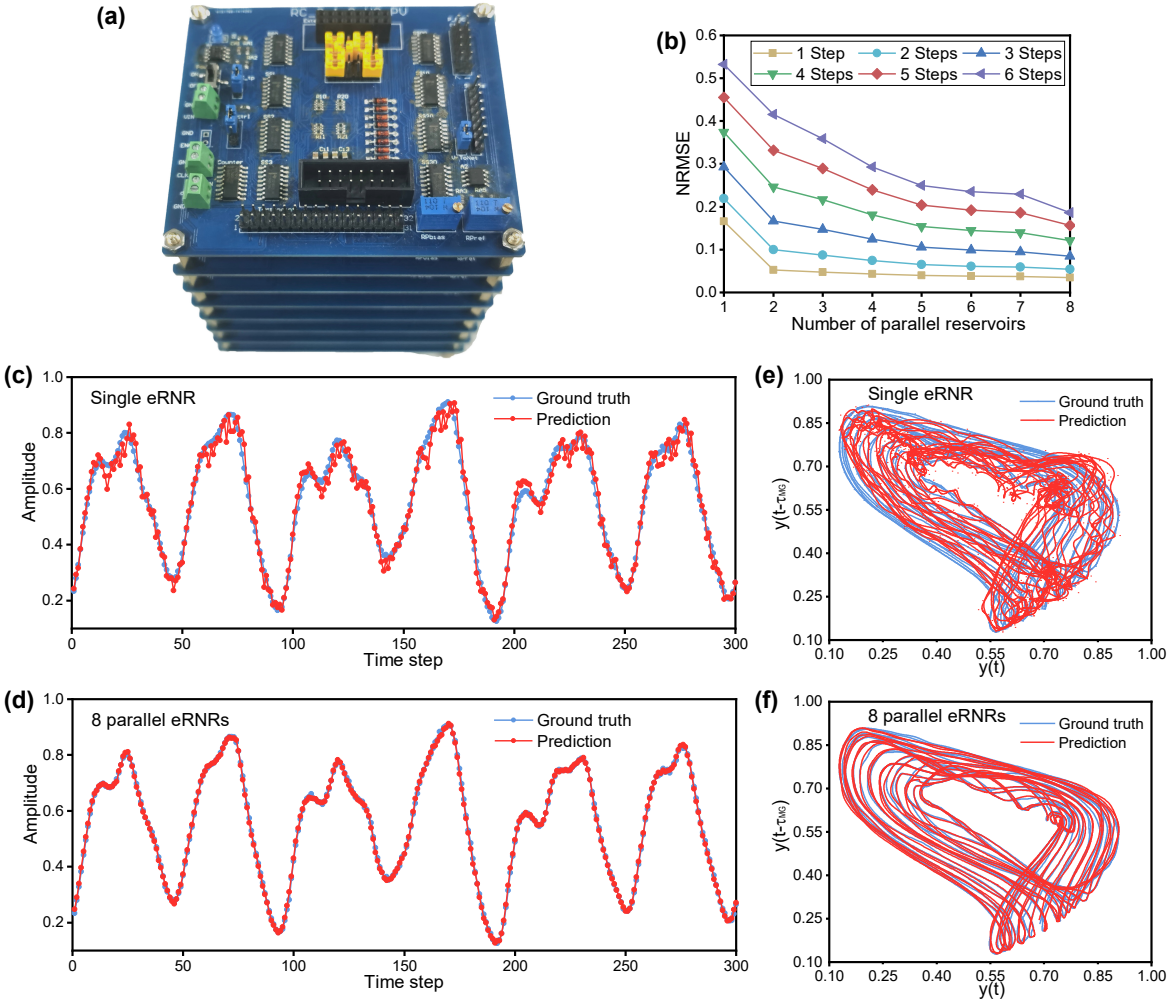


Fig. 5.3 8×8 eRNR prototype for Mackey-Glass time series prediction. (a) An eRNR prototype consisting of eight 8-neuron eRNRs (i.e., M = 8 and N = 8). (b) NRMSE result for multistep-ahead Mackey-Glass time series prediction. The state matrix used in this experiment was obtained from the parallel output channels of the eRNR hardware. (c-d) Two cases of one-step-ahead prediction with the Mackey-Glass time series result compared with the ground truth using (c) one eRNR (NRMSE = 0.17) and (d) eight parallel eRNRs (NRMSE = 0.03). (e-f) Phase space of the prediction compared with the ground truth using (c) one eRNR and (d) eight parallel eRNRs. The phase diagram was created by plotting the predicted and ground truth series $y(t)$ for the x-axis and $y(t - \tau_{MG})$ for the y-axis.

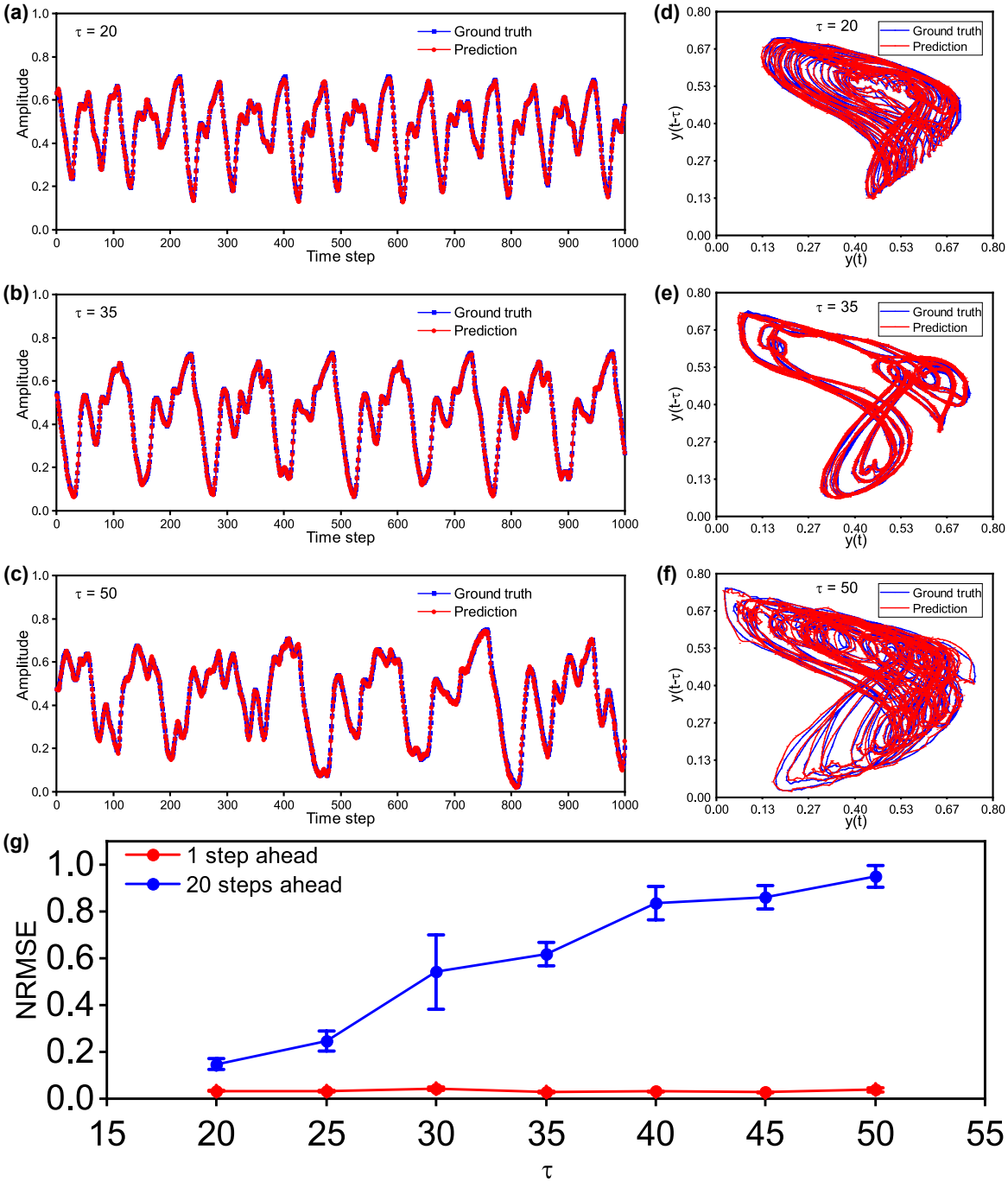


Fig. 5.4 Experimental results for Mackey-Glass chaotic signal prediction with $\tau_{MG} > 17$. (a-c) Three episodes of one-step ahead prediction of Mackey-Glass time series result compared with the ground truth using the chaotic signal with τ_{MG} = (a) 20, (b) 35 and (c) 50. (d-f) Phase space of the prediction compared with ground truth using the chaotic signal with τ_{MG} = (d) 20, (e) 35 and (f) 50. (g) NRMSE results of 1 and 20 steps ahead prediction with varied τ_{MG} values.

of our eRNR prototype is that it can directly receive analogue sensory signals and produce the parallel state output without any digital memory use or preprocessing, which could considerably reduce the power consumption of the overall system. In fact, this strength is highly attractive for emerging applications in analogue near-sensor computing; notably, the processor can act as a direct interface for sensory signals for cognitive computing purposes [109].

5.3.1 Experimental setups

To demonstrate analogue near-sensor computing, a resistive touch screen was employed to provide an analogue sensory signal for a handwritten vowel recognition task. In the experimental setup, a front-end circuit converted the resistive variations into two continuous signals representing the X and Y coordinates of the activated pixel on the screen. The 8×8 eRNR system used in the Mackey-Glass task was divided into two 4×8 eRNR subsystems (i.e., N = 8 and M = 4) to process X and Y temporal signals, and the total length of the state channel was still 64. In this case, the two subsystems still shared common power and counter, but had different positive and negative inputs from the X and Y axes. A photograph of the hardware is shown in Fig. 5.5. This experiment demonstrates that five different handwritten vowels (A, E, I, O, and U) can be distinguished after high-dimensional nonlinear mapping in the eRNR. Additionally, one important advantage of using RC systems is that their short-term memory property allows the network to retain the fading information of previous inputs in the state matrix at each time step. Thus, the state matrix obtained at the end of a handwritten event contains the information for the entire handwritten trace. After training, the eRNR system can perform point-by-point analogue reservoir state generation without accessing digital memory. Consequently, the memory unit for storing a certain length of data, such as the data in a sliding window or segmented signal, in conventional machine learning approaches can be eliminated by making full use of the MC. Further advancement of this

system involves the analogue output weights stored in a memristor crossbar array to realize all-analogue signal processing [13, 150], for which the power consumption can be further reduced by taking advantage of the computing-in-memory capability of memristors. Thus, from the sensory signal to the classification result, the entire system can perform near-sensor computing in the analogue domain, as shown in Fig. 5.5.

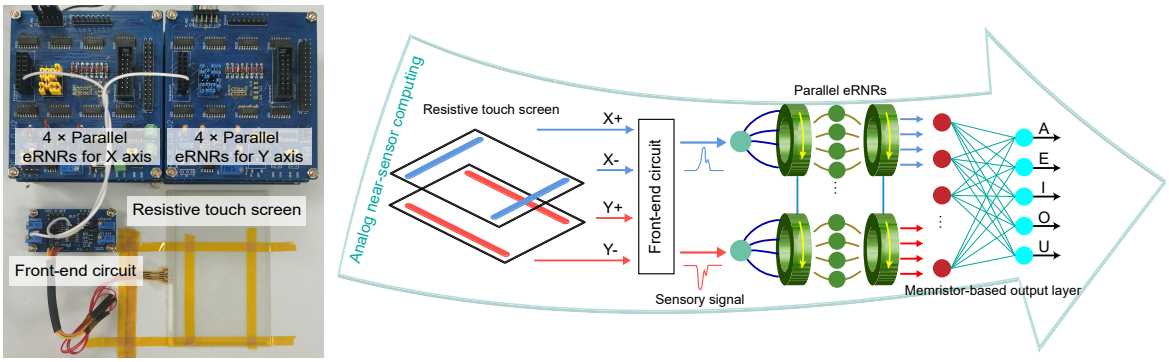


Fig. 5.5 Experimental setup for analogue near-sensor computing for handwriting recognition. The hardware used in this experiment: a handwriting sensor (resistive touch screen), a front-end circuit, and two 4×8 eRNR circuits for the x- and y-axes of the sensor.

5.3.2 Data collection and processing

The parameters of the eRNR used in the handwritten vowel recognition task are $\tau_{neuron} = 1s, \tau_{rotor} = 0.1s, N = 8$, and $M = 4$ (for each X and Y channel). All data were collected with our customized platform. In total, 66 channel data streams, including the two axis signals and signals from 64 reservoir state channels, were collected at each time step. During the data collection process, eight participants were asked to write the five vowels on a resistive touch screen, and repeat at least 20 times for each vowel. Data for 1103 handwritten vowels (2802 s) were successfully collected. The location and class of each handwritten vowel were labeled at the final rising/falling edge of the X and Y raw data. the end of each handwritten vowel were labelled (the blue square in Fig. 5.6(b)) where the state matrix at this time step contains the information of the handwritten trace because of memory capacity. Specifically,

the 64×1 state matrix collected at the time denoted by the green dot can be considered a feature vector for the corresponding handwritten trace.

In our experiment, handwritten vowel data from eight participants were collected, and typical handwritings are displayed in Fig. 5.6(a). For different handwritten vowels, Fig. 5.6(b) shows the X and Y signals input into the eRNRs, and Fig. 5.6(c) shows the resulting state output of the 64 channels. After data collection and labeling, the database was divided into a training set (400 handwritten vowels; 1025.8 s) and a testing set (703 handwritten vowels; 1776.2 s). According to the point-by-point computation introduced above, the size of the training label matrix Y_{train} for the five classes should be a five-dimensional data stream in which only the locations of green squares are set to 1, and values of 0 are assigned at other points. For training $\mathbf{W_{out}}$ (64×5), ridge regression with the target = Y_{train} (five-dimensional label for 1025.8 s) and variables = S_{train} (64-dimensional state vector for 1025.8 s) was used. Next, $\mathbf{W_{out}}$ was multiplied by the test state matrix $(y'_{test} = S_{test} \times \mathbf{W_{out}})$ to obtain a five-dimensional output representing the possibility of five potential classes at each time step, which corresponded to the graphs in Fig. 5.6(d). To quantify the classification accuracy, the predicted output for the testing set y'_{test} was compared with the manually labeled locations y_{test}. For every location in a handwritten event, for example, $y_{test}(n)|n = nx$, the actual output was investigated to find the maximum value in the range of $y_{test}(nx-7)$ to $y_{test}(nx+3)$. The corresponding channel that output the maximum value was considered the predicted class.

Using the labeling, training and testing procedure introduced above, 683 handwritten vowels (of a total of 703 in the test set) were correctly recognized, yielding a high accuracy of 97.1%. Examples of the point-by-point outputs for the five classes are illustrated in Fig. 5.6(d), and the confusion matrix is shown in Fig. 5.7. The errors mainly occurred when predicting 'O', which was misclassified as 'U' in some cases since these two classes are associated with similar writing traces.

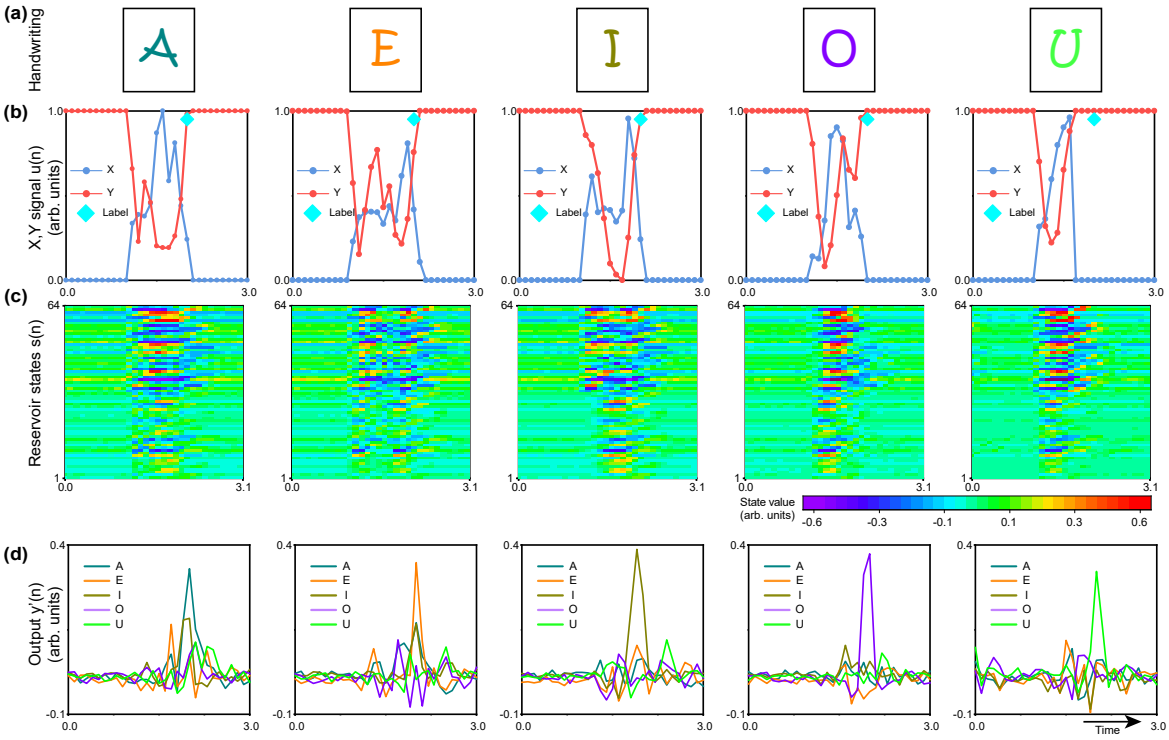


Fig. 5.6 The signal flows measured from the eRNR hardware for different handwritten patterns, including (a) the five handwritten vowels, (b) the sensory signals for the x- and y-axes $x(n)$, (c) the 64 channel reservoir states $s(n)$ of the eRNRs, and (f) the output $y(n)$ computed based on $s(n)$ and the trained weights.

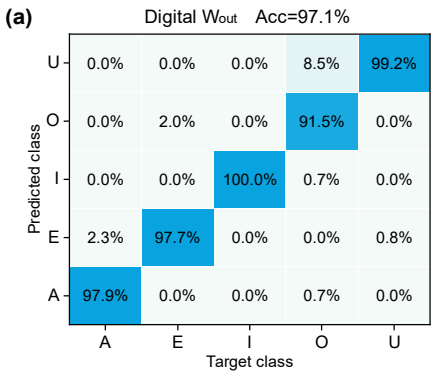


Fig. 5.7 Confusion matrix using digital $\mathbf{W_{out}}$ without noise-aware training. The overall accuracy is 97.1%.

5.3.3 Memristor-based output layer

The next experiment further integrated the eRNR system with a memristor crossbar array that served as the output layer. Memristor-based analogue computing has displayed excellent potential in neuromorphic computing. While the input and reservoir layer are generally established based on eRNR design, the output layer, which employs standard vector-matrix multiplication operations, can be effectively implemented by a memristor array for end-to-end all-analogue computing [80, 95]. The memristor array has a unit cell of one-transistor-one-resistor (1T1R). Each 1T1R consists of a resistive switching memristor with a material stack of TiN/HfOx/TaOy/TiN connected to a Si transistor that is fabricated using a standard 130nm Si CMOS process [150, 151]. The description of the memristor array can be found in Fig. 5.8. The computation principles of memristor-based analogue computing can be expressed as $I = V \times G = V \times (G_p - G_n)$, where G represents the weight matrix W, and G_p and G_n are the positive and negative conductance matrices, respectively. Furthermore, a standard write-with-verify scheme was used to map the weight matrix $\mathbf{W_{out}}$ to the conductance of the memristor array [13].

In this experiment, a differential pair of two memristors was used to represent one synaptic weight, so 640 memristors were used to represent all the weights in the above $\mathbf{W_{out}}$. It is noted that the analogue weights in a memristor array usually suffer from conductance variation issues (e.g., read noise) due to the nonideal device characteristics, leading to certain performance degradation compared with the floating-point digital weights in software [150]. The next simulation evaluated the effect of memristor conductance noise on the classification performance of the system to establish a proper training scheme. Fig. 5.10(a) shows the result of directly mapping $\mathbf{W_{out}}$ without noise-aware training; notably, the accuracy decreased significantly as the noise level increased. In our experiment, the intrinsic noise of the memristor was the dominant noise source in the all-analogue system. To achieve high accuracy, a noise-aware training method was adapted to obtain a robust $\mathbf{W_{out}}$ in the presence

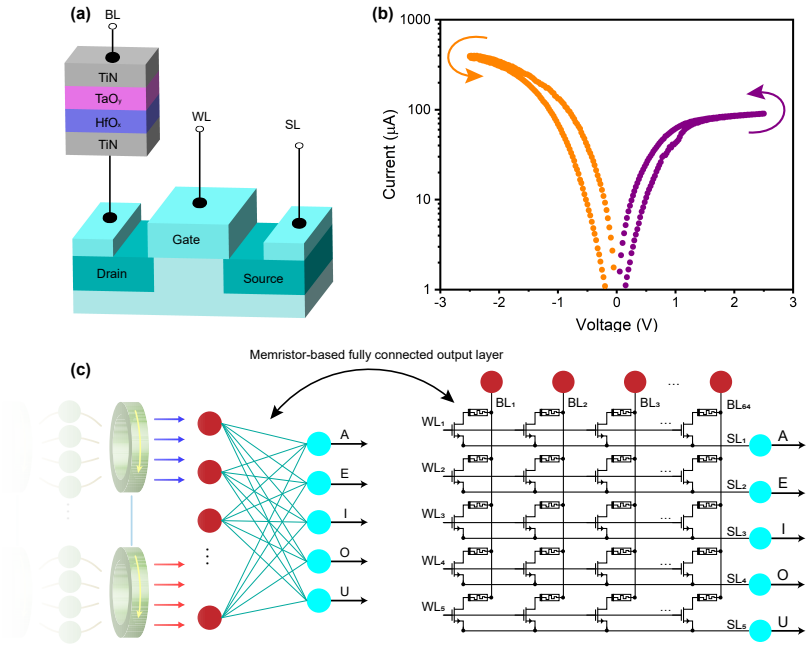


Fig. 5.8 Memristor-based fully connected output layer for eRNR. (a) Schematic of the 1T1R cell consisting of one transistor and one TiN/HfOx/TaOy/TiN memristor. (b) DC I-V characteristics of the memristor. (c) memristor-based fully connected output layer implemented by the 1T1R array. The WL, BL and SL indicate the word line, bit line and source line, respectively.

of memristor conductance variation [152, 153]. In the noise-aware training scheme, Gaussian white noise with a standard deviation of ± 0.03 was added to the normalized training state data before regression, and the resulting accuracy is plotted in Fig. 5.10(a). The comparisons between digital $\mathbf{W_{out}}$, target analogue $\mathbf{W_{out}}$ and the average values of the measured $\mathbf{W_{out}}$ after mapping are visualized in Fig. 5.9.

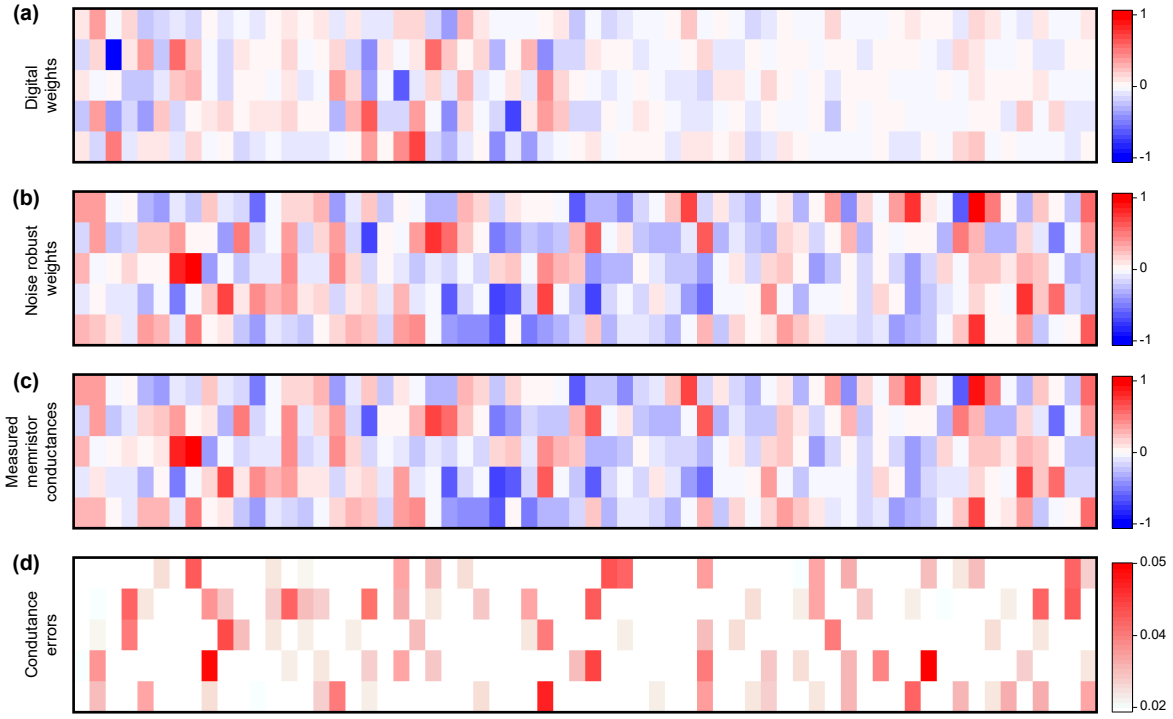


Fig. 5.9 Normalized output weights and error. (a) Output weights without noise-aware training. (b) Output weights with noise-aware training. (c) Analog output weights measured from the memristor array. (d) weights error resulted from the difference between the measured and target memristor conductance.

Most of the weight values can be successfully mapped to the memristor array with acceptable device variation, and the standard deviation (target conductance minus measured conductance) is approximately $0.368\mu S$. Finally, the confusion matrix using analogue $\mathbf{W_{out}}$ measured from the memristor array is shown in Fig. 5.10(b). Using the noise-aware training method and the measured analogue $\mathbf{W_{out}}$, the classification accuracy was improved from

29.2±0.9% (without noise-aware training) to 94.0±0.8% (with noise-aware training). The recognition result for each participant is summarized in Fig. 5.11.

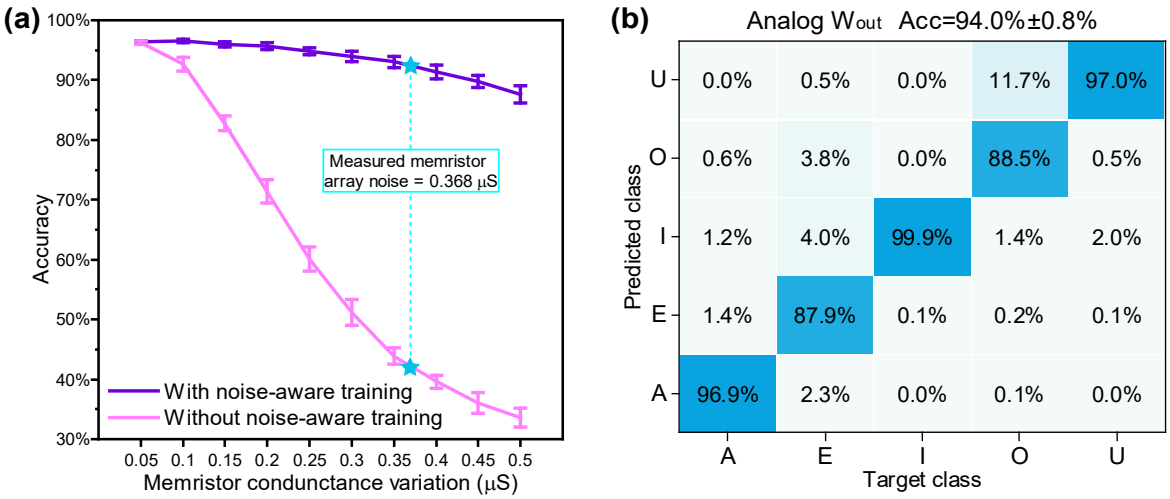


Fig. 5.10 Result of using memristor-based output layer with noise-aware training. (a). Classification accuracy as a function of simulated memristor conductance variation with and without the noise-aware training method. The measured average variation of the memristor array was $0.368\mu S$. (b) Confusion matrix using analogue $\mathbf{W_{out}}$ stored in the memristor array. The overall accuracy was 94.0%, with a standard deviation of 0.8%.

5.4 System-level power estimation and benchmark testing

The power consumption for the whole eRNR-based RC system can be divided into two parts: eRNR circuit consumption and the memristor array consumption. For the eRNR circuit, an 8-neuron eRNR was designed and simulated using a standard 65nm CMOS process based on the parameters used in the handwriting recognition task.

As shown in Fig. 4.2, the neurons, as passive components, are driven by the negative and positive sensory signals, providing a power source P_s. Also, the energy consumed by the counter and transmission gates depends on not only the static power but also the rate of rotation τ_{rotor}. The total power consumption P of the system consisting of M 8-neuron

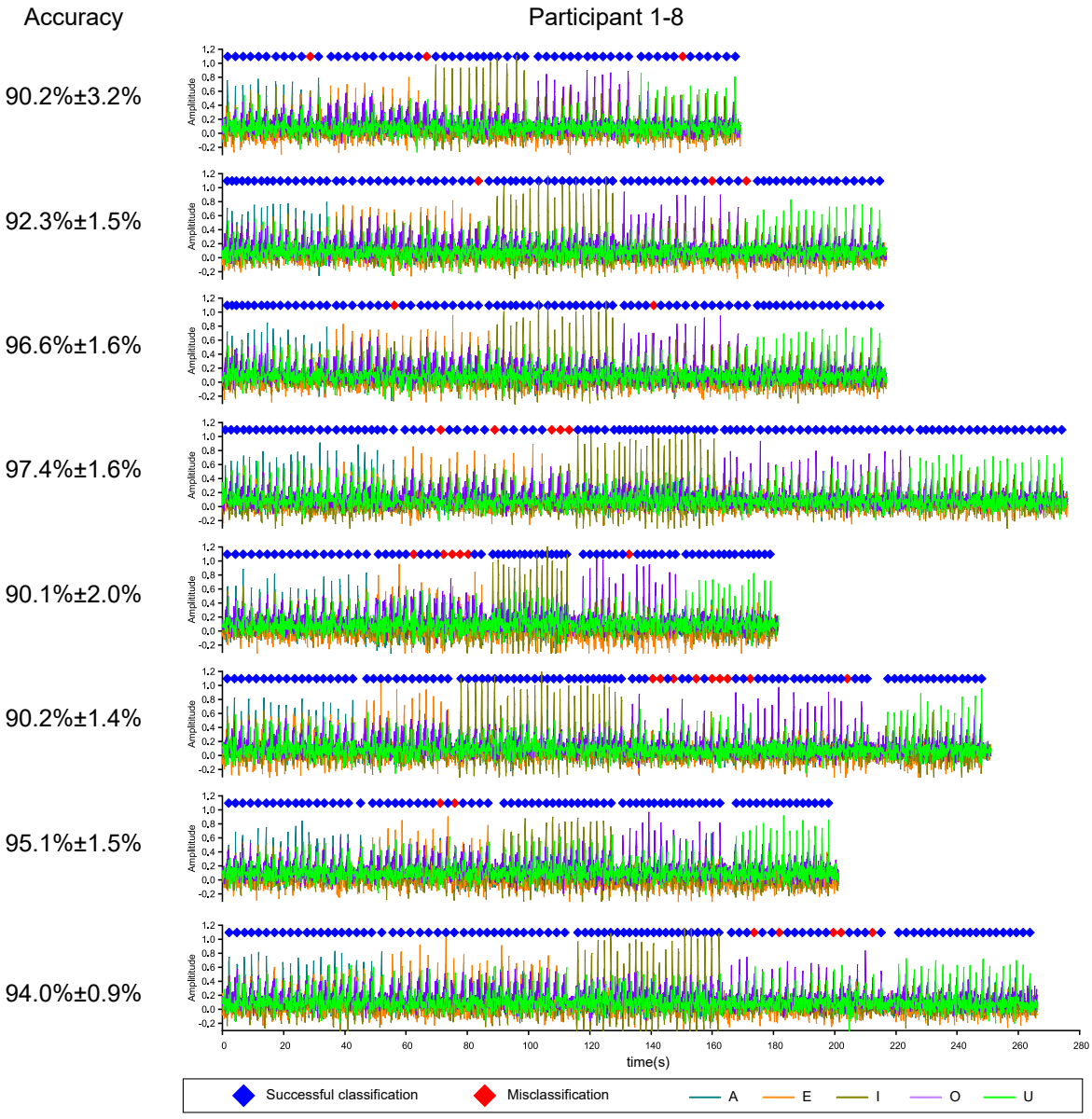


Fig. 5.11 Handwriting recognition result for each participant.

eRNRs (where the number of neurons N is fixed at 8) can be expressed as:

$$P = P_c + \left(P_s + P_t + \frac{E_c^{dyn} + E_t^{dyn}}{\tau_{rotor}} \right) \times M + \frac{E_m^{dyn}}{\tau_{rotor}} \tag{5.2}$$

where P_c and P_t represent the static power of the counter and transmission gates, respectively, and E_c^{dyn} and E_t^{dyn} represent the dynamic energy dissipated in the transition region driven by rate of rotation $1/\tau_{rotor}$. E_m^{dyn} is the energy consumed in the output layer (memristor array) for one inference. The M parallel eRNRs can share one counter, but the power for the other components increases with the number of parallel eRNRs M. For our application involving real-time handwritten signals, the operation period τ_{rotor} is a relatively slow (0.1 s) to match the time scale of human operations. The simulation result shows that $P_s = 3.27\mu W, P_c = 0.93\mu W$, and $Pt = 0.70\mu W$, regardless of how fast the rotors are operating. Moreover, the energy related to the rotation rate is $E_c^{dyn} = 0.31pJ$ and $E_t^{dyn} = 0.07pJ$. For the memristor-based output layer, the power dissipated by the voltage buffer driving the memristor array and the memristor array itself are $144\mu W$ and $0.8\mu W$ respectively. During every τ_{rotor}, only one-time inference is needed since all state channels are monotonously increased or decreased. The memristor array takes 50 ns to respond to the state voltage. Therefore, the dynamic energy of the memristor array for every inference step is $E_m^{dyn} = (144\mu W + 0.8\mu W) \times 50ns \times 64 = 463.36pJ/class$. The total power consumption of an 8×8 eRNR can then be calculated using Eq. 5.2. The simulated power breakdown at different frequencies is shown in Table 5.1. The simulation also suggests that the static power, mainly associated with the dynamic neurons and the leakage current of transistors, plays a dominant role when the processing rate $(1/\tau_{rotor})$ is lower than 100 kHz (for which the power consumption was estimated to be 79.1 μW). This striking advantage is associated with the unique all-analogue computing capability of our eRNR-implemented RC system, which saves the energy for frequent data conversion between digital and analogue domains. It should also be highlighted that our all-analogue eRNR provides more than three orders of

magnitude lower system-level power consumption compared with previous cutting-edge RC systems, whose power are in the ranges of 83 mW to 150 W using different implementation methods (Table 5.2) [76, 154–156]

Table 5.1 Simulated power breakdown for 8 × 8 eRNR system (μW)

Processing rate (Hz)	eRNR			Memristor	Total power
	Counter	Rotor	Neurons		
10	0.93	5.59	26.16	46.3×10^{-4}	32.7
10^3	0.93	5.59		46.3×10^{-4}	32.9
10^5	0.96	5.64		46.3	79.0
10^7	3.98	11.03		4633.6	4674.8

Table 5.2 Comparison with system-level power of literature-reported reservoir systems

Reference	Implementation	N	Processing rate (Hz)	Power
Alomar et. al.[154]	FPGA	48	10^6	1.5W
Kleyko et. al.[156]	FPGA	100	-	1.6W
Alomar et. al.[155]	FPGA	50	1142	83mW
Brunner et. al.[76]	Optoelectronic	388	13×10^6	150
This work	All-analog eRNR	64	10	$32.7\mu W$
			10^3	$32.9\mu W$
			100×10^3	$79.0\mu W$
			10^7	4.7 mW

5.5 Discussion: Why eRNR can be more resource-efficient?

From a fundamental perspective, the different mechanisms of introducing memory in rotation-based architecture and other architectures mainly determine their power efficiency. In the

rotation-based architecture, the memory is provided by the rotating dynamic node itself (see Fig. 4.2(a)). The excellent consistency between the rotation behavior and software algorithm frees the system from using extra control units, ADC, buffer and memory, which remarkably reduce the system complexity and power consumption. Also, implementing the logic switches for rotation is a resource-efficient use of CMOS-based transmission gates. Meanwhile, the rotating dynamic node serves to process signal and retain previous information simultaneously. Such in-memory computing paradigm is advantageous for low-power computing. In other architectures, such as the well-studied delay-based approach, the memory is actually separated from the processor. Although carrying out the processing in the nonlinear dynamic node was a significant progress, the memory is mainly provided by the delay unit which is constrained by the limitations of conventional digital computing, such as power consumption, throughput and latency [20]. These fundamental differences result in the better power efficiency for the proposed rotation-based architecture.

Compared with the classic random RC, the key difference of cyclic reservoir is the connection in the reservoir layer defined by $\mathbf{W_{res}}$. The $\mathbf{W_{res}}$ of random reservoir is a randomly generated matrix with a proper spectral radius, while the cyclic counterpart is a shifted identity matrix which can be implemented in a more deterministic manner without performance degradation [28]. In this work, it has been proven that the cyclic $\mathbf{W_{res}}$ can be equivalent to a physical rotor, while an effective physical counterpart of random $\mathbf{W_{res}}$ is yet to be found, which remains an exciting challenge to be addressed for future studies.

5.6 Conclusion

In summary, a hardware-friendly RNR architecture for all-analogue neuromorphic computing was developed; the resulting structure represents a fundamentally different reservoir architecture than those used in conventional hardware implementations. The proposed RNR has been

validated in theory, simulation, and experimental analyses. The theoretical analysis of RNR rigorously mapped the CR algorithm onto the physical rotation of dynamic neuron array, providing a solid foundation for hardware implementation. Such an RNR can be embedded into natural rotating components in various electronics, mechanical systems or even nanorobotics and empower them with computing capability. In the simulation using the eRNR model, the NARMA10 prediction task was performed to benchmark the system with varying hyperparameters, and record-low NRMSE values of 0.078 for a single eRNR and 0.055 for parallel eRNRs were achieved. It was found that the additional nonlinearity provided by the hardware-based dynamic neurons enhanced system performance in the approximation of the NARMA10 system, thus highlighting the computing potential of the proposed RNR. Furthermore, an 8×8 eRNR prototype was developed based on RNR theory for near-sensor analogue computing. The prototype successfully demonstrated multistep-ahead prediction of chaotic time series, and eight parallel reservoirs were found to reduce the prediction NRMSE from 0.17 to 0.03 for the studied Mackey-Glass chaotic system. This experimental result further validates the computing capability of our eRNR prototype under different experimental configurations. By further integrating the eRNR with an analogue memristor array as the fully connected output layer, an all-analogue RC system was realized to perform handwriting recognition tasks. A noise-aware training method was used to accommodate the conductance variation of the memristor array and improved the classification accuracy to 94.0%. In the simulation of the eRNR circuit, the overall system power consumption was estimated to be as low as 32.7 μW for the handwriting tasks operating at 10 Hz (τ_{rotor}= 0.1 s), reflecting an advantage of more than three orders of magnitude compared to the consumption reported for RC systems in the literature. Additionally, further power analysis suggested that the static power, mainly dissipated by the dynamic neurons, dominates the system at processing rates below 100 kHz, while the overall system power remains at a low level for high processing rates (>100 kHz) (see Table 5.2). This result can be explained by the fact

that most computations occur in the analogue domain that only contribute to static power, which is a general advantage of analogue neuromorphic computing. Dynamic power, mainly attributed to logic switches and memristor arrays, starts to dominate the system at processing rates higher than 100 kHz (see Table 5.1)).

To further enhance the eRNR system capabilities when performing complex tasks, a useful approach is to increase the number of neurons (N) or the number of parallel eRNRs (M) to expand the network size. Furthermore, a deep eRNR, consisting of multiple eRNR cells in series, could enhance the classification performance for inputs of different classes. From a hardware perspective, dynamic neurons could be replaced by recently reported emerging devices (e.g., dynamic memristors [63, 64] and spintronic devices [17]) to further reduce the system size and power consumption. Different configurations of neurons could be beneficial for enhancing state richness and improving system performance. In addition, the eRNR design can be miniaturized and monolithically integrated onto chips to reduce power requirements and promote ultrafast computing. It is also worth mentioning that various rotational hardware could be explored for constructing efficient pre- and post-neuron rotors, which are the key to implement the RNR. Our work demonstrates that the RNR is well-suited for large-scale and high-speed neuromorphic computing systems and has tremendous potential for use in applications involving the Internet of Things and edge computing, among others.

Chapter 6

Discussion and Future Perspectives

6.1 Main conclusions of this thesis

The dynamical properties of Rhat involves multiple implementation substrates such as optical devices and mechanics, this research studies physical RC with the two aspects of focus: electronic implementations and architectures. Overall, this thesis reviewed the existing reservoir computers in the literature. Then, the delay-based RC and parallel devices RC therein were investigated in depth, along with simulation and experiment results. Next, the RNR architecture originally proposed by the author is introduced and discussed in details, representing a promising paradigm of physical RC towards practical applications. The main conclusions of this thesis are summarized as follows:

- The achievements and novel applications of artificial intelligence are all based on the development of modern computer, whose fundamental element is transistor. In fact, apart from the 0' and 1' bits offered by transistor, there exists rich dynamics in electronics that can be explored as computational, which is of particularly high interest in the post-Moore's era. Physical reservoir computing is a promising paradigm to harvest

computing resource from hardware. The state-of-the-art physical reservoir computers are reviewed in Chapter 1. During the development of physical RC, architectural innovations played an more significant role. For most existing physical RC, this thesis divide their architectures into four categories according to their implementation of the reservoir layer (or middle layer), including delay-based reservoir, parallel devices reservoir, in-materia reservoir and rotating neurons reservoir.

- Delay-based reservoir computing is a promising architecture to implement physical RC. Because the use of time-multiplexing, only one or few neurons are needed to generate complex dynamic for high-dimensional mapping. In this work, the proposed DRC model directly receives raw and continuous ECG signal in the absence of signal segmentation and feature extraction, which is the prime difference compared with the existing software-based algorithms. After fully optimizing the parameters, the model successfully detect the VEB in the continuous ECG stream, while keeping the majority of the computing in analogue domain.

- Volatile memristors are well-suited to implement parallel devices RC because of its intrinsic short-term memory property. Using the time-multiplexing technique for the parallel memristors can increase the state richness and therefore improve the system performance. The parallel memristors system can act like a reservoir to handle simple temporal processing tasks like waveform classification and one-step ahead prediction for Hénon map chaotic signal. However, for more demanding tasks like human activity recognition, extra operations, such as delayed feedback, would be required in order to yield an acceptable accuracy.

- RNR is a recently proposed novel architecture for implementing physical. It was found that when rotating a specifically designed neuron array that processes nonlinearity and dynamical behaviours, the input and output of the hardware system can be roughly

equivalent to a CR algorithm, which can be proved mathematically. Such network-level equivalence is rare to found in other architectures. The simulation results emphasized the fact that, on one hand, the RNR hardware and software are roughly equivalent since their similar network behaviours were observed; on the other hand, software and hardware are not ideally equal and the hardware provides rich dynamics within a certain range that can be explored to enhance the performance.

- The hardware simulations and experiments on eRNR further demonstrated the potentials of this novel architecture. The eRNR prototype successfully demonstrated real-time Mackey-Glass chaotic signal prediction and near-sensor handwriting recognition. Another experiment involves using memristor array as output layer for the VMM operation to form end-to-end all-analogue computing. Finally, the power analysis suggested that the all-analogue eRNR system consumed three order of magnitude lower power than the existing physical RC systems. This is attribute to the RNR architecture that is equivalent to CR on network level, enabling the coherent, straightforward and elegant hardware implementation.

6.2 Future work and application

Algorithm

From the viewpoint of algorithm, RC provides an efficient way to implement RNN with less training cost. However, the network capability is relatively limited compared with deep and large networks such as CNN. As a machine learning algorithm, the existing RC-based algorithms, including the deep-RC proposed in recent years, have not demonstrated state-of-the-art result in benchmark tasks. Further advancing RC-based algorithms could be helpful for improving the computing potentials of physical RC. Also, a new network topology

could inspire new hardware architectures. For example, RNR stems from CR that is not a conventional RC. Therefore, developing RC algorithms for higher computing capabilities and more ingenious network typology is an important topic.

Architecture

Architecture of implementing physical RC remains a crucial topic in the field of RC. As mentioned in previous chapters, an effective architecture can fully explore the electronics' dynamics for computing. The existing architectures mainly includes delay-based RC, parallel devices RC, in-materia RC and RNR, which have their own strengths and drawbacks. For more practical uses, the further development of RC architecture should pay more attention to the cost of overall system including the peripheral circuit, assistive module and input and output layers, rather than only demonstrating the reservoir layer.

Processing core

Processing core plays an important role in a physical RC by providing nonlinearity (or activation function) and dynamical properties that significantly affect the performance. Given an architecture, an optimal processing core could minimize energy consumption and maximize processing result. In electrical RC, other factors, such as size, integration and large scale fabrication, should also be taken into consideration. Previous works had proposed various processing cores (see Chapter 1) for the architectures except recently-reported RNR. Therefore, investigating different processing cores and their compatibility with the architectures could be an important topic in the field of RC.

Integration

Implementing physical RC in integrated circuit is a promising direction. The previous works have proved that the RCs are achievable by electrical components and compatible with existing electronics. Particularly, the recently proposed eRNR can be achieved by CMOS circuit. Therefore, a further development on RC chip is therefore suggested.

Unified systematic benchmark

In the past decade, hundreds of physical RCs were demonstrated. Physical RC is becoming a more and more comprehensive field. However, most physical RC works explain their RC system from their perspective and several questions remain unanswered at present. For example, given a physical RC system, researchers from material science, electrical engineering and computer science would concern over completely different aspects. How to setup a criteria to judge that a physical RC is 'good' or 'bad' remains a problem in the RC community. Toward practical RC and make them comparable, an unified systematic benchmark is needed to more comprehensively evaluate an RC system.

Application

The application of physical RC should fully utilize its unique properties and advantages in comparison with other computing accelerators, including (1) trainable dynamical system, (2) intrinsic memory property, (3) temporal signal processing and (4) elegant hardware implementation. Possible future applications of physical RC are:

- Edge/in-sensor/near-sensor computing. Physical RC is a lightweight network which is well-suited to deploy at edge for pre-processing or feature extraction purposes.

Meanwhile, physical RC could integrate with sensor for in-sensor or near-sensor computing.

- High-speed nonlinear computing. As a analogue computer, physical RC is free from von Neumann bottleneck and thus makes ultra-high speed nonlinear computing possible.

- Nonlinear controller. Previous researches mainly used RC for machine learning tasks. However, RC can approximate measurable dynamic systems, enabling its application in control. Using physical RC as a trainable nonlinear controller is also a interesting topic.

- Complex system solver. RC algorithms have been used in solving spatiotemporally chaotic systems like Kuramoto-Sivashinsky equation [29]. Performing such RC algorithms on hardware could significantly accelerate the computing efficiency.

6.3 Epilogue

Physical RC is an interesting and promising topic. The recent studies has proved the rich dynamic existed in electronics that can be explored as computing resource under the concept of RC. However, there are still many unanswered questions about the physical RC before practical uses, which requires joint-effort from different disciplines.

References

[1] D. J. Frank, R. H. Dennard, E. Nowak, P. M. Solomon, Y. Taur, and H. S. P. Wong, "Device scaling limits of Si MOSFETs and their application dependencies," *Proceedings of the IEEE*, vol. 89, no. 3, pp. 259–287, 2001.

[2] E. Covi *et al.*, "Adaptive Extreme Edge Computing for Wearable Devices," *Frontiers in Neuroscience*, vol. 15, no. May, pp. 1–27, 2021.

[3] J. Tang *et al.*, "Bridging Biological and Artificial Neural Networks with Emerging Neuromorphic Devices: Fundamentals, Progress, and Challenges," *Advanced Materials*, vol. 31, no. 49, 2019.

[4] W. Zhang *et al.*, "Neuro-inspired computing chips," *Nature Electronics*, vol. 3, no. 7, pp. 371–382, 2020.

[5] G. E. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[6] R. R. Schaller, "Moore's law: past, present and future," *IEEE Spectrum*, vol. 34, no. 6, pp. 52–59, 1997.

[7] L. Xiu, "Time Moore: Exploiting Moore's Law from the Perspective of Time," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 1, pp. 39–55, 2019.

[8] W. Olin-Ammentorp and N. Cady, "Biologically-Inspired Neuromorphic Computing," *Science Progress*, vol. 102, no. 3, pp. 261–276, 2019.

[9] C. D. Schuman *et al.*, "A Survey of Neuromorphic Computing and Neural Networks in Hardware," *arXiv preprint*, pp. 1–88, 2017.

[10] J. Grollier, D. Querlioz, and M. D. Stiles, "Spintronic Nanodevices for Bioinspired Computing," *Proceedings of the IEEE*, vol. 104, no. 10, pp. 2024–2039, 2016.

[11] J. Shalf, "The future of computing beyond Moore's Law," *Philosophical Transactions Royal Society*, vol. 378, no. 20190061, pp. 1–14, 2020.

[12] X. Liang *et al.*, "Rotating neurons for all-analog implementation of cyclic reservoir computing," *Nature Communications*, vol. 13, no. 1, p. 1549, Dec. 2022.

[13] P. Yao *et al.*, "Fully hardware-implemented memristor convolutional neural network," *Nature*, vol. 577, no. 7792, pp. 641–646, 2020.

[14] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "The cat is out of the bag: Cortical simulations with 109 neurons, 1013 synapses," *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, SC '09*, no. c, 2009.

[15] X. Liang *et al.*, "Fusion of Wearable and Contactless Sensors for Intelligent Gesture Recognition," *Advanced Intelligent Systems*, vol. 1, no. 7, p. 1 900 088, 2019.

[16] G. Tanaka *et al.*, "Recent advances in physical reservoir computing: A review," *Neural Networks*, vol. 115, pp. 100–123, Jul. 2019.

[17] J. Torrejon *et al.*, "Neuromorphic computing with nanoscale spintronic oscillators," *Nature*, vol. 547, no. 7664, pp. 428–431, 2017.

[18] C. Mead, "Neuromorphic Electronic Systems," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[19] C. Mead, "How we created neuromorphic engineering," *Nature Electronics*, vol. 3, no. 7, pp. 434–435, 2020.

[20] G. Indiveri and S. C. Liu, "Memory and Information Processing in Neuromorphic Systems," *Proceedings of the IEEE*, vol. 103, no. 8, pp. 1379–1397, 2015.

[21] Y. Xi *et al.*, "In-Memory Learning With Analog Resistive Switching Memory: A Review and Perspective," *Proceedings of the IEEE*, 2020.

[22] J. D. Kendall and S. Kumar, "The building blocks of a brain-inspired computer," *Applied Physics Reviews*, vol. 7, no. 1, 2020.

[23] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks - with an Erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 1, no. 148, pp. 1–47, 2001.

[24] H. Jaeger and H. Haas, "Harnessing Nonlinearity: Predicting Chaotic Systems and Saving Energy in Wireless Communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004.

[25] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural Computation*, vol. 14, no. 11, pp. 2531–2560, 2002.

[26] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Networks*, vol. 20, no. 3, pp. 391–403, 2007.

[27] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, no. 3, pp. 127–149, 2009.

[28] A. Rodan and P. Tiňo, "Minimum complexity echo state network," *IEEE Transactions on Neural Networks*, vol. 22, no. 1, pp. 131–144, 2011.

[29] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach," *Physical Review Letters*, vol. 120, no. 2, p. 24 102, 2018.

[30] F. Palumbo, C. Gallicchio, R. Pucci, and A. Micheli, "Human activity recognition using multisensor data fusion based on Reservoir Computing," *Journal of Ambient Intelligence and Smart Environments*, vol. 8, no. 2, pp. 87–107, 2016.

[31] W. Wang, X. Liang, M. Assaad, and H. Heidari, "Wearable wristworn gesture recognition using echo state network," *2019 26th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2019*, pp. 875–878, 2019.

[32] A. Rodan and P. Tiňo, "Simple deterministically constructed cycle reservoirs with regular jumps," *Neural Computation*, vol. 24, no. 7, pp. 1822–1852, 2012.

[33] M. Dale, S. O'Keefe, A. Sebald, S. Stepney, and M. A. Trefzer, "Reservoir computing quality: connectivity and topology," *Natural Computing*, vol. 20, no. 2, pp. 205–216, 2020.

[34] P. Verzelli, C. Alippi, L. Livi, and P. Tino, "Input-to-State Representation in Linear Reservoirs Dynamics," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2021.

[35] A. Sheta, H. Faris, A. Rodan, E. Kovač-Andrić, and A. M. Al-Zoubi, "Cycle reservoir with regular jumps for forecasting ozone concentrations: Two real cases from the east of Croatia," *Air Quality, Atmosphere and Health*, vol. 11, no. 5, pp. 559–569, 2018.

[36] W. Maass, T. Natschläger, and H. Markram, "A model for real-time computation in generic neural microcircuits," *Advances in Neural Information Processing Systems*, 2003.

[37] Q. Ma, L. Shen, and G. W. Cottrell, "DeePr-ESN: A deep projection-encoding echo-state network," *Information Sciences*, vol. 511, pp. 152–171, 2020.

[38] C. Gallicchio, A. Micheli, and L. Pedrelli, "Deep reservoir computing: A critical experimental analysis," *Neurocomputing*, vol. 268, pp. 87–99, 2017.

[39] J. Long, S. Zhang, and C. Li, "Evolving Deep Echo State Networks for Intelligent Fault Diagnosis," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 7, pp. 4928–4937, 2020.

[40] C. Sun, M. Song, S. Hong, and H. Li, "A Review of Designs and Applications of Echo State Networks," *arXiv preprint*, pp. 1–37, 2020.

[41] H. Jaeger, "Adaptive nonlinear system identification with Echo State networks," *Advances in Neural Information Processing Systems*, 2003.

[42] H. Soh and Y. Demiris, "Iterative temporal learning and prediction with the sparse online echo state gaussian process," *Proceedings of the International Joint Conference on Neural Networks*, pp. 10–15, 2012.

[43] N. A. K. Doan, W. Polifke, and L. Magri, "Physics-informed echo state networks for chaotic systems forecasting," in *Computational Science – ICCS 2019*, J. M. F. Rodrigues *et al.*, Eds., Cham: Springer International Publishing, 2019, pp. 192–198.

[44] X. Yao, Z. Wang, and H. Zhang, "Prediction and identification of discrete-time dynamic nonlinear systems based on adaptive echo state network," *Neural Networks*, vol. 113, pp. 11–19, 2019.

[45] L. Sun, B. Jin, H. Yang, J. Tong, C. Liu, and H. Xiong, "Unsupervised EEG feature extraction based on echo state network," *Information Sciences*, vol. 475, pp. 1–17, 2019.

[46] M. Alfaras, M. C. Soriano, and S. Ortín, "A Fast Machine Learning Model for ECG-Based Heartbeat Classification and Arrhythmia Detection," *Frontiers in Physics*, vol. 7, p. 103, Jul. 2019.

[47] S. Ortín, M. C. Soriano, M. Alfaras, and C. R. Mirasso, "Automated real-time method for ventricular heartbeat classification," *Computer Methods and Programs in Biomedicine*, vol. 169, pp. 1–8, Feb. 2019.

[48] X. Zhu, Q. Wang, and W. D. Lu, "Memristor networks for real-time neural activity analysis," *Nature Communications*, vol. 11, no. 1, 2020.

[49] M. Lukoševičius, "A practical guide to applying echo state networks," in *Neural Networks: Tricks of the Trade: Second Edition*, G. Montavon, G. B. Orr, and K.-R. Müller, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 659–686.

[50] L. Appeltant, J. Danckaert, I. Fischer, and G. V. D. Sande, "Reservoir Computing based on Delay-dynamical Systems," Ph.D. dissertation, Joint PhD Vrije Universiteit Brussel and Universitat de les Illes Balears, 2012.

[51] L. Grigoryeva and J. P. Ortega, "Echo state networks are universal," *Neural Networks*, vol. 108, pp. 495–508, 2018.

[52] D. V. Christensen *et al.*, "2021 Roadmap on Neuromorphic Computing and Engineering," *Neuromorphic Computing and Engineering*, 2021.

[53] A. Goudarzi and C. Teuscher, *Reservoir Computing*. Springer, 2016, pp. 1–6.

[54] G. Van Der Sande, D. Brunner, and M. C. Soriano, "Advances in photonic reservoir computing," *Nanophotonics*, vol. 6, no. 3, pp. 561–576, 2017.

[55] K. Nakajima, "Physical reservoir computing—An introductory perspective," *arXiv*, 2020.

[56] L. Appeltant *et al.*, "Information processing using a single dynamical node as complex system," *Nature Communications*, vol. 2, no. 1, p. 468, 2011.

[57] M. C. Soriano, D. Brunner, M. Escalona-MorÃ¡n, C. R. Mirasso, and I. Fischer, "Minimal approach to neuro-inspired information processing," *Frontiers in Computational Neuroscience*, vol. 9, no. June, pp. 1–11, 2015.

[58] J. Li, C. Zhao, K. Hamedani, and Y. Yi, "Analog hardware implementation of spike-based delayed feedback reservoir computing system," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2017-May, pp. 3439–3446, 2017.

[59] J. Li, K. Bai, L. Liu, and Y. Yi, "A deep learning based approach for analog hardware implementation of delayed feedback reservoir computing system," *Proceedings - International Symposium on Quality Electronic Design, ISQED*, vol. 2018-March, pp. 308–313, 2018.

[60] B. Vettelschoss, A. Rohm, and M. C. Soriano, "Information Processing Capacity of a Single-Node Reservoir Computer: An Experimental Evaluation," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2021.

[61] D. Marković *et al.*, "Reservoir computing with the frequency, phase, and amplitude of spin-torque nano-oscillators," *Appl. Phys. Lett*, vol. 114, p. 12 409, 2019.

[62] R. Nakane, A. Hirose, and G. Tanaka, "Spin waves propagating through a stripe magnetic domain structure and their applications to reservoir computing," *Physical Review Research*, vol. 3, no. 3, pp. 1–15, 2021.

[63] Y. Zhong, J. Tang, X. Li, B. Gao, H. Qian, and H. Wu, "Dynamic memristor-based reservoir computing for high-efficiency temporal signal processing," *Nature Communications*, vol. 12, no. 1, pp. 1–9, 2021.

[64] J. Moon *et al.*, "Temporal data classification and forecasting using a memristor-based reservoir computing system," *Nature Electronics*, vol. 2, no. 10, pp. 480–487, 2019.

[65] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu, "Reservoir computing using dynamic memristors for temporal information processing," *Nature Communications*, vol. 8, no. 1, pp. 1–10, 2017.

[66] L. Sun *et al.*, "In-sensor reservoir computing for language learning via two-dimensional memristors," *Science Advances*, vol. 7, no. 20, eabg1455, May 2021.

[67] J. Yang, H. Cho, H. Ryu, M. Ismail, C. Mahata, and S. Kim, "Tunable Synaptic Characteristics of a Ti/TiO2/Si Memory Device for Reservoir Computing," *ACS Applied Materials and Interfaces*, vol. 13, no. 28, pp. 33 244–33 252, 2021.

[68] T. Wang, H.-M. Huang, X.-X. Wang, and X. Guo, "An artificial olfactory inference system based on memristive devices," *InfoMat*, vol. 3, no. 7, pp. 804–813, 2021.

[69] Y. Deng and Y. Li, "A 2D Hyperchaotic Discrete Memristive Map and Application in Reservoir Computing," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. XX, no. 19, pp. 1–1, 2021.

[70] N. Lyapunov *et al.*, "A Bifunctional Memristor Enables Multiple Neuromorphic Computing Applications," *Advanced Electronic Materials*, vol. 2101235, pp. 1–7, 2022.

[71] F. Duport, B. Schneider, A. Smerieri, M. Haelterman, and S. Massar, "All-optical reservoir computing," *Optics Express*, vol. 20, no. 20, pp. 1958–1964, 2012.

[72] F. Duport, A. Smerieri, A. Akrout, M. Haelterman, and S. Massar, "Fully analogue photonic reservoir computer," *Scientific Reports*, vol. 6, no. October 2015, pp. 1–12, 2016.

[73] L. Larger, A. Baylón-Fuentes, R. Martinenghi, V. S. Udaltsov, Y. K. Chembo, and M. Jacquot, "High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification," *Physical Review X*, vol. 7, no. 1, pp. 1–14, 2017.

[74] Y. Kuriki, J. Nakayama, K. Takano, and A. Uchida, "Impact of input mask signals on delay-based photonic reservoir computing with semiconductor lasers," *Optics Express*, vol. 26, no. 5, p. 5777, 2018.

[75] Y. Paquot *et al.*, "Optoelectronic reservoir computing," *Scientific Reports*, vol. 2, no. 1, p. 287, 2012.

[76] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nature Communications*, vol. 4, 2013.

[77] H. Tanaka *et al.*, "A molecular neuromorphic network device consisting of single-walled carbon nanotubes complexed with polyoxometalate," *Nature Communications*, vol. 9, no. 1, pp. 1–7, 2018.

[78] K. Fu *et al.*, "Reservoir Computing with Neuromemristive Nanowire Networks," *Proceedings of the International Joint Conference on Neural Networks*, pp. 1–8, 2020.

[79] R. Zhu *et al.*, "Information dynamics in neuromorphic nanowire networks," *Scientific Reports*, vol. 11, no. 1, pp. 1–15, 2021.

[80] G. Milano *et al.*, "In materia reservoir computing with a fully memristive architecture based on self-organizing nanowire networks," *Nature Materials*, 2021.

[81] M. Dale, J. F. Miller, S. Stepney, and M. A. Trefzer, "Evolving carbon nanotube reservoir computers," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9726, no. January 2018, pp. 49–61, 2016.

[82] J. Hochstetter, R. Zhu, A. Loeffler, A. Diaz-Alvarez, T. Nakayama, and Z. Kuncic, "Avalanches and edge-of-chaos learning in neuromorphic nanowire networks," *Nature Communications*, vol. 12, no. 1, 2021.

[83] T. Zheng *et al.*, "Enhancing performance of reservoir computing system based on coupled mems resonators," *Sensors*, vol. 21, no. 9, pp. 1–17, 2021.

[84] B. Barazani, G. Dion, J. F. Morissette, L. Beaudoin, and J. Sylvestre, "Microfabricated Neuroaccelerometer: Integrating Sensing and Reservoir Computing in MEMS," *Journal of Microelectromechanical Systems*, vol. 29, no. 3, pp. 338–347, 2020.

[85] J. Sun *et al.*, "Novel nondelay-based reservoir computing with a single micromechanical nonlinear resonator for high-efficiency information processing," *Microsystems and Nanoengineering*, vol. 7, no. 1, 2021.

[86] K. Nakajima, H. Hauser, R. Kang, E. Guglielmino, D. G. Caldwell, and R. Pfeifer, "A soft body as a reservoir: Case studies in a dynamic model of octopus-inspired soft robotic arm," *Frontiers in Computational Neuroscience*, vol. 7, no. JUN, pp. 1–19, 2013.

[87] P. Aaser *et al.*, "Towards making a cyborg: A closed-loop reservoir-neuro system," in *Proceedings of the European Conference on Artificial Life*, September, MIT Press, 2017, pp. 430–437.

[88] T. Lymburn, S. D. Algar, M. Small, and T. Jüngling, "Reservoir computing with swarms," *Chaos*, vol. 31, no. 3, 2021.

[89] J. Cao *et al.*, "Emerging dynamic memristors for neuromorphic reservoir computing," *Nanoscale*, vol. 14, no. 2, pp. 289–298, 2022.

[90] S. Lilak *et al.*, "Spoken Digit Classification by In-Materio Reservoir Computing With Neuromorphic Atomic Switch Networks," *Frontiers in Nanotechnology*, vol. 3, no. May, pp. 1–11, 2021.

[91] E. C. Demis, R. Aguilera, K. Scharnhorst, M. Aono, A. Z. Stieg, and J. K. Gimzewski, "Nanoarchitectonic atomic switch networks for unconventional computing," *Japanese Journal of Applied Physics*, vol. 55, no. 11, 2016.

[92] Y. Usami *et al.*, "In-Materio Reservoir Computing in a Sulfonated Polyaniline Network," *Advanced Materials*, vol. 2102688, pp. 1–9, 2021.

[93] M. Cucchi *et al.*, "Reservoir computing with biocompatible organic electrochemical networks for brain-inspired biosignal classification," *Science Advances*, vol. 7, no. 34, pp. 1–9, 2021.

[94] J. Lao, M. Yan, B. Tian, C. Jiang, C. Luo, and Z. Xie, "Ultralow-Power Machine Vision with Self-Powered Sensor Reservoir," *Advanced Science*, vol. 2106092, pp. 1–11, 2022.

[95] J. Yu *et al.*, "Energy efficient and robust reservoir computing system using ultrathin (3.5 nm) ferroelectric tunneling junctions for temporal data learning," *2021 Symposium on VLSI Technology*, vol. 2, no. 10, pp. 16–4, 2021.

[96] P. Yao *et al.*, "Face classification using electronic synapses," *Nature Communications*, vol. 8, no. May, pp. 1–8, 2017.

[97] Z. Liu *et al.*, "Multichannel parallel processing of neural signals in memristor arrays," *Science Advances*, vol. 6, no. 41, pp. 2–10, 2020.

[98] A. Serb, J. Bill, A. Khiat, R. Berdan, R. Legenstein, and T. Prodromakis, "Unsupervised learning in probabilistic neural networks with multi-state metal-oxide memristive synapses," *Nature Communications*, vol. 7, 2016.

[99] A. Serb *et al.*, "Memristive synapses connect brain and silicon spiking neurons," *Scientific Reports*, vol. 10, no. 1, pp. 1–7, 2020.

[100] X. Liang, R. Ghannam, and H. Heidari, "Wrist-Worn Gesture Sensing with Wearable Intelligence," *IEEE Sensors Journal*, vol. 19, no. 3, pp. 1082–1090, 2019.

[101] A. Tanwear *et al.*, "Spintronic Sensors Based on Magnetic Tunnel Junctions for Wireless Eye Movement Gesture Control," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 14, no. 6, pp. 1299–1310, 2020.

[102] W. Gao *et al.*, "Fully integrated wearable sensor arrays for multiplexed in situ perspiration analysis," *Nature*, vol. 529, no. 7587, pp. 509–514, 2016.

[103] A. López, M. Fernández, H. Rodríguez, F. Ferrero, and O. Postolache, "Development of an EOG-based system to control a serious game," *Measurement: Journal of the International Measurement Confederation*, vol. 127, no. April, pp. 481–488, 2018.

[104] M. Witkowski, M. Cortese, M. Cempini, J. Mellinger, N. Vitiello, and S. R. Soekadar, "Enhancing brain-machine interface (BMI) control of a hand exoskeleton using electrooculography (EOG)," *Journal of NeuroEngineering and Rehabilitation*, vol. 11, no. 1, pp. 1–6, 2014.

[105] Z. Wu, X. Ding, and G. Zhang, "A novel method for classification of ECG arrhythmias using deep belief networks," *International Journal of Computational Intelligence and Applications*, vol. 15, no. 4, pp. 1–14, 2016.

[106] S. He, C. Yang, M. Wang, L. Cheng, and Z. Hu, "Hand gesture recognition using MYO armband," *Proceedings - 2017 Chinese Automation Congress, CAC 2017*, vol. 2017-Janua, pp. 4850–4855, 2017.

[107] X. Liang, H. Heidari, and R. Dahiya, "Wearable capacitive-based wrist-worn gesture sensing system," in *Proceedings - 2017 1st New Generation of CAS, NGCAS 2017*, 2017.

[108] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, vol. 40, no. 1, pp. 1–12, 2010.

[109] F. Zhou and Y. Chai, "Near-sensor and in-sensor computing," *Nature Electronics*, vol. 3, no. 11, pp. 664–671, 2020.

[110] X. Liang, H. Li, A. Vuckovic, J. R. Mercer, and H. Heidari, "A Neuromorphic Model with Delay-based Reservoir for Continuous Ventricular Heartbeat Detection," *IEEE Transactions on Biomedical Engineering*, vol. 69, no. 6, pp. 1–12, 2022.

[111] Y. Chen, X. Liang, M. Assaad, and H. Heidari, "Wearable Resistive-based Gesture-Sensing Interface Bracelet," *2019 UK/China Emerging Technologies, UCET 2019*, pp. 14–17, 2019.

[112] Y. Liu, S. Zuo, X. Liang, H. Khanbareh, H. Heidari, and R. Ghannam, "Gesture recognition wristband device with optimised piezoelectric energy harvesters," *ICECS 2020 - 27th IEEE International Conference on Electronics, Circuits and Systems, Proceedings*, pp. 2020–2023, 2020.

[113] H. Li *et al.*, "Hierarchical Sensor Fusion for Micro-Gesture Recognition with Pressure Sensor Array and Radar," *IEEE Journal of Electromagnetics, RF and Microwaves in Medicine and Biology*, vol. 4, no. 3, pp. 225–232, 2020.

[114] World Health Organization, *Cardiovascular diseases (cvds)*, 2017.

[115] E. J. d. S. Luz, W. R. Schwartz, G. Cámara-Chávez, and D. Menotti, "ECG-based heartbeat classification for arrhythmia detection: A survey," *Computer Methods and Programs in Biomedicine*, vol. 127, pp. 144–164, 2016.

[116] G. B. Moody and R. G. Mark, "The impact of the mit-bih arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[117] G. Garcia, G. Moreira, D. Menotti, and E. Luz, "Inter-Patient ECG Heartbeat Classification with Temporal VCG Optimized by PSO," *Scientific Reports*, vol. 7, no. 1, pp. 1–11, 2017.

[118] C. L. Herry, M. Frasch, A. J. Seely, and H. T. Wu, "Heart beat classification from single-lead ECG using the synchrosqueezing transform," *Physiological Measurement*, vol. 38, no. 2, pp. 171–187, 2017.

[119] S. Raj, K. C. Ray, and O. Shankar, "Cardiac arrhythmia beat classification using DOST and PSO tuned SVM," *Computer Methods and Programs in Biomedicine*, vol. 136, pp. 163–177, 2016.

[120] R. Ghorbani Afkhami, G. Azarnia, and M. A. Tinati, "Cardiac arrhythmia classification using statistical and mixture modeling features of ECG signals," *Pattern Recognition Letters*, vol. 70, pp. 45–51, 2016.

[121] M. M. Rahhal, Y. Bazi, H. Alhichri, N. Alajlan, F. Melgani, and R. R. Yager, "Deep learning approach for active classification of electrocardiogram signals," *Information Sciences*, vol. 345, pp. 340–354, 2016.

[122] S. Ortín and L. Pesquera, "Reservoir Computing with an Ensemble of Time-Delay Reservoirs," *Cognitive Computation*, vol. 9, no. 3, pp. 327–336, 2017.

[123] M. C. Soriano *et al.*, "Delay-based reservoir computing: Noise effects in a combined analog and digital implementation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 2, pp. 388–393, 2015.

[124] J. K. Lau *et al.*, "Iphone ecg application for community screening to detect silent atrial fibrillation: A novel technology to prevent stroke," *International Journal of Cardiology*, vol. 165, no. 1, pp. 193–194, 2013.

[125] S. Bains, "The business of building brains," *Nature Electronics*, vol. 3, no. 7, pp. 348–351, 2020.

[126] P. De Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features," *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 7, pp. 1196–1206, 2004.

[127] ANSI/AAMI, "Testing and reporting performance results of cardiac rhythm and ST segment measurement algorithms," *ANSI/AAMI EC38*, vol. 1998, 1998.

[128] L. Appeltant, G. Van Der Sande, J. Danckaert, and I. Fischer, "Constructing optimized binary masks for reservoir computing with delay systems," *Scientific Reports*, vol. 4, pp. 1–5, 2014.

[129] R. Tibshirani, "Regression Shrinkage and Selection Via the Lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[130] V. Ceperic and A. Baric, "Reducing complexity of echo state networks with sparse linear regression algorithms," *Proceedings - UKSim-AMSS 16th International Conference on Computer Modelling and Simulation, UKSim 2014*, pp. 26–31, 2014.

[131] M. Inubushi and K. Yoshimura, "Reservoir Computing beyond Memory-Nonlinearity Trade-off," *Scientific Reports*, vol. 7, no. 1, pp. 1–10, 2017.

[132] L. Chua, "Memristor-The missing circuit element," *IEEE Transactions on Circuit Theory*, vol. 18, no. 5, pp. 507–519, 1971.

[133] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[134] T. Prodromakis, C. Toumazou, and L. Chua, "Two centuries of memristors," *Nature Materials*, vol. 11, no. 6, pp. 478–481, 2012.

[135] X. Li *et al.*, "Power-efficient neural network with artificial dendrites," *Nature Nanotechnology*, vol. 15, no. 9, pp. 776–782, 2020.

[136] S. Stathopoulos *et al.*, "Multibit memory operation of metal-oxide Bi-layer memristors," *Scientific Reports*, vol. 7, no. 1, pp. 1–7, 2017.

[137] D. Vaidya *et al.*, "Compact Modeling of the Switching Dynamics and Temperature Dependencies in TiO-Based Memristors-Part I: Behavioral Model," *IEEE Transactions on Electron Devices*, vol. 68, no. 10, pp. 4877–4884, 2021.

[138] E. Covi, S. Brivio, A. Serb, T. Prodromakis, M. Fanciulli, and S. Spiga, "Analog memristive synapse in spiking networks implementing unsupervised learning," *Frontiers in Neuroscience*, vol. 10, no. OCT, pp. 1–13, 2016.

[139] T. Serrano-Gotarredona, T. Masquelier, T. Prodromakis, G. Indiveri, and B. Linares-Barranco, "STDP and STDP variations with memristors for spiking neuromorphic learning systems," *Frontiers in Neuroscience*, vol. 7, no. 7 FEB, pp. 1–15, 2013.

[140] S. Sayyaparaju, M. S. Ara Shawkat, and G. S. Rose, "Robust implementation of memristive reservoir computing with crossbar based readout layer," *Proceedings of the 2020 IEEE Dallas Circuits and Systems Conference, DCAS 2020*, no. 1, pp. 31–34, 2020.

[141] A. Serb and T. Prodromakis, "An Analogue-Domain, Switch-Capacitor-Based Arithmetic-Logic Unit," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[142] J. Prinzie, F. M. Simanjuntak, P. Leroux, and T. Prodromakis, "Low-power electronic technologies for harsh radiation environments," *Nature Electronics*, vol. 4, no. 4, pp. 243–253, 2021.

[143] A. Serb, A. Khiat, and T. Prodromakis, "Seamlessly fused digital-analogue reconfigurable computing using memristors," *Nature Communications*, vol. 9, no. 1, 2018.

[144] Y. H. Jang *et al.*, "Time-varying data processing with nonvolatile memristor-based temporal kernel," *Nature Communications*, vol. 12, no. 1, pp. 1–9, 2021.

[145] R. Hu *et al.*, "Investigation of Resistive Switching Mechanisms in Ti/TiOx/Pd-Based RRAM Devices," *Advanced Electronic Materials*, vol. 2100827, pp. 1–7, 2021.

[146] M. Benedicks and L. Carleson, "The dynamics of the henon map," *Annals of Mathematics*, vol. 133, no. 1, pp. 73–169, 1991.

[147] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[148] A. Ignatov, "Real-time human activity recognition from accelerometer data using Convolutional Neural Networks," *Applied Soft Computing Journal*, vol. 62, pp. 915–922, 2018.

[149] S. Ortín *et al.*, "A Unified Framework for Reservoir Computing and Extreme Learning Machines based on a Single Time-delayed Neuron," *Scientific Reports*, vol. 5, no. October, pp. 1–11, 2015.

[150] Z. Liu *et al.*, "Neural signal analysis with memristor arrays towards high-efficiency brain–machine interfaces," *Nature Communications*, vol. 11, no. 1, pp. 1–9, 2020.

[151] H. Wu *et al.*, "Device and circuit optimization of RRAM for neuromorphic computing," *Technical Digest - International Electron Devices Meeting, IEDM*, pp. 1–11, 2018.

[152] V. Joshi *et al.*, "Accurate deep neural network inference using computational phase-change memory," *Nature Communications*, vol. 11, no. 1, pp. 1–13, 2020.

[153] S. Kariyappa *et al.*, "Noise-Resilient DNN: Tolerating Noise in PCM-Based AI Accelerators via Noise-Aware Training," *IEEE Transactions on Electron Devices*, vol. 68, no. 9, pp. 4356–4362, 2021.

[154] M. L. Alomar *et al.*, "Efficient parallel implementation of reservoir computing systems," *Neural Computing and Applications*, vol. 32, no. 7, pp. 2299–2313, 2020.

[155] M. L. Alomar *et al.*, "Digital Implementation of a Single Dynamical Node Reservoir Computer," *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: EXPRESS BRIEFS*, vol. 62, no. 10, pp. 977–981, 2015.

[156] D. Kleyko, E. P. Frady, M. Kheffache, and E. Osipov, "Integer Echo State Networks: Efficient Reservoir Computing for Digital Hardware," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2020.