Temporal Data Analysis Using Reservoir Computing and Dynamic Memristors

by

John Moon

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering)
in the University of Michigan
2021

Doctoral Committee:

Professor Wei D. Lu, Chair
Assistant Professor Reetuparna Das
Professor L. Jay Guo
Associate Professor Zhengya Zhang

John Moon

moonjohn@umich.edu

ORCID iD: 0000-0002-6516-2700

Dedication

To the almighty God, my wife Seoyoung, my family and my friends.

Acknowledgements

I would like to express my sincere appreciation for all supports I have been given during my Ph.D. study. First, I would like to thank my advisor, Prof. Wei D. Lu for giving me valuable comments and constant support. He has always guided me with his passion and immerse knowledge. This work would not have been possible without his patience, encouragement, and support.

I would also like to thank my committee members, Prof. Reetuparna Das, Prof. L. Jay Guo, and Prof. Zhengya Zhang for their invaluable discussions and critical comments on my works. It was a great opportunity for me to enrich my works in interdisciplinary points of view.

My sincere thanks go to my wife, family, and friends for supporting me through my Ph.D. study. I would like to express special thanks to my wife Seoyoung Lim for unconditional love, endless support, and encouragement.

I would like to appreciate my past and current group members for their useful discussion. I especially thank Dr. Yeonjoo Jeong, Dr. Jonghoon Shin, and Dr. Seunghwan Lee, who gave me a lot of invaluable advices in academic work and personal life. Thank Dr. Mohammed Zidan, Dr. Xiaojian Zhu, Dr. Chao Du, Dr. Fuxi Cai, Dr. Wen Ma, Dr. Jihang Lee, Qiwen Wang, Fan-Hsuan Meng, Xinxin Wang, Yuting Wu, Yongmo Park, Ziyu Wang, and Sangmin Yoo for friendship and support.

I would also like to thank the Lurie Nanofabrication Facility (LNF) staff: Sandrine Martin, Matthew Oonk, Vishva Ray, Gregory Allion, David Sebastian, and Anthony Sebastian for their technical support.

Table of Contents

List of Tables

List of Figures

Abstract

Temporal data analysis including classification and forecasting is essential in a range of fields from finance to engineering. While static data are largely independent of each other, temporal data have a considerable correlation between the samples, which is important for temporal data analysis. When models and parameters used to describe a temporal dynamic system are complex, or even unknown, it becomes very difficult to analyze the system accurately using statistics-based techniques, which require knowledge on the possible configurations prior to the actual data analysis. Neural networks thus offer a more general and flexible approach since they do not depend on parameters of specific tasks but are driven only by the data. In particular, recurrent neural networks have gathered much attention since the temporal information captured by the recurrent connections improves the prediction performance. Recently, reservoir computing (RC), which evolves from recurrent neural networks, has been extensively studied for temporal data analysis as it can offer efficient temporal processing of recurrent neural networks with a low training cost.

This dissertation presents hardware implementation of the RC system using an emerging device - memristor, followed by a theoretical study on hierarchical architectures of the RC system.

A RC hardware system based on dynamic tungsten oxide (WO_x) memristors is first demonstrated. The internal short-term memory effects of the WO_x memristors allow the memristor-based reservoir to nonlinearly map temporal inputs into reservoir states, where the projected features can be readily processed by a simple linear readout function. We use the

system to experimentally demonstrate two standard benchmarking tasks: isolated spoken digit recognition with partial inputs and chaotic system forecasting. High classification accuracy of 99.2% is obtained for spoken digit recognition and autonomous chaotic time series forecasting has been demonstrated over the long term.

We then investigate the influence of the hierarchical reservoir structure on the properties of the reservoir and the performance of the RC system. Analogous to deep neural networks, stacking sub-reservoirs in series is an efficient way to enhance the nonlinearity of data transformation to high-dimensional space and expand the diversity of temporal information captured by the reservoir. These deep reservoir systems offer better performance when compared to simply increasing the size of the reservoir or the number of sub-reservoirs. Low-frequency components are mainly captured by the sub-reservoirs in the later stages of the deep reservoir structure, similar to observations that more abstract information can be extracted by layers in the late stage of deep neural networks. When the total size of the reservoir is fixed, the tradeoff between the number of sub-reservoirs and the size of each sub-reservoir needs to be carefully considered, due to the degraded ability of the individual sub-reservoirs at small sizes. Improved performance of the deep reservoir structure alleviates the difficulty of implementing the RC system on hardware systems.

Beyond temporal data classification and prediction, one of the interesting applications of temporal data analysis is inferring the neural connectivity patterns from the high-dimensional neural activity recording data. By computing the temporal correlation between the neural spikes, connections between the neurons can be inferred using statistics-based techniques, but it becomes increasingly computationally expensive for large scale neural systems. We propose a second-order memristor-based hardware system using the natively implemented spike-timing-

dependent plasticity learning rule for neural connectivity inference. By incorporating biological features such as transmission delay to the neural networks, the proposed concept not only correctly infers the direct connections but also distinguishes direct connections from indirect connections. Effects of additional biophysical properties not considered in the simulation and challenges of experimental memristor implementation will be also discussed.

Chapter 1 Introduction

1.1 Background

The availability of large amounts of datasets is one of the major factors that led machine learning [1] to successfully achieve human-level performance for tasks such as image classification [2] and speech recognition [3]. As the performance of models trained by machine learning algorithms is usually affected by the size of the dataset, a larger number of training samples generally improves the trained model performance. One of the popular tasks is image classification that aims to capture the spatial information in the input image to recognize what object is in the image. Since individual input images are independent of each other, image datasets such as ImageNet are static, which do not have a time axis and there is no temporal correlation between the image samples.

With the impressive advance in static data analysis, temporal data analysis has also been intensively studied to deal with the temporal data, which has an additional data space axis - time. In contrast to the independence of the samples of the static datasets, data points of the temporal datasets such as time-series are highly correlated. Thus, capturing the correlation between data points is the most important function for temporal data analysis. Traditionally, statistics-based techniques such as autoregressive integrated moving average models [4] have been used to analyze simple temporal systems, which can be described by the combination of autoregressive, differencing, and moving-average models. When the temporal dynamics of the temporal dataset is simple enough to be described by a small set of parameters, they can produce reasonably accurate results. However, as the temporal systems become more complex, it becomes

increasingly difficult to achieve accurate prediction results from the statistics-based techniques. Since neural networks are trained by the data rather than relying on human insight, neural networks have recently been extensively studied as a potentially more general and flexible tool to analyze time series data.

Particularly, the advances in computer hardware and algorithm developments have allowed recurrent neural networks (RNNs) [5] to model complex temporal dynamics. However, despite the success of gradient-based learning rules for static datasets, applying similar rules to RNNs is challenging because the vanishing gradient or exploding gradient problems that make it difficult to find optimal weights [6]. To address these issues, advanced models such as long short-term memory (LSTM) [7] and reservoir computing (RC) [8, 9] have been developed and been shown to effectively capture the complex temporal dynamic systems. LSTM deals with the vanishing gradient or exploding gradient problems by enforcing constant error flow through internal states of the LSTM units, but still requires significant computation costs to train the weight matrices. RC efficiently reduces the training cost and mitigates the difficulty of the training process of RNNs by allowing only the linear output layer to be tuned instead of training every weight in the recurrent connections.

1.2 Reservoir Computing

Reservoir computing [8, 9] is an emerging computing concept in machine learning, which has shown excellent ability to process temporal data with an efficient training process. As the diverse connections in a reservoir computing system can effectively capture the temporal dynamics, RC systems are good at performing tasks such as chaotic time series prediction or speech recognition. The underlying mechanism of reservoir computing is inspired by how the brain processes input sensory signals. The input sensory signals potentiate (*i.e.* excite) neurons in

the biological neural system and produce neuron firings when the neuronal membrane potential exceeds the threshold. The input information is thus mapped into the neuron firing patterns, which are then further processed in the subsequent biological neural networks. Similarly, in reservoir computing, the input signals excite the reservoir composed of nonlinear nodes. The excited reservoir states help decode the temporal information and can in turn be efficiently analyzed by the subsequent trained readout layer. It is noteworthy that an RC system can be separated into two parts: the first part (which does not require training) for capturing the temporal information and generating the excited patterns, the second part (which will be trained) for analyzing the generated patterns and deriving the desired output.

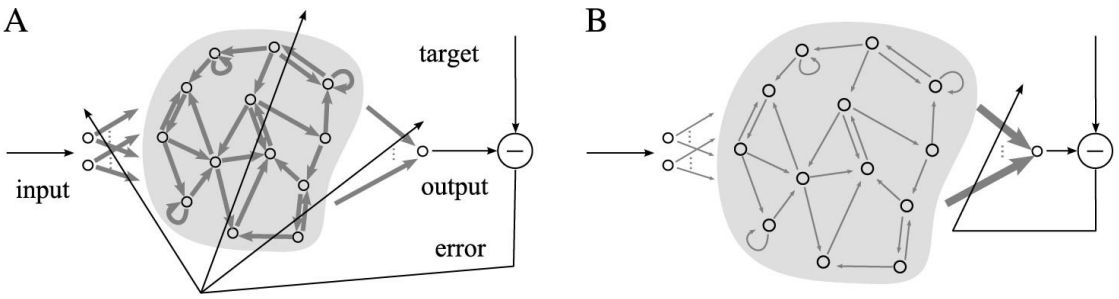

Figure 1-1 Schematics of (A) recurrent neural network and (B) reservoir computing system. *Image adapted from Reference [9].* Image credit: Lukoševičius, M.

As shown in Figure 1-1, a conventional RC system consists of three basic components: an input layer, a recurrent layer (*i.e.* the reservoir), and an output layer (called the readout layer). The input layer feeds the input signals to the reservoir through randomly generated connections. The reservoir is randomly initialized, and its connections remain unchanged during the training process. The nodes in the reservoir have nonlinear activation functions, and offer diverse connections including feedforward and recurrent connections to effectively convey the temporal information in the reservoir. The role of the reservoir is to nonlinearly map the temporal inputs into a high-dimensional feature space, represented by the excited states of the nodes in the reservoir, such that features that may not be linearly separable in the original input space can

become linearly separable after this nonlinear mapping. The output layer is tuned during the training process so that it can derive the desired output from a linear combination from the excited reservoir nodes.

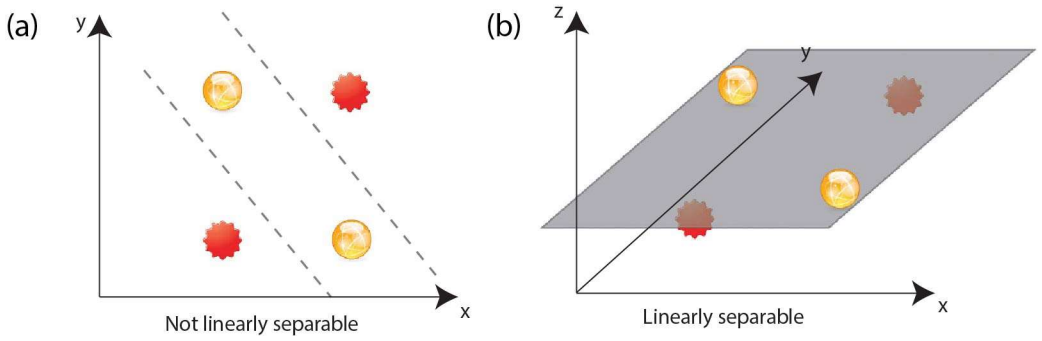

Figure 1-2 Illustration of the working principle of reservoir computing.
(a) In two-dimensional space, the spheres and the stars cannot be separated with a single straight line. (b) With a nonlinear mapping into a higher dimension (*i.e.* three-dimensional) space, the spheres and the stars can be separated by a single linear hyperplane.
Image adapted from Reference [10]. Image credit: Appeltant, L.

The working principle of reservoir computing can be illustrated as in Figure 1-2. Assume there are two classes: yellow sphere and red star. Before the input signals are applied to the reservoir, the input data in the original two-dimensional space cannot be linearly separated, *i.e.* with a single straight line as shown in Figure 1-2 (a). When the data are nonlinearly mapped onto a higher dimension, *i.e.* three-dimensional feature space, it becomes possible to introduce a two-dimensional plane to linearly separate the yellow spheres from the red stars, as shown in Figure 1-2 (b). In RC, the reservoir efficiently performs the nonlinear mapping from a low dimensional space to a high dimensional space, and the readout layer analyzes the data in the high dimensional space by drawing the hyperplane for separating different classes. Since only the simple readout layer needs to be trained, training cost of the reservoir computing system can be effectively reduced compared to the conventional neural networks, which need to train every connection using the gradient-based learning rule.

The most important part of an RC system is the reservoir since the performance of reservoir computing is mainly determined by the quality of the reservoir. The reservoir should have a high degree of nonlinear transformation and a high dimension of feature space. Typically, a nonlinear activation function such as the hyperbolic tangent function is used to perform the nonlinear transformation in the reservoir. Since the dimension of the feature space corresponds to the number of nodes in the reservoir, the high dimension of the feature space can be achieved by increasing the number of nodes in the reservoir. In addition, a fading memory [9] is another crucial property of the reservoir to process temporal data. The fading memory property means that the reservoir state is dependent on inputs from the recent past, but is independent of the inputs from the far past. With the fading memory property, the reservoir state returns to idle states when there is no input for a while, and expresses the temporal patterns governed by only the inputs from the recent past.

1.3 Memristor

A memristor is a two-terminal electrical component that has internal states, normally reflected by its resistance or conductance, which are in turn affected by the history of external stimulations such as applied voltage and current. The concept of memristor was initially proposed [11] in the 1970s to complete a theoretical picture of fundamental electrical components including the resistor, capacitor, and inductor, as shown in Figure 1-3. The theoretical memristor stands for a fundamental nonlinear electrical element linking magnetic flux and charge. As the magnetic flux and the charge can be regarded as the time integral of the voltage and the time integral of the current, respectively, the memristor shows the dynamic relationship between the voltage and current and reflects the importance of temporal history of the inputs.

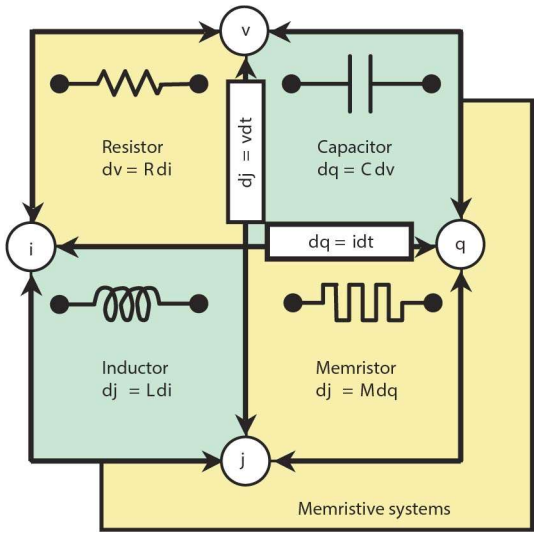

Figure 1-3 The four fundamental two-terminal circuit elements.
The fourth element, the memristor with memristance M, defined as the rate of change of flux with charge.
Image adapted from Reference [12]. Image credit: Dr. Dmitri B. Strukov.

A signature characteristic of memristors is the pinched hysteresis loop [13], as shown in Figure 1-4 (a). As the state of memristors depends on the history of external stimulations, the conductance of memristors gradually increases or decreases when the positive or negative voltage is applied to the memristors, respectively. Moreover, as shown in Figure 1-4 (b), consecutive voltage pulses can incrementally modulate the conductance of memristors, which allows them to mimic the potentiation or depression of synapses in neural systems. In addition to analog conductance modulation, memristor devices can offer a large on/off resistance ratio, a fast resistive switching process, and a high storage density system when incorporated in the crossbar structure. These properties make the memristor an attractive candidate for memory applications [14] as well as neuromorphic and in-memory computing systems [15].

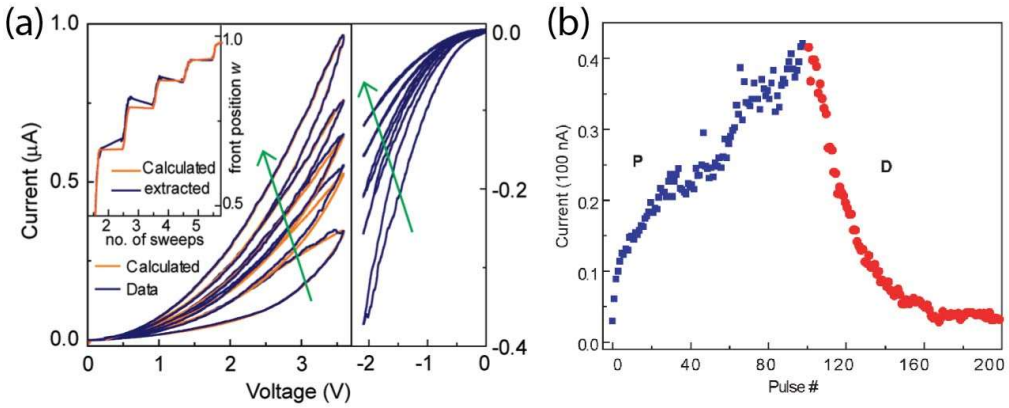

Figure 1-4 Memristor characteristics.
(a) Pinched hysteresis loop in current-voltage characteristic, (b) Memristor response to programming pulses.
Image adapted from Reference [13]. Image credit: Dr. Sung Hyun Jo.

A memristor is typically built with a metal-insulator-metal configuration, where the conductance is determined by the conductive filament growth/dissolution in the insulator. Depending on the filament material, memristors can be roughly categorized into electrochemical metallization memory (ECM) [16, 17] and valence change memory (VCM) types [18, 19]. The resistive switching of ECM is governed by the electrochemical growth/dissolution of metal (*e.g.* Ag and Cu) filaments within the insulating layer. ECMs typically have a high on/off ratio and low switching energy as the metal atoms introduced to the insulating layer have a large contrast in conductivity with the insulating layer, and no chemical reaction takes place with the insulating layer during resistive switching. However, since the introduction and removal of metal atoms can lead to physical damages to the insulating layer over time, ECMs typically have limited write/erase endurance, which makes it difficult to build systems requiring frequent weight updates. In contrast, the resistive switching mechanism of VCM is based on the redistribution of oxygen vacancies (V_Os) in the insulating layer, usually composed of a transition metal oxide. The external voltage applied to the memristor induces the migration of V_Os so that the local stoichiometry of the transition metal oxide changes. As the V_O-rich region has a higher conductivity than the V_O-poor region, growth and shrinkage of the V_O-rich region leads to

conductance changes in VCM cells. As the conductivity difference between the V_O-rich region and the V_O-poor region is not as significant as the one between metal atom and the insulating layer in ECM, VCMs typically have a smaller on/off ratio and a leakier high-resistance state than ECM. However, modulation of the stoichiometry of the transition metal oxide is less likely to cause physical damage to the switching layer, and is also more favorable to produce multiple conductance levels through gradual tuning of the V_O -rich region shape. The long write/erase endurance and multi-level conductance thus make VCM cells attractive candidates for neuromorphic computing applications which require frequent conductance updates and analog conductance modulations.

VCM-type memristors have been extensively studied and recently implemented as machine learning (ML) hardware accelerators [15, 20]. Thanks to the excellent device properties and the high-density crossbar structure, such systems can implement efficient vector-matrix multiplication computations in parallel and thus significantly improve the throughput and energy efficiency of ML systems.

Additionally, volatile memristors (*e.g.* tungsten-oxide based memristor) which offer short-term memory behaviors due to spontaneous diffusion of the ionic species that form the conduction channel have also been studied, particularly to utilize the volatile device characteristics for temporal data analysis [21-23]. The conductance of volatile memristors gradually decays to the initial condition when there is no external stimulus. This decaying characteristic can help the volatile memristors efficiently encode temporal information in time-series input signals. For example, WO_x memristors have been used to form reservoirs in reservoir computing (RC) systems, where the short-term memory effect of WO_x memristors allows the devices to natively implement the fading memory property of a reservoir to process temporal

inputs. Thanks to the diverse temporal dynamics captured by the memristor-based reservoir, the temporal data can be analyzed by a simple readout network (e.g. a perceptron network) [21-23], as will be discussed in more detail in Chapters 2, 3.

1.4 Organization of Dissertation

This thesis discusses several studies on memristor-based hardware systems for temporal data analysis.

In Chapter 1, we introduced the background information on temporal data analysis, reservoir computing, and the memristor concept.

Chapter 2 discusses a memristor-based reservoir computing system to perform temporal data classification and forecasting tasks. The device characteristics of WO_x memristors, and simulation and experimental results of standard benchmarking tasks for temporal data analysis will be presented.

Chapter 3 discusses the effects of hierarchical structures on the properties of the reservoir and the performance of the RC system. Tradeoffs between the number of sub-reservoirs and the size of each sub-reservoir will be discussed.

Chapter 4 discusses the inference of neural connectivity patterns from neural spike trains using bio-inspired learning rule, *i.e.* spike-timing- dependent plasticity, which can be natively in second-order memristors. Performance comparison with statistic-based inference methods and the effects of transmission delay mismatch will be discussed.

Chapter 5 discusses future works related to the neural connectivity inference task. Biophysical effects and physical device properties, which should be carefully considered to improve the inference performance, will be discussed.

References

1. LeCun, Y., Bengio, Y., & Hinton, G. Deep learning. *Nature* 521, 436-444 (2015).
2. He, K., Zhang, X., Ren, S. & Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *Proc. IEEE Int. Conf. Comput. Vis.* 1026–1034 (2015). doi:10.1109/ICCV.2015.123
3. Hinton, G. *et al.* Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Process. Mag.* 29, 82–97 (2012).
4. Box, G. E., & Pierce, D. A. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *J.Am. Stat. Assoc.* 65, 1509-1526 (1970).
5. Mandic, D. P., & Chambers, J. A. Recurrent Neural Networks for Prediction: Learning Algorithms Architectures and Stability, New York:Wiley (2001).
6. Bengio, Y., Simard, P., & Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* 5, 157-166 (1994).
7. Hochreiter, S., & Schmidhuber, J. Long short-term memory. *Neural computation* 9, 1735-1780 (1997).
8. Jaeger, H., & Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science* 304, 78-80 (2004).
9. Lukoševičius, M. & Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* 3, 127–149 (2009).
10. Appeltant, L. *et al.* Information processing using a single dynamical node as complex system. *Nat. Commun.* 2, 468 (2011).
11. Chua, L. O. Memristor - the missing circuit element. *IEEE Trans. Circuit Theory* 18, 507–519 (1971).

12. Strukov, D. B., Snider, G. S., Stewart, D. R. & Williams, R. S. The missing memristor found. *Nature* 453, 80–83 (2008).

13. Jo, S. H. *et al.* Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Lett.* 10, 1297–1301 (2010).

14. Prezioso, M. *et al.* Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* 521, 61–64 (2015).

15. Cai, F. *et al.* A fully integrated reprogrammable memristor–CMOS system for efficient multiply–accumulate operations. *Nat. Electron.* 2, 290–299 (2019).

16. Valov, I., Waser, R., Jameson, J. R. & Kozicki, M. N. Electrochemical metallization memories - Fundamentals, applications, prospects. *Nanotechnology* 22, 254003 (2011).

17. Yang, Y. *et al.* Observation of conducting filament growth in nanoscale resistive memories. *Nat. Commun.* 3, 1-8 (2012).

18. Yang, J. J., Strukov, D. B. & Stewart, D. R. Memristive devices for computing. *Nat. Nanotechnol.* 8, 13–24 (2013).

19. Park, G. S. *et al.* In situ observation of filamentary conducting channels in an asymmetric Ta2O5-x/TaO2-x bilayer structure. *Nat. Commun.* 4, 1–9 (2013).

20. Sheridan, P. M. *et al.* Sparse coding with memristor networks. *Nat. Nanotechnol.* 12, 784-789 (2017).

21. Du, C. *et al*. Reservoir computing using dynamic memristors for temporal information processing. *Nat. Commun.* 8, 2204 (2017).

22. Zhu, X., Wang, Q., & Lu, W. D. Memristor networks for real-time neural activity analysis. *Nat. Commun.* 11, 1-9 (2020).

23. Moon, J., Ma, W., Shin, J. H., Cai, F., Du, C., Lee, S. H., & Lu, W. D. Temporal data classification and forecasting using a memristor-based reservoir computing system. *Nature Electronics*, 2(10), 480-487 (2019).

Chapter 2
Temporal Data Analysis Using a Memristor-based Reservoir Computing System

2.1 Introduction

Recurrent neural networks (RNNs) [1,2] offer greatly improved ability to process temporal data compared to conventional feedforward neural networks. Due to the cyclic connections among hidden neurons, which are absent in feedforward neural networks, outputs in RNNs depend on both the current inputs and the neurons' previous states, allowing RNNs to discover temporal correlations in the data. However, the cyclic connections in RNNs cause problems commonly known as vanishing gradient and exploding gradient that make the training process expensive and difficult.

Variations of RNNs have been proposed to solve these problems, including Long Short-Term Memory [3] and reservoir computing (RC) [4]. In reservoir computing, in particular, a dynamic 'reservoir' that offers short-term memory (*i.e.* fading memory) property is used to nonlinearly map the temporal inputs into a high-dimensional feature space, represented by the states of the nodes forming the reservoir. This nonlinear mapping can cause the initial complex inputs to become linearly separable in the new space based on reservoir states, so that further processing can be performed using a simple, linear network layer (often called the readout layer) [4]. Since only the readout layer needs to be trained in an RC system, effective processing of temporal data can be achieved with low training cost. Software-based RC systems have already achieved state-of-the-art performance for tasks including speech recognition [5] (such as phoneme recognition [6]). More importantly, temporal data analysis, which is difficult to

perform for conventional neural networks, is naturally suited for RC systems due to the reservoir's capability to map diverse features at different time scales. For example, chaotic system prediction is an especially difficult problem due to the high sensitivity of the system on error, and forecasting of large spatiotemporally chaotic systems has only recently been demonstrated using a software-based RC system [7,8]. RC systems have also shown superior performance compared to conventional neural networks in other time-series forecasting tasks, including prediction of financial systems [9], and water inflow [10].

Recent studies have aimed at hardware implementation of RC systems, using dynamic memristors [11], atomic switch networks [12], silicon photonics [13] and spintronic oscillators [14]. Of the different approaches, memristor-based RC systems can be fabricated using standard foundry processes and materials to allow direct integration with control and sensing elements with high density, making them particularly attractive for hardware implementations. Memristors [15-18] have already been extensively studied for neuromorphic computing applications [19-22], and studies on memristor-based RC systems [11] show that the intrinsic nonlinear characteristics and short-term memory effects of memristors provide central properties of a good reservoir, namely, separation and echo state properties [4]. Tasks such as hand-written digit recognition have been experimentally demonstrated using memristor-based RC systems [11]. In the memristor-based RC systems, the fading memory property is natively implemented by the volatile effect of the memristor itself (e.g. in a volatile WO_x memristor). Additionally, since the reservoir only needs to be excited, the internal connections in the memristor-based reservoir will remain fixed so training is not needed for the reservoir itself. The only trainable part in the RC system is the readout network, which should be implemented in other systems with non-volatile weights to facilitate training.

Since the performance of the RC system depends strongly on the dimensionality of the reservoir space, increasing the number of nodes in the reservoir has become an experimental challenge. An interesting approach to this problem is to build the reservoir using a single physical nonlinear node subjected to delayed feedback, which can effectively act as a chain of virtual nodes without significant performance degradation compared with a conventional reservoir [23]. Since only a single physical node is needed, the delay-system based RC approach is attractive for hardware implementation using emerging devices, and has been demonstrated recently using photonic- and spintronic-based systems [24-26].

In this chapter, we show memristor-based RC systems employing the virtual node concept can be used to efficiently process temporal data, producing excellent results for important tasks such as speech recognition and time-series forecasting. For example, time-series forecasting is an active research area that can impact broad fields. The traditional approach for time-series forecasting is to attempt to derive a set of equations describing the desired system from abundant observations. However, in most cases it is practically impossible to obtain the equations representing the system accurately. Alternatively, prediction may be performed based on past and present data using statistics-based [27,28] or machine learning techniques [29,30]. However, prediction accuracy of statistics-based techniques is limited by the parameters of the assumed model and the complexity of the system. Typical systems often could not adequately capture the nonlinear relationships in the data, even with nonlinear dependencies included in the model. Neural networks offer a more general and flexible tool since they do not depend on parameters of specific tasks but are driven only by the data [31]. In particular, recurrent neural networks have gathered much attention for forecasting since the temporal information captured by the recurrent connections improves the prediction performance [32-34], although at increased

training cost. To this end, RC systems offer the capability of efficient temporal processing at low training cost, and are thus well suited for times series analysis and forecasting tasks, including autonomous prediction of chaotic systems [7, 8, 26].

2.2 Memristor-based Reservoir Computing System

In a typical memristor, the applied electrical stimulus triggers the migration of oxygen vacancies or metal ions in the switching layer, leading to modulations of the local resistivity and the overall device resistance [35]. In some devices, the system can quickly relax back to the original state due to spontaneous ion diffusion, leading to a short-term memory behavior [36, 37]. These devices natively offer the 'fading memory' properties required by a reservoir and have been used to implement RC systems for tasks such as hand-written digit recognition where the features in the spatial domain are converted into features in streaming inputs [11].

We note, however, there are significant differences in the reservoir configurations between a conventional RC system and the memristor-based one. Typical reservoirs may consist of several hundreds or thousands of internal nodes through complex interconnections, where the states of the nodes are all accessible by the readout network. In the memristor-based implementation, the short-term memory effect is obtained natively from a single device, instead of loops formed by multiple nodes. As a result, a reservoir may consist of a single device (node) where the reservoir state is represented by the excited memristor state and extracted from conductance measurements. This allows much simpler experimental implementation, but also limits the size of the reservoir. Multiple devices can be used to expand the reservoir size based on device-device variations where the reservoir state is represented by the collective states of all devices [11], but the nodes are independent of each other in this case instead of being nonlinearly coupled. To expand the dimensionality of the reservoir and improve the system's performance,

we adapt the concept of virtual nodes originally developed in delay systems [23]. In a delay system, a single physical nonlinear node is subjected to delayed feedback, where the excitations of the physical node in response to the delayed signals can effectively act as a chain of virtual nodes [23]. The state of virtual nodes depends on the node's own previous state, the current state of adjacent nodes and the masked input signal, allowing them to be nonlinearly coupled. By using randomly generated masks for input signals, diverse responses can be obtained from the virtual nodes, and the delay systems have been shown to be able to achieve comparable performance as conventional, well-designed reservoirs [23].

We hypothesize that the general virtual node concept, represented as the nonlinear response of a physical memristor device at selected time steps in response to a streaming input, can similarly be used to increase the reservoir size and allow better mapping of the input features. Below we experimentally implement this concept and show that this approach indeed allows the memristor-based RC system to successfully perform complex tasks such as temporal data analysis and forecasting.

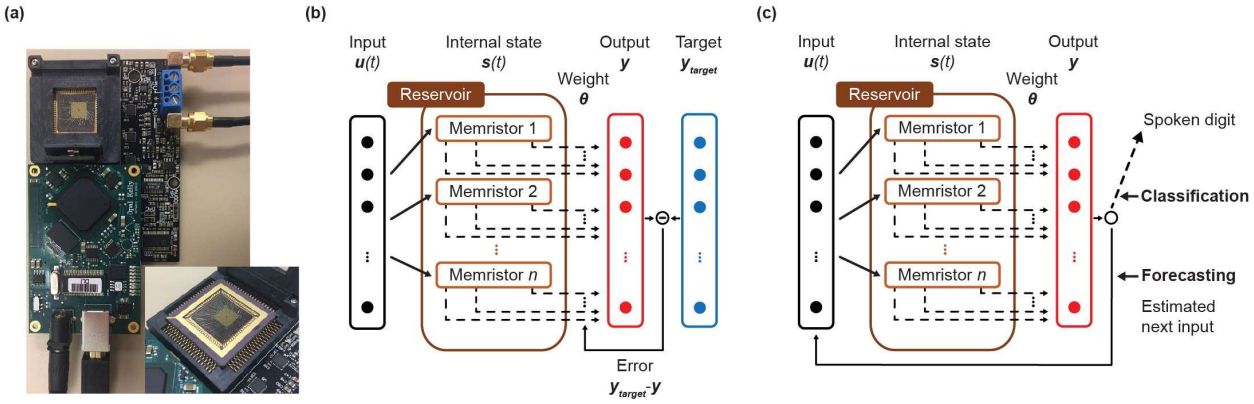

Figure 2-1 Memristor-based reservoir computing system.
(a) Optical image of the hardware system. The memristor array is wire-bonded and mounted on the test board system. Inset : Optical image of the 32×32 memristor chip. (b) Schematic of the training phase for the memristor-based reservoir computing system. Weights in the readout layer are updated to reduce the output error. (c) Schematic of the testing phase for the memristor-based reservoir computing system. The system can be used to either perform classification tasks, or forecasting tasks where the output produced from the system is fed back to the reservoir as input for the next frame.

The hardware implementation is based on a 32×32 WO_x memristor array integrated onto a custom-built test board (Figure 2-1 (a)). The operation of the RC system is shown in Figure 2-1 (b) and (c), for the training phase and the testing phase, respectively. During the training phase, a teacher signal $\boldsymbol{u(t)}$ is applied to the reservoir to bring the reservoir to different excited states (depending on the temporal streaming inputs). The weights $\boldsymbol{\theta}$ in the readout layer are then trained to minimize the error between the output of the readout network $\boldsymbol{y}$ and the target output $\boldsymbol{y_{target}}$. The target can be either the correct digit label (for classification) or the next value in the time series data (for forecasting). Note that only weights in the linear readout layer need to be trained, while the connections in the reservoir remain fixed. In the testing phase, testing inputs, not included in the training set, are applied to the reservoir. Depending on the type of tasks, the final output implementation can be quite different. For classification, the system configuration remains the same as in the training phase, and the performance of the system is evaluated by analyzing how many testing samples are correctly classified. On the other hand, in forecasting applications, the output from the readout layer is fed back to the system as the input signal for the reservoir at the next time step. In this case, external inputs are not needed, and the system is left running autonomously to generate the target time series.

2.2.1 Device Fabrication

The memristor-based reservoir system was fabricated in a crossbar structure, in which WO_x memristors are formed at each cross point. Starting from a substrate with 100 nm thermally grown silicon oxide on a silicon wafer, 60 nm W was deposited by magnetron sputtering and patterned by e-beam lithography and reactive ion etching (RIE) using Ni as a hard mask to form W bottom electrodes (BEs) with 500 nm width. The Ni hard mask was then removed by HCl wet etching. A spacer structure formed along the sidewalls of the W BEs was employed to improve

the fabrication yield. The space was patterned by the deposition of 250 nm think SiO_2 through plasma-enhanced chemical vapor deposition, followed by etch back by RIE. The WO_x switching layer was then formed on the exposed W BEs by rapid thermal annealing in oxygen gas ambient at 375°C for 45 seconds. Afterwards, the top electrodes (TEs) with 500 nm width were patterned by e-beam lithography, e-beam evaporation of 90 nm Pd and 50 nm Au, and lift-off process. An RIE process was then used to remove the WO_x between the TEs to isolate the devices and to expose the BEs for electrical contacts. Finally, photolithography, e-beam evaporation and liftoff processes were performed to form wire bonding pads of 150 nm thick Au. After fabrication, the memristor chip was wire bonded to a chip carrier and mounted on a custom-built test board for electrical testing. Figure 2-2 (a) shows a schematic of the memristor device structure and a zoomed-in scanning electron microscopy image of the 32×32 memristor array.

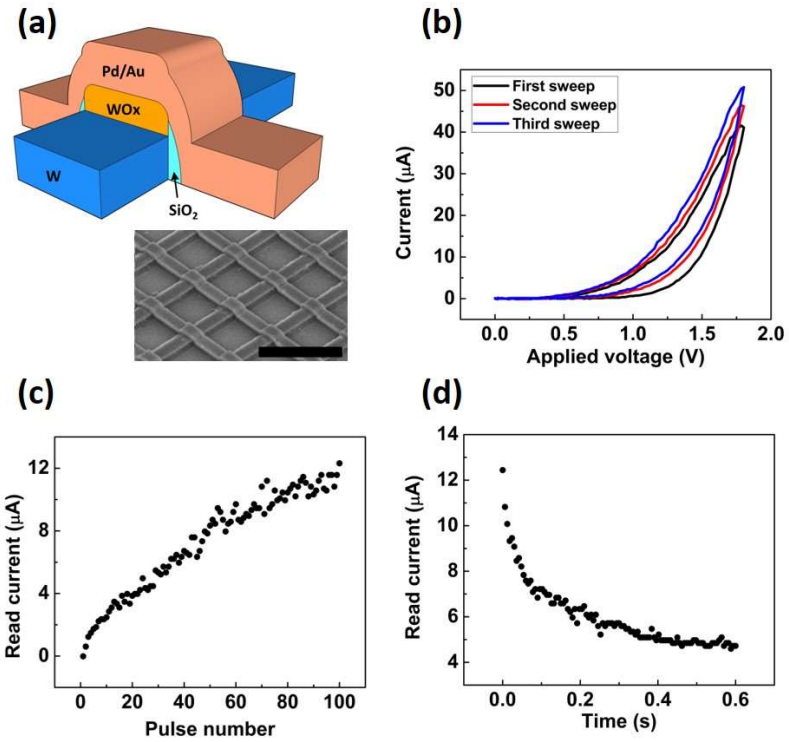

Figure 2-2 Characteristics of the WO_x memristor.
(a) Schematic of the device structure, showing the W bottom electrode, WO_x switching layer, SiO_2 spacer and the Pd/Au top electrode. Inset: A zoomed-in scanning electron microscopy image of the 32×32 memristor array. Scale bar: 3 μm. (b) DC

current-voltage characteristics of the WO_x memristor. (c) Incremental conductance increase obtained from consecutive programming pulses. 100 consecutive programming pulses (1.6 V, 5 ms) were applied to the device, and the response of the device was monitored by a small read pulse (0.6 V, 500 μs) after each programming pulse. (d) Conductance decay showing the short-term memory behavior of the WO_x memristor. The device conductance was measured by the same read pulses (0.6 V, 500 μs) without any programming pulse.

2.2.2 Device Characteristics

A memristor's resistance is determined by the internal ion configuration, which can be modulated by the external electric field and the ion's concentration gradient. The WO_x device's behavior is attributed to the migration of oxygen vacancies (V_Os) inside the device. The device characteristics can be described by the following equations:

$$I = (1 - w)\alpha[1 - \exp(-\beta V)] + w\gamma \sinh(\delta V) \quad (2\text{-}1)$$

$$\frac{dw}{dt} = \lambda \sinh(\eta V) - \frac{w}{\tau} \quad (2\text{-}2)$$

where equation (2-1) is the I-V equation and equation (2-2) is the rate equation for the state variable *w* of the memristor. The conduction channels can be divided into two regions: V_O-poor region for Schottky conduction (1st term of equation (2-1)) and V_O-rich region for tunneling conduction (2nd term of equation (2-1)). The internal state variable w is the relative weight of the two regions, *i.e.* $w = 0$ indicates fully Schottky-dominated conduction while $w = 1$ indicates fully tunneling-dominated conduction. The dynamic of the internal state variable is governed by the drift effect under an external electric field (1st term of equation (2-2)) and the spontaneous diffusion effect (2nd term of equation (2-2)). $\alpha, \beta, \gamma, \delta, \lambda, \eta$ and τ are all positive-valued parameters determined by material properties. To include the device variation effect in the simulation, each device was assigned a different value of η, *i.e.* 3.6~4.4, in the simulation.

The WO_x device shows gradual conductance changes and short-term memory behaviors, which allow it to form nodes with an intrinsic fading memory effect in the reservoir. As shown in

Figure 2-2 (b), during the three consecutive positive DC voltage sweeps the conductance increases continuously with each sweep. Additionally, a significant amount of overlap between the hysteresis loops is observed, indicating that there is some "memory loss" between the sweeps. The gradual conductance change behavior was more clearly observed using pulse programming tests as shown in Figure 2-2 (c), while the short-term memory effect was more clearly observed in Figure 2-2 (d), where gradual decay of the device conductance, characteristics of a stretched exponential function, can be observed after the initial increase due to the application of a programming pulse.

2.3 Speech Recognition

2.3.1. Isolated Spoken Digit Recognition

To validate the proposed approach, we first perform a standard speech recognition benchmark– isolated spoken digit recognition using the memristor-based RC system. The inputs for the reservoir are sound waveforms of isolated spoken digits (0-9 in English) from the NIST TI-46 database [38], preprocessed using the Lyon's passive ear model [39] based on human cochlear channels. The preprocessing transforms the sound waveforms into a set of 50-dimensional vectors (corresponding to the frequency channels) with up to 40 time steps. One example representing digit 0 is shown in Figure 2-3 (a). Each data point on the graph represents the firing probability of a neuron corresponding to a specific frequency channel at a time point, with 50 frequency channels in total plotted along the *y* axis and time plotted along the *x*-axis. The input graph (termed cochleagram) is then digitized to form the streaming inputs to the reservoir by setting a threshold (0.5) for the firing probability [40] (Figure 2-3 (b)). For this isolated spoken digit classification task, 50 WO_x memristor devices are used to experimentally implement the RC system, with each device processing the input spike train at one of the 50

channels. The inputs are converted into digitized spike trains and applied to the memristor-based reservoir. The amplitude/width of the spikes are 3.0 V/10 μs, and the device conductance is read out with read pulses of 0.6 V/200 μs. The length of a unit time step in the experiment is 250 μs.

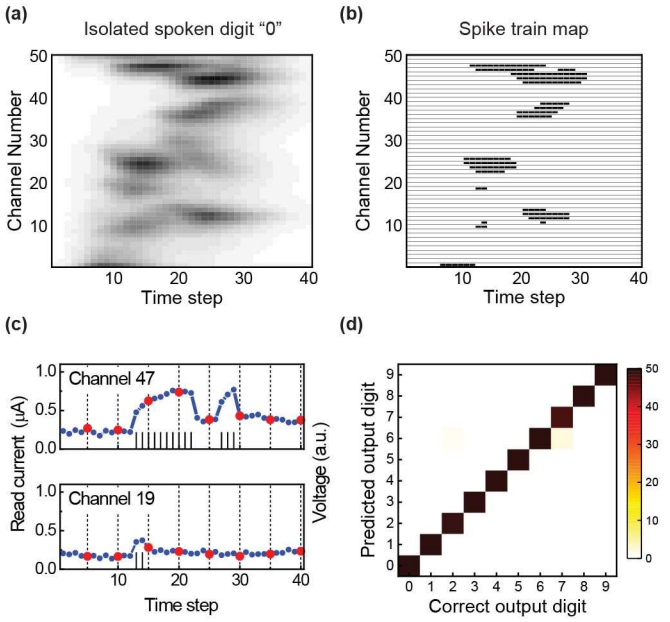

Figure 2-3 Spoken-digit recognition task implementation. (a) Cochleagram of the first female speaker, first utterance speech sample after being processed by the Lyon's passive ear model. Each data point represents the firing probability of a hair cell sensitive to a certain frequency (channel) at a given time point. (b) Digitized spike trains converted from the cochleagram shown in (a), by setting a threshold (0.5) for the firing probability. (c) Temporal response of memristors to the spike trains in channel 47 and 19. The current (blue dots) was measured by read pulses through the test board. The spike train inputs (black lines) are also plotted for reference. The complete input can be divided into 8 intervals, representing 8 virtual nodes whose states are measured at the end of the intervals (red dots). (d) Confusion matrix showing the experimentally obtained classification results from the memristor-based reservoir system vs. the correct outputs. An overall recognition rate of 99.2 % is achieved.

Figure 2-3 (c) shows the responses obtained experimentally from the memristors to spike trains at frequency channels 47 and 19, respectively. The input data were taken from female speaker 1, first utterance in the database. Due to the short-term dynamics of the memristors, the temporal patterns in the input spike trains led to diverse but deterministic device responses, where the device conductance is increased when stimulated by a spike and decays spontaneously, with the device conductance at a specific time depending on the recent history of the temporal inputs [40]. However, the temporal information of the early part of the input sequence (*i.e.* far history) is not conveyed in the final responses of memristors. For example, as shown in Figure 2-

3 (c), even though the device responses are significantly different within time steps 10-35 of the streaming inputs, the responses in the end at time step 40 do not show significant difference between the two inputs due to similarities at the last part of the two inputs (time steps 36-40). The loss of information will lead to poor classification results. To address this problem, we divide the whole input sequence into n equal intervals and measure and record the device state at the end of each interval. This process effectively creates n virtual nodes from a single device. The extracted virtual node state is affected by the temporal input pattern in the near history of that virtual node, and also by the previous virtual node states. The chain of virtual node states in turn form nodes in the reservoir to allow effective temporal data analysis.

Specifically, we measure the device conductance at time steps $40/n$, $40\times2/n$, $40\times3/n$, $40\times4/n$ … $40\times(n\text{-}1)/n$, and 40, as the virtual nodes' states in the reservoir. One example of $n = 8$ is shown in Figure 2-3 (c), where the red dots represent the measured virtual node states. The virtual node states are then supplied to the readout layer (a 400×10 perceptron in this case) for training and testing. Figure 2-3 (d) shows the confusion matrix obtained experimentally during testing, after training the readout layer with 450 speech samples. Overall a high recognition rate of over 99.2 % was obtained for all inputs using the memristor-based RC system.

2.3.2. Comparison with Conventional Neural Networks

To verify the effectiveness of the memristor-based reservoir in capturing temporal features in the input, the memristor-based system was compared to other systems based on conventional convolutional filters: the memristor-based reservoir system, a single-valued system that keeps the value at the end of each input window, an averaged system that obtains the averaged value over the input window, and a CNN (convolutional neural network) system that

applies a trained filter to the data in the input window. The outputs from these systems are then fed to readout functions having identical sizes, as shown in Figure 2-4 (g).

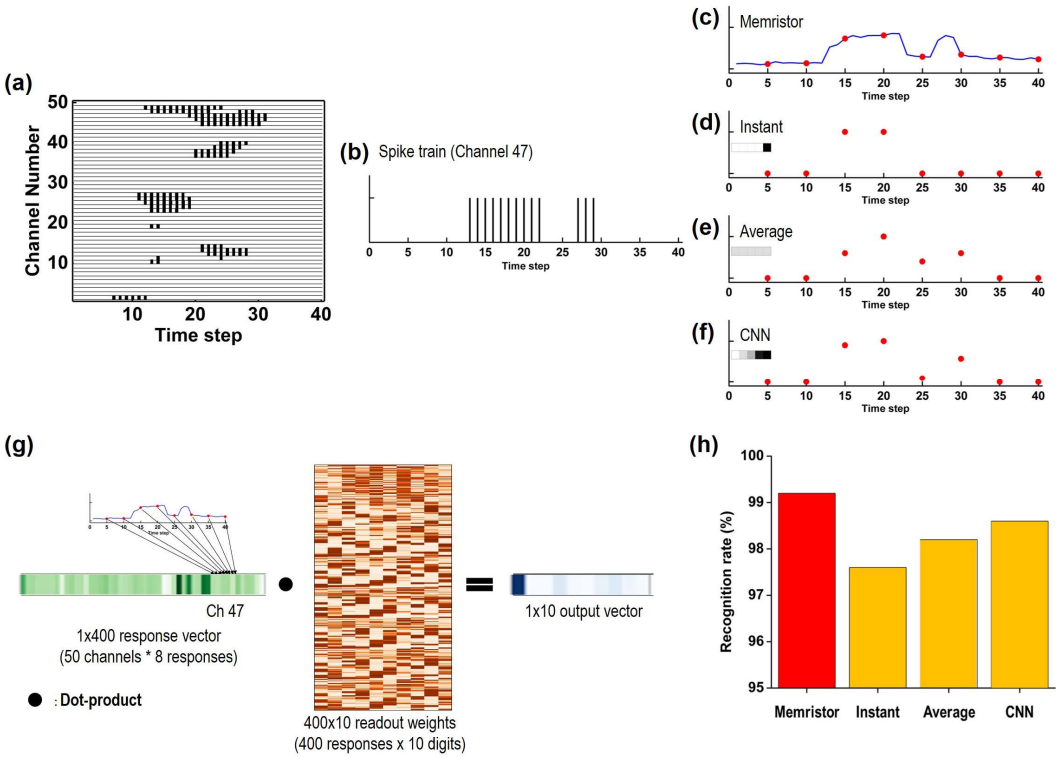

Figure 2-4 Spoken digit classification using four different systems. (a) Digitized spike trains converted from the first female speaker, first utterance '0' speech sample. (b) Digitized spike train of channel 47. (c)-(f), Response (red dots) from the four different systems: (c) memristor-based reservoir system, (d) single-valued system, (e) averaged system and (f) CNN system with a trained filter. Insets show the weights of an equivalent filter used in the three control systems. (g) Schematic showing output from the reservoir (or the control systems) applied to the readout network to produce the classification results. (h) Recognition rates of the four different systems. The memristor-based system was tested both by simulation and experimentally, where identical classification rates, 99.2% were obtained. The other 3 control systems were implemented in software.

Additionally, to allow a more direct comparison between our analysis and other studies [14, 23], we used the same size of readout network with these earlier studies, which is 400×10. In all 4 systems, the sound waveforms were digitized in 40 time steps and transformed to 50 frequency channels. Figure 2-4 (a) shows digitized spike trains of the first female speaker, first utterance '0' speech sample, and Figure 2-4 (b) shows a schematic of the input from one of the frequency channels. Each input sequence is then divided into 8 intervals (windows), so that after being processed by the reservoir or the other 3 systems used in the control study, a 1×400 vector

is produced and used as the input for the readout layer. Figure 2-4 (c)-(f) show the outputs (represented by the red dots) from the memristor reservoir and the three control systems.

We also adopted the standard *n*-fold cross validation approach to improve the testing reliability. In other words, training and testing are repeated *n* times on the same combined data set, with each time having a different assignment of training and testing samples. In this work, we used 10-fold cross validation among 500 total samples, where 450 samples were chosen for training and 50 samples were chosen for testing at each run. The recognition rates are defined as the mean across these 10 runs.

For the CNN system, we set the parameters as: filter size = (1,5), stride = (1,5) and the number of filters = 1. We used the Python toolkit Keras, which provides a high-level application programming interface to access TensorFlow deep-learning libraries. To train the weights in the CNN system, we used a gradient-based optimization method (RMSProp) and a categorial cross-entropy cost function. Note that the other two systems, using the last point in the input window and using the average in the input window, can be considered as special cases of the CNN system, having hand-crafted filters, *i.e.* fixed as [0,0,0,0,1] for the singled-valued system and [0.2, 0.2, 0.2, 0.2, 0.2] for the averaged system, instead of trained filters.

As can be seen in Figure 2-4 (h), the memristor-base reservoir system (with accuracy 99.2 %) clearly outperforms the other three systems: the single-valued system (97.6 %), the averaged system (98.2 %) and the CNN system (98.6 %). Both the single-valued system and the averaged system showed poorer ability to process the temporal information, since the single-valued system loses all information contained in the first four pulses in the input window, while the averaged system only keeps the information related to the amplitude of the pulses but loses the temporal information about the specific pulses, *i.e.* whether the pulses come early or late in

the window. Thus, the single-valued system and the averaged system produced worse performance compared to the CNN system and the memristor-based system.

For the CNN system, the fact that the filter can be trained to reduce classification error allows the system to better process the input than the other two systems discussed earlier. However, the transformation by the CNN filter is still linear and the output is only determined by data in the current input window, since the system has no memory of inputs from previous intervals. In contrast, the memristor-based reservoir not only nonlinearly transforms the temporal information in the current input window, but also reflects effects from previous inputs due to the short-term memory effect of the device. These characteristics allow the memristor-based reservoir system to better capture both the local temporal features within an input window as well as the more global features among the input windows, leading to higher performance.

These results are also compared with standard speech processing algorithms such as the Long Short-Time Memory (LSTM) model. With a single layer, the LSTM algorithm can achieve recognition rate of 98.6% with 150 hidden units, while requiring a much larger number of trainable parameters (122,110 vs. 4,000 used in the memristor RC system). Better accuracy can be obtained by using stacked LSTM but at the expense of even higher training costs.

2.3.3. Isolated Spoken Digit Recognition with Partial Inputs

More interestingly, as the memristor-based reservoir maps the temporal features associated with different classes of inputs, it is possible to use the RC system to predict the spoken digit before the utterance is completed. This hypothesis was tested experimentally as shown in Figure 2-5.

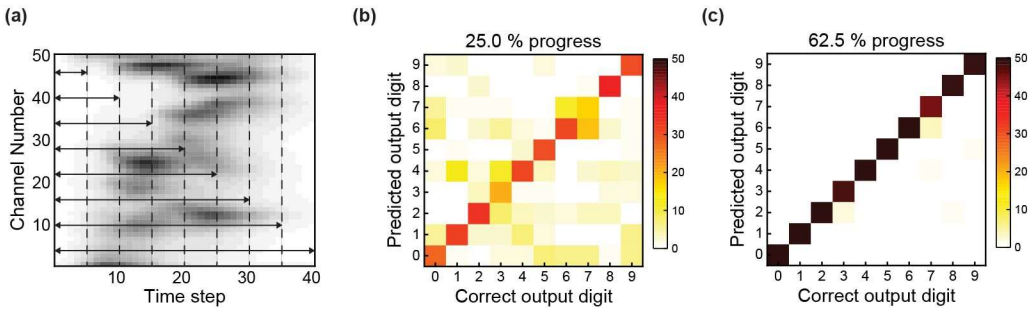

Figure 2-5 Classification using partial inputs. (a) Schematic of the different portions of data used for classification. (b)-(c) Confusion matrices showing the experimental classification results obtained using 25.0 % (b) and 62.5 % (c) of the input signal.

When the first 25% of the speech signal is used, each device produces only one virtual node so that a 100×10 readout network is used during the training phase. As shown in Figure 2-5 (b), after training an 57.8 % recognition rate can already be achieved using only the first 25 % of the whole sequence. The recognition accuracy increases when the portion of the available input increases (*e.g.* 98.2% recognition rate for using the first 62.5 % of the input, and 99.2% recognition rate when the whole speech sequence is used). The evolution of the recognition rate as a function of the portion of input used is shown in Figure 2-6. The 99.2% recognition rate is comparable to results achieved in RC systems based on spintronic [14] or photonic [24, 25] devices, although to the best of our knowledge, this experiment represents the first RC experimental implementation of spoken digit classification using only part of the input sequence.

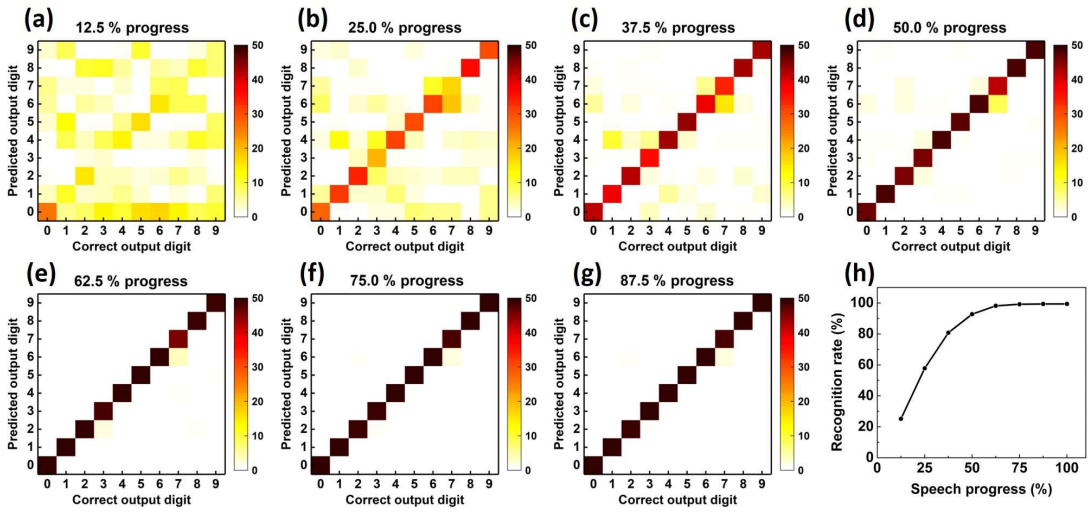

Figure 2-6 Spoken digit classification with partial inputs. (a)-(g), Confusion matrices showing classification results of the spoken digits experimentally obtained from the memristor-based RC system, as a function of the fraction of inputs used for analysis , using only the first 12.5 % (a), 25 % (b), 37.5 % (c), 50% (d), 62.5% (e), 75% (f), 87.5% (g) of the data of each sequence. (h), Recognition rate as a function of the fraction of data used in analysis.

2.4 Time-series Forecasting

2.4.1 Mackey-Glass Time Series Prediction

The isolated spoken digit recognition task has relatively good tolerance to prediction error because the task only requires identifying the largest output among the 10 outputs. Thus, as long as the selected label is correct, small errors in the exact output value does not degrade the system's performance. In contrast, forecasting time series data is a much more difficult task since the quantitative difference between the ground truth and the predicted value matters and the difference can be further accumulated in subsequent predictions. One standard benchmark test for time-series forecasting is predicting a chaotic system, which is inherently very challenging due to the positive Lyapunov exponent [40] in chaotic systems, which leads to exponential growth of separation of close trajectories so that even small errors in prediction can quickly lead to divergence of the prediction from the ground truth.

To show if the memristor-based RC system can be used for long term prediction, we tested the system using the Mackey-Glass time series [42, 43] given by

$$\frac{dx}{dt} = \beta \frac{x(t-\tau)}{1+(x(t-\tau))^n} - \gamma x(t) \quad (2\text{-}3)$$

These types of chaotic systems have a deterministic form but are difficult to be predicted [7, 26] and thus has been widely used as a benchmark for forecasting task tests. The Mackey-Glass time series is based on a non-linear time delayed differential equation that can display a wide range of periodic and chaotic behaviors, depending on the values of the parameters. For example, $\tau < 4.43$ produces a fixed-point attractor, $4.43 < \tau < 13.3$ produces a stable limit cycle attractor, $13.3 < \tau < 16.8$ produces a double limit cycle attractor, and $\tau > 16.8$ produces chaotic behaviors. To obtain chaotic dynamics, we set the parameters: $\beta = 0.2$, $\gamma = 0.1$, $\tau = 18$, $n = 10$ in our studies. The time series data are normalized into the range [-0.5,0.5].

To improve the accuracy of prediction, it is necessary to build a reservoir system that can capture the temporal dynamics of the given chaotic system as closely as possible, which in turn allows the readout network to achieve accurate prediction. To achieve this goal, we used several techniques to expand the reservoir dynamics. First, we used *m* different memristors for our memristor-based reservoir system. Due to the naturally occurring device-to-device variations (Figure 2-7), the devices produce responses that are qualitatively similar but quantitatively different even with the same input, and such device variations expand the response of the reservoir. Second, *n* virtual nodes are obtained from each physical memristor, based on the device response at the present time step and *n*-1 previous time steps, similar to the approach used in the classification task.

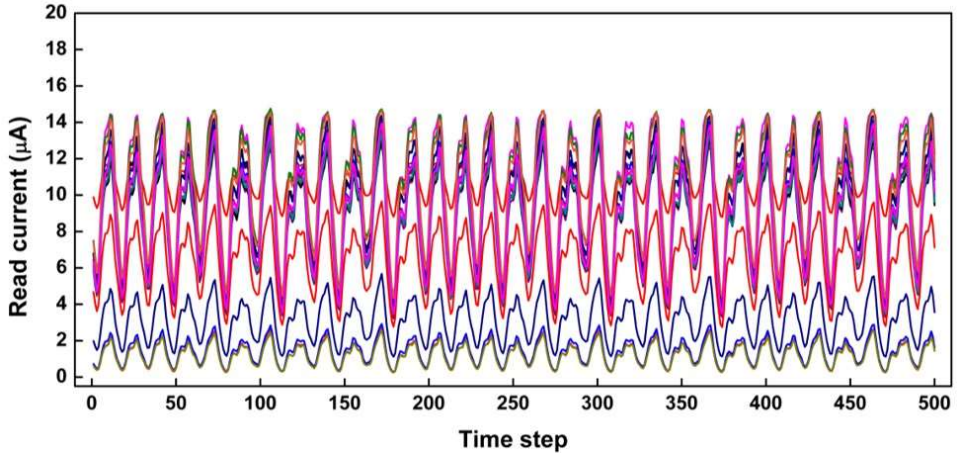

Figure 2-7 Read currents showing the temporal response of the 20 memristor devices during the training phase of the Mackey-Glass forecasting task. The devices show qualitative similar behaviors but sizeable device-device variations in the current values.

The ability of the memristor-based RC system for time-series prediction was tested both experimentally using a reservoir with m=20 devices and n=50 virtual nodes for each device, and through simulations of the reservoir using a realistic device model. The reservoir states are then applied to the readout layer (a 1000×1 network) to generate the predicted data for the next time step. Figure 2-8 (a) and (c) show the results obtained during training, from the experimental measurements and the simulation, respectively. Excellent agreement between the target and the predicted value can be obtained, indicating that the trained readout weights can correctly calculate the next time step signal based on the internal states of reservoir. Further evidence of successful training can be found by examining the network performance in the frequency domain and in the phase space, as shown in Figure 2-8 (e) and (f) for the experimental results, where excellent match can again be observed between the memristor RC output and the ground truth.

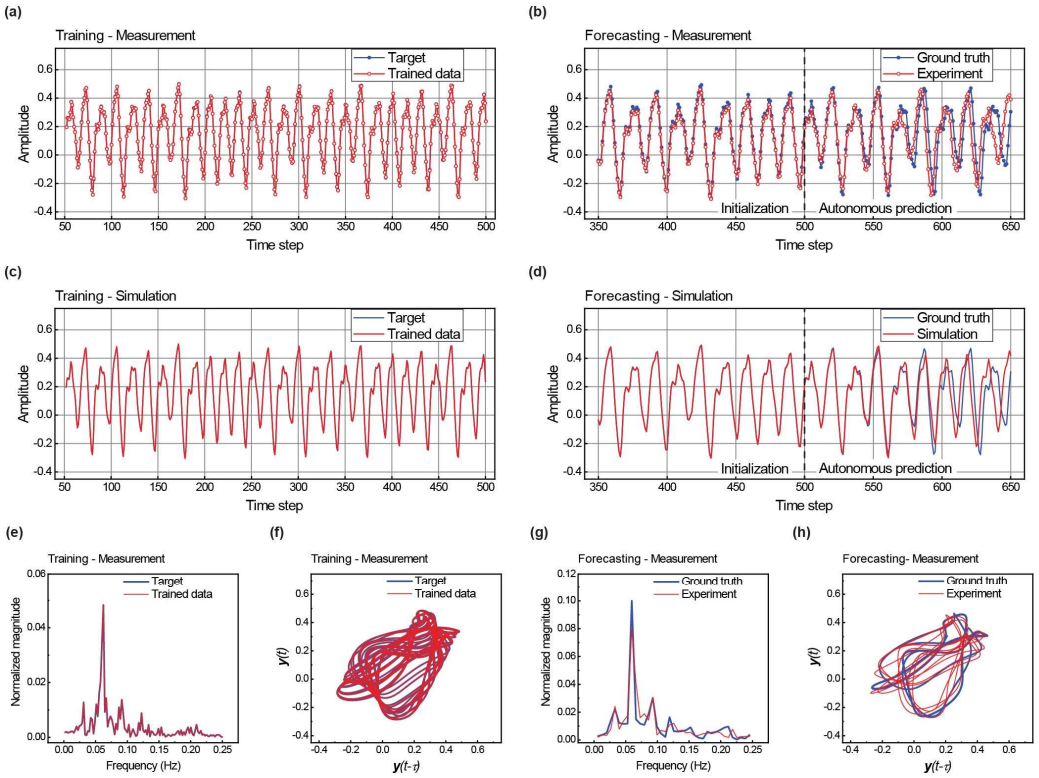

Figure 2-8 Autonomous forecasting of Mackey-Glass time series. (a)-(b) Training (a) and forecasting (b) results obtained experimentally from the memristor-based RC system. The ground truth (blue) and the predicted output from the RC system (red) are plotted. An initialization step is used before the autonomous forecasting process, which starts from time step 501 in (b). (c)-(d) Training (c) and forecasting (d) results obtained from simulation using a realistic device model and the same RC system setup as the experiments. (e)-(f) Experimental results of the memristor RC system output plotted in the frequency domain (e) and in the phase space (f) for the training stage. (g)-(h) Experimental results of the memristor RC system output during autonomous forecasting, plotted in the frequency domain (g) and in the phase space (h).

The network is then used to forecast the time series autonomously. Before the start of the autonomous prediction, an initialization stage is needed to prepare the internal states of the reservoir since chaotic systems depend strongly on the initial conditions. During the initialization stage, the feedback connection between the network output and the reservoir input is removed, and the true input data are applied to the reservoir instead. Note the goal of the initialization is to simply excite the reservoir to the state just before autonomous prediction starts, not to train the system. After the reservoir is initialized, the output from the readout function, that is, the predicted data for the next time step, is then connected to the reservoir as the new input, and the system autonomously produces the forecasted time series continuously.

Figure 2-8 (b) and (d) show the results of the experimental measurements and simulation for autonomous time-series prediction using the memristor-based RC system, respectively. As the reservoir states are stabilized during the initialization stage, the output values obtained from the readout layer follow accurately the correct values. Afterwards, the autonomously generated output (from the 500th time step onwards) still matches very well the ground truth (which is not used in the signal generation but simply used as a reference in the plot), showing the ability of the memristor based RC system to autonomously forecast the chaotic system. After 60~70 time steps of autonomous prediction, the predicted signal starts to diverge from the correct value, in both the simulation and the experiment. Careful analysis of the divergence shows that small errors in the prediction are accumulated during the autonomous prediction, and can lead to a phase shift of the time series data as represented by missing one or more data points in the prediction, resulting in significantly increased prediction error as shown in Figure 2-8 (b) at time steps 570 – 650. Nevertheless, even though in the long-term phase shifts occur due to the small network size in the experiment and the accumulation of errors, the memristor-based RC system can still correctly capture the characteristic temporal dynamics of the Mackey-Glass series. Interestingly, with the phase shifts we often observe the autonomous forecasting to precede the ground truth (vs. following the ground truth), while maintaining the same characteristic dynamics. Examination of data in the frequency domain also show similar frequency peaks in the autonomously generated data and the ground truth (Figure 2-8 (g)), and similar trace plots can be observed in the phase space (Figure 2-8 (h)). We note that if the memristor-based RC system could not emulate the chaotic system's dynamics, the prediction would instead converge to a stable point or periodic orbits. Indeed, if the reservoir size is reduced, *e.g.* to only one physical device, the autonomous predictions obtained both in the experiment and in the simulation decay

gradually and finally converge to a stable point (Figure 2-9) instead of showing the targeted chaotic behaviors. In this regard, the device-to-device variations are beneficial in helping expand the reservoir space and improving the RC system's performance.

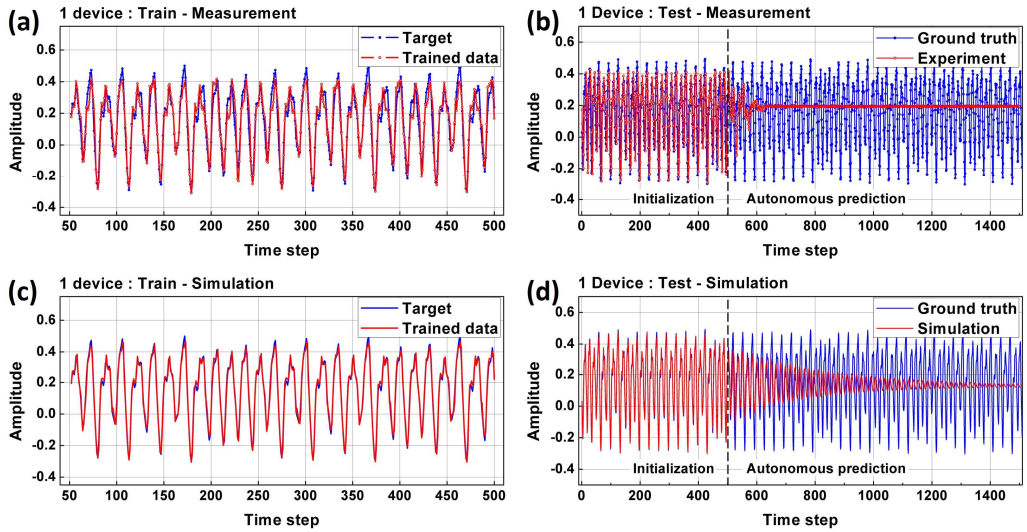

Figure 2-9 Mackey-Glass time series forecasting using an RC system based on only one memristor device. (a)-(b) Training (a) and forecasting (b) results obtained experimentally, showing the ground truth (blue) and the predicted output (red). The reservoir was initialized in the first 500 time steps during the forecasting task. (c)-(d) Training (c) and forecasting (d) results obtained through simulation. With a reduced reservoir size, the RC system quickly converges to a single stable point and cannot predict the desired chaotic behavior, showing the beneficial effects of device variations in this neural network implementation.

2.4.2 Comparison with Feedforward Neural Networks

To verify the function of the memristor reservoir system for chaotic system forecasting, we compared the memristor-based RC system performance with a conventional feedforward neural network (FFNN) having larger network sizes. To make a fair comparison, we allow the inputs to the FFNNs to include input from the current time step as well as inputs from 49 previous time steps. We chose hyperbolic tangent function as the neuron activation function for the FFNN, although different neuron functions led to the same qualitative results. In the training stage, the learning rate and number of training epochs for the FFNN are set to be the same as those for the memristor-based reservoir.

We tested several different types of FFNNs from the simple perceptron network to networks with multiple hidden layers, as shown in Figure 2-10. After training, the output from these networks corresponds to the predicted value at the next time step (t+1). This value is then applied to the input and the process is repeated to produce long term prediction, similar to the memristor-based RC system.

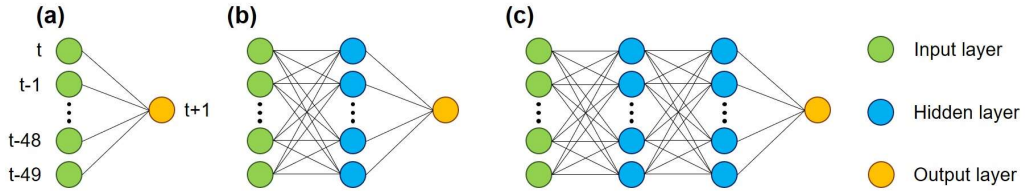

Figure 2-10 Schematic of FFNNs used in the tests. (a) Single layer perceptron. (b) Bilayer network with a hidden layer. (c) Three-layer neural network with two hidden layers.

Figure 2-11 shows comparison of the autonomous predicting results from the memristor-based reservoir system and the FFNNs. The results are shown both in the time domain (first column) and in the phase domain (second column and third column, for different timesteps into the prediction). These results, particularly when plotted in the phase domain, clearly show that predictions from the FFNNs are not producing chaotic dynamics and following the ground truth, but are instead stuck in a periodic behavior. These behaviors are very different from those obtained from the memristor system, both through simulation and through experiments using the fabricated devices and the test board. Note even with a much larger network, *e.g.* 1,051,000 parameters for the FFNN with two hidden layers, the forecasting performance is still much worse than the much smaller memristor-based RC system with only 1,000 parameters.

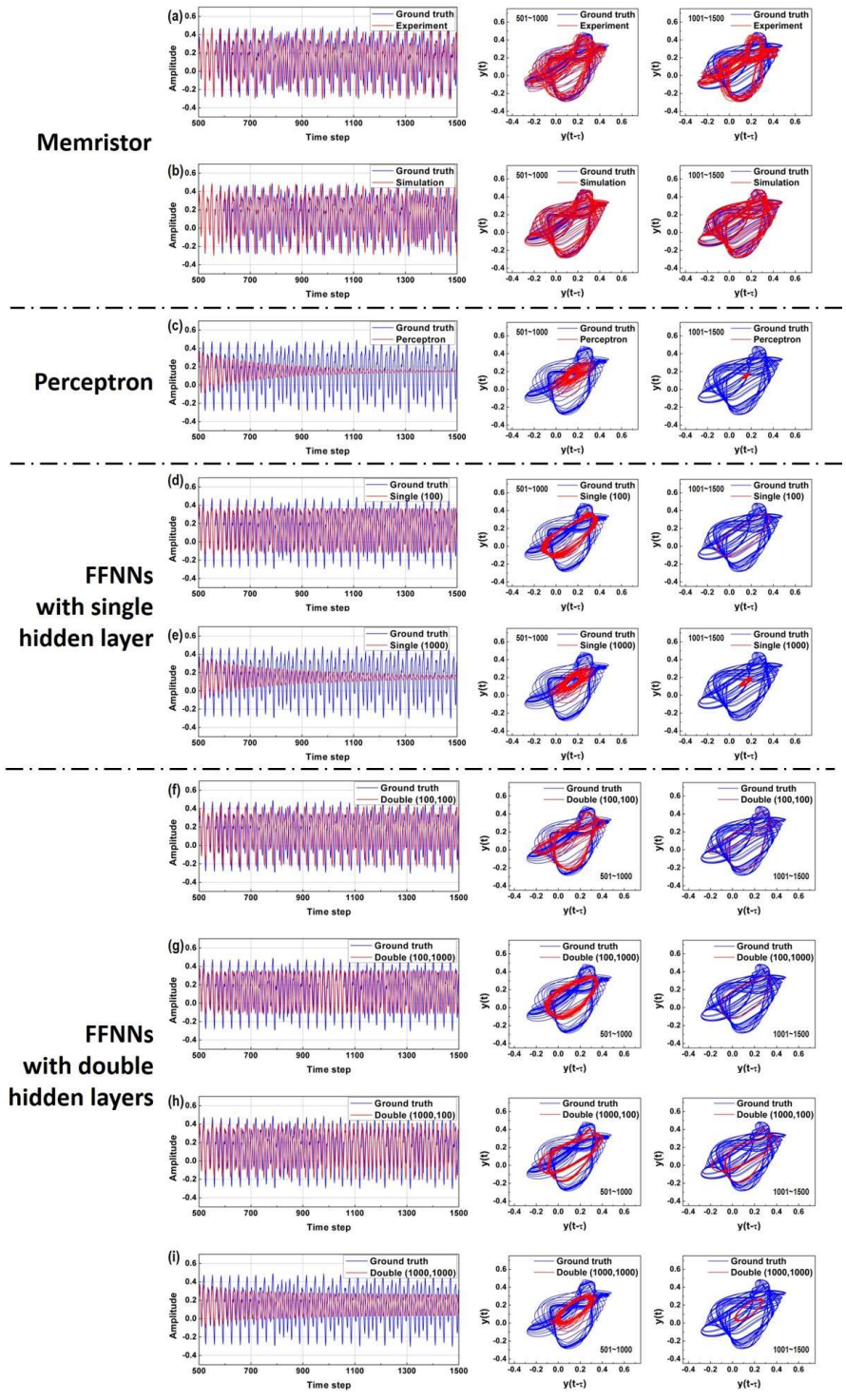

Figure 2-11 Autonomous forecasting of the Mackey-Glass time series using different systems. (a)-(b) Experiment (a) and simulation (b) results of the memristor-based reservoir system. (c) Simulation results of FFNN with single perceptron. (d)-(e) Simulation results of FFNNs with one hidden layer, with (d) 100 and (e) 1000 neurons. (f)-(i), Simulation results of FFNNs with two hidden layers, with (f) [first hidden layer: 100, second hidden layer: 100], (g) [100,1000], (h) [1000,100], and (i) [1000,1000] neurons. First column: results of the first 1000 predictions in the time domain, Second column: results of the first 500 predictions in the phase domain, Third column: results of the second 500 predictions in the phase domain.

These differences can be explained by the different network structures. The conventional FFNNs are based on linear sum of the weighted inputs. In this implementation, the inputs correspond to inputs at different time steps, *e.g.* Figure 2-10 (a). Although some degree of nonlinearity exists in the form of the nonlinear activation function, the same non-linear function is applied to all inputs. This is very different from the RC systems, where inputs at different time steps are transformed differently. For example, in the memristor based RC system, the effect of previous time steps diminishes following a stretched-exponential function in our devices. These very nonlinear transformation of inputs from different time steps in the past allows RC systems to capture diverse temporal features and gives them better capability to forecast the chaotic systems. On the other hand, the FFNNs are easily stuck in a periodic behavior because they do not offer the required nonlinearity in temporal domain and can only capture the main temporal features from the input instead. Increasing the size of the FFNN by making the network deeper and using more neurons and weights improves the system performance, but the system still quickly relaxes to a periodic behavior, as shown in Figure 2-11.

2.4.3 Mackey-Glass Time Series Prediction with Periodic Updates

Increasing the size of the reservoir further by using more memristors and using more previous states (virtual nodes) may further reduce the prediction error so that the length of accurate prediction can be further increased. However, it will increase hardware implementation cost as well as the training cost of the readout layer. Furthermore, even with a reduced error using a larger reservoir, accumulation of the prediction error during the autonomous prediction is inevitable, and more importantly, the length of accurate prediction will not increase significantly since the overall error grows exponentially in time. Thus, the autonomous prediction will

exponentially diverge from the original chaotic system, no matter how small the original prediction error is.

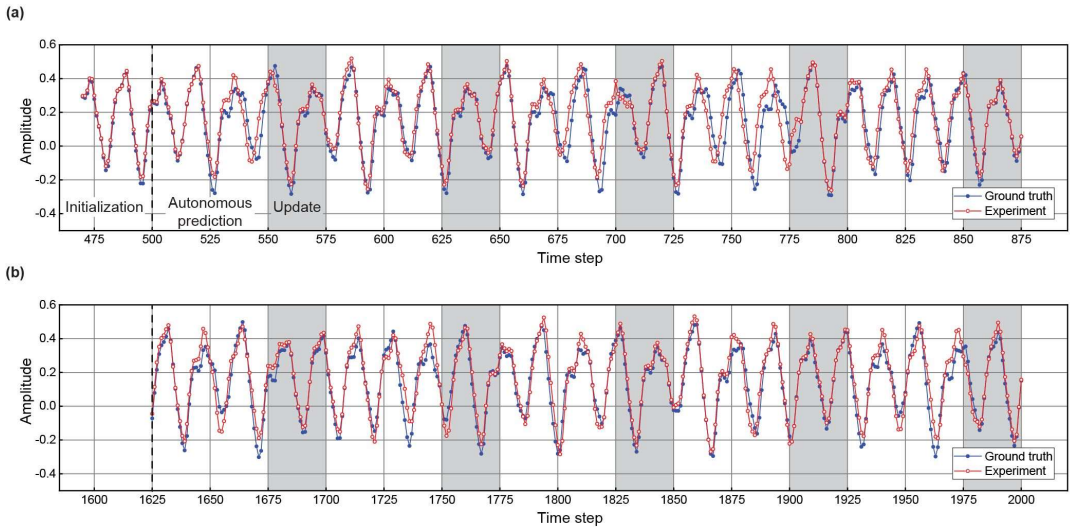

Figure 2-12 Long-term forecasting of Mackey-Glass time series with periodic updates. (a) Experimental output from the memristor-based RC system at the beginning of the test, showing results from the autonomous forecasting segments (50 time step each), followed by the update segments (shaded regions, 25 time step each). (b) Experimental output from the memristor-based RC system at the end of the test. Reliable forecasting can still be obtained, aided by the periodic updates.

Instead of increasing the network size to extend the range of correct predict further into the future, we explored an approach where the reservoir forecasts moderately into the future, and the reservoir state is then periodically pushed back to the original dynamics before the prediction significantly diverges from the ground truth. This 'update' stage is similar to the initialization stage, and is applied after a period of autonomous prediction by sending the true input instead of the predicted value to the reservoir for a short time period, as shown in Figure 2-12. No re-training is needed in the update stage. With these periodic updates, long-term prediction of chaotic systems becomes feasible. For example, by iteratively implementing 50 time steps of autonomous prediction, followed by 25 time steps of update, the experimental prediction shows excellent agreement with the true values over 2000 time steps (limited only by the buffer size of the on-board FPGA system). During the update stage, the reservoir reverses deviation from the

original chaotic system caused by the prediction error, and after the update sequence the memristor-based reservoir can again correctly perform prediction for the next 50 time steps.

Compared to the autonomous prediction without updates, prediction with periodic updates produces more stable and accurate results, both in the time and frequency domains and in the phase space (Figure 2-13). Note in many cases, the ground truth can indeed be periodically measured, so that periodically updating the reservoir to maintain stable operation of the system is feasible, where reliable predictions can be obtained during the periods when the ground truth cannot be measured.

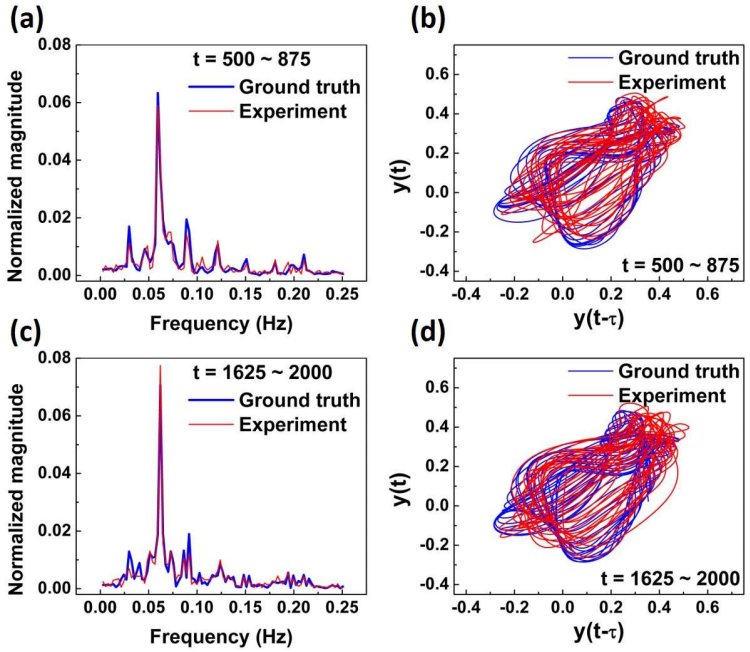

Figure 2-13 Mackey-Glass time series forecasting with periodic updates. (a)-(b) Experimental results at the beginning of the test (t = 500 - 875), plotted in frequency domain (a) and in phase space (b), showing the output from the memristor-based RC system (red) and the ground truth (blue). (c)-(d) Experimental results at the end of the test (t = 1625 - 2000), plotted in frequency domain (c) and in phase space (d), showing the output from the memristor-based RC system (red) and the ground truth (blue).

2.5 Conclusions

In this study, a memristor-based reservoir computing system that utilizes the internal short-term ionic dynamics of memristor devices and the concept of virtual nodes is successfully demonstrated for time-series analysis and prediction tasks. A high classification accuracy of

99.2% was obtained for spoken digit recognition. Since the reservoir maps the temporal features of the input, good classification was still obtained even with partial inputs. With the use of periodic updates to bring the reservoir back to the original dynamics, prediction can be maintained long-term without retraining the system even for chaotic tasks.

The memristor crossbar array used in this work provides the high-density devices, where the memristor devices in the reservoir work independently and in parallel to process the spatio-temporal data, *e.g.* inputs from different frequency channels. Further improvements in the reservoir system may involve using interconnected devices to form more complex reservoir structures, with properly designed connecting weights and loops in the system. Additional theoretical and algorithm developments will be necessary to help illustrate how broadly RC systems based on intrinsic device dynamics can be used in general machine-learning tasks. These theoretical developments, along with continued material and device optimizations and advances of integrated systems consisting of memristor arrays directly fabricated on CMOS control and logic circuits [44], will further broaden the appeal of the memristor based RC systems for practical applications, such as real-time temporal data processing in power constrained environments [45, 46].

References

1. Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences* 79, 2554-2558 (1982).
2. Werbos, P. J. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE* 78, 1550-1560 (1990).
3. Hochreiter, S., & Schmidhuber, J. Long short-term memory. *Neural computation* 9, 1735-1780 (1997).
4. Lukoševičius, M. & Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* 3, 127–149 (2009).
5. Verstraeten, D., Schrauwen, B. & Stroobandt, D. Reservoir-based techniques for speech recognition. In *Proceedings of IJCNN06*, International Joint Conference on Neural Networks*,* 1050–1053 (2006).
6. Triefenbach, F., Jalalvand, A., Schrauwen, B., & Martens, J. P. Phoneme recognition with large hierarchical reservoirs. in *Advances in neural information processing systems* 2307-2315 (2010).
7. Jaeger, H., & Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science* 304, 78-80 (2004).
8. Pathak, J., Hunt, B., Girvan, M., Lu, Z., & Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach. *Phys. Rev. Lett.* 120, 024102 (2018).
9. Ilies, I. *et al*. Stepping forward through echoes of the past: forecasting with echo state networks. *Short report on the winning entry to the NN3 financial forecasting competition, available online at http://www. neural-forecasting-competition. com/downloads/NN3/methods/27-NN3 Herbert Jaeger report. pdf* 53, 76. (2007).
10. Sacchi, R., Ozturk, M. C., Principe, J. C., Carneiro, A. A., & Da Silva, I. N. Water inflow forecasting using the echo state network: a brazilian case study. In *Proceedings of IJCNN07*, International Joint Conference on Neural, 2403-2408 (2007).
11. Du, C. *et al*. Reservoir computing using dynamic memristors for temporal information processing. *Nat. Commun.* 8, 2204 (2017).
12. Sillin, H. O. *et al*. A theoretical and experimental study of neuromorphic atomic

switch networks for reservoir computing. *Nanotechnology 24*, 384004 (2013).

13. Vandoorne, K. *et al.* Experimental demonstration of reservoir computing on a silicon photonics chip. *Nat. Commun.* 5, 3541 (2014).
14. Torrejon, J. *et al*. Neuromorphic computing with nanoscale spintronic oscillators. *Nature* 547, 428-431 (2017).
15. Chua, L. O. Memristor - the missing circuit element. *IEEE Trans. Circuit Theory* 18, 507–519 (1971).
16. Waser, R. & Aono, M. Nanoionics-based resistive switching memories. *Nat. Mater.* 6, 833–840 (2007).
17. Strukov, D. B., Snider, G. S., Stewart, D. R. & Williams, R. S. The missing memristor found. *Nature* 453, 80–83 (2008).
18. Pershin, Y. V. & Di Ventra, M. Neuromorphic, digital, and quantum computation with memory circuit elements. *Proceedings of the IEEE* 100, 2071–2080 (2012).
19. Jo, S. H. *et al.* Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Lett.* 10, 1297–1301 (2010).
20. Yang, J. J., Strukov, D. B. & Stewart, D. R. Memristive devices for computing. *Nat. Nanotechnol.* 8, 13–24 (2013).
21. Prezioso, M. *et al.* Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* 521, 61–64 (2015).
22. Sheridan, P. M. *et al.* Sparse coding with memristor networks. *Nat. Nanotechnol.* 12, 784-789 (2017).
23. Appeltant, L. *et al.* Information processing using a single dynamical node as complex system. *Nat. Commun.* 2, 468 (2011).
24. Larger, L. *et al.* Photonic information processing beyond Turing: an optoelectronic implementation of reservoir computing. *Opt. Express* 20, 3241–3249 (2012).
25. Brunner, D., Soriano, M. C., Mirasso, C. R., & Fischer, I. Parallel photonic information processing at gigabyte per second data rates using transient states. *Nat. Commun.* 4, 1364 (2013).
26. Antonik, P., Haelterman, M., & Massar, S. Brain-inspired photonic signal processor for generating periodic patterns and emulating chaotic systems. *Phys. Rev. Appl.* 7, 054014 (2017).

27. Box, G. E., & Pierce, D. A. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *J.Am. Stat. Assoc.* 65, 1509-1526 (1970).
28. Said, S. E., & Dickey, D. A. Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika* 71, 599-607 (1984).
29. Kim, K. J. Financial time series forecasting using support vector machines. *Neurocomputing* 55, 307-319 (2003).
30. Kuremoto, T., Kimura, S., Kobayashi, K., & Obayashi, M. Time series forecasting using a deep belief network with restricted Boltzmann machines. *Neurocomputing* 137, 47-56 (2014).
31. Zhang, G. P. An investigation of neural networks for linear time-series forecasting. *Computers & Operations Research* 28, 1183-1202 (2001).
32. Connor, J. T., Martin, R. D., & Atlas, L. E. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks* 5, 240-254 (1994).
33. Assaad, M., Boné, R., & Cardot, H. A new boosting algorithm for improved time-series forecasting with recurrent neural networks. *Information Fusion* 9, 41-55 (2008).
34. Mirikitani, D. T., & Nikolaev, N. Recursive bayesian recurrent neural networks for time-series modeling. *IEEE Transactions on Neural Networks* 21, 262-274 (2009).
35. Yang, Y. *et al.* Observation of conducting filament growth in nanoscale resistive memories. *Nat. Commun.* 3, 732 (2012).
36. Chang, T., Jo, S.-H. H. & Lu, W. Short-term memory to long-term memory transition in a nanoscale memristor. *ACS Nano* 5, 7669–7676 (2011).
37. Wang, Z. *et al*. Memristors with diffusive dynamics as synaptic emulators for neuromorphic computing. *Nat. Mater.* 16, 101-108 (2017).
38. Texas Instruments-Developed 46-Word Speaker-Dependent Isolated Word Corpus (TI46), September 1991, NIST Speech Disc 7-1.1 (1 disc).
39. Lyon, R. F. A computational model of filtering, detection, and compression in the cochlea. *Proc. IEEE-ICASSP'82* 7, 1282–1285 (1982).
40. Bennett, C. H., Querlioz, D., & Klein, J. O. Spatio-temporal learning with arrays of analog nanosynapses. in *2017 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)* 125-130 (2017).

41. Frazier, C., & Kockelman, K. M. Chaos theory and transportation systems: Instructive example. *Transportation Research Record* 1897, 9-17 (2004).
42. Mackey, M. C., & Glass, L. Oscillation and chaos in physiological control systems. *Science* 197, 287-289 (1977).
43. Farmer, J. D. Chaotic attractors of an infinite-dimensional dynamical system. *Physica D:Nonlinear Phenomena* 4, 366-393 (1982).
44. Cai, F. *et al.* A fully integrated reprogrammable memristor-CMOS system for efficient multiply-accumulate operations. *Nat. Electron.* 2, 290-299 (2019)
45. Tanaka, G. *et al.* Recent advances in physical reservoir computing: a review. *Neural Networks* 115, 100-123 (2019).
46. Zhu, X., Wang, Q., & Lu, W. D. Memristor networks for real-time neural activity analysis. *Nature communications*, 11(1), 1-9 (2020).

Chapter 3
Hierarchical Architecture in Reservoir Computing Systems

3.1 Introduction

Due to the dramatically increased computing power, advances in algorithms, and abundantly available data, machine learning, especially deep learning [1] has been successfully applied in a wide range of artificial intelligence domains recently, from computer vision to natural language processing. Depending on the tasks, different deep learning architectures have been optimized. For instance, convolutional neural networks (CNNs) [2]-[4] are mainly used for static data processing such as image recognition, as individual input images are independent so that there is no need to capture the correlation between the input images. On the other hand, temporal data processing such as time-series prediction and speech recognition is typically performed by recurrent neural networks (RNNs) [5]-[7] since the recurrent connections allow the network to capture temporal features in the sequential input data.

Although the cyclic connections in RNNs are useful to process temporal dynamics of input signals in RNNs, these connections make training RNNs very computationally expensive and difficult. In error backpropagation through time, which is a standard method to train the RNNs, calculating the gradients of the error with respect to weights involves a large amount of computation process as the current states depend on not only the current inputs but also the previous states and inputs. Furthermore, when applying the chain rule to compute the gradients, repeated multiplying the derivatives of the current states with respect to the previous states can cause vanishing gradient or exploding gradient problems [8] that make it difficult to find optimal

weights. Long short-term memory (LSTM) [6] is developed to deal with the problems by enforcing constant error flow through internal states of LSTM units. As the weight matrices of the internal gates are trained by gradient-based learning algorithms, LSTM networks have been applied in a broad range of applications by fine-tuning the network for a given task, but nevertheless still requires significant computation costs to train the weight matrices.

Reservoir computing (RC) [7], [9] is proposed to reduce the training cost and mitigate the difficulty of the training process of RNNs. Since the expensive cost and difficulty of training RNNs come from the attempt to control the recurrent connections, RC systems avoid this problem by tuning only the linear output layer (often called the readout layer) instead of training every weight in the recurrent connections of the main RNN body. In RC, the recurrent networks called 'reservoirs' are randomly initialized and fixed during the training process. The reservoir needs to offer fading memory property and should nonlinearly map the temporal inputs into a high-dimensional feature space, represented by the states of the nodes forming the reservoir. Due to the recurrent connections in the reservoir, this nonlinear mapping also involves time. Afterwards, the initial complex inputs can become linearly separable in the new, high dimensional reservoir state space. Software-based RC systems have achieved state-of-the-art performance for tasks such as speech recognition [10] and showed superior ability to forecast large spatiotemporally chaotic systems [11]. Recently, hardware implementations of RC systems have paved the way for applying RC systems to real-time operating hardware systems in power-constrained environments [12],[13].

With the successful implementation of RC algorithms and hardware, several methods [14]-[18] to design the reservoir have been proposed to improve the performance of RC systems. A conventional reservoir is initialized randomly with a single set of hyperparameters such as

input scaling and spectral radius, such that the states of the reservoir nodes are coupled so strongly that it is difficult to process data with different time scales. Decoupled echo state networks (ESNs) [15] can break the coupling between the nodes by dividing the reservoir into decoupled sub-reservoirs and by introducing lateral inhibition unit between the sub-reservoirs. Unlike the conventional random process to build the reservoir, dynamic methods can also be used to generate reservoirs such as the simple cycle reservoir concept [16]. The deterministically constructed reservoirs not only can achieve comparable performance to the conventional randomized reservoir but can also allow reservoir properties to be analyzed in a more comprehensive manner. Additionally, reservoirs using a single physical nonlinear node subjected to delayed feedback [17] have been proposed to allow hardware-friendly implementation of deterministic reservoirs, similar to the simple cycle reservoir concept used in power-efficient hardware systems.

Besides the efforts to improve the single reservoir structure, several studies have been conducted to apply hierarchical structures to RC systems. Early attempts [18], [19] to introduce an architectural modification to the reservoir included adding feedforward or static layers in the reservoir to amplify the non-linearity through additional nonlinear mapping of outputs from the recurrent part, and to overcome the tradeoff between short-term memory and non-linearity of a single reservoir structure. With great success of hierarchical architectures in convolutional neural networks [2]-[4], several works [20]-[22] have proposed deep reservoir architectures by stacking several sub-reservoirs to capture multiscale dynamics and to increase the richness of reservoir. MESM [20] and DeepESN [21], which stacks the sub-reservoirs directly, and DeePr-ESN [22], which stacks the reservoirs and the encoders alternatively (requiring extra training process), can provide robust prediction performance and effectively capture rich multiscale dynamics.

However, previous studies on MSEM and DeepESN tried to find the optimal hyperparameters manually and analyze only general properties of the proposed architectures, without providing a comprehensive analysis of the effect of stacking the sub-reservoirs.

In this chapter, we show that deep RC systems that stack sub-reservoirs vertically improve the system performance for a broad range of tasks, by analyzing the distribution of node states in each sub-reservoir and the frequency spectrum captured by each sub-reservoir. To focus on the effects of hierarchical architectures, three different structures based on ESN are compared: Shallow ESN, which has a single reservoir, Wide ESN, which has independent sub-reservoirs in parallel, and Deep ESN, which stacks the sub-reservoirs in series. As the optimal hyperparameters for each structure may be different from each other, a genetic algorithm is used to obtain the best performance of each structure by individually finding the optimal hyperparameters. Moreover, we find that there is a tradeoff between the number of sub-reservoirs and the size of the sub-reservoirs, when the size of the readout network is fixed.

3.2 Architecture

A conventional ESN consists of three basic components: an input layer, a recurrent layer (*i.e.* the reservoir), and an output layer (called the readout layer). The reservoir state, which is the collection of the node states in the reservoir, depends on the input signal and the previous reservoir state. Without losing generality, in our study we choose the following equation to describe the reservoir state update:

$$x(t) = (1 - \alpha) \times x(t-1) + \alpha \times \tanh\left(W_{in}u(t) + W_{res}x(t-1)\right) \quad (3\text{-}1)$$

where $u(t) \in \mathbb{R}^{N_U}$, $x(t) \in \mathbb{R}^{N_R}$ denote respectively the input and the reservoir state at time t, $W_{in} \in \mathbb{R}^{N_R \times N_U}$ is the input-to-reservoir weight matrix, $W_{res} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent reservoir weight matrix, $\tanh()$ is the element-wise applied hyperbolic tangent activation

function, and $\alpha \in (0,1]$ is a leaky rate. The weight values in W_{in} is chosen from a uniform distribution over [-*IS*, *IS*], where *IS* is an input scaling factor. W_{res} is rescaled from the randomly generated W, following:

$$W_{res} = SR \times \frac{W}{\lambda_{max}(W)} \tag{3-2}$$

where *SR* is the spectral radius, W is chosen from a uniform distribution over [-1,1], and $\lambda_{max}(W)$ is the largest eigenvalue of matrix W. *IS*, *SR*, and α are the important hyperparameters affecting the reservoir properties. Typically, *IS* determines how nonlinear the node states are: a small *IS* makes the reservoir nodes operate around the origin where the $\tanh()$ activation function is virtually linear, while a more nonlinear transformation can be achieved with a large *IS* that lets the reservoir nodes operate near 1 or -1. *SR* affects the stability of the node states and should be less than unity to ensure the echo state property, which is a central characteristic of the RC system that states the reservoir state is uniquely defined by the fading history of the input signal. α is related to the speed of the reservoir update dynamics. A small α means the reservoir nodes depend more on the previous reservoir state than on the current input, resulting in a slow update speed. In our approach, these three hyperparameters for each sub-reservoir will be optimized to minimize the error by using the genetic algorithm, as discussed later.

The output of the ESN is a linear combination of the node states:

$$y(t) = W_{out}x(t) \tag{3-3}$$

where $y(t) \in \mathbb{R}^{N_Y}$ denotes the output at time t, and $W_{out} \in \mathbb{R}^{N_Y \times N_R}$ is the reservoir-to-output weight matrix. In contrast to the fixed weight matrices W_{in} and W_{res} in the reservoir, W_{out} is trained by ridge regression, also known as regression with Tikhonov regularization.

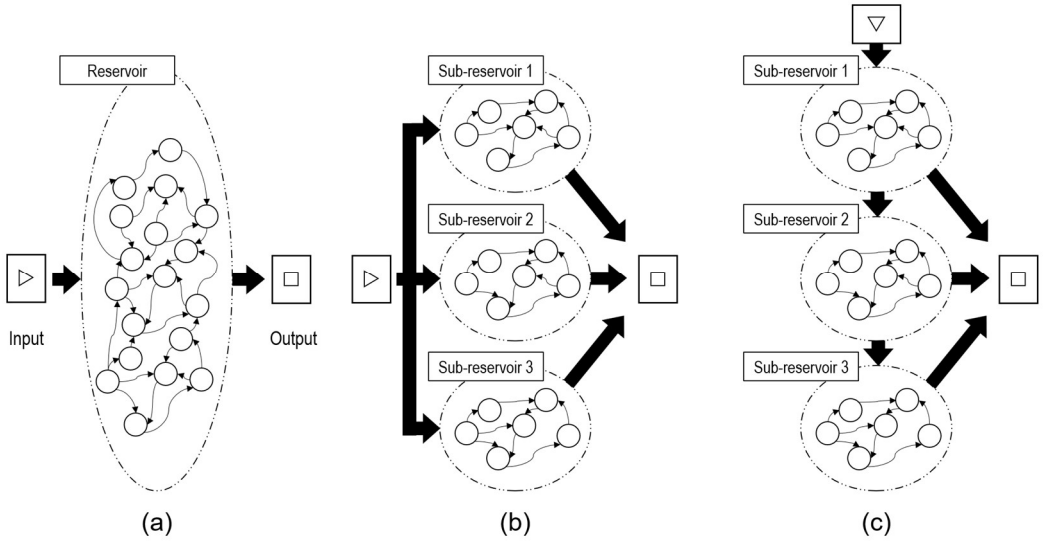

Figure 3-1 Reservoir architectures analyzed in this study. (a) Shallow ESN. (b) Wide ESN. (c) Deep ESN.

Figure 3-1 shows the three different ESN structures used to analyze the architectures, namely, Shallow ESN, Wide ESN, and Deep ESN. Shallow ESN is a conventional single reservoir structure, which is governed by equation (3-1)-(3-3). Wide ESN has independent sub-reservoirs in parallel, which receive the input signals independently and have no interaction between the sub-reservoirs. When the sub-reservoirs have different hyperparameters, each sub-reservoir will capture different temporal dynamics from the input signals, and thus Wide ESN may offer better performance than Shallow ESN. The equations of the Wide ESN are following:

$$x^{(l)}(t) = \left(1 - \alpha^{(l)}\right) \times x^{(l)}(t-1) + \alpha^{(l)} \times \tanh\left(W_{in}^{(l)} u(t) + W_{res}^{(l)} x^{(l)}(t-1)\right) \quad (3\text{-}4)$$

$$y(t) = W_{out}[x^{(1)}(t); x^{(2)}(t); \cdots; x^{(N_L-1)}(t)] \quad (3\text{-}5)$$

where $x^{(l)}(t) \in \mathbb{R}^{N_R}$ denotes the reservoir state of sub-reservoir l at time t, $W_{in}^{(l)} \in \mathbb{R}^{N_R \times N_U}$ is the input-to-reservoir weight matrix of sub-reservoir l, $W_{res}^{(l)} \in \mathbb{R}^{N_R \times N_R}$ is the recurrent reservoir weight matrix of sub-reservoir l, $\alpha^{(l)} \in (0,1]$ is the leaky rate of sub-reservoir l, $W_{out}^{(l)} \in \mathbb{R}^{N_Y \times N_L N_R}$ is the reservoir-to-output weight matrix, and $[\cdot\,; \cdots\,; \cdot]$ stands for vertical vector concatenation.

Deep ESN is the hierarchical ESN structure that stacks the sub-reservoirs in series. Only the first sub-reservoir can see the input signals, and the subsequent sub-reservoirs receive data from the output of the previous sub-reservoir, in the form of linear combination of the previous sub-reservoir's node states. If stacking the sub-reservoirs improves the system's capability in processing temporal information, Deep ESN will outperform both Shallow ESN and Wide ESN. The equations of Deep ESN are following:

$$x^{(l)}(t) = \begin{cases} \left(1-\alpha^{(l)}\right) \times x^{(l)}(t-1) + \alpha^{(l)} \times \tanh\left(W_{in}^{(l)} u(t) + W_{res}^{(l)} x^{(l)}(t-1)\right) \; if \; l = 1 \\ \left(1-\alpha^{(l)}\right) \times x^{(l)}(t-1) + \alpha^{(l)} \times \tanh\left(W_{in}^{(l)} x^{(l-1)}(t) + W_{res}^{(l)} x^{(l)}(t-1)\right) \; if \; l > 1 \end{cases} \tag{3-6}$$

$$y(t) = W_{out}[x^{(1)}(t); x^{(2)}(t); \cdots; x^{(N_L-1)}(t)] \tag{3-7}$$

where $x^{(l)}(t) \in \mathbb{R}^{N_R}$ denotes the reservoir state of sub-reservoir l at time t, $W_{in}^{(1)} \in \mathbb{R}^{N_R \times N_U}$ is the input-to-reservoir weight matrix for the first sub-reservoir, $W_{in}^{(l)} \in \mathbb{R}^{N_R \times N_R}$ is the inter-reservoirs weight matrix from sub-reservoir $l-1$ to sub-reservoir l, $W_{res}^{(l)} \in \mathbb{R}^{N_R \times N_R}$ is the intra-reservoir weight matrix of sub-reservoir l, $\alpha^{(l)} \in (0,1]$ is the leaky rate of sub-reservoir l, $W_{out}^{(l)} \in \mathbb{R}^{N_Y \times N_L N_R}$ is the reservoir-to-output weight matrix, and $[\cdot\,; \cdots; \cdot]$ stands for vertical vector concatenation.

The main function of the reservoir in an RC system is to nonlinearly map the sequential input signals onto the reservoir state, which allows certain tasks such as classification and prediction to be processed by the readout network using a linear combination of the reservoir node states. Thus, the quality of the reservoir can be measured in two aspects: how well the reservoir can nonlinearly transform the input signal; and how diverse the temporal dynamics the reservoir can capture. As the performance of the RC system is dominantly affected by how the reservoirs are designed, the hyperparameters for building the reservoirs should be carefully

selected to ensure the designed reservoirs produce the best result in given evaluation criteria. To avoid the node states from being easily saturated and ensure the echo state property, *IS*, *SR*, and α are set to be less than 1 as shown in Table 3-1. Grid search is widely used to analyze the influence of hyperparameters on the RC performance, and is a simple way to design the optimal single reservoir system since the hyperparameter space of a single reservoir is not large. However, when the RC system has several sub-reservoirs like Wide ESN and Deep ESN, the grid search is not an efficient optimization method because the computing cost to test every point in the hyperparameter space will increase exponentially as a function of the dimension of the space. Thus, in this paper, a genetic algorithm [24], one of the widely used evolutionary optimization methods, is used to find the optimal set of hyperparameters for a given RC architecture.

Hyperparameters for reservoir	Symbol	Value
Input scaling	*IS*	(0, 1]
Spectral radius	*SR*	(0, 1]
Leaky rate	α	(0, 1]

Parameters for the genetic algorithm	Value
Generation	1000
Population size	15
Crossover rate	0.33
Mutation rate	0.33

Table 3-1 Hyperparameters for reservoir and parameters for the genetic algorithm.

Genetic algorithms can efficiently explore the large hyperparameter space through the metaheuristic optimization process inspired by biological evolutions. Specifically, a population of sets of hyperparameters for a given RC architecture is randomly generated, and will evolve toward better reservoir structures through an iterative process, where the population with each

iteration is called a generation. In each generation, two sets among the population are randomly selected, and two RC systems based on the selected sets are evaluated for a given task by the normalized root mean squared error (NRMSE):

$$NRMSE = \sqrt{\frac{\sum_{t=1}^{T}[y_{target}(t)-y(t)]^2}{\sum_{t=1}^{T}[y_{target}(t)-\overline{y_{target}}]^2}} \quad (3\text{-}8)$$

where $y_{target}(t)$ and $y(t)$ denote the target signal and the output from the RC system at time t, $\overline{y_{target}}$ is the mean of the target signal, and T is the number of time steps in the test sequence. The loser set, which has higher error than the other, winner set, goes through two genetic operations: crossover, which replaces the part of the loser with the part of the winner based on a crossover rate; and mutation, which changes part of the loser to random values based on a mutation rate. The modified loser replaces the original loser in the population, and this process is repeated until the maximum number of generations is reached. The parameters for the genetic algorithm are summarized in Table 3-1.

We evaluate RC systems with different architectures on three different time series data: 10^{th} order nonlinear autoregressive moving average (NARMA10) [24], Santa Fe Laser time series [25], and Mackey-Glass time series [26]. The NARMA10 system is widely used to test ESN models since it needs both nonlinear transformation and memory, which are the main characteristics of RC systems [16]-[18], [22]. The output of the NARMA10 system is computed as following:

$$y(t+1) = 0.3y(t) + 0.05y(t)(\sum_{i=}^{9} y(t-i)) + 1.5u(t-9)u(t) + 0.1 \quad (3\text{-}9)$$

where $u(t)$ is a random input at time step t, generated from a uniform distribution over [0,0.5], and $y(t)$ is the output at time step t. The readout network is trained to predict $y(t+1)$ from the reservoir state and $u(t)$. The lengths of the training and test sequences are 3000 and

1000, respectively. The first 100 reservoir state values, which are called initial transient, are not used to train the readout network or to predict the next value in the test stage to avoid the influence of initial states. The Santa Fe Laser time series is a one-step ahead prediction on the data obtained by sampling the intensity of a far-infrared laser in a chaotic regime [16], [18]. The data are normalized in the interval [0,1], and the lengths of training, test and initial transient sequences are 3000, 1000, and 100, respectively. The Mackey-Glass time series is a standard benchmark for chaotic time series prediction task [7], [12], [18], [20], [22]. The time series is defined by the following differential equation:

$$\frac{dy(t)}{dt} = \frac{0.2y(t-\tau)}{1+y(t-\tau)^{10}} - 0.1y(t) \tag{3-10}$$

where $y(t)$ is the output at time step t, and τ is the time delay. When $\tau > 16.8$, the system has a chaotic attractor. In this study, we set $\tau = 17$, and define the task as to perform 84-step-ahead direct prediction, *i.e.*, the readout network is trained to predict $y(t + 84)$ from the reservoir state when $y(t)$ is given. The lengths of training, test and initial transient sequences are 1000, 1000, and 100, respectively.

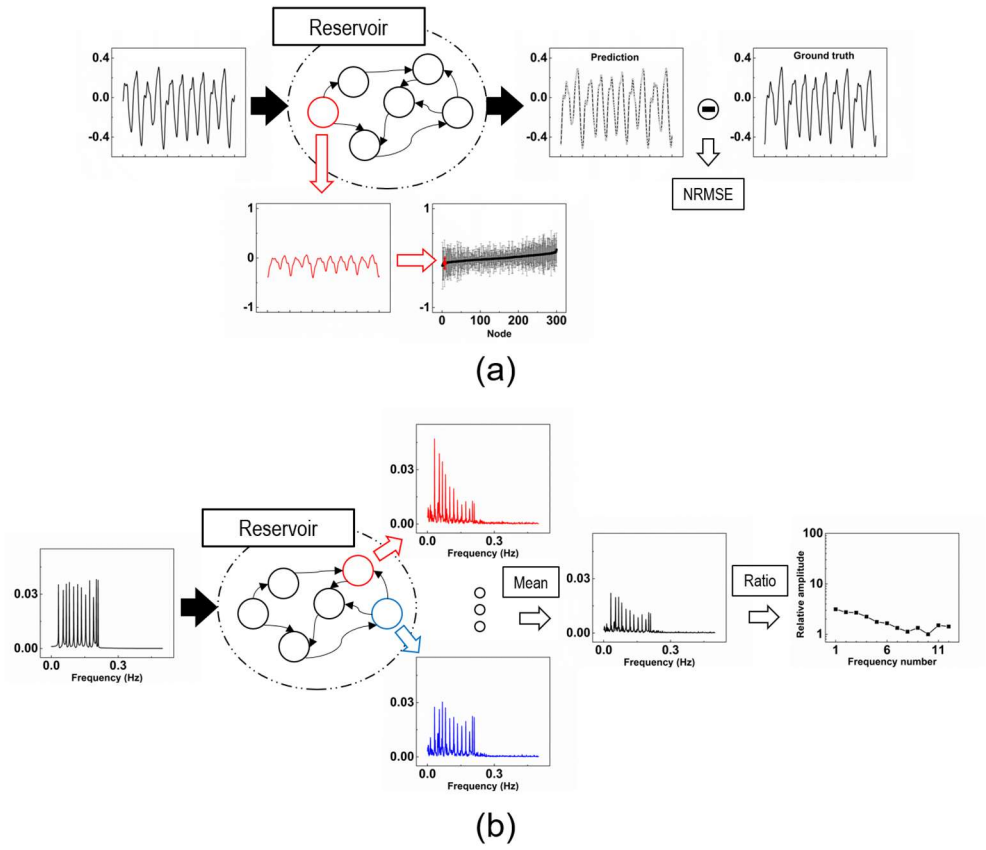

Figure 3-2 Schematic of the evaluation methods. (a) NRMSE and the distribution of node states. NRMSE is measured from the difference between the prediction of the test sequence and the ground truth. The temporal response (red line) of a node (*e.g.* represented by the red circle) is recorded for a test sequence. The mean and standard variation of the states from each node in a sub-reservoir are plotted and sorted in ascending order by the mean value. (b) The frequency spectrum of the node states. When the MSO12 time series is applied to the reservoir, the FFT of the states from each node is calculated (*e.g.* red, blue lines) and averaged (black line) on a sub-reservoir-by-sub-reservoir basis. The peak values of the 12 different frequency components are normalized by the minimum among them.

In the genetic algorithm, the NRMSE is used to determine which hyperparameter set is the winner, as shown in Figure 3-2 (a). Additionally, the distribution and frequency spectrum of the node states are measured to study the properties of different reservoir structures and to explain why a specific reservoir structure performs better than others. To get better performance, the RC system should nonlinearly transform the input signals into a high-dimensional space and allow the readout network to linearly separate the corresponding reservoir states. Although it is easy to decide whether the transformation is linear or nonlinear, it is difficult to quantify the degree of nonlinearity in a single evaluation term. Here, the distribution of node states is

analyzed since nodes whose states are far away from the origin indicate these nodes go through a more nonlinear transformation than the nodes whose states are close to the origin. After finding the optimal hyperparameter set for a given reservoir structure and task, the node states of the test sequence are recorded, and the mean and standard deviation of the node states are calculated as shown in Figure 3-2 (a). To clearly visualize the difference of each sub-reservoir, the node states are grouped according to the sub-reservoir they belong to, and the means of node states in the same sub-reservoir are sorted in ascending order.

Another important property of the RC system is the ability to capture diverse temporal dynamics with different time scales. The diversity of temporal dynamics can be analyzed by examining the differences of the frequency components each sub-reservoir captures. The multiple superimposed oscillator (MSO) task [15], [27], [28] is used to analyze the multiple time-scales processing ability of the different reservoir structures. The MSO12 time series, given by a sum of sinusoidal functions, is used:

$$u(t) = \sum_{i=1}^{12} \sin(\varphi_i t) \tag{3-11}$$

where φ_i determines the frequency of the i-th sinusoidal function. The φ_i coefficients are set as in [28], *i.e.* $\varphi_1 = 0.2, \varphi_2 = 0.331, \varphi_3 = 0.42, \varphi_4 = 0.51, \varphi_5 = 0.63, \varphi_6 = 0.74, \varphi_7 = 0.85, \varphi_8 = 0.97, \varphi_9 = 1.08, \varphi_{10} = 1.19, \varphi_{11} = 1.27, \varphi_{12} = 1.32$. As shown in Figure 3-2 (b), the frequency spectra of each node state after being excited by the MSO12 input are calculated by the Fast Fourier Transformation (FFT). The FFT values are averaged on a sub-reservoir-by-sub-reservoir basis. To emphasize which frequency components are mainly captured by the specific sub-reservoir, the peak values of the 12 different frequency components are normalized by the minimum among them.

3.3 Effects of Hierarchical Reservoir Computing Systems

We will compare the performance of the three different reservoir structures on time series prediction tasks, and discuss the effects of stacking sub-reservoirs on the properties of the RC system.

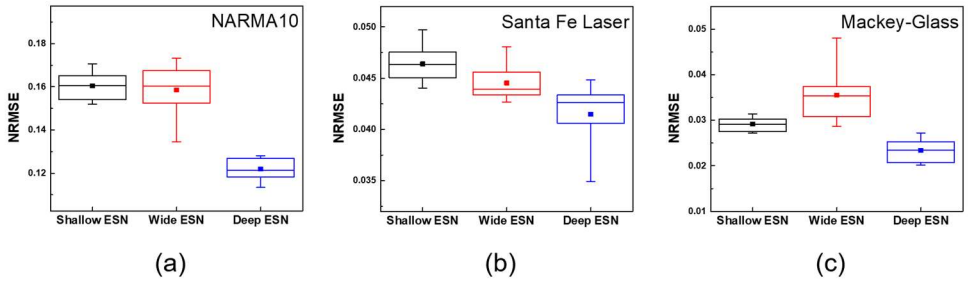

Figure 3-3 NRMSE of Shallow ESN, Wide ESN and Deep ESN for (a) NARMA10 system. (b) Santa Fe Laser time series, and (c) Mackey-Glass time series tasks. The boxes show the 25th, 50th and 75th percentiles for the NRMSE data measured from 10 random reservoir initializations for each case, along with the mean (dot). The whiskers represent the minimum and maximum.

As the performance of an RC system is strongly affected by the size of the readout network, we first fixed the size of the readout network to 300, *i.e.* corresponding to a single reservoir with 300 nodes (Shallow ESN), three independent sub-reservoirs with 100 nodes each (Wide ESN), and three stacked sub-reservoirs with 100 nodes each (Deep ESN). Figure 3-3 (a), (b) and (c) show the NRMSEs of the three different reservoir structures when performing NARMA10, Santa Fe Laser and Mackey-Glass time series tasks, respectively. The box charts show the results from 10 different randomly generated reservoir systems with the hyperparameter set optimized by the genetic algorithm. In all cases, Deep ESN shows the best performance among the different reservoir structures. In contrast, compared to Shallow ESN, Wide ESN shows slightly better performance for the first two tasks but worse performance for Mackey-Glass time series.

3.3.1. Enhanced Nonlinearity of Data Transformation

To find out why Deep ESN performs better than others, the distributions of node states are plotted in Figure 3-4. The nodes of Deep ESN in the third (*i.e.* last) sub-reservoir show a wider spectrum than not only the nodes of Deep ESN in the first two sub-reservoirs, but also the nodes in every sub-reservoir in other tested structures. This indicates that the nodes of the third sub-reservoir in the Deep ESN go through a higher degree of nonlinear transformation. Considering the nodes of Deep ESN in the deeper sub-reservoir have a wide spectrum than the nodes in the previous sub-reservoirs, the wide range of node states in Deep ESN can be attributed to the stacked sub-reservoir structure in the Deep ESN.

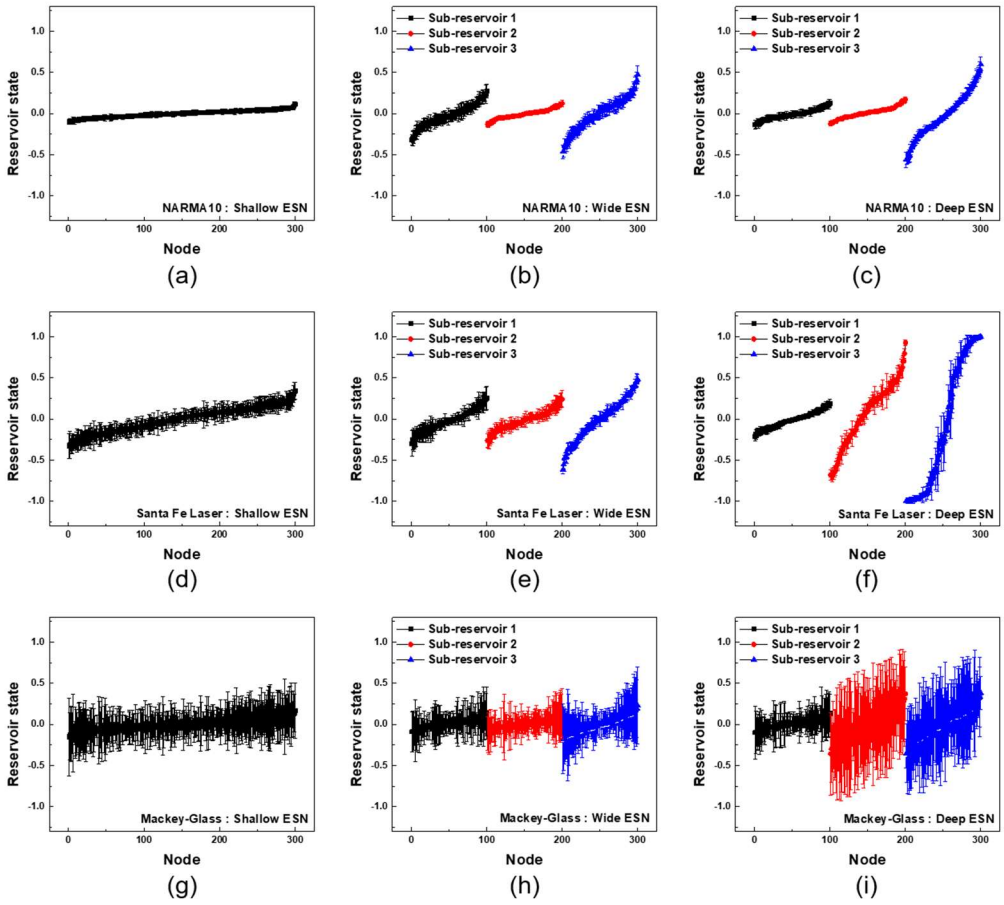

Figure 3-4 Distribution of the mean and standard deviation of the node states. (a) Shallow ESN, (b) Wide ESN, (c) Deep ESN for the NARMA10 system. (d) Shallow ESN, (e) Wide ESN, (f) Deep ESN for Santa Fe Laser time series. (g) Shallow ESN, (h)

Wide ESN, (i) Deep ESN for Mackey-Glass time series. The node states in the same sub-reservoir are sorted in ascending order of their mean values.

Besides the absolute magnitude of the node states range, the diversity of the ranges among the sub-reservoirs is also a good factor to examine the quality of the reservoir structure, because having different ranges between the sub-reservoirs means that the node states in each sub-reservoir have different degrees of nonlinear transformation so that the RC system can capture broader temporal dynamics. It can be seen from Figure 3-4 that Wide ESN has a slightly larger range and diversity of node states than Shallow ESN, but the enhancement is not as significant as Deep ESN. Thus, a more efficient way to achieve a high degree of diverse, nonlinear transformation is stacking the sub-reservoirs in series rather than building independent sub-reservoirs in parallel.

3.3.2. Expanded Diversity of Captured Temporal Information

Another method to estimate the temporal dynamics captured by each sub-reservoir is the frequency spectrum of node states, which shows which frequency components are picked up by each sub-reservoir. In our study, due to the simple but distinguishable frequency spectrum of the MSO12 input signal, it makes it easier to recognize the distinct frequency spectra among the reservoir structures, as shown in Figure 3-5. While Wide ESN shows similar frequency spectra among the sub-reservoirs, the sub-reservoirs in Deep ESN show significantly different frequency spectra. Specifically, the sub-reservoirs in the late stages of Deep ESN tends to capture the low frequency component rather than the high frequency component. Since each sub-reservoir offers a kind of integrating function through its fading memory property that maps the input sequence to the reservoir state, which is then used as inputs to the next stage, lower frequency components are more efficiently captured in the later stages of Deep ESN. Analogous to the fact that the layers in the late stage of CNNs capture more abstract or global information, *i.e.* lower spatial

frequency features of the input signal, the sub-reservoirs in the late stages of Deep ESN thus emphasizes on the low temporal frequency component or the global (long term) temporal information. We also note that, although both CNNs and Deep ESNs put similar roles to the layers in the late stage, they utilize the early stage layers significantly differently. Typically a CNN only uses the outcome of the last layer since its goal is to derive an abstract representation of the input signal, such as identifying the object, and the outcome of the last layer is typically enough for the task. On the other hand, Deep ESN needs to utilize the reservoir states of every sub-reservoir, since a wide range of temporal information from low to high frequency components is usually required to perform the temporal data processing tasks such as predicting the next value of the time series data.

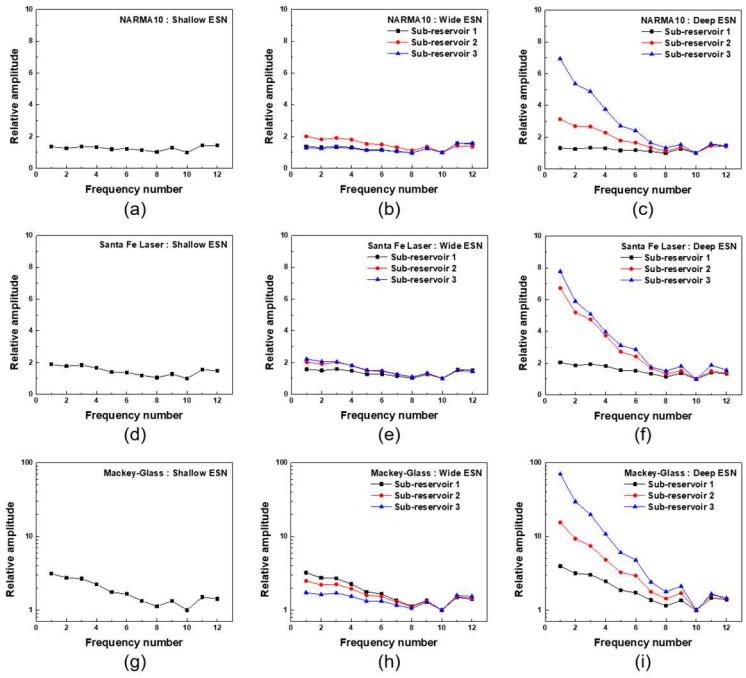

Figure 3-5 Frequency spectrum of node states. (a) Shallow ESN, (b) Wide ESN, (c) Deep ESN for the NARMA10 system. (d) Shallow ESN, (e) Wide ESN, (f) Deep ESN for Santa Fe Laser time series. (g) Shallow ESN, (h) Wide ESN, (i) Deep ESN for Mackey-Glass time series.

3.4 Optimization of Hierarchical Structure

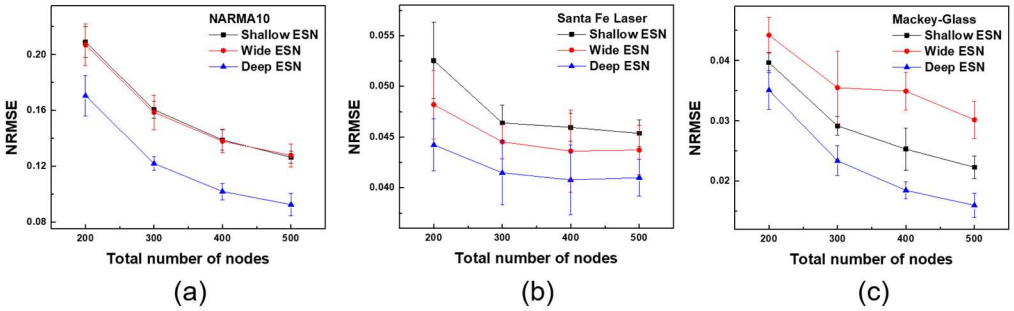

Figure 3-6 NRMSE of Shallow ESN, Wide ESN and Deep ESN with different total numbers of nodes in the system, for (a) NARMA10 system, (b) Santa Fe Laser time series, and (c) Mackey-Glass time series. The mean and standard variations are plotted for each case, based on 10 different random initializations of the reservoirs following the optimized hyperparameter sets.

The simplest way to improve the performance of the RC system is increasing the total size of the reservoir, since larger reservoirs with more diverse internal connections should capture richer temporal dynamics. If stacking the sub-reservoirs in series helps the RC system to predict the time series data more accurately, Deep ESN should still outperform the others when the total size of the reservoir increases. To expand the size of the reservoir, extra sub-reservoirs with 100 nodes are serially added to the output of the last sub-reservoir in Deep ESN, or connected as parallel, independent sub-reservoirs in Wide ESN. Shallow ESN has still a single reservoir, but the size of the reservoir increases to match the total size of the expanded Deep ESN or Wide ESN. Figure 3-6 (a), (b) and (c) show the NRMSEs of the three different reservoir structures on NARMA10, Santa Fe Laser and Mackey-Glass time series, respectively, as the total size of the reservoir increases from 200 to 500. Again, Deep ESN shows the lowest NRMSE among the tests, and Wide ESN is better than Shallow ESN in most cases, which are consistent with the findings for the case with 300 total nodes. In general, as the total number of nodes in the reservoir increases, the performance of the RC system improves no matter what reservoir structure is used, but the improvement in prediction error tends to be saturated. Moreover, when

considering both the performance measured in NRMSE and the computation cost measured in the size of weight matrices, the performance improvement by simply increasing the reservoir size may actually be lower than the increased expense needed to compute the reservoir states and train the readout network.

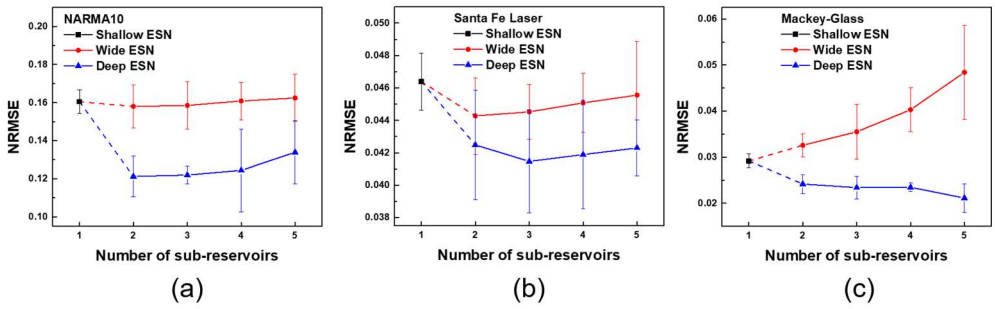

Figure 3-7 NRMSE for Shallow ESN, Wide ESN and Deep ESN with different number of sub-reservoirs, for (a) NARMA10 system, (b) Santa Fe Laser time series, and (c) Mackey-Glass time series. The mean and standard variations are plotted for each case, based on 10 different random initializations of the reservoirs following the optimized hyperparameter sets.

Instead of adding extra sub-reservoirs to Wide ESN or Deep ESN, we can instead fix the total size of the reservoir (thus keeping the total compute cost low) and vary the number and size of sub-reservoirs. While Shallow ESN always has a single reservoir with 300 nodes, we varied the number of sub-reservoirs in Wide ESN and Deep ESN from 2 to 5 while keeping the total number of nodes constant at 300. Due to the fixed total node size, the cost to train the readout network will be identical for the three different reservoir structures, but the expense to calculate the node states will be reduced as the weight matrix size in Wide ESN or Deep ESN is reduced quadratically as the sub-reservoir size is reduced, while the number of matrices increases linearly. Figure 3-7 (a), (b) and (c) show the NRMSEs of the three different reservoir structures on NARMA10, Santa Fe Laser and Mackey-Glass time series tasks, respectively, as the number of sub-reservoirs increases from 2 to 5 while keeping the total node size at 300. Again, Deep ESN outperforms the others for all tasks in all cases.

For the NARMA10 and Santa Fe Laser tasks, Wide ESN and Deep ESN show similar trend on the number of sub-reservoirs, *i.e.* the performance is improved when the reservoir consists of 2 or 3 sub-reservoirs, but becomes worse when the reservoir is divided into more than 2 or 3 sub-reservoirs. This can be explained by the tradeoff between the number of sub-reservoirs and the size of each sub-reservoir. Given the fixed total number of nodes, the size of sub-reservoirs is reduced as the number of sub-reservoirs increases. When the reservoir is split into several sub-reservoirs, the temporal dynamics captured by the whole system become more diverse since each sub-reservoir can have different hyperparameters leading to different temporal dynamics. Moreover, stacking the sub-reservoirs enhances the nonlinearity of transformation from the input signal and the diversity of the temporal dynamics captured by each sub-reservoir, as shown in Figure 3-4 and 3-5. However, at the same time, the feature space of each sub-reservoir will shrink due to the reduced size of the sub-reservoirs, *i.e.* the ability of the sub-reservoirs to extract the temporal information will be weakened.

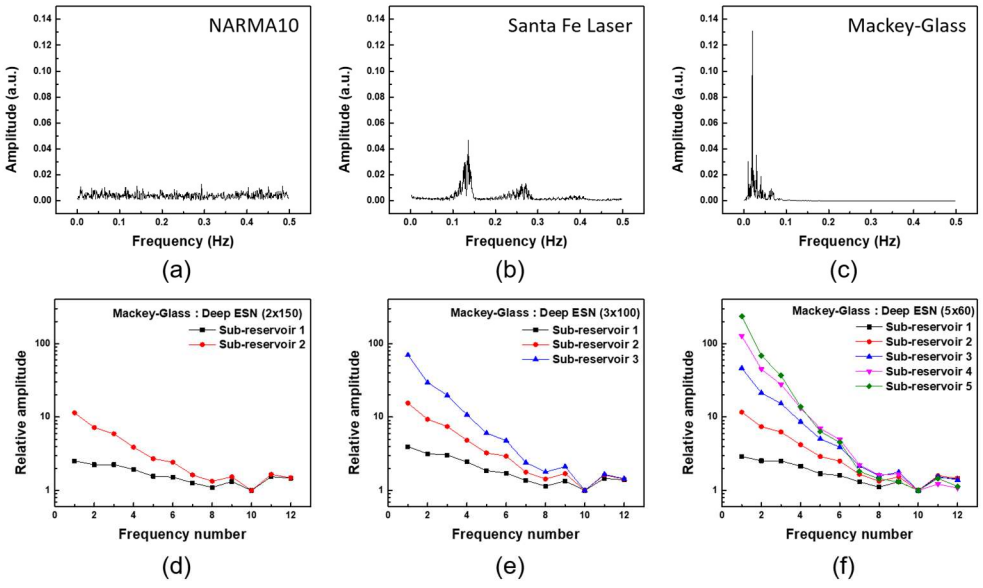

Figure 3-8 Fast Fourier Transform results of the input signal of (a) NARMA10, (b) Santa Fe Laser time series, and (c) Mackey-Glass time series, and frequency spectra of node states of Deep ESN with 2 sub-reservoirs having 150 nodes each (d), 3 sub-reservoirs having 100 nodes each (e), and 5 sub-reservoirs having 60 nodes each (f), for the Mackey-Glass time series problem.

The optimal number and size of the sub-reservoirs depend on both the reservoir structure and the task. For example, the result for Mackey-Glass time series shown in Figure 3-7 (c) is different from the trends observed in Figure 3-7 (a) and (b). As the number of sub-reservoirs increases, Deep ESN performance always improves, but Wide ESN performance becomes worse. This clear dependence on tasks originates from the intrinsic characteristics of each task. Figure 3-8 (a), (b) and (c) show the FFT results of the input signals of NARMA10, Santa Fe Laser and Mackey-Glass time series, respectively. While the output of NARMA10 is governed by the input signal based on equation (3-9), the input of NARMA10 is randomly generated, not affected by the output. Thus, the frequency spectrum of the input signal of NARMA10 has no major frequency dependence. In contrast, Santa Fe Laser and Mackey-Glass time series have clear frequency peaks since in these tasks the input is the previous output, not an independent variable. Importantly, the major peaks of Mackey-Glass time series are located near the low frequency range rather than spreading out evenly, which is the case of Santa Fe Laser time series. As shown in Fig. 8d, e and f, the sub-reservoirs in the late stage of Deep ESN can effectively capture the low frequency components. Especially for Mackey-Glass time series, this strength of Deep ESN can compensate the degradation of the sub-reservoir feature space. Thus, the NRMSE of Deep ESN for Mackey-Glass time series improves as the number of sub-reservoirs increases. On the other hand, Wide ESN performance for Mackey-Glass time series becomes worse when having more sub-reservoirs.

In the cases of NARMA10 and Santa Fe Laser time series, the improvement comes from having more diverse temporal dynamics rather than extracting more information from the low frequency component, as both tasks do not have major peaks in the low frequency range. Thus, although the Deep ESN still outperforms Shallow ESN and Wide ESN, the benefits in capturing

more low-frequency signals are not fully utilized and the Deep ESN performance degrades when the sub-reservoir size becomes too small. Following similar reasoning, even small differences among the independent sub-reservoirs of Wide ESN contribute to the improvement of Wide ESN over Shallow ESN for these two tasks.

Finally, we compare our Deep ESN with other networks. For the task of predicting Mackey-Glass time series 84 step ahead, the Deep ESN (NRMSE = 0.021) outperforms reported performance from feed-forward neural networks (NRMSE = 0.038) [29] and LSTM (NRMSE = 0.470) [30] because of the Deep ESN's ability to capture diverse temporal dynamics. The small number of units and trainable parameters (4 units and 113 trainable parameters, respective) may contribute to the worse reported LSTM performance than the feed-forward neural networks. However, although the performance of LSTM network can be improved by increasing the number of units, the training cost will also increase dramatically because the weight matrix size of internal gates increases quadratically. Moreover, the gradient-based learning algorithm is computationally more expensive than ridge-regression, which is the learning rule commonly used for RC systems. Among reported methods to design RC systems, DeePr-ESN (NRMSE = 9.08E-04) [22] shows better performance than the Deep ESN analyzed here, but DeePr-ESN inserts additional encoder layers between sub-reservoirs, which requires extra training for auto-encoders based on extreme learning machines. The large total size of DeePr-ESN (8 sub-reservoirs with 300 nodes each vs. 5 sub-reservoirs with 60 nodes each for the Deep ESN) is another factor of the better DeePr-ESN performance. However, expanding the size of the RC system in hardware implement is expensive, due to the cost of building the large number of physical interconnections between nodes or devices that grows quadratically with the number of nodes [13]. Thus, it is noteworthy that the Deep ESN concept, by stacking the sub-reservoirs in

series and keeping the sub-reservoir size constant, may be a desirable approach to improve the performance of the RC system since the number of required connections now increases linearly with the total number of nodes.

3.4 Conclusions

The effects of hierarchical reservoir structures on the performance of reservoir computing are investigated. Wide-ESNs that build independent sub-reservoirs in parallel, and Deep-ESNs that stack the sub-reservoirs in series, can capture more diverse temporal dynamics than simply increase the reservoir size. Furthermore, Deep-ESNs not only offers stronger nonlinear transformation of the input to the reservoir states, but also captures more diverse temporal dynamics than Wide-ESNs and Shallow-ESNs. When the total size of the reservoir is fixed, a trade-off may be observed between the number of sub-reservoirs and the size of each sub-reservoir. For tasks where the low frequency signals are important, such as long-term forecasting of time-series such as Mackey-Glass time series, the strength of Deep ESN becomes more evident as the enhanced ability of the sub-reservoirs in the late stage to capture low frequency signals overcomes the effect of reduced feature space of the sub-reservoirs.

The analysis of hierarchical reservoir structures may provide useful insight into designing practical reservoir systems. Especially, when it comes to hardware implementation of RC systems, reducing the number of connections between the nodes by stacking sub-reservoirs offers a promising approach to building large reservoirs in hardware. Enhancement of nonlinearity and diversity from the stacked sub-reservoir structure should be considered and utilized for solving complex tasks such as multivariate systems. Further studies on hyperparameter optimization algorithms can help fine tune the reservoir design for specific tasks,

and provide a more comprehensive understanding of how the specific hyperparameter set affects the overall performance of the RC system.

References

1. LeCun, Y., Bengio, Y., & Hinton, G. Deep learning. *Nature* **521**, 436-444 (2015).
2. Krizhevsky, A., Sutskever, I., & Hinton, G. Imagenet classification with deep convolutional neural networks. *Proc. Adv. Neural Inf. Process. Syst.*1, 4 (2012).
3. He, K., Zhang, X., Ren, S., & Sun, J. Deep residual learning for image recognition. *Proc. CVPR*, 770-778 (2016).
4. Tan, M., & Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:*1905.11946 (2019).
5. Mandic, D. P., & Chambers, J. A. Recurrent Neural Networks for Prediction: Learning Algorithms Architectures and Stability, New York:Wiley (2001).
6. Hochreiter, S., & Schmidhuber, J. Long short-term memory. *Neural Comput.* 9, 1735-1780 (1997).
7. Jaeger, H., & Haas, H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304, 78-80 (2004).
8. Bengio, Y., Simard, P., & Frasconi, P. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Netw.* 5, 157-166 (1994).
9. Lukoševičius, M., & Jaeger, H. Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.* 3, 127-149 (2009).
10. Verstraeten, D., Schrauwen, B., & Stroobandt, D. Reservoir-based techniques for speech recognition. *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 1050-1053 (2006).
11. Pathak, J., Hunt, B., Girvan, M., Lu, Z., & Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Phys. Rev. Lett.* 120, 024102 (2018).
12. Moon, J., Ma, W., Shin, J.H., Cai, F., Du, C., Lee, S.H., & Lu, W.D. Temporal data classification and forecasting using a memristor-based reservoir computing system. *Nature Electronics* 2, 480-487 (2019).
13. Tanaka, G., Yamane, T., Héroux, J. B., Nakane, R., Kanazawa, N., Takeda, S., Numata, H., Nakano, D., & Hirose, A. Recent advances in physical reservoir computing: A review. *Neural Networks* 115, 100-123 (2019).
14. Lukoševičius, M. A Practical Guide to Applying Echo State Networks, Berlin, Germany:Springer, 659-686 (2012).

15. Xue, Y., Yang, L., & Haykin, S. Decoupled echo state networks with lateral inhibition. *Neural Netw.* 20, 365-376 (2007).
16. Rodan, A., & Tino, P. Minimum complexity echo state network. *IEEE Trans. Neural Netw.* 22, 131-144 (2011).
17. Appeltant, L., Soriano, M.C., Van der Sande, G., Danckaert, J., Massar, S., Dambre, J., Schrauwen, B., Mirasso, C.R., & Fischer, I. Information processing using a single dynamical node as complex system. *Nature Commun.* 2, 468-472 (2011).
18. Gallicchio, C., & Micheli, A. Architectural and Markovian factors of echo state networks. *Neural Netw.* 24, 440–456 (2011).
19. Butcher, J., Verstraeten, D., Schrauwen, B., Day, C., & Haycock, P. Reservoir computing and extreme learning machines for non-linear time-series data analysis. *Neural Netw.* 38, 76-89 (2013).
20. Malik, Z. K., Hussain, A., & Wu, Q. J. Multilayered echo state machine: a novel architecture and algorithm. *IEEE Transactions on cybernetics* 47, 946-959 (2017).
21. Gallicchio, C., Micheli, A., & Pedrelli, L. Deep reservoir computing: A critical experimental analysis. *Neurocomputing* 268, 87-99 (2017).
22. Ma, Q., Shen, L., & Cottrell, G.W. DeePr-ESN: A deep projection-encoding echo-state network. *Information Sciences* 511, 152-171 (2020).
23. Mitchell, M. An Introduction to Genetic Algorithms, Cambridge, MA, USA:MIT Press (1999).
24. Atiya, A. F., & Parlos, A. G. New results on recurrent network training: Unifying the algorithms and accelerating convergence. *IEEE Trans. Neural Netw.* 11, 697-709 (2000).
25. Weigend, A. S., & Gershenfeld, N. A. Time Series Prediction: Forecasting the Future and Understanding the Past, Routledge (1993).
26. Mackey, M. C., & Glass, L. Oscillation and chaos in physiological control systems. *Science* 197, 287-289 (1977).
27. Gallicchio, C., Micheli, A., & Pedrelli, L. Hierarchical temporal representation in linear reservoir computing. *Proceedings of the 27th Italian Workshop on Neural Networks (WIRN)*, 119-129 (2017).
28. Otte, S., Butz, M. V., Koryakin, D., Becker, F., Liwicki, M., & Zell, A. Optimizing recurrent reservoirs with neuro-evolution. *Neurocomputing* 192, 128-138 (2016).

29. Lazzús, J.A., Salfate, I., & Montecinos, S. Hybrid neural network-particle swarm algorithm to describe chaotic time series. *Neural Network World* 24, 601 (2014).
30. Gers, F.A., Eck, D., & Schmidhuber, J. Applying LSTM to time series predictable through time-window approaches. *Artificial Neural Networks-ICANN.* Heidelberg, Germany: Springer, 669-676 (2001).

Chapter 4
Neural Connectivity Inference with Spike Timing Dependent Plasticity Algorithm

4.1 Introduction

Understanding how the brain works is one of the most interesting, but extremely challenging scientific question. Recently, several international brain science programs, such as the BRAIN Initiative in the U.S. [1], the Human Brain Project in the E.U. [2], and the Brain/MINDS program in Japan [3] have been launched to try to map and understand the dynamics of neural activity in the brain. One of the major goals in these programs is inferring neural connectivity patterns from high-dimensional neural activity data recorded by multiple electrode arrays and fluorescence imaging. Knowing the connections between the neurons in the neural system is essential and fundamental information to understand how the signals are processed in the neural system and eventually learn the operating principle of the brain.

Genetic data processing consists of five stages [4]: data acquisition, pre-processing, network inference, post-processing, and validation. Neural activity data can be measured by several tools with different spatial scales, such as diffusion MRI at the macroscopic level, multiple electrode recordings at the mesoscopic level, and serial electron microscopy at the microscopic level. Since the macroscopic (microscopic) data are too coarse (dense) to capture the important information of the connections between local neurons, data measured at the mesoscopic level are often used. The overall processing architecture of the brain can be analyzed through understanding the connections between the anatomical areas at the macroscopic level, and the operational mechanisms of single neurons can be studied through examining the locations and

features of synapses at the microscopic level. At the pre-processing stage, spikes from each neuron can be identified by techniques such as principal component analysis and independent component analysis, and the noise can be reduced to produce more clear spike trains from the raw measured data for subsequent processing.

In general, connectivity inference methods can be classified into two classes: descriptive model-free methods [5-9], and generative model-based methods [10-12]. Descriptive model-free methods compute the cross-correlation or transfer entropy to estimate the relation between the neurons. For instance, the cross-correlation measures the strength of the delayed linear relationship between two neurons. From the time delay value of the highest cross-correlation, the direction of neural connection can be estimated. However, it cannot discriminate direct from indirect connections. Generative model-based methods assume a mathematical model describing the neural activity data and infer the parameters of the model. For example, assuming the neural activity is governed by the autoregressive model, the parameters of the model can then be determined by a standard method such as the maximum likelihood method. As the assumed mathematical model does not consider the physiological properties of the real neural networks, it may show significant gaps between the model and the biological reality. At the post-processing, the connectivity matrix obtained by the connectivity inference method can be further processed to achieve biologically realistic results. For instance, matrix deconvolution may be used to discriminate the direct connection and the indirect connections. Finally at the validation stage, the performance of the proposed model is evaluated using synthetic data and real neural data.

Although the inference methods based on the statistical measures have been widely used to infer neural connectivity, they have several limitations for analyzing biological neural systems with large number of neurons and large variations. First, the computational cost for the statistical

measures will grow dramatically as the length of neural spike trains and the neural system size increase. For example, the number of possible synaptic connections grows quadratically as the size of neural system increases. Second, when post processing techniques such as the generalized linear model are applied to the cross-correlogram for inferring both excitatory and inhibitory connections, it is necessary to have a significantly large number of spikes to reliably capture the peak and the trough in the cross-correlogram. Although the required minimum number of spikes depends on the strength of connections and the firing rate of neurons, typically it should be larger than a few thousand spikes for each neuronal spike train [9], which makes it difficult to reduce the computational cost. Third, detecting the specific profile, such as a peak or a trough, in the cross-correlogram is sensitive to the variations of the neural system. Since the model needs to assume several parameters such as a transmission delay to describe the neural system accurately, variations in the system and any mismatch between the assumed parameter and the true parameter will cause the inference accuracy to degrade significantly.

In this study, we propose a network based on second-order memristors for neural connectivity inference. The second-order memristor can natively implement biological features such as spike-timing-dependent plasticity (STDP), which modulates synaptic strength according to the relative timing between the pre- and postsynaptic spikes. Our simulation results using synthetic neural activity data show the STDP algorithm can reconstruct the neural connectivity pattern in the neural network more accurately and more efficiently than statistic-based inference methods. Real-time and local training by the STDP algorithm in turn allow the proposed system to reduce the computational cost/time and offer a strong tolerance to the variation of neural system.

4.2. Spike-timing-dependent Plasticity

Spike-timing-dependent plasticity (STDP) is a synaptic modification process observed in the neural system in animals' brains [13]. According to STDP, the strength of connections between the neurons (*i.e.* synaptic weight) is adjusted by the relative timing between the spikes of pre- and postsynaptic neurons. A presynaptic spike arriving a few milliseconds before the postsynaptic spike induces the synaptic weight to be strengthened (potentiation), while a presynaptic spike arriving after the postsynaptic spike induces the synaptic weight to be weakened (depression). The origin of STDP can be attributed to the sensitivity of postsynaptic receptors, which is affected by the calcium level, which itself can be elevated by the presynaptic spike. It is believed that STDP plays a key role in the functions of the neural system in the brain, such as signal processing and learning.

Spiking neural networks (SNNs) trained by STDP have already attracted much attention for tasks such as image recognition [14,15]. Since spike-based approaches are event-based rather than frame-based, the sparsity of input signals allows the SNNs to achieve high power-efficiency with high throughput when compared with conventional deep neural networks. Moreover, local learning rules such as STDP can further reduce the computation cost of training because they do not require error backpropagation which is common in conventional gradient descent learning rules.

4.3 Second-order Memristor

The memristor conductance is believed to be determined by the spatial profile of oxygen vacancy, where regions with a high concentration of oxygen vacancies produce high conductance channels. External stimulus such as voltage or current can modulate the internal Vo profile and thus the conductive channel's size during the programming stage. The conductive

channel size, which directly affects the device conductance, is considered as a first-order state variable of the memristor. The first-order state variable's evolution can be modeled by a set of coupled equations, describing the dynamic ionic migration process and the electronic transport process, respectively. All memristors by definition will have such first-order state variables. However, such first-order state variables, which typically attribute to the long-term memory effect, do not provide a mechanism to capture the dynamics at shorter time scales that are critical to implement rate-dependent behaviors such as pair-pulse facilitation and timing-dependent behaviors such as STDP.

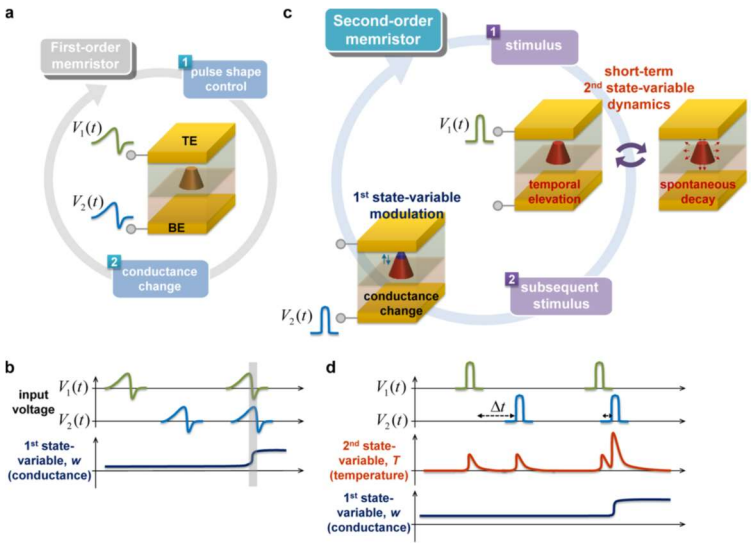

Figure 4-1 Schematics of first-order and second-order memristors.
(a) Operation of a first-order memrsitor. (b) The conductance change of the first-order memristor is directly determined by the external stimulus. The pulse shapes should be carefully engineered through overlapping the pre- and post-spikes to encode the temporal information. (c) Operation of a second-order memristor. (d) By including the second internal state variable, the second-order memristor can directly encode temporal data using simple and non-overlapping inputs.
Image adapted from Reference [17]. Image credit: Kim, S.

In reality, actual memristor devices' behaviors may be governed by multiple internal state variables that evolve at different time scales. As shown in Figure 4-1, the comprehensive physical model [16-18] recently proposed by our group was able to accurately describe memristors' dynamic behavior by including the effect of internal temperature, which considered as a second-order state variable since it is not directly reflected by an externally measurable

physical property like conductance. In the second-order device model, the Fourier equation for Joule heating and heat conduction is added to the first-order device model. The oxygen vacancy concentration, electrical potential, and internal temperature can then be obtained by self-consistently solving the three coupled equations (oxygen vacancy transport equation, electrical current continuity equation, and Fourier equation for thermal transport). When the effects of 2^{nd}-order state variables are non-negligible, richer dynamics can be observed in the devices. For example, after an external stimulus is applied to the memristor, the internal temperature is elevated locally due to Joule heating, and then the subsequent stimulus can induce a more considerable conductance change due to the enhanced diffusivity and drift velocity in the elevated internal temperature. Since the elevated internal temperature gradually decays to room temperature, the subsequent stimulus's enhancement strongly depends on the time interval between the stimuli. Thus, the second-order device model can natively implement rate- and timing-dependent behaviors such as STDP, as shown in Figure 4-2. In contrast, in a first-order memristor encoding the temporal information has to be performed by carefully controlling the overlapping of the pre- and postsynaptic pulses to achieve an engineered pulse shape on the device during the overlapped region. A second-order memristor, however, can efficiently process temporal data where information is natively encoded in the relative timing difference between the pre- and postsynaptic pulses without overlapping of the pulses.

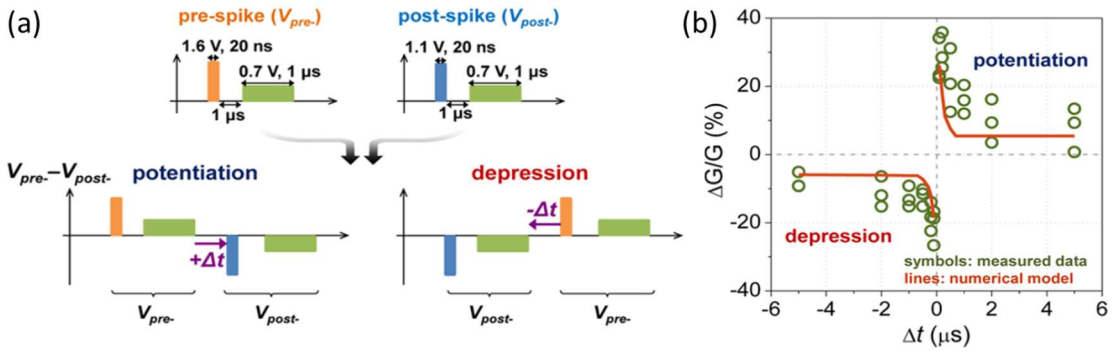

Figure 4-2 Implementation of STDP using the second-order memristor.
(a) Configuration of the pre- and post-spikes. The spikes consists of a (high and narrow) programming pulse followed by a (low and long) heating pulse. Due to the natural decay of temperature, the conductance change depends on both the order of pre- and post-spikes and the timing between the spikes. (d) STDP results showing the measured conductance changes (dots) and simulation results based on a second-order memristor model (solid lines).
Image adapted from Reference [17]. Image credit: Kim, S.

4.4 STDP-based Connectivity Inference

In a biological neural system, it is believed that the neural connectivity is developed and refined by STDP, which adjusts the strength of connections between the neurons based on the relative timing between the spikes of pre- and postsynaptic neurons. During the network evolution, STDP strengthens the connection between the neurons with causality to determine neural circuit functionality and improve signal communication efficiency. Based on the idea that STDP may be able to identify potential causal relationship between the neurons, in this work, we aim to construct the neural connectivity map from the neural spike trains by using the STDP learning rule.

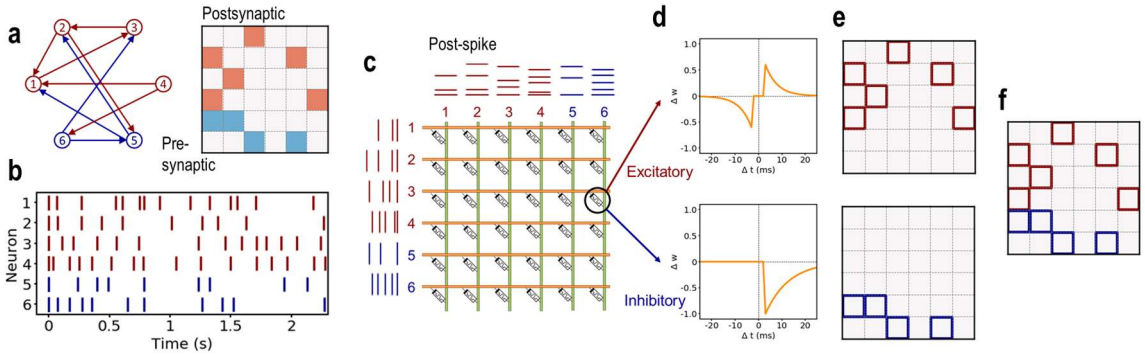

Figure 4-3 Schematic of STDP-based inference methods.
(a) Network graph and connectivity matrix. The type and direction of connections are depicted in the network graph as the color

(excitatory: red, inhibitory: blue) and direction of the arrow.In the connectivity matrix, the color of the pixel (excitatory: red, inhibitory: blue) shows the ground truth connection from the i^{th} neuron (y-axis) to the j^{th} neuron (x-axis). (b) Raster plot of the spiking neural network simulated by NEST simulator. (c) Overview of STDP-based inference methods implemented in the memristor array. The second-order memristor conductance follows the STDP-based inference method given the series of simulated spike trains. (d) STDP learning rules for excitatory (upper panel) and inhibitory connections (bottom panel). (e) Estimated connectivity matrices for excitatory (upper panel) and inhibitory connections (bottom panel). (f) Final estimated connectivity matrix. Red (blue) squares correspond to the excitatory (inhibitory) connections inferred by the STDP-based inference methods.

To evaluate how accurately the STDP-based inference method can reconstruct the connectivity pattern, we simulate the SNNs using the NEST simulator [19] and infer the neural connectivity matrix by applying the simulated neural spike trains on a memristor network consisting of 2^{nd}-order memristors. The inference accuracy is then compared with the statistic-based inference methods, as shown in Figure 4-3.

Two different versions of SNNs are tested to verify the ability of the STDP-based inference method. One is a simple ternary-weighted network of leaky integrated-and-fire (LIF) neurons with constant ternary synaptic weights: positive for excitatory connection, negative for inhibitory connection, and zero for no connection. Another is a bio-realistic analog-weighted network of Hodgkin-Huxley (HH) neurons, which has log-normal distributed excitatory synaptic weights [20] and normal distributed inhibitory synaptic weights [21]. Once the neural spike trains are obtained from the NEST simulator, they are applied to the second-order memristor array, where the device conduction evolutions are computed either by assuming an ideal device model following the ideal STDP learning rule or by using a physical device model governed by the coupled differential equations. As shown in Figure 4-3 (c), two different STDP learning rules were used for memristor conductance modulation: one for excitatory connection (eSTDP) and another for inhibitory connection (iSTDP). Following the original idea of STDP, the eSTDP potentiates (depresses) the device conductance when a post-spike comes after (before) a pre-spike so that the memristor device corresponding to an excitatory connection between a neuron pair will evolve to a higher conductance compared to others. On the other hand, an inhibitory

connection suppresses the activation of the postsynaptic neuron right after the pre-spike. This leads to the absence of post-neuron spikes, and makes it difficult to detect the inhibitory connection via conventional STDP due to the absence of pre-spike and post-spike pairs. To address this issue, we adopt a strategy based on the process of elimination. Specifically, the iSTDP rule was developed to depress the device conductance for cases corresponding to excitatory connection or no connection so that the device corresponding to the inhibitory connection will have a higher conductance than other devices in the end. The learning equations of eSTDP and iSTDP are following:

$$\Delta w_{eSTDP} = \begin{cases} \eta_{\mathrm{eSTDP}} \times (1 - \mathrm{w}_{\mathrm{eSTDP}}) \times \mathrm{A}_{\mathrm{p,eSTDP}} \times \exp\left(\frac{\Delta t}{\tau_{p,\mathrm{eSTDP}}}\right), & \Delta t \geq \tau \\ -\eta_{\mathrm{eSTDP}} \times \mathrm{w}_{\mathrm{eSTDP}} \times \mathrm{A}_{\mathrm{d,eSTDP}} \times \exp\left(\left|\frac{\Delta t}{\tau_{d,\mathrm{eSTDP}}}\right|\right), & \Delta t \leq -\tau \\ 0, & -\tau < \Delta t < \tau \end{cases} \quad (4\text{-}1)$$

$$\Delta w_{iSTDP} = \begin{cases} -\eta_{\mathrm{iSTDP}} \times \mathrm{w}_{\mathrm{iSTDP}} \times \mathrm{A}_{\mathrm{d,iSTDP}} \times \exp\left(\frac{\Delta t}{\tau_{d,\mathrm{iSTDP}}}\right), & \Delta t \geq \tau \\ 0, & \Delta t < \tau \end{cases} \quad (4\text{-}2)$$

where $\boldsymbol{w_{eSTDP}, w_{iSTDP}} \in [\mathbf{0}, \mathbf{1}]$ are respectively the normalized device conductance for inferring the excitatory and inhibitory connections, $\boldsymbol{\eta_{eSTDP}, A_{p(d),eSTDP}, \tau_{p(d),eSTDP}}$ are respectively the training rate, the amplitude of potentiation(depression), and the time constant of potentiation(depression) of eSTDP, $\boldsymbol{\eta_{iSTDP}, A_{d,iSTDP}, \tau_{d,iSTDP}}$ are respectively the training rate, the amplitude of depression, and the time constant of depression of iSTDP, $\boldsymbol{\tau}$ is the transmission delay of the neural network, $\boldsymbol{\Delta t}$ is the time interval between the pre-spike and the post-spike, $\boldsymbol{t_{post} - t_{pre}}$.

We note that the final device conductance is affected by the number of pre-spike and post-spike pairs. For instance, the neurons activated more frequently may have a high chance of having a more frequent device conductance updates than the neurons activated less frequently. Thus, to correctly detect the connectivity map across the network using a uniform threshold

value, the training rate should be modified according to each neuron's firing rate to compensate for the effects of different neural activity. In general the higher firing rate, the lower the training rate should be. We thus modify the training rate of synapse connecting the ith neuron and the jth neuron as following:

$$\eta_{ij} = \eta_0 \times \frac{\lambda_i \lambda_j}{\lambda_{avg}^2} \qquad (4\text{-}3)$$

where $\boldsymbol{\eta_0}$ is the initial training rate, and $\boldsymbol{\lambda_i, \lambda_{avg}}$ are the firing rate of the ith neuron and the average firing rate, respectively. We term the STDP-based inference method with (without) the modified training rate 'STDP+' ('STDP0'). After the device arrays trained by the STDP-based inference methods, the optimal threshold for deciding whether there is a specific type of connection or not is determined by maximize the Matthews correlation coefficient (MCC) [22].

$$MCC = \frac{N_{TP}N_{TN} - N_{FP}N_{FN}}{\sqrt{(N_{TP}+N_{FP})(N_{TP}+N_{FN})(N_{TN}+N_{FP})(N_{TN}+N_{FN})}} \qquad (4\text{-}4)$$

where $\boldsymbol{N_{TP}, N_{TN}, N_{FP}, N_{FN}}$ are the numbers of true positive, true negative, false positive, and false negative, respectively. A coefficient of +1 represents a perfect classification. The overall inference accuracy, a mean of MCC for excitatory (eMCC) and MCC for inhibitory (iMCC), can then be calculated from comparing the ground truth connectivity matrix and the inferred connectivity matrix.

4.4.1 Ternary-weighted Neural Network with LIF Neurons

To verify that the STDP-based methods can infer neural connectivity from recorded spike trains, including the type and direction of connections, we first test the inference performance of the STDP-based methods for a ternary-weighted neural network of LIF neurons. The network consists of 800 excitatory neurons and 200 inhibitory neurons. In the biological neural system, in general only a small portion of the whole neural system can be monitored due to the limited capability of measurement tools. Similarly, we sample 20 neurons (16 excitatory neurons and 4

inhibitory neurons) out of the whole neural population. Each neuron receives the synaptic signals from randomly chosen 100 excitatory neurons and 100 inhibitory neurons. The weights of the excitatory and inhibitory connection, and the transmission delay are assumed to be 1 mV, -2 mV, and 3 ms, respectively. The parameters for the ternary-weighted neural network are summarized in Table 4-1. The neural network is simulated for a period representing 30 min.

Neuron parameters		
τ_m	20 ms	Membrane time constant
τ_r	2 ms	Refractory period
C_m	1 pF	Membrane capacity
V_{reset}	0 mV	Reset potential
V_{th}	20 mV	Firing threshold
Synapse parameters		
w_{ex}	1 mV	Excitatory weight
w_{in}	-2 mV	Inhibitory weight
τ	3 ms	Transmission delay

Table 4-2 Parameters for ternary-weighted neural network.

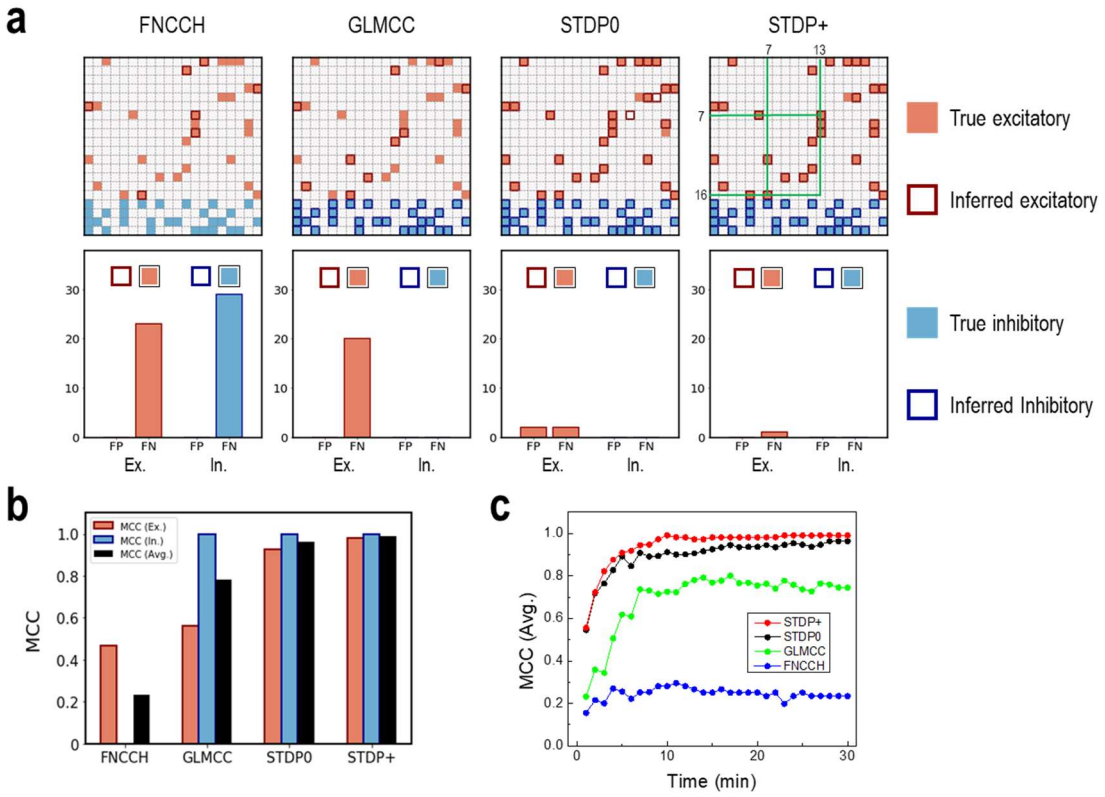

Figure 4-4 Ternary-weighted neural network with LIF neurons.
(a) Connectivity matrices (upper panel) with the ground truth (pixel) and the inferred connections (excitatory: red, inhibitory: blue) and the numbers of false positive and false negative cases for the excitatory and inhibitory connections (bottom panel). (b) Inference accuracy in terms of MCC. Averaged MCC is a mean of MCC for excitatory and MCC for inhibitory. (c) Averaged MCCs with respect to the simulation time length.

We compare the inference accuracy of the STDP-based methods (STDP0, STDP+) with that of the statistic-based methods such as the filtered normalized cross-correlation histogram (FNCCH) [8] and the generalized linear model with cross-correlation (GLMCC) [9]. Figure 4-4 (a) shows the estimated connectivity matrices and the numbers of false positive and false negative cases for the excitatory and inhibitory connections. The inference accuracies in terms of MCC are summarized in Figure 4-4 (b). FNCCH is based on the normalized cross-correlogram followed by subtracting the mean value of the cross-correlogram. The FNCCH method can distinguish between peaks and troughs by considering the signs: a positive value refers to an

excitatory connection, and a negative value refers to an inhibitory connection. The GLMCC method applies the generalized linear model to the normalized cross-correlogram, where the generalized linear model helps distinguish short-range synaptic impacts (*i.e.* direct connections) from slow, large-scale wavy fluctuations (*i.e*. noise and indirect connection) by fitting a coupling filter. The STDP-based inference methods outperform the statistic-based methods in inferring both excitatory and inhibitory connections.

Due to the transmission delay, the STDP-based inference methods can also accurately distinguish the direct connection from the indirect connection. For instance, there are direct excitatory connections from the 16^{th} neuron to the 7^{th} neuron and from the 7^{th} neuron to the 13^{th} neuron. If the inference method is not able to discriminate the direct connection from the indirect connection, it will infer there is a connection from the 16^{th} neuron to the 13^{th} neuron, which will lead to a false positive in the case of detecting only direct connections. As shown in Figure 4-4 (a), although there is interdependency between the 16^{th} neuron and the 13^{th} neuron due to the indirect connection, the STDP-based inference methods correctly did not mark it as a (direct) connection.

Additionally, the STDP-based inference methods show better tolerance to the randomness of the time interval between the spikes since real-time conductance updates based on STDP, which is a local learning rule and whose effects only become measurable after multiple updates, can largely average out the effect of randomness or fluctuation. On the other hand, FNCCH has a vulnerability to fluctuations in the cross-correlograms since it detects the maximum amplitude of the peak or the trough. If FNCCH detects any peak or trough not caused by the actual connection, it will induce a large number of false estimations as shown in Figure 4-4 (a). Compared to FNCCH, GLMCC has better tolerance to fluctuations by using the generalized

linear model to capture the specific profile such as peaks or troughs. Thus, it shows better performance than FNCCH, but still produces worse accuracy than the STDP-based methods in inferring excitatory connections. In particular, GLMCC requires a large number of spikes before it can reliably capture the specific profiles in the cross-correlograms. Figure 4-4 (c) shows the averaged MCCs of the STDP-based methods and the statistic-based methods with respect to the length of simulation time. The more neural spikes, the smoother profile in the cross-correlograms. Thus, the GLMCC inference accuracy improves as larger number of spikes are applied. On the other hand, the STDP-based methods can produce higher inference accuracy and estimate the correct connectivity matrix using only a smaller number of spikes since it's based on online updates instead of assuming a specific statistical model. Notably, STDP+, which is based on the modulated training rate, shows better performance than STDP0 even for small variations of firing rates, *e.g.* 4.62 ± 0.28 Hz.

4.4.2 Effect of Transmission Delay Mismatch

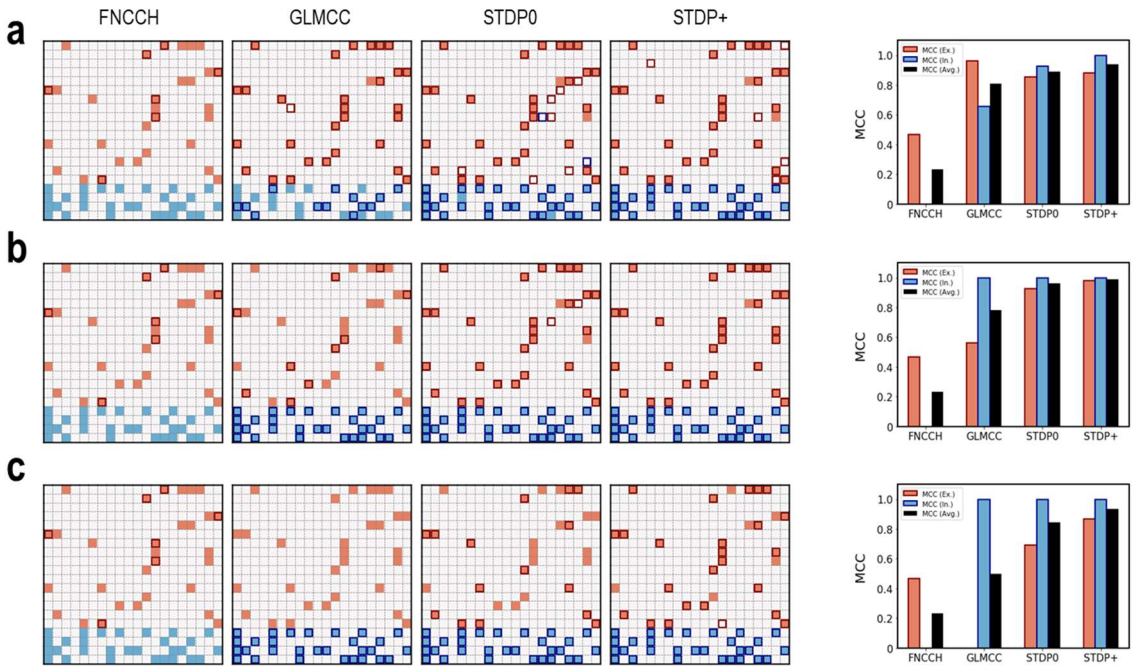

Figure 4-5 Effect of transmission delay mismatch.
(a) Connectivity matrices and inference accuracy in terms of MCC with the underestimated value of transmission delay, 2 ms. (b) Connectivity matrices and inference accuracy in terms of MCC with the correct value of transmission delay, 3 ms. (c) Connectivity matrices and inference accuracy in terms of MCC with the overestimated value of transmission delay, 4 ms.

In addition to distinguishing the types of connections, the direction of the connection offers important information to understand how the signal flows in the brain. The transmission delay, which represents the time delay for the neural signal to propagate from one neuron to another, is an important factor in deciding the direction of the connection. While in this work we know the transmission delay in the simulated neural spike data, knowing the exact values of transmission delays for all neurons in the biological neural network is impossible. Since the STDP-based inference methods and the GLMCC methods need to assume the transmission delay value, there will be a high chance of having a mismatch between the true value and estimated value of the transmission delay and the effects of variations in assumed transmission delays thus need to be carefully analyzed.

Figure 4-5 shows the inferred connectivity matrices with different estimated values of transmission delays. When the estimated value of the transmission delay, 2 ms, is smaller than the true value, 3 ms, the STDP-based methods show slightly degraded inference accuracies but still offers better performance than the statistic-based methods. It is interesting to note that with the underestimated transmission delay the GLMCC method produces almost the same averaged inference accuracy as the GLMCC method with the correct transmission delay. In general, with the underestimated transmission delay, the inference accuracy for excitatory connection is improved, and the inference accuracy for inhibitory connection is degraded. Several individual correlation values can coincidentally be produced within 2 ms and 3 ms lags in the cross-correlograms. These individual values are caused by a noise or random fluctuations rather than by a direct excitatory connection, which results in a wide peak profile. The coincident correlation in the extra margin due to the underestimated transmission delay may help the GLMCC method to capture the peak profile more reliably, but at the same time disturb the GLMCC method to detect the trough profile. When the estimated value of transmission delay, 4 ms, is larger than the true value, 3ms, the degradation of GLMCC is much significant compared to the degradation of STDP-based methods, as shown in Figure 4-5 (c). With an over-estimated value of transmission delay, the most valuable peaks right after the transmission delay in the cross-correlograms is lost so that GLMCC cannot infer any excitatory connection. The inference accuracy for inhibitory is not degraded much, since the inhibitory connections are stronger than the excitatory connections and produce a wider trough profile wider than the peak profile in the cross-correlograms. Thus, although some parts of the trough profile are excluded due to the overestimated transmission delay, GLMCC can still detect the trough profile in the cross-correlogram. Overall, these results

show STDP-based inference methods show stronger tolerance in the mismatch of transmission delay than the statistic-based inference methods.

4.4.3 Analog-weighted Neural Network with HH Neurons

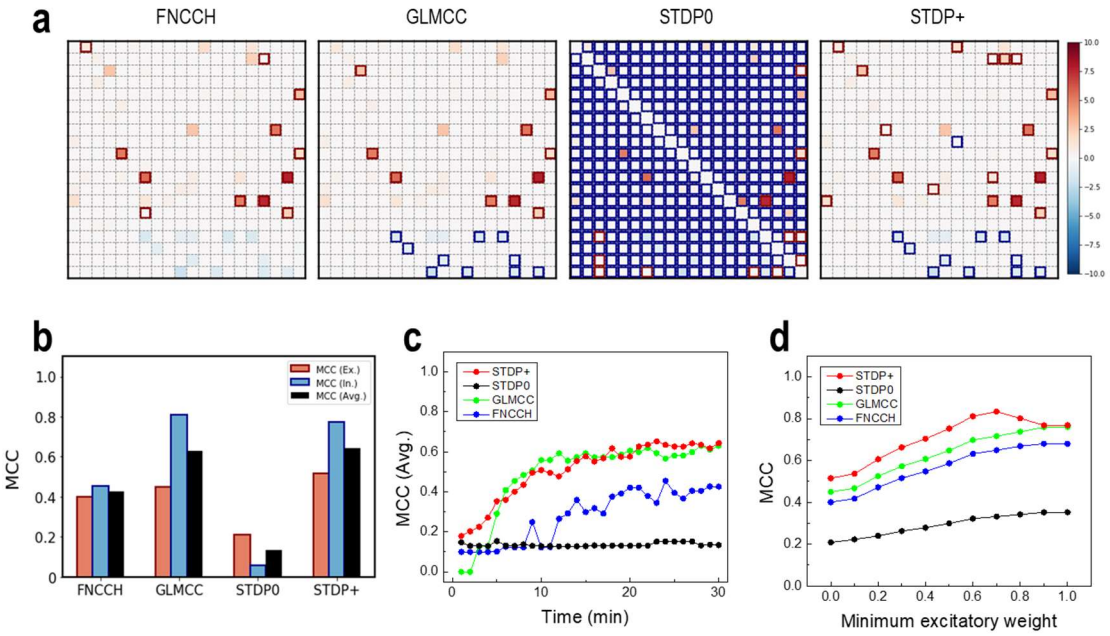

Figure 4-6 Analog-weighted neural network with HH neurons.
(a) Connectivity matrices with the ground truth (pixel) and the inferred connections (excitatory: red square, inhibitory: blue square). (b) Inference accuracy in terms of MCC. Averaged MCC is a mean of MCC for excitatory and MCC for inhibitory. (c) Averaged MCCs with respect to the simulation time length. (d) MCC for excitatory with respect to the different minimum threshold values for excitatory weights.

Typically, biological neural systems have analog synaptic weights rather than ternary synaptic weights. To verify the ability of STDP-based inference methods to reconstruct the connectivity pattern in biological neural systems, we test the system's performance for analog-weighted neural networks with HH neurons. For a fair comparison with the statistics-based inference methods, the neural spike data in Ref. 9 are used to test and evaluate STDP-based inference methods' performance. The network consists of 800 excitatory neurons and 200 inhibitory neurons. The excitatory neurons are connected to 12.5 % of other neurons with log-normal distributed synaptic weights. The inhibitory neurons are connected to 25 % of other

neurons with normal distributed synaptic weights. The details of the analog-weighted neural network with HH neurons can be found in Ref. 9. Figure 4-6 (a) and (b) show the estimated connectivity matrices and the inference accuracy in MCC terms, respectively. The STDP+ method and GLMCC can infer most excitatory and inhibitory connections with relatively strong synaptic weights. The STDP0 method, however, estimates every connection to be inhibitory, which means it fails to distinguish the inhibitory connections from others. In contrast to the small performance gap between the STDP0 and STDP+ methods in the ternary-weighted neural network with LIF neurons, the STDP+ method shows much higher inference accuracy than the STDP0. This behavior can be attributed to a larger variation of the analog-weighted neural network's firing rate, 2.21 ± 1.03 Hz, compared to the ternary-weighted neural network, 4.62 ± 0.28 Hz. Without compensating for the different firing rates of neurons in the network, the final conductance obtained from STDP0 is primarily affected by the number of conductance updates, which is in turn determined by the post-synaptic neuron's firing rate, rather than the strength of excitatory or inhibitory connections. This effect is more severe when the iSTDP rule is applied to the memristor array since iSTDP with only depression updates cannot produce balanced conductance updates as eSTDP which offers both potentiation and depression updates.

Figure 4-6 (c) shows the averaged MCCs for different inference methods with respect to the simulation time length. The STDP+ and the GLMCC clearly outperform other methods in the analog weight network case. While the GLMCC is specialized in estimating the analog synaptic weights, the STDP+ predicts the probability of connection from the device conductance updated by the STDP learning rule to produce an estimated binary connectivity matrix with a low computational cost. Since the excitatory synaptic weights follow the log-normal distribution, the large amounts of excitatory connections with weak synaptic strength in fact have minimal effects

on the neurons' activation compared to the small number of strong excitatory connections. Thus, it is important for the inference methods to correctly infer the strong excitatory connections, not just producing an acceptable average MCC value for all connections. Figure 4-6 (d) shows the MCC for excitatory connections with different minimum threshold values used to classify the connections. Excitatory connections weaker than the minimum threshold value are ignored when the true positive cases for excitatory connections are calculated. By focusing on the inference of only strong excitatory connections, the performances of all methods are improved, and the STDP+ method outperforms all other methods highlighting the capability of this approach to pick up causality among the recorded signals. The improvement of inference accuracy for inhibitory connection is not significant since it follows a normal distribution which is not as long-tailed as the log-normal distribution.

4.4.4 Connectivity Inference with Physical Device Models

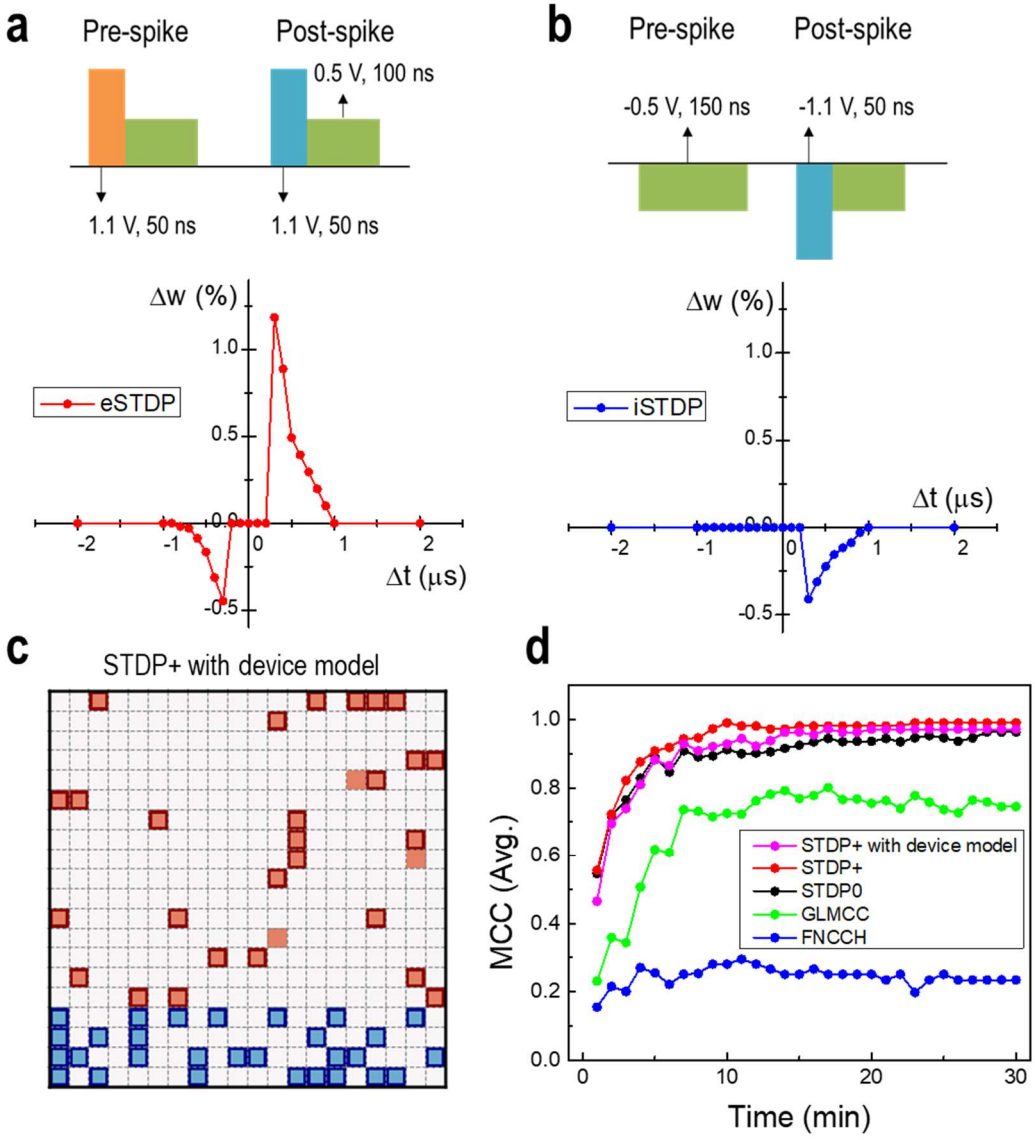

Figure 4-7 Second-order memristor array for connectivity inference.
(a) Voltage pulse configuration for eSTDP and eSTDP obtained from the circuit simulation. (b) Voltage pulse configuration for iSTDP and iSTDP obtained from the circuit simulation. (c) Connectivity matrix with the ground truth (pixel) and the inferred connections of STDP-based inference method with device model (excitatory: red, inhibitory: blue) (d) Averaged MCCs with respect to the simulation time length.

The second-order memristor model [16-18] with coupled differential equations can comprehensively explain the switching mechanism of memristor devices and accurately predict memristor's dynamic behavior. By utilizing the temporal dynamics of the 2nd-state variable, *i.e.* the internal device temperature, the STDP behavior can be reproduced without any overlapped

voltage pulse. Thus, the STDP-based inference methods can be natively implemented in an array composed of 2nd-order memristors. Figure 4-7 (a) and (b) show the voltage pulse configuration for eSTDP and iSTDP, and the simulated eSTDP and iSTDP learning rules using the physical device model, where the pre-spike and post-spike are applied to the top electrode and the bottom electrode of the memristor, respectively.

In eSTDP, when the post-spike comes after the pre-spike, the memristor conductance is potentiated as the total pulse sequence, including the positive heat pulse of the pre-spike followed by the positive programming voltage of the post-spike at the elevated temperature, induces enhanced SET programming. In iSTDP, to produce only depression update, the pre-spike consists of only the heat pulse, which induces enhanced RESET programming when the negative programming pulse of post-spike arrives after the pre-spike. Since the memristor's internal dynamics occur at a much faster time scales, a few μs, the time scale of neural spike trains can be converted to the time scale of the memristor device by reducing the scale by a factor of 10,000. Figure 4-7 (c) and (d) show the ternary-weighted neural network's connectivity matrix inferenced by 2^{nd}-order memristor device array when using the physical device model, and the averaged MCCs with respect to the simulation time length. Although the STDP learning curves obtained from the circuit simulation using the device model are not exactly the same as the ideal STDP learning curves shown in Figure 4-3 (d), comparable inference accuracy results can still be obtained.

To allow the memristor network to directly process the neural spikes, the characteristic time of the memristor's internal dynamics need to be slowed down to ~ms from ~μs. This can possibly be achieved through device structure optimizations to control the internal temperature dynamics by managing the heat dissipation. For example, by inserting a heat insulating layer or a

heat reservoir to prevent fast heat dissipation, or changing the switching layer or the base layer's material to tailor the specific heat or thermal conductivity. Memristor devices based on the migration of other ionic species will also be considered, as recent developments on perovskite halide-based memristors [23] have showed short-time dynamics on the order of a few tens of ms time scale, which is comparable with that of the biological neurons.

4.5 Conclusion

In this work, the neural connectivity patterns, including the type and direction of the synaptic connections, are inferred by STDP-based methods. For ternary-weighted neural networks with LIF neurons, the STDP-based inference methods not only show better accuracy but also strong tolerance to fluctuations in transmission delays when compared with statistics-based methods. When the synaptic weights are analog, such as log-normal or normal distributed, the proposed STDP-based inference methods have comparable accuracy to the statistics-based methods at reduced computational cost. Since the STDP rules can be natively implemented in second-order memristors, the proposed approach can potentially be implemented cheaply using arrays of second-order memristors with little pre- and postprocessing.

The variation-tolerance and simplified computation can be attributed to the online and local training property of the STDP learning rule. Rather than estimating the causality in the global profile of the cross-correlogram, the STDP-based inference methods update the possibility of connection based on the time sequence of the local spike pairs. When natively implemented in a second-order memristor, the computational cost/time can be further reduced compared with software-based STDP implementations. Successful implementations of the STDP algorithm on neural connectivity inference may encourage further theoretical study on the use of local learning

rules such as STDP in other tasks, and stimulate the developments of new applications based on bio-inspired features.

References

1. Insel, T. R., Landis, S. C., & Collins, F. S. The NIH brain initiative. *Science* 340, 687-688 (2013).
2. Amunts, K., Ebell, C., Muller, J., Telefont, M., Knoll, A., & Lippert, T. The human brain project: creating a European research infrastructure to decode the human brain. *Neuron* 92, 574-581 (2016).
3. Okano, H., Sasaki, E., Yamamori, T., Iriki, A., Shimogori, T., Yamaguchi, Y., Kasai, K. & Miyawaki, A. Brain/MINDS: a Japanese national brain project for marmoset neuroscience. *Neuron* 92, 582-590 (2016).
4. de Abril, I. M., Yoshimoto, J., & Doya, K. Connectivity inference from neural recording data: Challenges, mathematical bases and research directions. *Neural Networks* 102, 120-137 (2018).
5. Garofalo, M., Nieus, T., Massobrio, P., & Martinoia, S. Evaluation of the performance of information theory-based methods and cross-correlation to estimate the functional connectivity in cortical networks. *PloS one* 4, e6482 (2009).
6. Ito, S., Hansen, M. E., Heiland, R., Lumsdaine, A., Litke, A. M., & Beggs, J. M. Extending transfer entropy improves identification of effective connectivity in a spiking cortical network model. *PloS one* 6, e27431 (2011).
7. Shimono, M., & Beggs, J. M. Functional clusters, hubs, and communities in the cortical microconnectome. *Cerebral Cortex* 25, 3743-3757 (2015).
8. Pastore, V. P., Massobrio, P., Godjoski, A., & Martinoia, S. Identification of excitatory-inhibitory links and network topology in large-scale neuronal assemblies from multi-electrode recordings. *PLoS computational biology*, *14*(8), e1006381 (2018).
9. Kobayashi, R., Kurita, S., Kurth, A., Kitano, K., Mizuseki, K., Diesmann, M., ... & Shinomoto, S. Reconstructing neuronal circuitry from parallel spike trains. *Nature communications*, *10*(1), 1-13 (2019).
10. Harrison, L., Penny, W. D., & Friston, K. Multivariate autoregressive modeling of fMRI time series. *Neuroimage* 19, 1477-1491 (2003).
11. Kim, S., Putrino, D., Ghosh, S., & Brown, E. N. A Granger causality measure for

point process models of ensemble neural spiking activity. *PLoS Comput Biol* 7, e1001110 (2011).

12. Song, D., Wang, H., Tu, C. Y., Marmarelis, V. Z., Hampson, R. E., Deadwyler, S. A., & Berger, T. W. Identification of sparse neural functional connectivity using penalized likelihood estimation and basis functions. *Journal of computational neuroscience* 35, 335-357 (2013).
13. Bi, G. Q., & Poo, M. M. Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience* 18, 10464-10472 (1998).
14. Diehl, P. U., & Cook, M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers in computational neuroscience* 9, 99 (2015).
15. Kheradpisheh, S. R., Ganjtabesh, M., Thorpe, S. J., & Masquelier, T. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Networks* 99, 56-67 (2018).
16. Kim, S., Choi, S., & Lu, W. Comprehensive physical model of dynamic resistive switching in an oxide memristor. *ACS nano* 8, 2369-2376 (2014).
17. Kim, S., Du, C., Sheridan, P., Ma, W., Choi, S., & Lu, W. D. Experimental demonstration of a second-order memristor and its ability to biorealistically implement synaptic plasticity. *Nano letters* 15, 2203-2211 (2015).
18. Lee, S. H., Moon, J., Jeong, Y., Lee, J., Li, X., Wu, H., & Lu, W. D. Quantitative, Dynamic TaO x Memristor/Resistive Random Access Memory Model. *ACS Applied Electronic Materials* 2, 701-709 (2020).
19. Gewaltig, M. O., & Diesmann, M. Nest (neural simulation tool). *Scholarpedia*, *2*(4), 1430 (2007).
20. Song, S., Sjöström, P. J., Reigl, M., Nelson, S., & Chklovskii, D. B. Highly nonrandom features of synaptic connectivity in local cortical circuits. *PLoS Biol*, *3*(3), e68 (2005).
21. Hoffmann, J. H., Meyer, H. S., Schmitt, A. C., Straehle, J., Weitbrecht, T., Sakmann, B., & Helmstaedter, M. Synaptic conductance estimates of the connection between local inhibitor interneurons and pyramidal neurons in layer 2/3 of a cortical

column. *Cerebral Cortex*, *25*(11), 4415-4429 (2015).

22. Matthews, B. W. Comparison of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, *405*(2), 442-451 (1975).

23. Zhu, X., Wang, Q., & Lu, W. D. Memristor networks for real-time neural activity analysis. *Nature communications*, 11(1), 1-9 (2020).

Chapter 5 Future Works

5.1 Connectivity Inference with Physically Recorded Neural Signals

In the Chapter 4, we showed that STDP-based inference methods can correctly infer the neural connectivity from synthetic neural spike data. Several assumptions were made to simplify the simulation configuration, such as assuming identical neuronal characteristics and fixed transmission delay. However, several studies [1-3] have shown that the biological neural system has diverse types of neurons, non-uniform transmission delays, and connectivity patterns that evolve with time. These biophysical properties need to be included and carefully analyzed so that the STDP-based inference methods can be used to analyze physically recorded biological neural systems.

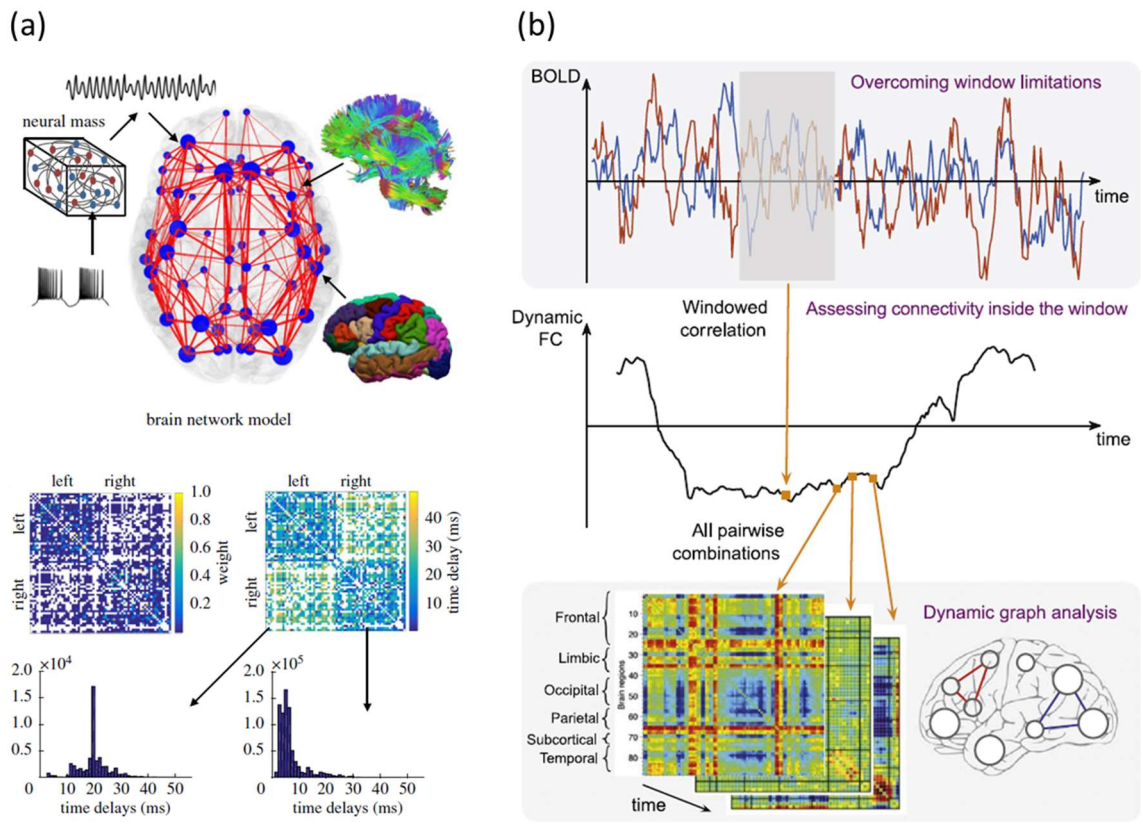

Figure 5-1 Biophysical properties of neural systems.
(a) Schematic of neural connectivity inference model (top), and matrices of strengths, time delays, and distribution of time delays (bottom). (b) Schematics of the commonly used approach to infer the dynamic neural connectivity.
Image adapted from Reference [2,3].

First, biological neural systems are not composed of identical neurons. It is believed that the diversity of neurons is one of the reasons why the biological neural systems can perform complex tasks and have a good tolerance in the variation. Figure 5-1 (a) shows the schematics of the neural connectivity inference model (top) and the matrices of strengths, time delays, and the distribution of time delays (bottom). Due to the neurons' various types and characteristics, the transmission delay may be more accurately described by the normal or log-normal distribution rather than a constant value. Although the STDP-based inference methods show a strong tolerance to the transmission delay mismatch, a constant value of the transmission delay was still assumed in the SNNs' simulations. It is necessary to understand how the distributed transmission delays affect the inference methods' performance. In addition, we need to carefully analyze the

effects of large variations of the neural characteristics on the performance of the STDP-based inference methods.

Additionally, the neural connectivity pattern in biological neural systems can and will often change during the development. Detecting the transitions of the connections in real-time is very useful to analyze the dynamics of the neural system. Figure 5-1 (b) shows the schematic of the commonly used approach to infer the evolution of neural connectivity. When model-free methods such as correlation are used to extract the neural connectivity, they assume the connectivity is fixed. If the connectivity changes during the measurement, the inferred connectivity from the correlation technique will no longer be accurate since the correlation technique does not compensate for the effects of connectivity changes. As shown in Figure 5-1 (b), to consider the evolution of neural connectivity, the sliding window technique is normally utilized to keep monitoring the neural connectivity. This technique however requires redundant calculations since the neural spikes within the overlapped region should be used several times to obtain the time-series correlation data. Moreover, a sufficiently large number of spikes in the given window are needed to produce a reliable inference accuracy. On the other hand, the STDP-based inference methods update the inferred synaptic weights in a real-time manner and will be well suited for detecting the evolutions of connectivity patterns. As discussed in the Chapter 4, the STDP-based inference methods require much fewer number of spikes to infer the neural connectivity than statistic-based methods, and it thus will be interesting to verify the performance STDP-based inference methods to reconstruct neural connectivity evolutions.

5.2 Memristor Array Implementation

As discussed in the Chapter 4, the second-order memristor can natively implement the STDP learning rule and arrays of such devices can potentially be used to directly performance

connectivity inference tasks. However, several factors still need to be carefully considered. In the ideal STDP case, the conductance change depends only on the relative timing between the pre- and post-spikes, regardless of the current device conductance. However, in an actual memristor device the conductance change also depends on the current device conductance. For example, it is more difficult to increase the memristor's conductance when it is already at a high conductance state, compared to when it is at a low conductance state. Thus, the conductance change is inversely dependent on the current device conductance. The effects of the non-uniform conductance change should be carefully analyzed on the neural connectivity inference task.

Additionally, nonideal effects coming from the array configuration also need to be considered. For example, when the memristor array is fabricated in the crossbar structure, the line resistance will cause the voltage drop along the metal lines and reduce the actual voltage applied to the memristor devices. The effects of line resistance in turn depend on the location of the device in the array. As the reduced voltage can weaken the effect of potentiation/depression pulse, heat pulse, or both, the memristor device located far from the voltage sources show different behaviors from devices located close to the sources. To minimize this effect, the line resistance should be minimized, or the internal memristor resistance should be increased. Before performing actual measurements, the location-dependent pulse conditions should be included in the simulation to test how it affects the neural connectivity inference task's accuracy.

References

1. Bock, D. D. *et al.* Network anatomy and in vivo physiology of visual cortical neurons. *Nature* 471, 177-182 (2011).
2. Petkoski, S., & Jirsa, V. K. Transmission time delays organize the brain network synchronization. *Philosophical Transactions of the Royal Society A*, 377, 20180132 (2019).
3. Preti, M. G., Bolton, T. A., & Van De Ville, D. The dynamic functional connectome: State-of-the-art and perspectives. *Neuroimage*, 160, 41-54 (2017).