THE UNIVERSITY OF SYDNEY

MASTERS THESIS

---

# Reservoir Computing with Neuro-memristive Nanowire Networks

---


*Author:*
Kaiwei FU

*Supervisor:*
Prof. Zdenka KUNCIC


*A thesis submitted in fulfillment of the requirements for the degree of Master of Philosophy*

August 27, 2021

# Declaration of Authorship

I, Kaiwei FU, declare that this thesis titled, "Reservoir Computing with Neuromemristive Nanowire Networks" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated. Part of Chapter three of this thesis is published as Fu et al. [1].
- Where I have consulted the published work of others, this is always clearly attributed.
- This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

Signed:
_____

Date: 2021-08-27
_____

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Signed:

_____

THE UNIVERSITY OF SYDNEY

# *Abstract*

Master of Philosophy

**Reservoir Computing with Neuro-memristive Nanowire Networks**

by Kaiwei FU

We present simulation results based on a model of self–assembled nanowire networks with memristive junctions and neural network–like topology. We analyse the dynamical voltage distribution in response to an applied bias and explain the network conductance fluctuations observed in previous experimental studies. We show $I - V$ curves under AC stimulation and compare these to other bulk memristors. We then study the capacity of these nanowire networks for neuro-inspired reservoir computing by demonstrating higher harmonic generation and short/long–term memory. Benchmark tasks in a reservoir computing framework are implemented. The tasks include non-linear wave transformation, wave auto-generation, and hand-written digit classification.

# *Acknowledgements*

Thanks to my supervisor Professor Zdenka Kuncic for research and academic guidance. Thanks to my colleagues, Ruomin Zhu, Joel Hochstetter and Alon Loeffler, for their help and support. Thanks to the School of Physics and my family for study abroad support. Thanks to the University of Sydney.

# Contents

# List of Figures

# Chapter 1

# Introduction

This thesis presents a study on neuromorphic nanowire networks. These represent a unique type of neuromorphic system created by bottom-up self-assembly of nanowires into a complex, disordered network with a neural network–like topology and cross-point junctions with resistive switching memory (memristive). New simulation results are presented based on a physically-motivated model of nanowire networks. The results reveal diverse neural-like dynamical properties and show how these properties can be used for neuro-inspired information processing in a reservoir computing framework

Chapter two presents a survey of the current literature on neuro-inspired computing relevant to this thesis. We highlight the most relevant aspects of artificial neural networks before reviewing the most salient literature on neuromorphic computing and reservoir computing.

Chapter three describes the methods (Sec. 3.2) developed and applied in this work and presents the simulation results (Sec. 3.3). The results can be broadly subdivided into two categories: dynamics at the individual switch junction level and at the network level; and implementation of reservoir computing tasks.

Our analysis of switch junction and network dynamics (Sec. 3.3.1) reveals: (i) The dynamical state changes of individual memristive switch junctions. Importantly, this type of study is not possible in hardware experiments and thus shows the value of simulations in this case; (ii) The interplay between the junctions and complex network circuit and its impact on the entire network. Simulations show how voltage is dynamically redistributed as switches turn on and current paths form.

Key results from the application to reservoir computing (Sec. 3.3.2) are: (i) An accuracy of $95\% - 98\%$ can be readily achieved for the nonlinear waveform transformation task using piece-wise linear regression. These results are better than that obtained in previous studies. (ii) Sine wave auto-generation is demonstrated with 100% accuracy. (iii) Handwritten digit classification to an accuracy of 92% achieved with linear discriminant analysis. This result is better than that obtained in previous studies using multiple bulk memristors.

The main conclusions of the thesis and an outlook for future work are presented in Chapter four.

# Chapter 2

# Literature Survey – Neuro-inspired computing

The human brain is the most complex organ of the human body, composed of roughly 86 billion neural cells (neurons) and just as many non-neuronal cells (e.g. glial cells). Each neuron typically has thousands of electrically stimulated synapses connecting it to other neurons [2]. Neurons and their synaptic connections form a vastly complex network in which the total length of neural connections can reach thousands of kilometers [3]. Information is continuously and effortlessly exchanged between neurons via their dynamic synapses (cf. Figure 2.1). No physical connection exists between the neurons, with a gap of about 20 nanometers, rather information is transmitted primarily via neurotransmitter molecules [4, 3].

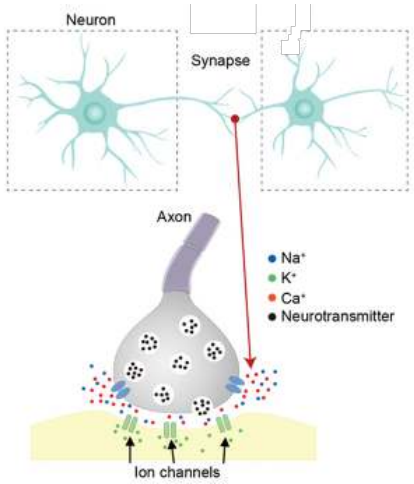



FIGURE 2.1: Schematic illustration of neurons and synapse. Electrical signals propagate down the axon of a stimulated neuron, causing release of neurotransmitter molecules that transmit the encoded information to a receptor neuron via its ion channels (adapted from [5]).

In the simplest representation, a neuron can be thought of as a cell with several states including two main states: excitation and inhibition. The state of a neuron depends on the amount of input signal received from other neurons and the strength (inhibition or excitation) of the synapse. Above a certain voltage threshold, a neuron is excited and produces electrical impulses [6]. Electrical impulses are transmitted along the axons and across synapses to other neurons.

Neuro-inspired information processing aims to broadly mimick the process of electrical signal transduction. The goal is to use biological mechanisms operating within the brain's neocortex, where higher-order brain functions such as cognition and sensory perception originate, as a blueprint to construct novel computer architectures, in both software and hardware (cf. Figure 2.2) [7, 8, 5, 9]. In software, the most well-known type of neuro-inspired computing architectures are Artificial Neural Networks (ANNs). In hardware, brain-like architectures underlie the rapidly growing research field known as *neuromorphic computing*, the subject of this thesis.

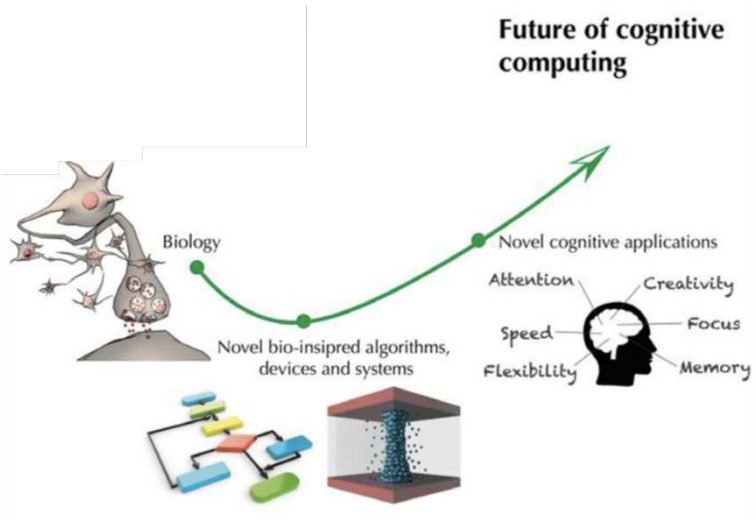



FIGURE 2.2: Cartoon illustrating the premise of neuro-inspired computing and potential for future cognitive computing (adapted from [10]).

## 2.1 Artificial neural networks

Artificial Neural Networks (ANNs) are computational algorithms that are loosely based on the biological neural network in the brain [11]. ANNs are an active area of research that has emerged in the field of artificial intelligence

since the 1980s [12, 9]. The algorithms abstract the brain's neural network from the perspective of information processing, typically using a bipartite network structure. ANNs use nodes (artificial neurons) interconnected by edges (artificial synapses). Each node is bound by a specific output function, called an activation function. Each connection between two nodes has a corresponding weighted value for the signal passing through, representing the strength of the connection [13].

From a system perspective, although ANNs are inspired by the neural network structure of the brain, their computation method differs from the brain's method of information processing [14]. In ANNs, learning is generally implemented using a mathematical update rule known as backpropagation and with gradient descent optimisation to train the network weights [15]. The brain's neural network, on the other hand, does not learn this way and does not have static, quantifiable weights, but rather highly dynamic and adaptive synapses [5]. Fig.2.3 shows the basic structure of a Feed-forward Neural Network (FNN), which is the simplest type of ANN. In FNNs, information flows from neurons in an input layer to output neuron(s) via a hidden layer of neurons. Every single neuron processes the information by a preset function and adjustable weights [16].

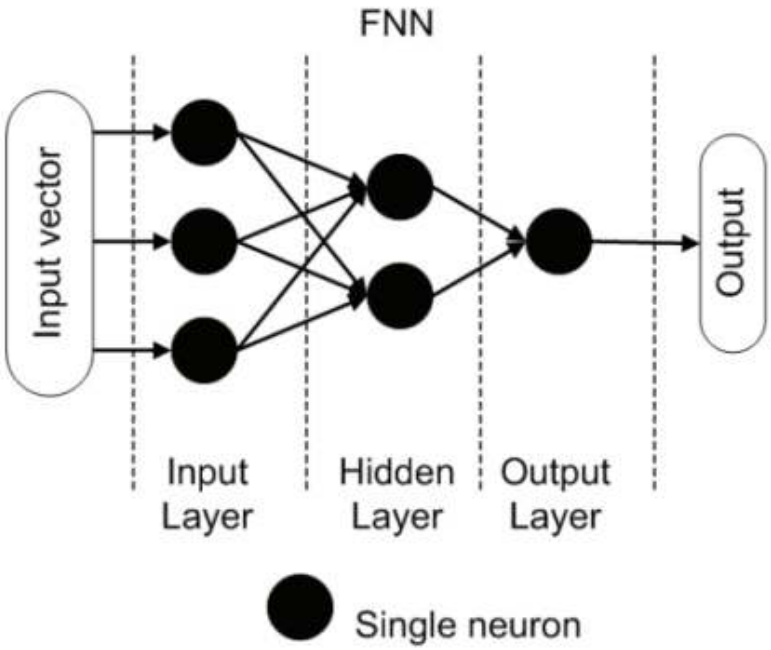



FIGURE 2.3: The structure of a general Feed-forward Neural Network (FNN), a classic, broadly-used ANN in which information flows from the input layer to the output layer via a hidden layer. [16]

Spiking Neural Networks (SNNs) are an important type of neural network as they emulate the spiking nature of biological neurons. With the development of the EU Human Brain Project and the US BRAIN Initiative, spiking neural networks have increasingly become a major focus of research [17, 5]. As a phenomenological model, the neurons in SNNs are not linked by weight or activation function as in traditional ANNs. Instead, each neuron in an SNN owns a potential threshold which may be measured by voltage (in a physical device) or weight (in an algorithm) [17]. If the voltage exceeds a threshold, a spike is generated and passed to all connnected neurons. Fig.2.4 depicts this process.

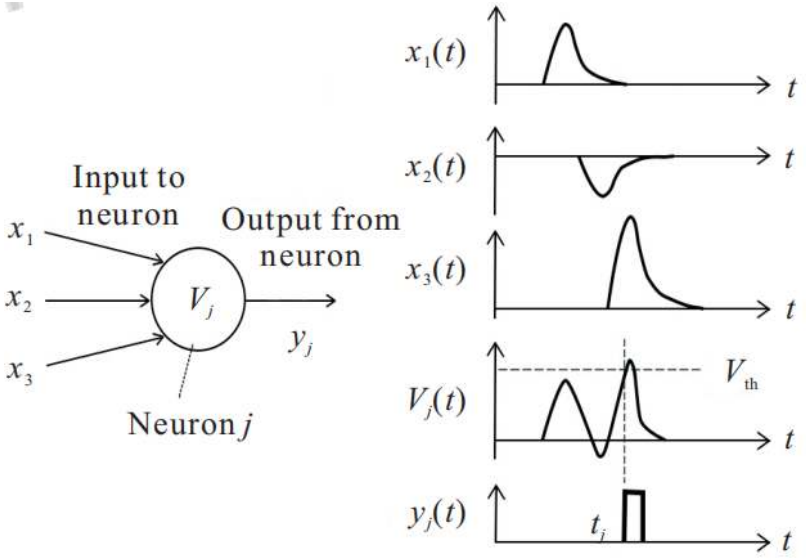



FIGURE 2.4: Schematic of a neuron in an SNN. Multiple inputs ($x_1, x_2, x_3$ from other neurons) activate the neuron, prompting a spike output $y_j$ when its potential $V_j$ exceeds a threshold $V_{\text{th}}$.

The potential weight in each neuron can be treated as a kind of memory, thus SNNs are able to provide temporally dynamic activation patterns and increased computing power [18, 19]. Different SNN learning algorithms have also been developed. The first unsupervised learning algorithms to be developed are based on Hebbian rules and Spike-Timing-Dependent Plasticity (STDP) [20]. Then there are more typical learning algorithms for SNN like gradient descent learning algorithm, [21] supervised STDP learning algorithm and learning algorithm based on pulse sequence convolution [22].

Another type of ANN is a Recurrent Neural Network (RNN). This is a recursive neural network that takes sequence data as input, recurses the evolution direction of the sequence, and all nodes (recurrent units) are connected

in a chain [23]. As the artificial neurons (nodes) in RNNs can participate several times in a network computation, efficiency is enhance. In RNNs, many neurons are interconnected through the same abstract synaptic connections (or links), so that activation can be propagated throughout the network. The difference between RNN and the more widely used FNN is that the connection topology in RNNs has many loops (i.e. delay lines) that produce nonlinear feedback and enable efficient temporal signal processing [24]. As shown in Fig. 2.5, the structure of a general RNN is fully recurrent, with every basic building block (artificial neuron) directly connected to every other basic building block. In this respect, it is similar to an FNN (i.e. fully connected). A crucial difference, however, is that information no longer flows only in one direction. This creates an internal state of the network that allows it to exhibit rich temporal dynamics [16].

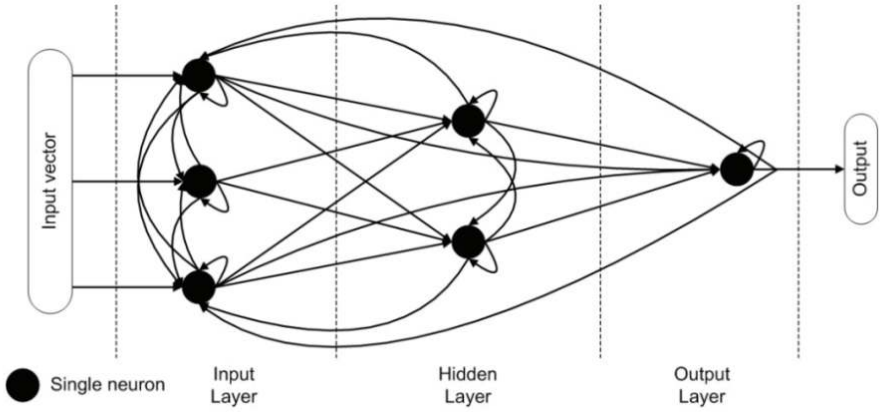



FIGURE 2.5: The structure of a general Recurrent Neural Network (RNN), in which the artificial neurons also support recursive connections (i.e. feedback loops), so information can be transmitted in any direction [16]

.

## 2.2 Neuromorphic computing

Although ANN methods are powerful, they have certain limitations, such as "catastrophic forgetting", which prevents them from being able to adapt to changing tasks [14, 25]. This lack of generalisability means that ANN approaches cannot lead to genuinely brain-like Artificial Intelligence (AI). As the best agent in nature, the biological brain is the most important reference for AI research. Therefore, studying brain mechanisms and developing

brain-inspired technologies are active research areas with potential for significant advances and impact in the future intelligence era [26].

The field of neuromorphic engineering is based around the broad goal of emulating the biological nervous system in electronic hardware [27]. An important sub-field of this is neuromorphic computing, which aims to emulate the information processing capability of neurons in the brain's neocortex [9]. Neuromorphic computing can be broadly subdivided into two approaches: emulating spiking neurons in silicon chips [28] and emulating synapses using novel nanomaterials that exhibit analogue conductance properties [14].

To achieve the high-performance, low-power, and parallel computing operating mechanism of the biological brain, neuromorphic computing goes beyond the conventional von Neumann architecture [29], implementing computing components that integrate memory and processing and that can mimic some of the biological functionalities thought to be important for learning [29].

### 2.2.1 Spiking neurons in silico

In neuromorphic computing research, a significant focus is on implementing SNN algorithms in silicon-based hardware. Silicon aim to neurons emulate the electrophysiological spiking behavior of real neurons when their membrane action potential exceeds a voltage theshold. These are implemented in hybrid analog/digital very large scale integration (VLSI) circuits [30]. Simulation of spiking neurons in an SNN framework is more power efficient in hardware than in software. This is in large part attributed to the co-location of processing and memory in neuromorphic chips. Commercial neuromorphic chips, such as IBM's TrueNorth and Intel's Loihi, have been developed specifically as hardware accelerators for ANN training, which is notoriously power hungry. Fig.2.6 compares the power consumption of various silicon-CMOS based chips, including neuromorphic chips, to that of other technologies and to the brain. Note, however, that in terms of energy per operation, the brain remains superior to all other existing technologies, including neuromorphic chips.

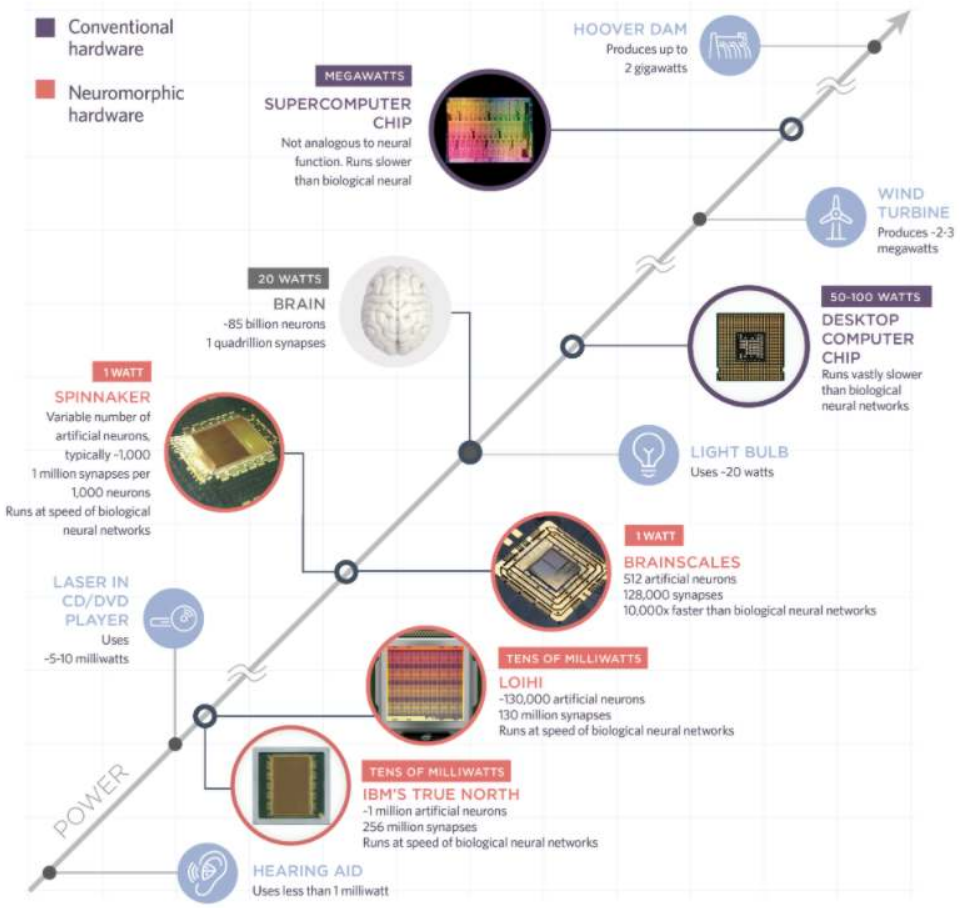



FIGURE 2.6: Infographic comparing power consumption of neuromorphic and conventional computing hardware (red and black circles, repsectively). Also shown for comparison are power consumption of other man-made technologies and of the brain. *Image credit:* www.the-scientist.com/infographics/infographic–brain-like-computers-provide-more-computer-power-65799.

Silicon-CMOS-based circuitry offers a medium suitable for large-scale neuromorphic computing, as has been demonstrated with the EU-funded BrainScaleS and SpiNNaker systems[1]. BrainScaleS is based on mixed-signal and analog emulations of spiking neurons and synaptic plasticity models. SpiNNaker, on the other hand, is used to train and run SNN models (scripted in Python) in real-time on its digital multicore neuromorphic chips. In general, VLSI circuits provide tangible advantages in the investigation of questions concerning the strict real-time interaction of the system with its environment [31, 32]. They are also suitable for various platforms like single chips or distributed across chips. These characteristics make this field very promising.

[1] humanbrainproject.eu/en/silicon-brains/

Fig.2.7 shows an example of a basic Si-CMOS circuit implementing the Integrate-and-Fire (I&F) model. The I&F model is the simplest impulse neuron model. It abstracts the neuron as an RC circuit. After a neuron receives input current, its membrane potential rises until it reaches the activation threshold and releases the pulse ("fires"). After the pulse is released, the neuron membrane potential immediately returns to the resting potential [33, 34, 17].

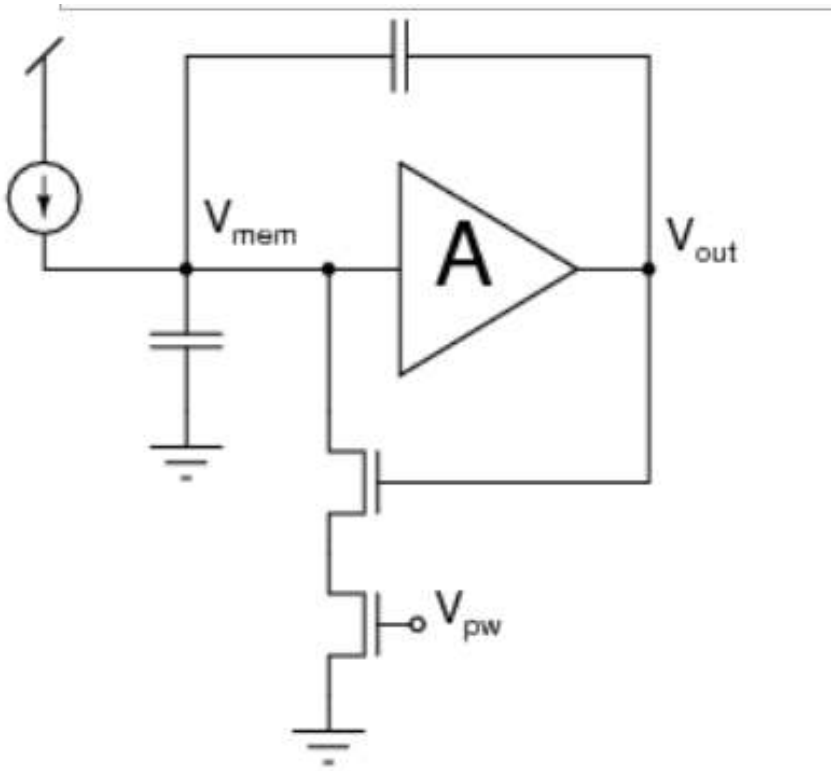



FIGURE 2.7: An example of a basic circuit that implements the integrate-and-fire model of spiking neurons [30].

While SNNs offer enormous promise for realising a more brain-like learning paradigm than ANNs (which technically are optimisation algorithms), no existing SNN algorithm can match the performance of ANNs on published datasets. However, recent research has demonstrated this may be overcome with a novel algorithm that converts a trained ANN model into very sparse spikes (only two) to achieve highly efficient learning [35]. It will be interesting to see if the predicted efficiency gains can be realised in neuromorphic hardware [36].

### 2.2.2 Memristor synapses

The concept of a memristor was first proposed by Chua [37] and Strukov and Williams [38]. Decades later, this concept was generalized to a class of nanoelectronic devices exhibiting resistive switching memory, as more and more memristive materials were discovered. The following paragraph briefly summarizes the characteristic fingerprints of memristive devices [39].

The first fingerprint is a pinched hysteresis loop that passes through the origin in an $I - V$ response curve [40]. The response to periodic excitation could be a periodic steady-state solution or a chaotic attractor solution, where the $I - V$ curves deviate in each sweep, but remain relatively localized in the phase-space. In both cases, the solution must pass through the origin, i.e. $i(t) = 0$ when $v(t) = 0$. With the shortening of the periodic excitation period, or a decrease in amplitude, the area enclosed by the response curve also decreases and theoretically eventually becomes zero (see Fig. 2.8). This is because the memristance is a piecewise-continuous function of $I/V$, so the integral of the change of the memristance over time is proportional to $I/V$ over the time interval.

The second fingerprint of a memristor is its response is inversely proportional to frequency. This behaviour is evident at frequencies $\omega$ above a threshold $\omega^*$. As $\omega$ increases, the area enclosed by the $I - V$ loops eventually becomes zero. This character means the interval of memristance over time has a certain range which depends on a particular model.

Since 2000, researchers have studied the use of a variety of materials to make physical non-volatile memristive devices [10]. The vast majority are bulk metal oxides (e.g. $TiO_2$ [41]), but other more novel devices have been realised with various low-dimensional nanomaterials (e.g. Ge nanowires [42]).

## 2.3 Reservoir computing

Reservoir computing is a sub-category of neuro-inspired computing that uses an RNN as a higher-dimensional reservoir [43]. The concept of a reservoir is inspired by the information processing capacity of the human brain, which is a high dimensional and complex system. [44] Thus typically, the reservoir is

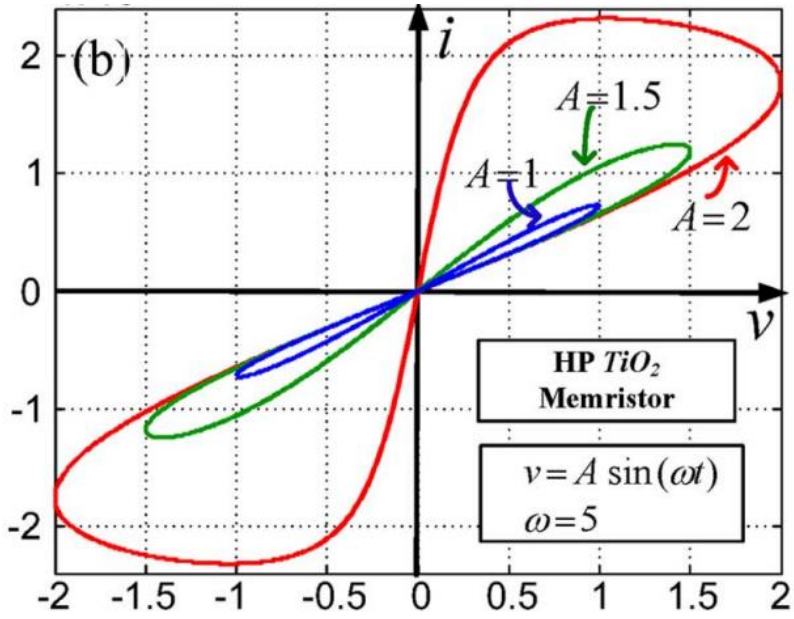



FIGURE 2.8: $I - V$ curves of the Hewlett-Packard (HP) $TiO_2$ memristor for different amplitudes $A$ of the input periodic voltage signal. Each loop exhibits the pinched hysteresis that is characteristic of memristors [40].

a sparse, large, undirected or directed recurrent network with many nodes. The input and the previous states of the reservoir's nodes determine the current states which can be treated as the output of the reservoir. The node outputs are trained, usually using a simple linear regression or decision tree, or classification scheme (i.e. supervised learning). This is the general process of reservoir computing. A characteristic feature of this process is that the reservoir has a fixed internal structure and a certain amount of memory [43, 45, 24]. The reservoir must be complex enough to capture all the salient features of the input and behave as a time-dependent non-linear kernel function that maps the input into a higher-dimensional feature space with linearly separable outputs [46]. This approach is inspired from Cover's theorem [47] that states that if a high dimensional ANN meets certain mathematical properties, training only the readout layer is sufficient to achieve excellent performance [24].

### 2.3.1 Echo state networks

There are two main reservoir computing models, namely echo-state network (ESNs) [48] and Liquid State Machines (LSMs) [49]. ESNs are based on continuous reservoir states (RNNs are used in ESNs), whereas LSMs are based

on threshold states, similar to spiking neurons (SNNs are used in LSMs). Chua [50] experimentally proved that ESNs and LSMs are essentially identical. Although the two methods have different angles, their essence can be considered as an alternative to traditional RNN training algorithm.

As mentioned above, a reservoir needs to have memory. This memory can be described as the output of the reservoir at any time being dependent on previous states [48]. Additionally, in ESNs, the influence of input on output decreases gradually with time. This property is similar to the "echo" in a closed room, where the correlation of the data is the "sound", and the reservoir itself is the echo "room". As well as using an abstract mathematical RNN as the reservoir, it is also possible to build an ESN with hardware using memristors which will be described below in Sec.2.3.2.

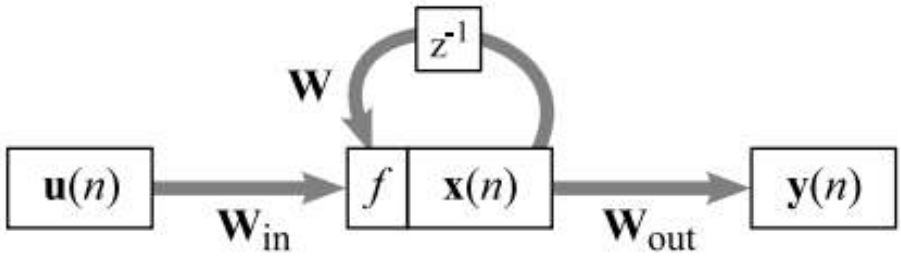



FIGURE 2.9: The simplest ESN includes an input weight $W_{\text{in}}$, output weight $W_{\text{out}}$ and a dynamical reservoir $W$. $Z^{-1}$ represents the effective time delay in the recurrent dynamics of the reservoir. $\mathbf{u}(n), \mathbf{x}(n), \mathbf{y}(n)$ are the states of inputs, ESN and outputs, respectively [24].

Fig.2.9 shows the simplest structure of an ESN. The inputs $\mathbf{u}(n)$ are weighted by $\mathbf{W}_{\text{in}}$, while the outputs $\mathbf{y}(n)$ are obtained by weighting the reservoir states $\mathbf{x}(n)$ by $\mathbf{W}_{\text{out}}$. $Z^{-1}$ represents the time delay related to the recurrent dynamics of the reservoir. $\mathbf{W}_{\text{in}}$ is constant while $\mathbf{W}_{\text{out}}$ is trained and $\mathbf{W}$ dynamically evolves in time. The training process generally only needs to solve a linear problem. Compared to conventional RNN training, the biggest advantage of ESNs is that training $\mathbf{W}_{\text{out}}$ is vastly more efficient than training $\mathbf{W}$.

### 2.3.2 Physical reservoir computing

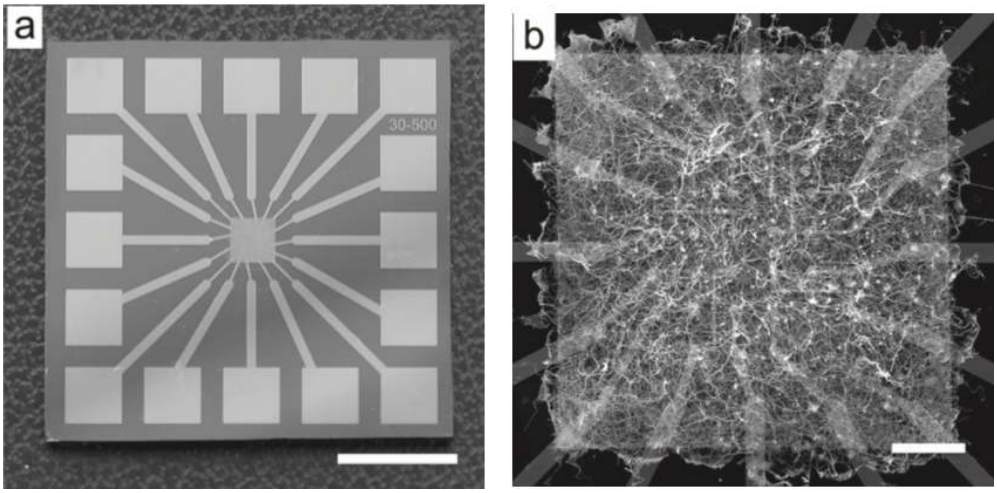
Physical reservoir computing is based on the concept that any physical dynamical system has the potential to serve as a reservoir if it meets several requirements. According to Tanaka et al. [51], these include: high dimensionality, non-linearity, fading memory and separability of outputs. A number of different physical reservoir models fulfil these requirements, including in particular ones based on analog circuits [52] and memristors [37]. Memristor-based physical RC is attractive because of the synapse-like dynamical electrical switching properties of memristive devices [53, 54, 37, 55, 56]. This has led to the development of a class of neuromorphic systems and circuits in which memristive RC has been successfully implemented, as evidenced by the ability to perform benchmark learning tasks such as hand-written digit and speech pattern recognition, as well as the Mackey-Glass nonlinear time series forecasting task [57, 58, 59].

Nanowire networks represent another class of neuro-memristive systems with an additional neuromorphic property: their neural network-like topology, which arises from their self–assembly, analogous to biological neural networks [60, 61] (cf. Fig.2.10). Inorganic nanowires comprised of silver readily self–assemble into a complex network, forming two-terminal memristive junctions where nanowires intersect. Memristive switching occurs at the junctions as a result of the formation of a conductive Ag filament above a voltage threshold [62].

FIGURE 2.10: Comparison of network graphs. Left – C. Elegans biological neural network (277 neurons) [60]. Middle – nanowire network (300 nanowires). Right – random network (300 nodes) [63].

Previous experimental and simulation studies based on Ag-$Ag_2S$-Ag atomic switch nanowire networks developed by Stieg, Gimzewski and colleagues at UCLA demonstrated their potential for physical RC [64, 62, 65, 66, 67]. While

the synthesis of those particular nanowire networks was aided by a pre-patterned substrate, the resulting network topology was sufficiently complex to observe emergent nonlinear (i.e. power-law) dynamics at the network level. Figure 2.11 shows a Multi-Electrode Array (MEA) device fabricated for $Ag_2S$-Ag atomic switch nanowire network measurement and characterisation. Ag nanowires were grown by self-assembly on a substrate that was lithographically pre-patterned with seed Cu posts [65], thus demonstrating their compatibility with CMOS technologies.




(a) (b)

FIGURE 2.11: (a) Multi-Electrode Array (MEA) fabricated for Ag-$Ag_2S$ atomic switch nanowire network characterisation (scalebar = 4 mm). (b) SEM image (scalebar = 0.5 mm) of MEA atomic switch nanowire network device [66].

Fig 2.12 shows the $I - V$ curve of an $Ag_2S$-Ag atomic switch nanowire network [67]. Two main states (related to the on/off of current path) mark the formation of the conductive Ag filament. This differs somewhat from a traditional memristor, which stores information by multiple conductance states, while nanowire networks accomplish this by network dynamics [67]. The two stable states in Fig 2.12 correspond to zero current and Ohmic current transport regimes, whereas switching and fluctuations correspond to nonlinear, non-Ohmic regimes.

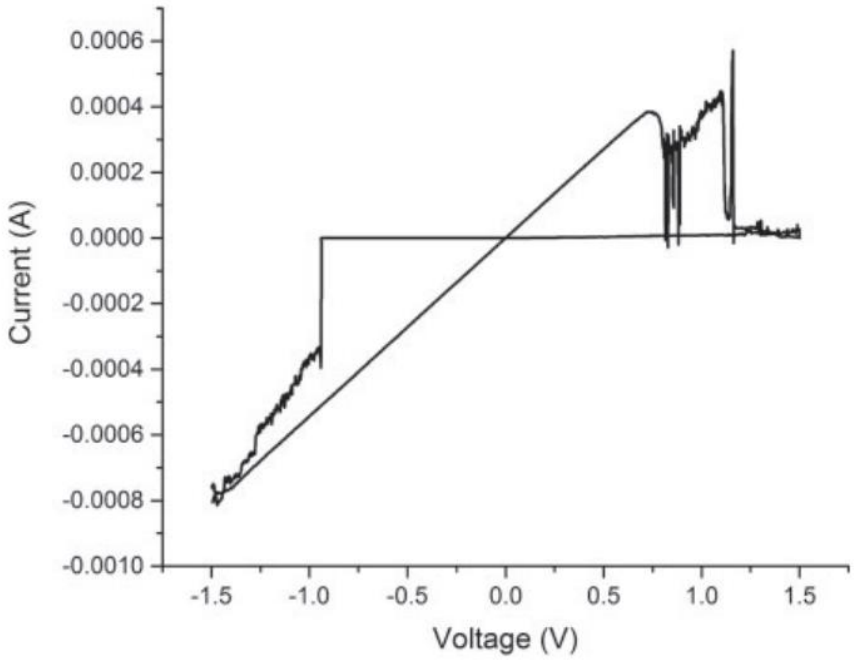



FIGURE 2.12: A 1.5 $V$ voltage sweep for an $Ag_2S$-Ag atomic switch network [67] which shows stable states (straight line segments) and hard switching (abrupt increase in $I$), as well as stochastic fluctuations.

Stieg et al. [68] and Avizienis et al. [64] also investigated the response of atomic switch networks to a constant DC bias. Fig. 2.13 shows current time series measured under 1 V DC over three different timescales. Persistent fluctuations are evident as well as abrupt changes in current that increase in magnitude $\Delta I$ over time. This behavior was attributed to voltage redistribution by recurrent loops in the network. The authors considered the influence of the DC bias on the filamentary mechanism of each junction. Stochastic (thermodynamic) breakdown of some filaments causes a memristive junction to reset. Under DC, the reset is partial, so switching events continue and cause increasing $\Delta I$. The authors also concluded that the current fluctuations persisting over long durations is a whole-of-network collective effect. The recurrent loops in the network are not steady under DC bias, so the voltage redistributes across the network continuously with the on and off switching of those loops.

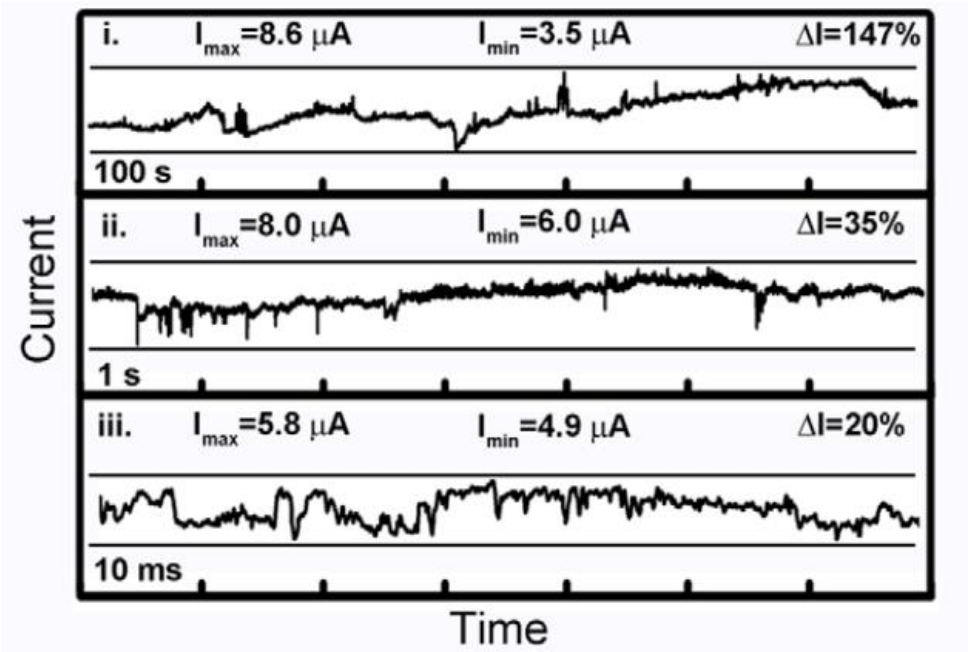



FIGURE 2.13: Current time series under 1 V DC bias for Ag-$Ag_2S$ atomic switch nanowire networks, showing abrupt jumps and persistent fluctuations of varying maximum magnitude, $\Delta I$, over varying timescales: 100 s (top), 1 s (middle) and 10 ms (bottom) [64].

Fig 2.14 shows the corresponding Power Spectral Density (PSD) of $I(t)$ measured for $Ag_2S$-Ag atomic switch networks and also for Ag-Ag nanowire networks (i.e. un-functionalised). A power-law over several decades in frequency is clearly evident for $Ag_2S$-Ag networks (with slope $\approx -1.3$), but not for Ag-Ag networks. This demonstrates that the memristive switching dynamics introduced by functionalising the nanowires results in scale-invariant non-linear dynamics at the network level. Similar broadband power-law spectra has also been found from electrical measurements of the human brain [69].

Fig. 2.15 shows the Fourier amplitude spectrum for current measured under a 2 V, 10 Hz AC input into an $Ag_2S$-Ag atomic switch network [64]. The spectrum reveals many higher harmonics. This higher harmonic generation was exploited by Demis et al. [67] to perform nonlinear wave transformation using an RC implementation. The authors input a 2 $V$, 1 Hz triangular wave into one source electrode in contact with the atomic switch network and read out current from other randomly selected electrodes, as shown in Fig. 2.16. Then the non-linear transformation task is implemented using this network. This time the input is sine-wave and the target is effectively transforming the

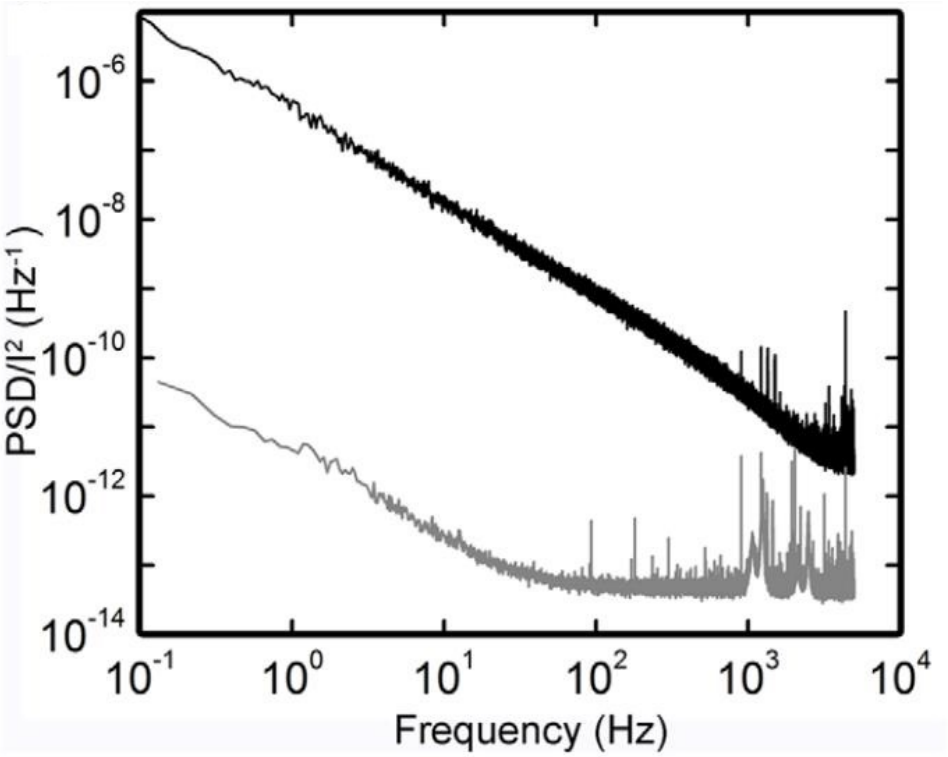



FIGURE 2.14: Power Spectral Density (PSD) of measured $I(t)$ under 1 V DC for $Ag_2S$-Ag atomic switch network (black) and an un-functionalised Ag-Ag network (grey). The slope of the power-law PSD (black) is $\approx -1.3$ [64].

input to cosine, triangle, sawtooth and square waveforms, as shown in Fig. 2.17. Linear regression was then applied to the readout signals. Accuracy is lowest for the square waveform regression because this requires infintely many higher harmonics.

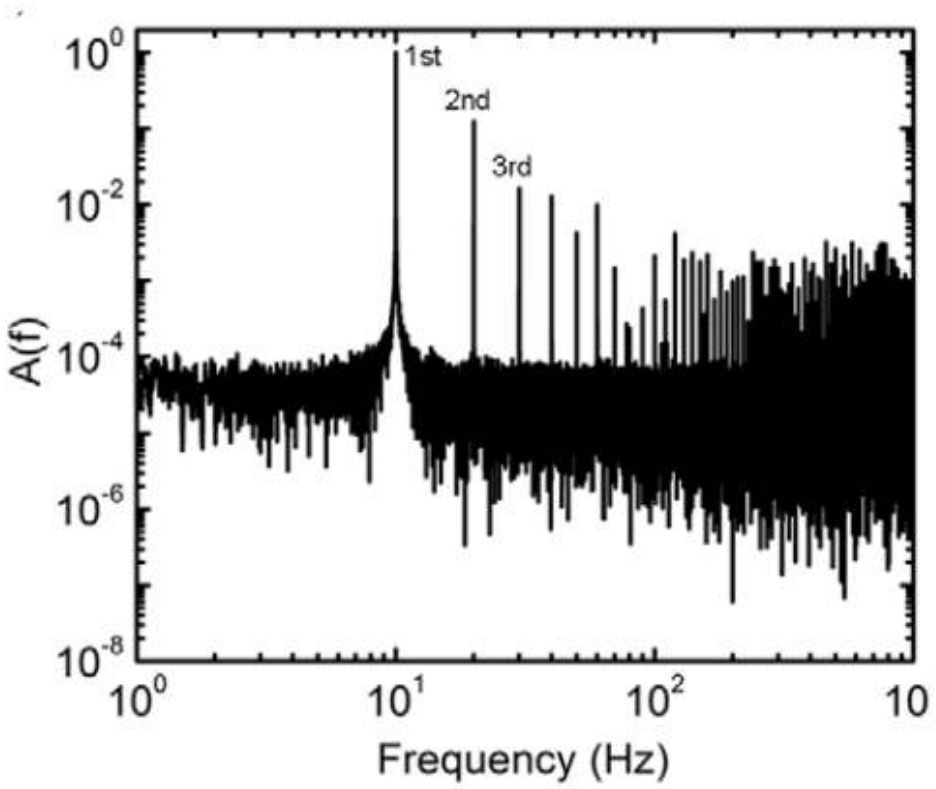



FIGURE 2.15: Fourier amplitude spectrum of current measured from an $Ag_2S$-Ag atomic switch network under AC input (2 V, 10 Hz) [64].

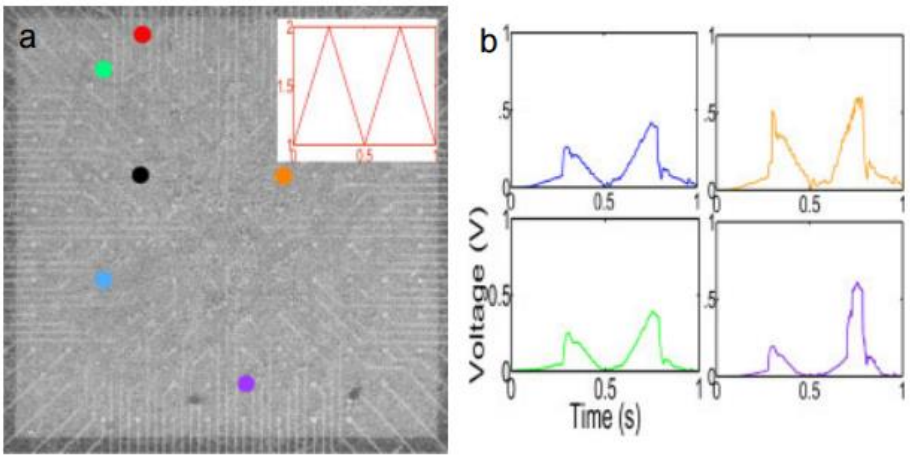



FIGURE 2.16: Reservoir computing implementation of nonlinear wave transformation using an $Ag_2S$-Ag atomic switch network device: (a) electrodes used for input (red circle), drain (black circle) and readout (other coloured circles). Inset shows the unipolar 2 V, 1 Hz triangular input signal. (b) Voltage readouts from other electrodes, showing nonlinear variations produced by the network [67].

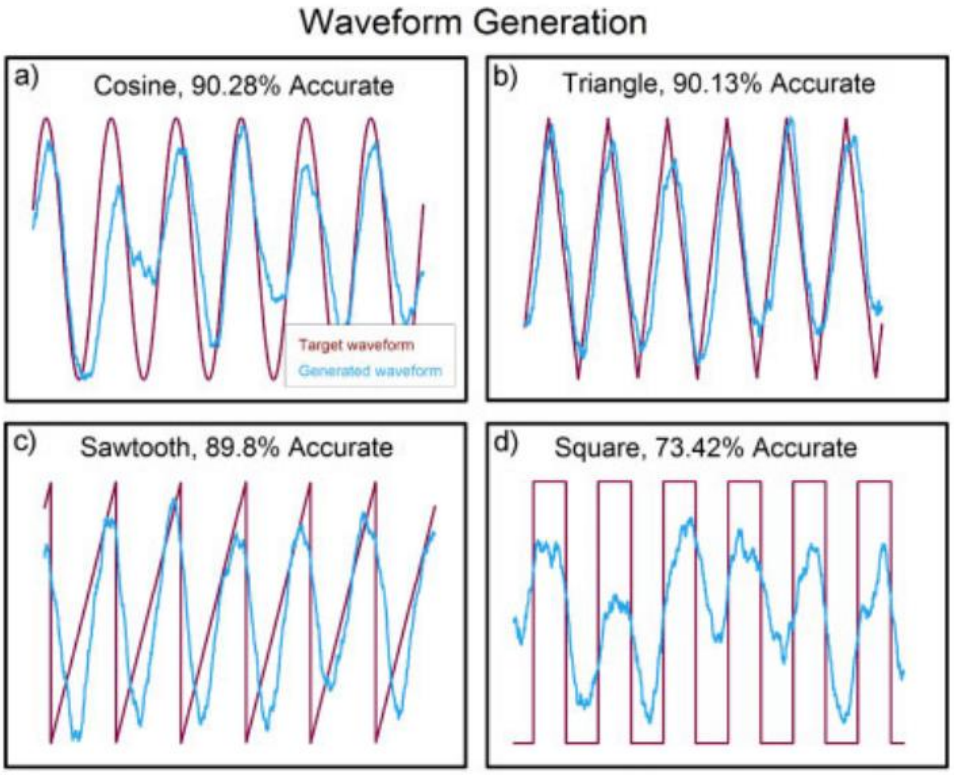



FIGURE 2.17: Nonlinear wave transformation RC task by Demis et al. [67]. Voltage readouts from an input 11 Hz sine wave were linearly regressed to generate different waveforms. Corresponding accuracies are shown.

Another implementation of physical RC is hand-written digit recognition, which is a benchmark machine learning task. This task requires not only non-linearity but a fading memory of the reservoir. Du et al. [57] studied the fading memory of memristors and performed handwritten digit recognition with separate memristors. Each digit from the MNIST data was first transformed to a temporal impulse stream row by row and input into separate memristors. The outputs of each memristor (either current or conductance) was read out. The set-up is shown in Fig.2.18. The authors used an ANN to perform classification on the readouts. The computing resources in this implementation was found to be much smaller than using an ANN directly in a conventional approach, while still achieving an accuracy of 90%.

Polymer (PVP) coated Ag nanowires also self–assemble into a densely interconnected, complex network (cf. Fig.2.19). When electrically activated, PVP-Ag nanowire networks also exhibit similar emergent dynamical properties as $Ag_2S$-Ag networks, even though their memristive junctions differ [70].

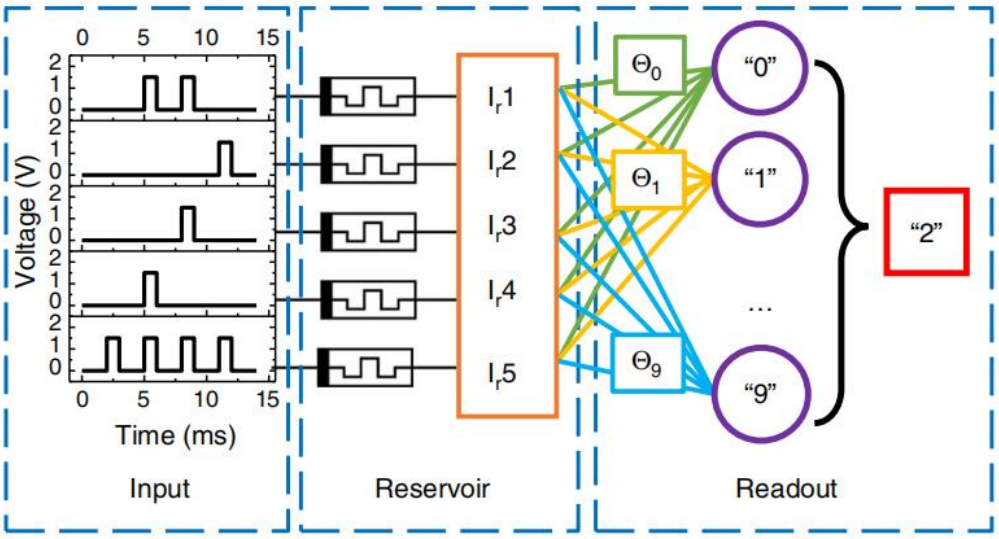



FIGURE 2.18: Physical RC implementation of hand written digit classification using individual separate memristors. The digits are transformed into multiple voltage streams (left box) and input into separate memristors (middle box). Classification is performed on the readouts (right box) [57].

It has been demonstrated that these PVP-Ag nanowire networks can associatively learn spatial patterns by recalling previously established current pathways [71, 72]. Studies to date show that PVP-Ag nanowire networks meet the requirements of high dimensionality, non-linearity, and fading memory, thus suggesting that physical RC may also be implemented on PVP-Ag nanowire networks. This thesis presents the results of a simulation study based on a physically motivated model of self-assembled PVP-Ag nanowire networks. The simulations allow us to modify various parameters at the junction level and to explore the junction-level dynamics, which is impossible to do experimentally. Simulations also enable us to develop and test learning tasks using the reservoir computing framework.

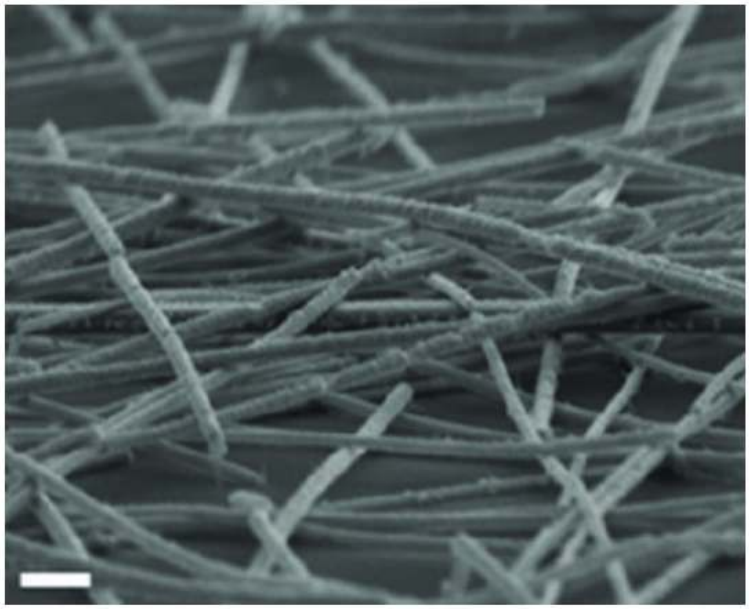


FIGURE 2.19: SEM image (scalebar = 0.5 $\mu$m) of self-assembled PVP-coated Ag nanowires [73].

## 2.4 Motivation and rationale for this thesis

This thesis is motivated by recent developments in neuromorphic information processing using self-assembled nanowire networks. These neuromorphic systems are unique in exhibiting not just memristive junctions, but also neural network-like circuitry. To better understand the potential of neuromorphic nanowire networks for information processing, a simulation study was carried out using a physically motivated model based on PVP-Ag nanowire networks. The focus of this thesis is on the simulation results that demonstrate the rich dynamical behavior of nanowire networks and how these dynamics can be harnessed for information processing in a reservoir computing framework. Several benchmark tasks are implemented and their performance evaluated. It is envisaged these simulations will provide valuable information on how to implement similar tasks on a physical nanowire network neuromorphic hardware device.

# Chapter 3

# Nanowire network model and reservoir computing implementation

## 3.1 Introduction

As discussed in the previous Chapter, neuromorphic nanowire networks exhibit memristive switching junctions due to conductive filament formation by electro-chemical metalisation [74]. In addition, their network topology is sparse, exhibiting small-worldness and modularity [60]. This makes electrical signal transmission more complex to analyse than in a conventional electrical circuit of bulk memristors. This warrants modeling and simulation to gain insights into the switch and network dynamics that could not otherwise be directly gained from experimental measurements. This chapter investigates the switch junction and network dynamics under an applied bias using simulations based on a physically–motivated model, outlined in Sec.3.2.1. Results are presented in Sec.3.3.1 that shows how network-level states are related to junction states, which are determined by the applied voltage and the evolution of filament states. The impact of dynamical voltage distribution on current path formation is studied under a DC bias and network $I - V$ curves for an AC input signal are also presented and compared with previous studies.

Following this, reservoir computing (RC) methods (outlined in Sec.3.2.2) are used to perform three temporal signal processing supervised learning tasks: (i) nonlinear wave transformation, where an input sine wave is linearly regressed to different waveforms; (ii) sine wave generation, where auto-regression is used to predict the next time step; and (iii) handwritten digit recognition using linear discriminant analysis (LDA) classification. The results are presented in Sec.3.3.2. Previous studies that have implemented RC using memristors [57] and memristive atomic switch networks (ASNs) [62, 67] demonstrated that the nonlinear memristive properties are useful for reservoir learning from temporal input signals. Demis et al. [67] and Sillin et al. [62] performed the nonlinear transformation RC task using ASNs both experimentally and in simulation, respectively. Du et al. [57] performed handwritten digit recognition using RC on a memristor hardware device. We compare our RC simulation results for nanowire networks to those obtained in these previous studies. It is envisaged that our simulations will inform RC implementation on a nanowire network hardware device in a follow-up study.

Some of the results presented in this chapter were presented at the 2020 International Joint Conference on Neural Networks (IJCNN), which was part

of the IEEE World Congress on Computational Intelligence (WCCI), held (virtually) in Glasgow, UK, 19-24 July 2020. The presented paper has been published as the following peer-reviewed article:

> K. Fu, R. Zhu, A. Loeffler, J. Hochstetter, A. Diaz-Alvarez, A. Stieg, J. Gimzewski, T. Nakayama, Z. Kuncic, "Reservoir Computing with Neuromemristive Nanowire Networks", 2020 International Joint Conference on Neural Networks (IJCNN), 2020, pp. 1-8, 10.1109/IJCNN48605.2020.9207727 [1].

## 3.2 Methods

All simulation results are based on a model developed by Kuncic et al. [75] and Kuncic et al. [76]. The model has been validated against experimental measurements [75, 70, 74]. All simulations were performed in Matlab 2019a.

### 3.2.1 Modeling switch and network dynamics

Self-assembled nanowire networks were modeled, as described in Kuncic et al. [75] and Kuncic et al. [76]. Nanowire–nanowire junctions were modeled as ideal voltage–controlled memristive junctions described by a state-dependent Ohm's law, $I = G(\lambda)\, V$, where $I$ is current, $G$ is conductance, $V$ is voltage and $\lambda = \lambda(t)$ is a state variable representing flux in memristor theory [64, 70]. At each junction, $\lambda$ represents the conducting filament which parameterizes the conductance. All junctions are initially in a high resistance "off" state. After a voltage bias is established, an individual junction switches to a low resistance "on" state when $\lambda \geq \lambda_{\text{crit}}$, where $\lambda_{\text{crit}}$ is a set threshold. The ratio of these resistance states is set to $R_{\text{off}}/R_{\text{on}} = 10^3$, with $R_{\text{on}} = G_0^{-1}$, where $G_0 = (13\,\text{k}\Omega)^{-1}$ is the conductance quanta.

A junction's state $\lambda(t)$ depends on its past history of voltage input. The evolution of $\lambda(t)$ is prescribed by the following bipolar threshold voltage

model [75, 76]:

$$\frac{d\lambda}{dt} = \begin{cases} (|V(t)| - V_{\text{set}})sgn[V(t)] \,, & |V(t)| > V_{\text{set}} \\ 0 \,, & V_{\text{reset}} < |V(t)| < V_{\text{set}} \\ b(|V(t)| - V_{\text{reset}})sgn[\lambda(t)] \,, & |V(t)| < V_{\text{reset}} \end{cases} \tag{3.1}$$

where $V_{\text{set}}$ is the on-threshold and $V_{\text{reset}}$ is the off-threshold, and $b$ is a positive constant relating the relative rates of decay and formation of the conductive filament, such that when a junction switches off, the filament state can immediately return to the state $\lambda = \lambda_{crit}/b$ to suppress fluctuations around $\lambda_{\text{crit}}$. The following default parameter values were used for all the simulation results presented here: $V_{\text{set}} = 10^{-2}$ V, $V_{\text{reset}} = 10^{-3}$ V and $b = 10$. These values were found to produce simulation results that most closely matched experimental measurements [75, 70].

A self–assembled nanowire network was modeled using equation (3.1) for the junctions and a modified nodal analysis [77] to solve Kirchoff's circuit law equations at each time point. This is done by first defining the weighted adjacency matrix $W$ and the weighted degree matrix $D$ as

$$D = \text{diag}\,(d_i) \quad , \quad d_i = \sum_{k=1}^{N} W_{ik} \quad , \tag{3.2}$$

to obtain the graph Laplacian $\mathcal{L} = D - W$ and the expanded graph Laplacian of the network:

$$\mathcal{L}^{\dagger} = \left[\begin{array}{c|c} \mathcal{L} & C \\ \hline C & 0 \end{array}\right] \quad ,$$

where $C$ is either 1 if the nanowire node is connected to an external electrode or 0 otherwise. Finally, the system of linear equations $\mathcal{L}^{\dagger}V = I$ is solved at each time point to obtain the conductances of each junction [63].

As shown in Fig. 3.1, the model simulates wires scattered on a plane with uniformly random distributions of orientations and positions and with lengths sampled from a gamma distribution. At the intersection of overlapping nanowires, memristive junctions that have switched on are shown as large circles, while those that are off are small squares. The wire and junction color reflect the network voltage distribution (with wires as equipotentials) and white arrows indicate the current flow across the network, from source (green star) to drain (red star). The voltage distribution and hence current flow continuously adapt to the dynamically changing memristive junctions

and to the network's structural connectivity [60].

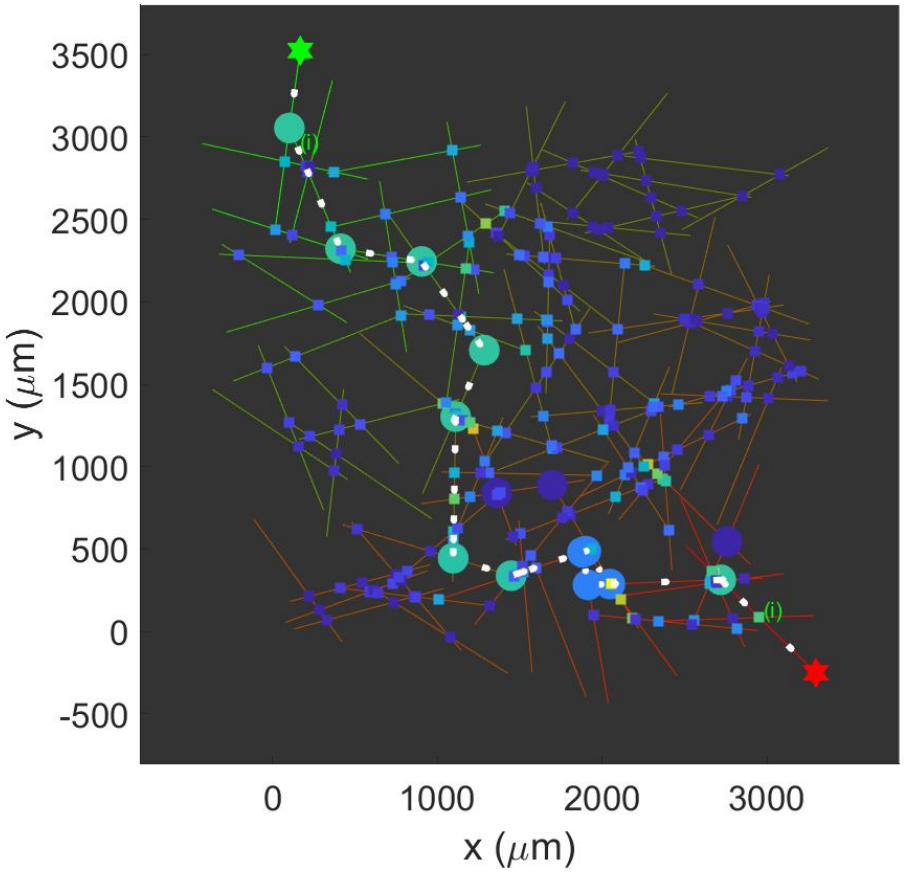



FIGURE 3.1: Snapshot visualisation of a neuromorphic nanowire network simulation: self-assembled nanowires form a neural-like network, with each intersection of overlapping nanowires forming a memristive junction. For visual clarity, a network with only 100-nanowires and 262-junctions is shown. A voltage bias applied across the network (green and red stars denote source and drain) induces memristive switching from a low conductance state (small squares) to a high conductance state (large circles). Wire and junction color reflects voltage distribution and white dashes denote current flow.

### 3.2.2 Reservoir computing implementation

Reservoir Computing (RC) was implemented using the nanowire network as a reservoir by allocating specific input and readout nodes and training the readout using either linear regression or classification, depending on the task. Two temporal information processing tasks were performed: (i) nonlinear waveform transformation; and (ii) wave generation. A third task, MNIST

handwritten digit classification, was also performed by converting the images into a continuous-time stream and delivering them to network nodes.

For the temporal tasks using reservoir computing, the output signal error was calculated using the mean square error (MSE):

$$\text{MSE} = \frac{1}{M}\sum_{m=1}^{M}[T(t_m) - y(t_m)]^2 \tag{3.3}$$

where $T(t_m)$ is the target signal, $y(t_m)$ is the trained readout result, and $M$ is number of time points. Performance accuracy is quantified by $1 - \text{RNMSE}$, where RNMSE is the root–normalized MSE:

$$\text{RNMSE} = \sqrt{\frac{\sum_{m=1}^{M}[T(t_m) - y(t_m)]^2}{\sum_{m=1}^{M}[T(t_m)]^2}} \tag{3.4}$$

**Nonlinear transformation**

For this task, RC was implemented by setting two of $N$ total nanowires in the network as the source and drain. The voltage of all other nanowire nodes was used to record the reservoir state at each time point. In an experimental setting, this readout scheme may be implemented using a multi-electrode array. A sine wave was used as the input voltage signal and the target signal $T(t)$ was one of either four waveforms: sawtooth, square, doubled–frequency sine, and cosine. The voltage readout was trained by linear regression to nonlinearly transform the input by solving the system of linear equations

$$X(t)\theta^T = T(t) \tag{3.5}$$

where

$$X(t) = [v_1(t), v_2(t), v_3(t), \cdots, v_n(t)] \tag{3.6}$$

is the reservoir state, represented by a vector containing $n$ elements (the voltage values of the $N$ nanowires in the network reservoir), and $\theta$ is the output weight. The procedure is shown schematically in Fig.3.2.

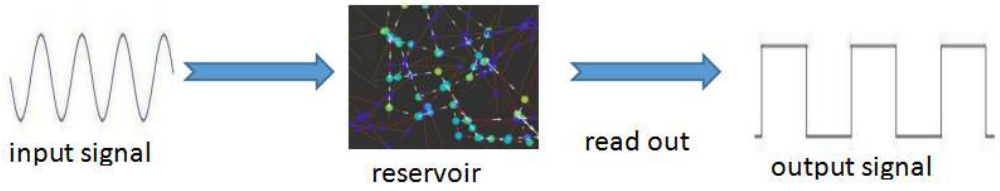



FIGURE 3.2: Schematic depicting nonlinear transformation of an input signal (e.g. sine wave) using a nanowire network reservoir. In our implementation, one reservoir node (nanowire) receives the input signal and others serve as readout nodes which are trained by linear regression to produce the desired output signal (e.g. square wave).

To train the readout and solve (3.5) for $\theta$, the MSE was used as the cost function,

$$J(\theta) = \frac{1}{M}\sum_{m=1}^{M}[T(t_m) - y(t_m)]^2 \tag{3.7}$$

with $y(t) = X(t)\theta^T$, and minimized using the gradient descent method:

$$\frac{\partial J(\theta)}{\partial \theta_n} = \frac{1}{M}\sum_{m=1}^{M}[T(t_m) - y(t_m)]v_n(t_m) \tag{3.8}$$

where $\theta_n$ is the weight for each readout node ($N$ totally readouts). The training of weights was achieved in Matlab 2019a using the *regress* command.

To increase the accuracy of regression, piecewise linear regression was used [78]. Using 1000 time points in each period, the input and target signals were segmented into several equal ($i$) intervals that were linearly regressed independently, i.e.

$$X(t_i)\theta_i^T = T(t_i) \tag{3.9}$$

where $X(t_i)$ is the voltage readout of every nanowire in time interval $t_i$ and $\theta_i$ are the corresponding output weights. Each segment is a matrix, with rows representing the nodes ($n$) and columns representing the time points $t_i$ of the current segment.

**Wave auto–generation**

This task is performed differently from nonlinear waveform generation. The goal is to generate the desired wave without any external input after a training period. The signal was divided into eight equal intervals. During the

first 1/8 interval, the network was primed with the input signal (same sine wave as used in the previous task) delivered to one source node, and no output is readout. In this period the network is pre-activated. The training was applied during the 2/8 – 4/8 period by delivering a teacher sine signal $u(t)$ to the source node. The voltage of every node was read out and the output weights calculated using linear regression.

The predicted next time point value of the target signal $u(t)$ is a linear combination of $u(t-1)$ and the readout signal values at $t$ and $t-1$ according to the following auto–regression equation:

$$u(t) = \theta_{2N+2} v_0 + \theta_{2N+1} u(t-1) + \vec{\theta} \cdot [\vec{v}(t-1), \vec{v}(t-2)] \tag{3.10}$$

where $v_0 = 1\,\mathrm{V}$, $\vec{v}(i)$ is the $N$-element vector containing absolute values of voltage readout from each node at time $i$, and $\vec{\theta}$ is the corresponding $2N$-element weight vector, so that the output weights are given by

$$\Theta = [\theta_1, \theta_2, \theta_3, ..., \theta_{2N}, \theta_{2N+1}, \theta_{2N+2}] \tag{3.11}$$

This recording of history of states is also referred to as the virtual nodes method when the network is a delayed feedback system [58]. Two virtual nodes are set in the sine wave generation task. After the training period, the input was set to zero. The last 4 intervals (from 5/8 to the end) is the wave generating period, when $u(t)$ is used as input for the next time point. In this way, the network generates a wave without any external input.

### Handwritten Digit Classification

The nanowire network model was also applied to the popular benchmarking task of classifying handwritten digits from the Mixed National Institute of Standards and Technology (MNIST) database of handwritten digits [79]. The database consists of a total of $70,000$ gray-level images of hand-written digits from 0 to 9, subdivided into a training set (60,000 digits) and testing set (10,000 digits). A sample of digits is shown in Fig.3.3. Each digit is a $28 \times 28$ pixel matrix with a $0 - 255$ gray-scale level.

FIGURE 3.3: Sample of handwritten digits from the MNIST database. Each digit image is a matrix of $28 \times 28$ pixels with 256 gray-scale levels. *Image credit: blog.csdn.net/weixin_44613063.*

To implement this classification task using an RC approach, which is inherently dynamic, it is necessary to input the data to the network as a time-series stream. Thus, pre-processing the data involved normalizing the pixel gray-levels to a value in $0 - 1$, as shown in Fig.3.4, and then delivering this as the continuous voltage signal with duration $\Delta t$ per pixel. The digit image was streamed either row-by-row or column-by-column, thus creating 28 time-series voltage vectors.

For each digit, the row or column voltage streams are input to 28 randomly selected network nodes (representing contact electrodes) and one node is set as a drain. Fig. 3.5 shows an example of the input to one electrode-node for the digit '5' using $\Delta t = 0.1$ s (i.e. 2.8 s stream per electrode). The 28 input streams are delivered to the network electrode nodes simultaneously. Conductance between the electrode and drain is read out at 8-time points selected evenly across each 2.8 s stream. The readout at other time points (e.g. 4, 6, 10-time points) was also investigated, although only the 8-time-points readout results are shown here as this was found to provide the best results for this implementation. Both rows and columns are converted to streams and input to the network separately. Thus, the total number of features for each image

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.01 | 0.07 | 0.07 | 0.07 | 0.49 | 0.53 | 0.69 | 0.10 | 0.65 | 1.00 | 0.97 | 0.50 |
| 0 | 0 | 0 | 0 | 0.12 | 0.14 | 0.37 | 0.60 | 0.67 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.88 | 0.67 | 0.99 | 0.95 | 0.76 | 0.25 |
| 0 | 0 | 0 | 0.19 | 0.93 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.98 | 0.36 | 0.32 | 0.32 | 0.22 | 0.15 | 0 |
| 0 | 0 | 0 | 0.07 | 0.86 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.78 | 0.71 | 0.97 | 0.95 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0.31 | 0.61 | 0.42 | 0.99 | 0.99 | 0.80 | 0.04 | 0 | 0.17 | 0.60 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0.05 | 0.00 | 0.60 | 0.99 | 0.35 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.55 | 0.99 | 0.75 | 0.01 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.04 | 0.75 | 0.99 | 0.27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.14 | 0.95 | 0.88 | 0.63 | 0.42 | 0.00 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.32 | 0.94 | 0.99 | 0.99 | 0.47 | 0.10 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.18 | 0.73 | 0.99 | 0.99 | 0.59 | 0.11 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.06 | 0.36 | 0.99 | 0.99 | 0.73 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.98 | 0.99 | 0.98 | 0.25 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.18 | 0.51 | 0.72 | 0.99 | 0.99 | 0.81 | 0.01 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.15 | 0.58 | 0.90 | 0.99 | 0.99 | 0.99 | 0.98 | 0.71 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0.09 | 0.45 | 0.87 | 0.99 | 0.99 | 0.99 | 0.99 | 0.79 | 0.31 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0.09 | 0.26 | 0.84 | 0.99 | 0.99 | 0.99 | 0.99 | 0.78 | 0.32 | 0.01 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0.07 | 0.67 | 0.86 | 0.99 | 0.99 | 0.99 | 0.99 | 0.76 | 0.31 | 0.04 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.22 | 0.67 | 0.89 | 0.99 | 0.99 | 0.99 | 0.99 | 0.96 | 0.52 | 0.04 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.53 | 0.99 | 0.99 | 0.99 | 0.83 | 0.53 | 0.52 | 0.06 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

FIGURE 3.4: Data matrix of MNIST digit '5' after normalization.

is $28 \times 2 = 56$ and the dimension of whole output matrix is $448 \times 60,000$.

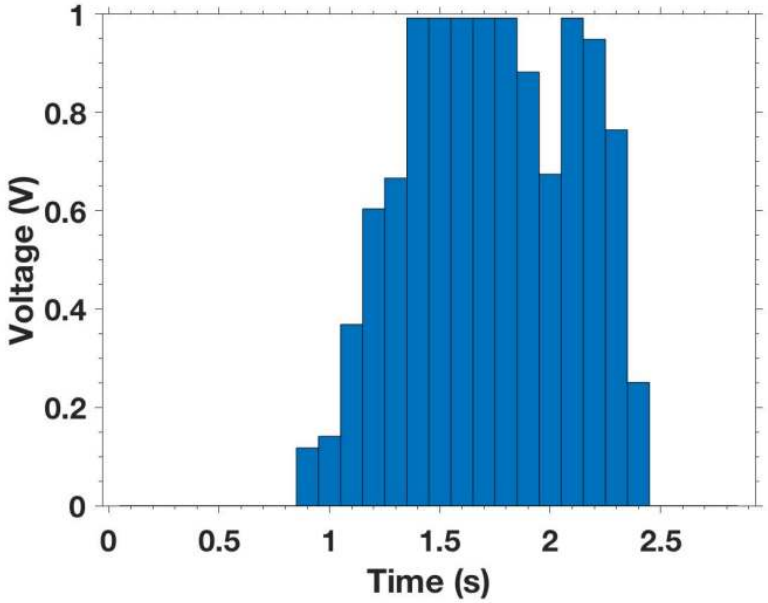



FIGURE 3.5: Example of voltage time stream for one row of digit '5'. The time-step is 0.1 s, corresponding to the duration of each pixel value.

For the readout, Principal Component Analysis (PCA) is applied to reduce dimensionality before classification training using Linear Discriminant Analysis (LDA). Other training methods were also considered (e.g. random forest and bagging), however LDA was found to be superior. LDA is most useful for multi-class classification. It attempts to find a linear combination

of the features of different classes in order to be able to characterise or distinguish them. The resulting combination can be used as a linear classifier, or for dimensionality reduction for subsequent classification.

## 3.3 Results and discussion

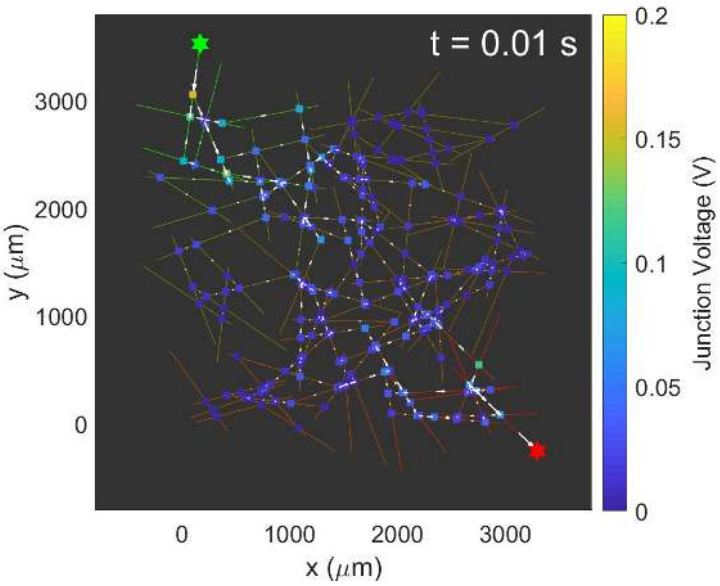
Unless otherwise indicated, all the simulation results presented here are for a 100-nanowire network, with 261 junctions. This is a default network used to test results for a range of applications; other network sizes were also used to investigate effect on different tasks.

### 3.3.1 Switch dynamics

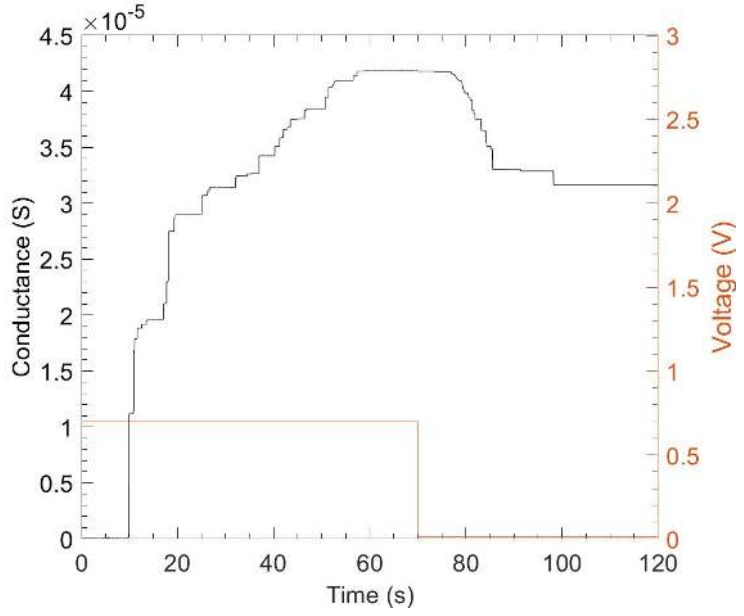
Fig. 3.6 (a) shows a snapshot of the network structure upon initialisation, where all junctions (squares) are in a high-resistance state ($R_{off}$ = 10 k$\Omega$ and $\lambda = 0$). Fig. 3.6 (b) shows the one source and one drain network's conductance response (blue) to a square input voltage pulse (red) of amplitude 0.7 V and duration 70 s, after which the voltage drops to 0.015 V.

This stimulation protocol was used to analyze the effect of the network's circuit loops on local voltage redistribution and its memory property as the response persists with a relatively slow decay after the input signal drop. Both reservoir fading memory, due to the recurrent network structure (feedback loops), and internal memory, due to the memristive junctions, contribute to the network memory properties [43]. The following analysis of switch dynamics is based on this 120 s simulation.

As shown in Fig. 3.6(b), conductance continues to increase under a DC bias. This increase is not smooth but stepped as individual junctions switch on. When the conductance reaches a maximum, it appears to be constant, but actually continues to exhibit fluctuations. These fluctuations are evident in the zoom-in shown in Fig. 3.7 for the 1 s period during $59 - 60$ s. The network and junction behavior during the 120 s simulation in Fig. 3.6 and the fluctuations in Fig. 3.7 are explained in the following paragraphs.




(a)




(b)

FIGURE 3.6: Nanowire network simulation geometry and activation. (a) Snapshot visualization of nanowire network at t = 0.01 s after a square pulse is delivered to a source node (green star) with voltage 0.7 V (red star denotes drain node), showing the distribution of nanowires (lines) and junctions (squares). Nanowire color denotes absolute voltage and junction color denotes voltage drop from low (blue) to high (yellow). White lines indicate the current direction (arrow) and value (length). (b) Voltage bias as a function of time (orange) and the resulting network conductance (black) between source and drain. The voltage bias after the square pulse is 0.015 V.

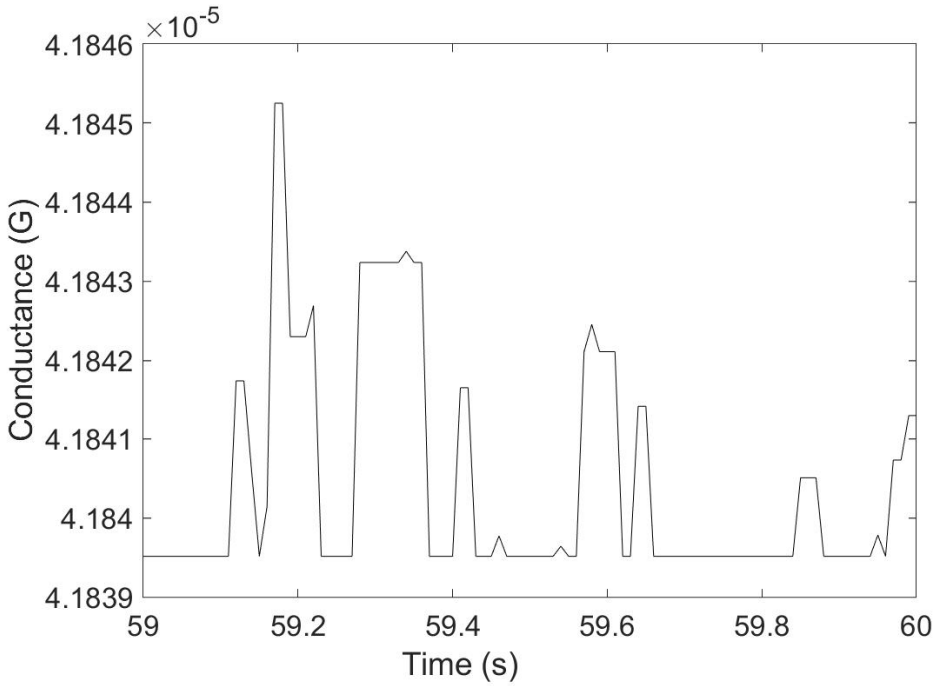



FIGURE 3.7: Zoom–in of network conductance in Fig. 3.6(b) in the interval 59 − 60 s, revealing fluctuations.

The conductance time series reflects the voltage redistribution dynamics in the network, which depends on the local circuitry (i.e. series vs. parallel). When only one pair of source-sink electrodes are applied, as in Fig. 3.6, there are more parallel junctions around the electrodes. This area has lower conductance. Thus, junctions at the edge of the network where there are fewer loops and close to the source (drain), have higher (lower) voltages, while junctions near the center of the network receive relatively lower absolute voltage. As is evident in the network snapshot visualizations in Fig. 3.6, junctions in the peripheral regions of the network closest to the electrodes have higher voltage (appear more yellow) than junctions closer to the center of the network (appear mostly blue).

Fig. 3.8 shows the network and junction states at two neighboring time points from the simulation in Fig. 3.6 . The white circle highlights a memristive junction (number 244) that switches from off to on, with its voltage drop decreasing accordingly by several orders of magnitude to a value lower than $V_{\text{reset}}$.

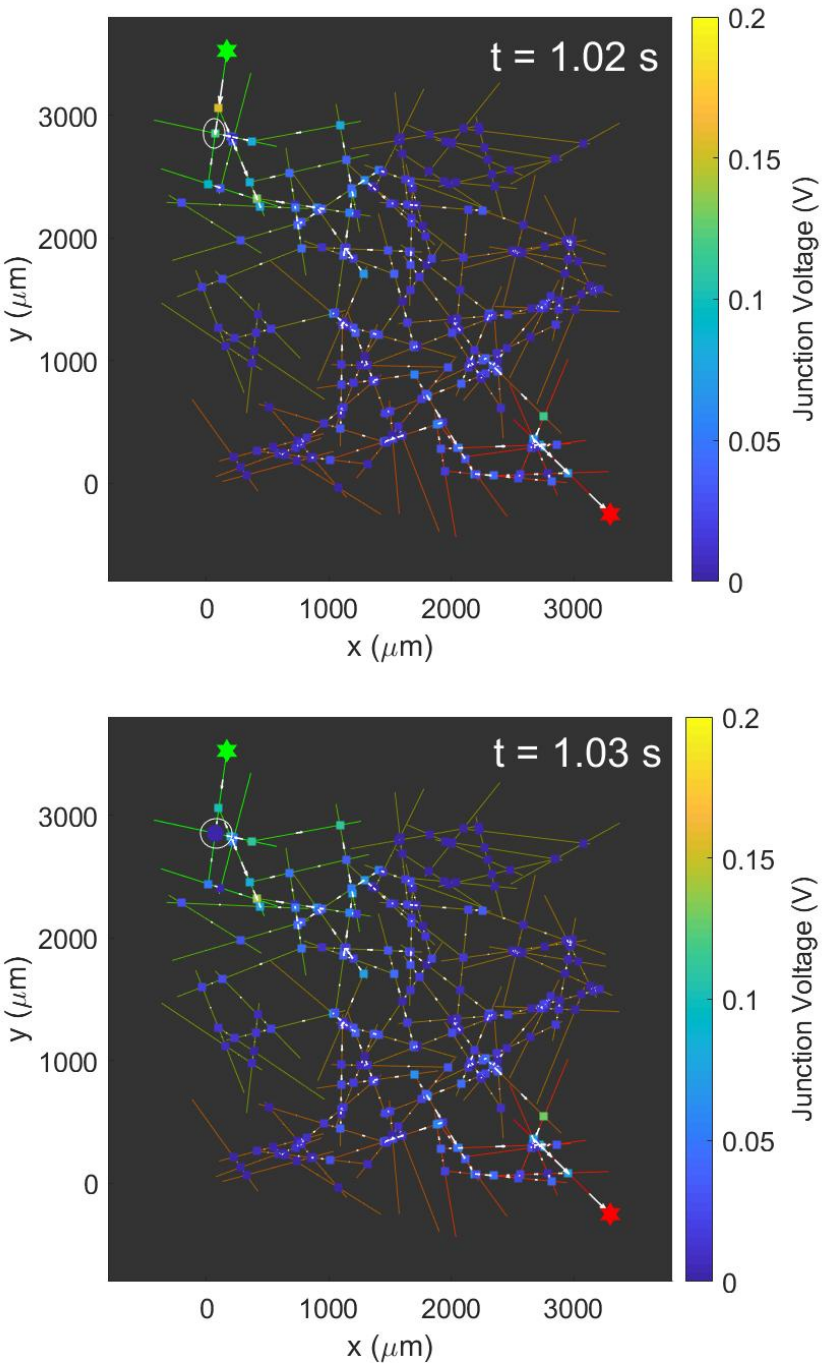
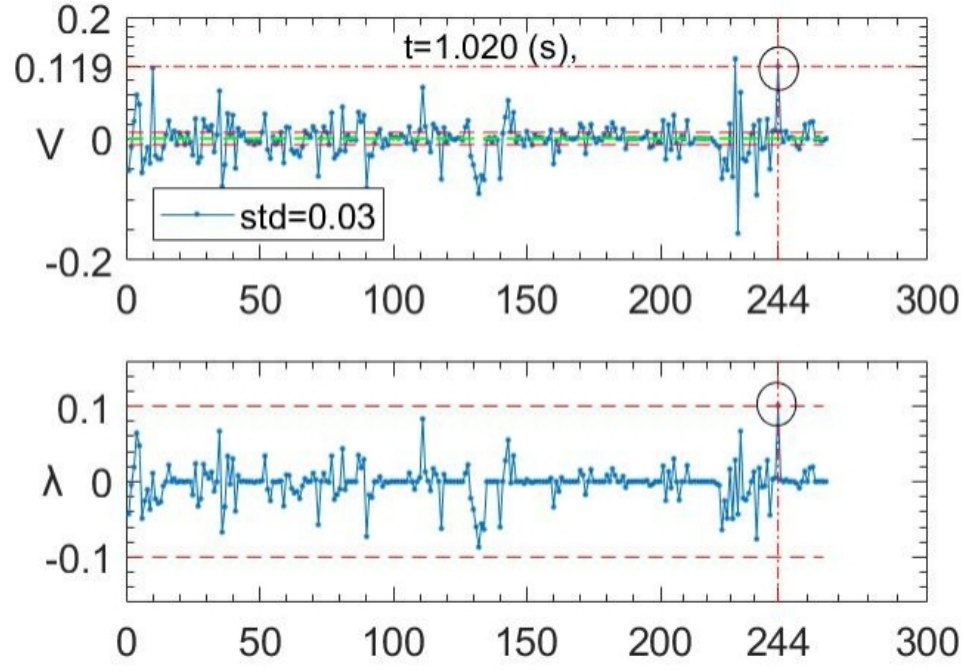



FIGURE 3.8: Network and junction states at two neighbouring time points as shown in Fig. 3.6: $t = 1.02\,\mathrm{s}$ (top); and $t = 1.03\,\mathrm{s}$ (bottom). The colourbar shows voltage absolute values. The white circle marks a junction (number 244) whose voltage decreases significantly once it turns on.

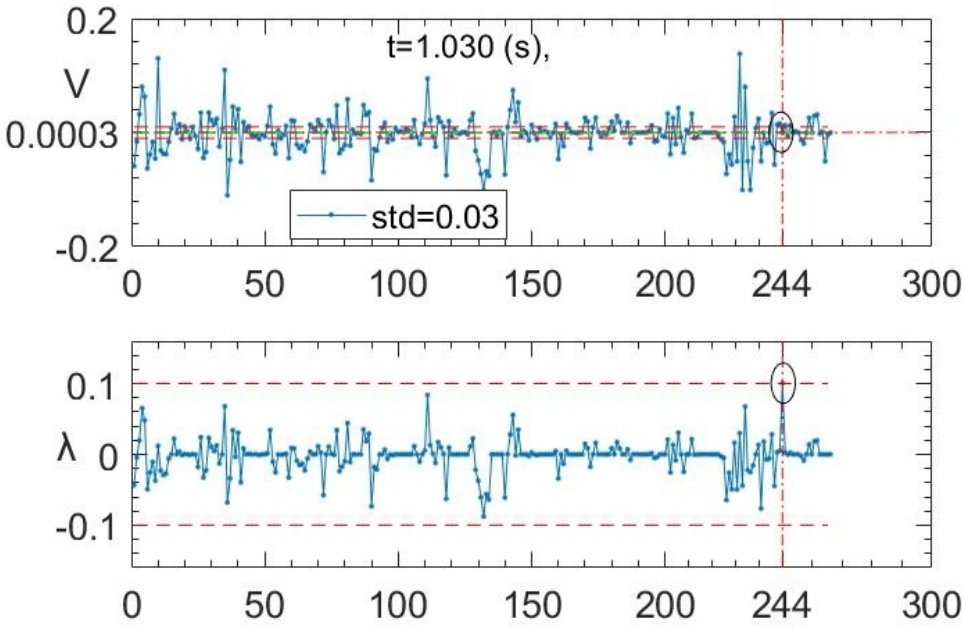
Fig. 3.9 plots $V$ and $\lambda$ for all the junctions at the corresponding two time-points in Fig. 3.8(a) and (b), highlighting junction 244. $V$ changes significantly with the on/off switching of this junction. When off, $V = 0.119\,\mathrm{V}$, which exceeds $V_{\mathrm{set}} = 10^{-2}\,\mathrm{V}$. $V$ drops dramatically (below $V_{\mathrm{reset}} = 10^{-3}\,\mathrm{V}$) when the junction switches in response to the conductance increase. Subsequently, this junction turns off again and regains a high voltage, switching on again once $|V(t)| > V_{\mathrm{reset}}$ and $\lambda > \lambda_{\mathrm{crit}}$ (cf. eqn. (3.1)). Thus, voltage redistribution and switch dynamics are inextricably linked via the network's complex circuitry.

Under a DC bias (Fig. 3.6), the junctions in a series circuit between the two electrodes fluctuate between on and off until the appearance of a current path. Fig. 3.10 shows the moment a current path forms in the simulation of Fig. 3.6. At $t = 9.87\,\mathrm{s}$, the off junctions just before the current path forms have the highest voltage, and once the current path forms, the voltage rapidly redistributes. The voltage redistributes from the off-junctions in the current path to other junctions across the whole network. This process is accompanied by a decrease in the standard deviation (std) in voltage, as shown in Fig. 3.11. Once the current path is formed, all junctions in it are in the on state and the input voltage is divided equally. Since $V_{\mathrm{reset}}$ is much smaller than $V_{\mathrm{set}}$, this current path can be maintained even when the input is small. Fig. 3.6 shows this process: the multiple states of conductance represent multiple current paths; conductance is still high under a low input when multiple current paths exist.

The voltage changes at the single junction level lead directly to the fluctuations observed in network conductance (cf. Fig. 3.7). Notice that each time the junction turns off, the filament state immediately returns to its initial state ($\lambda$ drops by a factor of 10 in the model). The just-off junctions require a period of time to evolve and turn on again, in the process creating sustained fluctuations. Fig. 3.12 shows $V$ and $G$ fluctuations in the individual junction 244 in our network over the course of the entire simulation of Fig. 3.6. Avizienis et al. [64] observed similar DC-driven fluctuations in current in their experimental Ag-$Ag_2S$ ASN device, but in such a device, it is impossible to measure what is happening at the single junction level.




(a)




(b)

FIGURE 3.9: (a) Voltage (top) and filament state $\lambda$ (bottom) of each junction at $t = 1.02$ s, corresponding to the snapshot in Fig. 3.8 (top) and the time-series in Fig.3.6(b). Voltage of junction number 244 is circled and marked by the red horizontal line at $V = 0.119$ V. In the $\lambda$ plot, the horizontal red dashed lines mark $\lambda_{\text{crit}}$. Junction 244 turns on at this time, when $\lambda = \lambda_{\text{crit}}$. (b) Same as (a), except for $t = 1.03$ s, corresponding to the snapshot in Fig. 3.8 (bottom)

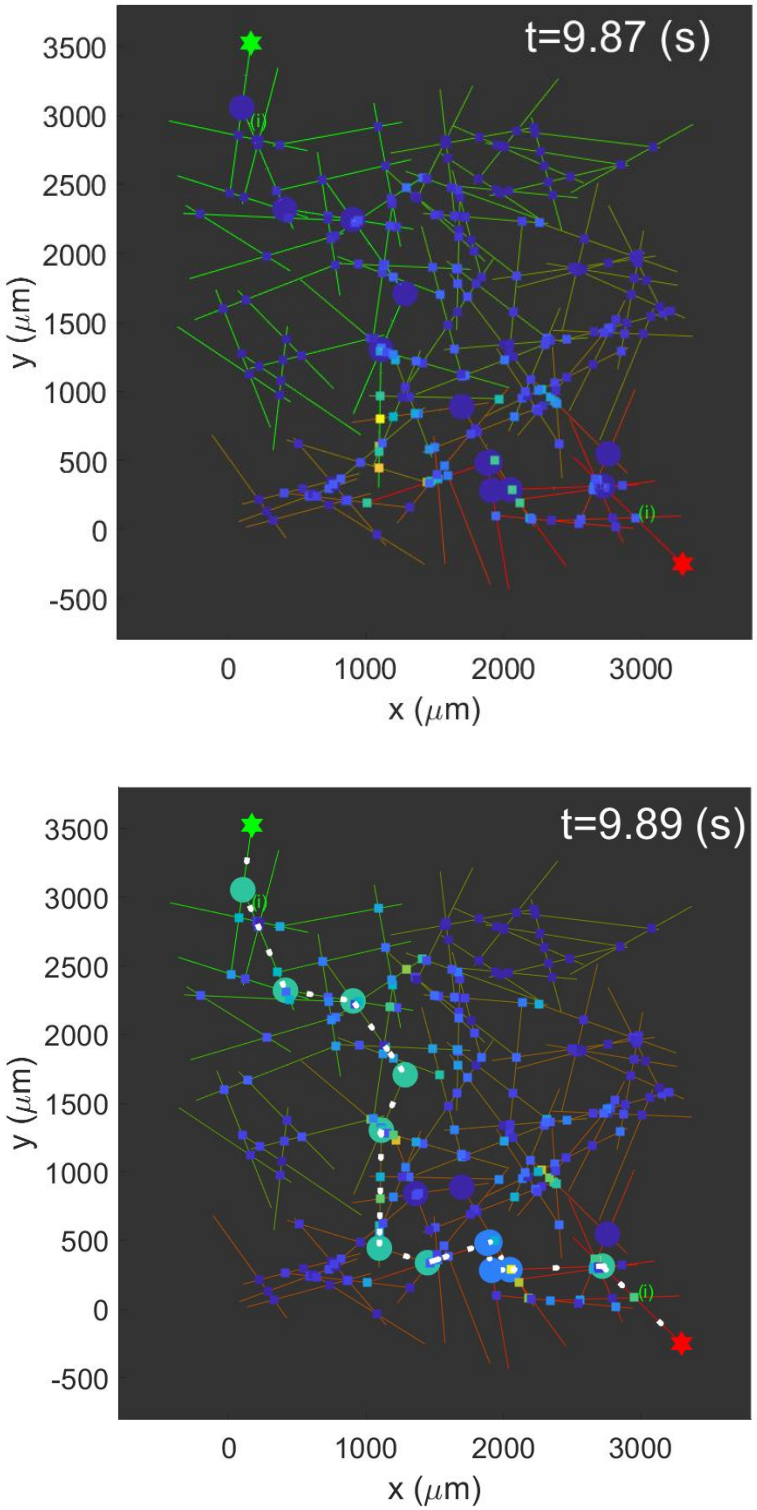
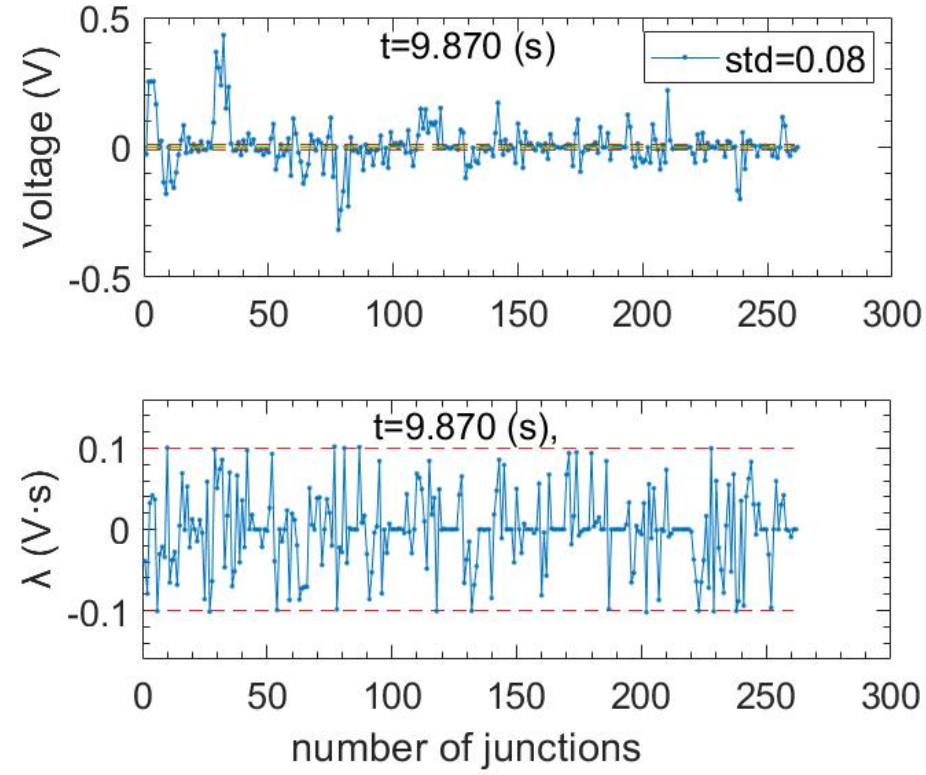
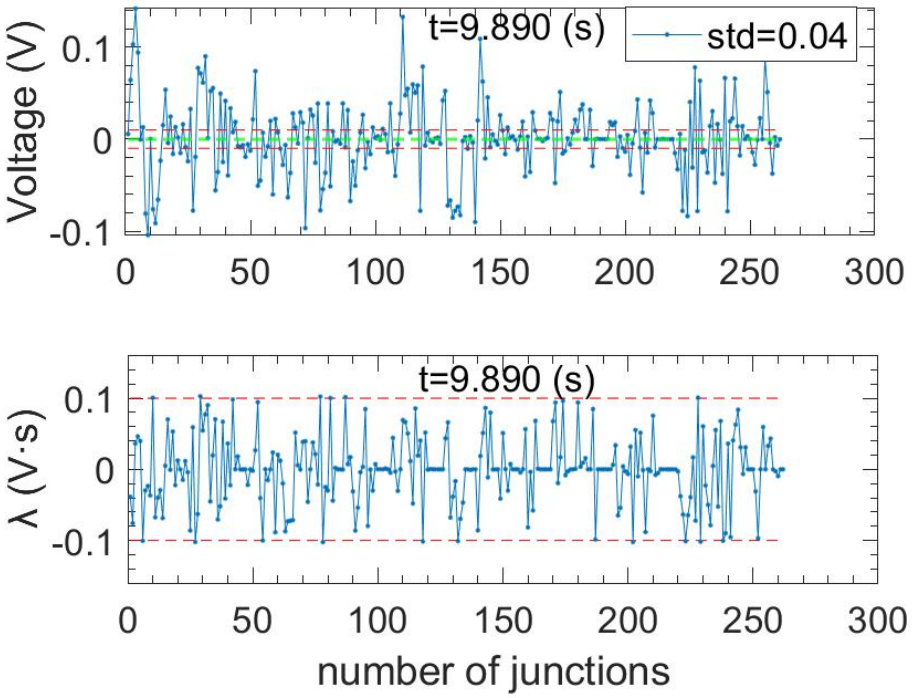



FIGURE 3.10: Network and junction states in the simluation of Fig. 3.6 at two neighbouring time points : $t = 9.87\,s$ (top) and $t = 9.89\,s$ (bottom), marking the onset of current path formation. Large circles denote junctions in a high conductance state; colour indicates wire voltage. Junctions change significantly, indicating rapid voltage redistribution.




(a)




(b)

FIGURE 3.11: Junction voltage redistribution and filament state changes from (a) 9.87 s to (b) 9.89 s corresponding to Fig.3.10. Also shown is the standard deviation (std) in voltage.

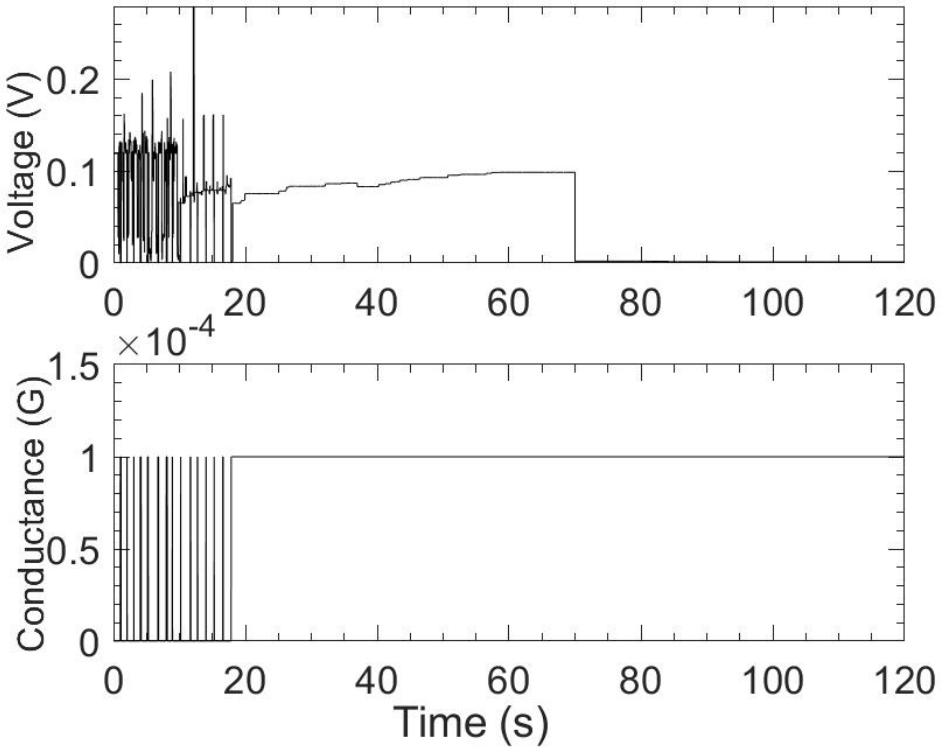



FIGURE 3.12: Fluctuations in voltage (top) and conductance (bottom) of an individual junction (244) in a network over the course of the entire simulation of Fig. 3.6.

Under a square pulse (Fig. 3.6(b)), followed by a residual DC bias of 0.015 V, the network conductance exhibits fading memory immediately after the pulse ends, but still maintains a relatively high value, which suggests capacity for long–term memory. The snapshot and the junction states are shown in Fig. 3.13. The amplitude of residual DC bias determines the magnitude of residual conductance. While short–term fading memory can be attributed to feedback loops (i.e. delay lines) in the recurrent network structure, longer-term memory results from the change in filament state from $\lambda_{\max}$ to $\lambda_{\text{crit}}$. The rate at which this occurs depends on the characteristic dynamical time–the scale of the input signal and the voltage distribution dynamics in the post-activation stage. While the fading memory property is essential for reservoir computing applications [51], additional internal resistive memory enables the network to retain information from more slowly varying input signals, and thus provides better timescale separation that may enable the reservoir to perform more diverse information processing tasks [43].

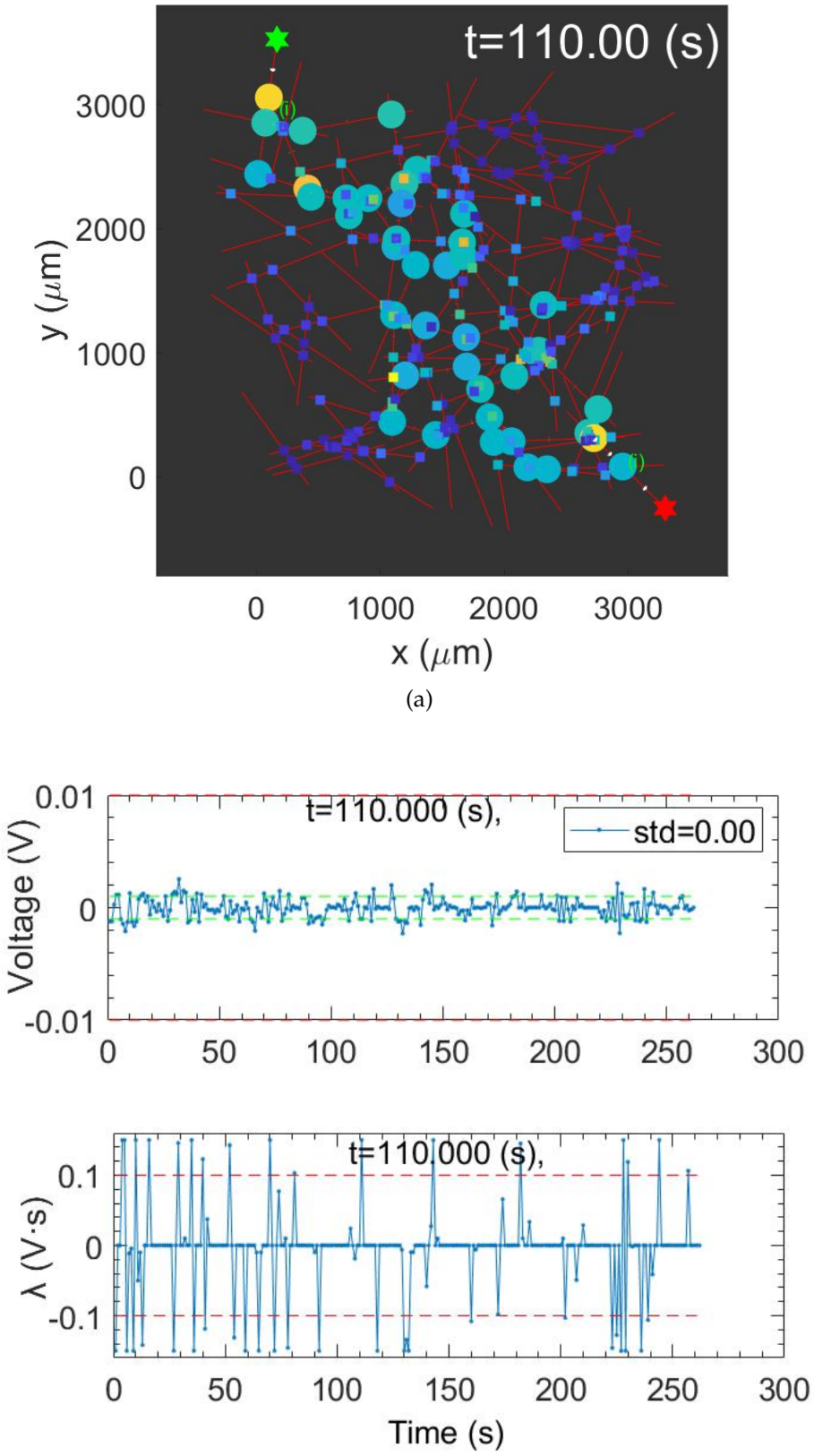
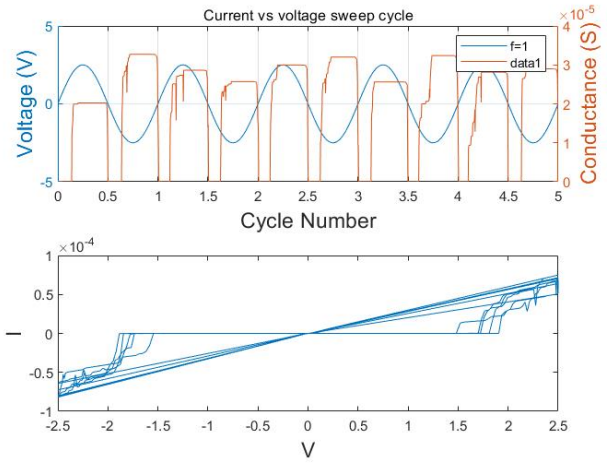


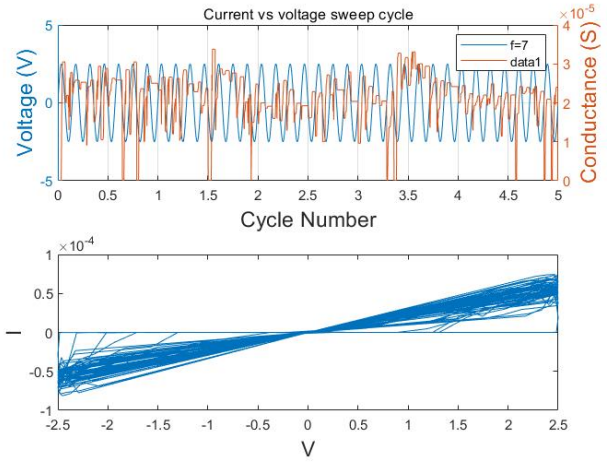

FIGURE 3.13: Network snapshot (a) and junction states (b) at $t = 110$ s in the simulation of Fig. 3.6, where the input is 0.015 V.

In the final set of results, we shift from the DC bias case shown in Fig. 3.6 to investigating the dynamics under AC stimulation. As introduced in 2.2.2, there are two main fingerprints of memristive systems: the pinched hysteresis loop that passes through the origin of an $I-V$ response curve; and the area of that loop being inversely proportional to frequency. These qualities also exist in memristive nanowire networks. In addition to junction fluctuations, the response to an AC signal can be chaotic, as is evident from the $I-V$ curves shown in Fig. 3.14. In this figure, conductance varies unpredictably from cycle to cycle, forming the characteristic chaotic–attractor $I-V$ curves in (a) and (b). In Fig. 3.14(c), however, this is not the case. Here, the input sine wave frequency is much higher (20 Hz), and conductance maintains a stable periodic behavior, although the period differs from the input. The trajectories are consistent with ordered attractor dynamics. This is an interesting result because it implies that there may exist a dynamical regime somewhere in between chaotic and ordered states that the system could be tuned into for optimal information processing, as has been proposed for echo state networks [80].

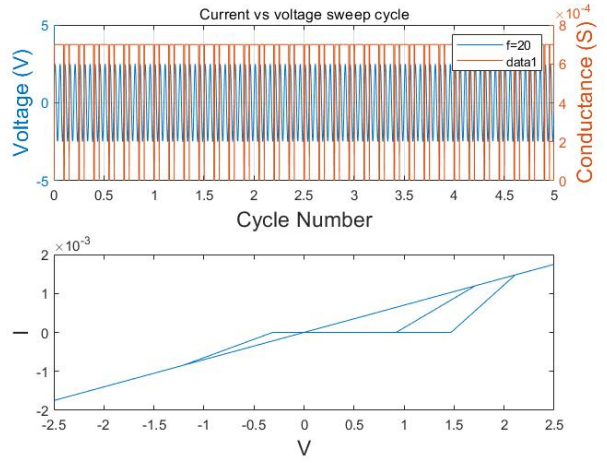
The significantly smaller area of the $I-V$ loop and more coincident trajectories in Fig. 3.14(c), compared to (a) and (b), also indicates that with two electrodes, the network behaves like one single memristor. The result shows the stable state and dramatic increase of conductance which are also shown in other studies Fig. 2.12 in chapter two. This corresponds to the formation of the current path under specific frequency.




(a)




(b)




(c)

FIGURE 3.14: Network dynamics under AC (sine wave) stimulation at varying frequencies: (a) 1 Hz; (b) 7 Hz; (c) 20 Hz. In each plot, the top panel shows input voltage (blue) and network conductance (red), while the bottom panel shows corresponding $I - V$ curves with characteristic pinched hysteresis loop.

### 3.3.2 Application to Reservoir Computing

#### Nonlinear transformation

Nonlinear transformation tasks were performed with our network as a reservoir using a 0.1 Hz sine wave input signal delivered to one nanowire allocated as the source node, with another allocated as the drain. Simulations were performed for networks of two different sizes: 100 and 700 nanowires. Fig. 3.15 confirms the ability of the nanowire networks to generate higher harmonics and thus its potential to perform the waveform regression task.

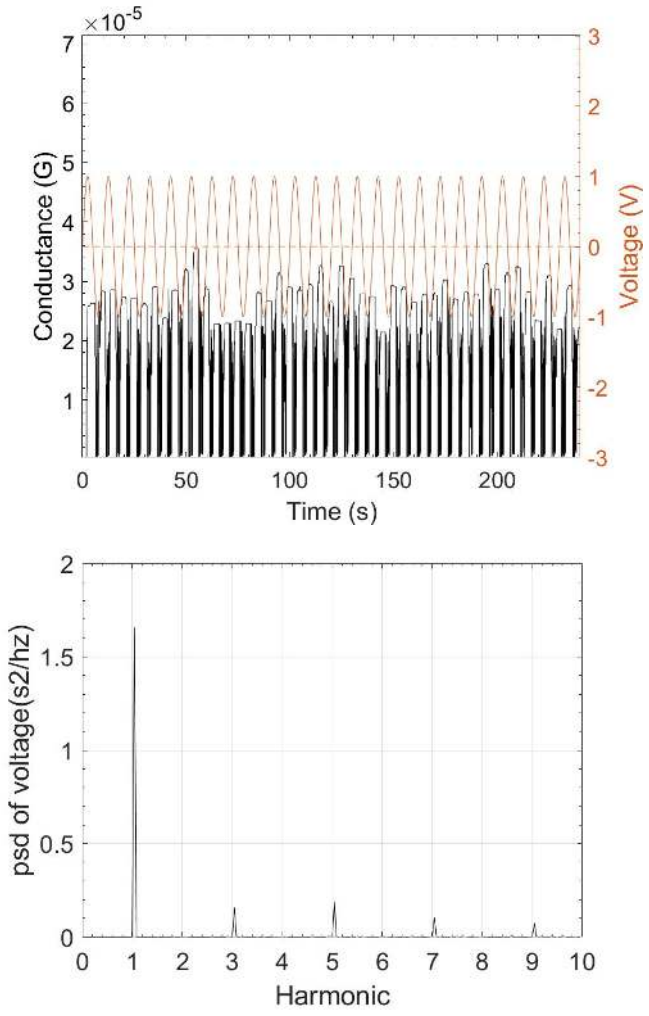



FIGURE 3.15: Sine wave input and higher harmonics generation in a 100-node network. Top panel: Sine wave (0.1 Hz) voltage input used for nonlinear transformation tasks and corresponding network conductance; Bottom panel: Power Spectral Density (PSD) of one nanowire node, showing odd harmonics at 3,5,7,9.

In these simulations, the target waveforms are: sawtooth, square, cosine and doubled–frequency sine wave. Fig.3.16 and Fig.3.17 show the linear regression results for 100-node and 700-node networks, respectively. For the 700-node case, 100 readout nodes were used for better comparison against the 100-node results. Fig. 3.18 shows the results for the 700-node network using piecewise linear regression with 4 time intervals. Table3.1 lists the accuracy results for both network sizes and for single vs. piecewise linear regression with 4 segments. The 700-node network, which is a larger reservoir with more degrees of freedom, performs consistently better than the 100-node network for all transformation tasks. For each network, piecewise linear regression improves accuracy significantly in all cases, most significantly for the $2f$ sine and cosine transformations, which show the lowest accuracies for single-segment regression.

TABLE 3.1: Accuracy of nonlinear transformation tasks for networks of different sizes (100 and 700 nodes) and for single (1) vs. piecewise linear (4) linear regression.

| Accuracy | 100-node (1) | 700-node (1) | 100-node (4) | 700-node (4) |
|---|---|---|---|---|
| Square | 58.1 % | 64.3 % | 92.0 % | 92.0 % |
| Sawtooth | 39.2 % | 44.9 % | 88.5 % | 89.3 % |
| $2f$-sine | 2.4 % | 31.6 % | 71.3 % | 93.7 % |
| Cosine | 2.6 % | 16.9 % | 88.2 % | 97.2 % |

Using the $2f$ sine wave transformation as an example, Fig.3.19 plots the accuracy achieved for this task as a function of the regression time segmentation. For both the 100-node and 700-node networks, the accuracy increases most rapidly up to 5-times regression. Saturation is reached by the 700-node network (i.e. improvement is marginal) beyond this number of temporal regression segments. The noticeable decrease in accuracy for the 100-node network (Fig. 3.19 (a)) at 10-times regression maybe due to how the time series of the input signal (sine wave) is segmented around the monotonically increasing/decreasing intervals. A similar effect was also observed for the 700-node network for other targets. It may be possible to improve piecewise linear regression using time segments of variable length, to better resolve parts of the waveform that change most rapidly.

These results corroborate previous experimental studies by Demis *et al.*

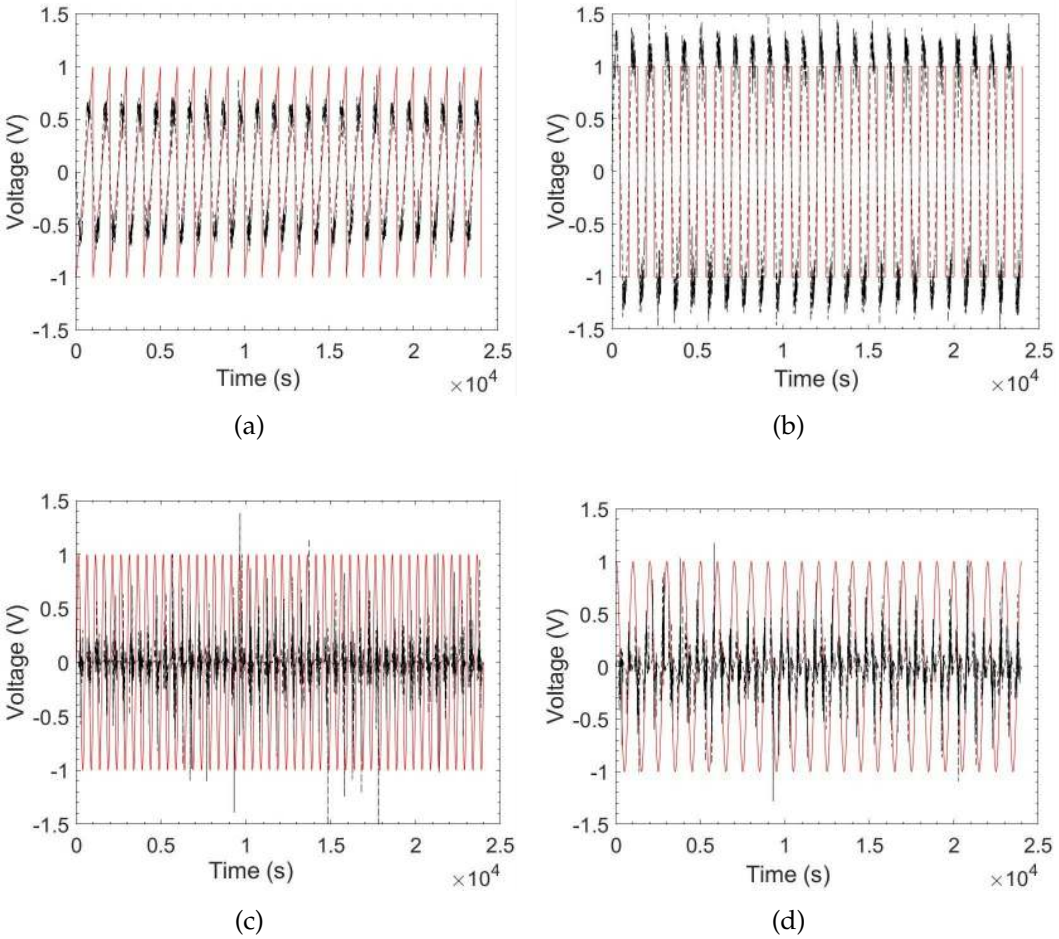



FIGURE 3.16: Nonlinear transformation of an input sine wave by a 100-nanowire network with 100 readout nodes (target in black, result in red): top panel – sawtooth target, accuracy = 39%; middle panel – square wave target, accuracy = 58.1%; bottom panel – doubled–frequency sine wave target, accuracy = 2.4%, and the cosine wave targer, accuracy = 2.6%

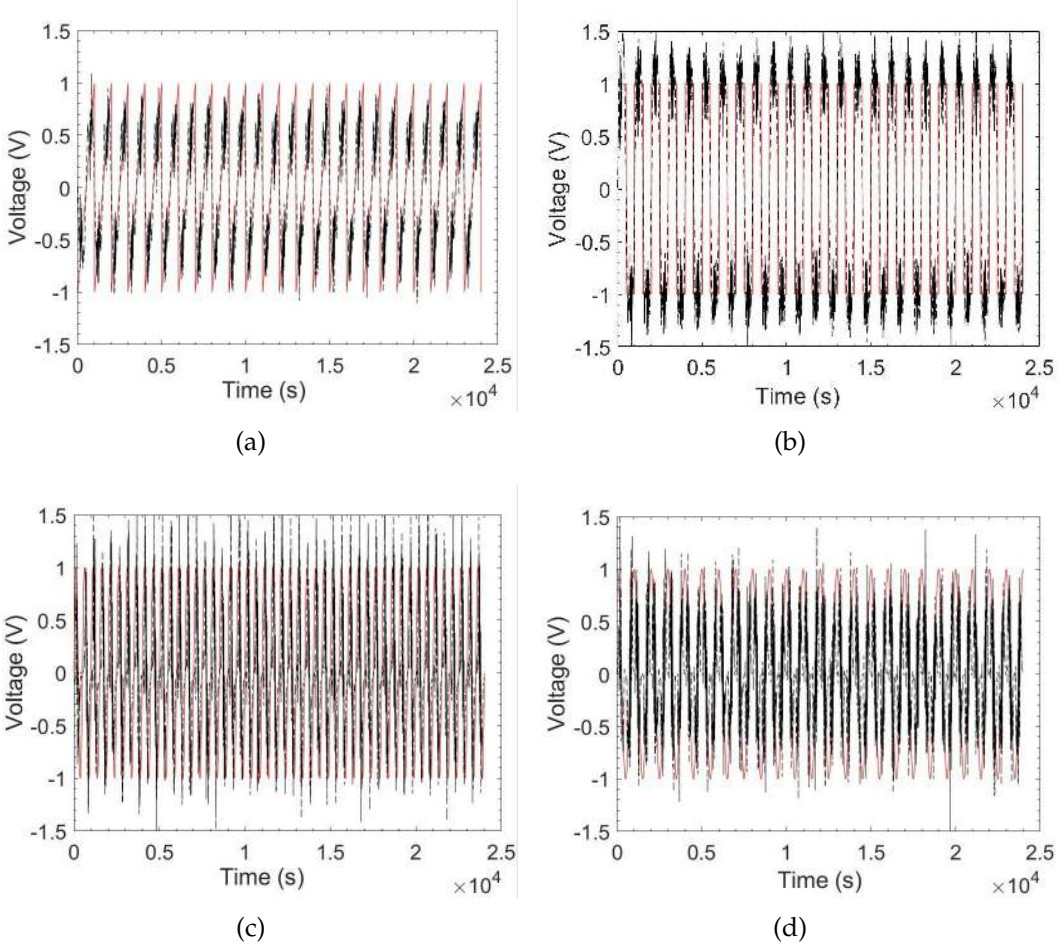



FIGURE 3.17: Nonlinear transformation of an input sine wave by a 700-nanowire network with 100 readout nodes (target in blue, result in red): top panel – sawtooth target, accuracy = 44.9%; middle panel – square wave target, accuracy = 64.3%; bottom panel – doubled–frequency sine wave target, accuracy = 31.6%, and the cosine wave targer, accuracy = 16.9%

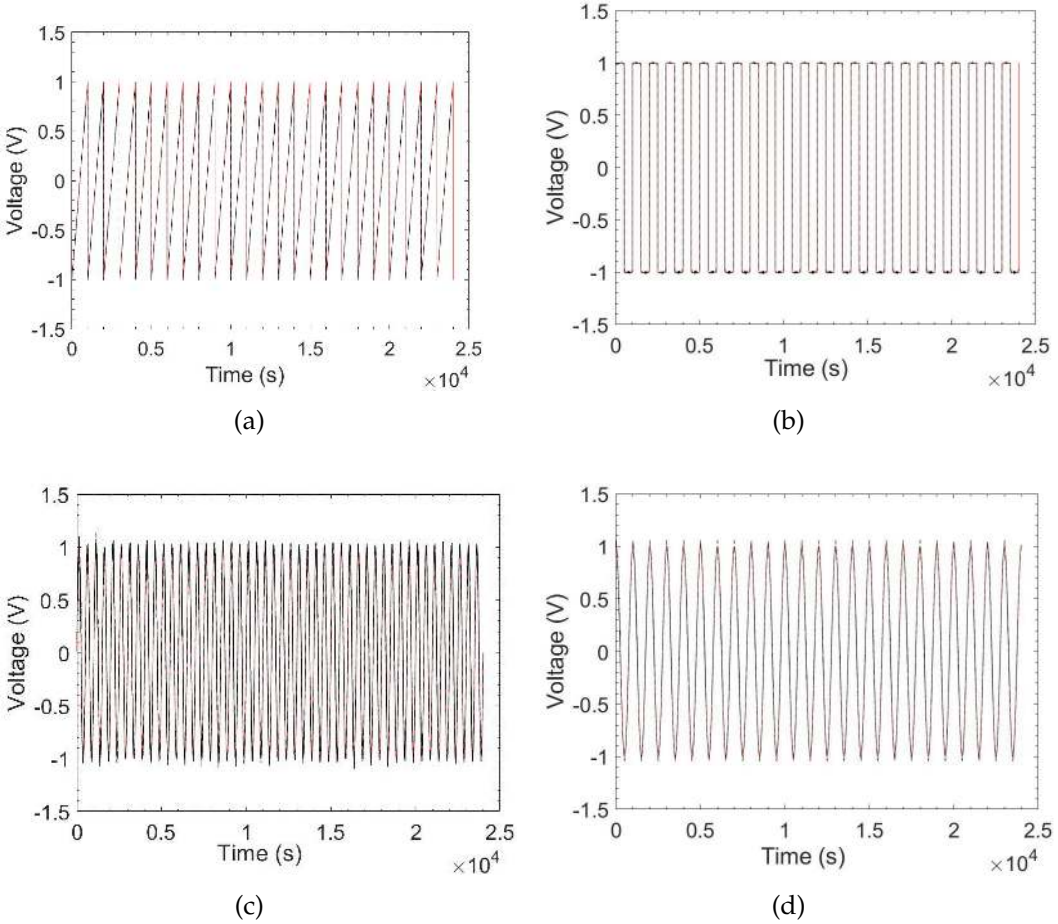
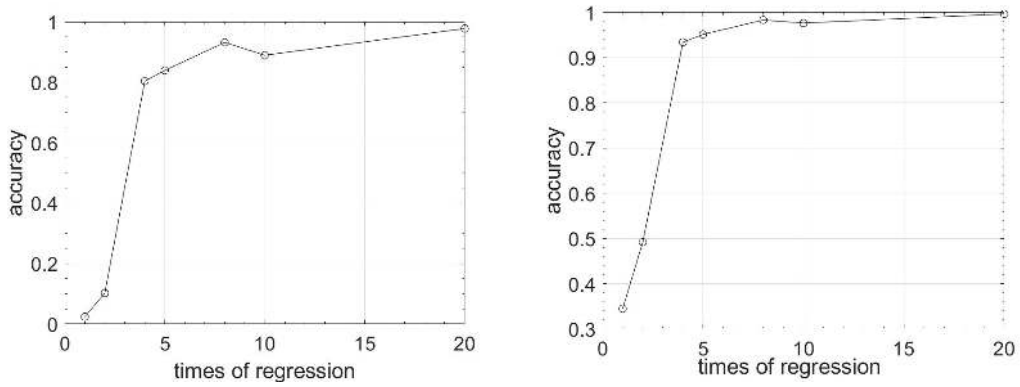



FIGURE 3.18: Nonlinear transformation of an input sine wave by a 700-node nanowire network using piecewise linear regression (target in red, result in black): (a) sawtooth target with 4-times regression, accuracy=89.3%; (b) square wave target with 2-times regression, accuracy=92%; (c) doubled-frequency sine wave target with 4-time regression, accuracy=93.7%; (d) cosine wave target with 4-times regression, accuracy=97.2%.

[67] using their Ag-$Ag_2S$-Ag physical nanowire reservoir for nonlinear transformation tasks as shown in Fig.2.17. They applied one input electrode and read out the voltage of other measurement electrodes in contact with their network (the size of which is difficult to determine experimentally). The accuracy of their experimental measurement is $< 90\,\%$ (using standard single regression readout). Here, piecewise regression improves the accuracy remarkably and thus could be an effective way to process periodic signals in RC.




(a) (b)

FIGURE 3.19: Accuracy of nonlinear transformation as a function of linear regression segmentation for the doubled–frequency sine wave task using: (a) a 100-node network; and (b) a 700-node network.

### Wave auto-generation

In this task, the sine wave used previously as the input signal is now generated by the network without any external input after a training period. The result is shown in Fig.3.20 for a 700-node network. The MSE is negligible, giving an accuracy of $\approx 100\,\%$. Notice that the length of the generated sine wave (120 s) is longer than the training period (90 s) and indeed, the period of auto-generation can be prolonged further for hundreds of seconds. However, the error increases rapidly as the iteration time increases, leading to divergence of the final result. Therefore, predictions close to 100% accuracy can only be maintained for a finite period of time, although they will be longer than predicted time steps of nonlinear signals. In the particular case of the Mackey glass signal, which is highly nonlinear, the MSE quickly diverges (exponentially in the chaotic regime) [1, 76, 63].

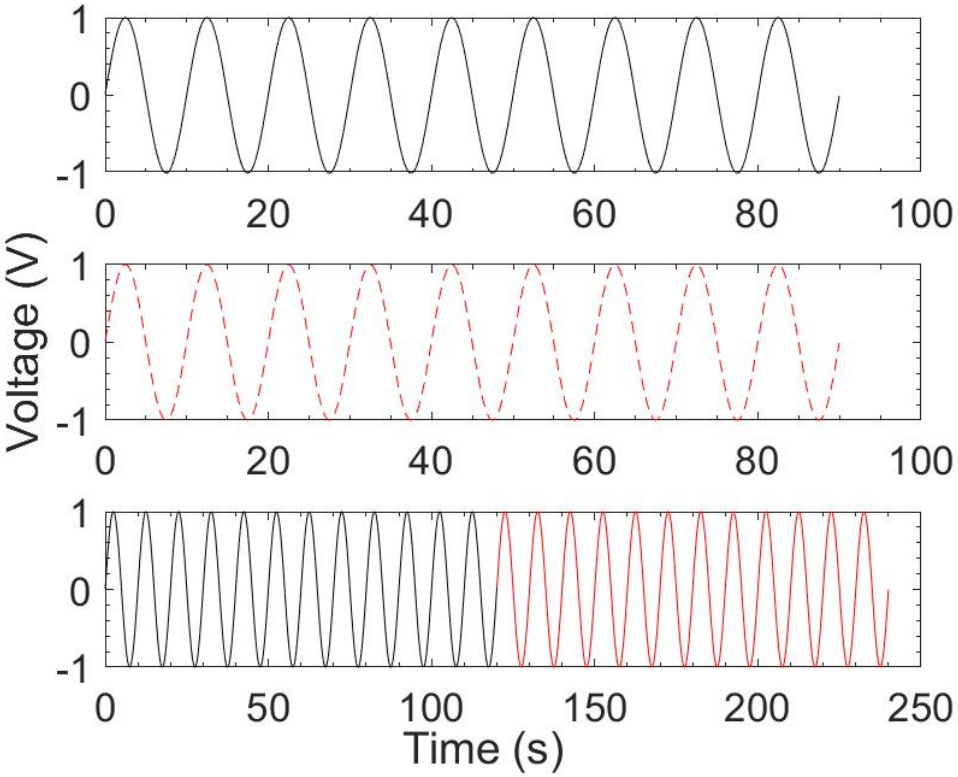



FIGURE 3.20: Sine wave generation by a 700-node network. Top panel – target sine wave signal (teacher input). Middle panel – training by linear regression. Bottom panel – pre-activation period (30 s) and training period (90 s, black) followed by generating period (120s, red). Accuracy is $\approx$ 100%.

**Handwritten digit classification**

Fig.3.21 shows the confusion matrix for the MNIST handwritten digit classification task for a 700-node network. The average fraction of true positives is TP = 91.6% and coefficient of determination is $R_2 = 0.835$. The overall MSE-based recognition accuracy achieved is 91.2%, and 90.4% for a 1000-node network, but decreases to 87.5% for 100 nodes.

These results depend on additional parameters introduced for the specific reservoir computing implementation adopted here. First, is the input voltage of each pixel, which regulates memristive switching via the growth of the conducting filament. If the input voltage is too small, it is difficult to form current paths in the network. On the other hand, if the voltage is too large, different input streams can cause indistinguishable high local conductance states, thus effectively washing out the feature space. Thus the amplitude $V$ needs to be tuned to an optimal value. We found 1 V works well, but this in turn depends on the timescale used for streaming the digits in our temporal implementation; here we presented each digit for 2.8 s (i.e. 0.1 s per pixel, with one row of 28 pixels streamed into a dedicated input electrode–node). The distribution of the 28 electrode nodes in the network was also considered. If the distribution of electrodes is too dense, it can affect the ability of the local network to distinguish different input streams. We considered two configurations: a uniform distribution and a random distribution. Both were found to give roughly the same results, with the uniform distribution slightly better. We choose a uniform distribution of electrode nodes since it is more realistically compatible with a CMOS multi-electrode device. While an even higher accuracy might be attained by streaming each pixel into a large network, this would require $28 \times 28 = 784$ electrode–nodes, which is not realistically feasible in hardware.

The results presented here were obtained using LDA classification on the readout layer. Random forest and other machine learning methods were also investigated, but they are more complex than LDA and prone to overfitting. They were also found to take longer to reach the same accuracy on test data (93%). For example, the accuracy of random forest can reach 99% in training and 92% in testing using a 700-node network. This accuracy on the test data is close to LDA, but the training time is considerably longer ( 21 times for 700-nodes network). Considering the overfitting and efficiency problems, LDA was found to be overall superior to these methods. Other models like

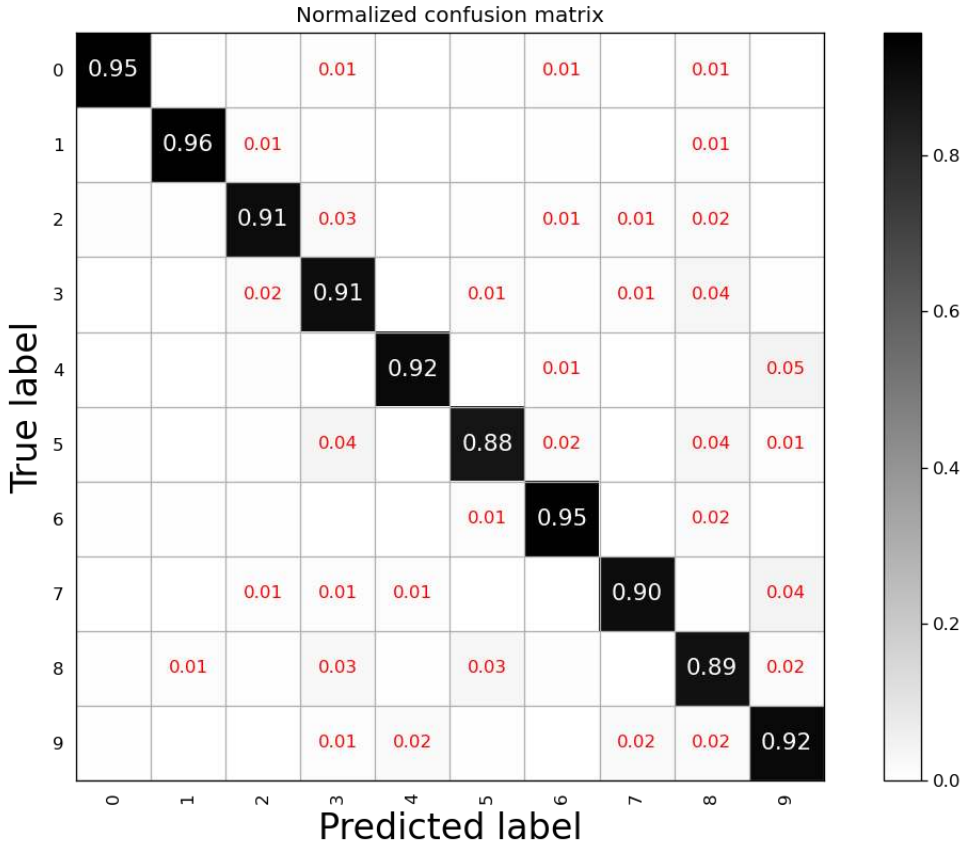



FIGURE 3.21: Confusion matrix for MNIST handwritten digit classification performed using a reservoir computing implementation for a 700-node network.

SVM and Multilayer Perceptrons were not compared as they perform this task directly without a reservoir, which is used here to linearly separate the features.

The MNIST classification accuracy obtained here compares favorably to results obtained from a similar experimental study by Du et al. [57] using memristors arranged in parallel as the reservoir, where recognition accuracies of 91.1% and 91.5% were achieved in simulation using 88 and 112 memristors, respectively, and 88.1% was achieved in experiment using 88 memristors. Their experimental system differs from our memristive nanowire system. First, they used $WO_x$ bulk memristors which exhibit multiple conductance states, while our memristive junctions only exhibit two states. Second, they pre-processed the input voltage stream by delivering it into a separate memristor, using the single memristor's non-linearity and memory to create features from the input stream. Here, the input voltage streams were directly

delivered into multiple nodes of a memristive reservoir to create a higher dimensional feature space with linearly separable outputs. Another important difference is that in addition to the long-term memory of memristive junctions, the recurrent structure of our network gives it short-term (i.e. fading) memory as well. Thus, our network has the capacity to separate features with different dynamical timescales. For the read-out layer, logistic regression, a generalised linear model, was used by Du et al. [57], whereas LDA was used in this work.

Other studies have also used a memristor as a non-linear read-out layer [81]. In Lilak et al. [81], 2-state (on/off) memristor nano-synapses were used to record $W_{\text{in}}$ and $W_{\text{out}}$, with ridge regression used on the readout layer. Their result depends on the size and implementation method of the reservoir which is based on a CMOS cross-bar array . The best accuracy achieved on the MNIST task in that study is 95%. Chen et al. [82] used another memristive filter with multiple electrodes. Each filter had 8 electrodes with non-linearity and short-term memory. This filter was able to handle multiple pixels of MNNIST digit images. They used $27 \times 27 \times 16 = 11,664$ filters and obtained an accuracy of 96%. The accuracy is only marginally better than that obtained here while using many more memristive devices.

Overall, the MNIST simulation results presented here demonstrate proof of principle of the RC approach using memristive nanowire networks. This warrants a follow-up study to implement this image classification task in hardware, which has now commenced using the nanowire network CMOS device developed at UCLA [83].

In general, results obtained using the RC approach are not as precise or reproducible as those obtained using ANNs. Different statistical realisations of a reservoir network produce slightly different results. However, results are statistically robust (i.e. statistical variations are bounded). In our simulations, results are averaged over several network realisations. The generalisability of the RC approach is worth exploring further using different types of data, beyond that devised for machine learning tasks. The results of this thesis demonstrate the potential of nanowire networks for neuromorphic computing and will provide a valuable contribution to devising new ways to benchmark neuromorphic systems.

# Chapter 4

# Conclusions and outlook

The results presented in this thesis show that the neuromorphic dynamics of self-assembled nanowire networks are useful for information processing using a reservoir computing approach.

Our analysis of the network dynamics via junction-level modelling and simulation revealed unique insights that could not otherwise be gained from experimental measurements using hardware devices. In particular, our simulations revealed how single junction memristive switching (due to formation/decay of a conductive filament) causes dynamical redistribution of voltage across the whole network. This redistribution is observed as fluctuations in the conductance time series and explains previously reported experimental results.

Simulations of the network response under AC showed that the $I - V$ curve exhibits the characteristic fingerprints of a memristor. From this we conclude that a nanowire network with memristive junctions can be treated as one single bulk memristor. This is a new result that may be leveraged in future experimental measurements.

The network's non-linearity was demonstrated via higher harmonic generation under AC and short/long term memory demonstrated by sustained high conductance following a square pulse. These results are important because they confirm that the network has the potential as a physical reservoir for reservoir computing.

Machine learning tasks were successfully implemented in the reservoir computing framework, namely nonlinear wave regression, sine wave autogeneration, and MNIST hand-written digit classification. Compared to other

studies that used an array of bulk memristors as a reservoir, our method using a memristive network reservoir is easier to implement in both simulation and physical hardware devices.

The results of this thesis open up new avenues for future research. The chaotic attractor dynamics found as a result of the redistribution of voltage across the network warrants further investigation to identify the transition from ordered to chaotic dynamics under different stimulation. Additionally, the idea that the whole network can be treated as a single memristor can be explored further using memristor theory. And more complex learning tasks, such as human action recognition, can be implemented using reservoir computing to further demonstrate the network's capacity for temporal signal processing and compare those result with traditional ANN. Finally, The influence of many features of hardware network on the results can be studied when we have mature hardware production process.

# Bibliography


[1] Kaiwei Fu et al. “Reservoir Computing with Neuromemristive Nanowire Networks”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207727.

[2] Suzana Herculano-Houzel. “The remarkable, yet not extraordinary, human brain as a scaled-up primate brain and its associated cost”. In: *Proceedings of the National Academy of Sciences* 109 (2012), pp. 10661–10668. DOI: 10.1073/pnas.1201895109.

[3] Javier DeFelipe. “Neuroanatomy and Global Neuroscience”. In: *Neuron* 95 (2017), pp. 14–18. ISSN: 0896-6273. DOI: https://doi.org/10.1016/j.neuron.2017.05.027.

[4] Antonio R. Damasio. *The Scientific American book of the brain*. New York: Lions Press, 1999.

[5] Jianshi Tang et al. “Bridging biological and artificial neural networks with emerging neuromorphic devices: fundamentals, progress, and challenges”. In: *Advanced Materials* 31 (2019), p. 1902761.

[6] Alberto E Pereda. “Electrical synapses and their functional interactions with chemical synapses”. In: *Nature Reviews Neuroscience* 15 (2014), pp. 250–263.

[7] H. Kim et al. “Memristor Bridge-Based Artificial Neural Weighting Circuit”. In: *Memristor Networks*. Ed. by A. Adamatzky and L. Chua. Springer, Cham, 2014, pp. 249–265. DOI: 10.1007/978-3-319-02630-5.

[8] Andy Thomas and Christian Kaltschmidt. “Bio-inspired neural networks”. In: *Memristor Networks*. Springer, 2014, pp. 151–172.

[9] Jack D Kendall and Suhas Kumar. “The building blocks of a brain-inspired computer”. In: *Applied Physics Reviews* 7 (2020), p. 011305.

[10] Adnan Mehonic et al. “Memristors—From In-Memory Computing, Deep Learning Acceleration, and Spiking Neural Networks to the Future of Neuromorphic and Bio-Inspired Computing”. In: *Advanced Intelligent Systems* 2 (2020), p. 2000085. DOI: https://doi.org/10.1002/aisy.202000085.

[11] Y. Xin. “Evolving artificial neural networks”. In: *Proceedings of the IEEE* 87 (1999), pp. 1423–1447.

[12] A. K. Jain, J. Mao, and K. M. Mohiuddin. “Artificial neural networks: a tutorial”. In: *Computer* 29 (2015), pp. 31–44.

[13] Andrej Krenker, Janez Bešter, and Andrej Kos. “Introduction to the artificial neural networks”. In: 2011, pp. 1–18.

[14] V. Clay et al. “Fast Concept Mapping: The Emergence of Human Abilities in Artificial Neural Networks when Learning Embodied and Self-Supervised”. In: *arXiv preprint arXiv:2102.02153* (2021).

[15] Hecht-Nielsen Robert et al. “Theory of the backpropagation neural network”. In: vol. 1. 1989, pp. 593–605.

[16] Andrej Krenker, Janez Bešter, and Andrej Kos. “Introduction to the artificial neural networks”. In: *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech* (2011), pp. 1–18.

[17] Stanisław Woźniak et al. “Deep learning incorporating biologically inspired neural dynamics and in-memory computing”. In: *Nature Machine Intelligence* 2 (2020), pp. 325–336.

[18] Wolfgang Maass. “Networks of spiking neurons: the third generation of neural network models”. In: *Neural Networks* 10 (1997), pp. 1659–1671.

[19] K. Saggie-Wexler, A. Keinan, and E. Ruppin. “Neural Processing of Counting in Evolved Spiking and McCulloch-Pitts Agents”. In: *Artificial Life* 12 (2014), pp. 1–16.

[20] J. R. Knott. “The organization of behavior: A neuropsychological theory”. In: *Electroencephalography and Clinical Neurophysiology* 3 (1951), pp. 119–120.

[21] Sander M. Bohte A, Joost N. Kok A C, and Han La Poutré a b. “Error-backpropagation in temporally encoded networks of spiking neurons”. In: *Neurocomputing* 48 (2002), pp. 17–37.

[22] A. K. Nski and F. Ponulak. "Comparison of supervised learning methods for spike time coding in spiking neural networks". In: *International Journal of Applied Mathematics and Computer Science* 16 (2006), pp. 101–113.

[23] G. Bellec et al. "A solution to the learning dilemma for recurrent networks of spiking neurons". In: *Nature Communications* 11 (2020), pp. 1–15.

[24] Mantas Lukoševičius and Herbert Jaeger. "Reservoir computing approaches to recurrent neural network training". In: *Computer Science Review* 3 (2009), pp. 127–149.

[25] V. Roby, C. Jeff, and S. N. Irene. "Diffusion-based neuromodulation can eliminate catastrophic forgetting in simple neural networks". In: *Plos One* 12 (2017), e0187736.

[26] Y. Hao et al. "Preface to the Special Issue on Beyond Moore: Resistive Switching Devices for Emerging Memory and Neuromorphic Computing". In: *Journal of Semiconductors* 42 (2021), 010101 (2pp).

[27] C. Mead. "How we created neuromorphic engineering". In: *Nature Electronics* 3 (2020), pp. 434–435.

[28] Wenqiang Zhang et al. "Neuro-inspired computing chips". In: *Nature electronics* 3 (2020), pp. 371–382. ISSN: 2520-1131.

[29] Fernando Corinto et al. "Cellular nonlinear networks with memristor synapses". In: (2014), pp. 267–291.

[30] G. Indiveri et al. "Neuromorphic Silicon Neuron Circuits". In: *Frontiers in Neuroscience* 5 (2011), p. 73.

[31] G. Indiveri. "Modeling Selective Attention Using a Neuromorphic Analog VLSI Device". In: *Neural Computation* 12 (2000), pp. 2857–2880.

[32] Gwendal Le Masson et al. "Feedback inhibition controls spike transfer in hybrid thalamic circuits". eng. In: *Nature (London)* 417 (2002), pp. 854–858. ISSN: 0028-0836.

[33] M. Bove, M. Giugliano, and M. Grattarola. "Regulatory effects of long-term biochemical processes in integrate-and-fire model neurons". In: *Nato Asi* 167 (1999), pp. 189–204.

[34] Wulfram. Gerstner. *Spiking neuron models : single neurons, populations, plasticity*. eng. Cambridge, U.K. ; Cambridge University Press, 2002. ISBN: 0521890799.

[35] Christoph Stöckl and Wolfgang Maass. "Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes". In: *Nature Machine Intelligence* 3 (2021), p. 230. DOI: https://doi.org/10.1038/s42256-021-00311-4.

[36] Tara Hamilton. "The best of both worlds". In: *Nature Machine Intelligence* 3 (2021), p. 194. DOI: https://doi.org/10.1038/s42256-021-00315-0.

[37] Leon Chua. "Memristor-the missing circuit element". In: *IEEE Transactions on circuit theory* 18 (1971), pp. 507–519.

[38] DB. Strukov and R. S. Williams. "Intrinsic constrains on thermally-assisted memristive switching". In: *Applied Physics A* 102 (2011), pp. 851–855.

[39] P. Sheridan and W. Lu. "Memristors and Memristive Devices for Neuromorphic Computing". In: *Springer International Publishing* (2014), pp. 129–149.

[40] Shyam Prasad Adhikari et al. "Three Fingerprints of Memristor". In: *IEEE Transactions on Circuits & Systems I Regular Papers* 60 (2013), pp. 3008–3021. DOI: 10.1109/TCSI.2013.2256171.

[41] J Joshua Yang et al. "The mechanism of electroforming of metal oxide memristive switches". eng. In: *Nanotechnology* 20 (2009), pp. 215201–215201. ISSN: 0957-4484.

[42] Vinod K Sangwan and Mark C Hersam. "Neuromorphic nanoelectronic materials". In: *Nature nanotechnology* 15 (2020), pp. 517–528. ISSN: 1748-3387.

[43] D. Verstraeten et al. "An experimental unification of reservoir computing methods". In: *Neural Networks* 20 (2007), pp. 391–403.

[44] W. Maass, T Natschläger, and H. Markram. "A Model for Real-Time Computation in Generic Neural Microcircuits". In: *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*. MIT; 1998, 2002, pp. 229–236.

[45] Joni Dambre et al. "Information Processing Capacity of Dynamical Systems". In: *Scientific Reports* 2 (2012), p. 514.

[46] Filippo Maria Bianchi, Lorenzo Livi, and Cesare Alippi. "Investigating echo state networks dynamics by means of recurrence analysis". In: *IEEE Transactions on Neural Networks & Learning Systems* 29 (2016), pp. 427–439.

[47] Thomas M. Cover. "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition". In: *IEEE Transactions on Electronic Computers* EC-14 (1965), pp. 326–334. DOI: 10.1109/PGEC.1965.264137.

[48] Herbert Jaeger. "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note". In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148 (2001), p. 13.

[49] Wolfgang Maass, Thomas Natschlger, and Henry Markram. "Real-time computing without stable states: A new framework for neural computation based on perturbations". In: *Neural Computation* 14 (2002), pp. 2531–2560.

[50] L.O Chua. "The Fourth Element". In: *Proceedings of the IEEE* 100 (2012), p.1920–1927.

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

[52] Miguel C Soriano et al. "Minimal approach to neuro-inspired information processing". In: *Frontiers in computational neuroscience* 9 (2015), p. 68.

[53] Kazuya Terabe et al. "Quantized conductance atomic switch". In: *Nature* 433 (2005), p. 47.

[54] Takeo Ohno et al. "Short-term plasticity and long-term potentiation mimicked in single inorganic synapses". In: *Nature Materials* 10 (2011), p. 591.

[55] Masakazu Aono. "Nanoionics". In: *Nature Materials* 6 (2007), pp. 833–840.

[56] Rainer Waser et al. "Redox-based resistive switching memories–nanoionic mechanisms, prospects, and challenges". In: *Advanced Materials* 21 (2009), pp. 2632–2663.

[57] Chao Du et al. "Reservoir computing using dynamic memristors for temporal information processing". In: *Nature communications* 8 (2017), p. 2204.

[58] John Moon et al. “Temporal data classification and forecasting using a memristor-based reservoir computing system”. In: *Nature Electronics* (2019), pp. 1–8.

[59] Christopher H Bennett, Damien Querlioz, and Jacques-Olivier Klein. “Spatio-temporal learning with arrays of analog nanosynapses”. In: IEEE. 2017, pp. 125–130.

[60] Alon Loeffler et al. “Topological properties of neuromorphic nanowire networks”. In: *Frontiers in Neuroscience* 14 (2020), p. 184.

[61] Zdenka Kuncic and Tomonobu Nakayama. “Neuromorphic nanowire networks: principles, progress and future prospects for neuro-inspired information processing”. In: *Advances in Physics: X* 6 (2021), p. 1894234. DOI: 10.1080/23746149.2021.1894234.

[62] Henry O Sillin et al. “A theoretical and experimental study of neuromorphic atomic switch networks for reservoir computing”. In: *Nanotechnology* 24 (2013), p. 384004.

[63] Ruomin Zhu et al. “Harnessing adaptive dynamics in neuro-memristive nanowire networks for transfer learning”. In: *2020 International Conference on Rebooting Computing (ICRC)*. 2020, pp. 102–106. DOI: 10.1109/ICRC2020.2020.00007.

[64] Audrius V Avizienis et al. “Neuromorphic atomic switch networks”. In: *PloS one* 7 (2012), e42772.

[65] Adam Z Stieg et al. “Self-organization and emergence of dynamical structures in neuromorphic atomic switch networks”. In: *Handbook of Memristor Networks*. Springer, 2014, pp. 391–427.

[66] Eleanor C Demis et al. “Atomic switch networks—nanoarchitectonic design of a complex system for natural computing”. In: *Nanotechnology* 26 (2015), p. 204003.

[67] Eleanor C Demis et al. “Nanoarchitectonic atomic switch networks for unconventional computing”. In: *Japanese Journal of Applied Physics* 55 (2016), 1102B2.

[68] Adam Z Stieg et al. “Emergent criticality in complex turing B-type atomic switch networks”. In: *Advanced Materials* 24 (2012), pp. 286–293.

[69] K. J. Miller et al. “Power-Law Scaling in the Brain Surface Electric Potential”. In: *PLoS Computational Biology* 5 (2009), e1000609.

[70] Adrian Diaz-Alvarez et al. "Emergent dynamics of neuromorphic nanowire networks". In: *Scientific reports* 9 (2019), pp. 1–13.

[71] A. Diaz-Alvarez et al. "Associative routing through neuromorphic nanowire networks". In: *AIP Advances* 10 (2020), p. 025134.

[72] Hugh G Manning et al. "Emergence of winner-takes-all connectivity paths in random nanowire networks". In: *Nature communications* 9 (2018), p. 3219.

[73] Gianluca Milano et al. "Brain-Inspired Structural Plasticity through Reweighting and Rewiring in Multi-Terminal Self-Organizing Memristive Nanowire Networks". eng. In: *Advanced intelligent systems* 2 (2020), 2000096–n/a.

[74] Qiao Li et al. "Dynamic electrical pathway tuning in neuromorphic nanowire networks". In: *Advanced Functional Materials* 30 (2020), p. 2003679.

[75] Z Kuncic et al. "Emergent brain-like complexity from nanowire atomic switch networks: Towards neuromorphic synthetic intelligence". In: *2018 IEEE 18th International Conference on Nanotechnology (IEEE-NANO)*. IEEE. 2018, pp. 1–3.

[76] Zdenka Kuncic et al. "Neuromorphic Information Processing with Nanowire Networks". In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020, pp. 1–5. DOI: 10.1109/ISCAS45731.2020.9181034.

[77] Chung-Wen Ho, A. Ruehli, and P. Brennan. "The modified nodal approach to network analysis". In: *IEEE Transactions on Circuits and Systems* 22 (1975), pp. 504–509. ISSN: 1558-1276. DOI: 10.1109/TCS.1975.1084079.

[78] Victor E. Mczgee and Willard T. Carleton. "Piecewise Regression". In: *Journal of the American Statistical Association* 65 (1970), pp. 1109–1124.

[79] Y. Lecun and C. Cortes. "The mnist database of handwritten digits". In: *http://www.research.att.com/yann/ocr/mnist/* (2010).

[80] Joschka Boedecker et al. "Information processing in echo state networks at the edge of chaos". In: *Theory in Biosciences* 131 (2012), p. 235. DOI: https://doi.org/10.1007/s12064-011-0146-8.

[81] Sam Lilak et al. "Spoken Digit Classification by In-Materio Reservoir Computing with Neuromorphic Atomic Switch Networks". In: vol. 3. Frontiers, 2021, p. 38.

[82] T. Chen et al. "Classification with a disordered dopant-atom network in silicon". In: *Nature* 577 (2020), pp. 341–345.

[83] Sam Lilak et al. “Spoken Digit Classification by In-Materio Reservoir Computing with Neuromorphic Atomic Switch Networks”. In: *Frontiers in Nanotechnology* 3 (2021), p. 38.