Moving Toward Intelligence: A Hybrid Neural Computing Architecture for Machine Intelligence Applications

Kang Jun Bai

Dissertation submitted to the Faculty of
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Electrical Engineering

May 7, 2021
Blacksburg, Virginia

Keywords: Neuromorphic Architecture, Machine Intelligence, Deep Learning, Reservoir Computing Network, Delay-Feedback Reservoir Network, Hybrid Neural Network, Internet of Things

Moving Toward Intelligence: A Hybrid Neural Computing Architecture for Machine Intelligence Applications

Kang Jun Bai

Abstract

Rapid advances in machine learning have made information analysis more efficient than ever before. However, to extract valuable information from trillion bytes of data for learning and decision-making, general-purpose computing systems or cloud infrastructures are often deployed to train a large-scale neural network, resulting in a colossal amount of resources in use while themselves exposing other significant security issues. Among potential approaches, the neuromorphic architecture, which is not only amenable to low-cost implementation, but can also deployed with in-memory computing strategy, has been recognized as important methods to accelerate machine intelligence applications. In this dissertation, theoretical and practical properties of a hybrid neural computing architecture are introduced, which utilizes a dynamic reservoir having the short-term memory to enable the historical learning capability with the potential to classify non-separable functions. The hybrid neural computing architecture integrates both spatial and temporal processing structures, sidestepping the limitations introduced by the vanishing gradient. To be specific, this is made possible through four critical features: (i) a feature extractor built based upon the in-memory computing strategy, (ii) a high-dimensional mapping with the Mackey-Glass neural activation, (iii) a delay-dynamic system with historical learning capability, and (iv) a unique learning mechanism by only updating readout weights. To support the integration of neuromorphic architecture and deep learning strategies, the first generation of delay-feedback reservoir network has been successfully fabricated in 2017, better yet, the spatial-temporal hybrid neural network with an improved delay-feedback reservoir network has been successfully fabricated in 2020. To demonstrate the effectiveness and performance across diverse machine intelligence applications, the introduced network structures are evaluated through (i) time series prediction, (ii) image classification, (iii) speech recognition, (iv) modulation symbol detection, (v) radio fingerprint identification, and (vi) clinical disease identification.

Moving Toward Intelligence: A Hybrid Neural Computing Architecture for Machine Intelligence Applications

Kang Jun Bai

General Audience Abstract

Deep learning strategies are the cutting-edge of artificial intelligence, in which the artificial neural networks are trained to extract key features or finding similarities from raw sensory information. This is made possible through multiple processing layers with a colossal amount of neurons, in a similar way to humans. Deep learning strategies run on von Neumann computers are deployed worldwide. However, in today's data-driven society, the use of general-purpose computing systems and cloud infrastructures can no longer offer a timely response while themselves exposing other significant security issues. Arose with the introduction of neuromorphic architecture, application-specific integrated circuit chips have paved the way for machine intelligence applications in recently years.

The major contributions in this dissertation include designing and fabricating a new class of hybrid neural computing architecture and implementing various deep learning strategies to diverse machine intelligence applications. The resulting hybrid neural computing architecture offers an alternative solution to accelerate the neural computations required for sophisticated machine intelligence applications with a simple system-level design, and therefore, opening the door to low-power system-on-chip design for future intelligence computing, what is more, providing prominent design solutions and performance improvements for internet of things applications.

Acknowledgements

I would like to express my deepest gratitude to my Ph.D. advisor, Prof. Yang Yi, for her dedicated support and guidance during my Ph.D. Study. I would also like to express my deep appreciation to all my Ph.D. advisory committee members, Prof. Dong S. Ha, Prof. A. Lynn Abbott, Prof. Wei Zhou and Prof. Rafael Davalos, and to my research mentor, Prof. Lingjia Liu, for their valuable comments and suggestions. I would also be grateful to my lab mates and friends in the Multifunctional Integrated Circuits and Systems (MICS) Group and the Wireless @ Virginia Tech Group as I benefited a lot from them. Specifically, I would like to show my special appreciation to my friends, Dr. Hongyu An (former Ph.D. student in MICS), Osaze Shears (current Ph.D. student in MICS) and Zhou Zhou (current Ph.D. student in Wireless @ Virginia Tech), for their support on emerging memory technology, field-programmable gate array (FPGA) implementations and wireless communication systems, respectively.

Over the years of research collaborations and internships with the Air Force Research Laboratory (AFRL) Information Directorate, I would like to express my deepest gratitude to my research mentors, Dr. Bryant Wysocki, Mr. Clare Thiem, Mr. Nathan McDonald and Ms. Lisa Loomis. I would also be grateful to my research mentor in Qualcomm, Mr. Sameer Wadhwa, for his support and advice.

Words fail to express my appreciation to my beloved parents for their unconditional love, care, support and devotion that they dedicated to me in my life.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Emerging data processing capabilities are highly desired in our data-driven society. General-purpose computers with the conventional von Neumann architecture have demonstrated their exceptional performance when executing basic mathematical instructions. However, the limitation of system buses between central processing unit (CPU), graphics processing unit (GPU) and storage unit (*e.g.*, flash memory) significantly reduces the computational efficiency when dealing with highly sophisticated machine intelligence applications, such as pattern recognition [1–4]. More importantly, the demand on computational resources and energy has increased with the increasing demand for data density, especially in today's data-intensive environment. Such required resources and energy have become the burden to the world's energy consumption [5].

In our daily life, we are constantly impacted by sensory impressions, for instance, recognizing familiar faces and hearing ongoing traffic. All these external impulses instantly produce a huge neural activity in our brain. To be specific, when recognizing familiar faces or sound from a crowd, our brain not only analyzes each trait, but also classifies and compares them with the known ones. More importantly, our brain could recognize the scene from a fraction of blurry images within a second, in which a classical computer takes minutes or even hours. It has been proved that our human brain is capable to constantly categorizing external impulses in different patterns for analyzing and learning with merely 20W of power consumption [6], which is more efficient than any supercomputers.

Emerged with the evolution of artificial intelligence (AI), deep learning strategies, taking advantages from human brains, provide systems the ability to automatically learn and optimize, in a way that is sufficient by training a large-scale artificial neural network (ANN) with a general-purpose learning algorithm. Throughout the development history of AI, deep learning strategies are optimized by utilizing high performance processors and big data [7–9]. Despite that the deep learning strategies have broken many performance records in machine intelligence applications, the general-purpose learning algorithm relied on the conventional von Neumann architecture significantly restricts the computational efficiency [10–12], more importantly, impeding such a powerful learning module to be deployed onto resource-constrain or power-limited portable platforms.

Arose with the introduction of neuromorphic architecture [13], application-specific integrated circuit (ASIC) chips have paved the way for deep learning strategies, accelerating the computational efficiency while reducing the hardware overhead. For instance, *TrueNorth*, fabricated by IBM in 28nm process, demonstrates $25,000\times$ power reduction over the one used by a classical computer [14,15]. *Loihi*, fabricated by Intel in 14nm process, demonstrates $1,000\times$ speedup with $100\times$ power reduction over CPU [16]. Moreover, a specialized ASIC-based convolutional neural network (AlexNet), fabricated by Stanford University in 45nm process, demonstrates $189\times$ speedup with $24,000\times$ power reduction over CPU and $13\times$ speedup with $3,400\times$ power reduction over GPU [17]. Last but not least, *Tianjic*, fabricated by Tsinghua University in 28nm process, demonstrates $100\times$ throughput enhancement with $10,000\times$ power efficiency acceleration over GPU [9,18].

In general, many learning algorithms used in deep learning strategies utilize some version of gradient descent during the training operation, and yet, such approaches are complicated by the vanishing gradient problem while themselves consuming a colossal amount of computational resources (*e.g.*, the storage capacity and the memory bandwidth during the operation). To this end, it is crucial to investigative a new class of neuromorphic architecture with a computational-efficient processing structure and learning algorithm.

1.2 Research Contribution

During my Ph.D study, my research aims to bridge the neuromorphic architecture and deep learning strategies by designing and fabricating a new class of hybrid neural computing architecture. To be specific, I introduce a spatial-temporal hybrid neural network built upon a delay-feedback reservoir network with a unique learning algorithm, reducing the implementation complexity of deep learning strategies on ASIC chips, while exhibiting a competitive accuracy across diverse machine intelligence applications. Major contributions are summarized as follows:

1. **Design and Fabrication of Delay-Feedback Reservoir Network**
   My goal in this project is to design a new class of nonlinear neural activation and spiking neurons for the reservoir computing network (RCN) and to explore the applications of RCNs in time series prediction and facial recognition. The insight of my approach is to adapt by simplifying the conventional RCNs through a single nonlinear neural activation and a delay-feedback topology with a chain of spiking neurons. The resulting delay-feedback reservoir (DFR) network has been successfully fabricated in *GlobalFrondries* 130nm BiCMOS process.

2. **Investigation of Neuromorphic Architecture with Memristive Synapses**
   In this project, my goal aims to design a convolution-immersed DFR (Ci-DFR) network with emerging memristor devices as electronic synapses, enabling a new computing solution for neuromorphic architecture with extremely high efficiency. The resulting Ci-DFR network has been successfully implemented on a printed circuit board with discrete memristor devices.

3. **Design and Fabrication of Spatial-Temporal Hybrid Neural Network**
   To support the integration of neuromorphic architecture and deep learning strategies, my goal in this project is to build a new class of hybrid neural network (HNN) by integrating both spatial and temporal information processing capabilities with the unique nature from DFR network. My techniques involve integrating the multilayer perceptron (MLP) and the DFR network to improve the network's learning capability, utilizing the in-memory computing strategy to accelerate the neural computations,

and adopting the unique learning mechanism to sidestep the vanishing gradient problem. The resulting spatial-temporal HNN (STHNN) has been successfully fabricated in *GlobalFrondries* 180nm CMOS process.

4. **Implementation of Spike-Timing-Dependent Plasticity**
   In the endeavor to accelerate the training efficiency, my goal in this project is to adapt the neural activity (*i.e.*, spikes) to carry out the training operation without the need of data conversion. My approach is to design a new class of training circuits and systems by utilizing the spike-time-dependent plasticity (STDP) topology with the use of supervised learning framework. The resulting STDP training circuits and systems are designed and optimized in *GlobalFrondries* 180nm CMOS process.

5. **Deployment of Deep Learning Strategies on Internet of Things**
   My goal in this project is to explore the applications of deep learning strategies on internet of things (IoT) applications, including communication and healthcare. More specifically, my approach is to deploy various deep neural network (DNN) models for modulation symbol detection in 5G multiple-input multiple-output orthogonal frequency-division multiplexing (MIMO-OFDM) systems, radio fingerprint identification in over-the-air WiFi environments, and clinical disease identification in healthcare.

The rest of this dissertation is organized as follows: Chapter 2 provides an overview of neuromorphic architecture and deep learning strategies. Chapter 3 and Chapter 4 exhibit the design solution for high-performance DFR network and neuromorphic architecture with reconfigurable memristive synapses, respectively, followed by the design strategy of STHNN with in-memory computing acceleration in Chapter 5. The implementation of deep learning strategies on IoT applications is discussed in Chapter 6, and the dissertation is then concluded in Chapter 7.

Chapter 2

Programming versus Deep Learning

Benefited by the Moor's Law, general-purpose computers built based upon the von Neumann architecture have been deployed worldwide in past several decades [19]. The success in the development was firstly enabled by doubling the computational performance, followed by the multi-core computing architecture [20]. However, the fact that separating the location of processing units and memory storage significantly restricts the computational efficiency, especially in today's data-driven society. This is where neuromorphic architecture comes into help, in a way that is sufficient by training large-scale artificial neural networks (ANNs) with a colossal amount of data, replicating the way that we humans learn.

2.1 Neuromorphic Architecture

With merely 20W of power consumption, our human brain is capable to process raw sensory impulses to enable learning and analyzing activities. More importantly, our humans along with other mammalian creatures are capable to adapt their behaviors according to environment changes with a unique historical self-learning capability. These advantages are attributed from the parallel operation in low frequency. Compared to conventional von Neumann computers based on arithmetic operations, the unique signal processing natures in mammalian brain are fundamentally different, as summarized in Table 2.1. Such distinctions

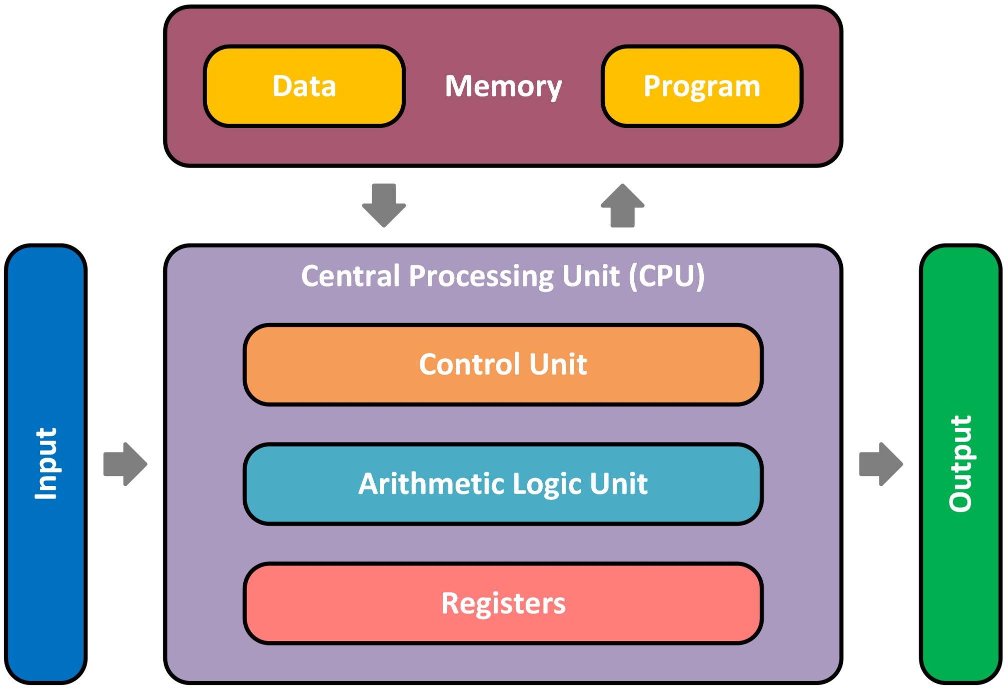
Table 2.1: Comparison of von Neumann computer to human brain.

	Von Neumann Computer	Human Brain
Processing Elements	central processing unit (CPU)	soma
Computing Units	arithmetic logical unit (ALU)	artificial neuron
Memory Storage	random-access memory	synapses
Signal Transmission	system buses	dendrites and axons
Transmission Scheme	one-to-one	many-to-many
Format of Signal	binary	spike
Learning Mechanism	pre-programmed instructions	self-learning
Level of Complexity	low	high
Operating Frequency	high (within GHz range)	low (within kHz range)
Power Consumption	high (within kW range)	low ($\approx$ 20W)

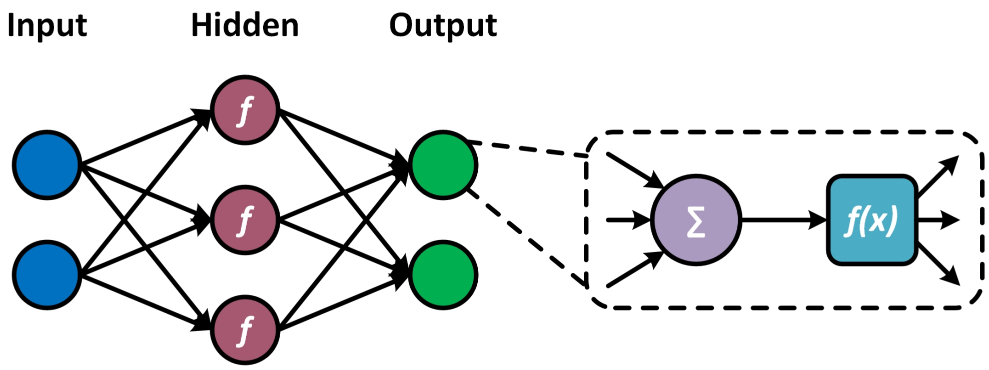
of processing structure and working mechanism result in a performance difference between von Naumann computers and human brain. As the development of von Naumann computers on machine intelligence applications is suffered by the computational resources spent on data transmission [21], it is essential to develop a novel computing architecture, in a way to replicate the working mechanism of our human brain.

The concept of neuromorphic architecture was developed by Dr. Carver Mead in 1980s [13], replicating the natural neuro-biological behaviors with very large-scale integration (VLSI) technology and highly parallel computing architectures. This is made possible by rebuilding three critical components: (i) neuron, (ii) synapse, and (iii) network structure.

General architectures of von Neumann computers and ANNs are depicted in Fig. 2.1. In a classical von Neumann computer, raw sensory inputs are digitized and subsequently processed by a set of pre-programmed instructions through arithmetic logical units (ALUs), which are usually executed by a combination of logical blocks [22–24]. After a series of executions, the computed digitized information is then converted back into analogue signals for



(a)



(b)

Figure 2.1: General architectures of (a) von Neumann computers with pre-programmed instructions and (b) artificial neural networks with self-learning capability.

Table 2.2: Overview of key features on contemporary neuromorphic chips.

	SpiNNake [25]	Neurogrid [26]	TrueNorth [14]	Loihi [16]
Technology	130nm	180nm	28nm	14nm
# of Neurons	20,000	65,000	1 million	130,000
Design Strategy	–	analog	digital	digital
# of Synapses	20 million	–	256 million	130 million
Silicon Area	$102mm^2$	$168mm^2$	$430mm^2$	$60mm^2$
Neuron Density	204 per mm^2	390 per mm^2	2,438 per mm^2	2,184 per mm^2
Synapse Density	0.2 million per mm^2	–	0.6 million per mm^2	2.1 million per mm^2
Power Density	$0.012mW/mm^2$	$18mW/mm^2$	$0.15mW/mm^2$	–

visualization. ANNs, on the other hand, are capable to generate a general solution by learning a colossal amount of problems without any predefined instructions. More importantly, analogue information can be processed without digitization. Machine learning algorithms are what underlies ANNs, implementing a general-purpose learning algorithm that allow ANNs to automatically learn and optimize.

The methodology behind ANNs is that a network is built upon layers of neurons with interconnected electronic synapses. In general, raw sensory information propagates between neurons while having an appropriate scaling with weights and a nonlinear transformation. The final outcomes are influenced not only by the nonlinear transformation, but also by the way that the electronic synapses are interconnected. In other words, the output can be manipulated by adapting the strength of electronic synapses. Such a manipulating mechanism can be seen as the training operation, while the neural-like computing structure is generally referred to the neuromorphic architecture.

Neuromorphic architecture, also referred to the cognitive computing systems in recent years, exhibits a path of designing a high-performance and high-efficient computing system, specifically targeted on accelerating computational efficiency while reducing the hardware overhead for machine intelligence applications [9, 14, 16, 25–46]. An overview of key features on contemporary neuromorphic chips is summarized in Table 2.2.

2.2 Deep Neural Network

Deep learning strategies are the cutting edge of artificial intelligence (AI), in which ANNs are trained to extract key features or finding similarities from raw sensory information. This is made possible through multiple processing layers with a colossal amount of neurons, that is, deep neural networks (DNNs) are what underpins deep learning strategies.

The general structure of DNNs can be represented as a hierarchical organization of neurons interconnected by electronic synapses. By concatenating layers of neurons in a processing pipeline, a complex network is created to carry out neural computations and to learn with some feedback mechanisms. To be specific, a neuron is activated and passes computed information to others only if the incoming signal results in a value greater than certain threshold, otherwise ignored. The incoming signal to a neuron is influenced by the computed signals from its former layer and the associated synaptic weights. In a DNN, initial synaptic weights are all random but during the training operation, these synaptic weights are updated interactively, in a way that is sufficient by learning to predict a correct output.

Depending on network configurations, the structure of DNNs can be generally categorized into two aspects, *i.e.*, the feedforward neural networks (FNNs) and the recurrent neural networks (RNNs). The former aims to extract key features from static data (*e.g.*, images) while the latter aims to discover similarities from temporal information (*e.g.*, speech). In recent machine intelligence applications, DNNs can represent functions of increasing complexity by deploying more hidden layers and associated neurons.

2.2.1 Feedforward Neural Network

In the FNN family, neurons are divided into separate sequence layers while signal can only be propagated forward without internal loops. Each layer simply executes the operation of

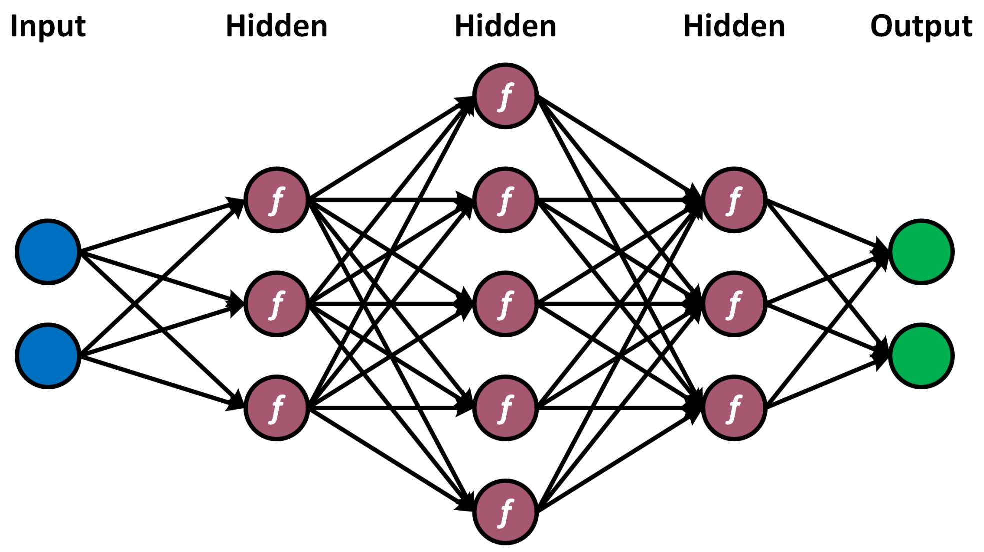


Figure 2.2: General processing structure of feedforward neural network.

$$h_l = \begin{cases} f(x \cdot W_{in} + b_{in}), & \text{if } l = 1, \\ f(h^{l-1} \cdot W_h^l + b_h^l), & \text{otherwise,} \end{cases} \tag{2.1}$$

where $f()$ is a nonlinear neural activation, x is a set of input vectors, l indicates the l-th hidden layer, W_{in} and W_h denote input weights and internal weights, respectively, and b is bias vectors. The output state can be then calculated as

$$\hat{y} = g(h^{last} \cdot W_{out} + b_{out}), \tag{2.2}$$

where $g()$ is the softmax function to compute the probability distributions of a given input, h^{last} is network states in the last hidden layer, and W_{out} is output weights. The resulting output only contains information of a single trajectory of input history, and thus, such a computing structure is designed to process static data. Multilayer perceptron (MLP) and convolutional neural networks (CNNs) are quintessential models in the FNN family. The former is only made of dense layers, while the latter is made of convolutional layers, pooling layers and dense layers. With the capability of capturing both spatial and temporal dependencies, CNNs perform a better design solution for filtering images, and thus, understanding the sophistication of images better. Despite that CNNs have broken many performance records in pattern recognition [7, 47–57], the network's accuracy and the training time are significantly impacted by the number of convolution layers and kernels.

2.2.2 Spiking Neural Network

In the endeavor to integrate neuroscience and deep learning strategies, DNN models in recent years aim to create a network that replicates the behavior of brain cortex. By incorporating with the discrete event-driven processing structure, spiking neural networks (SNNs), a new class of FNN, propagates neural information through biologically-realistic signals (*i.e.*, spikes), as demonstrated in Fig. 2.3. Essentially, SNNs take advantages of specialized network topology, exhibiting favorable properties in neural circuits and systems [58, 59]. In recent machine intelligence applications, SNNs have been proven to be a powerful network model in the domain of pattern recognition [60–65]. However, SNNs are not differentiable, where the gradient descent cannot be adapted for the training operation, otherwise losing the precise temporal information within spikes. Such training properties significantly limit the performance of SNNs in real-world applications.

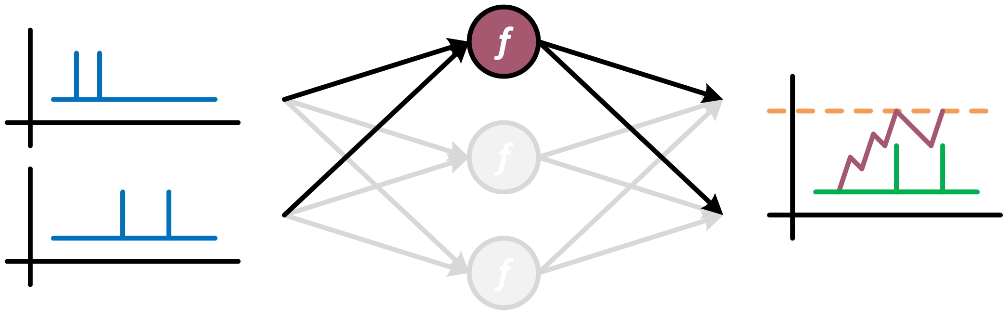


Figure 2.3: General operating principle of spiking neural network.

2.2.3 Recurrent Neural Network

By contrast, RNNs are built upon FNNs with recurrent connections added to the hidden layer, as shown in Fig. 2.4. Each neuron within the hidden layer has a feedback loop to create a dynamical memory, and thus, having a similar temporal dynamic as in SNNs [66]. With the recurrent nature, the state of network are influenced not only by the present input, but also by the context based on historical information stored in the network itself, which can be expressed as

$$h_t = f(x_t \cdot W_{in} + h_{t-1} \cdot W_h + b_h), \tag{2.3}$$

where x_t is the input at present time, and h_{t-1} is the state of network at previous computing cycle. The output state of RNNs at present time can be then calculated with the same correlation as in FNNs, as depicted in Eq. 2.2.

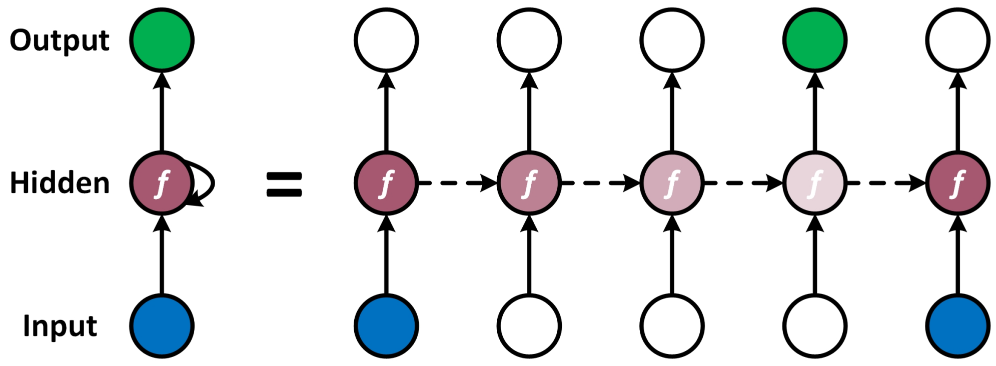


Figure 2.4: General processing structure of recurrent neural network.

With the implementation of an interacting cell and several regulating gates built upon the classical RNN model (known as vanilla RNN [67]), both long short-term memory (LSTM) [68] and gated recurrent unit (GRU) [69] have the capability to determine how much of each information should be passed forward or removed. Such a learning mechanism enables both LSTM and GRU to have a better control-ability and performance over vanilla RNN. Beyond that, the Hopfield network demonstrates the capability to emulate the human memory, offering the content-addressable (*i.e.*, associative) memory natures during the training operation [70]. Because of the sequential nature of recurrent connections, RNNs are widely deployed in temporal-related applications, such as time series prediction, speech recognition, and natural language processing [12, 71–78]. Nevertheless, the highly nonlinear nature of RNNs has become a major factor that limits the computational efficiency on the training operation. What is more, the required computational resources lead to a significant hardware overhead, impeding such powerful computing modules to be deployed onto resource-constrain or power-limited portable devices.

2.2.4 Reservoir Computing Network

Reservoir computing networks (RCNs) are a recently introduced deep learning paradigm that is built by simplifying the processing structure based on classical RNNs [79–85]. The concept underlying RCNs is the mechanism that our human brain process information by producing patterns of transient neuronal activity [86]. A general RCN consists of three computing layers as in classical RNNs, in which the reservoir layer is built with a group of sparsely-connected neurons, as depicted in Fig. 2.5. The state of reservoir dynamics expresses a similar correlation as in classical RNNs, which can be written as

$$h_t = f(x_t \cdot W_{in} + h_{t-1} \cdot W_{res} + y_{t-1} \cdot W_{fb}), \tag{2.4}$$

where y_{t-1} is the output state at previous computing cycle, W_{res} is specialized internal weights, and W_{fb} denote feedback weights from the output layer to the reservoir layer. The output state of the network can be also calculated with the same correlation as in FNNs, as depicted in Eq. 2.2.

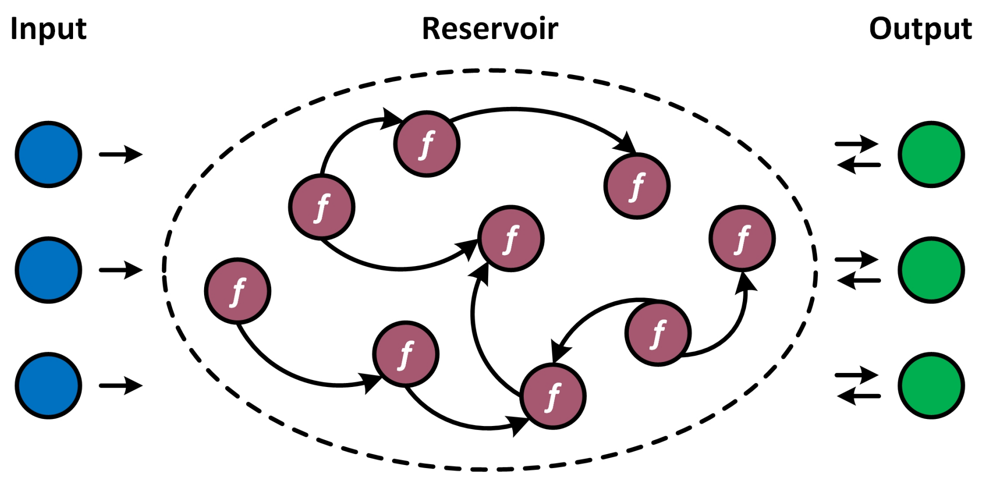


Figure 2.5: General processing structure of reservoir computing network.

In classical RNNs, all weight matrices and bias vectors are needed to be trained. By contrast, RCNs take the advantages from linear algorithms, in a way that is sufficient by only training the linear readout weights. In order to properly compute the reservoir principle, internal weights in the reservoir layer are must initialized according to the echo state property (ESP) [87], which can be expressed as

$$W_{res} := \frac{|\lambda_{max}(W_{res})|}{\sigma} \cdot W_{res}, \tag{2.5}$$

where $|\lambda_{max}(W_{res})|$ is the absolute maximum eigenvalue of $W_{res} \in [-1, 1]$, and $\sigma < 1$ denote the spectral radius. Once W_{in} and W_{res} are initialized, these weighted values remain fixed for the entire simulation.

High-dimensional mapping and fading memory are two other critical properties for computing the reservoir principle. The former aims to nonlinearly project sequential inputs onto a higher dimensional space to enhance the network's separability, while the latter adopts the historical learning mechanism as in classical RNNs to improve the network's learning capability. High-dimensional mapping is the key operation in RCNs for separating random inputs into different categories. As illustrated in Fig. 2.6, two various objects cannot be linearly separated in a low-dimensional space. By contrast, the network's separability optimizes accordingly as inputs are projected onto a higher dimensional space (e.g., from two-dimensional to three-dimensional), and thus, separating two various objects with a linear hyperplane.

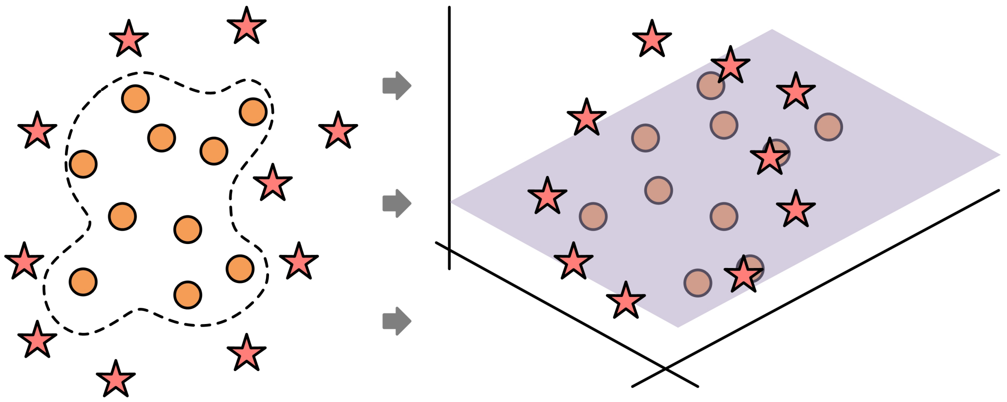


Figure 2.6: Illustration of high-dimensional mapping.

The echo state network (ESN) [79] and the liquid state machine (LSM) [80] are the two well-known RCN paradigms. In general, the use of signal topology is the major attribute that set these two models apart, where numeral numbers are adopted in ESN while spikes are adopted in LSM. In recent machine intelligence applications, it has been found that the performance metrics of RCNs outperform classical RNNs by three-order of magnitude more accurate in time series prediction [45, 82]. Better yet, RCNs have also proven their benefit in speech recognition [88–92], optical character recognition [93, 94], grammar modeling [95], noise modeling [96], and robotics [97–100].

2.3 Nonlinear Neural Activation

In DNN designs, the correlation between pre-neuron and post-neuron activities is generally defined by a nonlinear activation function (NNA), mapping the resulting values within $[0, 1]$, $[-1, 1]$, etc. based upon applications.

2.3.1 Unit Step Function

In the past century, the unit step function, as shown in Fig. 2.7a, was firstly introduced as a NNA [101], which can be expressed as

$$f_{step}(x) = \begin{cases} 0, & \text{if } x < 0. \\ 1, & \text{otherwise.} \end{cases} \tag{2.6}$$

The unit step function is widely adopted in SNNs, allowing spiking neurons to be activated only if the incoming signal results in a value grater than a specific threshold. However, as the unit step function is not differentiable and immutable, such a function cannot be applied in classical FNNs and RNNs.

2.3.2 Rectified Linear Unit

The rectified linear unit (ReLU) [102] is the most commonly used NNA in recent machine intelligence applications, more specifically, in FNNs. The ReLU function, as depicted in Fig. 2.7b, ranges within $[0, \infty]$, which can be written as

$$f_{ReLU}(x) = max(0, x). \tag{2.7}$$

The ReLU function is monotonic and differentiable, where its derivative can be denoted using a unit step function as presented in Eq. 2.6. In practice, the fact that forcing all negative values to zero significantly reduces the network's learning capability, known as the dying ReLU problem [103]. To be specific, synaptic weights will not be updated and corresponding neurons will stop responding to variations in gradient errors, because the gradient goes towards zero during the training operation.

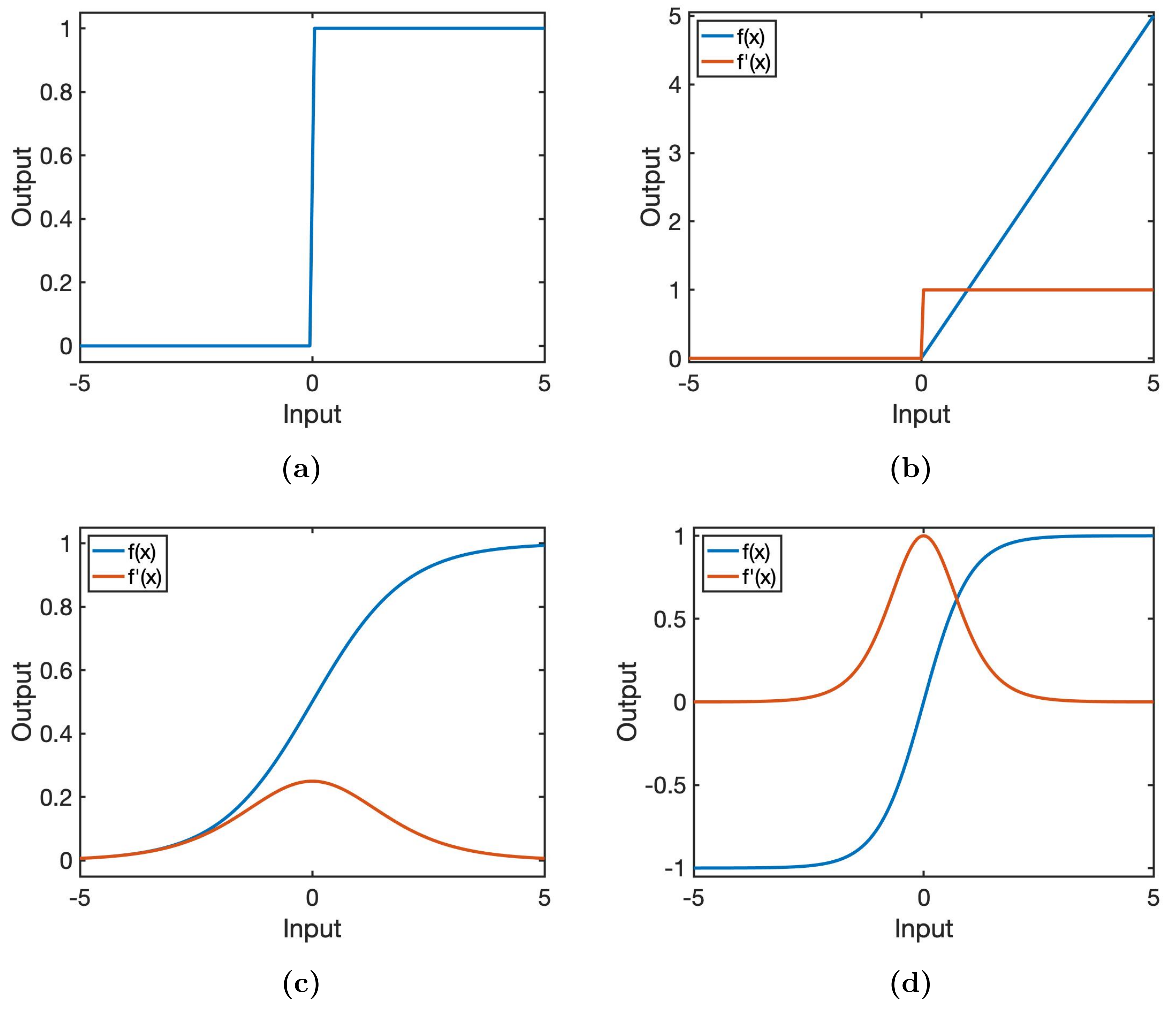


Figure 2.7: Common nonlinear activation functions and their derivative on (a) unit step, (b) rectified linear unit, (c) sigmoid, and (D) hyperbolic tangent.

2.3.3 Sigmoid and Hyperbolic Tangent

By contrast, sigmoid and hyperbolic tangent (tanh) are the most commonly used NNAs in RNNs, as demonstrated in Fig. 2.7c and 2.7d, respectively, in which tanh is antisymmetric with respect to the origin. The mathematical representation of sigmoid function and its derivative can be expressed as

$$f_{sigmoid}(x) = \frac{1}{1 + e^{-x}}, \tag{2.8}$$

$$f'_{sigmoid}(x) = f_{sigmoid}(x)(1 - f_{sigmoid}(x)), \tag{2.9}$$

while the mathematical representation of tanh function and its derivative can be denoted as

$$f_{tanh}(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}. \tag{2.10}$$

$$f'_{tanh}(x) = 1 - f_{tanh}(x)^2 \tag{2.11}$$

It has been proven that the tanh function converges faster than the sigmoid function [104]. However, both sigmoid and tanh functions are suffered from the vanishing gradient problem [105]. Such a property indicates that the gradient towards either end of these functions tends to respond very less to change, and therefore, slowing down the learning operation or even refusing the network to learn.

2.4 Training Algorithm

DNNs simply carry out the neural computations based on their network structure, in which the resulting outputs would be a classification or a prediction. Based upon the resulting outputs, some feedback mechanisms can be applied to enable the learning operation, allowing the network to improve so as to classify or predict better. Such a learning operation is adapted by updating synaptic weights between neurons and hidden layers, in a way to minimize the cost function, which can be generally defined as

$$C = cost(y, \hat{y}) \tag{2.12}$$

where $\hat{y}$ denotes the predicted output from network, and y denotes the expected outcome. Several cost functions can be adapted during the learning operation, such as the empirical loss, the mean square error, the cross-entropy loss, etc.

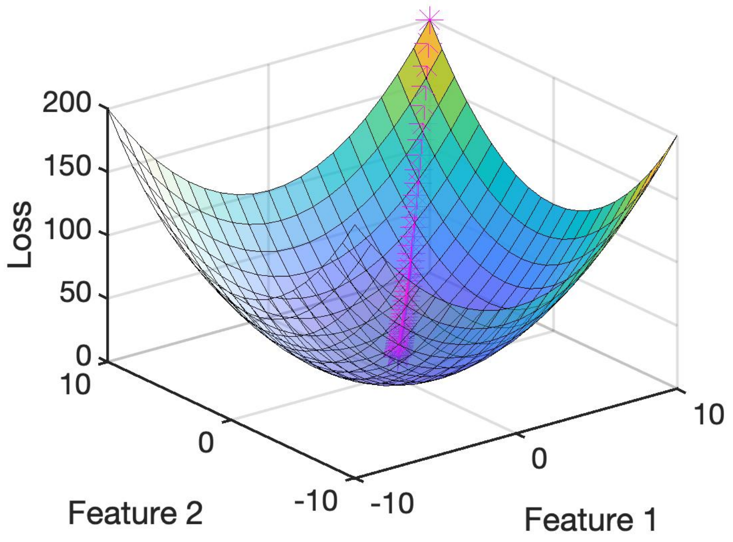


Figure 2.8: Applying gradient decent training algorithm to minimize the loss.

The backpropagation algorithm, as depicted in Fig. 2.8, aims to minimize the cost function by optimizing synaptic weights and bias vectors with some version of gradient descent with respect to a learning rate of η, which can be generally written as

$$W := W - \eta \frac{\partial C}{\partial W}, \tag{2.13}$$

$$b := b - \eta \frac{\partial C}{\partial b}. \tag{2.14}$$

In general, the bigger the gradient, the bigger the adjustment to synaptic weights and bias victors, or vice versa. In practice, such a method would be suffered by the vanishing gradient problem [106]. To be specific, as the gradient of cost function propagates recursively backward, the resulting gradient will gradually shrink. More importantly, as the network gets deeper, the gradient presented in the earlier layers will be plainly small or even non-existent, and thus, slowing down the learning operation or even refusing the network to learn. Nevertheless, the backpropagation algorithm is still the most powerful and successful training algorithm for DNNs.

Chapter 3

High-Performance Delay-Feedback Reservoir Network

3.1 Introduction

Deep neural networks (DNNs) have matured to provide intelligence systems that replicate the neural-biological processes of our human brain, demonstrating remarkable success across diverse machine intelligence applications. A pristine DNN with a colossal amount of neurons is not capable to generate a general solution until its synaptic weights are properly trained by tremendous amount of data. That is, the capability of DNNs deployed for real-world applications is associated not only with the network structure, but also with the data volume [107–109]. It can be observed from Fig. 3.1 that the data volume used for big data analytic has explosively increased over the past decade [110], potentially improving the capabilities of DNNs. Nevertheless, the demand on computational resources and energy has significantly increased with the increasing demand for data volume. To this end, it is essential to investigate a new class of neuromorphic architecture with a simple processing structure, high reliability and low power consumption.

Reservoir computing networks (RCNs), an emerging machine learning paradigm, are built based upon the classical recurrent neural networks (RNNS), in which the echo state network

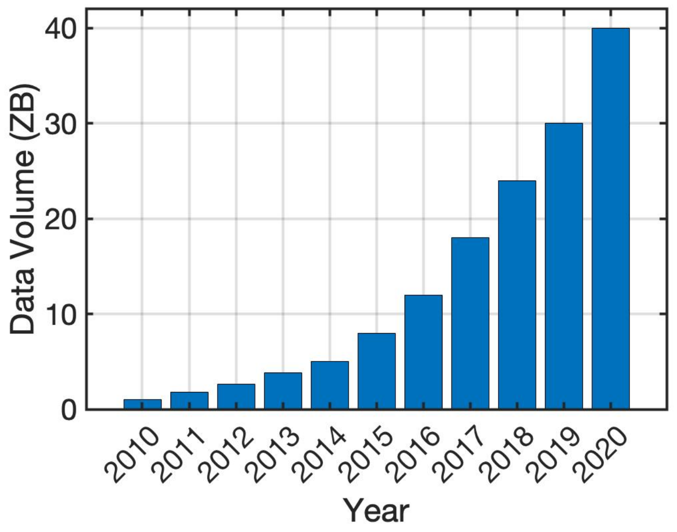


Figure 3.1: Global growth trend of data volume over the past decade.

(ESN) [79] and the liquid state machine (LSM) [80] are deployed to keep pace with the explosive escalation of data density on machine intelligence applications. The internal layer of RCNs is built of sparsely connected neurons with fixed synaptic weights, and thus, the learning operation is hereby carried out by only updating the linear readout weights. RCNs, in recent years, have been fully developed in both software (*e.g.*, TensorFlow, PyTorch, Caffe, etc) and hardware (*e.g.*, field programmable gate array (FPGA)), and have demonstrated their advantages over classical RNNs across diverse machine intelligence applications [90, 111–114]. Despite that the RCNs offer significant reduction on training complexity while yielding a competitive classification or prediction accuracy, the fact that realizing a colossal amount of nonlinear neurons in hardware is still suffered from the computational cost and design overhead.

A key finding emerged that the our human brain operates in the transition regime between periodic and chaotic [115], known as the edge-of-chaos [116], leading to more advanced information processing capabilities. Such a hypothesis can be also deployed in DNNs with the introduction of delay-feedback systems and time-multiplexing, particularly suitable for RNNs due to the recurrent nature [117]. It has been proven that the computational performance in targeted systems can be significantly improved with the embedded edge-of-chaos computing characteristic [118, 119].

My goal in this project is to design a new class of nonlinear neural activation and spiking neurons for the delay-feedback reservoir network and to explore the applications of RCNs in time series prediction and facial recognition. The insight of my approach is to adapt by simplifying the classical RCNs through a single nonlinear neural activation and a delay-feedback topology with a chain of spiking neurons. Major contributions of this project are summarized as follows:

- A novel design solution for delay-feedback reservoir network built based upon a delay-dynamic architecture with the time-multiplexing characteristic, exhibiting a rich dynamic behaviors by varying the network dynamic in between periodic and chaotic.
- A prototype fabrication with fully-analog components, yielding an average power consumption of 529μW. To the best of our knowledge, this fabricated prototype is the first implementation of delay-feedback reservoir network with analog integrated circuit (IC) design technique.
- Up to $6.79\times$ error reduction over the state-of-the-art RCN models on a time series prediction benchmark.
- A classification accuracy of 98% on facial recognition, yielding 26 percentage points more robust against noise compared to the multilayer perceptron.

The rest of this chapter is organized as follows: Section 3.2 provides an overview and related works of RCNs. The design methodology of the introduced delay-feedback reservoir network and the performance evaluations of the fabricated prototype are discussed in Section 3.3 and Section 3.4, respectively, followed by the software-based experimental evaluations in Section 3.5. This chapter is then concluded in Section 3.6.

3.2 Time-Delay Reservoir Network

Nonlinear systems with a delay-feedback coupling can be referred to a class of dynamic systems, which are ubiquitous in a variety of real life systems [120]. Such behaviors can be found in the transport of substances, the conduction time of nerves, the regulation of gene, etc [121]. With the everlasting enthusiasm of neuroscience, a key finding emerged that our human brain operates in the transition regime between periodic and chaotic. To support the integration of neuroscience and DNNs, such a dynamic variation nature is hereby deployed in RCNs with the capability to improve the computational performance. From the mathematical point of view, the transition of system dynamic can be controlled by varying the timing-coefficient of a delay-feedback system [118]. Such a property is described by the delay differential equations, defining a system in which the dynamics depend on both present and previous states, and changing its dynamic behaviors accordingly as the delay varies.

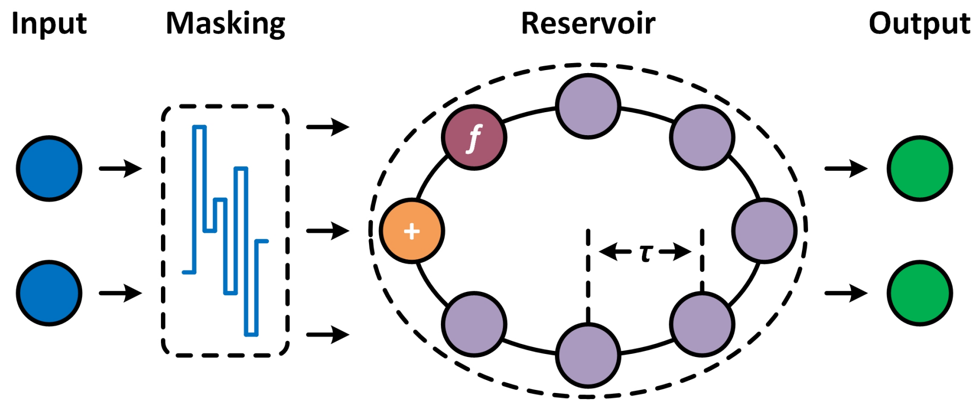


Figure 3.2: General processing structure of time delay reservoir network.

The delay-feedback system is expected to be the most suitable architecture in RCNs because of its nonlinear transformation and high-dimensional mapping natures. With the introduction of time-multiplexing deployed on a delay-feedback system, the reservoir dynamics are hereby given by a single nonlinear neural activation and a delay-feedback loop, as demonstrated in Fig. 3.2. Such an evolutionary structure is referred to the time-delay reservoir (TDR) network, in which the state of reservoir dynamics can be simplified as

$$h_t = f(x_t \cdot MK + h_{t-1}), \tag{3.1}$$

where MK is a mask function, in which the time length per frame is identical to the time interval between neurons in the delay-feedback loop. The mask function introduces the time-multiplexing with random scaling factors to inputs, ensuring that the system always resides in the temporal domain. Along the delay-feedback loop, the temporal separation together with the short-term memory experience the high-dimensional mapping with the potential to classify otherwise non-separable functions.

Such a delay-feedback-immersed RCN significantly reduces the design complexity of DNNs in application-specific integrated circuit (ASIC) chips with analogue hardware in either electronically [27, 88, 112, 113, 122–125] or optically [126–133]. For instance, in [112], a LSM model is implemented with 135 neurons in a FPGA for pattern recognition, yielding a recognition accuracy of 96.4%. Moreover, in [113] and [123], two different LSM models with 200 and 343 neurons, respectively, are implemented in a FPGA for speech recognition, yielding a recognition accuracy of 95% and 99.79%, respectively. Last but not least, in [124], an ESN model is implemented in a FPGA for time series prediction, yielding a mean square error as low as 2.39×10^{-4}.

The photonic implementations of TDR Network introduce the phenomenon of optical chaos, attracting wide-spread attention in recent years. For instance, [128] and [130] introduce a photonic implementation of TDR network with the semiconductor optical amplifiers (SOAs), offering a high-speed optical information processing. More importantly, photonic devices are nonlinear in nature, potentially reducing the hardware overhead. Nevertheless, the photonic implementation often requires expensive peripheral devices, such as digitizer and waveform generator, resulting in a lower mobility.

On the other hand, digital IC implementations offer compact design area, low power consumption and noise immunity. For instance, [88] and [125] introduce a digital TDR network built with discrete components, demonstrating a potential implementation capability of TDR network with very large-scale integration (VLSI) circuits and systems. However, real-time operations require the interface with raw sensory information in an analog format, and therefore, power-hungry peripherals are necessary, for instance, operational amplifiers, analog-to-digital converters (ADCs), digital-to-analog converters (DAC), etc. By contrast, analog IC implementations process analog signals directly without the need of data conversions, hence, significantly reducing the design complexity and hardware overhead. More importantly, analog implementations closely mimics the physical characteristic of neurological systems, in a similar way that how we human brain process information.

3.3 Design Strategy of Delay-Feedback Reservoir Network

3.3.1 Network Architecture

An overview of the delay-feedback reservoir (DFR) network is demonstrated in Fig. 3.3. The DFR network is built based upon the conventional TDR network, comprising of a temporal encoder and a dynamic reservoir layer with the delay-feedback topology.

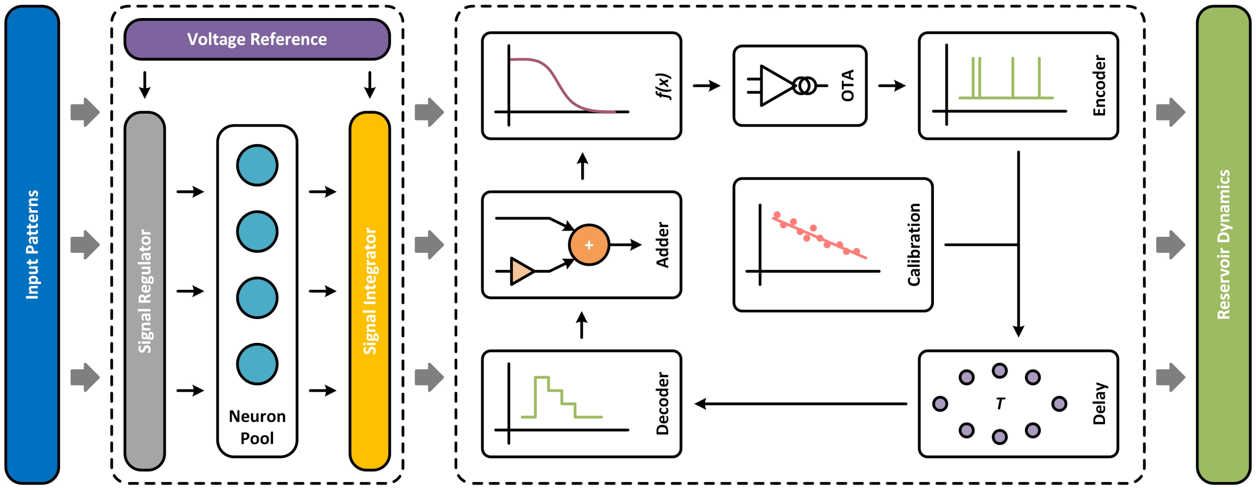


Figure 3.3: System-level architecture of delay-feedback reservoir network.

Unlike the classical TDR network, the masking interface is substituted by a temporal encoder [134–136] in the introduced DFR network, relating input with time-multiplexing nature while representing the post-neuron signals by an inter-spike-interval (ISI) temporal spike train. To be specific, the carried information from raw sensory inputs is encoded into the time intervals, D_i, between spikes, which can be written as

$$D_i = s_{t+\tau}(C_m, V_{th}) - s_t(C_m, V_{th}), \tag{3.2}$$

where $s_{t+\tau}$ and s_t denote the firing time of spikes presented at time $t+\tau$ and t, respectively, which can be generally expressed as

$$s(C_m, V_{th}) = C_m \cdot \frac{V_{th}}{I_{in} - I_{leak}}, \tag{3.3}$$

where C_m denotes the membrane capacitance, V_{th} denotes the firing threshold, I_{in} and I_{leak} denote raw sensory input and leakage currents, respectively. Such an encoded ISI spike train allows each spike to be the reference frame with respect to each other, as depicted in Fig. 3.4, conveying more information for latter computations.

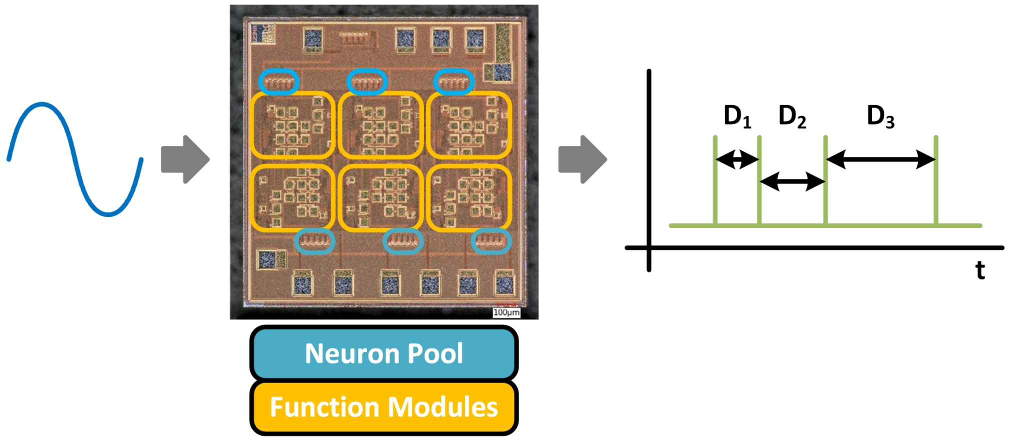


Figure 3.4: Illustration of temporal encoder and the fabricated prototype.

The processing structure of the dynamic reservoir layer in the introduced DFR network is built and optimized based upon the classical TDR network implementation strategy, composing of a sigmoid nonlinear neural activation (NNA), an analog-to-spike (A/S) encoder, a dynamic delay-feedback loop (DFL) with a delay calibration module, a spike-to-analog (S/A) decoder, and an analog adder. During the operation, time intervals between spikes on an ISI temporal spike train from the input encoder are fetched into the sigmoid NNA to carry out the nonlinear transformation. The selected activation data are then digitized back into the a temporal spike train by the A/S encoder, propagated along the dynamic DFL, and eventually integrated with the next incoming input to create the short-term memory. Such a property establishes connections within the context of data, enabling the historical learning capability, which can be expressed as

$$h_t = f((1-\alpha) \cdot D_t^i + \alpha \cdot h_{t-1}). \tag{3.4}$$

To minimize the information lost during computations, the dynamic DFL is built followed by a S/A decoder and a signal gain regulator, α, ensuring that the new incoming input is dominant. Moreover, to realize a variety of dynamic behaviors with respect to the time-multiplexing, a controllable delay calibration module is deployed to regulate the timing-coefficient along the dynamic DFL. Last but not least, by adopting the spiking information processing technique, the introduced DFR network not only enables a novel design solution for TDR network with analog IC implementations, but also eliminates the complex data conversion, and thus, power-hungry peripherals are unnecessary, potentially making the system suitable for large-scale DNN designs.

3.3.2 Learning Rule

During the training operation, the trajectory of reservoir dynamics is computed by feeding the training samples, $\{x_j\}_{j=1}^{m}$, where m is the input dimension. As the outcomes, a set of internal states, $\{h_t\}_{t=1}^{m}$, is obtained. Consequently, optimal output weights can be calculated directly through the Tikhonov regularization, which can be defined as

$$W_{out} = Y \cdot S' \cdot (S \cdot S' + \eta \cdot I)^{-1}, \tag{3.5}$$

where S' represents the transpose matrix of reservoir dynamics $S = [h_0, h_1, \cdots, h_n]$, $Y = [y_0, y_1, \cdots, y_n]$ is the target output, n is the number of training samples, $\eta \geq 0$ is a constant, and I is the identity matrix with the same size of reservoir state. The general learning operation of the introduced DFR network is summarized in Algorithm 1.

3.3.3 Hardware Implementation

Sigmoid Nonlinear Neuron Activation

It can be observed that the sigmoid NNA is a critical module that plays an important role in the operation of high-dimensional mapping. The simplified design scheme of sigmoid NNA is depicted in Fig. 3.5.

Algorithm 1: Delay-Feedback Reservoir Network

```
Data: $x = [n, m], y = [n, u], scaling = \alpha$
Result: $W_{out}$
initialization
for $i \rightarrow 1$ to $n$ do
    for $ii \rightarrow 1$ to $m$ do
        $D_t = s_{t+\tau}(m_{ii}) - s_t(m_{ii})$
    end
    return $D_t$
    for $ii \rightarrow 1$ to $D_t$ do
        $h_t = f((1-\alpha) \cdot D_t + \alpha \cdot h_{t-1} + b_h)$
    end
end
return $h_t$
$S = [h_0, h_1, \cdots, h_n]$
$Y = [y_0, y_1, \cdots, y_n]$
$W_{out} = Y \cdot S' \cdot (S \cdot S' + \eta \cdot I)^{-1}$
```

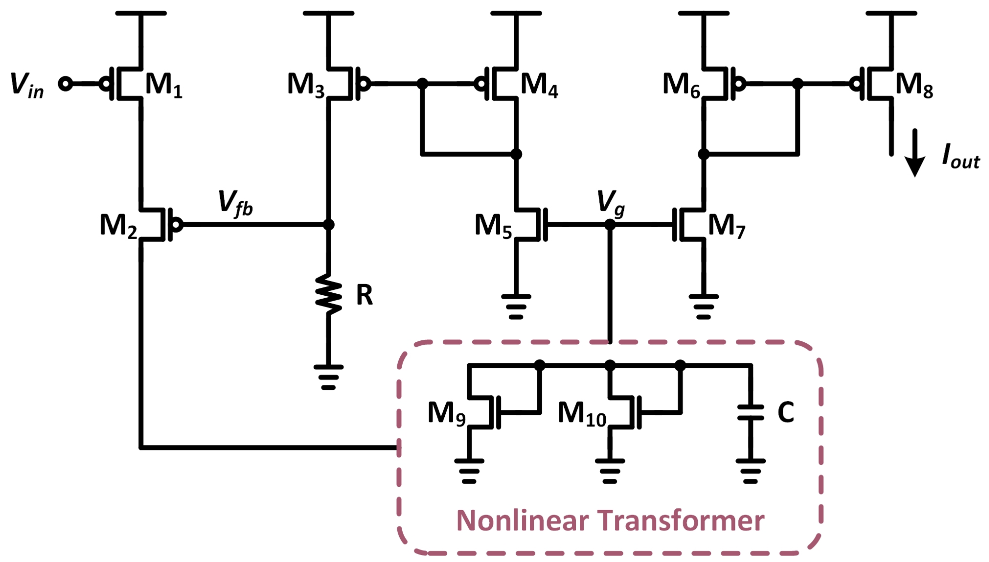


Figure 3.5: Design scheme of sigmoid nonlinear neural activation.

The nonlinear transformation is achieved by sensing the voltage level of inputs, V_{in}, through the nonlinear transformer, as shown in the dashed box in Fig. 3.5. In the resetting phase, the input, V_{in}, is set to be V_{DD}. Such a high voltage disables the input trigger, M_1 & M_2, and discharges the nonlinear transformer, thereby, the output remains at 0V. In the decision-making phase, the nonlinear transformer continuously tracks the variations of input and charges up its intrinsic capacitor to regulate V_g. When $V_g > V_{th,M7}$, the drain-to-source voltage of M_5 increases accordingly and eventually reaches its saturation region, and thus, the diode-connected M_6 fully enables the output current mirror, M_6 & M_8, to generate the corresponding output current. In the meantime, the feedback current mirror, M_3 & M_4, induces a high voltage at V_{fb} to disable the input trigger, and therefore, fully resting the NNA until the arrival of next input. As such, the nonlinear transformation for one input sample is accomplished.

Delay-Feedback Loop with Integrate-and-Fire Neurons

With the embedded delay characteristic, the dynamic behaviors of a system change accordingly as the delay varies. Such a property can be realized by a dynamic DFL, which is built of multiple integrate-and-fire (IF) neurons, as depicted in Fig. 3.6.

During the operation, the membrane capacitor, C_m, continuously tracks the variations of current that is generated from the delay calibration module and charges up its potential. When the voltage potential across the membrane capacitor, V_{th}, exceeds the firing threshold of input transistor, M_1, two cascading inverters, M_3 & M_4 and M_7 & M_8, fire a spike as output. In the meantime, the positive feedback loop, M_5 & M_6, induces a high voltage at V_{reset}, enabling the resetting transistor, $M_1 1$, to fully discharge the membrane capacitor. As such, the firing process for one output spike is accomplished.

In a typical IF neuron, the firing time, τ_{fire}, depends on the total integration time over the membrane capacitor, which can be controlled by the current conducted from the delay calibration module, I_{cal}. Such a property can be denoted with the similar correlation as in Eq. 3.3, which can be written as

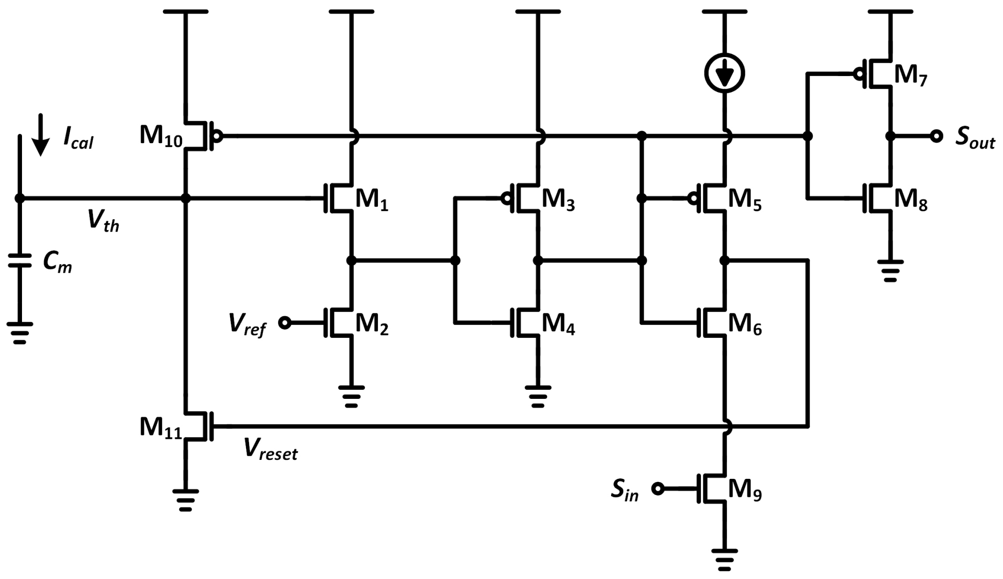


Figure 3.6: Design scheme of integrate-and-fire neuron.

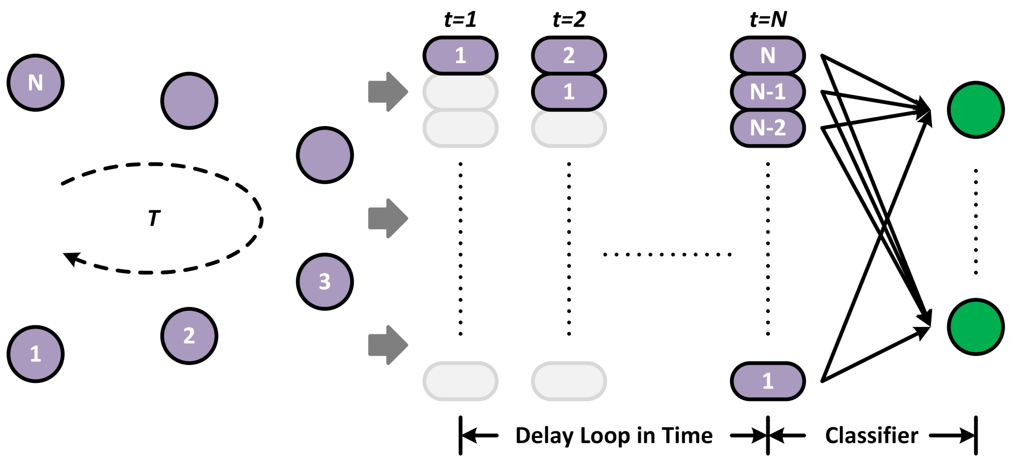


Figure 3.7: Operating principle of dynamic delay-feedback loop.

$$\tau_{delay} = C_m \cdot \frac{V_{th}}{I_{cal}}. \tag{3.6}$$

In general, the mathematical model of timing-coefficient on a delay system is represented by the product of resistance and capacitance (*e.g.*, $\tau = R \cdot C$). The equivalent input resistance of an IF neuron can be defined as $R_{in} = \frac{V_{th}}{I_{cal}}$, and thus, the firing time (also referred to the timing-coefficient) can be then simplified as

$$\tau_{delay} = C_m \cdot R_{in}. \tag{3.7}$$

Compared to the classical RC delay that is built upon a large silicon area of resistors and capacitors, the IF neuron is capable to process spiking information directly with a superior dynamic range of controllable timing-coefficient and a small silicon area.

The dynamic DFL is obtained by dividing the total timing-coefficient, T, into N equidistant IF neurons with an identical timing-coefficient, as depicted in Fig. 3.7. During the operation, the output spike train generated from a neuron is adopted as the clock triggering signal to its following. To be specific, when the temporal spike train is generated from the first neuron, N_1, it resets its following neuron, N_2. In the meantime, the voltage potential across the membrane capacitor of N_2 starts to charge up. Over the period of τ_{delay}, N_2 fires a spike as output, introducing a delay to the input spike train.

Delay Calibration

To realize a variety of dynamic behaviors with respect to various timing-coefficients, a controllable delay calibration module, as shown in Fig. 3.8, is deployed to regulate the timing-coefficient along the dynamic DFL. The delay calibration module is built based upon a voltage-to-current converter. The current mirror array, as shown in the dashed box in Fig. 3.8, keeps tracking the variations between the input, V_{cal}, and the feedback voltage, V_{fb}, and linearly generates the corresponding calibration current, I_{cal}, which can be expressed as

$$I_{cal} = (V_{cal} - V_{fb}) \cdot g_m, \tag{3.8}$$

where g_m is the trans-conductance of the operational amplifier. The current mirror array is optimized with identical design parameters, ensuring that the calibration currents for each IF neuron along the dynamic DFL are also identical.

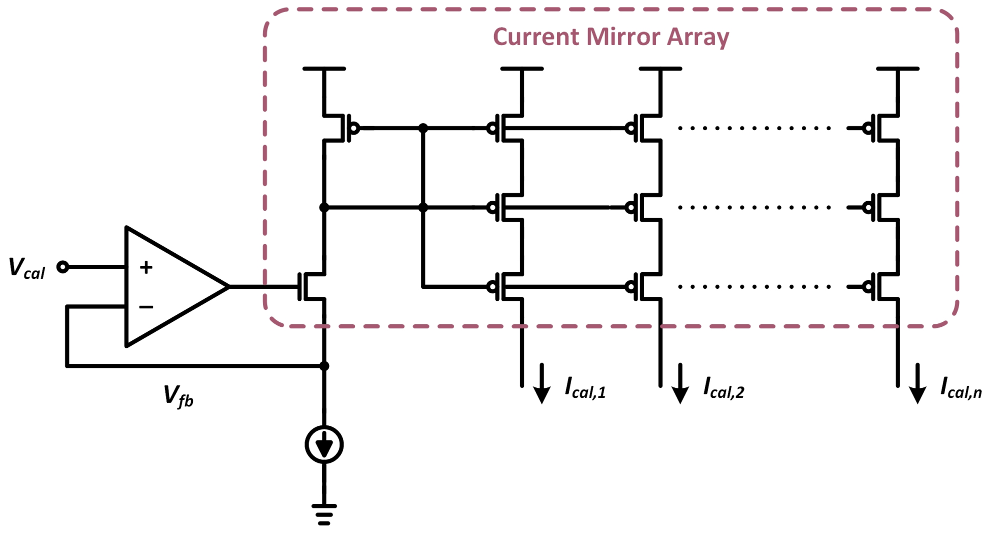


Figure 3.8: Design scheme of delay calibration module.

3.4 Performance Evaluation of Fabricated On-Silicon Prototype

A small-scale prototype of the introduced DFR network has successfully fabricated in *GlobalFoundries* 130nm BiCMOS process for basic function verification. Fig. 3.9 demonstrates the die micrograph of the fabricated DFR network prototype. Twelve DFR network prototypes implemented with four various design methodologies along with peripheries measure 2.25mm^2, where each DFR network occupies 0.0098mm^2.

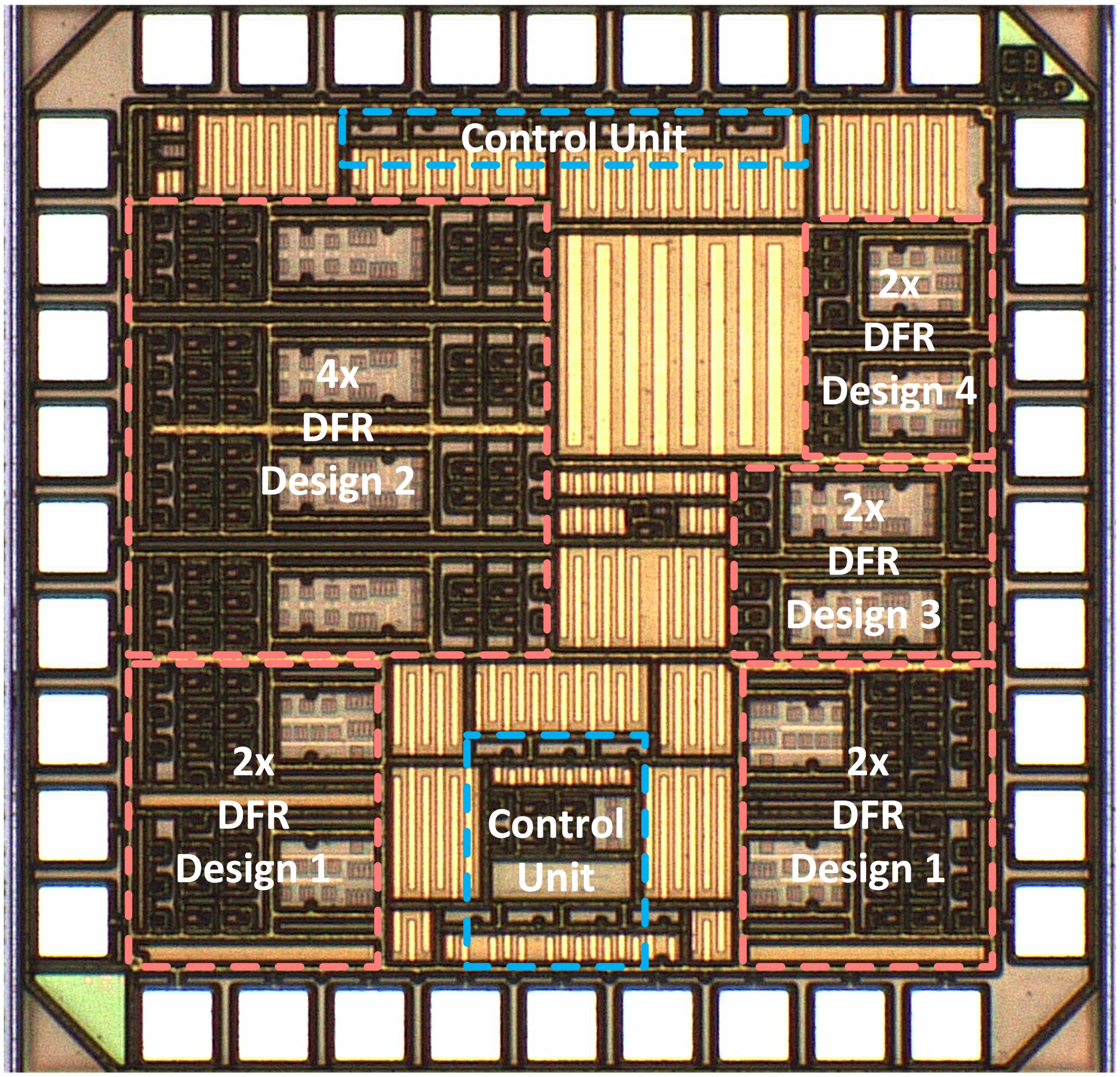


Figure 3.9: Die Micrograph of fabricated delay-feedback reservoir network prototype in 130nm BiCMOS process, occupying 2.25mm^2 silicon area.

3.4.1 Nonlinear Behavior of Sigmoid Neural Activation

The mathematical representation of a delay-feedback system can be separated into two computing modules, *i.e.*, the nonlinear transformation and delay regulation. The nonlinear response of the designed sigmoid NNA can be expressed as:

$$X_{out} = \frac{\beta}{1 + e^{\epsilon \cdot (X_{in} - \gamma)}}, \tag{3.9}$$

where β is the scaling parameter that controls the amplitude of output, ϵ is the nonlinear exponent, and γ determines the time-shift of input. By gradually increasing the input from 0–to–1.2V, the measured nonlinear behavior of sigmoid NNA is plotted in Fig. 3.10 together with the ideal fit. It can be observed that the measured nonlinear behavior fits the ideal function with design parameters of $\beta = 1$, $\epsilon = 12$ and $\gamma = 0.75$.

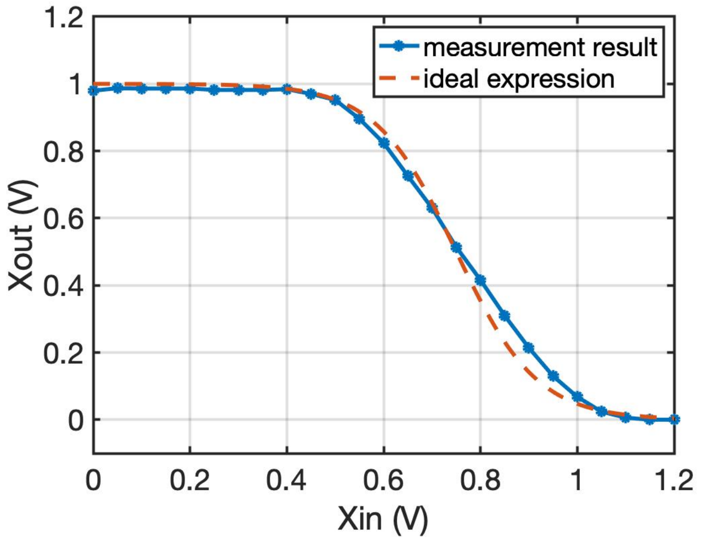


Figure 3.10: Measured nonlinear behavior of sigmoid neural activation together with the ideal fit.

In the circuit implementation, the robustness of sigmoid NNA can be improved by reducing the nonlinear effect introduced from transistors because of the channel length modulation, which can be defined as

$$error = \frac{|\Delta L|}{L_{ideal}} \cdot 100\%, \tag{3.10}$$

where L_{ideal} is the desired channel length of transistors and $\Delta L = L_{ideal} - L_{actual}$ is the difference due to the random process variations.

3.4.2 Delay Regulation

The dynamic range of delay characteristic of an IF neuron is depicted in Fig. 3.11. It can be observed that the IF neuron has a superior dynamic range of timing-coefficient from 20ns to 1.38μs. To acquire a 1.38μs delay time using the classical RC delay, such a circuit requires a 100kΩ resistor and a 13.8pF capacitor, resulting in a large silicon area while themselves are sensitive to noise. With the capacitor-sensing technique, such an identical timing-coefficient can be realized with merely 54.7nA of calibration current in the designed IF neuron with a 150fF capacitor, where the equivalent resistance $R_{on} = 10.96$MΩ. It is reasonable to conclude that the designed IF neuron is capable to process spiking information directly with superior dynamic range of controllable timing-coefficient and a small silicon area.

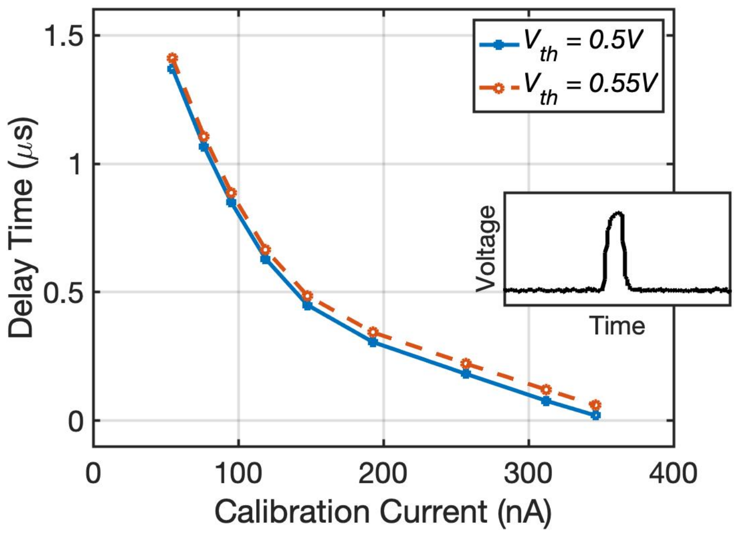


Figure 3.11: Measured delay characteristic of dynamic delay-feedback loop and the demonstration of an analog spike.

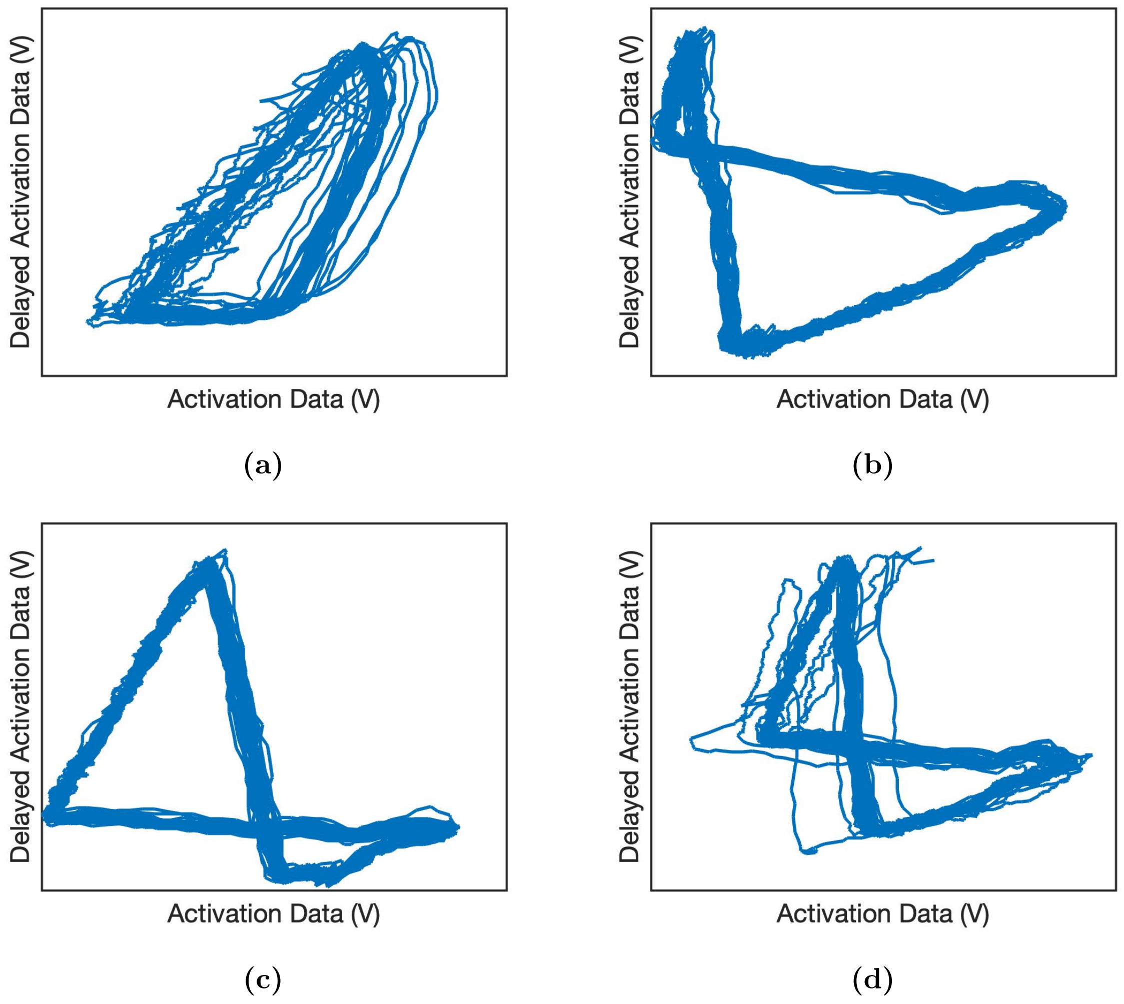


Figure 3.12: Measured dynamic behaviors of reservoir layer with (a) $T = 0.64\mu$s, (b) $T = 1\mu$s, (c) $T = 1.2\mu$s, and (d) $T = 1.4\mu$s.

3.4.3 Dynamic Behavior

The phase portrait is a graphical tool to visualize how the solutions of a given delay-feedback system would behavior in a long run. As demonstrated in Fig. 3.12, plotted phase portraits were obtained from measurements using two activation data within the dynamic reservoir layer, in which one of them was collected with time delay. By varying the total timing-coefficient, T, along the dynamic DFL, the dynamic behavior of DFR network changes accordingly. As depicted in Fig. 3.12b, the delayed activation data repeatedly traces its initial path even in a long run when the total timing-coefficient is set to be 1μs, resulting as in the periodic regime. As the timing-coefficient increases (*e.g.*, 1.4μs), the delayed activation data diverges from its initial path but without off-tracking from the equilibrium point even in a long run, resulting as in the edge-of-chaos regime, as depicted in Fig. 3.12d. It can be observed that the introduced DFR network indeed goes through a range of dynamic behaviors with respect to various timing-coefficients. It is reasonable to conclude that the fabricated DFR network prototype successfully implements the desired functionality of a delay-feedback system and enables richness of dynamic behaviors.

3.4.4 Power Analysis

Fig. 3.13 demonstrates the power distribution of dynamic reservoir layer in the fabricated DFR network prototype, where the total power consumption of 529μW with a supply voltage of 1.2V was reported when the operating frequency was set to be 1MHz. The sigmoid NNA occupies 36% of total power, the A/S encoder and the dynamic DFL occupy 7% and 20% of total power, respectively, and the rest are occupied by the S/A decoder. Design specifications of the fabricated DFR network prototype and the comparison to the state-of-the-art TDR

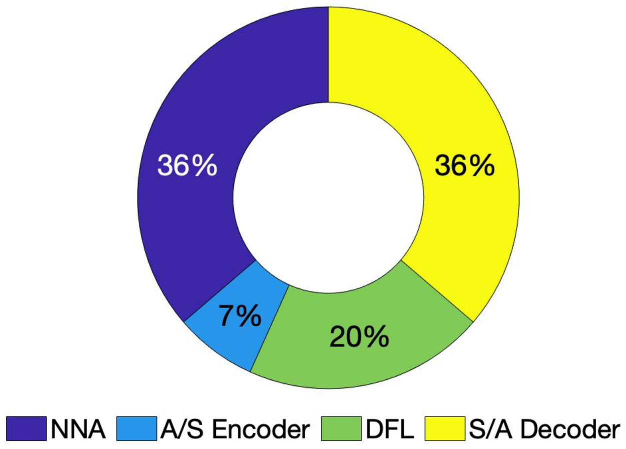


Figure 3.13: Power distribution of fabricated dynamic reservoir prototype in silicon @ 529μW with an operating frequency of 1MHz.

Table 3.1: Design specifications of fabricated delay-feedback reservoir network and comparison to the state-of-the-art time-delay reservoir network designs.

	[137]	[88]	[125]	This Work
Implementation Strategy	PCB	FPGA	FPGA	Analog IC
Technology	–	–	–	130nm
Die Area	–	–	–	2.25mm^2
Core Structure	–	ADC/DAC	ADC/DAC	neuron
Neuron Type	–	–	–	IF
Neural Activation	Mackey-Glass	Mackey-Glass	Mackey-Glass	sigmoid
Learning Capability	no	yes	yes	no
Supply Voltage	±10V	1.8–to–2.5V	1.8–to–2.5V	1.2V
Power Consumption	–	–	–	529μW

network designs are summarized in Table 3.1. This fabricated prototype demonstrates a new design solution for TDR network and, to the best of our knowledge, is the first hardware implementation of DFR network with analog IC design technique.

3.5 Application Evaluation of Delay-Feedback Reservoir Network

The fabricated small-scale DFR network prototype is mainly used for basic function verification, demonstrating how the reservoir dynamic would behave, and yet, real-world applications are not supported. To further demonstrate the performance and the reliability of the introduced DFR network, in this section, a mathematical model was implemented in MATLAB, and its accuracy was evaluated through a time series prediction benchmark and a facial recognition through video frames. In these experiments, the performance was evaluated through the 4-core Intel i7-6700 CPU and 16G RAM.

3.5.1 Time Series Prediction

The performance was initially evaluated through a time series prediction benchmark, the tenth-order nonlinear autoregressive moving average system (NARMA10) [138], which is a well-known statistical model to evaluate the performance of a dynamic system. The general expression of NARMA10 can be written as

$$y_t = 0.3 \cdot y_{t-1} + 0.05 \cdot y_{t-1} \cdot \sum_{i=0}^{9} y(t-i) + 1.5 \cdot x_{t-9} \cdot x_t + 0.1, \tag{3.11}$$

where x_t is a random input. In this task, a total of 10,000 sampling points were generated, in which 100 samples were adopted for initialization, 5,900 samples were adopted for training and the rest were adopted for testing. The prediction error was then examined using the normalized mean square error (NMSE), which can be calculated as

$$NMSE = (\frac{norm|y_i - \hat{y}_i|}{norm|y_i|})^2, \tag{3.12}$$

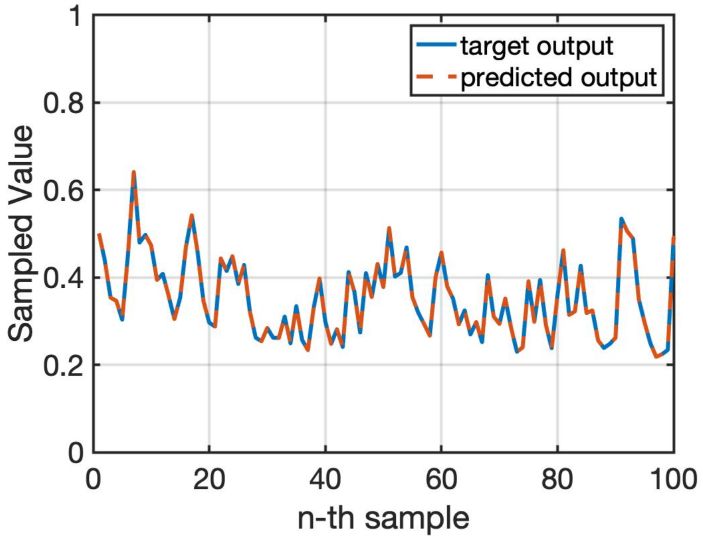


Figure 3.14: Prediction results on tenth-order nonlinear autoregressive moving average system benchmark.

Table 3.2: Prediction error comparison to the state-of-the-art reservoir computing network models on time series prediction benchmark.

	Algorithm	# of Neurons	Training Error (NMSE)	Prediction Error (NMSE)
[139]	TDR	1,990	0.065	0.464
[140]	TDR	200	–	0.17
[88]	TDR	400	–	0.15
[141]	ESN	100	–	0.1075
This Work	DFR	100	0.0849	0.0683

where y_i and $\hat{y}_i$ are target output and predicted output from the network, respectively. Experimental results of predicted output against target output are plotted in Fig. 3.14, and the comparison of prediction error to the state-of-the-art RCN models is summarized in Table 3.2. It can be observed that the prediction error from the introduced DFR network exhibits 1.57–to–6.79× reduction over the state-of-the-art RCN models.

3.5.2 Facial Recognition

The performance was then evaluated through a facial recognition. As the DFR network by itself is not capable to perform a classification task, a multilayer perceptron (MLP) with two hidden layers is deployed as the classifier for the introduced DFR network.

Total of 48 images, containing three different persons with multiple face angles, were drawn from the head pose image database [142], as demonstrated in Fig. 3.15a, in which 24 images were adopted for training and the rest were adopted for testing. For the training set, the horizontal angle of face rotates from 0° to 75° with an increment of 15° per rotation, while the vertical angle of face remains at 0°. For the testing set, the alteration of horizontal angles of face follow the training set but with an additional 15° applied to the vertical angle. The reliability of network was then investigated by introducing various levels of salt-and-pepper noise to the down-sampled testing set, as depicted in Fig. 3.15b.

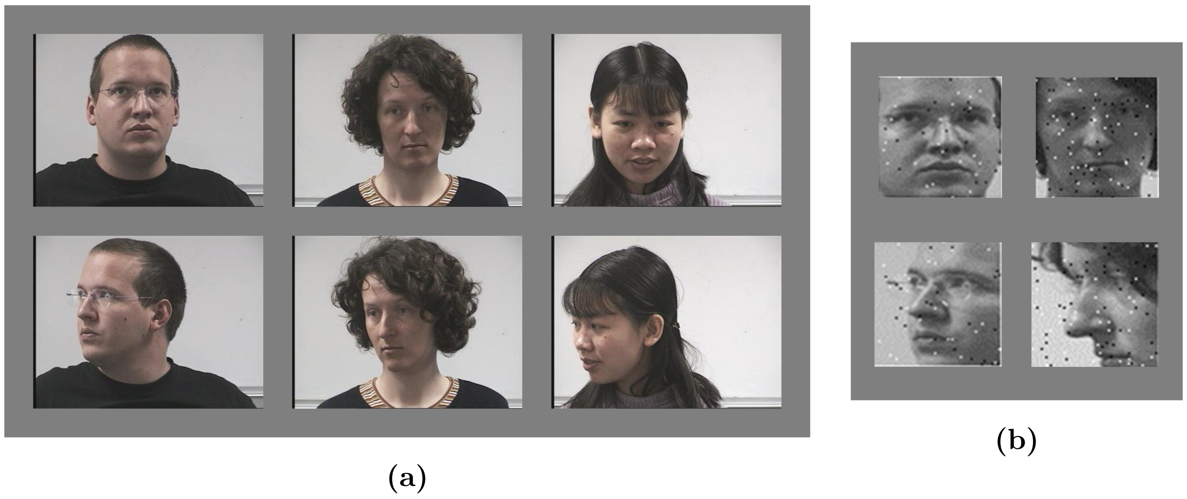


Figure 3.15: Demonstration of head pose image database separated into (a) training set of three persons and (b) down-sampled testing set with various levels of salt-and-pepper noise.

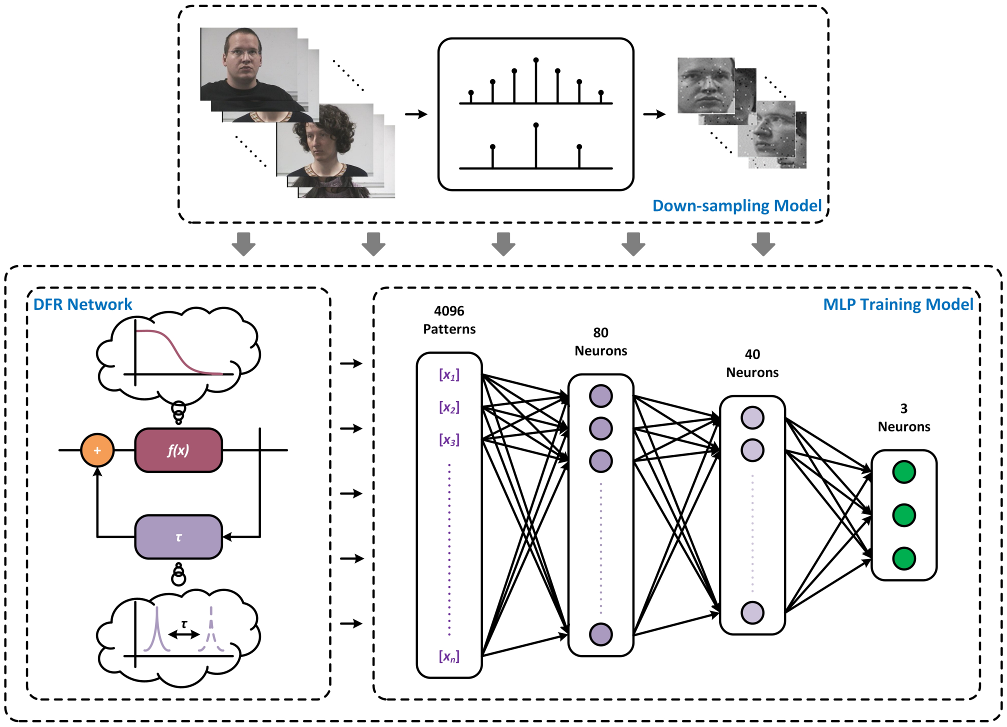


Figure 3.16: Experimental testbench for facial recognition using mutiple perceptron as classifier for the introduced delay-feedback reservoir network.

Fig. 3.16 depicts the experimental testbench for facial recognition. During the operation, each input image was firstly down-sampled into 64×64 pixels, in which each pixel was then mapped into a higher dimensional space for linear separation by the introduced DFR network. As the outcome from the dynamic reservoir layer, total of 4,096 spike trains, representing the nonlinearly transformed information of each pixel, were generated and converted back into numerical numbers for the training operation. Initially, synaptic weights in the MLP classifier were generated randomly. As each piece of data was processed in the training stage, synaptic weights were calibrated based on the corrections that minimizing the error between predicted and target outputs. To sidestep the overfitting issue, the cross-validation technique was applied.

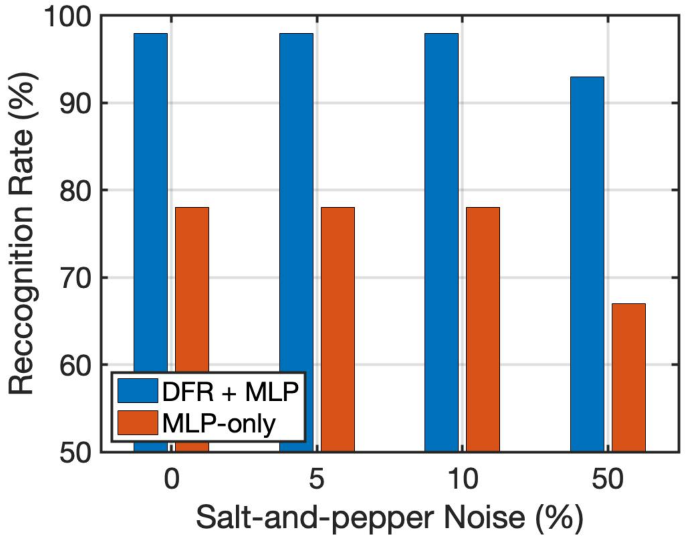


Figure 3.17: Recognition rate with respect to various levels of salt-and-pepper noise on head pose image database.

As demonstrated in Fig. 3.17, the recognition rate maintains at 98% with the introduced DFR network when the noise level is set to be less than 10%. The same trend can be found with the MLP-only training model, yielding a constant recognition rate of 78%. At the low-noise scenario, the use of DFR network exhibits much higher recognition rate than that the one with MLP-only training model. When the noise level reaches 50%, the recognition rate of DFR network decreases to 93%. However, with the MLP-only training model at 50% of noise level, the recognition rate has dropped to 67%, which is 26 percentage points poorer than the introduced DFR network.

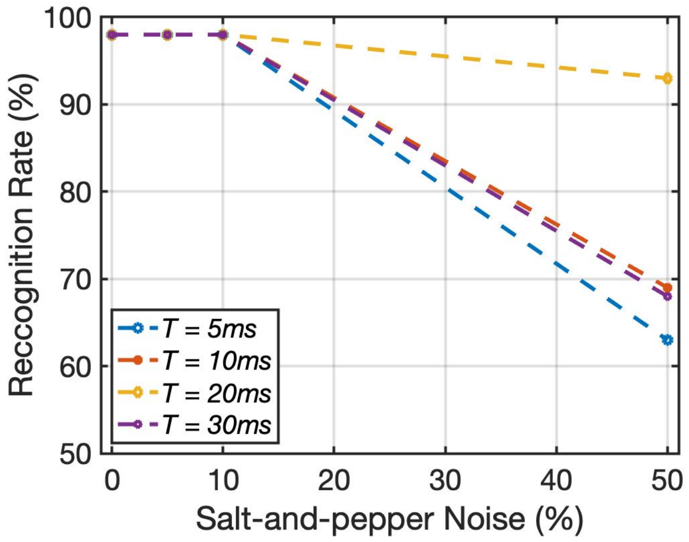


Figure 3.18: Recognition rate with respect to various timing-coefficients and levels of salt-and-pepper noise on head pose image database.

As the delay plays an important role in a delay-feedback system, in this experiment, the recognition rate was also evaluated with respect to various timing-coefficients, as plotted in Fig. 3.18. It can be observed that when the timing-coefficient is set to be 20ms, the recognition rate remains at 98% with a noise level below 10%. As the noise level approaches to 50%, the recognition rate remains above 93%. On the other hand, if the timing-coefficient deviates from 20ms, the recognition rate is significantly impacted by the variation of noise, resulting in approximately 26 percentage points poorer in the recognition rate.

3.6 Conclusions

In this project, a novel design methodology for DFR network is demonstrated together with a small-scale prototype fabrication. By mimicking dynamic aspects of the neural information processing, the introduced DFR network is built of a sigmoid NNA for high dimensional mapping and a dynamic DFL with an ISI temporal encoder to enable the spiking information processing. With a controllable timing-coefficient, the introduced DFR network exhibits richness in dynamic behaviors, signaling the successful implementation of delay-feedback system.

A 130nm prototype is fabricated for basic function verification with a power consumption of 529μW at an operating frequency of 1MHz. Moreover, numerical evaluations demonstrate that the introduced DFR network offers up to 6.79$\times$ error reduction on NARMA10 benchmark over the state-of-the-art RCN models. What is more, the introduced DFR network offers 26 percentage points more robust against noise in the facial recognition compared to the classical MLP model. This project demonstrates a new design solution for TDR network and, to the best of our knowledge, is the first hardware implementation of DFR network with analog IC design technique.

Chapter 4

Convolution-Immersed Delay-Feedback Reservoir Network with Memristive Synapses

4.1 Introduction

The continue success in deep learning has immensely pushed the artificial intelligence (AI) forward. Deep neural networks (DNNs) are what underpins deep learning, introducing a neural computing strategy to accelerate the computational efficiency for machine intelligence applications. DNNs are generally relied on the vector-matrix and matrix-matrix multiplications, in which the vectors represent the raw sensory information and the matrices denote the synaptic weights between layers. More specifically, such computations are primarily built based upon the multiplication-and-accumulation (MAC) operation. It is obvious that the high level of complexity in machine intelligence applications requires the use of a colossal amount of memory storage, and yet, the bottleneck of nonvolatile weight storage significantly hinders the hardware realization of DNNs. To be specific, synaptic weights can be stored in capacitors or floating-gate transistors (*e.g.*, flash memory) [143]. Nevertheless, due to the leakage of charges, synaptic weights on capacitors fade in time, and thus, requiring the weight updating operation with more frequent intervals. Moreover, because of the nonlinear effect and the costly double-poly process, the feasibility when exploiting floating-gate transistors as analog memory cells in neuromorphic applications remains limited [144].

The resistive random-access memory (ReRAM), a type of memristor [145] that relies on the nonvolatile memory technology, can be leveraged in realizing reconfigurable systems [146]. Such an emerging memory device offers extremely high storage density and low power consumption [11], becoming one of the most promising candidates for the next-generation memory technology and yielding a promising perspective toward energy-efficient neural computations. With the intrinsic parallel computing nature in a crossbar, ReRAM is widely adopted to develop energy- and area-efficient synaptic arrays for DNNs [147], enabling a new era for brain-inspired information processing. However, similar as in the digital implementation of DNNs, the interface with raw sensory information in real-time requires power-hungry peripherals, and thus, concealing the advantages introduced by ReRAM.

My goal in project aims to adopt the emerging memristor devices as electronic synapses to enable a new computing solution for delay-feedback reservoir (DFR) network. The insight of my approach is to introduce a convolutional layer with memristive crossbar as a feature extractor for the previously designed DFR network, improving the network's learning capability. Major contributions of this project are summarized as follows:

- A novel design solution to improve the learning capability for DFR network by integrating convolutional layers and a dynamic reservoir layer in a processing pipeline.
- A novel Mackey-Glass nonlinear neural activation (NNA) with delay-dynamic property in nature to enable the efficient neural computations.
- A prototype built on a printed circuit board with discrete ReRAM cells, yielding an average power consumption of 1.05mW.
- Up to 13.77× error reduction over the state-of-the-art reservoir computing network (RCN) models on time series prediction benchmarks.
- A classification accuracy ups to 99.03% and 99.63% on handwritten digits and spoken digits, respectively, exhibiting a higher classification accuracy over the one with classical NNAs.

The rest of this chapter is organized as follows: Section 4.2 provides an overview of emerging memristor devices. The design methodology of the introduced convolution-immersed DFR network and the performance evaluations of the implemented prototype are discussed in Section 4.3 and Section 4.4, respectively, followed by the software-based experimental evaluations in Section 4.5. This chapter is then concluded in Section 4.6.

4.2 Resistive Memory and Crossbar Array

Due to the needs of a colossal amount of memory storage for neuromorphic applications, emerging nonvolatile memory technologies are hereby investigated for seeking an alternative solution to expand the storage density and to reduce the power consumption. Examples include phase change RAM (PCRAM) [148–153], ferroelectric RAM (FeRAM) [154–159], spin-transfer torque RAM (STT-RAM) [160–165], and ReRAM [145,166–170]. Among these emerging technologies, ReRAM is considered as a promising candidate for memory storage used in neuromorphic applications because of its extremely high storage density and low power consumption.

The resistive switching effect is the basic principle in ReRAM, describing a hysteretic switching phenomenon in a silicon oxide thin film [171]. The concept of memristor was then developed by Dr. Leon Chua in 1970s, introducing the fourth fundamental circuit element in relation to the magnetic flux and charge [145]. Until 2008, HP Labs fabricates a prototype of ReRAM with analogue resistive states, approving the extensive of memristor [172]. In general, ReRAM is a two-terminal metal-oxide-based nanoscale device, performing the same functionality as a variable resistor with the nonvolatility characteristic. ReRAM is usually formed based on the transition metal oxides such as titanium dioxide (TiO_2) [173], hafnium dioxide (HfO_2) [174], etc. Due to the formation of conductive filaments in the insulator material between two terminals, the resistance of ReRAM can be switched from its high-resistance-state (HRS) to low-resistance-state (LRS) when the stimulus across the device excesses a specific threshold, as demonstrated in Fig. 4.1.

With the appearance of emerging memory technologies, a novel design solution for neuromorphic architecture are widely investigated [175]. It can be observed that the vector-matrix

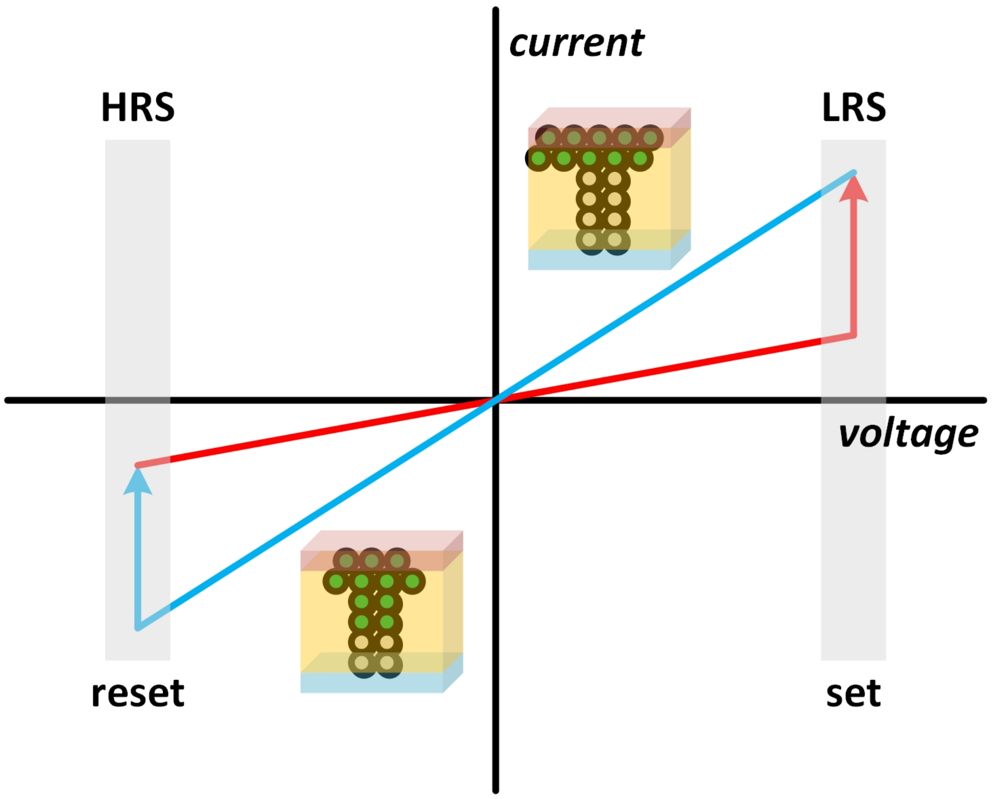


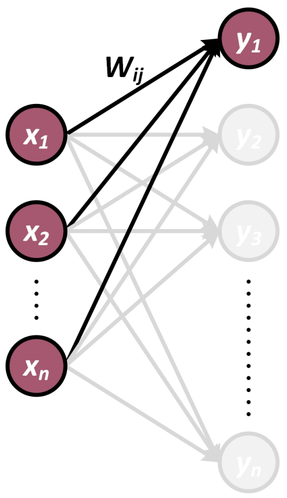
Figure 4.1: Resistive switching behavior of resistive random-access memory cell.

and matrix-matrix multiplications are the fundamental operator in neural computations that determines the accuracy, power consumption and operational speed. A simple neural computation with synaptic weights is illustrated in Fig. 4.2a. The output state, $\hat{y}$, can be expressed as the sum-of-product of inputs and a weight matrix

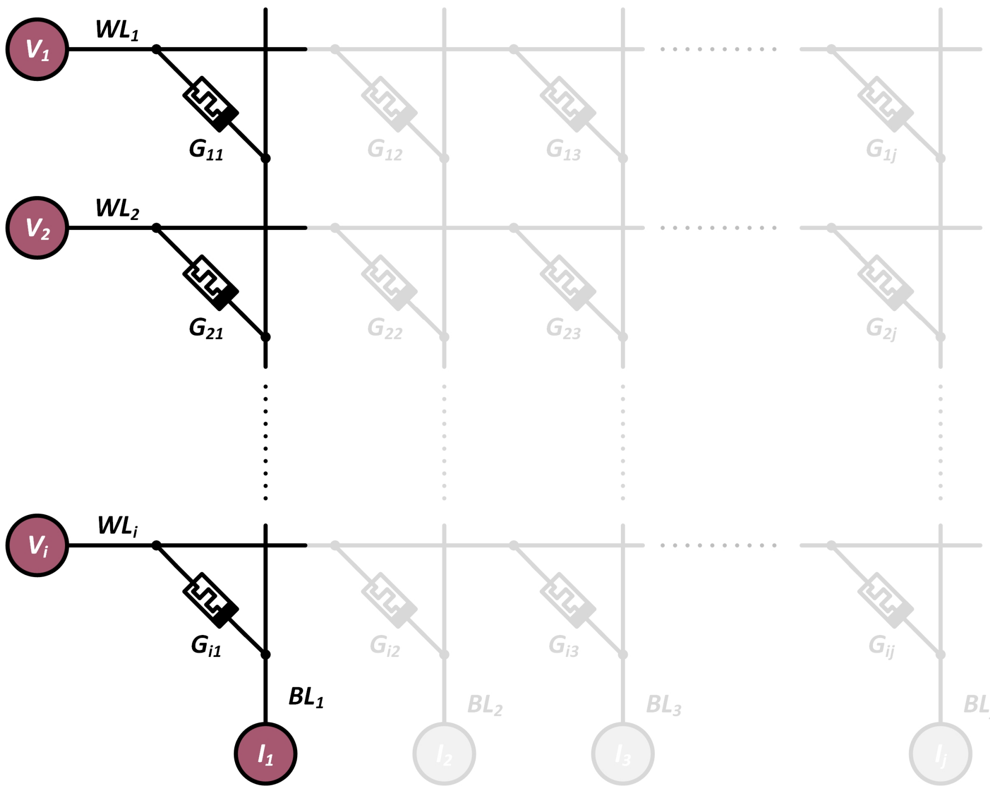
$$\hat{y}_j = \sum_{i=1}^{m} x_i \cdot W_{ij}, \tag{4.1}$$

where x_i is input vectors, m defines the data length of inputs, and W_{ij} represents the weighted value located at the i-th input and the j-th output.

The crossbar structure with current-based ReRAM cells can frankly implement such a vector-matrix multiplication as required in neural computations, as demonstrated in Fig. 4.2b. By mapping input vectors to voltages and the weight matrix to a crossbar, the vector-matrix multiplication can be realized by sampling the total output current, I_j, on each bit-line,



(a)



(b)

Figure 4.2: Representation of vector-matrix multiplication in (a) artificial neural network and (b) resistive crossbar.

which can be written as

$$I_j = \sum_{i=1}^{m} V_i \cdot G_{ij}, \tag{4.2}$$

where V_i is input voltages and G_{ij} defines the conductance of a ReRAM cell located at the i-th word-line (WL) and the j-th bit-line (BL).

The ReRAM is naturally adept in the historical behavior [176]. Firstly, the ReRAM-based crossbar supports a numerous amount of signal connections within a small silicon area, mimicking the synaptic connections in neurological system. Secondly, the ReRAM-based crossbar inherently provides the vector-matrix multiplication with the intrinsic parallel computing nature, imitating the operation of dendrite potential [177]. Benefited by the crossbar structure, the ReRAM cell has become a promising candidate to develop energy- and area-efficient neural networks. For instance, in [178], a convolutional neural network (CNN) accelerator is introduced with a 128×128 ReRAM-based crossbar, yielding improvements of $14.8\times$ and $5.5\times$ in throughput and energy, respectively, compared to the von Neumann supercomputer. Moreover, in [179], a neuromorphic hardware with STT-RAM obtains more than 15–to–300$\times$ energy reduction over the state-of-the-art CMOS design.

4.3 Design Strategy of Convolution-Immersed Delay-Feedback Reservoir Network

4.3.1 Network Architecture

Fig. 4.3 demonstrates the general architecture of the introduced convolution-immersed DFR (Ci-DFR) network, in which the input layer is built of a reconfigurable convolutional net (ConvNet). To enable efficient data processing and training operations with the historical learning capability, the previously designed dynamic reservoir [27,180–182] is deployed. During the operation, each data sample of a given input pattern is represented by an analog voltage, and weighted values in the convolution kernels are corresponded to the conductance of ReRAM cells within the crossbar. For every computing cycle, all data samples of an input pattern are applied into the crossbar in parallel. The output current, I_j, at the j-th BL can be then calculated by Eq. 4.2. At the end of the convolutional layer, $K \times [P, Q]$ data points, representing the extracted features of one input pattern, are generated, where P and Q define the dimension of the generated feature map, and K represents the number of convolution kernels.

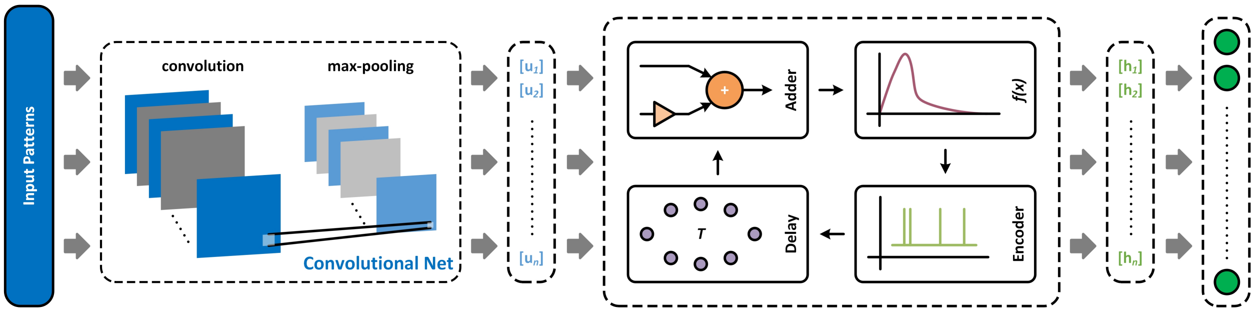


Figure 4.3: System-level architecture of convolution-immersed delay-feedback reservoir network.

The spatial dimension of generated feature maps is then reduced through the max-pooling layer. In practice, depending on the complexity of applications, more ConvNets can be deployed for the feature extraction. The final outcomes from the max-pooling layer are then reshaped into an one-dimensional (1D) column vectors and process through the dynamic reservoir layer. Consequently, the corresponding number of spike trains, represented the equivalent information of the nonlinearly transferred features, are generated. These spike data patterns are then decoded back into analog signals for the training operation through the Tikhonov regularization. The general learning operation of the introduced Ci-DFR network is summarised in Algorithm 2

4.3.2 Convolutional Net with Current Sensing Methodology

Fig. 4.4 depicts how to deploy the feature extraction on a ReRAM-based crossbar. In the ConvNet, the convolution kernels are typically formed by a $[3, 3]$ or a $[5, 5]$ two-dimensional (2D) matrix. To implement the convolution kernels on a ReRAM-based crossbar, kernels are unrolled into multiple 1D column vectors, and thus, K_j represents the j-th kernel is

Algorithm 2: Convolution-Immersed Delay-Feedback Reservoir Network

```
Data: $x = [n, m], y = [n, u], kernel = k, scaling = \alpha$
Result: $W_{out}$
initialization
for $i \to 1$ to $n$ do
    for $ii \to 1$ to $k$ do
        $conv = convolution(m)$
        $fact = f_{MG}(conv)$
        $pool = max.pooling(fact)$
    end
    return $pool$
    $flat = flatten(pool)$
    for $j \to 1$ to $len(flat)$ do
        $h_t = f((1 - \alpha) \cdot flat_t + \alpha \cdot h_{t-1} + b_h)$
    end
end
return $h_t$
$S = [h_0, h_1, \cdots, h_n]$
$Y = [y_0, y_1, \cdots, y_n]$
$W_{out} = Y \cdot S' \cdot (S \cdot S' + \eta \cdot I)^{-1}$
```

mapped into the j-th column in the crossbar. With the intrinsic parallel computing nature, receptive input voltages, representing information of an input pattern, are applied to each WL in parallel. Therefore, multiple features, in a format of analog current, can be generated through each BL synchronously, which can be calculated by Eq. 4.2.

In general, a specific weighted value located at the i-th row and the j-th column satisfies the constraints of

$$W_{min} \leq W_{ij} \leq W_{max}, \tag{4.3}$$

where W_{min} and W_{max} denote the minimum and maximum weighted values, respectively. In the meantime, the conductance of a ReRAM cell located at the i-th WL and the j-th BL

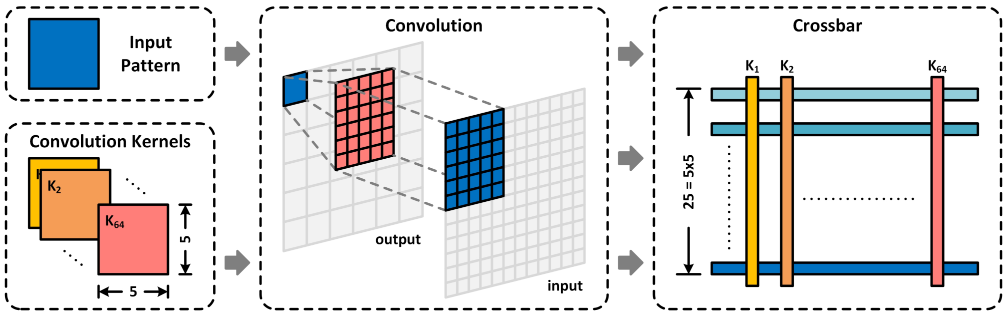


Figure 4.4: Deploying the convolution kernels on a resistive crossbar.

satisfies the constraints of

$$G_{off} \leq G_{ij} \leq G_{on}, \tag{4.4}$$

where G_{off} and $G_[on]$ denote the off and on conductance of a ReRAM cell, respectively, thereby, the expression to map a weighted value into the conductance of a ReRAM cell can be written as

$$G_{ij} = \frac{W_{ij} - W_{min}}{W_{max} - W_{min}} \cdot (G_{on} - G_{off}) + G_{off}. \tag{4.5}$$

It has been proven that the computation with a crossbar structure has several orders of magnitudes more energy- and area-efficient over the central processing unit (CPU) and graphics processing unit (GPU) [183]. In practice, extracting valuable information from crossbar requires a sensing amplifier, and thus, the voltage sensing methodology is widely adopted due to its ease of conception. In [184], a voltage sensing amplifier is implemented by converting each BL current through a fixed resistor and an operational amplifier. However, the use of fixed resistor severely degrades the accuracy due to its sensitivity to noise, and the output headroom of sensing amplifier is limited in the analog integrated circuit (IC) design.

In this project, a linear current amplifier with the inlaid current-to-voltage converter is introduced, as shown in Fig. 4.5. Such a linear current amplifier isolates the sum-of-product computation in the crossbar and the current-to-voltage conversion in the current amplifier, and therefore, the computational results in crossbar will not be distorted. During the

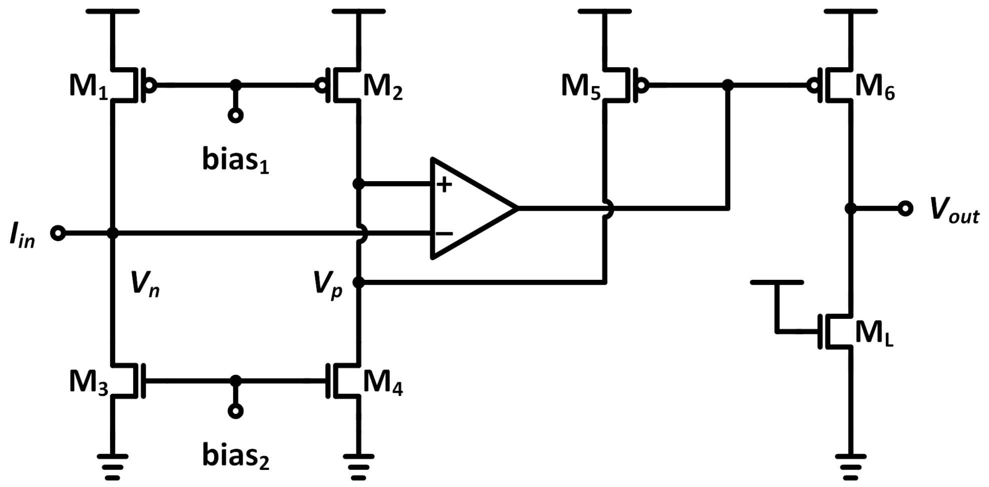


Figure 4.5: Design scheme of voltage-model current sensing amplifier.

operation, the input current, I_{in}, along with the bias current from the transistor M_1 are collected by the transistor M_3 and mirrored to the transistor M_4 with a 1:1 ratio, and thus, $I_{M4} = I_{M3} = I_{in} + I_{M1}$. An operational amplifier with transistor M_5 creates a negative feedback, allowing the voltage of V_p to track the variation of the negative input, V_n. By clamping the voltage of V_p to the same level of V_n, the current through the transistor M_5 can be then calculated as $I_{M5} = I_{M4} - I_{M2} = I_{in}$. The output current mirror, M_5 & M_6, duplicates the input current to transistor M_6 with a 1:10 ratio and consistently converts into a voltage signal through a loading transistor, M_L. By isolating the sum-of-product computation in the crossbar and the current-to-voltage conversion in the current amplifier, the introduced linear current amplifier maintains the linearity and the stability during the current-to-voltage conversion.

4.3.3 Mackey-Glass Nonlinear Neural Activation

The rectified linear unit (ReLU) function is the most commonly used NNA in CNNs. However, the ReLU function does not have the timing-coefficient in nature to emulate the dynamic behaviors of neurological systems. In recent years, the Mackey-Glass (MG) function has become a decent candidate for delay-feedback system designs [185]. Originating from the ordinary differential equation, the MG function defines a system in which dynamics depend on both current and previous states, where its nonlinear response can be expressed as

$$X_{out} = \frac{\beta \cdot X_{in}}{1 + \tau^{\epsilon} \cdot X_{in}^{\epsilon}}, \tag{4.6}$$

where β is the arbitrary design parameters that define the scaling factor of the system, ϵ and τ are the nonlinear and delay exponents, respectively.

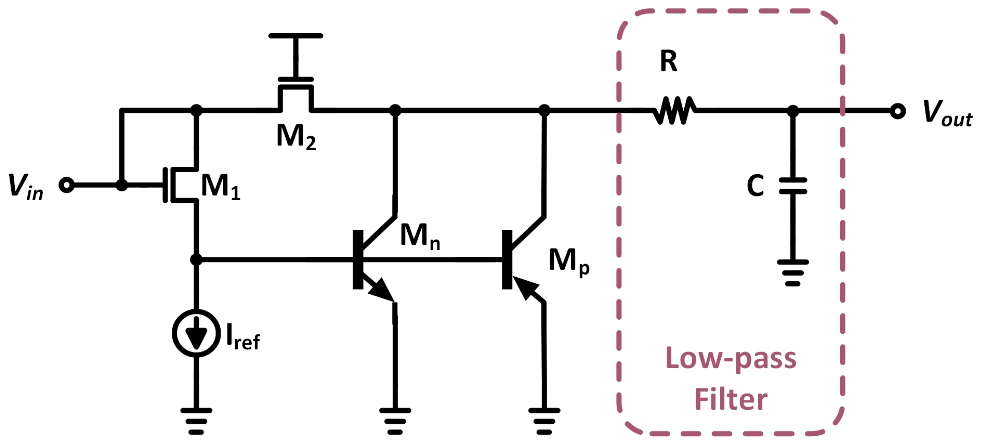


Figure 4.6: Design scheme of Mackey-Glass nonlinear neural activation.

Fig. 4.6 depicts the electronic circuit model of MG NNA. In general, the nonlinear behavior of MG function can be realized by controlling the switching conditional of a n-type switch, M_n, and a p-type switch, M_p. During the operation, M_p fully turns on to reduce charges from the low-pass filter, as shown in the dashed box in Fig. 4.6, under the condition of $V_{in} < V_{th,p}$, where $V_{th,p}$ is the threshold voltage of M_p; as such, the voltage across the low-pass filter remains at 0V. Under the condition of $V_{th,p} < V_{in} < V_{th,n}$, where $V_{th,n}$ is the threshold voltage of M_n, charges from the input source are accumulated in the low-pass filter, and thus, the voltage across the low-pass filter follows the input voltage. When $V_{in} > V_{th,n}$, M_n fully turns on to reduce charges from the low-pass filter again, such that the voltage across the low-pass filter is discharged to 0V. To accurately model the explicit representation of MG function, the transistor M_2 is implemented to serve as the scaling parameter in the circuit point of view. The nonlinearity can be turned by the aspect ratio of M_n and M_p, while the delay exponent can be adjusted by the reference current source, I_{ref}.

The MG function triumphs the ReLU function over its intrinsic delay characteristic, naturally suitable for the delay-feedback system design.

4.4 Performance Evaluation of Fabricated On-Board Prototype

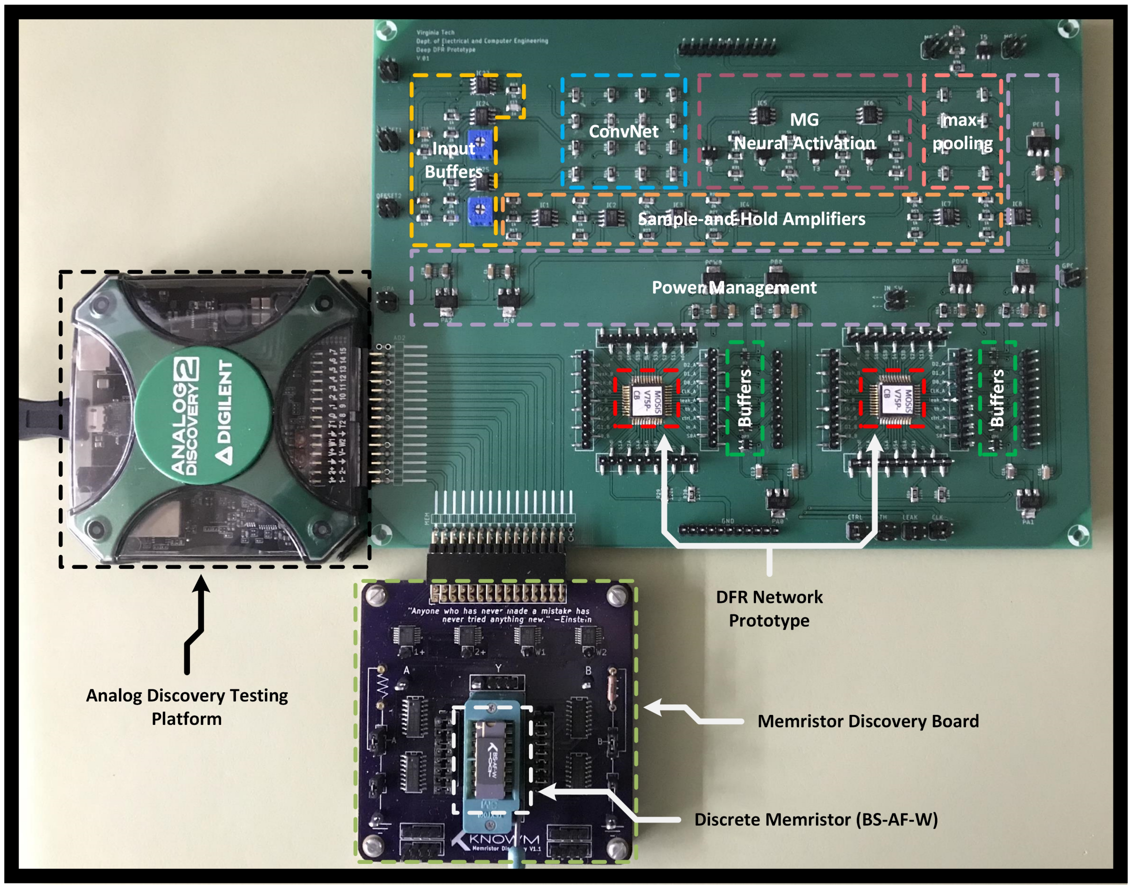
A prototype of the introduced Ci-DFR network has successfully built on a printed circuit board (PCB) for basic function verification, as depicted in Fig. 4.7. To demonstrate the design possibility of deploying ReRAM cells as synaptic weights and to evaluate the computing capabilities of feature extraction, the prototype is built with both off-chip resistive and memristive crossbars, sample-and-hold (SH) amplifiers, MG NNA, and the previously fabricated on-chip DFR network.

To realize the operation of feature extraction, a 4×4 off-chip resistive crossbar is deployed on the prototype. Synaptic weights were randomly implemented, whereby the conductance states of each resistor were calculated by Eq. 4.5. The rate of data-flow is mainly controlled by the off-chip input buffers. To reduce the device variation of crossbar, only the binary weights were implemented, in which the HRS and the LRS of the off-chip resistive crossbar were set to be 50μS and 1μS, respectively, where the according resistance is $20\text{k}\Omega$ and $1\text{M}\Omega$. This particular resistance state is allocated within the range of measured resistance states on the discrete ReRAM cell (BS-AF-W) [186] from the Knowm Inc.

During the operation, each BL current from the ConvNet is accumulated to generate the desired voltage for latter computations. A supplementary off-chip MG NNA is then used to carry out the nonlinear transformation. The max-pooling layer fetches the maximum selected activation data to SH amplifiers, following by processing through the on-chip DFR network. The output classifier is built of a 2×2 crossbar with discrete ReRAM cells, generating the predicted outputs from the network.

4.4.1 Resistive Switching Behavior

The bipolar resistive switching behavior of discrete ReRAM cell is analyzed through its *iv* characteristic. In this measurement, a bias voltage was applied between the top and bottom electrodes of a ReRAM cell with the latter being grounded. The current compliance was set to be 10μA while a voltage range of -1–to–1V was applied. The measured *iv* characteristic



Input Buffers
ConvNet
MG Neural Activation
max-pooling
Sample-and-Hold Amplifiers
Power Management
Buffers
Buffers
DFR Network Prototype
Analog Discovery Testing Platform
Memristor Discovery Board
Discrete Memristor (BS-AF-W)

Figure 4.7: Prototype of convolutional-immersed delay-feedback reservoir network with discrete memristor device mounted on a printed circuit board.

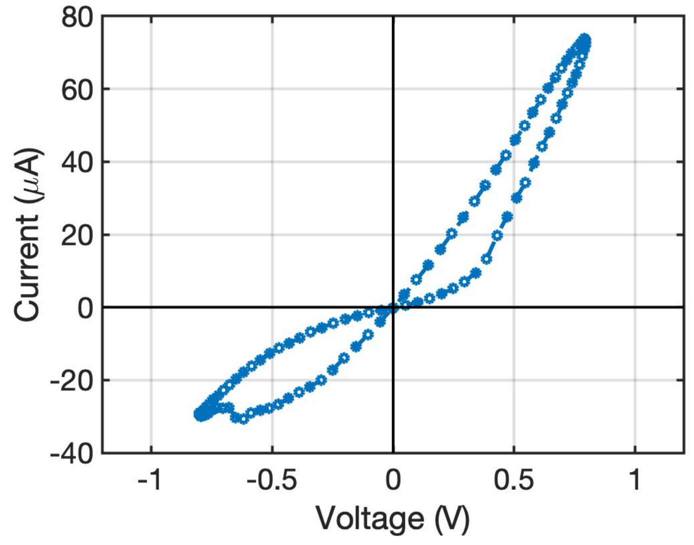


Figure 4.8: Measured bipolar resistive switching behavior of discrete memristor device (BS-AF-W).

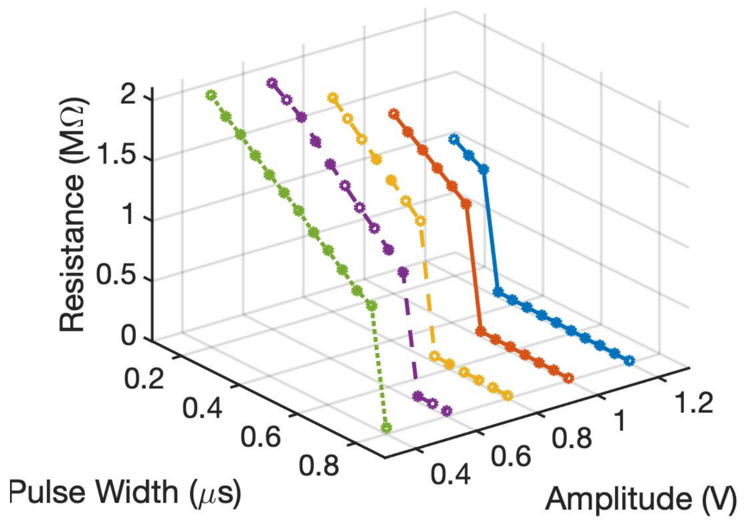


Figure 4.9: Measured resistance states of discrete memristor device (BS-AF-W) with respect to various switching pulse width and voltage amplitudes.

of the discrete ReRAM cell is depicted in Fig. 4.8. It can be observed that the current jumps abruptly at a positive voltage of 0.4V as the bias voltage increases gradually. By kept increasing the bias voltage, the ReRAM cell switches from its HRS to LRS, indicating as the “SET” operation. By contrast, at a negative voltage of −0.4V, the ReRAM cell starts switching from LRS to HRS, indicating as the “RESET” operation.

By nature, the resistance states of a ReRAM cell can be controlled by the switching voltage across the device and its switching pulse width. Fig. 4.9 demonstrates the measured resistance states of discrete ReRAM cell with respect to various switching pulse widths and voltage amplitudes. It can be observed that the tunable electrical resistance exists in the range of 15k–to–2MΩ, where the equivalent conductance ranges from 0.5μS to 67.38μS. It can be also observed that as the bias voltage increases, the ReRAM cell is more easily to be switched from its HRS to LRS even with a shorter switching pulse width.

4.4.2 Nonlinear Behavior of Mackey-Glass Neural Activation

The off-chip supplemental MG NNA is built of bipolar junction transistors (BJTs). To demonstrate the nonlinear behavior of the designed circuitry, a sequential bias voltage was applied as the input with a range of −1.8–to–1.8V. By gradually increasing the bias voltage, the nonlinear behavior of MG NNA is recorded together with the corresponding ideal fit, as plotted in Fig. 4.10. It can be observed that the designed analogue circuit of MG NNA fits the ideal MG function with the scaling parameter, nonlinear and delay exponents of $\beta = 1$, $\epsilon = 12$ and $\tau = 1$, respectively.

4.4.3 Linearity of Current Sensing Amplifier

The introduced linear current amplifier isolates the sum-of-product computation in the crossbar and the current-to-voltage conversion in the current amplifier, and thus, computational results in the crossbar cannot be distorted, and the linearity as well as the stability during the current-to-voltage conversion can be preserved. To demonstrate such a functionality, the BL current, collected from crossbar with a range of 0–to–1mA, was applied. As plotted

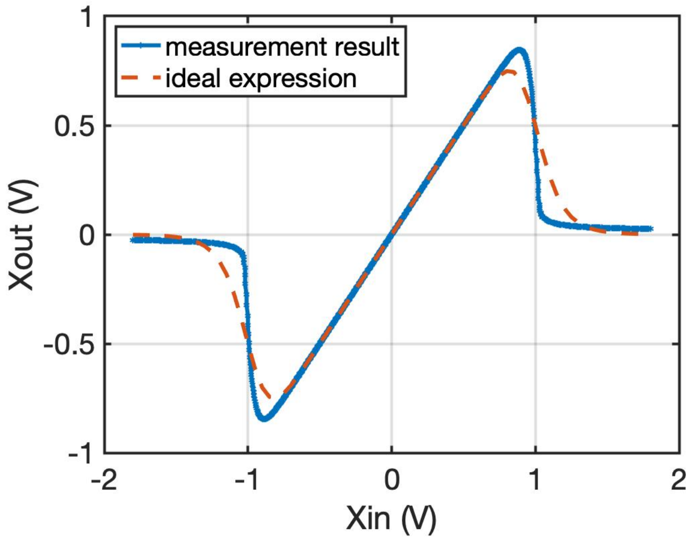


Figure 4.10: Measured nonlinear behavior of Mackey-Glass neural activation together with ideal fit.

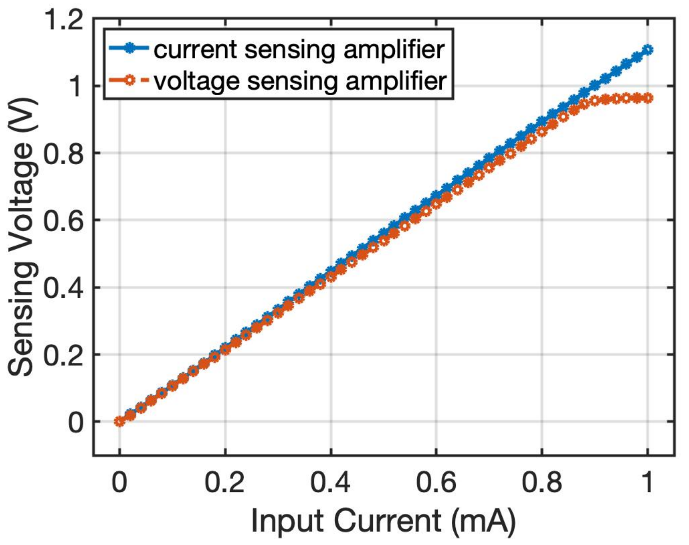


Figure 4.11: Simulated characteristic of voltage-model current sensing amplifier compared to voltage sensing amplifier.

in Fig. 4.11, it can be observed that the simulated linear correlation between BL current and output voltage can be obtained. It is reasonable to conclude that the introduced linear current amplifier is capable of providing a stable and accurate current-to-voltage conversion compared to the classical voltage sensing amplifier.

4.4.4 Power Analysis

In this experiment, the power consumption of Ci-DFR network prototype was measured on PCB with an operating frequency of 1MHz and a global supply voltage of 9V. The power distribution of Ci-DFR network prototype is demonstrated in Fig. 4.12. The overall power consumption reaches 9.45mW. Two on-chip DFR networks occupy merely 11.1% of total power with a 1.2V supply voltage, while the rest are consumed by off-chip components, in which the resistive crossbar with input buffers occupy 21% of total power, the MG NNA and SH amplifiers occupy 6% and 38% of total power, respectively, while the rest are occupied by peripheries. Design specifications of the fabricated DFR network prototype and the comparison to the state-of-the-art neuromorphic chips are summarized in Table 4.1.

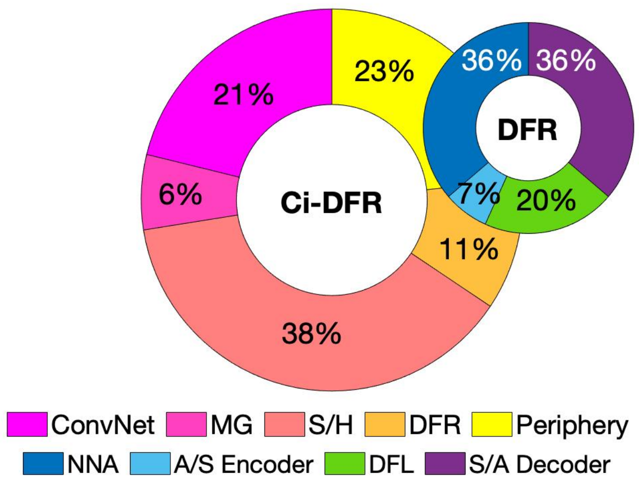


Figure 4.12: Power distribution of implemented convolution-immersed delay-feedback reservoir network prototype on a printed circuit board @ 9.45mW with an operating frequency of 1MHz.

Table 4.1: Design specifications of fabricated delay-feedback reservoir network prototype and comparison to the state-of-the-art neuromorphic chips.

	[187]	[34]	[188]	[189]	[190]	[191]	This Work
Technology	65nm	180nm	10nm	28nm	65nm	130nm	130nm
Die Area	16mm^2	51.4mm^2	1.72mm^2	4.6mm^2	1.44mm^2	–	2.25mm^2
Algorithm	CNN	SNN	SNN	CNN	SVM	AdaBoost	DFR
Methodology	digital	analog	digital	digital	digital	digital	analog
On-chip Learning	no	yes	yes	yes	yes	no	no
Neuron Type	–	IF	LIF	–	–	–	IF
Supply Voltage	1V	1.8V	0.9V	0.8V	1V	–	1.2V
Energy Metric	7.9mJ/decision	3.7pJ/spike	3.8pJ/SOP	–	42pJ/decision	46.6pJ/cycle	9.63pJ/spike
Maximum Frequency	250MHz	1kHz	506MHz	–	1GHz	50MHz	20MHz
Power Consumption	–	4mW	–	899μW	–	–	529μW

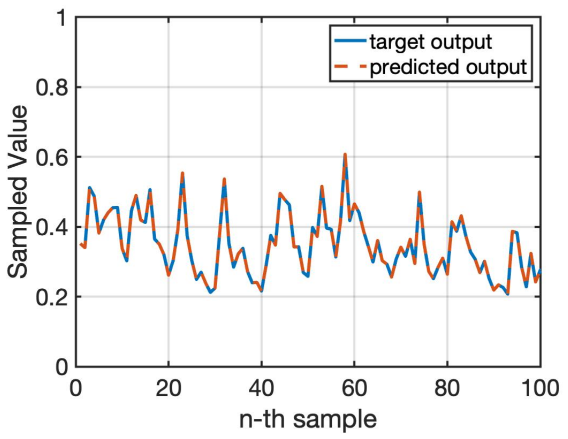
4.5 Application Evaluation of Convolution-Immersed Delay-Feedback Reservoir Network

Similar as the previously fabricated DFR network prototype in silicon, the Ci-DFR network prototype built on a PCB also does not support the computations of real-world applications. To further demonstrate the performance and the reliability of the introduced Ci-DFR network, a mathematical model was implemented in TensorFlow, and its accuracy was evaluated through three various time series prediction benchmarks, a handwritten digit classification, and a spoken digit recognition. Two convolution layer (where each has $64 \times [5, 5]$ kernels) and two max-pooling layers (where each has $[2, 2]$ pooling size) were implemented. In this experiment, the performance was evaluated through the 8-core Intel i7-9700K CPU, 16G RAM and the Nvidia RTX 2080 GPU.

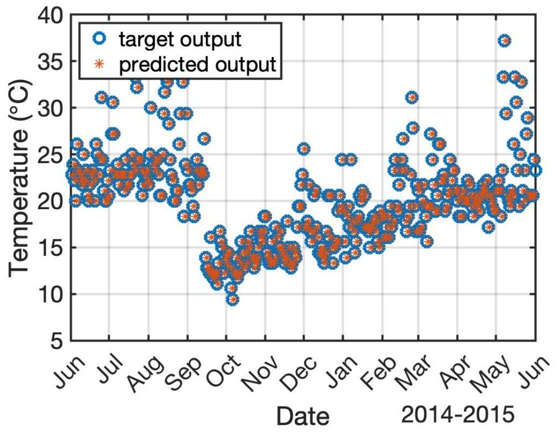
4.5.1 Time Series Prediction

In this experiment, the performance of introduced Ci-DFR network was initially evaluated through three time series prediction benchmarks. The tenth-order nonlinear auto regressive moving average system (NARMA10) benchmark [138] used in this experiment has the same configuration as the one used in the DFR network as depicted in Section 3.5. The other selected benchmark is the temperature forecasting in the city of San Francisco [192], containing the average daily temperature variation in San Francisco from January 1st, 2010 to October 8th, 2015. This dataset consists of 2,104 samples, in which 100 samples were adopted for initialization, 1,500 samples were adopted for training and the rest were adopted for testing. The last benchmark used in this experiment is the temperature forecasting during the El Nino, which refers to the cycle of warm and cold temperatures of the tropical central at South America Ocean since 1980 [193]. This dataset consists of 10,950 temperature data of sea surface, whereby 950 samples were adopted for initialization, 6,000 samples were adopted for training and the rest were adopted for testing. The prediction error was then examined using the normalized mean square error (NMSE) as defined in Eq. 3.12 and compared to the state-of-the-art DNN models.

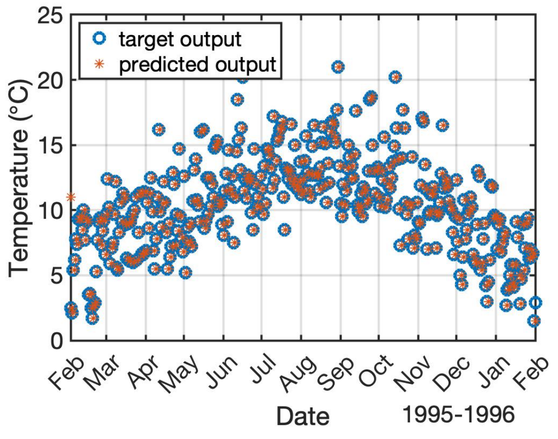
Experimental results of predicted outputs against target outputs on three various time series prediction benchmarks are plotted in Fig. 4.13. The prediction error comparison to the state-of-the-art DNN models are summarized in Table 4.2. It can be observed that the prediction error from the introduced Ci-DFR network exhibits 1.6–to–13.77× reduction over the state-of-the-art DNN models.



(a)



(b)



(c)

Figure 4.13: Experimental results of time series prediction benchmarks in (a) tenth-order nonlinear auto regressive moving average system, (b) temperature forecasting in San Francisco, and (c) temperature forecasting in South America Ocean during El Nino.

Table 4.2: Prediction error comparison to the state-of-the-art deep neural network models on time series prediction benchmarks.

Benchmark		Algorithm	# of Neurons	Prediction Error (NMSE)
NARMA10	[96]	DFR	100	0.464
	[140]	DFR	200	0.17
	[194]	Deep ESN	–	0.1572
	[195]	Deep ESN	–	0.0832
	[27]	DFR	100	0.0683
	This Work	Ci-DFR	100	0.0377
San Francisco	[196]	MLP	–	0.1009
	This Work	Ci-DFR	100	0.0873
El Nino	[197]	MLP	–	0.16
	This Work	Ci-DFR	100	0.045

4.5.2 Handwritten Digit Classification

The MNIST handwritten digit database [198] used in this experiment contains 70,000 28$\times$28-pixel bi-level images of handwritten digits, in which 60,000 samples were adopted for training and the rest were adopted for testing. Table 4.3 summarizes the classification accuracy comparison with respect to various NNAs. Experimental results demonstrate that the introduced Ci-DFR network with MG NNA exhibits 0.67–to–1.4 percentage points of improvement over alternative NNAs, and yet, the resulting training time requires approximately 19% longer. The classification accuracy comparison to the state-of-the-art DNN models is summarized in Table 4.4.

4.5.3 Spoken Digit Recognition

The spoken digit command dataset [203] used in this experiment contains 10,000 spoken digits with 997 different speakers, in which 4,000 samples were adopted for training, 800 samples adopted used for testing, and the rest were unused. The contained waveform audio (WAV) file is inherently a 1D continuous signal across time. In order to extract valuable

Table 4.3: Classification accuracy comparison with respective to various nonlinear neural activation on handwritten digit database.

NNA Type	Classification Accuracy	Training Time
MG	99.03%	1X
tanh	97.6%	0.814×
ReLU	98.37%	0.805×
Leaky ReLU	98.37%	0.816×

Table 4.4: Classification accuracy comparison to the state-of-the-art deep neural network models on handwritten digit database.

	Algorithm	Network Structure	Classification Accuracy
[32]	MLP	6× dense	98.36%
[199]	Binary MLP	3× dense	97.2%
[200]	Spiking CNN	3× convolution 3× pooling 1× dense	97.2%
[201]	Binary MLP	3× dense	95.8%
[202]	MLP	2× dense	90%
This Work	Ci-DFR	2× convolution 2× pooling 1× DFR 1× dense	99.03%

Table 4.5: Recognition accuracy comparison with respective to various nonlinear neural activation on spoken digit command dataset.

NNA Type	Recognition Accuracy	Training Time
MG	99.63%	1X
tanh	97.38%	0.586×
ReLU	97%	0.591×
Leaky ReLU	97.5%	0.657×

features from a time-dependent input pattern, a pre-processing operation was done by deploying a sampling window to sample the input pattern for a short period of time. The sampled audio information will be then converted into a mel spectrogram by calculating the strength of frequencies across a band of filters. Such a mel spectrogram can be treated as a single-channel image, and thus, features can be extracted by the introduced Ci-DFR network. Table 4.5 summarizes the recognition accuracy comparison with respect to various NNAs. Experimental results demonstrate that the introduced Ci-DFR network with MG NNA exhibits approximately 2.5 percentage points of improvement over alternative NNAs, and yet, the resulting training time requires approximately 39% longer.

4.6 Conclusions

In this work, a Ci-DFR network with embedded ReRAM-based electronic synapses is introduced, demonstrating a new design possibility for DFR network. Such a processing structure is capable of establishing connections within the context of data and improving the learning capability of the previously designed DFR network. Measurement results on a PCB prototype along with resistive and memristive synapses occupy 9.45mW of power consumption. By applying to time series prediction benchmarks, the introduced Ci-DFR network exhibits an error reduction up to 13.77×. Moreover, experimental results on handwritten digit classification and spoken digit command recognition demonstrate that the introduced Ci-DFR network with MG NNA is capable of exhibiting approximately 1.4 and 2.5 percentage points of improvement, respectively. In conclusion, experimental results have proven the correctness and the effectiveness of the introduced Ci-DFR network in sequence prediction and classification, demonstrating the potential of DFR network for exploring more sophisticated machine intelligent applications.

Chapter 5

Hybrid Neural Network with In-Memory Computing Acceleration

5.1 Introduction

In the recent artificial intelligent (AI) society, deep learning strategies are optimized by utilizing high-performance processors and big data [7–9]. However, the fact that executing the neural computations through general-purpose computing systems or cloud infrastructures significantly degrade the user experience, while themselves exposing other security issues. More importantly, the demand on computational resources and complexity has increased with the increasing demand for data density, and thus, boosting the power consumption to a higher magnitude.

Arose with the introduction of neuromorphic architecture, application-specific integrated circuit (ASIC) chips have paved the way for deep learning strategies, accelerating the computational efficiency while reducing the hardware overhead. This is made possible by executing the neural computations through layers of artificial neurons, in a similar way to humans. That is, deep neural networks (DNNs) are what underpins deep learning strategies. On one side, ASIC implementations significantly accelerate the computational efficiency of DNNs. On the other hand, DNNs tend to maximize the accuracy with increments of complexity [204]. In particular, the use of backpropagation learning algorithm utilizes some version of gradient decent for calculating weight updates, that is, the deeper the network structure, the higher the complexity, or vice versa. More importantly, the complications are amplified by the imperfections in circuit components.

The introduced delay-feedback reservoir (DFR) network, as discussed in Chapter 3, has opened up new possibilities for ASIC-based DNN implementations by eschewing the complex backpropagation learning algorithm. The DFR network is built based upon a delay-dynamic system, offering a unique training mechanism by only updating linear readout weights, and thus, sidestepping the issue introduced from the vanishing gradient [27, 180–182]. However, the learning capability of DFR network by itself is limited as the dynamic reservoir layer does not contain synaptic weights. By implementing a convolutional net (ConvNet) as the input layer of DFR network, the introduced convolution-immersed DFR (Ci-DFR) network, as discussed in Chapter 4, offers a novel design solution to improve the network's learning capability [122, 205, 206]. Nevertheless, the number of convolution kernels and layers significantly impacts the training time, inference accuracy and implementation complexity.

My goal in this project is to build a new class of hybrid neural network by integrating both spatial and temporal information processing capabilities. The insight of my approach is to adapt by integrating the dynamic reservoir as a processing layer in a multilayer perceptron (MLP). Specifically, this project focus on the optimization of network structure to minimize the implementation complexity of ASIC-based DNN designs. Major contributions of this project are summarized as follows:

- A spatial-temporal computing structure by integrating both MLP and dynamic reservoir in a processing pipeline, improving the learning capability of DFR network without significantly increasing its implementation complexity.
- An analog multiplication-and-accumulation operator with in-memory computing strategy, enabling the parallel operation while reducing the memory bandwidth overhead.
- A prototype fabrication with microcontroller verification, yielding an average on-chip classification accuracy of 86.1% on handprinted alphabet characters with merely 33mW of power consumption.

- Software-based evaluations offer up to 6.5× speedup over the cutting-edge DNN models while yielding a competitive classification accuracy.

The rest of this chapter is organized as follows: Section 5.2 discusses the design methodology of the introduced spatial-temporal hybrid neural network. The performance evaluations of the fabricated prototype and the software-based experimental evaluations are demonstrated in Section 5.3 and Section 5.4, respectively, followed by the implementation of spike-timing-dependent plasticity for potential improvement in Section 5.5. This chapter is then concluded in Section 5.6.

5.2 Design Strategy of Spatial-Temporal Hybrid Neural Network

5.2.1 Network Architecture

Hybrid neural networks (HNNs) typically concatenate two or more feedforward neural networks (FNNs) and recurrent neural networks (RNNs) in a processing pipeline to improve the network's learning capability. In this project, a computing-in-memory (CIM)-based spatial-temporal HNN (STHNN) is introduced by integrating both MLP and dynamic reservoir in a processing pipeline, making the network becomes linear separable while computing static and dynamic data in both spatial and temporal domains. Fig. 5.1 demonstrates the general architecture of CIM-based STHNN, composing of a CIM-based dense layer as feature extractor, multiple dynamic reservoirs in parallel as an internal processing layer, and a CIM-based neural classifier as the output layer. More specifically, such a STHNN utilizes the spatial characteristic in the dense layer to extract key features from raw sensory information, while utilizing the temporal characteristic in the dynamic reservoir to enable the historical learning capability. In the meantime, the STHNN also takes advantages of the unique learning mechanism introduced in the classical reservoir computing networks (RCNs) to reduce the required resources for training and to diminish the implementation complexity on ASIC chip.

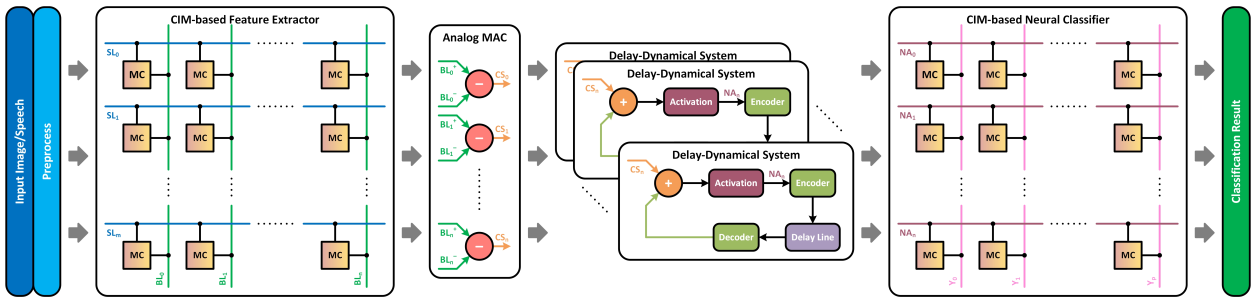


Figure 5.1: System-level architecture of spatial-temporal hybrid neural network.

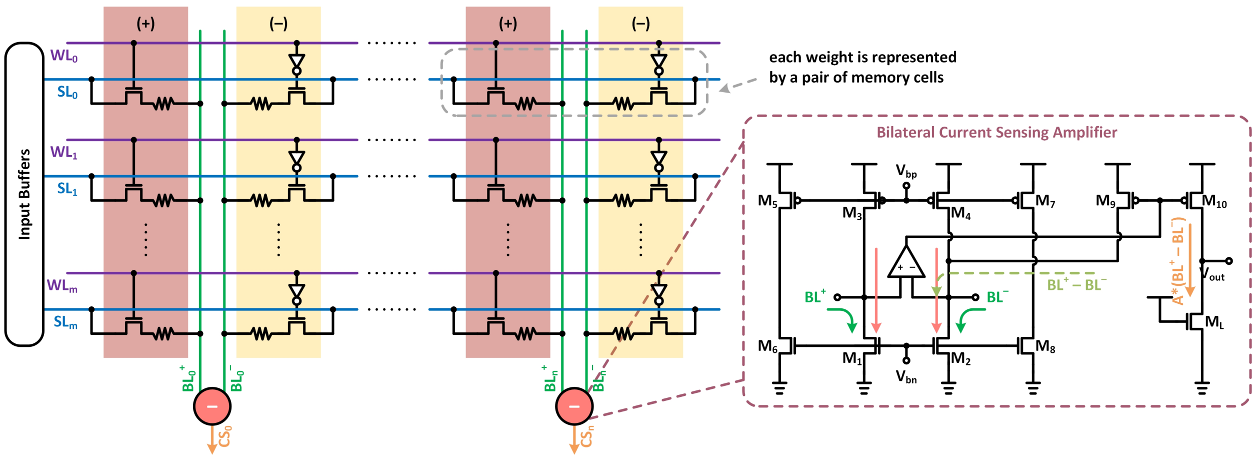

Input Buffers
(+)
(−)
each weight is represented by a pair of memory cells
Bilateral Current Sensing Amplifier

Figure 5.2: Deploying the analog multiplication-and-accumulation operator on a resistive crossbar with in-memory computing strategy.

5.2.2 Analog Multiplication-and-Accumulation Operator

The deployment of multiplication-and-accumulation (MAC) operation on a resistive crossbar is depicted in Fig. 5.2. In practice, any given patterns (*e.g.*, images or speech waveform) are commonly reshaped into an one-dimensional (1D) row vectors, $[1, m]$, where m is the input dimension, before the neural computation takes over. To extract m_f features, a crossbar with the shape of $[m, m_f]$ is deployed. With the consideration of precise modeling, both positive and negative weights are utilized in this project to form a double-column crossbar, in which a single weighted value of W_{in} is represented by a pair of memory cells (MCs) with a shared source-line (SL), that is, $G_{ij} = G_{ij}^{+} - G_{ij}^{-}$, where G_{ij}^{+} and G_{ij}^{-} denote the positive and negative conductance of MCs, respectively.

Table 5.1: Simplified truth table of multiplying-and-accumulating operation with binary input.

Input (1-bit)	(+) Weight	(-) Weight	MAC Out
0	HRS	LRS	0
0	LRS	HRS	0
1	HRS	LRS	$I_{high}@BL^{-}$
1	LRS	HRS	$I_{high}@BL^{+}$

The MCs used in this design are built of resistive memory, in which the high-resistance-state (HRS) and the low-resistance-state (LRS) of each MC are defined as $R \in$ [1k–to–20kΩ]. This particular resistance state is allocated within the range of measured resistance states through the discrete resistive random-access memory (ReRAM) cell (BS-AF-W) from the Knowm Inc. [186] with a linear scaling factor of 50,000. Table 5.1 demonstrates a simplified truth table of MAC operation with a binary input. In such a computation, a group of voltages, representing the corresponding information of raw sensory inputs, is applied to the horizontal SLs synchronously. As the outcome, an accumulated current is generated at each vertical bit-line (BL) where the MAC operation is deployed, which can be expressed as

$$I_j = I_j^+ - I_j^- = \sum_{i=1}^{m} V_i \cdot G_{ij}^+ - \sum_{i=1}^{m} V_i \cdot G_{ij}^-. \tag{5.1}$$

It can be observed that a subtraction operation is required for the double-column crossbar as the negative conductance nor current are non-existent. To support such an operation, a bilateral current sensing amplifier (BCSA) with the inlaid neural activation is deployed, as depicted in the dash box in Fig. 5.2. During the operation, the accumulated I_j^+ and I_j^- are respectively injected into the BSCA. The transistor M_1 senses the positive input current, I_j^+, and the reference current generated through the transistor M_3, such that $I_{M1} = I_j^+ + I_{M3}$. This current is then replicated through the associated current mirror, M_5–to–M_8, such that $I_{M8} = I_{M6} = I_{M1}$. As the negative input current, I_j^-, injects into the transistor M_2, the associated current mirror forces M_2 to replicate the current at M_1, and thus, $I_{M2} = I_{M1}$.

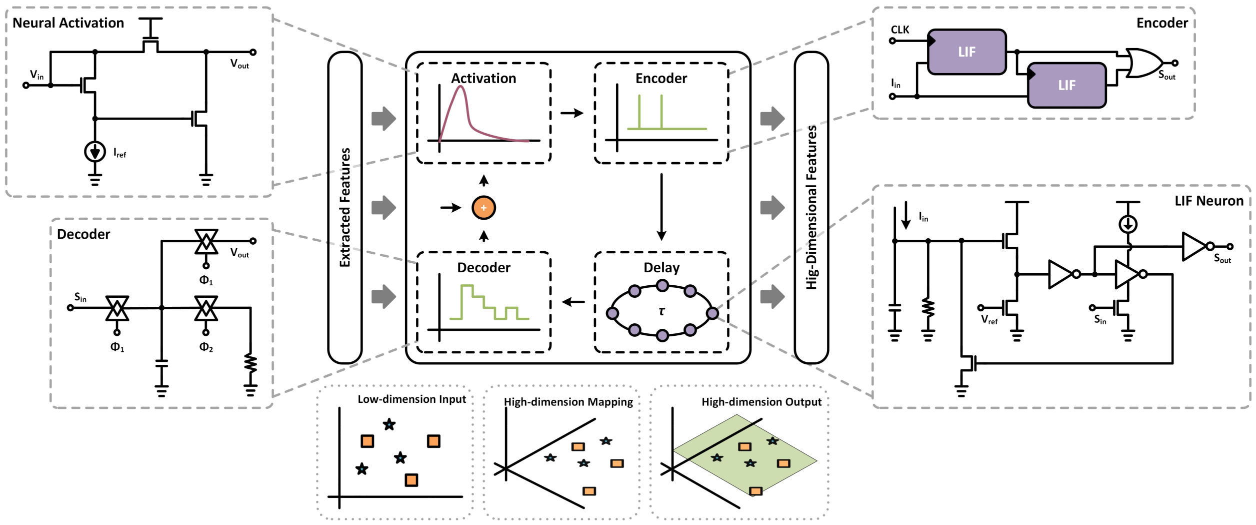


Figure 5.3: Computing structure of dynamic reservoir layer.

By balancing the current of I_{M1} and I_{M2}, the current through the transistor M_9 can be expressed as $I_{M9} = I_j^+ - I_j^-$, such that, $I_{M2} = I_{M9} + I_j^- + I_{M4} = I_j^+ + I_{M4} = I_{M1}$. The high-gain operational amplifier keeps tracking the variation of its positive and negative inputs, and dynamically regulates the driving voltage of M_9. The resulting current of $I_j^+ - I_j^-$ is replicated through the output current mirror, M_9 & M_{10}, with a current gain of A. The output voltage can be then consistently generated through a loading transistor M_L. Such a linear computation is obtained when $I_j^+ - I_j^- > 0$. By contrast, $I_{M9} = 0$ when $I_j^+ - I_j^- < 0$ as the current cannot be propagated reversely through the transistor M_9, and thus, output remains at 0V.

Such an analog MAC operator process multiple inputs in a single reading to enable the high-speed parallel operation, minimizes the output voltage variation under various BL currents, models the neural activation of the rectified linear unit (ReLU) function as required in the feature extractor, and isolates the MAC operation and latter computations. More importantly, the analog MAC operator generate the desired analog signal as required for latter computations, and thus, power-hungry data conversions are unnecessary, potentially making the system suitable for large-scale DNN designs.

Algorithm 3: Spatial-Temporal Hybrid Neural Network

```
Data: $x = [n, m], y = [n, u], scaling = \alpha$
Result: $W_{out}$
initialization
$W_{in} := \frac{|\lambda_{max}(W_{in})|}{\sigma} \cdot W_{in}$
for $i \to 1$ to $n$ do
    $flat = flatten(m)$
    $map = f_{ReLU}(flat \cdot W_{in} + b_{in})$
    for $ii \to 1$ to $len(map)$ do
        $h_t = f_{MG}((1 - \alpha) \cdot map_t + \alpha \cdot h_{t-1} + b_h)$
    end
end
return $h_t$
$S = [h_0, h_1, \cdots, h_n]$
$Y = [y_0, y_1, \cdots, y_n]$
$W_{out} = Y \cdot S' \cdot (S \cdot S' + \eta \cdot I)^{-1}$
```

5.2.3 Dynamic Reservoir

The dynamic reservoir layer embedded in the introduced STHNN is optimized based upon the previously designed DFR network, in which the Mackey-Glass (MG) NNA and the leaky integrate-and-integrate (LIF) neuron are deployed, as demonstrated in Fig. 5.3. The use of MG function as NNA in a delay-feedback system was first introduced in [88]; afterward, the previously designed Ci-DFR network demonstrate the advantages of MG function over classical NNAs. In this project, the electronic circuit model of MG NNA is optimized based on the previous design in [207] with 2.66× power reduction. Beyond that, the LIF neurons used in the dynamic feedback loop (DFL) is optimized based upon the previously designed IF neuron in [27] with 5.37× power reduction. The operating principle of the newly optimal dynamic reservoir layer follows the same trend as in the previously designed DFR network, in which the final outcomes from the dynamic reservoir layer are decoded back into analog signals for the training operation, whereby the Tikhonov regularization is adopted on the CIM-based neural classifier. The general learning operation of the introduced STHNN is summarised in Algorithm 3.

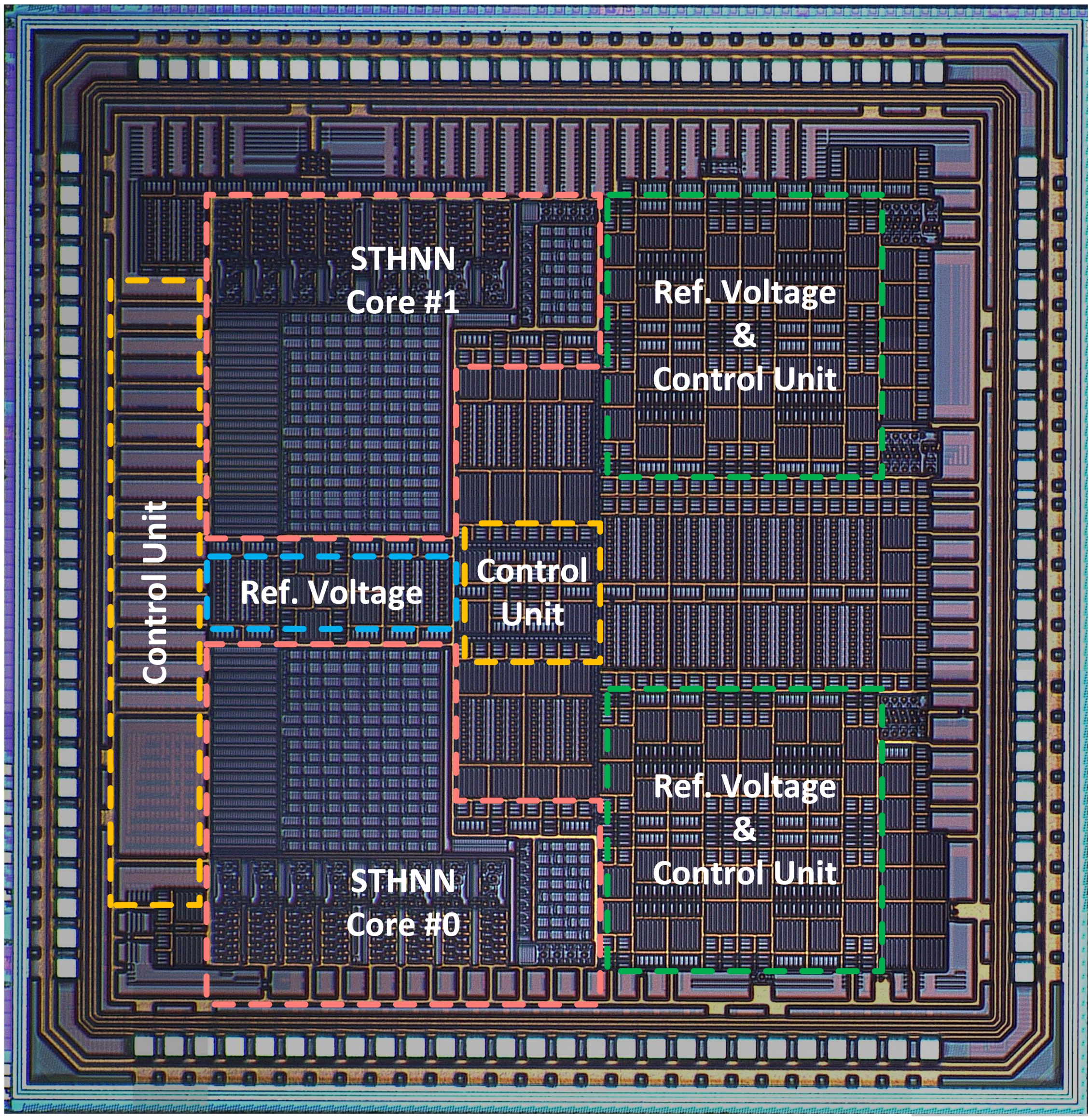


Figure 5.4: Die Micrograph of fabricated spatial-temporal hybrid neural network prototype in 180nm CMOS process, occupying 9mm^2 silicon area.

5.3 Performance Evaluation of Fabricated On-Silicon Prototype

A prototype of STHNN is fabricated in *GlobalFoundries* standard 180nm CMOS process for basic function verification of on-chip classification. Fig. 5.4 demonstrates the die micrograph of the fabricated STHNN prototype. Two STHNN cores along with the peripheries and basic function modules measure 9mm^2, where each STHNN core occupies 0.814mm^2. In this fabricated prototype, each STHNN core is built of a 16×8 CIM-based feature extractor with 8 analog MAC operators, 8$\times$ dynamic reservoirs with a total of 64 neurons, and a 8×4 CIM-based neural classifier. During the inference, input weights and output weights are initialized and calculated, respectively, through an offline compiler.

5.3.1 Characteristic of Bilateral Current Sensing Amplifier

With the introduced BCSA, the linearity and the stability between the MAC operation and latter computations can be maintained, in a similar way to the linear current amplifier as discussed in Section 4.3. To demonstrate such a functionality, input currents, I_{in}^{+} and I_{in}^{-}, respectively collected from BL^{+} and BL^{-} from the crossbar were directly applied to the designed BCSA. As plotted in Fig. 5.5, a linear response of output voltage, in a range of 0–to–0.7V, can be obtained when $I_{in}^{+} - I_{in}^{-} > 0$, in which the dynamic range of both I_{in}^{+} and I_{in}^{-} were set to be 0–to–1mA. By contrast, V_{out} remains at 0V when $I_{in}^{+} - I_{in}^{-} < 0$. Compared to the previous design in [122] and the literature reported in [208], the newly designed BCSA is capable to process information collected from both positive and negative synaptic weights and naturally achieve a ReLU NNA.

5.3.2 Nonlinear Behavior of Mackey-Glass Neural Activation

As demonstrated in Fig. 5.6, a nonlinear response of output voltage from the newly designed MG NNA can be observed, in which the input voltage is set to be 0–to–1.8V. It can be also observed that the measured nonlinear behavior fits the ideal MG function with the scaling

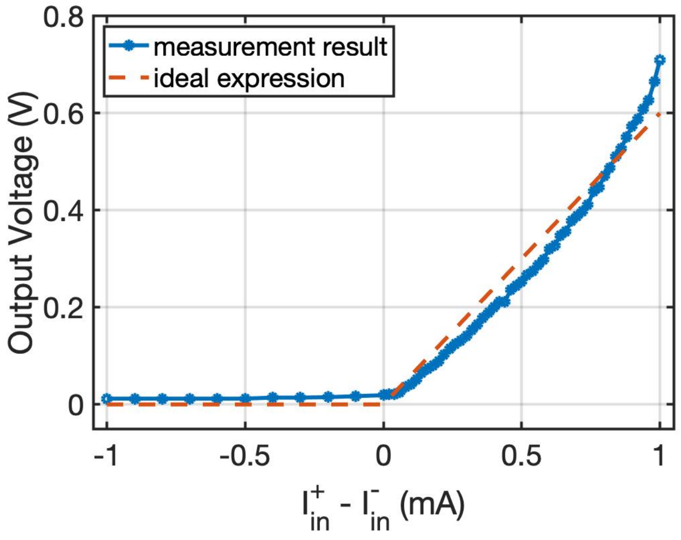


Figure 5.5: Measured characteristic of bilateral current sensing amplifier together with ideal fit.

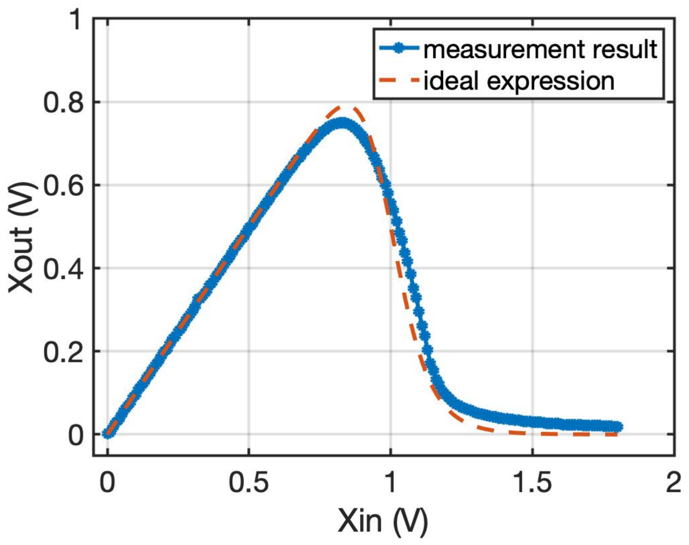


Figure 5.6: Measured nonlinear behavior of Mackey-Glass Neural Activation together with ideal fit.

parameter, nonlinear and delay exponents of $\beta = 1$, $\epsilon = 16$ and $tau = 1$, respectively, as depicted in Eq. 4.6.

5.3.3 On-Chip Character Classification

The neural classifier in the fabricated STHNN prototype contains four output neurons, which are capable to classify images into four categories. In this experiment, 12 capital characters were drawn from the NIST handprinted alphabet character database [209]. Each MC in the feature extractor and neural classifier were set according to the optimal weights after training a small-scale network through software. During the inference, selected images (128×128 pixels) were cropped (32×32 pixels), down-sampled (4×4 pixels) and reshaped into a 1D column vectors through a microcontroller (MCU). The final outcomes from the MCU were further scaled down to the desired voltage level supported by the fabricated test chip (*e.g.*, 0–to–1.8V) through off-chip voltage dividers and buffers. These voltage signals were then processed by the on-chip STHNN, in which the corresponding categories of input images were measured simply as the highest voltage amplitude among four output neurons. With the use of down-sampling operation, a large group of characters cannot be represented or differentiated by a 4×4 array, and thus, only 12 characters were adopted in this experiment.

During the evaluation, 128 measurements were carried out for each set of random characters to demonstrate the robustness of our fabricated STHNN prototype. Table 5.2 depicts the classification accuracy of 12 selected characters deployed in 8 different testing sets. Under the scenario without noise, the average classification accuracy for 4,096 images among 8 testing sets reaches 86.1%. Since the down-sampled images of characters "A, F and P" as well as "H and N" are difficult to be differentiated in a small array, a lower classification accuracy is reported when they are examined under the same group. Beyond that, the classification accuracy was also evaluated with the introduction of noise by randomly overwriting the information of 1–to–3 pixels for each down-sampled images. It can be deduced from measurement that a higher inference error occurs with the introduction of noise, yielding an average classification accuracy of 73.1%.

Table 5.2: Measured on-chip classification accuracy of 12 selected characters deployed in 8 testing sets.

		Test 1				Test 2			
Character		A	C	I	N	A	F	N	T
Accuracy	w/o noise	94.5%	93.0%	95.3%	96.9%	53.1%	46.9%	97.7%	96.9%
	w. noise	82.0%	78.9%	79.7%	81.3%	46.1%	40.6%	82.8%	79.7%

		Test 3				Test 4			
Character		C	F	T	U	C	N	O	P
Accuracy	w/o noise	96.9%	96.1%	92.2%	90.6%	93.0%	92.2%	94.5%	93.0%
	w. noise	82.8%	84.4%	80.5%	78.1%	79.7%	79.7%	81.3%	82.0%

		Test 5				Test 6			
Character		F	I	J	N	F	N	P	X
Accuracy	w/o noise	96.9%	95.3%	95.3%	97.9%	48.4%	97.7%	45.3%	96.9%
	w. noise	81.3%	76.6%	82.8%	84.4%	39.1%	81.3%	40.6%	79.7%

		Test 7				Test 8			
Character		H	N	U	X	J	N	P	X
Accuracy	w/o noise	46.9%	53.1%	90.6%	95.3%	93.8%	92.2%	91.4%	94.5%
	w. noise	37.5%	41.4%	76.7%	82.8%	82.0%	81.3%	76.6%	76.6%

5.3.4 Power and Area Analysis

The power distribution and area breakdown of the fabricated STHNN prototype are demonstrated in Fig. 5.7, where the total power consumption of 33mW with a supply voltage of 1.8V is reported during the inference operation. The feature extractor together with eight analog MAC operators occupy 40% of total power and 44% of total area; 8× dynamic reservoirs occupy 28% and 36% of total power and area, respectively; the neural classifier occupies 20% of total power and 10% of total area, and the rest are occupied by peripheries (*e.g.*, analog/digital buffers, reference voltage/current generators, etc.). Design specifications of the fabricated STHNN prototype and the comparison to the state-of-the-art neuromorphic chips are summarized in Table 5.3. Due to the limitation of a small-scale MAC operation, the reported performance efficiency of 393×10^{-6}TOPS/mm^2 is significantly lower. However, the fabricated STHNN prototype demonstrates the possibility of bridging MLP and dynamic reservoir in a processing pipeline for neuromorphic applications, potentially yielding a competitive performance with a simplified network structure.

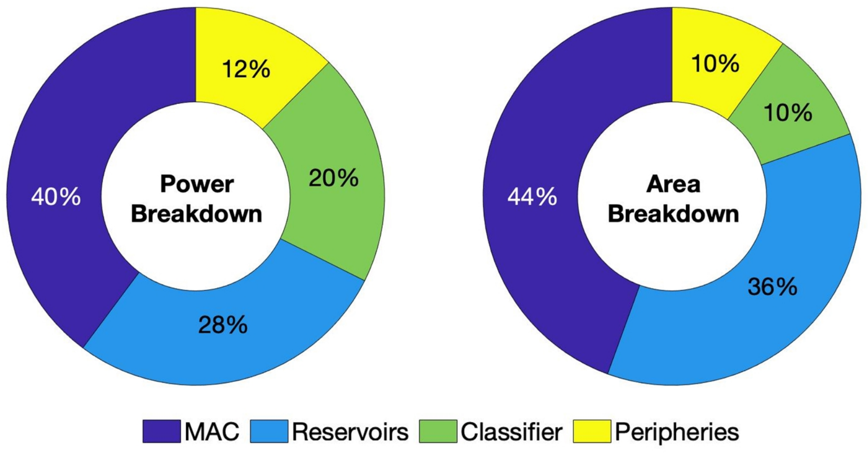


Figure 5.7: Power distribution and area breakdown of fabricated spatial-temporal hybrid neural network prototype in silicon @ 33mW with an operating frequency of 1MHz and @ 0.814mm^2, respectively.

5.4 Application Evaluation of Spatial-Temporal Hybrid Neural Network

Due to the limited silicon area and the fabrication cost, the fabricated on-chip STHNN prototype only contains eight analog MAC operators and four out neurons, having a limited classification capability on real-world applications. To further demonstrate the performance and the reliability of the introduced STHNN, a mathematical model was implemented in TensorFlow, and its accuracy was evaluated through the handwritten digital classification.

Table 5.3: **Design specifications of fabricated spatial-temporal hybrid neural network prototype and comparison to the state-of-the-art Neuromorphic Chips.**

	[187]	[188]	[189]	[210]	[211]	This Work
Technology	65nm	10nm	28nm	130nm	130nm	180nm
Die Area	16mm^2	1.72mm^2	5.95mm^2	21.82mm^2	1.79mm^2	9mm^2
Algorithm	CNN	SNN	CNN	MLP	RBM	MLP + DFR
Memory Cell	SRAM	SRAM	SRAM	ReRAM	ReRAM	ReRAM
Memory Mode	–	–	–	CIM	CIM	CIM
Weight Precision	–	7-bit	1-bit	3-bit signed	1-bit	1-bit
On-chip Learning	no	yes	–	yes	no	no
Core Structure	–	neuron	DAC	ADC	neuron	neuron
Neuron Type	–	LIF	–	–	IF	LIF
Latency	–	–	–	51.1ns	–	185ns
Supply Voltage	1V	0.9V	0.8V	5V	1.8V	1.8V
Power Consumption	278mW	–	0.899mW	–	0.05–to–2.2mW	33mW
Application	CIFAR-1000	MNIST	CIFAR-10	MNIST	MNIST	NIST
Inference Speed	34.7frames/s	–	380frames/s	77μs/image	–	10μs/image
Accuracy	–	89%	86.05%	94.4%	–	86.1% w/o noise 73.1% with noise

5.4.1 Experimental Setup

In this experiment, the performance was evaluated through the 4-core Intel i7-6700K CPU and 16G RAM. The MNIST handwritten digit dataset [198] used in this experiment has the same configuration as the on in the Ci-DFR network as depicted in Section 4.5. Four baseline DNN models were adopted for the performance comparison, including MLP, CNN, long short-term memory (LSTM) and gated recurrent unit (GRU). All weight matrices and bias vectors of baseline DNN models were updated through the gradient-based backpropagation learning algorithm by minimizing the cross-entropy loss through the Adam optimizer. Learning parameters were set as follows: an L2 regularization was utilized to prevent the overfitting, the dropout rate on the readout layer was set to be 0.5, and the learning rate was set to be 1×10^{-3}.

5.4.2 Classification Accuracy

The classification accuracy of the introduced STHNN with respect to various numbers of neurons in the MAC operator is plotted in Fig. 5.8. It can be observed that a higher classification accuracy can be achieved as the number of neurons increases. However, a longer training time is then required to train the network, *e.g.*, it takes 15secs to run a 256-neuron but 4mins to run a 2,048-neuron. The performance of the introduced STHNN and baseline DNN models are summarized in Table 5.4.

In this experiment, CNN demonstrates the highest classification accuracy due to its feature-based learning mechanism, and yet, the longest training time is reported as the convolution operation is computationally expensive. It can be observed that the introduced STHNN achieves the lowest classification accuracy when the number of neurons is set to be the same as in all other DNN models. To perform the same, the number of neurons in STHNN had to be increased to 2,048, and yet, still exhibiting 1.1–to–6.5$\times$ speedup over alternative approaches. Moreover, MLP performs 4$\times$ faster than the STHNN with a similar classification accuracy due to its simple structure and the simplicity of dataset, and yet, this would not been the case for other complicated machine intelligence applications.

Table 5.4: Classification accuracy comparison to baseline deep neural network models on handwritten digit database.

	MLP	CNN	LSTM	GRU	STHNN
Network Structure	2× dense	2× conv. 2× pool. 1× dense	1× rec.	1× rec.	1× dense 1× DFR 1× dense
# of Kernels	–	$64 \times [3,3]$	–	–	–
# of neurons	2× 256	256	256	256	various
Training Epochs	10	10	10	10	1
Classification Accuracy	97.37%	98.58%	98.75%	98.89%	97.5% @ 2,048 neurons 90.3% @ 256 neurons
Training Time	1min	26mins	5.5mins	4.5mins	4mins @ 2,048 neurons 15secs @ 256 neurons

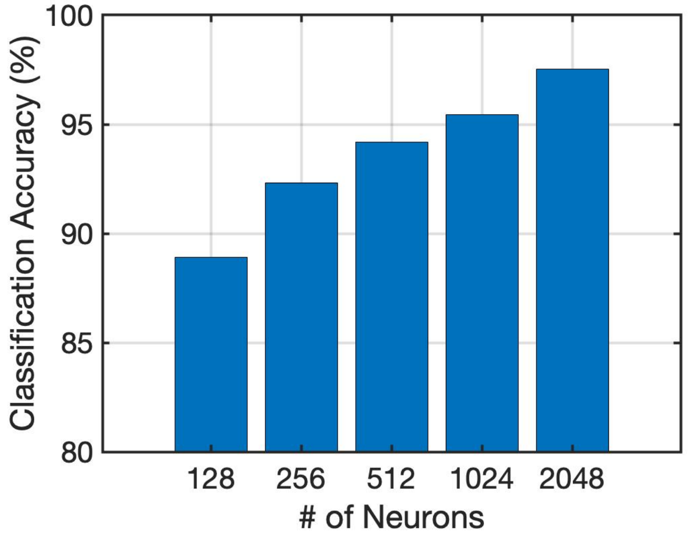


Figure 5.8: Classification accuracy of spatial-temporal hybrid neural network with respect to various numbers of neurons in the multiplication-and-accumulation operator on handwritten digit database.

5.4.3 Reliability

Since the output weights of STHNN are trainable in a single step, it is often possible to successfully train such a network with significantly fewer data. Such a capability is summarized in Fig. 5.9 together with baseline DNN models. It can be observed that both MLP, CNN and STHNN are capable to maintain their classification accuracy even with fewer training samples, while the others have a significant reduction. Unsurprisingly, with the recurrent nature embedded in the dynamic reservoir layer, the introduced STHNN does not have a strong capability in enhancing the classification accuracy of static data, and yet, the computational efficiency of STHNN is still 6.5× higher than the one with CNN. Table 5.5 summarizes the classification accuracy comparison to the cutting-edge DNN approaches.

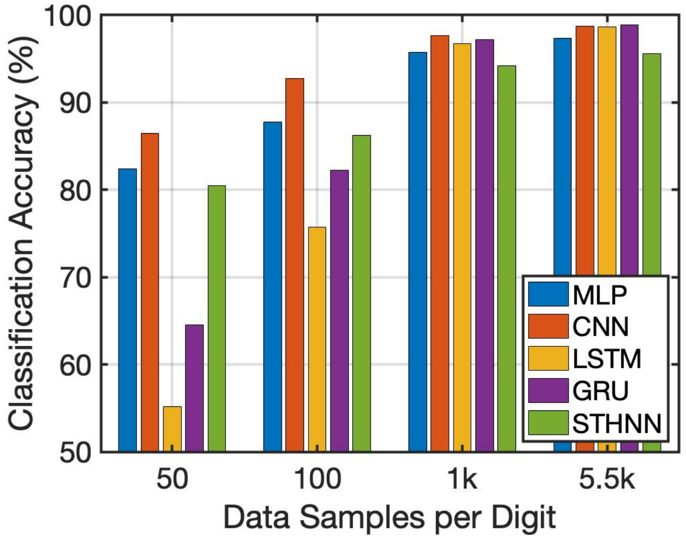


Figure 5.9: Classification accuracy with respect to various data samples on handwritten digit database.

5.5 Spike-Timing-Dependent Plasticity

The use of Tikhonov regularization in the introduce DFR network, Ci-DFR network and STHNN indeed accelerates the training operation while yielding a competitive accuracy over the gradient descent algorithm, and thus, potentially reducing the implementation com-

Table 5.5: Classification accuracy comparison to the cutting-edge deep neural network models on handwritten digit database.

	Algorithm	Network Structure	Classification Accuracy
[200]	Spiking CNN	3× convolution 3× pooling 1× dense	97.2%
[201]	Binary MLP	3× Dense	95.8%
[212]	MLP	7× Dense	98.3%
[213]	CNN (LeNET–5)	3× convolution 2× pooling 2× dense	99.1%
[214]	MLP	–	99.4%
[122]	Ci-DFR	2× convolution 2× pooling 1× DFR 1× dense	99.0%
This Work	STHNN	1× dense 1× DFR 1× dense	97.5%

plexity and hardware overhead. Nevertheless, the fact that decoding the temporal spike train back into numerical numbers (*i.e.*, analog signals) undoubtedly introduce additional operations, requiring more resources to be allocated, and thus, concealing the advantages introduced by the aforementioned network architectures.

By relating the plasticity at both excitatory and inhibitory synapses, a coincidence detection mechanism can be realized when two asynchronous spiking events are generated with relative timing. Such a mechanism is commonly known as the spike-time dependent plasticity (STDP) [215], illustrating a neuronal excitability with respect to the strength of coincidence between spiking events. In general, STDP provides systems the ability to learn and optimize based on neuronal excitability, and thus, becoming a quintessential learning rule for spiking neural networks (SNNs) [216–219]. The symmetric STDP and the asymmetric STDP are the two well-known STDP learning rules. The former performs the weight adjustment regardless of the temporal order while the latter performs the weight adjustment according to the temporal order and the absolute time difference, as depicted in Fig. 5.10.

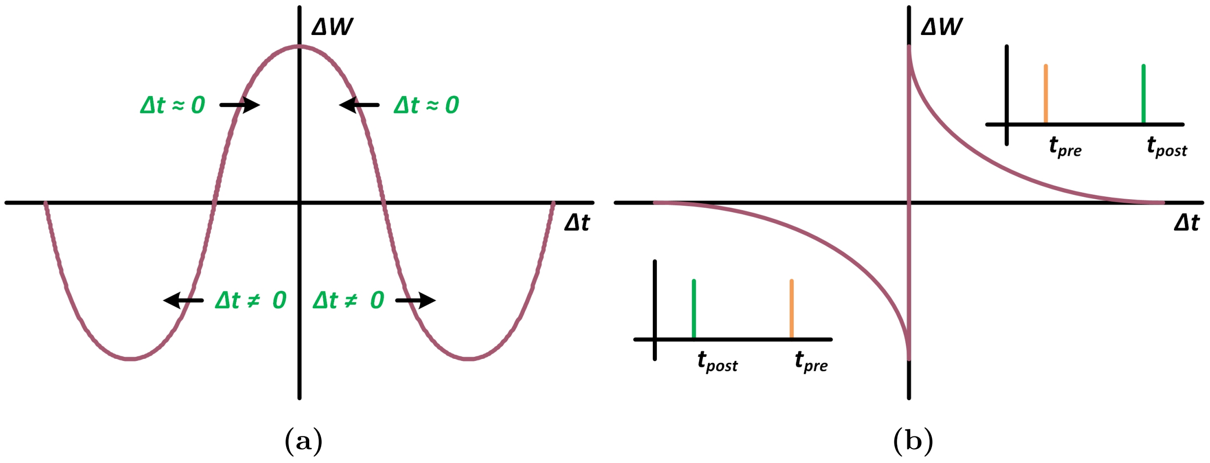


Figure 5.10: Illustration of spike-time dependent plasticity of (a) symmetric and (b) asymmetric rules.

STDP learning rule adopted the neuronal excitability to perform the weight adjustment without the need of gradient computation and backpropagation operation, and thus, potentially improving the computational efficiency over classical DNNs. To this end, it is essential to investigate the possibility of implementing such an efficient training mechanism onto the introduced hybrid neural computing architecture, sidestepping the necessity of spike to analog decoding operation.

5.5.1 Circuit Principle

Fig. 5.11 demonstrates the electronic circuit model of asymmetric STDP, composing of a predicted spike detector, a target spike detector, a phase comparator, and a weight regulator. During the operation, the the predicted spike detector, M_1–to–M_4, extracts the absolute time of a predicted spike, S_{pre}, from the network and convert this spike data into a analog signal. In the meantime, the same operation is executed at the target spike detector, M_5–to–M_8

upon the arrival of a target spike train, S_{tar}. The phase comparator extract the absolute time difference between S_{pre} and S_{tar}, and correspondingly control the switching mechanism to adjust the weighted value on a capacitor, C_w. To be specific, weighted value is increased by increasing the charges stored in C_w under the scenario of $S_{tar} - S_{pre} < 0$, where the relative timing of target spike leads ahead predicted spike. By contrast, weighted value is reduced by removing charges from C_w under the scenario of $S_{tar} - S_{pre} > 0$, where the relative timing of target spike lacks behind predicted spike.

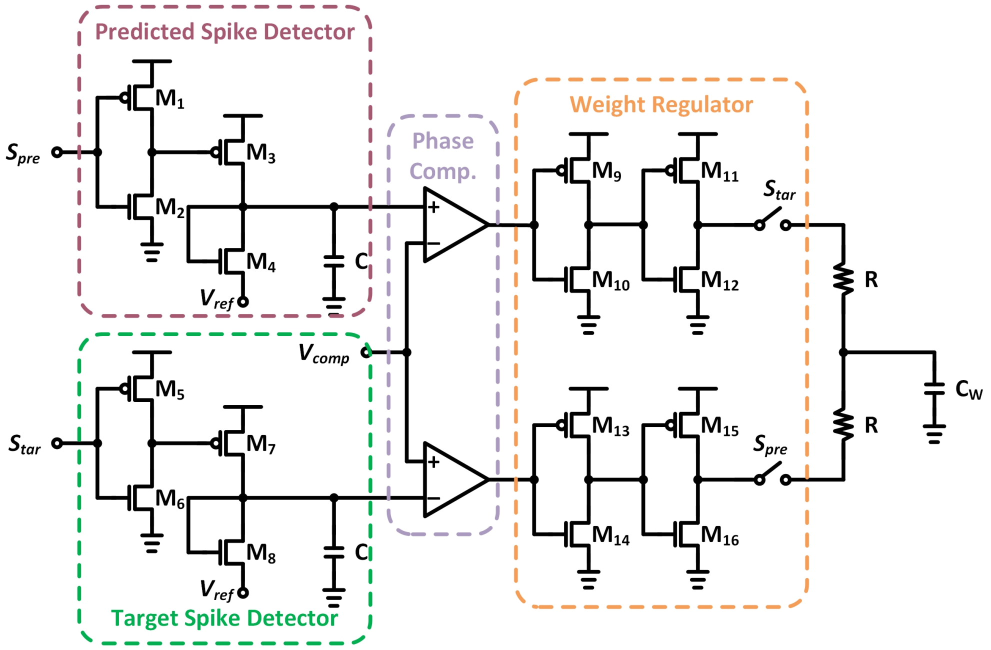


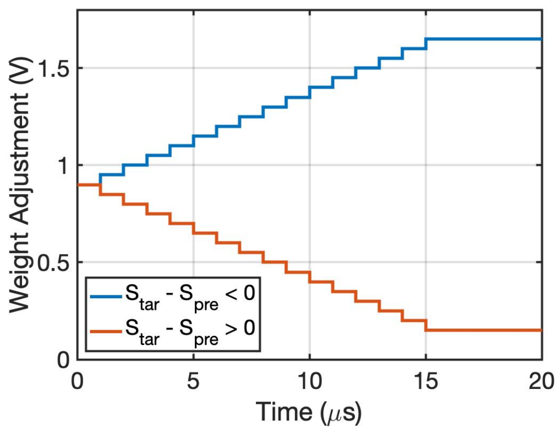
Figure 5.11: Design scheme of spike-time dependent plasticity.

5.5.2 Experimental Evaluations

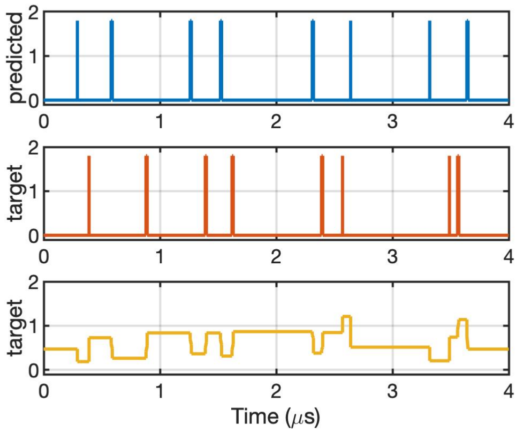
The simulated weight adjustment, as plotted in Fig. 5.12, was recorded by utilizing the predicted spike trains, collected from the dynamic reservoir layer of STHNN, and the target spikes trains, generated according to target labels. It can be observed from the simulated results that a weighted value can be trained with the temporal order and the absolute time different using STDP learning rule. The performance comparison to the published literature is summarized in Table 5.6.

Table 5.6: Performance comparison of introduced spike-time dependent plasticity circuit model to published literature.

	[220]	[218]	[221]	[219]	[188]	[222]	This Work
Technology	350nm	250nm	180nm	130nm	10nm	65nm	180nm
Design Strategy	analog	analog	analog	analog	digital	digital	analog
Memory Cell	DRAM	DRAM	–	DRAM	SRAM	SRAM	DRAM & ReRAM
Neuron Type	LIF	IF	LIF	LIF	LIF	–	LIF
Encoding	TTFS	–	ISI	–	–	–	ISI
Supply Voltage	3.3V	3.3V	1.8V	1.0V	0.9V	1.2V	1.8V
Power Consumption	–	250μW	200μW	55μW	–	–	150μW



(a)



(b)

Figure 5.12: Simulated weight adjustment with introduced spike-time dependent plasticity circuit model in (a) ideal scenario and (b) actual results from the network.

Table 5.7: Performance comparison of spike-time dependent plasticity to classical training algorithms on handwritten digit database.

	MLP	SNN	STHNN	
# of Layers	2	2	3	3
# of Neurons	2× 256	2× 256	2,048	2,048
Training Algorithm	gradient descent	unsupervised STDP	Tikhonov regularization	supervised STDP
Training Epochs	10	10	1	10
Classification Accuracy	97.4%	91.7%	97.5%	91.9%
Training Time	1min	1.5mins	4.5mins	1.3mins

Since additional data conversion is not necessary, it is often possible to accelerate the training efficiency with STDP learning rule. Such a capability is demonstrated in Table 5.7 together with the performance comparison to classical training algorithms. In this experiment, the same configuration of MNIST handwritten digit database were adopted. From the baseline models, it can be observed that the classification accuracy of a general SNN is lower than MLP even with the same network structure. The same tendency can be found in the introduced STHNN. It can be observed that the use of supervised STDP learning rule indeed accelerates the training time by 3.5×, and yet, the classification accuracy is 5.6 percentage points poorer.

5.6 Conclusions

To support the integration of deep learning strategies and neuromorphic architecture, a STHNN with the CIM strategy is introduced together with a small-scale prototype fabrication. To be specific, a set of neural computations is implemented to realize the spatial and temporal information processing, wherein CIM-based feature extractor and dynamic reservoir layer are incorporated. A 180nm prototype chip is fabricated with 86.1% on-chip classification accuracy on handprinted alphabet characters at a power consumption of 33mW. Beyond that, numerical evaluations demonstrate that the introduced STHNN offers up to 6.5$\times$ speedup over the state-of-the-art DNN models while yielding a competitive classification accuracy. In the endeavor to accelerate the training efficiency, a supervised STDP learning rule is designed, whereby 3.5$\times$ speedup are obtained over the Tikhonov regularization. In conclusion, this work simplifies the neural computing architecture with a unique training mechanism; therefore, a complex machine intelligence application can be processed with a simple system-level design. Even with a less advanced CMOS process, this work is still capable to realize a high classification accuracy with an appreciable power consumption, opening the door to future mobile edge intelligence computing.

Chapter 6

Deep Learning Strategies on Internet of Things

6.1 Introduction

Not only will future high-speed communication networks interconnect people, they will also interconnect machines and devices, such as the internet of things (IoT) applications, which will result in billions of devices deployed worldwide [223]. Emerging applications in the context of IoT often require high-bandwidth communication networks like 5G, empowering new user experience, and yet, causing a spectrum crunch and impacting spectrum usage [224]. More importantly, the demand on high-speed and reliable receiver has increased with the increasing demand for wireless propagation characteristics.

Multiple-input multiple-output orthogonal frequency-division multiplexing (MIMO-OFDM) technology is the foundation of modern high-speed cellular and wireless local area networks, playing an important role for signal transmission. The applications of IoT often require the big data transmissible and analytic capabilities in real time, which relies on heterogeneous environments and channel knowledge. In many mission-critical scenarios, due to the non-linear distortion caused by practical radio frequency (RF) components in the transceiver chain and the noise interference introduced by wireless connections, demodulating a group of modulation symbols or identifying a particular type of protocol in use have become more difficult than ever.

Deep learning strategies, on the other hand, offer an alternative approach for IoT applications, in a way that is sufficient by training to reconstruct corrupted signals from distortion, interference and noise at the receiver. While a vast majority of literature employs the convolutional neural networks (CNNs) [225–230], the fact that pre-processing continues streams of data significantly degrades the computational efficiency and requires additional resources to be allocated. Due to the nonlinear sequential nature in communication networks, recurrent neural networks (RNNs), including the vanilla RNN [67], the long short-term memory (LSTM) [68], and the gated recurrent unit (GRU) [69], are expected to be the most suitable deep learning strategies for the complex temporal information processing by regulating sequential inputs over arbitrary time intervals.

My goal in this project is to explore the applications of deep learning strategies on IoT applications [1], including communication and healthcare. Major contributions of this project are summarized as follow:

- A deep echo state network as modulation symbol detector for 5G MIMO-OFDM systems, yielding 47.73% more precise over the state-of-the-art approaches in the literate for 5G communication networks.
- An echo state network for radio fingerprint identification in over-the-air WiFi environments, yielding a classification accuracy as high as 98.11% and exhibiting 7.4$\times$ speedup over alternative deep neural network (DNN) models.
- Multiple DNN models for clinical disease identification in healthcare, yielding a classification accuracy as high as 91.7%, 80.0% and 96.9% in the identification of cardiovascular disease, breast cancer and coronavirus disease, respectively.

The rest of this chapter is organized as follows: Section 6.2 introduce the design solution for modulation symbol detection. The radio fingerprint identification and the clinical disease

[1]The performance of all applications presented in this chapter was evaluated through the 4-core Intel i7-6700K CPU and 16G RAM.

identification are demonstrated in Section 6.3 and Section 6.4, respectively. This chapter is then concluded in Section 6.5.

6.2 Modulation Symbol Detection

Conducting the modulation symbol detection in MIMO-OFDM systems under heterogeneous environments and channel conditions is one of the major challenges for 5G communication networks. This is because the received signal in a MIMO-OFDM system is the superposition of all modulation symbols associated with its sub-carriers. Increasing the size of the alphabet table can convey more information through one modulation symbol, and yet, enhancing the demodulation complexity in the meantime.

An accurate channel estimation is required in a classical modulation symbol detection approach, which typically consumes a colossal amount of resources. That is, more accurate channel estimation will require more resources to be allocated, resulting in fewer resources available for data transmission. The maximum likelihood (ML) [231] and the soft-output sphere decoding (SD) [232] are the two classical modulation symbol detection approaches in MIMO-OFDM systems. However, computational and implementation complexities scale exponentially with the increasing number of antennas and transmitted data streams.

Alternatively, data-driven approaches powered by deep learning strategies may pave the way towards large-scale MIMO detection with reduced hardware implementation overhead. Based on the supervised learning framework, deep learning strategies can be deployed to perform parameter tuning and formulated to perform a classification problem. For instance, in [233] and [234], an echo state network is deployed as the modulation symbol detector, sidestepping the required channel estimation and yielding a better energy efficient over classical approaches. Nevertheless, the memory capacity, representing the amount of input data that a reservoir layer can store, is limited. As the complexity of input data streams scales up, the learning capability of reservoir from short-term memory reduces. To this end, a deep echo state network is investigate in this project.

6.2.1 Deep Echo State Network

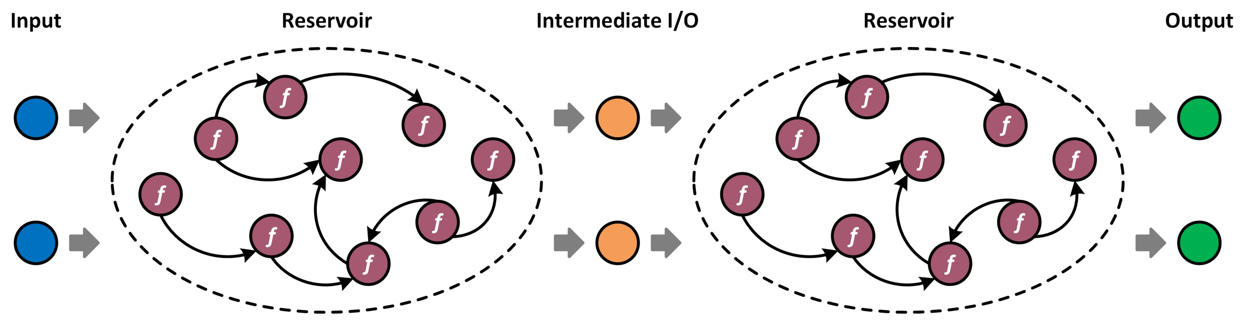


Figure 6.1: General processing structure of deep echo state network composing of two reservoir layers.

The general processing structure of deep echo state network (DESN) built of two cascading reservoir layers is demonstrated in Fig. 6.1. During the operation, a set of complex time-domain symbols of binary digits with both real and imaginary components are applied to the DESN, which can be defined as $x_t^{l=1} \in N_U$, where N_U denotes the input dimension and $l = 1$ denotes the first reservoir layer. The association between the raw sensory inputs and the reservoir layer is communicated through input weights, $W_{in}^{l=1} \in [N_U \times N_R]$, where N_R is the number of neurons in each reservoir layer. Through the hierarchical structure, the second reservoir layer adopts the intermediate output generated from its previous layer to compute its reservoir dynamics, *i.e.*, $x_t^{l=2} = \hat{y}_t^{l=1}$, in which the state of reservoir dynamics on each layer is calculated based on Eq. 2.4 with internal weights, $W_{res} \in [N_R \times N_R]$. The predicted output can be then computed by multiplying the reservoir dynamics from the second layer with output weight, $W_{out}^{l=2} \in [N_R \times N_Y]$, where N_Y is the output dimension.

By stacking multiple reservoir layers into a hierarchical processing structure, the state computation and the learning operation are carried out through the pipeline. To reduce the design complexity, the teacher forcing for each reservoir layer is the same. The general learning operation of the introduced DESN is summarized in Algorithm 4. The introduced DESN with stacked hierarchy of reservoir layers achieves multiple temporal representation for raw sensory information, allowing such a system to capture more features between input and output patterns, more importantly, improving the richness of reservoir dynamics and the memory capacity.

Algorithm 4: Deep Echo State Network

```
Data: $x = [n, m], y = [n, u], layer = l$
Result: $W_{out}$
initialization
$W_{res} := \frac{|\lambda_{max}(W_{in})|}{\sigma} \cdot W_{in}$
for $i \to 1$ to $n$ do
    for $ii \to 1$ to $m$ do
        if $l = 1$ then
            $h_t^{l=1} = f(m_{ii} \cdot W_{in}^{l=1} + h_{t-1}^{l=1} \cdot W_{res}^{l=1})$
            $\hat{y}_t^{l=1} = f(h_t^{l=1} \cdot W_{out}^{l=1})$
        else
            $h_t^{l=2} = f(\hat{y}_t^{l=1} \cdot W_{in}^{l=2} + h_{t-1}^{l=2} \cdot W_{res}^{l=2})$
        end
    end
end
return $h_t^{l=1}$, $h_t^{l=1}$
$S = [h_0, h_1, \cdots, h_n]$
$Y = [y_0, y_1, \cdots, y_n]$
$W_{out} = Y \cdot S' \cdot (S \cdot S' + \eta \cdot I)^{-1}$
```

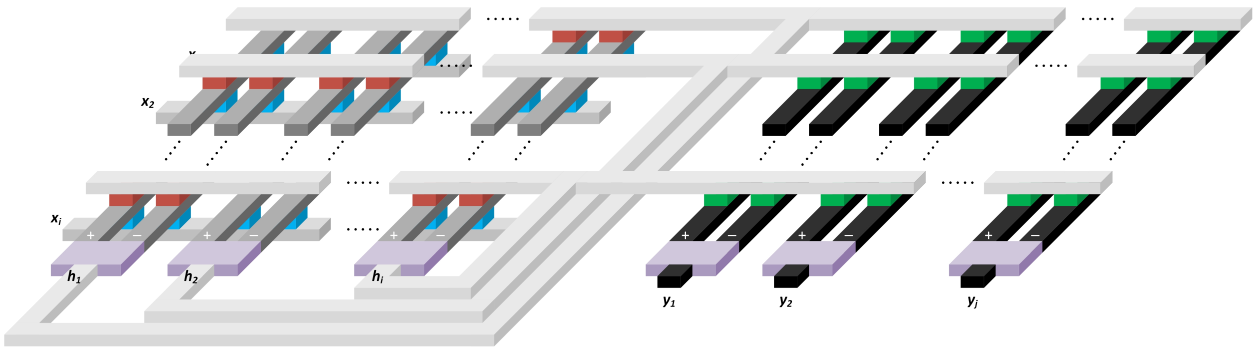


Figure 6.2: Deploying reservoir layer on memristive crossbars, in which W_{in} (highlighted in blue) and W_{out} (highlighted in green) are fully-connected, while W_{res} (highlighted in red) is sparsely-connected; "+" and "-" denote positive and negative weights, respectively.

Table 6.1: Design specifications of single reservoir layer with hybrid CMOS-memristor co-design and comparison to the state-of-the-art hardware modeling approaches.

	[235]	[236]	This Work
Implementation Strategy	FPGA	MATLAB	CMOS
Technology	45nm	–	180nm
Memory Cell	ReRAM	ReRAM	ReRAM
# of Neurons	30	1,000	128
Neural Activation	tanh	tanh	Mackey-Glass
Supply Voltage	0.55V	–	1.8V
Power Consumption	125.36mW	–	104.51mW

A generic hardware model of reservoir layer can be implemented with memristive crossbars, as depicted in Fig. 6.2, in which the analog multiplication-and-accumulation operators, as discussed in Section 5.2, and the resistive random-access memory (ReRAM) cells, as discussed in Section 4.4, are deployed. Design specifications of the single reservoir layer with hybrid CMOS-memristor co-design and the comparison to the state-of-the-art ESN hardware modeling approaches are summarized in Table 6.1.

6.2.2 Experimental Setup

In this experiment, the introduced DESN was deployed as the modulation symbol detector in the receiving chain of MIMO-OFDM systems, in which the MIMO-OFDM signal was generated according to the 5G new radio (NR) specification that follows the standard 3GPP TS 38.212 version 15.2.0 [237], where the channel was generated according to the Winner II channel model [238]. The modulation method was configured as 16 quadrature amplitude modulation (16-QAM). To be specific, pilots of the communication system, which are utilized for channel estimation, were evenly used for training. Details of the system specification were set as followings: the number of transmitting and receiving antennas was set to be 4, the

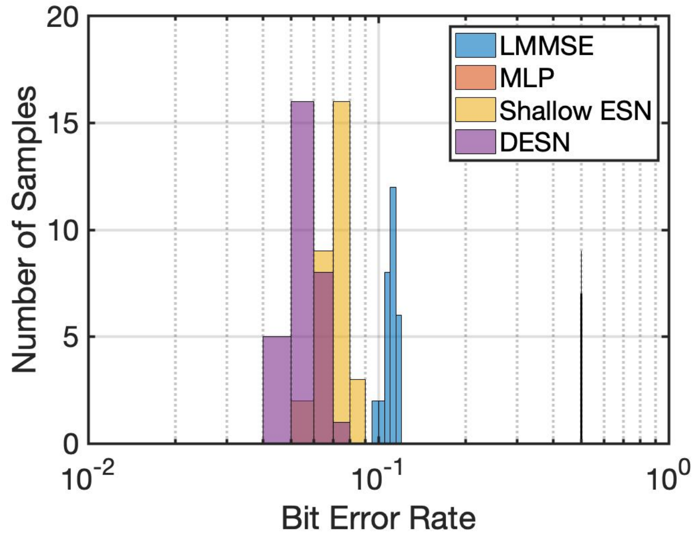


Figure 6.3: Testing bit error rate with respect to various detection approaches on demodulation symbol dataset.

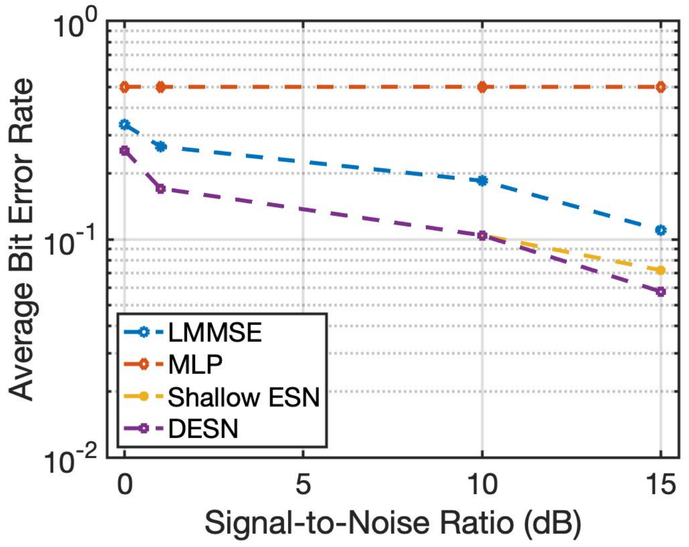


Figure 6.4: Testing bit error rate with respect to various detection approaches and signal-to-noise ratio on demodulation symbol dataset.

number of sub-carriers in the OFDM system was set to be 1024, and the number of neurons for each reservoir layer was set to be 128.

6.2.3 Detection Accuracy

The distribution of testing bit error rate (BER) with 30 samples is depicted in Fig. 6.3 and 6.4 with respect to various signal-to-noise ratios (SNRs). The linear minimum mean squared error (LMMSE) is a classical model-based detection approach relied on the linear processing method. Such an approach requires an accurate channel information, which is challenging to be obtained in the low SNR regime. The reported average BER of the introduced DESN is merely 5.76×10^{-2}, yielding 47.73% more precise over the LMMSE approach. Beyond that, the average BER of the introduced DESN is also compared to the multilayer perception (MLP) with 3 hidden layers and 1024 associated neurons per layer. Due to the limited training set, MLP has an average BER up to 50.12×10^{-2}. Thereby, it is convincing that the introduced DESN outperforms state-of-the-art modulation symbol detection strategies for all SNR regimes even with a very limited training set.

6.3 Radio Fingerprint Identification

Identifying and locating a particular device or protocol in use is one of the critical operations in IoT applications, and yet, such operations have become more difficult than even in the modern privacy-sensitive environment.

Many existing identification, localization and authentication mechanisms are based on cryptography algorithms and protocols, which consume considerable energy while themselves exposing other significant security issues. To identify a particular type of protocol or device in use, simple and unique identifiers are commonly utilized, for instance, the internet protocol (IP) address, the medium access control (MAC) address, and the international station equipment identity (IMEI) number [239]. Alternatively, geometry-based feature identifiers can be applied, such as the time-of-flight (ToF), the radio signal strength (RSS), the angle of arrival (AoA), and the channel state information (CSI) [240]. However, such simple and unique identifiers can be easily spoofed and stolen, while geometry-based features are susceptible to a device's physical mobility and environment changes.

It has been found that the likelihood of multiple transmitted radio signals having the same RF features is extremely low, even if such transmitters are broadcasting the same data in the same location [241]; this means every transmitter has a unique feature, known as the RF fingerprint. RF fingerprinting is a process that leverages characteristic features embedded in transmitted signals to discriminate radios in a shared spectrum environment. In general, RF fingerprints arise from inherent randomness in the manufacturing process, particularly the presence of analog components in a wireless transmitter, such as band-pass filters, frequency mixers, etc. Such randomness, also known as *hardware imperfections*, can be seen as a unique RF feature in a specific wireless transmitter, which are inherent to a device's hardware and cannot be replicated by malicious agents.

Based on the supervised learning framework, deep learning strategies can be trained to discover discriminating features caused by hardware imperfections for every RF signal, and thus, allowing the network to uniquely identify each radio device. Despite that RNNs are expected to be the most suitable deep learning strategies for complex temporal information processing, the fact that training all internal weights and bias vectors significantly increases the computational complexity. To this end, an ESN is investigate in this project to accelerate the training operation.

6.3.1 Experiment Setup

In this experiment, an ESN was trained to identify specific devices based on raw transmitted WiFi signals. The dataset used in this evaluation contains 16 USPR X310 transmitter radios, each of which has 20 millions bit-similar in-phase and quadrature (IQ) samples collected from over-the-air WiFi transmission with the same B210 radio as receiver [224]. All transmitters are bit-similar radios with random payload that emit IEEE 802.11a standards-compliant frames. The software-defined radio (SDR) of the receiver samples input signals at a sampling rate of 5MS/s with a center frequency of 2.4GHz, and the transmitter-receiver separation distance is set to be 2, 32, and 62ft. During the experiment, 512,000 complex IQ samples were drawn for each device, in which 75% of samples were adopted for training and the rest were adopted for testing. Collected IQ samples were further partitioned into multiple sub-sequences with a window length of 128, *i.e.*, the length of contiguous samples that will be used at a time for both training and testing.

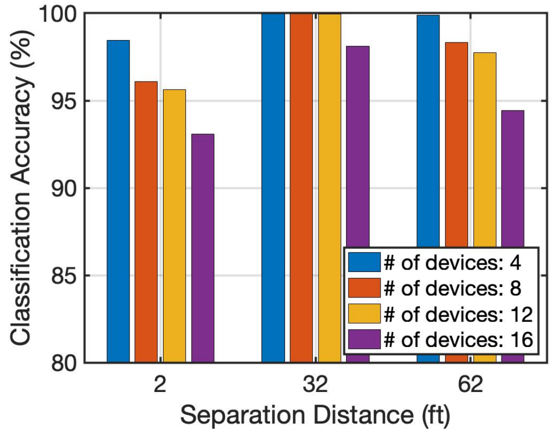
6.3.2 Classification Accuracy

The reservoir layer was initialized with 4,096 neurons, a dropout rate of 0.8, a leaky rate of 0.1, and a Gaussian noise of 0. Fig. 6.5a depicts the average classification accuracy with respect to various number of devices. It can be observed that more features are needed to be discriminated as the number of devices increases, and thus, resulting in a lower classification accuracy. Furthermore, as plotted in Fig. 6.5b, the average classification accuracy reaches 93.1%, 98.1%, and 94.4% when the transmitter-receiver separation distance is set to be 2, 32, and 62ft, respectively, in which the number of devices is kept at 16. The variation of classification accuracy with respect to different separation distances can be interpreted as the nonlinear distortion caused by the wireless transmission could impact RF signals. Rather, it can also be observed that a higher classification accuracy can be achieved as the number of neurons increases, and yet, a longer run-time is then required to train the network.

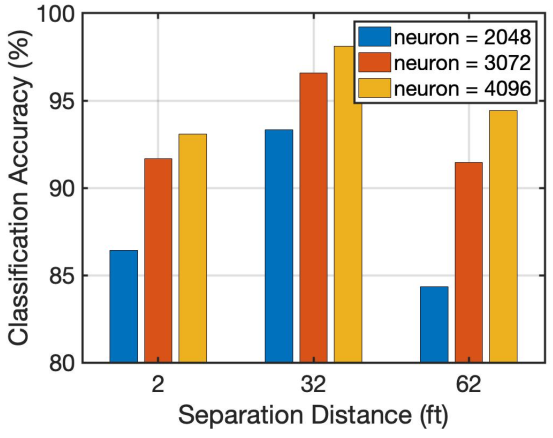
The average classification accuracy based on classical CSI location-based feature is approximately 92-97% [242]; therefore, it is reasonable to conclude that the identification strategy based on ESN is capable of achieving an identical classification accuracy when not considering security, mobility, and environment changes.

6.3.3 Stability Analysis

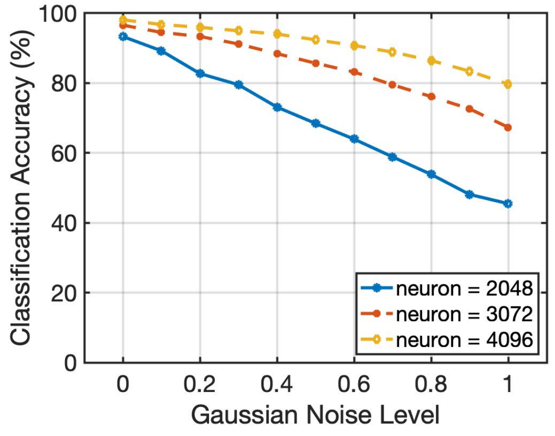
To evaluate the stability of ESN, a noise vector can be added to the reservoir dynamics. In this experiment, a Gaussian noise from 0 to 1 was introduced to the reservoir layer with respect to various number of neurons, while the transmitter-receiver separation distance was kept at 32ft. As demonstrated in Fig. 6.5c, the ESN is able to maintain a high classification accuracy above 90% with 4,096 neurons up to the Gaussian noise of 0.6. Unsurprisingly, as



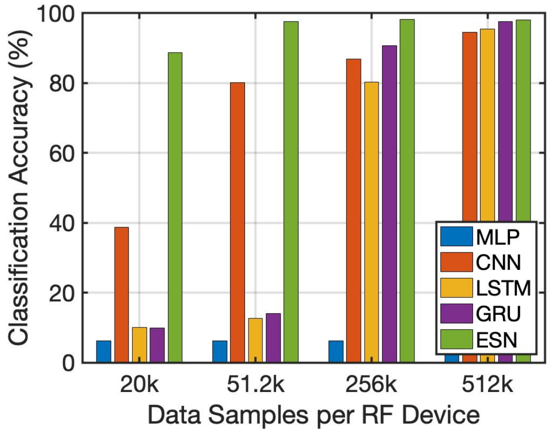
(a)



(b)



(c)



(d)

Figure 6.5: Classification accuracy on radio fingerprint dataset with respect to (a) various number of devices under different transmitter-receiver separation distances, (b) various number of neurons under different transmitter-receiver separation distances, (c) various number of neurons under different Gaussian noises, and (d) various number of training samples together with baseline deep neural network models.

the number of neurons is increased, so too does the network's resilience to noise. However, larger reservoir networks require 4× more time to train.

Since output weights of ESN are trainable in a single step, it is often possible to successfully train such a network with significantly fewer data. Such a capability is summarized in Fig.

Table 6.2: Performance comparison to baseline deep neural network models on radio fingerprint dataset.

	MLP	CNN	LSTM	GRU	ESN
Network Structure	2× dense	2× conv. 2× pool. 1× dense	1× rec.	1× rec.	1× reservoir
# of Kernels	–	$64 \times [2, 3]$	–	–	–
# of neurons	2× 512	512	512	512	various
Training Epochs	500	50	100	100	1
Classification Accuracy	6.25%	94.6%	95.4%	97.6%	98.1% @ 4,096 neurons 76.1% @ 512 neurons
Training Time	53mins	118mins	35mins	30mins	16mins @ 4,096 neurons 43secs @ 512 neurons

6.5d and Table 6.2 together with the baseline DNN models. From Fig. 6.5d, it can be observed that only the ESN has the capability to maintain its classification accuracy even with fewer training samples, while alternative models have a significant reduction. Even with 256,000 training samples, these alternative DNN models perform 8-12 percentage points poorer. To perform the same, the ESN had to be reduce to only 20,000 training samples. It can be also observed that the MLP is not trainable in such a temporal dataset.

The performance metric of ESN is summarized in Table 6.2 together with baseline DNN models. It can be observed that the ESN achieves the lowest classification accuracy when the number of neurons is set to be the same as in all other DNN models. To perform the same, the number of neurons in ESN had to be increased to 4,096. Despite that the ESN required 8× more neurons to approach an identical classification accuracy, such a simple training mechanism results 7.4× speedup over alternative DNN models. Beyond that, the performance comparison to the cutting-edge DNN models is summarized in Table 6.3, demonstrating that the ESN has the capability to achieve an identical classification accuracy even with a very limited training set.

Table 6.3: Classification accuracy comparison to the cutting-edge deep neural network models on radio fingerprint dataset.

	[224]	[243]	[244]	[245]	This Work
# of Devices	16	16	5	17	16
Samples per Device	2 million	576,000	720,000	–	51,200
Algorithm	CNN	CNN	CNN	CNN	ESN
Network Structure	8× conv. 4× pool. 2× dense	2× conv. 1× pool. 1× dense	2× conv. 1× pool. 1× dense	5× conv. 5× pool. 1× dense	1× reservoir
# of Kernels	128	50	50	128	–
# of Neurons	256 + 128	256	256	256	4,096
Classification Accuracy	98.6%	98.6%	98.0%	93.4%	98.1%

6.4 Clinical Disease Identification

In recent years, electronic medical services generate a colossal amount of heterogeneous biomedical data, from daily records of heart rate and blood pressure to images of magnetic resonance imaging (MRI) and computerized tomography (CT) scan. On one side, the increasing data volume provides an opportunity to improve the medical diagnostic. On the other hand, the increasing demand for medical diagnostic significantly reduces the diagnostic efficiency while consuming a large part of medical expenses.

The identification of a specific clinical disease based on symptoms and endocrine hormones is a primary procedure in medical diagnostic, creating a risk score system to identify patients who may need additional medical cares. It has been found that many clinical diseases, including the earlier stage of cancers, are highly related to the endocrine hormones [246–250], and can be observed by many symptoms and unusual changes of physical characteristics [251–255]. In most circumstances, the diagnostic efficiency is limited and errors are inevitable as complex medical diagnostics involve visual perception.

This is where the deep learning comes into help, in a way that is sufficient by learning from a colossal amount of data. From predicting the onset of clinical disease to identifying suspicious spots on tumors and brain bleeds, deep learning strategies have provided witnessed striking advances in medical applications [256–260]. Beyond that, arose with the introduction of IoT, a simple diagnostic procedure can be realized through a highly optimal machine learning algorithm deployed on wearable devices, improving the diagnostic efficiency while reducing the necessary medical expenses. To this end, multiple deep learning strategies are examined in this project to improve the classification performance of clinical diseases, including cardiovascular disease, breast cancer and coronavirus disease.

6.4.1 Cardiovascular Disease

Cardiovascular disease is the leading causes of death globally, killing approximately 17.9 million people worldwide annually [261], in particular, due to the heart failure caused by high blood pressure and diabetes. Given the importance of heart, it is essential for medical doctors to predict heart failure based on electronic health records from patients. In this context, deep learning strategies can be deployed to unveil the non-obvious correlations and relationships between patient's data.

In this experiment, multiple DNN models were trained to predict mortality based on medical records and vitals. The dataset used in this experiment contains 299 medical records from patients [262]. The collected features include clinical (*e.g.*, platelet, ejection fraction, etc.), physical (*e.g.*, age and gender) and lifestyle (*e.g.*, hypertension, diabetes, etc.) information, as described in Table 6.4. All patients are reported with left ventricular systolic dysfunction and previous heart failures. The objective of this project is to determine the possibility of heart failure for patients in the following year. During the experiment, 80% or samples were adopted for training and the rest were adopted for testing.

The average classification accuracy with multiple DNN models are plotted in Fig. 6.6 and summarized in Table 6.5. It can be observed that both ESN and STHNN outperform the alternative DNN models even with a fewer number of neurons, yielding an average classification accuracy up to 91.7% with merely 0.2s of training time. Under the scenario with

Table 6.4: Features drawn from medical records of patients on heart failure clinical records dataset.

Feature	Descriptions	Range
Age	age of patient	40 to 95
Gender	female or male	0 or 1
Hypertension	if a patient has hypertension	0 or 1
Smoking History	if a patient has smoking history	0 or 1
Diabetes	if a patient has diabetes	0 or 1
Anaemia	decrease of red blood cells or hemoglobin	0 or 1
Creatinine Phosphokinase	level of creatinine phosphokinase enzyme	23–to–$7,816$mcg/L
Ejection Fraction	percentage of blood leaving the heart at each contraction	14–to–80%
Platelets	amount of platelets in blood	25–to–850k/mL
Serum Creatinine	level of creatinine in blood	0.5–to–9.4mg/dL
Serum Sodium	level of sodium in blood	114–to–148mEq/L

Table 6.5: Performance comparison on heart failure clinical records dataset.

	MLP	LSTM	GRU	ESN	STHNN
Network Structure	2× dense	1× recurrent	1× recurrent	1× reservoir	1× dense 1× DFR 1× dense
# of neurons	2× 256	256	256	256	256
Training Epochs	20	20	20	1	1
Classification Accuracy	75.0%	77.1%	77.1%	91.7%	90.5%
Training Time	26s	35s	31s	0.2s	0.2s

Table 6.6: Performance comparison to the cutting-edge machine learning approaches on heart failure clinical records dataset.

	[263]					This Work
Algorithm	KNN	SVM	GB	RandF	LR	ESN
Classification Accuracy	62.4%	68.4%	73.8%	74.0%	83.3%	91.7%

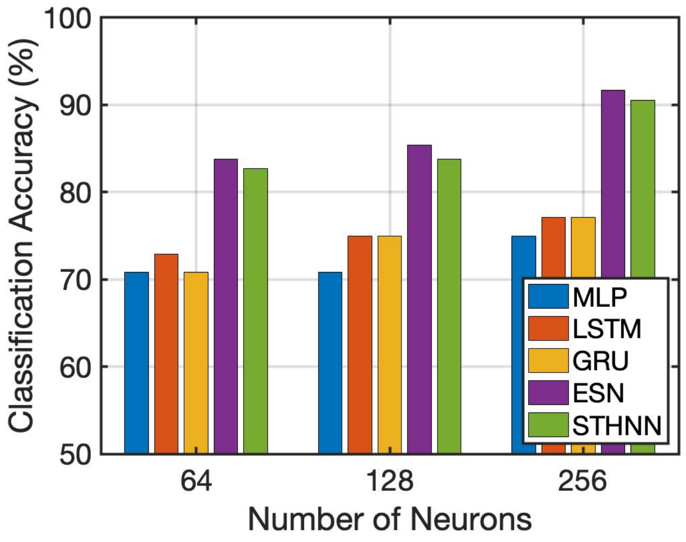


Figure 6.6: Classification accuracy with respect to various number of neurons on heart failure clinical records dataset.

64 neurons, both ESN and STHNN are still capable to achieve a classification accuracy up to 83.8%, which is approximately 13 percentage points better over alternative DNN models while yielding 175× speedup. Beyond that, the performance comparison to the cutting-edge machine learning approaches is summarized in Table 6.6, including k-nearest neighbors (KNN), support vector machine (SVM), gradient boosting (GB), random forests (RandF) and linear regression (LR). It can be observed from the comparison results that the introduced identification strategy based on deep learning approach demonstrates a better classification accuracy over the cutting-edge machine learning approaches.

Table 6.7: Features drawn from medical records of patients on breast cancer dataset.

Feature	Descriptions	Range
Age	age of patient	24 to 89
Body Mass Index	mass and height of patient	18.4–to–38.6kg/m^2
Glucose	level of sugar in blood	60–to–201mg/dL
Insulin	level of insulin in blood	2.4–to–58.7μU/mL
HOMA	level of insulin resistance in blood	0.5–to–25.0
Leptin	level of leptin in blood	4.3–to–90.3ng/mL
Adiponectin	level of adiponectin in blood	2.7–to–38.0μg/mL
Resistin	level of resistin	3.2–to–82.1ng/mL
MCP-1	level of monocyte chemoattractant protein in blood	45.8–to–1, 698.4pg/dL

6.4.2 Breast Cancer

Breast cancer is the most common cancer and is the second leading cause of cancer death globally, particular in women, in which 2.26 million people are diagnosed and over 685,000 deaths annually [264]. Similar as in the cardiovascular disease, it is essential to enable breast cancer screening for early detection, ensuring a greater probability of treatment. Routine consultation and blood analysis provide a robust predictive screening platform for medical doctors to predict breast cancer. In this context, DNN models can be also deployed for data analysis, discriminating the uniqueness between patient's data.

In this experiment, multiple DNN models were trained to predict the risk for breast cancer based on vitals and parameters collected from routine blood analysis. The dataset used in this experiment contains 166 medical records from patients [265]. The collected features include physical (*e.g.*, age) and haemal (*e.g.*, insulin, adiponectin, etc.) information, as described in Table 6.7. The distribution of percentage on training and testing samples was set to be the same as in the previous experiment.

Table 6.8: Performance comparison on breast cancer dataset.

	MLP	LSTM	GRU	ESN	STHNN
Network Structure	2× dense	1× recurrent	1× recurrent	1× reservoir	1× dense 1× DFR 1× dense
# of neurons	2× 256	256	256	256	256
Training Epochs	20	20	20	1	1
Classification Accuracy	68.0%	72.0%	72.0%	80.0%	78.0%
Training Time	25s	26s	24s	0.2s	0.2s

Table 6.9: Performance comparison to the cutting-edge machine learning approaches on breast cancer dataset.

	[265]			This Work
Algorithm	LR	SVM	RandF	ESN
Classification Accuracy	73.0%	78.0%	81.2%	80.0%

The average classification accuracy with multiple DNN models are plotted in Fig. 6.7 and summarized in Table 6.8. The trend on classification accuracy and training time is found be to the same as in the previous experiment on heart failure identification. The highest classification accuracy of 80% is reported by the ESN, while both ESN and STHNN demonstrate 130× speedup over alternative DNN models. Beyond that, the performance to the cutting-edge machine learning approaches is summarized in Table 6.9, demonstrating that both ESN and STHNN are capable to achieve a competitive classification accuracy even with a limited training set.

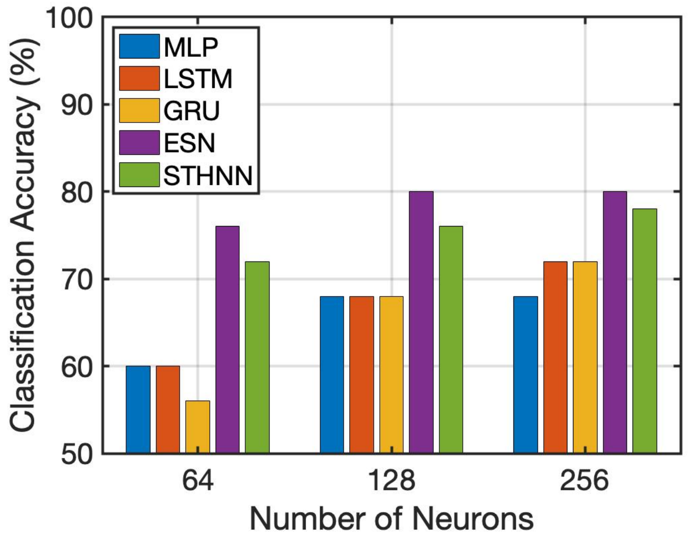


Figure 6.7: Classification accuracy with respect to various number of neurons on breast cancer dataset.

6.4.3 Coronavirus Disease

Following the coronavirus disease 2019 (COVID-19) since 2020, COVID-19 has spread to every country, causing 156 million confirmed cases and over 3.26 million deaths globally as of May 2021 (according to the Johns Hopkins Coronavirus Resource Center: `https://coronavirus.jhu.edu`). Beyond that, COVID-19 has become the third leading cause of death in United States [266]. Similar as in other diseases, an effective screening of COVID-19 would mitigate the burden on healthcare systems. As machine learning approaches are widely deployed to assist medical applications, the objective of this project is to predict the COVID-19 test results.

In this experiment, multiple DNN models were trained to predict the risk of COVID-19 based on simple features. The dataset used in this experiment contains 2.7 million test records from patients between March 2020 to November 2020 [267]. The collected features include physical (*e.g.*, age and gender), clinical (*e.g.*, appearance of symptoms) and lifestyle (*e.g.*, contact with infected individual and travel history) information, as described in Table 6.10. Features are only reported in three various states, for instance, true (1), false (0) or unknown (-1). The final results are either positive or negative, which have been confirmed

Table 6.10: Features drawn from medical records of patients on coronavirus dataset.

Feature	Descriptions	Range
Age	age of patient older than 60 years	0, 1 or -1
Gender	female, male or unreported	0, 1 or -1
Cough	symptom of cough	0, 1 or -1
Fever	symptom of fever	0, 1 or -1
Sore Throat	symptom of sore throat	0, 1 or -1
Shortness of Breath	symptom of shortness of breath	0, 1 or -1
Headache	symptom of headache	0, 1 or -1
Contact with Infected Individual	contact with infected individual or with travel history	0, 1 or -1

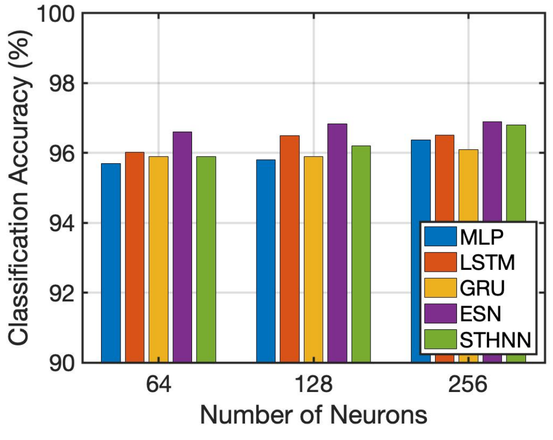
Table 6.11: Performance comparison on coronavirus dataset.

	MLP	LSTM	GRU	ESN	STHNN
Network Structure	2× dense	1× recurrent	1× recurrent	1× reservoir	1× dense 1× DFR 1× dense
# of neurons	2× 256	256	256	256	256
Training Epochs	10	10	10	1	1
Classification Accuracy	96.4%	96.5%	96.1%	96.9%	96.8%
Training Time	1min	2mins	1.8mins	1min	0.85mins

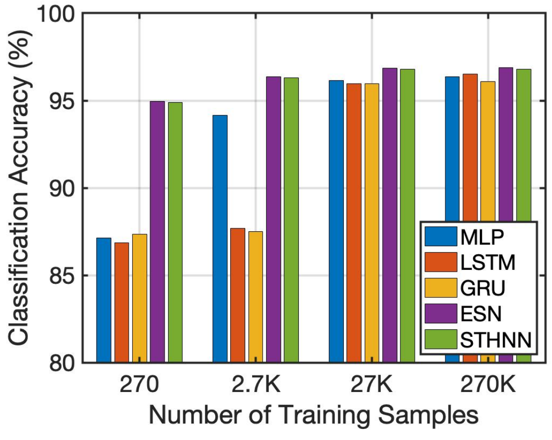
through the real-time polymerase chain reaction (RT-PCR) assay of a nasopharyngeal swab. The distribution of percentage on training and testing samples was set to be the same as in previous experiments.

Table 6.12: Performance comparison to the cutting-edge machine learning approaches on coronavirus dataset.

	[268]	This Work
Algorithm	GB	ESN
Classification Accuracy	90%	96.9%



(a)



(b)

Figure 6.8: Classification accuracy on coronavirus dataset with respect to (a) various number of neurons, (b) various number of training samples.

The average classification accuracy with multiple DNN models are plotted in Fig. 6.8 and summarized in Table 6.11. Due to the simple format of features, the average classification accuracy on all DNN models are identical, in which the gradient-based models (MLP, LSTM and GRU) can be trained with less epochs. Unsurprisingly, as the number of training samples is reduced, so too does the network's classification accuracy, and yet, both ESN and STHNN still have the capability to maintain their classification accuracy due to the unique training mechanism. Beyond that, the performance to a cutting-edge machine learning algorithm is summarized in Table 6.12.

6.5 Conclusions

In this work, multiple deep learning strategies are deployed for IoT applications, demonstrating their competitive performance in both communication and healthcare. The DESN-based modulation symbol detector improves the separability and the memory capacity of reservoir computing networks, yielding an average BER as low as 5.76×10^{-2}, whereby the robustness improvement reaches 47.73% over the state-of-the-art approaches in the literate for 5G MIMO-OFDM systems. Beyond that, by deploying the ESN as RF identifier, unique features between radio devices can be discriminated, yielding an average classification accuracy as high as 98.11% with $7.4\times$ speedup over alternative DNN models. What is more, numerical evaluations in healthcare applications also demonstrate the advantages of DNN over alternative machine learning approaches, yielding an average classification accuracy as high as 91.7%, 80.0% and 96.9% on the identification of cardiovascular disease, breast cancer and coronavirus disease, respectively. In summary, even with a very limited training set, deep learning strategies, particular with ESN, are still capable of achieving a competitive classification accuracy with reduced computing and hardware overhead.

Chapter 7

Conclusions

To support the integration of deep learning strategies and neuromorphic architecture, in this dissertation, a delay-feedback reservoir (DFR) network and two emerging hybrid neural computing architectures, the convolution-immersed DFR (Ci-DFR) network and the spatial-temporal hybrid neural network (STHNN), are introduced.

With a delay-feedback system embedded into the network, the introduced DFR network together with a fabricated prototype on 130nm process demonstrate a new design solution for reservoir computing network (RCN) and, to the best of our knowledge, is the first hardware implementation of DFR network in silicon. Beyond that, the introduced DFR network offers up to 6.79× error reduction on time series prediction and up to 26 percentage points more robust against noise on facial recognition. On the other hand, both Ci-DFR network and STHNN demonstrate a new design possibility for DFR network, exhibiting an error reduction up to 13.77× on time series prediction, approximately 2.5 percentage points classification accuracy improvement on spoken digit recognition, and a training acceleration up to 3.5× on handwritten digit classification. Beyond that, a fabricated prototype of STHNN on 180nm process demonstrates an on-chip classification capability for handprinted alphabet characters, yielding an average classification accuracy up to 86.1%. These emerging neural computing architectures demonstrate a promising solution to realize the next-generation computing platform for machine intelligence applications.

What is more, multiple deep learning strategies are deployed for internet of things (IoT) applications, demonstrating their competitive performance in both communication (*e.g.*, modulation symbol detection and radio fingerprint identification) and healthcare (*e.g.*, identification of cardiovascular disease, breast cancer and coronavirus disease). The resulting outcomes demonstrate the possibility of deploying deep learning strategies on IoT applications, opening the door for future mobile intelligence computing.

List of Publications

Journal Articles

1. **K. Bai**, L. Liu and Y. Yi, "Spatial-Temporal Hybrid Neural Network with Computing-in-Memory Architecture," in *IEEE Transactions on Circuits and Systems - I: Regular Papers*, 2021. (accepted)

2. C. Zhao, Q. An, **K. Bai**, B. Wysocki, C. Thiem, L. Liu and Y. Yi, "Energy Efficient Temporal Spatial Information Processing Circuits based on STDP and Spike Iteration," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 10, pp. 1715-1719, Oct. 2020.

3. **K. Bai**, Y. Yi, Z. Zhou, S. Jere and L. Liu, "Moving Toward Intelligence: Detecting Symbols on 5G Systems Through Deep Echo State Network," in *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 2, pp. 253-263, June 2020.

4. Q. An, **K. Bai** and Y. Yi, "A Unified Information Perceptron Using Deep Reservoir Computing," in *Elsevier Journal on Computers and Electrical Engineering*, vol. 85, July 2020.

5. **K. Bai**, Q. An, L. Liu and Y. Yi, "A Training-efficient Hybrid-structured Deep Neural Network with Reconfigurable Memristive Synapses," in *IEEE Transactions on Very Large Scale Integration (VLSI) System*, Vol. 28, No. 1, pp. 62-75, Jan. 2020.

6. **K. Bai** and Y. Yi, "DFR: An Energy-efficient Analog Delay Feedback Reservoir Computing System for Brain-inspired computing," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, Vol. 14, No. 4, pp. 1-22, December 2018.

Conference Proceedings

1. **K. Bai**, C. Thiem, N. McDonald, L. Loomis and Y. Yi, "Toward Intelligence in Communication Networks: A Deep Learning Identification Strategy for Radio Frequency Fingerprints," *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, 2020. (accepted)

2. H. An, **K. Bai** and Y. Yi, "Three-dimensional Memristive Deep Neural Network with Programmable Attention Mechanism," *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, 2020. (accepted)

3. H. Zheng, N. Mohammadi, **K. Bai** and Y. Yi, "Low-power Analog and Mixed-signal IC Design of Multiplexing Neural Encoder in Neuromorphic Computing," *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, 2020. (accepted)

4. **K. Bai**, L. Liu, Z. Zhou and Y. Yi, "Detection Through Deep Neural Networks: A Reservoir Computing Approach for MIMO-OFDM Symbol Detection," *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, San Diego, CA, USA, 2020, pp. 1-7.

5. Q. An, **K. Bai**, M. Zhang, Y. Yi and Y. Liu, "Deep Neural Network Based Speech Recognition Systems Under Noise Perturbations," *2020 21st International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, 2020, pp. 377-382.

6. **K. Bai**, S. Liu and Y. Yi, "High Speed and Energy-efficient Deep Neural Network for Edge Computing," *in proceedings of the 4th Symposium on Edge Computing (SEC)*, New York, NY, USA, 2019, pp. 347-349.

7. **K. Bai**, Q. An and Y. Yi, "Deep-DFR: A Memristive Deep Delayed Feedback Reservoir Computing System with Hybrid Neural Network Topology," 2019 56th ACM/IEEE Design Automation Conference (DAC), Las Vegas, NV, USA, 2019, pp. 1-6.

8. **K. Bai**, J. Li, K. Hamedani and Y. Yi, "Enabling An New Era of Brain-inspired Computing: Energy-efficient Spiking Neural Network with Ring Topology," 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2018, pp. 1-6.

9. J. Li, **K. Bai**, L. Liu and Y. Yi, "A deep learning based approach for analog hardware implementation of delayed feedback reservoir computing system," 2018 19th International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 2018, pp. 308-313.

10. **K. Bai** and Y. Y. Bradley, "A path to energy-efficient spiking delayed feedback reservoir computing system for brain-inspired neuromorphic processors," 2018 19th International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 2018, pp. 322-328.

Open Source Articles

1. Z. Zhou, **K. Bai**, N. Mohammadi, Y. Yi and L. Liu, "Making Intelligent Reflecting Surfaces More Intelligent: A Roadmap Through Reservoir Computing," *arXiv preprint*, arXiv: 2102.03688, Feb. 2021.
2. K. Hamedani, Z. Zhou, **K. Bai** and L. Liu, "The Novel Applications of Deep Reservoir Computing in Cyber-Security and Wireless Communication," *IntechOpen Intelligent System and Computing*, Nov. 2019.
3. **K. Bai** and Y. Yi, "Opening the "Black Box" of Silicon Chip Design in Neuromorphic Computing," *IntechOpen Bio-inspired Technology*, Mar. 2019.
4. H. An, **K. Bai**, and Y. Yi, "The Roadmap to Realize Memristive Three-dimensional Neuromorphic Computing System," *IntechOpen Advances in Memristor Neural Networks Modeling and Applications*, Oct. 2018.

References

[1] C.-h. Chen, *Handbook of pattern recognition and computer vision.* World Scientific, 2015.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] S. Samarasinghe, *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition.* Crc Press, 2016.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning.* MIT press, 2016.

[5] M. Dayarathna, Y. Wen, and R. Fan, "Data center energy consumption modeling: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 732–794, 2015.

[6] S. Yu, Y. Wu, R. Jeyasingh, D. Kuzum, and H.-S. P. Wong, "An electronic synapse device based on metal oxide resistive switching memory for neuromorphic computation," *IEEE Transactions on Electron Devices*, vol. 58, no. 8, pp. 2729–2737, 2011.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[8] W. Xiong, L. Wu, F. Alleva, J. Droppo, X. Huang, and A. Stolcke, "The microsoft 2017 conversational speech recognition system," in *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP).* IEEE, 2018, pp. 5934–5938.

[9] L. Deng, G. Wang, G. Li, S. Li, L. Liang, M. Zhu, Y. Wu, Z. Yang, Z. Zou, J. Pei *et al.*, "Tianjic: A unified and scalable chip bridging spike-based and continuous neural computation," *IEEE Journal of Solid-State Circuits*, vol. 55, no. 8, pp. 2228–2246, 2020.

[10] A. Ghani, T. McGinnity, L. Maguire, L. McDaid, A. Belatreche, and N. Shabtai, "Neuro-inspired speech recognition based on reservoir computing," *Advances in Speech Recognition*, vol. 164, 2010.

[11] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[12] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning.* PMLR, 2015, pp. 2048–2057.

[13] C. Mead, “Neuromorphic electronic systems,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[14] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam *et al.*, “Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip,” *IEEE transactions on computer-aided design of integrated circuits and systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[15] J. Sawada, F. Akopyan, A. S. Cassidy, B. Taba, M. V. Debole, P. Datta, R. Alvarez-Icaza, A. Amir, J. V. Arthur, A. Andreopoulos *et al.*, “Truenorth ecosystem for brain-inspired computing: scalable systems, software, and applications,” in *SC’16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis.* IEEE, 2016, pp. 130–141.

[16] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, “Loihi: A neuromorphic manycore processor with on-chip learning,” *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[17] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, “Eie: efficient inference engine on compressed deep neural network,” *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[18] J. Pei, L. Deng, S. Song, M. Zhao, Y. Zhang, S. Wu, G. Wang, Z. Zou, Z. Wu, W. He *et al.*, “Towards artificial general intelligence with hybrid tianjic chip architecture,” *Nature*, vol. 572, no. 7767, pp. 106–111, 2019.

[19] J. Von Neumann and R. Kurzweil, *The computer and the brain.* Yale University Press, 2012.

[20] D. Geer, “Chip makers turn to multicore processors,” *Computer*, vol. 38, no. 5, pp. 11–13, 2005.

[21] J. Hasler and H. B. Marr, “Finding a roadmap to achieve large neuromorphic hardware systems,” *Frontiers in neuroscience*, vol. 7, p. 118, 2013.

[22] D. Harris and S. Harris, *Digital design and computer architecture.* Morgan Kaufmann, 2010.

[23] J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach.* Elsevier, 2011.

[24] R. Preissl, T. M. Wong, P. Datta, M. Flickner, R. Singh, S. K. Esser, W. P. Risk, H. D. Simon, and D. S. Modha, “Compass: A scalable simulator for an architecture for cognitive computing,” in *SC’12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis.* IEEE, 2012, pp. 1–11.

[25] E. Painkras, L. A. Plana, J. Garside, S. Temple, F. Galluppi, C. Patterson, D. R. Lester, A. D. Brown, and S. B. Furber, “Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation,” *IEEE Journal of Solid-State Circuits*, vol. 48, no. 8, pp. 1943–1953, 2013.

[26] B. V. Benjamin, P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, “Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations,” *Proceedings of the IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[27] K. Bai and Y. Yi, "Dfr: An energy-efficient analog delay feedback reservoir computing system for brain-inspired computing," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, no. 4, pp. 1–22, 2018.

[28] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, "A survey of neuromorphic computing and neural networks in hardware," *arXiv preprint arXiv:1705.06963*, 2017.

[29] A. Joubert, B. Belhadj, O. Temam, and R. Héliot, "Hardware spiking neurons design: Analog or digital?" in *The 2012 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2012, pp. 1–5.

[30] L. S. Smith, "Neuromorphic systems: past, present and future," *Brain Inspired Cognitive Systems 2008*, pp. 167–182, 2010.

[31] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm," in *2011 IEEE custom integrated circuits conference (CICC)*. IEEE, 2011, pp. 1–4.

[32] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Frontiers in neuroscience*, vol. 9, p. 141, 2015.

[33] K. Ovtcharov, O. Ruwase, J.-Y. Kim, J. Fowers, K. Strauss, and E. S. Chung, "Accelerating deep convolutional neural networks using specialized hardware," *Microsoft Research Whitepaper*, vol. 2, no. 11, pp. 1–4, 2015.

[34] G. Indiveri, F. Corradi, and N. Qiao, "Neuromorphic architectures for spiking deep neural networks," in *2015 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2015, pp. 4–2.

[35] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[36] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun *et al.*, "Dadiannao: A machine-learning supercomputer," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE, 2014, pp. 609–622.

[37] Y. Chen, T. Chen, Z. Xu, N. Sun, and O. Temam, "Diannao family: energy-efficient hardware accelerators for machine learning," *Communications of the ACM*, vol. 59, no. 11, pp. 105–112, 2016.

[38] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, "Overview of the spinnaker system architecture," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2454–2467, 2012.

[39] S. K. Esser, A. Andreopoulos, R. Appuswamy, P. Datta, D. Barch, A. Amir, J. Arthur, A. Cassidy, M. Flickner, P. Merolla *et al.*, "Cognitive computing systems: Algorithms and applications for networks

of neurosynaptic cores," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013, pp. 1–10.

[40] T. Pfeil, A. Grübl, S. Jeltsch, E. Müller, P. Müller, M. A. Petrovici, M. Schmuker, D. Brüderle, J. Schemmel, and K. Meier, "Six networks on a universal neuromorphic computing substrate," *Frontiers in neuroscience*, vol. 7, p. 11, 2013.

[41] J.-s. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha *et al.*, "A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *2011 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2011, pp. 1–4.

[42] H. De Garis, C. Shuo, B. Goertzel, and L. Ruiting, "A world survey of artificial brain projects, part i: Large-scale brain simulations," *Neurocomputing*, vol. 74, no. 1-3, pp. 3–29, 2010.

[43] B. Goertzel, R. Lian, I. Arel, H. De Garis, and S. Chen, "A world survey of artificial brain projects, part ii: Biologically inspired cognitive architectures," *Neurocomputing*, vol. 74, no. 1-3, pp. 30–49, 2010.

[44] W. Gerstner and R. Naud, "How good are neuron models?" *Science*, vol. 326, no. 5951, pp. 379–380, 2009.

[45] J. Schemmel, J. Fieres, and K. Meier, "Wafer-scale integration of analog neural networks," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 431–438.

[46] M. Versace and B. Chandler, "The brain of a new machine," *IEEE spectrum*, vol. 47, no. 12, pp. 30–37, 2010.

[47] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[49] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, Red Hook, NY, USA, 2013, p. 2553–2561.

[50] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2147–2154.

[51] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[52] H. Jiang, W. Zhu, F. Luo, K. Bai, C. Liu, X. Zhang, J. J. Yang, Q. Xia, Y. Chen, and Q. Wu, "Cyclical sensing integrate-and-fire circuit for memristor array based neuromorphic computing," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 930–933.

[53] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[54] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[55] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[56] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 675–678.

[57] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[58] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *International journal of neural systems*, vol. 19, no. 04, pp. 295–308, 2009.

[59] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: opportunities and challenges," *Frontiers in neuroscience*, vol. 12, p. 774, 2018.

[60] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, "Swat: a spiking neural network training algorithm for classification problems," *IEEE transactions on neural networks*, vol. 21, no. 11, pp. 1817–1830, 2010.

[61] S. Schliebs and N. Kasabov, "Evolving spiking neural network—a survey," *Evolving Systems*, vol. 4, no. 2, pp. 87–98, 2013.

[62] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen, "A large-scale model of the functioning brain," *science*, vol. 338, no. 6111, pp. 1202–1205, 2012.

[63] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in neuroscience*, vol. 12, p. 331, 2018.

[64] N. K. Kasabov, "Neucube: A spiking neural network architecture for mapping, learning and understanding of spatio-temporal brain data," *Neural Networks*, vol. 52, pp. 62–76, 2014.

[65] K. Roy, A. Jaiswal, and P. Panda, "Towards spike-based machine intelligence with neuromorphic computing," *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[66] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv preprint arXiv:1506.00019*, 2015.

[67] M. I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Artificial neural networks: concept learning*, 1990, pp. 112–127.

[68] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[69] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[70] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[71] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[72] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[73] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE international conference on acoustics, speech and signal processing.* Ieee, 2013, pp. 6645–6649.

[74] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *arXiv preprint arXiv:1409.3215*, 2014.

[75] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[76] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proceedings*, vol. 89. Presses universitaires de Louvain, 2015, pp. 89–94.

[77] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," *arXiv preprint arXiv:1601.06733*, 2016.

[78] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[79] H. Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, no. 34, p. 13, 2001.

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

[81] J. J. Steil, "Backpropagation-decorrelation: online recurrent learning with o (n) complexity," in *2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541)*, vol. 2. IEEE, 2004, pp. 843–848.

[82] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *science*, vol. 304, no. 5667, pp. 78–80, 2004.

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

[84] D. V. Buonomano and W. Maass, "State-dependent computations: spatiotemporal processing in cortical networks," *Nature Reviews Neuroscience*, vol. 10, no. 2, pp. 113–125, 2009.

[85] W. Maass, P. Joshi, and E. D. Sontag, "Computational aspects of feedback in neural circuits," *PLoS Comput Biol*, vol. 3, no. 1, p. e165, 2007.

[86] M. Rabinovich, R. Huerta, and G. Laurent, "Transient dynamics for neural processing," *Science*, pp. 48–50, 2008.

[87] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, "Re-visiting the echo state property," *Neural networks*, vol. 35, pp. 1–9, 2012.

[88] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, "Information processing using a single dynamical node as complex system," *Nature communications*, vol. 2, no. 1, pp. 1–6, 2011.

[89] D. Verstraeten, B. Schrauwen, D. Stroobandt, and J. Van Campenhout, "Isolated word recognition with the liquid state machine: a case study," *Information Processing Letters*, vol. 95, no. 6, pp. 521–528, 2005.

[90] M. D. Skowronski and J. G. Harris, "Automatic speech recognition using a predictive echo state network classifier," *Neural networks*, vol. 20, no. 3, pp. 414–423, 2007.

[91] A. Ghani, T. M. McGinnity, L. P. Maguire, and J. Harkin, "Neuro-inspired speech recognition with recurrent spiking neurons," in *International Conference on Artificial Neural Networks.* Springer, 2008, pp. 513–522.

[92] H. Jaeger, "Discovering multiscale dynamical features with hierarchical echo state networks," Jacobs University Bremen, Tech. Rep., 2007.

[93] Y. Jin, Q. Zhao, H. Yin, and H. Yue, "Handwritten numeral recognition utilizing reservoir computing subject to optoelectronic feedback," in *2015 11th International Conference on Natural Computation (ICNC).* IEEE, 2015, pp. 1165–1169.

[94] D. Verstraeten, B. Schrauwen, and D. Stroobandt, "Reservoir computing with stochastic bitstream neurons," in *Proceedings of the 16th annual Prorisc workshop*, 2005, pp. 454–459.

[95] X. Hinaut and P. F. Dominey, "On-line processing of grammatical structure using reservoir computing," in *International Conference on Artificial Neural Networks.* Springer, 2012, pp. 596–603.

[96] A. Goudarzi, M. R. Lakin, and D. Stefanovic, "Reservoir computing approach to robust computation using unreliable nanoscale networks," in *International Conference on Unconventional Computation and Natural Computation.* Springer, 2014, pp. 164–176.

[97] J. Hertzberg, H. Jaeger, and F. Schönherr, "Learning to ground fact symbols in behavior-based robots," in *ECAI*, vol. 2, 2002, pp. 708–712.

[98] H. Jaeger, "Reservoir riddles: Suggestions for echo state network research," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 3. IEEE, 2005, pp. 1460–1462.

[99] E. A. Antonelo, B. Schrauwen, and D. Stroobandt, "Event detection and localization for small mobile robots using reservoir computing," *Neural Networks*, vol. 21, no. 6, pp. 862–871, 2008.

[100] E. Antonelo, B. Schrauwen, and D. Stroobandt, "Modeling multiple autonomous robot behaviors and behavior switching with a single reservoir computing network," in *2008 IEEE International Conference on Systems, Man and Cybernetics.* IEEE, 2008, pp. 1843–1848.

[101] D. Myers and R. Hutchinson, "Efficient implementation of piecewise linear activation function for digital vlsi neural networks," *Electronics Letters*, vol. 25, p. 1662, 1989.

[102] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets.* Springer, 1982, pp. 267–285.

[103] L. Lu, Y. Shin, Y. Su, and G. E. Karniadakis, "Dying relu and initialization: Theory and numerical examples," *arXiv preprint arXiv:1903.06733*, 2019.

[104] S. Haykin and N. Network, "A comprehensive foundation," *Neural networks*, vol. 2, no. 2004, p. 41, 2004.

[105] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[106] B. Hanin, "Which neural net architectures give rise to exploding and vanishing gradients?" *arXiv preprint arXiv:1801.03744*, 2018.

[107] X.-W. Chen and X. Lin, "Big data deep learning: challenges and perspectives," *IEEE access*, vol. 2, pp. 514–525, 2014.

[108] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Information Fusion*, vol. 42, pp. 146–157, 2018.

[109] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of big data*, vol. 2, no. 1, pp. 1–21, 2015.

[110] J. Gantz and D. Reinsel, "The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east," *IDC iView: IDC Analyze the future*, vol. 2007, no. 2012, pp. 1–16, 2012.

[111] X. Lin, Z. Yang, and Y. Song, "Intelligent stock trading system based on improved technical analysis and echo state network," *Expert systems with Applications*, vol. 38, no. 9, pp. 11 347–11 354, 2011.

[112] Q. Wang, Y. Li, and P. Li, "Liquid state machine based pattern recognition on fpga with firing-activity dependent power gating and approximate computing," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2016, pp. 361–364.

[113] Y. Zhang, P. Li, Y. Jin, and Y. Choe, "A digital liquid state machine with biologically inspired learning and its application to speech recognition," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 11, pp. 2635–2649, 2015.

[114] Y. Jin and P. Li, "Performance and robustness of bio-inspired digital liquid state machines: A case study of speech recognition," *Neurocomputing*, vol. 226, pp. 145–160, 2017.

[115] M. G. Kitzbichler, M. L. Smith, S. R. Christensen, and E. Bullmore, "Broadband criticality of human brain network synchronization," *PLoS Comput Biol*, vol. 5, no. 3, p. e1000314, 2009.

[116] C. G. Langton, "Computation at the edge of chaos: Phase transitions and emergent computation," *Physica D: Nonlinear Phenomena*, vol. 42, no. 1-3, pp. 12–37, 1990.

[117] N. Bertschinger and T. Natschläger, "Real-time computation at the edge of chaos in recurrent neural networks," *Neural computation*, vol. 16, no. 7, pp. 1413–1436, 2004.

[118] R. Legenstein and W. Maass, "Edge of chaos and prediction of computational performance for neural circuit models," *Neural networks*, vol. 20, no. 3, pp. 323–334, 2007.

[119] ——, "What makes a dynamical system computationally powerful," *New directions in statistical signal processing: From systems to brain*, pp. 127–154, 2007.

[120] T. Erneux, *Applied delay differential equations.* Springer Science & Business Media, 2009, vol. 3.

[121] L. Chen and K. Aihara, "Stability of genetic regulatory networks with time delay," *IEEE Transactions on circuits and systems I: Fundamental Theory and Applications*, vol. 49, no. 5, pp. 602–608, 2002.

[122] K. Bai, Q. An, L. Liu, and Y. Yi, "A training-efficient hybrid-structured deep neural network with reconfigurable memristive synapses," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 1, pp. 62–75, 2020.

[123] B. Schrauwen, D. Stroobandt *et al.*, "Using reservoir computing in a decomposition approach for time series prediction," in *ESTSP 2008 European Symposium on Time Series Prediction.* Multiprint Oy/Otamedia, 2008, pp. 149–158.

[124] M. L. Alomar, V. Canals, N. Perez-Mora, V. Martínez-Moll, and J. L. Rosselló, "Fpga-based stochastic echo state networks for time-series forecasting," *Computational intelligence and neuroscience*, vol. 2016, 2016.

[125] M. C. Soriano, S. Ortín, L. Keuninckx, L. Appeltant, J. Danckaert, L. Pesquera, and G. Van der Sande, "Delay-based reservoir computing: noise effects in a combined analog and digital implementation," *IEEE transactions on neural networks and learning systems*, vol. 26, no. 2, pp. 388–393, 2014.

[126] K. E. Callan, L. Illing, Z. Gao, D. J. Gauthier, and E. Schöll, "Broadband chaos generated by an optoelectronic oscillator," *Physical review letters*, vol. 104, no. 11, p. 113901, 2010.

[127] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nature communications*, vol. 4, no. 1, pp. 1–7, 2013.

[128] L. Larger, M. C. Soriano, D. Brunner, L. Appeltant, J. M. Gutiérrez, L. Pesquera, C. R. Mirasso, and I. Fischer, "Photonic information processing beyond turing: an optoelectronic implementation of reservoir computing," *Optics express*, vol. 20, no. 3, pp. 3241–3249, 2012.

[129] L. Larger and J. M. Dudley, "Optoelectronic chaos," *Nature*, vol. 465, no. 7294, pp. 41–42, 2010.

[130] Y. Paquot, F. Duport, A. Smerieri, J. Dambre, B. Schrauwen, M. Haelterman, and S. Massar, "Optoelectronic reservoir computing," *Scientific reports*, vol. 2, no. 1, pp. 1–6, 2012.

[131] M. C. Soriano, S. Ortín, D. Brunner, L. Larger, C. R. Mirasso, I. Fischer, and L. Pesquera, "Optoelectronic reservoir computing: tackling noise-induced performance degradation," *Optics express*, vol. 21, no. 1, pp. 12–20, 2013.

[132] F. Duport, B. Schneider, A. Smerieri, M. Haelterman, and S. Massar, "All-optical reservoir computing," *Optics express*, vol. 20, no. 20, pp. 22 783–22 795, 2012.

[133] L. Larger, A. Baylón-Fuentes, R. Martinenghi, V. S. Udaltsov, Y. K. Chembo, and M. Jacquot, "High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification," *Physical Review X*, vol. 7, no. 1, p. 011015, 2017.

[134] C. Zhao, B. T. Wysocki, Y. Liu, C. D. Thiem, N. R. McDonald, and Y. Yi, "Spike-time-dependent encoding for neuromorphic processors," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 12, no. 3, pp. 1–21, 2015.

[135] C. Zhao, B. T. Wysocki, C. D. Thiem, N. R. McDonald, J. Li, L. Liu, and Y. Yi, "Energy efficient spiking temporal encoder design for neuromorphic computing systems," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 4, pp. 265–276, 2016.

[136] C. Zhao, Y. Yi, J. Li, X. Fu, and L. Liu, "Interspike-interval-based analog spike-time-dependent encoder for neuromorphic processors," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 8, pp. 2193–2205, 2017.

[137] P. Amil, C. Cabeza, and A. C. Marti, "Exact discrete-time implementation of the mackey–glass delayed model," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 62, no. 7, pp. 681–685, 2015.

[138] Y. Hong, "Hypothesis testing in time series via the empirical characteristic function: a generalized spectral density approach," *Journal of the American Statistical Association*, vol. 94, no. 448, pp. 1201–1220, 1999.

[139] A. Goudarzi, P. Banda, M. R. Lakin, C. Teuscher, and D. Stefanovic, "A comparative study of reservoir computing for temporal signal processing," *arXiv preprint arXiv:1401.2224*, 2014.

[140] S. Ortín and L. Pesquera, "Reservoir computing with an ensemble of time-delay reservoirs," *Cognitive Computation*, vol. 9, no. 3, pp. 327–336, 2017.

[141] A. Rodan and P. Tino, "Minimum complexity echo state network," *IEEE transactions on neural networks*, vol. 22, no. 1, pp. 131–144, 2010.

[142] N. Gourier, D. Hall, and J. L. Crowley, "Estimating face orientation from robust detection of salient facial features," in *ICPR International Workshop on Visual Observation of Deictic Gestures.* Citeseer, 2004.

[143] S. P. Adhikari, H. Kim, R. K. Budhathoki, C. Yang, and L. O. Chua, "A circuit-based learning architecture for multilayer neural networks with memristor bridge synapses," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 62, no. 1, pp. 215–223, 2014.

[144] L. Danial, E. Pikhay, E. Herbelin, N. Wainstein, V. Gupta, N. Wald, Y. Roizin, R. Daniel, and S. Kvatinsky, "Two-terminal floating-gate transistors with a low-power memristive operation mode for analogue neuromorphic computing," *Nature Electronics*, vol. 2, no. 12, pp. 596–605, 2019.

[145] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[146] Q. Xia, W. Robinett, M. W. Cumbie, N. Banerjee, T. J. Cardinali, J. J. Yang, W. Wu, X. Li, W. M. Tong, D. B. Strukov *et al.*, "Memristor- cmos hybrid integrated circuits for reconfigurable logic," *Nano letters*, vol. 9, no. 10, pp. 3640–3645, 2009.

[147] K.-H. Kim, S. Gaba, D. Wheeler, J. M. Cruz-Albrecht, T. Hussain, N. Srinivasa, and W. Lu, "A functional hybrid memristor crossbar-array/cmos system for data storage and neuromorphic applications," *Nano letters*, vol. 12, no. 1, pp. 389–395, 2012.

[148] H.-S. P. Wong, S. Raoux, S. Kim, J. Liang, J. P. Reifenberg, B. Rajendran, M. Asheghi, and K. E. Goodson, "Phase change memory," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2201–2227, 2010.

[149] S. Lai, "Current status of the phase change memory and its future," in *IEEE International Electron Devices Meeting 2003.* IEEE, 2003, pp. 10–1.

[150] G. W. Burr, M. J. Breitwisch, M. Franceschini, D. Garetto, K. Gopalakrishnan, B. Jackson, B. Kurdi, C. Lam, L. A. Lastras, A. Padilla *et al.*, "Phase change memory technology," *Journal of Vacuum Science & Technology B, Nanotechnology and Microelectronics: Materials, Processing, Measurement, and Phenomena*, vol. 28, no. 2, pp. 223–262, 2010.

[151] R. Simpson, P. Fons, A. Kolobov, T. Fukaya, M. Krbal, T. Yagi, and J. Tominaga, "Interfacial phase-change memory," *Nature nanotechnology*, vol. 6, no. 8, pp. 501–505, 2011.

[152] M. K. Qureshi, V. Srinivasan, and J. A. Rivers, “Scalable high performance main memory system using phase-change memory technology,” in *Proceedings of the 36th annual international symposium on Computer architecture*, 2009, pp. 24–33.

[153] P. Zhou, B. Zhao, J. Yang, and Y. Zhang, “A durable and energy efficient main memory using phase change memory technology,” *ACM SIGARCH computer architecture news*, vol. 37, no. 3, pp. 14–23, 2009.

[154] T. Mikolajick, C. Dehm, W. Hartner, I. Kasko, M. Kastner, N. Nagel, M. Moert, and C. Mazure, “Feram technology for high density applications,” *Microelectronics Reliability*, vol. 41, no. 7, pp. 947–950, 2001.

[155] S. Dünkel, M. Trentzsch, R. Richter, P. Moll, C. Fuchs, O. Gehring, M. Majer, S. Wittek, B. Müller, T. Melde *et al.*, “A fefet based super-low-power ultra-fast embedded nvm technology for 22nm fdsoi and beyond,” in *2017 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2017, pp. 19–7.

[156] E. Yurchuk, J. Müller, S. Müller, J. Paul, M. Pešić, R. van Bentum, U. Schroeder, and T. Mikolajick, “Charge-trapping phenomena in hfo 2-based fefet-type nonvolatile memories,” *IEEE Transactions on Electron Devices*, vol. 63, no. 9, pp. 3501–3507, 2016.

[157] S. Mueller, J. Müller, R. Hoffmann, E. Yurchuk, T. Schlösser, R. Boschke, J. Paul, M. Goldbach, T. Herrmann, A. Zaka *et al.*, “From mfm capacitors toward ferroelectric transistors: Endurance and disturb characteristics of hfo$_2$-based fefet devices,” *IEEE transactions on electron devices*, vol. 60, no. 12, pp. 4199–4205, 2013.

[158] H. Shiga, D. Takashima, S.-i. Shiratake, K. Hoya, T. Miyakawa, R. Ogiwara, R. Fukuda, R. Takizawa, K. Hatsuda, F. Matsuoka *et al.*, “A 1.6 gb/s ddr2 128 mb chain feram with scalable octal bitline and sensing schemes,” *IEEE journal of solid-state circuits*, vol. 45, no. 1, pp. 142–152, 2009.

[159] M. Qazi, A. Amerasekera, and A. P. Chandrakasan, “A 3.4-pj feram-enabled d flip-flop in 0.13-μm cmos for nonvolatile processing in digital systems,” *IEEE Journal of Solid-State Circuits*, vol. 49, no. 1, pp. 202–211, 2013.

[160] B. Engel, J. Akerman, B. Butcher, R. Dave, M. DeHerrera, M. Durlam, G. Grynkewich, J. Janesky, S. Pietambaram, N. Rizzo *et al.*, “A 4-mb toggle mram based on a novel bit and switching method,” *IEEE Transactions on Magnetics*, vol. 41, no. 1, pp. 132–136, 2005.

[161] S. Tehrani, J. Slaughter, E. Chen, M. Durlam, J. Shi, and M. DeHerren, “Progress and outlook for mram technology,” *IEEE Transactions on Magnetics*, vol. 35, no. 5, pp. 2814–2819, 1999.

[162] Y. Huai *et al.*, “Spin-transfer torque mram (stt-mram): Challenges and prospects,” *AAPPS bulletin*, vol. 18, no. 6, pp. 33–40, 2008.

[163] D. Apalkov, A. Khvalkovskiy, S. Watts, V. Nikitin, X. Tang, D. Lottis, K. Moon, X. Luo, E. Chen, A. Ong *et al.*, “Spin-transfer torque magnetic random access memory (stt-mram),” *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 9, no. 2, pp. 1–35, 2013.

[164] E. Kültürsay, M. Kandemir, A. Sivasubramaniam, and O. Mutlu, "Evaluating stt-ram as an energy-efficient main memory alternative," in *2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS).* IEEE, 2013, pp. 256–267.

[165] K. Wang, J. Alzate, and P. K. Amiri, "Low-power non-volatile spintronic memory: Stt-ram and beyond," *Journal of Physics D: Applied Physics*, vol. 46, no. 7, p. 074003, 2013.

[166] H. Akinaga and H. Shima, "Resistive random access memory (reram) based on metal oxides," *Proceedings of the IEEE*, vol. 98, no. 12, pp. 2237–2251, 2010.

[167] Z. Wei, Y. Kanzawa, K. Arita, Y. Katoh, K. Kawai, S. Muraoka, S. Mitani, S. Fujii, K. Katayama, M. Iijima *et al.*, "Highly reliable taox reram and direct evidence of redox reaction mechanism," in *2008 IEEE International Electron Devices Meeting.* IEEE, 2008, pp. 1–4.

[168] L. Song, X. Qian, H. Li, and Y. Chen, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA).* IEEE, 2017, pp. 541–552.

[169] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 27–39, 2016.

[170] E. Gale, "Tio2-based memristors and reram: materials, mechanisms and models (a review)," *Semiconductor Science and Technology*, vol. 29, no. 10, p. 104004, 2014.

[171] J. Simmons and R. Verderber, "New conduction and reversible memory phenomena in thin insulating films," *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, vol. 301, no. 1464, pp. 77–102, 1967.

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

[173] S.-G. Park, B. Magyari-Kope, and Y. Nishi, "Impact of oxygen vacancy ordering on the formation of a conductive filament in tio$_2$ for resistive switching memory," *IEEE Electron Device Letters*, vol. 32, no. 2, pp. 197–199, 2010.

[174] H. Lee, P. Chen, T. Wu, Y. Chen, C. Wang, P. Tzeng, C. Lin, F. Chen, C. Lien, and M.-J. Tsai, "Low power and high speed bipolar switching with a thin reactive ti buffer layer in robust hfo2 based rram," in *2008 IEEE International Electron Devices Meeting.* IEEE, 2008, pp. 1–4.

[175] H. An, K. Bai, and Y. Yi, "The roadmap to realize memristive three-dimensional neuromorphic computing system," *Advances in Memristor Neural Networks-Modeling and Applications*, pp. 25–44, 2018.

[176] T. Hasegawa, T. Ohno, K. Terabe, T. Tsuruoka, T. Nakayama, J. K. Gimzewski, and M. Aono, "Learning abilities achieved by a single solid-state atomic switch," *Advanced materials*, vol. 22, no. 16, pp. 1831–1834, 2010.

[177] U. Ramacher and C. von der Malsburg, *On the construction of artificial brains.* Springer Science & Business Media, 2010.

[178] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[179] M. Sharad, C. Augustine, G. Panagopoulos, and K. Roy, "Proposal for neuromorphic hardware using spin devices," *arXiv preprint arXiv:1206.3227*, 2012.

[180] K. Bai and Y. Y. Bradley, "A path to energy-efficient spiking delayed feedback reservoir computing system for brain-inspired neuromorphic processors," in *2018 19th International Symposium on Quality Electronic Design (ISQED).* IEEE, 2018, pp. 322–328.

[181] K. Bai, J. Li, K. Hamedani, and Y. Yi, "Enabling an new era of brain-inspired computing: Energy-efficient spiking neural network with ring topology," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC).* IEEE, 2018, pp. 1–6.

[182] K. Bai and Y. Yi, "Opening the "black box" of silicon chip design in neuromorphic computing," in *Bio-Inspired Technology.* IntechOpen, 2019.

[183] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1t1m crossbar to accelerate matrix-vector multiplication," in *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC).* IEEE, 2016, pp. 1–6.

[184] M. Hu, H. Li, Q. Wu, and G. S. Rose, "Hardware realization of bsb recall function using memristor crossbar arrays," in *DAC Design Automation Conference 2012.* IEEE, 2012, pp. 498–503.

[185] M. C. Mackey and L. Glass, "Oscillation and chaos in physiological control systems," *Science*, vol. 197, no. 4300, pp. 287–289, 1977.

[186] T. W. Molter and M. A. Nugent, "The generalized metastable switch memristor model," in *CNNA 2016; 15th International Workshop on Cellular Nanoscale Networks and their Applications.* VDE, 2016, pp. 1–2.

[187] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[188] G. K. Chen, R. Kumar, H. E. Sumbul, P. C. Knag, and R. K. Krishnamurthy, "A 4096-neuron 1m-synapse 3.8-pj/sop spiking neural network with on-chip stdp learning and sparse weights in 10-nm finfet cmos," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 992–1002, 2018.

[189] D. Bankman, L. Yang, B. Moons, M. Verhelst, and B. Murmann, "An always-on 3.8uj 86% cifar-10 mixed-signal binary cnn processor with all memory on chip in 28-nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 158–172, 2018.

[190] S. K. Gonugondla, M. Kang, and N. Shanbhag, "A 42pj/decision 3.12 tops/w robust in-memory machine learning classifier with on-chip training," in *2018 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2018, pp. 490–492.

[191] J. Zhang, Z. Wang, and N. Verma, "In-memory computation of a machine-learning classifier in a standard 6t sram array," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 4, pp. 915–924, 2017.

[192] T. N. Chennai, "Daily temperature of major cities," https://www.kaggle.com/sudalairajkumar/daily-temperature-of-major-cities, 2020.

[193] D. Cook, "El nino data set," https://archive.ics.uci.edu/ml/datasets/El+Nino, 1999.

[194] X. Sun, T. Li, Q. Li, Y. Huang, and Y. Li, "Deep belief echo-state network and its application to time series prediction," *Knowledge-Based Systems*, vol. 130, pp. 17–29, 2017.

[195] X. Sun, T. Li, Y. Li, Q. Li, Y. Huang, and J. Liu, "Recurrent neural system with minimum complexity: A deep learning perspective," *Neurocomputing*, vol. 275, pp. 1333–1349, 2018.

[196] J. Chen, G. Augenbroe, and X. Song, "Lighted-weighted model predictive control for hybrid ventilation operation based on clusters of neural network models," *Automation in Construction*, vol. 89, pp. 250–265, 2018.

[197] P. D. Nooteboom, Q. Y. Feng, C. López, E. Hernández-García, and H. A. Dijkstra, "Using network theory and machine learning to predict el nino," *arXiv preprint arXiv:1803.10076*, 2018.

[198] Y. LeCun, C. Cortes, and C. Burges, "Mnist handwritten digit database," 2010.

[199] S. Yu, Z. Li, P.-Y. Chen, H. Wu, B. Gao, D. Wang, W. Wu, and H. Qian, "Binary neural network with 16 mb rram macro chip for classification and online training," in *2016 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2016, pp. 16–2.

[200] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier, "Combining stdp and reward-modulated stdp in deep convolutional spiking neural networks for digit recognition," *arXiv preprint arXiv:1804.00227*, 2018.

[201] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2017, pp. 65–74.

[202] M. Jerry, P.-Y. Chen, J. Zhang, P. Sharma, K. Ni, S. Yu, and S. Datta, "Ferroelectric fet analog synapse for acceleration of deep neural network training," in *2017 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2017, pp. 6–2.

[203] P. Warden, "Spoken digit commands," https://datarepository.wolframcloud.com/resources/Spoken-Digit-Commands-Dataset, 2018.

[204] A. Ardakani, C. Condo, M. Ahmadi, and W. J. Gross, "An architecture to accelerate convolution in deep neural networks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 4, pp. 1349–1362, 2017.

[205] K. Bai, Q. An, and Y. Yi, "Deep-dfr: A memristive deep delayed feedback reservoir computing system with hybrid neural network topology," in *2019 56th ACM/IEEE Design Automation Conference (DAC).* IEEE, 2019, pp. 1–6.

[206] Q. An, K. Bai, M. Zhang, Y. Yi, and Y. Liu, "Deep neural network based speech recognition systems under noise perturbations," in *2020 21st International Symposium on Quality Electronic Design (ISQED).* IEEE, 2020, pp. 377–382.

[207] K. Bai, Y. Yi, Z. Zhou, S. Jere, and L. Liu, "Moving toward intelligence: Detecting symbols on 5g systems through deep echo state network," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 10, no. 2, pp. 253–263, 2020.

[208] C. Liu, Q. Yang, C. Zhang, H. Jiang, Q. Wu, and H. H. Li, "A memristor-based neuromorphic engine with a current sensing scheme for artificial neural network applications," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC).* IEEE, 2017, pp. 647–652.

[209] P. Grother and K. Hanaoka, "Nist special database 19 handprinted forms and characters 2nd edition," *National Institute of Standards and Technology, Tech. Rep*, 2016.

[210] Q. Liu, B. Gao, P. Yao, D. Wu, J. Chen, Y. Pang, W. Zhang, Y. Liao, C.-X. Xue, W.-H. Chen *et al.*, "33.2 a fully integrated analog reram based 78.4 tops/w compute-in-memory chip with fully parallel mac computing," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC).* IEEE, 2020, pp. 500–502.

[211] W. Wan, R. Kubendran, S. B. Eryilmaz, W. Zhang, Y. Liao, D. Wu, S. Deiss, B. Gao, P. Raina, S. Joshi *et al.*, "33.1 a 74 tmacs/w cmos-rram neurosynaptic core with dynamically reconfigurable dataflow and in-situ transposable weights for probabilistic graphical models," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC).* IEEE, 2020, pp. 498–500.

[212] D. Pedamonti, "Comparison of non-linear activation functions for deep neural networks on mnist classification task," *arXiv preprint arXiv:1804.02763*, 2018.

[213] C. Yadav and L. Bottou, "Cold case: The lost mnist digits," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 443–13 452.

[214] H. Guo, S. Wang, J. Fan, and S. Li, "Learning automata based incremental learning method for deep neural networks," *IEEE Access*, vol. 7, pp. 41 164–41 171, 2019.

[215] N. Caporale and Y. Dan, "Spike timing–dependent plasticity: a hebbian learning rule," *Annu. Rev. Neurosci.*, vol. 31, pp. 25–46, 2008.

[216] S. R. Kheradpisheh, M. Ganjtabesh, S. J. Thorpe, and T. Masquelier, "Stdp-based spiking deep convolutional neural networks for object recognition," *Neural Networks*, vol. 99, pp. 56–67, 2018.

[217] T. Iakymchuk, A. Rosado-Muñoz, J. F. Guerrero-Martínez, M. Bataller-Mompeán, and J. V. Francés-Víllora, "Simplified spiking neural network architecture and stdp learning algorithm applied to image classification," *EURASIP Journal on Image and Video Processing*, vol. 2015, no. 1, pp. 1–11, 2015.

[218] H. Tanaka, T. Morie, and K. Aihara, "A cmos spiking neural network circuit with symmetric/asymmetric stdp function," *IEICE transactions on fundamentals of electronics, communications and computer sciences*, vol. 92, no. 7, pp. 1690–1698, 2009.

[219] I. E. Ebong and P. Mazumder, "Cmos and memristor-based neural network design for position detection," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 2050–2060, 2011.

[220] K. Cameron, V. Boonsobhak, A. Murray, and D. Renshaw, "Spike timing dependent plasticity (stdp) can ameliorate process variations in neuromorphic vlsi," *IEEE Transactions on Neural Networks*, vol. 16, no. 6, pp. 1626–1637, 2005.

[221] C. Zhao, Q. An, K. Bai, B. Wysocki, C. Thiem, L. Liu, and Y. Yi, "Energy efficient temporal spatial information processing circuits based on stdp and spike iteration," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 10, pp. 1715–1719, 2019.

[222] H. Kim, H. Tang, W. Choi, and J. Park, "An energy-quality scalable stdp based sparse coding processor with on-chip learning capability," *IEEE transactions on biomedical circuits and systems*, vol. 14, no. 1, pp. 125–137, 2020.

[223] C. V. N. Index, "Global mobile data traffic forecast update, 2016–2021," *white paper*, vol. 7, 2017.

[224] K. Sankhe, M. Belgiovine, F. Zhou, L. Angioloni, F. Restuccia, S. D'Oro, T. Melodia, S. Ioannidis, and K. Chowdhury, "No radio left behind: Radio fingerprinting through deep learning of physical-layer hardware impairments," *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 165–178, 2019.

[225] L. Du, Y. Du, Y. Li, J. Su, Y.-C. Kuan, C.-C. Liu, and M.-C. F. Chang, "A reconfigurable streaming deep convolutional neural network accelerator for internet of things," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 1, pp. 198–208, 2017.

[226] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.

[227] P. Li, Z. Chen, L. T. Yang, Q. Zhang, and M. J. Deen, "Deep convolutional computation model for feature learning on big data in internet of things," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 2, pp. 790–798, 2017.

[228] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K. R. Choo, "A deep recurrent neural network based approach for internet of things malware threat hunting," *Future Generation Computer Systems*, vol. 85, pp. 88–96, 2018.

[229] G. Hinton, "Deep learning—a technology with the potential to transform health care," *Jama*, vol. 320, no. 11, pp. 1101–1102, 2018.

[230] M. Chen, Y. Hao, K. Hwang, L. Wang, and L. Wang, "Disease prediction by machine learning over big data from healthcare communities," *Ieee Access*, vol. 5, pp. 8869–8879, 2017.

[231] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, “Closest point search in lattices,” *IEEE Transactions on Information Theory*, vol. 48, no. 8, pp. 2201–2214, Aug 2002.

[232] B. M. Hochwald and S. ten Brink, “Achieving near-capacity on a multiple-antenna channel,” *IEEE Transactions on Communications*, vol. 51, no. 3, pp. 389–399, March 2003.

[233] S. S. Mosleh, L. Liu, C. Sahin, Y. R. Zheng, and Y. Yi, “Brain-inspired wireless communications: Where reservoir computing meets MIMO-OFDM,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4694–4708, Oct 2018.

[234] R. Shafin, L. Liu, J. Ashdown, J. Matyjas, M. Medley, B. Wysocki, and Y. Yi, “Realizing green symbol detection via reservoir computing: An energy-efficiency perspective,” in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[235] D. Kudithipudi, Q. Saleh, C. Merkel, J. Thesing, and B. Wysocki, “Design and analysis of a neuromemristive reservoir computing architecture for biosignal processing,” *Frontiers in neuroscience*, vol. 9, p. 502, 2016.

[236] A. M. Hassan, H. H. Li, and Y. Chen, “Hardware implementation of echo state networks using memristor double crossbar arrays,” in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 2171–2177.

[237] 3GPP TS 38.211, “NR; Physical channels and modulation,” September 2019.

[238] J. Meinilä, P. Kyösti, T. Jämsä, and L. Hentilä, “Winner ii channel models,” *Radio Technologies and Concepts for IMT-Advanced*, pp. 39–92, 2009.

[239] H. Bojinov, Y. Michalevsky, G. Nakibly, and D. Boneh, “Mobile device identification via sensor fingerprinting,” *arXiv preprint arXiv:1408.1416*, 2014.

[240] D. Henrici and P. Muller, “Hash-based enhancement of location privacy for radio-frequency identification devices using varying identifiers,” in *IEEE Annual Conference on Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second*. IEEE, 2004, pp. 149–153.

[241] K. Ellis and N. Serinken, “Characteristics of radio transmitter fingerprints,” *Radio Science*, vol. 36, no. 4, pp. 585–597, 2001.

[242] L. N. Kandel, Z. Zhang, and S. Yu, “Exploiting csi-mimo for accurate and efficient device identification,” in *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2019, pp. 1–6.

[243] K. Sankhe, M. Belgiovine, F. Zhou, S. Riyaz, S. Ioannidis, and K. Chowdhury, “Oracle: Optimized radio classification through convolutional neural networks,” in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 370–378.

[244] S. Riyaz, K. Sankhe, S. Ioannidis, and K. Chowdhury, “Deep learning convolutional neural networks for radio identification,” *IEEE Communications Magazine*, vol. 56, no. 9, pp. 146–152, 2018.

[245] Y. Shi, K. Davaslioglu, Y. E. Sagduyu, W. C. Headley, M. Fowler, and G. Green, "Deep learning for rf signal classification in unknown and dynamic spectrum environments," in *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN).* IEEE, 2019, pp. 1–10.

[246] L. H. Klotz, H. W. Herr, M. J. Morse, and W. F. Whitmore Jr, "Intermittent endocrine therapy for advanced prostate cancer," *Cancer*, vol. 58, no. 11, pp. 2546–2550, 1986.

[247] L. J. Fallowfield, S. K. Leaity, A. Howell, S. Benson, and D. Cella, "Assessment of quality of life in women undergoing hormonal therapy for breast cancer: validation of an endocrine symptom subscale for the fact-b," *Breast cancer research and treatment*, vol. 55, no. 2, pp. 187–197, 1999.

[248] G. Schatzl, C. Brössner, S. Schmid, W. Kugler, M. Roehrich, T. Treu, A. Szalay, B. Djavan, C. P. Schmidbauer, S. Söregi *et al.*, "Endocrine status in elderly men with lower urinary tract symptoms: correlation of age, hormonal status, and lower urinary tract function," *Urology*, vol. 55, no. 3, pp. 397–402, 2000.

[249] H. Chahal and W. Drake, "The endocrine system and ageing," *The Journal of Pathology: A Journal of the Pathological Society of Great Britain and Ireland*, vol. 211, no. 2, pp. 173–180, 2007.

[250] S. Li, Z. Zhang, L. Wu, X. Zhang, Y. Li, and Y. Wang, "Understanding zheng in traditional chinese medicine in the context of neuro-endocrine-immune network," *IET systems biology*, vol. 1, no. 1, pp. 51–60, 2007.

[251] R. A. Aronowitz, "When do symptoms become a disease?" *Annals of internal medicine*, vol. 134, no. 9_Part_2, pp. 803–808, 2001.

[252] X. Zhou, J. Menche, A.-L. Barabási, and A. Sharma, "Human symptoms–disease network," *Nature communications*, vol. 5, no. 1, pp. 1–10, 2014.

[253] M. Politis, K. Wu, S. Molloy, P. G. Bain, K. R. Chaudhuri, and P. Piccini, "Parkinson's disease symptoms: the patient's perspective," *Movement Disorders*, vol. 25, no. 11, pp. 1646–1651, 2010.

[254] S. Sveinbjornsdottir, "The clinical symptoms of parkinson's disease," *Journal of neurochemistry*, vol. 139, pp. 318–324, 2016.

[255] K. R. Chaudhuri, D. G. Healy, and A. H. Schapira, "Non-motor symptoms of parkinson's disease: diagnosis and management," *The Lancet Neurology*, vol. 5, no. 3, pp. 235–245, 2006.

[256] A. Esteva, A. Robicquet, B. Ramsundar, V. Kuleshov, M. DePristo, K. Chou, C. Cui, G. Corrado, S. Thrun, and J. Dean, "A guide to deep learning in healthcare," *Nature medicine*, vol. 25, no. 1, pp. 24–29, 2019.

[257] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in bioinformatics*, vol. 19, no. 6, pp. 1236–1246, 2018.

[258] O. Faust, Y. Hagiwara, T. J. Hong, O. S. Lih, and U. R. Acharya, "Deep learning for healthcare applications based on physiological signals: A review," *Computer methods and programs in biomedicine*, vol. 161, pp. 1–13, 2018.

[259] T. Pham, T. Tran, D. Phung, and S. Venkatesh, "Predicting healthcare trajectories from medical records: A deep learning approach," *Journal of biomedical informatics*, vol. 69, pp. 218–229, 2017.

[260] S. Purushotham, C. Meng, Z. Che, and Y. Liu, "Benchmarking deep learning models on large healthcare datasets," *Journal of biomedical informatics*, vol. 83, pp. 112–134, 2018.

[261] K. Mc Namara, H. Alzubaidi, and J. K. Jackson, "Cardiovascular disease as a leading cause of death: how are pharmacists getting involved?" *Integrated pharmacy research & practice*, vol. 8, p. 1, 2019.

[262] T. Ahmad, A. Munir, S. H. Bhatti, M. Aftab, and M. A. Raza, "Survival analysis of heart failure patients: A case study," *PloS one*, vol. 12, no. 7, p. e0181001, 2017.

[263] D. Chicco and G. Jurman, "Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone," *BMC medical informatics and decision making*, vol. 20, no. 1, p. 16, 2020.

[264] A. G. Waks and E. P. Winer, "Breast cancer treatment: a review," *Jama*, vol. 321, no. 3, pp. 288–300, 2019.

[265] M. Patrício, J. Pereira, J. Crisóstomo, P. Matafome, M. Gomes, R. Seiça, and F. Caramelo, "Using resistin, glucose, age and bmi to predict the presence of breast cancer," *BMC cancer*, vol. 18, no. 1, pp. 1–8, 2018.

[266] L.-A. Teuwen, V. Geldhof, A. Pasut, and P. Carmeliet, "Covid-19: the vasculature unleashed," *Nature Reviews Immunology*, vol. 20, no. 7, pp. 389–391, 2020.

[267] "Covid-19-government data," https://data.gov.il/dataset/covid-19, 2020.

[268] Y. Zoabi, S. Deri-Rozov, and N. Shomron, "Machine learning-based prediction of covid-19 diagnosis based on symptoms," *npj Digital Medicine*, vol. 4, no. 1, pp. 1–5, 2021.