ARCHITECTURES AND TRAINING ALGORITHMS OF DEEP SPIKING NEURAL NETWORKS

A Dissertation

by

YINGYEZHE JIN

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 2018

Major Subject: Computer Engineering

ABSTRACT

The spiking neural network (SNN), an emerging brain-inspired computing paradigm, is positioned to enable spatio-temporal information processing and ultra-low power event-driven neuromorphic hardware. The existing SNN architectures and their corresponding training algorithms suffer from the major limitations in terms of learning performance and efficiency. By leveraging the architectural characteristics and tackling training difficulties of SNNs, this dissertation explores various SNN architectures (feedforward and recurrent) and the corresponding training rules (numerical and bio-inspired algorithms), proposing a comprehensive set of solutions to embrace high performance and energy efficient (deep) spiking neuron networks.

The feedforward network topology is a straightforward structure for exploiting the computation power of spiking neuron networks. However, training feedforward SNNs to achieve a performance level on par with deep models is very challenging. The existing SNN error backpropagation methods are limited in terms of scalability, lack of proper handling of spiking discontinuities, and/or mismatch between the rate-coded loss function and computed gradient. We present a hybrid macro/micro level backpropagation (HM2-BP) algorithm for training multi-layer SNNs, achieving an accuracy level of 99.49% and 98.88% for MNIST and neuromorphic MNIST, respectively, outperforming the existing SNN BP algorithms.

It is widely anticipated that the wiring structure of biological brain will be more closely matched using the recurrent spiking reservoir computing model, or liquid state machine (LSM), to constitute a powerful bio-inspired computing paradigm. The LSM exploits the computation power of recurrent spiking neural networks by incorporating a randomly generated reservoir and a trainable readout layer. To realize adaptive LSMs thus boost learning performance, we propose a novel biologically plausible Activity-based Probabilistic Spiking-Timing Dependent Plastic (AP-STDP) mechanism for recurrent reservoir tuning. Then, a hardware-optimized STDP mechanism is proposed to enable efficient on-chip learning. We demonstrate that the proposed approaches boost the learning performance by up to 2.7% while reducing energy dissipation by up to 25%.

Furthermore, we present a unifying biologically inspired calcium-modulated supervised STDP approach for training and sparsification of readout synapses. We demonstrate that it outperforms a competitive spike-dependent training algorithm by up to 2.7% and prunes out up to 30% of readout synapses without significant performance degradation.

DEDICATION

To my parents, my grandparents and my fiancée for their great support.

ACKNOWLEDGMENTS

Over the past five years, my advisor, Prof. Peng Li, has been constantly supporting me throughout my Ph.D. research in Texas A&M University. Not only does he patiently tutor me to be a true researcher who can define and solve difficult scientific problems, but also enlighten me with helpful advice when I am troubled. I want to express my greatest gratitude to his tremendous help, without which this dissertation would not have been possible. His in-depth perspective and comprehensive expertise are crucial for guiding me, and his everlasting enthusiasm, perseverance and diligence always motivate me to set higher standards and keep challenging myself for pursuing greater goals. I truly appreciate for being his student in such a great research group.

I would love to thank Dr. Nicholas Duffield, Dr. Sebastian Hoyos and Dr. Yoonsuck Choe for their willingness to serve on my committee and offering valuable suggestions and comments. Their insightful feedback on my preliminary exam greatly benefits my major work for defense.

I would also like to thank my colleagues and friends in Computer Engineering & System Group. My specific thanks go to Dr. Yong Zhang, a graduated student in our research group, who introduced me to the field of spiking neuron networks that later becomes my main research focus. I am sincerely thankful for Yu Liu and Wenrui Zhang who have been truly reliable colleagues and friends when collaborating on various research projects. I would love to greatly thank Dr. Qian Wang, Dr. Honghuang Lin, Dr. Ya Wang and Xin Zhan, who have spared no effort to provide inspiring discussions and thoughts. I would like to thank High Performance Research Computing (HPRC) at Texas A&M University for providing computing support.

Finally, I would like to thank my parents, who have always loved me deeply. Their love, encouragement and trust have substantially comforted me during my most difficult time in my Ph.D. life. I would also like to thank my fiancée, Yijie, for her continuous support and love in me. Thanks her for always being there to bear with me and share tears and laughter together. She has became the purpose for me to fight for.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Peng Li, Professor Nicholas Duffield and Professor Sebastian Hoyos of the Department of Electrical and Computer Engineering and Professor Yoonsuck Choe of the Department of Computer Science and Engineering.

All work conducted for the dissertation was completed by the student independently.

Funding Sources

This dissertation is based upon work supported by the National Science Foundation (NSF) under Grant No.CCF-1639995 and the Semiconductor Research Corporation (SRC) under Task 2692.001.

Any opinions, findings, conclusions or recommendations expressed in this dissertation are those of the author and do not necessarily reflect the views of NSF and SRC and their contractors.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

1. INTRODUCTION AND LITERATURE REVIEW[*]

The human brain is remarkable in perceiving, learning, and adapting to the changing environment in some almost seemingly effortless manner. The way which brain processes information set up an extraordinary reference for building new computing systems that may address the performance and energy crisis faced by the computing industry today. Spiking neural networks (SNNs) [2, 3], imitating how realistic biological neuron represents, processes and learns from spatio-temporal information (see Fig. 1.1), providing a promising paradigm that closely resembles the brain behavior. And recently, silicon-based spiking neural emulators and systems have emerged to leverage the event-driven characteristics of SNNs for ultra-low power hardware, see for example [4, 5, 6, 7, 8, 9].

Despite the recent prevalence, the limitations in the existing architectures and their corresponding training algorithms greatly challenge the realization of high-performance and energy efficient SNNs. From the network architectural perspective, it is natural to explore both feedforward and recurrent network topology, as shown in Fig. 1.2. Given a specific network architecture, numerical-based and bio-inspired algorithms can be used to facilitate the learning with the trade-offs between training cost, training difficulty, and performance. However, the problem of training SNNs, which is not well understood at this point, is a long-standing challenge. Moreover, for different various architectures, the training difficulty and concerns might be different. For example, feedforward SNNs might be easier to deal with comparing with the recurrent SNNs because the former one does not have long-term dependency introduced by recurrent connections. But it is very interest-

[*]©2016 Elsevier B.V. Reprinted, with permission, from Yingyezhe Jin and Peng Li, "Performance and robustness of bio-Inspired digital Liquid State machines: A Case study of speech recognition", *Neurocomputing*, Volume: 226, Feb. 2017, Pages 145 - 160. ©2016 IEEE. Reprinted, with permission, from Yingyezhe Jin and Peng Li, "AP-STDP: A novel self-organizing mechanism for efficient reservoir computing", *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2016. ©2016 IEEE. Reprinted, with permission, from Yingyezhe Jin, Yu Liu and Peng Li, "SSO-LSM: A Sparse and Self-Organizing architecture for Liquid State Machine based neural processors", *Proceedings of the 2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. IEEE, July 2016. ©2017 IEEE. Reprinted, with permission, from Yingyezhe Jin and Peng Li, "Calcium-modulated supervised spike-timing-dependent plasticity for readout training and sparsification of the liquid state machine", *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, May 2017.

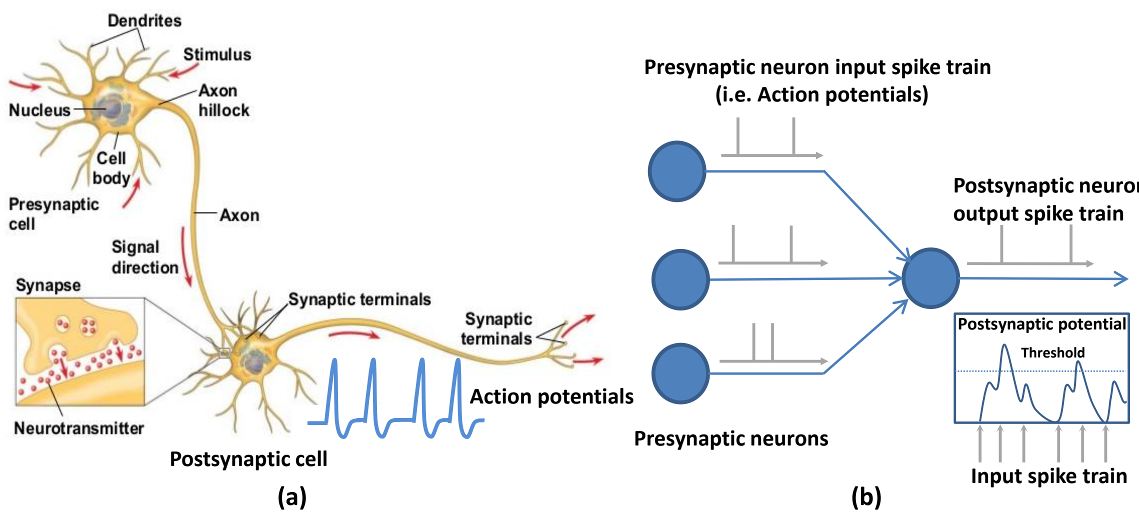

Figure 1.1: (a) The anatomy of biological neurons (adopted from [1] and slightly modified). The communication and processing are done through action potentials. The nucleus accumulates the incoming stimulus in the form of action potentials via the synapses and emits an action potential to its postsynaptic cell. (b) The spiking neurons. Similar to biological neurons, the post-synaptic spike neuron accumulates the action potentials from its presynaptic neuron, generating a spike when its membrane potential reaches the threshold.

ing to exploit the recurrent SNNs for more computing power since micro-circuitry of the brains have both feedforward and recurrent connectivity.

In the first part of this dissertation, we investigate the (deep) feedforward SNNs, which have a straight forward topology. Similar to trends in traditional deep models, we want to explore deep spiking neuron networks which possess huge learning capacity to be leveraged and exploited. However, the most existing spike-based learning algorithms fail to train the deep SNNs for obtaining the competitive performance [10, 11, 12]. Although the error backpropagation [13] is very successful in training deep artificial neural networks (ANNs), attaining the same success of backpropagation (BP) for deep SNNs is challenged by two fundamental issues: complex dynamics and non-differentiable spike events. The two major training difficulties are required to take care for enabling high-performing deep SNNs.

In the remaining parts of this dissertation, we explore the liquid state machine (LSM), a biologically plausible computational paradigm exploiting complex recurrent spiking neural networks [14]

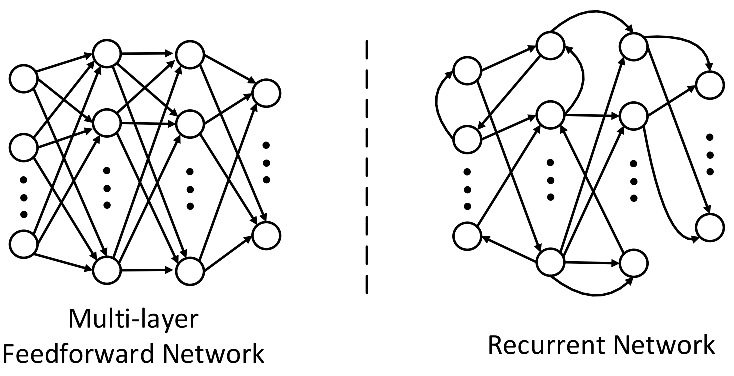

Figure 1.2: The feedforward network v.s. recurrent network.

to envision a good trade-off between the ability in tapping the computational power of recurrent spiking neuron networks and readiness for training. As shown in Fig. 1.3, the LSM consists of a fixed "reservoir", a randomly connected recurrent spiking neural network resembling biological cortical micro-circuitry in the brain, and a group of readout neurons that conduct classification decisions by processing the firing events of the reservoir. Via its nonlinear dynamics, the reservoir projects the input spike trains into a high-dimensional space of the network transient state, and memorizes the inputs received in the past. In the standard LSM model, only the feed-forward synapses projecting from the reservoir to the readout layer are plastic, which shall be trained properly to relax the overall training difficulty of the entire SNN [10, 15, 16]. The liquid state machine also provides an appealing paradigm for realizing energy-efficient VLSI learning processors. VLSI LSM processors and accelerators have been demonstrated recently [17, 18, 19].

Although the typical LSM model is attractive as it exploits the computational power of the recurrent reservoir without tuning it, many studies have argued that randomly generated fixed reservoirs do not act as an effective filter for specific applications [20, 21]. Furthermore, supervision-based adaptation of the recurrent connections in the reservoir is in general very challenging because of complex long-term dependencies in the dynamics [22]. The self-organizing behaviors can be introduced through the unsupervised training scheme of spike-timing dependent plasticity (STDP), which was both experimentally discovered in biology [23] and theoretically studied in

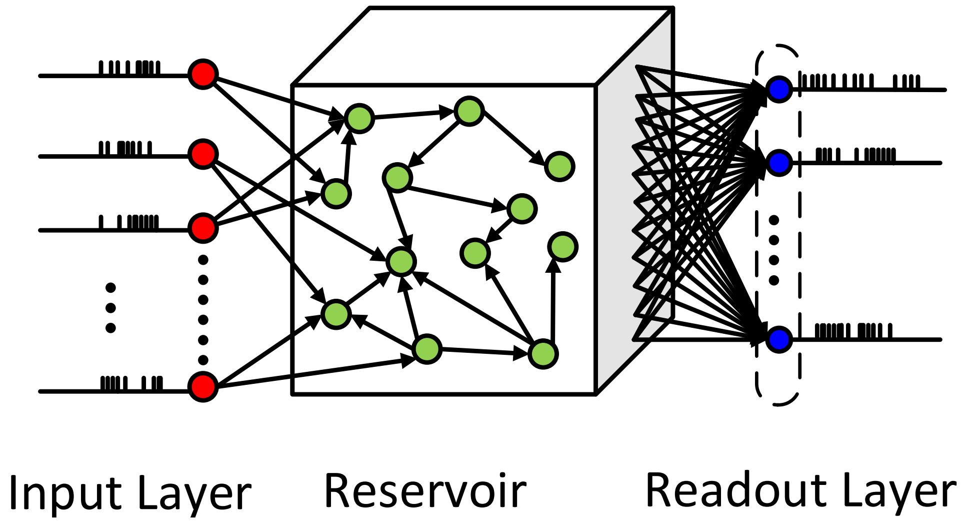

Figure 1.3: The model of the liquid state machine.

neuroscience [24]. Operating directly on spikes and facilitating efficient online learning, STDP can be utilized for tuning reservoir to further boost the performance. But how the STDP rule is designed to efficiently tune the reservoir under limited resolutions for realistic LSM learning architectures remains to be an interesting research problem.

Another key limitation of the existing LSM works is that the dense connections with high resolution between the reservoir and readout are typically required to ensure the good recognition performance. However, the dense connectivity imposes huge network complexity and consequently results in excessive overhead from a hardware point of view. This issue needs to be carefully addressed in order to improve the efficiency of the entire LSM-based architecture while maintain the good classification performance.

In this dissertation, we present a comprehensive set of techniques to enable the energy efficient and high performance spiking neuron networks from both architectural and learning perspectives. First of all, to train deep feedforward SNNs, we propose a hybrid macro/micro level backpropagation scheme, outperforming the existing best reported training algorithms for spiking neurons. Then, we investigate the liquid state machines with recurrent reservoirs. To tune the recurrent connections so as to boost recognition performance of the stand LSM model, we propose a novel probabilistic spike-timing dependent plasticity (STDP) based rule with stopping learning mechanism. Moreover, a hardware-optimized STDP is proposed for efficient on-chip reservoir computing with self-organization. To consistently improve the learning efficiency of reservoir and readout of the LSMs, we present a unifying bio-inspired supervised STDP approach that achieves both sparsity and good performance. Finally, a performance and robustness study of the spiking liquid state machine is conducted to explore the computation capability of recurrent reservoirs and show its performance and low overhead benefit.

1.1 Hybrid Macro/Micro Backpropagation for Deep Spiking Neural Networks

With the great success of deep learning, it is natural to think if we can benefit from cascading multiple feedforward layers of spike neurons as shown in Fig. 1.4. However, the most exist training algorithms for SNNs are insufficient to train the deep SNNs. Therefore, SNNs are yet to achieve the same performance level as conventional deep neuron networks (DNNs). The error backpropagation, which has been successful in training DNNs, cannot be directly applied to SNNs due to the two fundamental challenges: the complex dynamics and non-differentiable discrete spikes.

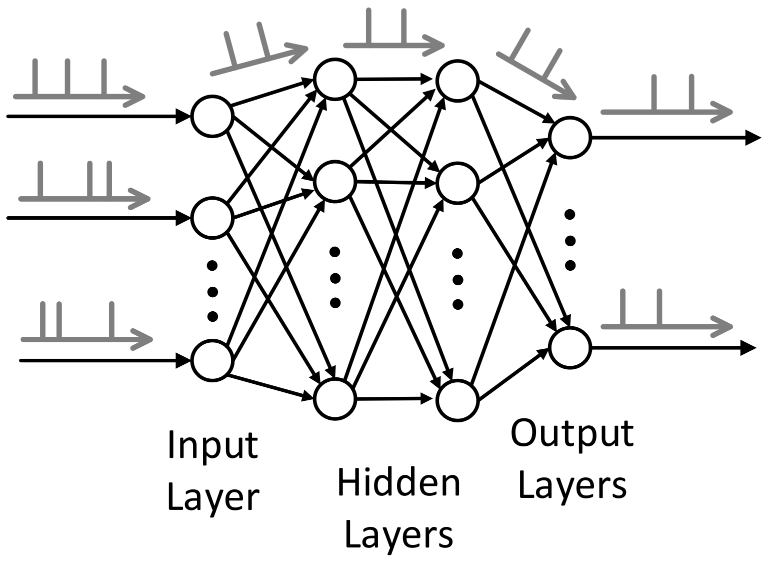

Figure 1.4: A deep spiking neuron network with multiple cascaded hidden layers. The complex temporal dynamics and discrete spike events are two main challenges for training deep SNNs.

The training problem is formulated as follow. As a common practice in SNNs, the rate coding is often adopted to define a loss for each training example at the output layer [25, 26]

$$E = \frac{1}{2}||\mathbf{o} - \mathbf{y}||_2^2, \tag{1.1}$$

where $\mathbf{o}$ and $\mathbf{y}$ are vectors specifying the actual and desired (label) firing counts of the output neurons. Firing counts are determined by the underlying firing events, which are adjusted discretely by tunable weights, resulting in great challenges in computing the gradient of the loss with respect to the weights.

There exist approaches that stay away from the SNN training challenges by first training an ANN and then approximately converting it to an SNN [27, 28, 29, 30]. [31] takes a similar approach which treats spiking neurons almost like non-spiking ReLU units. The accuracy of those methods may be severely compromised because of imprecise representation of timing statistics of spike trains. Although the latest ANN-to-SNN conversion approach [32] shows promise, the problem of direct training of SNNs remains unsolved.

The SpikeProp algorithm [33] is the first attempt to train an SNN by operating on discontinuous spike activities. It specifically targets temporal learning for which derivatives of the loss w.r.t. weights are explicitly derived. However, SpikeProp is very much limited to single-spike learning with very small learning rate, and its successful applications to realistic benchmarks have not been demonstrated. More recently, the backpropagation approaches of [25] and [26] have shown competitive performances. Nevertheless, [25] lacks explicit consideration of temporal correlations of neural activities. Furthermore, it does not handle discontinuities occurring at spiking moments by treating them as noise while only computing the error gradient for the remaining smoothed membrane voltage waveforms instead of the rate-coded loss. [26] addresses the first limitation of [25] by performing BPTT [34] to capture temporal effects. However, similar to [25], the error gradient is computed for the continuous membrane voltage waveforms resulted from smoothing out all spikes, leading to inconsistency w.r.t the rate-coded loss function. In summary, the existing SNNs

BP algorithms have three major limitations: i) suffering from limited learning scalability [33], ii) either staying away from spiking discontinuities (e.g. by treating spiking moments as noise [25]) or deriving the error gradient based on the smoothed membrane waveforms [25, 26], and therefore iii) creating a mismatch between the computed gradient and targeted rate-coded loss [25, 26].

In the second chapter, we present a hybrid macro/micro level backpropagation (HM2-BP) algorithm which perform error backpropagation across 1) firing rates (macro-level) and 2) spike trains (micro-level). The temporal correlation of exact timings is captured by *spike-train level post-synaptic potential* (S-PSP) defined at the micro-level. At the macro-level, the errors are defined and back-propagated by aggregating the effects of discrete spike trains on each neuron's firing count via S-PSPs. A decoupled model of the S-PSP is proposed to detach the effects of firing rates and spike timings to allow differentiation of the S-PSP at the micro-level. We evaluate the HM2-BP by training deep SNNs on the static MNIST [35] and dynamic neuromorphic N-MNIST [36], and show that our approach achieves an accuracy level of 99.49% and 98.88% for MNIST and N-MNIST, respectively, surpassing the state-of-the-art results of the existing SNN BP algorithm.

1.2 Adaptive Liquid State Machine Architectures with Efficient Reservoir Tuning

There has been increasing interest in exploring reservoir computing, a biologically plausible computation paradigm, to make use of the computational power of recurrent neural networks without tuning complex recurrent connections [14]. The liquid state machine (LSM) is one specific form of reservoir computing. Structurally, the LSM (shown in Fig. 1.5) is composed of a fixed "reservoir", a randomly connected recurrent spiking neural network, used as a generic filter through which the LSM maps the input into a high-dimensional space of network dynamics. Typically, for the sake of relaxing overall training difficulty, only the readout is trained to make the final classification decisions by processing the information coming from the reservoir. As such, the standard LSM serves as a good trade-off between the tractable trainability and exploitation of computation power.

Adapting the recurrent connections in the reservoir is in general very challenging because of complex long-term dependencies in the dynamics [22]. In this regard, the typical LSM model is

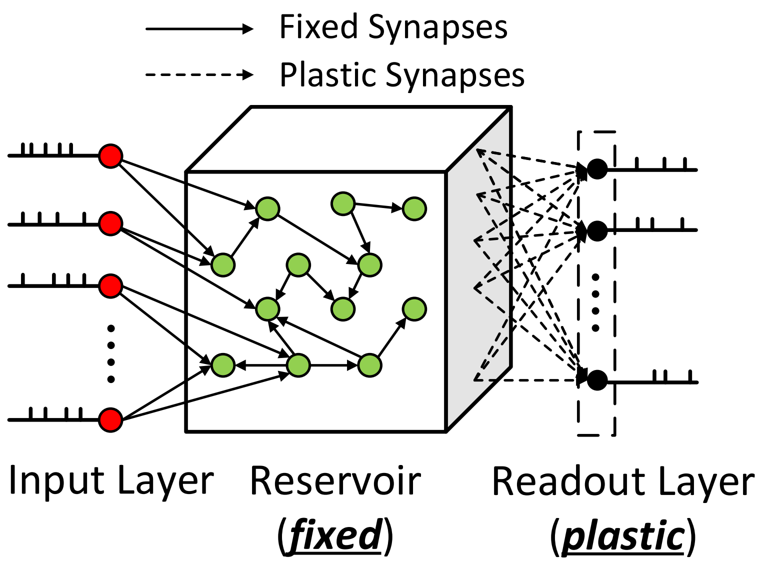

Figure 1.5: The reservoir of the standard LSM is fixed, only the synapses from the reservoir to the readout are plastic and trained for conducting classification.

attractive as it exploits the computational power of the spiking recurrent reservoir without tuning it. Although integrating a generic reservoir presents a simple solution to build general-purpose LSM processors as presented in [17], many studies have argued that randomly generated fixed reservoirs do not act as an effective filter for specific applications [20, 21].

While reservoir tuning has not been well studied in the literature, several attempts have shown that the separation capability of the reservoir and hence learning performance may be boosted by tuning the recurrent connections [21, 37, 38, 39, 40, 41, 42]. But the question of how to efficiently tune the recurrent reservoir for improving the performance of real-world applications remains unclear. In [21], an iterative refinement approach is proposed to modify the recurrent synaptic connections to boost the separation ability but no performance improvement for real-world tasks is reported. A gene regulatory network (GRN) regulated reservoir is proposed in [42] and shown to have self-organizing behaviors leading to improved performance. However, for both approaches, the global neuronal activities need to be known for altering synaptic weights, which is inefficient and costly to implement.

One can also introduce self-organizing behaviors managed through spike-timing dependent plasticity (STDP), which was both experimentally discovered in biology [23] and theoretically studied in computational neuroscience [24]. STDP explores the correlation between the firing activities of a pair of presynaptic and postsynaptic neurons and tunes the synaptic weight locally in an unsupervised manner. Therefore, compared to [21, 42], STDP is more suitable for efficient online learning because of its simplicity and locality, as explored in [37, 38, 39, 40, 41, 43, 44] for tuning reservoirs.

Here we explore efficient reservoir tuning in the context of hardware implementation where synaptic weights are realized digitally with a finite resolution. From a biological point of view, targeting discretized synaptic weights is reasonable because there is evidence that modification of individual biological synaptic strength is done in an all-or-none (i.e. digital) instead of graded (i.e. analog) manner [45, 46]. Furthermore, since we deal with realistic synapses, it is not possible to reduce synaptic modifications to an arbitrarily small value [47].

Unfortunately, standard STDP rules conduct continuous weight updates and may trigger a large number of small-valued weight updates throughout the reservoir, degrading the efficiency of hardware-based realizations on FPGA or in ASIC. Furthermore, there exist important tradeoffs between the range of weight tunability, the resolution of weights, performance and hardware cost. High resolution and wide tunability range can lead to good learning performance at the cost of high hardware overhead. Therefore, STDP rules must be carefully designed for cost-effective realization of the desired self-organizing properties. As such, low-resolution synapses represented by a small number of bits are strongly preferred.

Equally importantly, there is no prior STDP work that addresses the issue of synaptic weight saturation for reservoir tuning. Without a specific stop-learning mechanism in place, continuous on-going weight modifications may quickly saturate a synaptic weight, making it unresponsive to future inputs. This situation of synaptic memory saturation is very likely to happen in hardware because synapses have a limited number of states due to low resolution and narrow tuning range. Therefore, from a memory retention point of view, suitable stop-learning conditions are desired in order to prevent saturation such that synapses can learn from the new experience without being over-interfered by the past experience [47].

Meanwhile, several works investigate VLSI implementation of the conventional LSM model with a fixed reservoir, which include FPGA based speech recognition [48], a VLSI architecture incorporating a perceptron readout layer and the p-Delta learning algorithm [18], and a general-purpose LSM architecture for processing multiple applications [17, 49]. On the other hand, it has been argued that randomly generated fixed reservoirs may not act as an optimized filter for specific applications [20, 21]. To our best knowledge, there is no prior work that explores reservoir tuning in hardware based LSM architectures.

The goal of the third chapter is to enable efficient reservoir tuning for recurrent SNNs. First of all, we propose a novel Activity-based probabilistic STDP (AP-STDP) with a stop-learning scheme, achieving good learning performance boost under the limited synaptic resolution. Then targeting at the realistic reservoir tuning based LSM in hardware, we present a hardware-optimized STDP for self-adaptive LSM architectures with the reservoir sparsification to attain an energy efficient neural processor. A variance-based simple readout sparsification approach is proposed to further reduce the energy dissipation of the targeted LSM architecture.

1.3 Readout Training and Sparsification of Liquid State Machines

Training general spiking neural networks is a long-standing challenge. [33] proposes the Spike-Prop algorithm that trains feed-forward SNNs by propagating error back in time. While being of a theoretical interest, numerically computing the derivatives of the error function with respect to synaptic weights in terms of spike times is extremely involved. SpikeProp-like algorithms are far from being mature and have yet to be demonstrated for meaningful real-world applications. As such, the liquid state machine offers a good tradeoff between training cost and computational power. The standard LSM model has a fixed recurrent reservoir and deals with a much simpler training problem: only the feed-forward synapses projecting from the reservoir to the readout layer (referred to as *readout synapses*) need to be trained, which is referred to as *readout training*.

Instead of using numerical techniques that involve matrix factorization or backpropagation, the recent work of [10] introduces a biologically-inspired spike-dependent readout training approach with the advantages being local and amenable to VLSI implementation. However, as illustrated in Fig. 1.6, the key limitation of this approach is that good performance is typically guaranteed only with high-resolution and full connectivity between the reservoir and readout, which contributes significantly to the overall network complexity and is also costly from a hardware point of view.

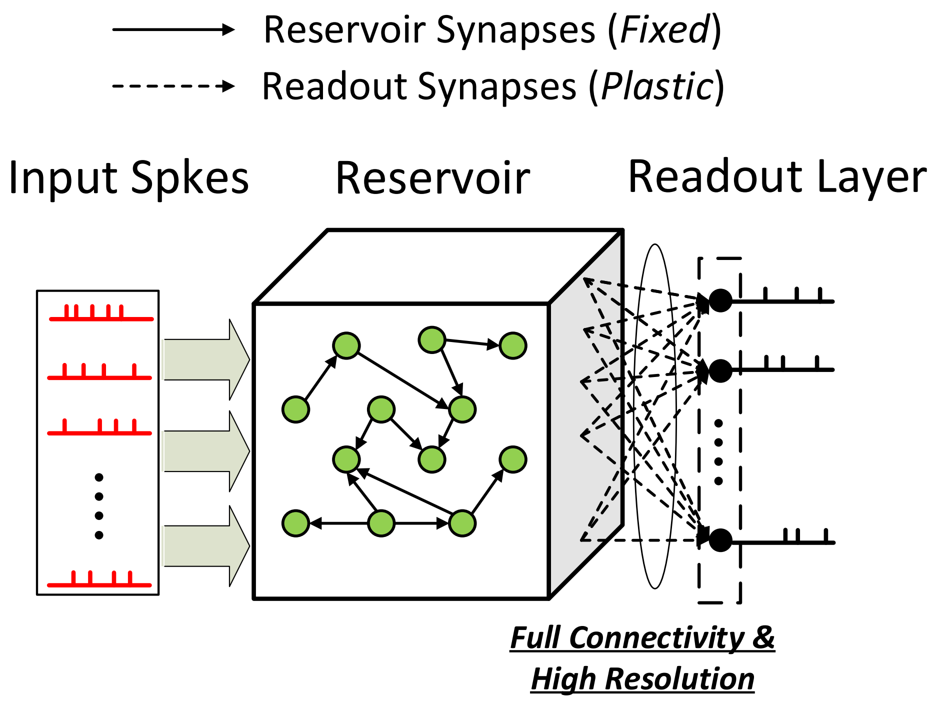

Figure 1.6: The dense connectivity and high resolution are required to achieve good performance for readout training.

The above challenges motivate us to seek an alternative for readout training. To this end, spike-timing-dependent plasticity (STDP) [23, 24], a well-known unsupervised learning mechanism, is able to locally tune spiking neural networks according to temporal spike correlations and produce interesting self-organizing behaviors. Towards supervised learning which we target in this work, ideas of combining supervision and STDP have been explored for precisely timed spike reproduction and decision making [11, 50, 51], however, without demonstrating success for real-world tasks.

For the first time, we target the following objectives under the context of the liquid state machine:

- **Objective-1**: develop supervised STDP mechanisms for readout training with improved learning performance for real-life applications;
- **Objective-2**: develop supervised STDP based techniques for sparsifying readout synapses so as to reduce network complexity and enable efficient hardware realization.

It is important to note that one challenge associated with Objective-1 and Objective-2 is that they are *competing* objectives in the sense that sparsity in readout synapses can easily degrade learning performance. In this work, both objectives are achieved under a unifying biologically motivated calcium-modulated supervised STDP approach.

Towards the first objective, we propose a new calcium-modulated learning algorithm based on supervised STDP, dubbed *CaL-S^{2}TDP*, and demonstrate its improved performance for readout training. One important limitation of the earlier work on supervised STDP is that the issue of synaptic weight saturation has not been addressed. Without a carefully-chosen stop-learning mechanism, continuous on-going weight modifications may quickly saturate a synaptic weight, overwhelming the synapse by the past experience and preventing it from responding to new stimuli [47]. This can result in bad utilization of memory that is presented in the network and poor learning performance. The problem of weight saturation exacerbates on digital hardware where each synaptic weight is realized using a limited number of bits and tuning range. Furthermore, standard STDP rules conduct continuous weight updates and may trigger many small-valued weight updates, resulting in bad hardware memory access efficiency.

In *CaL-S^{2}TDP*, supervision is realized by the combined use of a classification teacher (CT) signal and a new *depressive* STDP rule. For a given input class, the CT promotes the firing activity of targeted the readout neurons by injecting a positive current which serves as the supervision. It also addresses the robustness limitation of standard STDP mechanisms by making it possible to initiate the STDP-based temporal learning process regardless of initial weight values. Motivated by a large variety of STDP mechanisms discovered in the brain [52], we engine the depressive STDP to provide supervision to readout neurons that are desired to have low firing activity for the given input class.

To improve learning performance and address the challenge of weight saturation, we employ probabilistic weight updates and a stop-learning mechanism. Stop learning is trigged by monitoring the calcium concentration of the post-synaptic neuron, which is modeled by low-pass filtering the post-synaptic spike train. Our calcium-modulated supervised STDP approach, for the first time, combines STDP-based temporal learning with modulation provided by the averaged firing level.

Towards the second objective, we propose a new calcium-modulated sparsification algorithm based on supervised STDP, dubbed *CaS-S*^{2}*TDP*, for readout synapse sparsification and demonstrate that it can produce a high-degree of sparsity without significant degradation of learning performance. In the liquid state machine, readout synapses play a significant role in classification decision making. A high-degree of connectivity, often full connectivity, between the reservoir and readout layer must be realized with high-resolution synapses for good learning performance. Consequently, the readout synapses contribute greatly to the overall network complexity, and also to the silicon overhead and energy dissipation of hardware-based LSM processors. To date, the question of how to simultaneously achieve good learning and sparsity in readout synapses remains to be answered.

We achieve the two competing objectives by employing a two-step methodology: sparsificaiton by *CaS-S*^{2}*TDP* followed by readout training by *CaL-S*^{2}*TDP*. Essentially, *CaS-S*^{2}*TDP* exploits the automatic competition among afferent synapses of each readout neuron mediated by a typical unsupervised STDP mechanism [53] to produce a bimodal weight distribution with desired sparsity. Under the framework of calcium-modulated supervised STDP, *CaS-S*^{2}*TDP* adds a teacher signal, called *sparsity teacher*, and a relaxed stop-learning rule, to robustly sparsify the readout synapses while responding to the spatio-temporal structures of the presented training inputs. The sparsity discovered by *CaS-S*^{2}*TDP* is carried over to the second full-blown training phase based on *CaL-S*^{2}*TDP*.

The unified calcium-modulated supervised STDP approach for readout learning and sparsification is described in the fourth chapter.

1.4 Performance and Robustness Study of Liquid State Machines

Compared to traditional perceptrons, SNNs possess increased computational power [3] and are biologically more plausible because they model the communication of the temporal information among biological neurons. Recent years have witnessed an increased interest in the theoretical studies of SNNs including bio-inspired learning algorithms [54, 55, 56, 57, 58, 11] and network structure [59, 60]. Works targeting practical implementation of SNNs in hardware systems have also emerged [48, 61, 62, 63]. Furthermore, bio-inspired spiking neural networks are shown to have inherent error resilience and fault tolerance [64], a very appealing characteristic for VLSI implementation in highly scaled modern CMOS technologies, for which device reliability and process variability are grand challenges.

Inspired by micro-circuits in the cortex, the liquid state machine (LSM), has been demonstrated to provide powerful computational capability for many applications [15, 54, 65, 66]. The LSM consists of a reservoir, a recurrent spiking neural network with fixed but randomly chosen connections introduced to preprocess the external input signals, and a group of readout neurons performing classification by further processing and extracting relevant features of the input patterns from the reservoir. With recurrent connections in the reservoir, the LSM can map the input into a high-dimensional space by producing complex nonlinear dynamics in the reservoir, which makes the subsequent classification easier. The decaying dynamic responses of the input signals in the reservoir serve as a transient memory by which critical information about the inputs is captured. As a result, the LSM is especially competitive for dealing with temporal input patterns such as speech signals [15, 67].

However, the most existing works of LSMs either focus only on high-level computational principles [21, 60] without a real-world application background or on an application level without much theoretical analysis [15, 48]. [18] studied the design and VLSI implementation of the readout stage for LSMs based on perceptrons and the p-Delta learning algorithm, which were less biologically inspired and only applied to simple two-class recognition and rate-sum retrieval problems. Most importantly, little has been investigated for VLSI hardware implementation of LSMs while considering the optimization and tradeoffs involving learning performance, hardware overhead, and error resilience when using real-world challenging applications to benchmark.

In the fifth chapter of this dissertation, we present a systematic study of performance and robustness issues of LSMs, targeting specially at speech recognition and digital VLSI implementation. A structured design space exploration is performed to show that a favor trade-off can be obtained between the design complexity and recognition performance. By modelling various hardware manufacturing and noise from the behavioral level, we demonstrate that the LSM is relatively robust to the studied failures and errors.

1.5 Organization of Dissertation

The rest of this dissertation is organized as follows: In Chapter 2, we proposed the hybrid macro/micro level backpropagation (HM2-BP) algorithm for training deep spiking neuron networks. In Chapter 3, we present the Activity-based Probabilistic STDP (AP-STDP) for efficient recurrent reservoir tuning and a hardware-optimized STDP for self-organizing LSM architecture. The unifying calcium-modulated supervised STDP for readout learning and sparsification is demonstrated in Chapter 4. To supplement our architectural exploration, Chapter 5 provides the performance and robustness study of LSMs where a large design space is systematically explored and the robustness issue is studied by modeling process variations and noise from the behavior level. Chapter 6 concludes this dissertation and provides the discussions of potential future work.

2. HYBRID MACRO/MICRO BACKPROPAGATION FOR TRAINING DEEP SPIKING NEURAL NETWORKS

In spite of recent success in deep neural networks (DNNs) [68, 69, 70], it is believed that biological brains operate rather differently. Compared with DNNs that lack processing of spike timing and event-driven operations, biologically realistic spiking neural networks (SNNs) [2, 3] provide a promising paradigm for exploiting spatio-temporal patterns for added computing power, and enable ultra-low power event-driven neuromorphic hardware [7, 9, 28]. There are theoretical evidences supporting that SNNs possess greater computational power over traditional artificial neural networks (ANNs) [3]. SNNs are yet to achieve a performance level on a par with deep ANNs for practical applications. The error backpropagation [13] is very successful in training ANNs. Attaining the same success of backpropagation (BP) for feedforward SNNs is challenged by two fundamental issues: complex temporal dynamics and non-differentiability of discrete spike events.

Here, we formulate the problem as follow. As a common practice in SNNs, the rate coding is often adopted to define a loss for each training example at the output layer [25, 26]

$$E = \frac{1}{2}||\mathbf{o} - \mathbf{y}||_2^2, \tag{2.1}$$

where $\mathbf{o}$ and $\mathbf{y}$ are vectors specifying the actual and desired (label) firing counts of the output neurons. Firing counts are determined by the underlying firing events, which are adjusted discretely by tunable weights, resulting in great challenges in computing the gradient of the loss with respect to the weights.

Several attempts have been made to train the SNN using error backpropagation (BP) that directly operates on discontinuous spike events [33, 25, 26]. But the existing BP algorithms have three major limitations: i) limited learning scalability [33], ii) either circumvent the issue of handling spiking discontinuities [25] or deriving the gradient based on the smoothed membrane voltage waveforms [25, 26] and hence iii) generating a mismatch between the computed gradient and the targeted rate-coded loss function [25, 26].

We derive the gradient of the rate-coded error defined in (1.1) by decomposing each derivative into two components

$$\frac{\partial E}{\partial w_{ij}} = \underbrace{\frac{\partial E}{\partial a_i}}_{\text{bp over firing rates}} \times \underbrace{\frac{\partial a_i}{\partial w_{ij}}}_{\text{bp over spike trains}}, \tag{2.2}$$

where a_i is the (weighted) aggregated membrane potential for the post-synaptic neuron i per

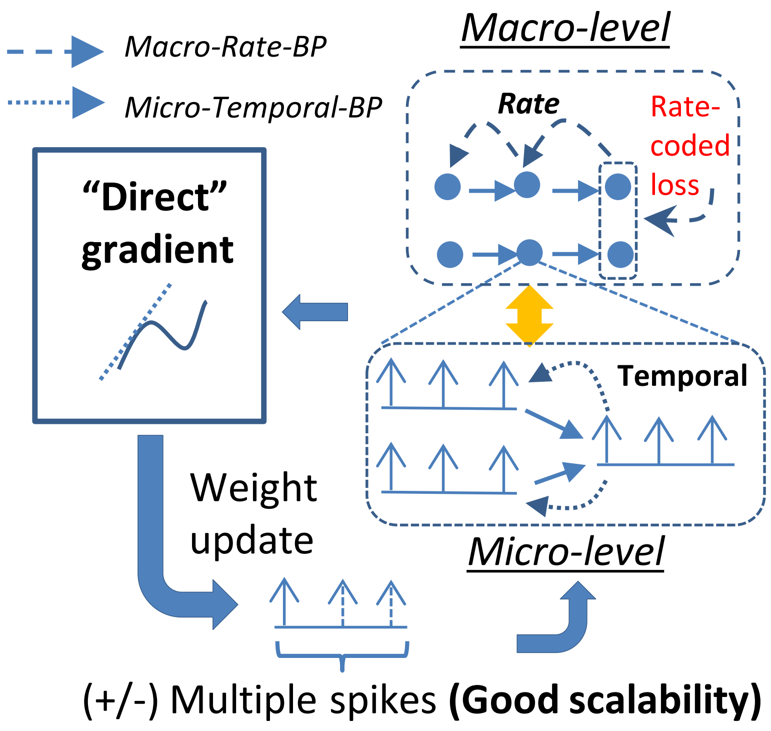

Figure 2.1: Hybrid macro-micro level backpropagation.

(2.11). As such, we propose a novel hybrid macro-micro level backpropagation (HM2-BP) algorithm which performs error backpropagation across two levels: 1) backpropagation over firing rates (***macro-level***), 2) backpropagation over spike trains (***micro-level***), and 3) backpropagation based on interactions between the two levels, as illustrated in Fig. 2.1.

At the microscopic level, for each pre/post-synaptic spike train pair, we precisely compute the *spike-train level post-synaptic potential*, referred to as *S-PSP* throughout this dissertation, to account for the temporal contribution of the given pre-synaptic spike train to the firings of the post-synaptic neuron based on exact spike times. At the macroscopic level, we back-propagate the errors of the defined rate-based loss by aggregating the effects of spike trains on each neuron's firing count via the use of S-PSPs, and leverage this as a practical way of linking spiking events to firing rates. To assist backpropagation, we further propose a decoupled model of the S-PSP for disentangling the effects of firing rates and spike-train timings to allow differentiation of the S-PSP w.r.t. pre and post-synaptic firing rates at the micro-level. As a result, our HM2-BP approach is able to evaluate the direct impact of weight changes on the rate-coded loss function. Moreover, the resulting weight updates in each training iteration can lead to introduction or disappearance of multiple spikes.

We evaluate the proposed BP algorithm by training deep fully connected and convolutional SNNs based on both the static MNIST [35] and dynamic neuromorphic N-MNIST [36] datasets. Our BP algorithm achieves an accuracy level of 99.49% and 98.88% for MNIST and N-MNIST, respectively, outperforming the best reported performances obtained from the existing SNN BP algorithms.

2.1 Hybrid Macro-Micro Backpropagation

The complex dynamics generated by spiking neurons and non-differentiable spike impulses are two fundamental bottlenecks for training SNNs using backpropagation. We address these difficulties at both macro and micro levels.

2.1.1 Micro-level Computation of Spiking Temporal Effects

The leaky integrate-and-fire (LIF) model is one of the most prevalent choices for describing dynamics of spiking neurons, where the neuronal membrane voltage $u_i(t)$ at time t for the neuron i is given by

$$\tau_m \frac{u_i(t)}{dt} = -u_i(t) + R\, I_i(t), \tag{2.3}$$

where $I_i(t)$ is the input current, R the effective leaky resistance, C the effective membrane capacitance, and $\tau_m = RC$ the membrane time constant. A spike is generated when $u_i(t)$ reaches the threshold ν. After that $u_i(t)$ is reset to the resting potential u_r, which equals to 0 in this work. Each post-synaptic neuron i is driven by a post-synaptic current of the following general form

$$I_i(t) = \sum_j w_{ij} \sum_f \alpha(t - t_j^{(f)}), \tag{2.4}$$

where w_{ij} is the weight of the synapse from the pre-synaptic neuron j to the neuron i, $t_j^{(f)}$ denotes a particular firing time of the neuron j. We adopt a first order synaptic model with time constant τ_s

$$\alpha(t) = \frac{q}{\tau_s} \exp\left(-\frac{t}{\tau_s}\right) H(t), \tag{2.5}$$

where $H(t)$ is the Heaviside step function, and q the total charge injected into the post-synaptic neuron i through a synapse of a weight of 1. Let $\hat{t}_i$ denote the last firing time of the neuron i w.r.t time t: $\hat{t}_i = \hat{t}_i(t) = \max\{t_i | t_i^{(f)} < t\}$. Plugging (2.4) into (2.3) and integrating (2.3) with $u(\hat{t}_i) = 0$ as its initial condition, we map the LIF model to the Spike Response Model (SRM) [71]

$$u_i(t) = \sum_j w_{ij} \sum_f \epsilon\left(t - \hat{t}_i, t - t_j^{(f)}\right), \tag{2.6}$$

with

$$\epsilon(s,t) = \frac{1}{C} \int_0^s \exp\left(-\frac{t'}{\tau_m}\right) \alpha\left(t - t'\right) \, \mathrm{d}t'. \tag{2.7}$$

Since q and C can be absorbed into the synaptic weights, we set $q = C = 1$. Integrating (2.7) yields

$$\epsilon(s,t) = \frac{\exp(-\max(t-s,0)/\tau_s)}{1 - \frac{\tau_s}{\tau_m}} \left[\exp\left(-\frac{\min(s,t)}{\tau_m}\right) - \exp\left(-\frac{\min(s,t)}{\tau_s}\right)\right] H(s)H(t). \tag{2.8}$$

ϵ is interpreted as the normalized (by synaptic weight) *post-synaptic potential*, which is evoked by a single firing spike of the pre-synaptic neuron j.

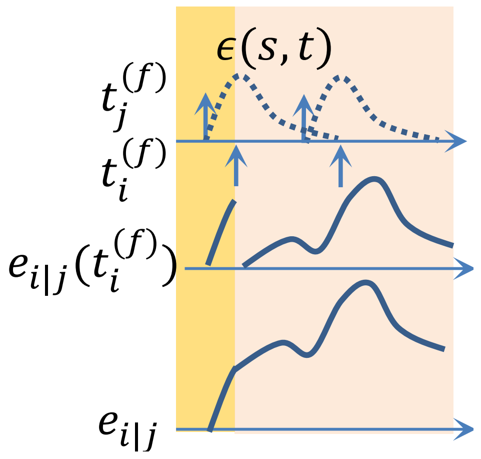

Figure 2.2: The computation of the S-PSP.

For any time t, the exact "contribution" of the neuron j's spike train to the neuron i's post-synaptic potential is given by summing (2.8) over all pre-synaptic spike times $t_j^{(f)}$, $t_j^{(f)} < t$. We particularly concern the contribution right before each post-synaptic firing time $t_i^{(f)}$ when $u_i(t_i^{(f)}) = \nu$, which we denote by $e_{i|j}(t_i^{(f)})$. Summing $e_{i|j}(t_i^{(f)})$ over all post-synaptic firing times as well as the end of spike sequence time gives the *total* contribution of the neuron j's spike-train to the firing activities of the neuron i as shown in Fig. 2.2

$$e_{i|j} = \sum_{t_i^{(f)}} \sum_{t_j^{(f)}} \epsilon(t_i^{(f)} - \hat{t}_i^{(f)}, t_i^{(f)} - t_j^{(f)}), \tag{2.9}$$

where $\hat{t}_i^{(f)} = \hat{t}_i^{(f)}(t_i^{(f)})$ denotes the last post-synaptic firing time before $t_i^{(f)}$.

Importantly, we refer to $e_{i|j}$ as the (normalized) *spike-train level post-synaptic potential* (S-PSP). As its name suggests, S-PSP characterizes the aggregated influence of the pre-synaptic neuron on the post-synaptic neuron's firings at the level of spike trains, providing a basis for relating firing counts to spike events and enabling scalable SNN training that adjusts spike trains rather than single spikes. Clearly, each S-PSP $e_{i|j}$ depends on both rate and temporal information of the pre/post spike trains. To assist the derivation of our BP algorithm, we make the dependency of $e_{i|j}$ on the pre/post-synaptic firing counts o_i and o_j explicit although o_i and o_j are already embedded in the spike trains

$$e_{i|j} = f(o_j, o_i, \mathbf{t}_j^{(f)}, \mathbf{t}_i^{(f)}), \tag{2.10}$$

where $\mathbf{t}_j^{(f)}$ and $\mathbf{t}_i^{(f)}$ represent the pre and post-synaptic timings, respectively. Summing the weighted S-PSPs from all pre-synaptic neurons results in the *total post-synaptic potential* (T-PSP) a_i, which is directly correlated to the neuron i's firing count

$$a_i = \sum_j w_{ij}\, e_{i|j}. \tag{2.11}$$

2.1.2 Error Backpropagation at Macro and Micro Levels

It is evident that the total post-synaptic potential a_i must be no less than the threshold ν in order to make the neuron i fire at least once, and the total firing count is $\left\lfloor \frac{a_i}{\nu} \right\rfloor$. We relate the firing count o_i of the neuron i to a_i approximately by

$$o_i = g(a_i) = \left\lfloor \frac{a_i}{\nu} \right\rfloor = \left\lfloor \frac{\sum_j w_{ij}\, e_{i|j}}{\nu} \right\rfloor \approx \frac{\sum_j w_{ij}\, e_{i|j}}{\nu}, \tag{2.12}$$

where the rounding error would be insignificant when ν is small. Despite that (2.12) is linear in S-PSPs, it is the interaction between the S-PSPs through nonlinearities hidden in the micro-level LIF model that leads to a given firing count o_i. Missing from the existing works [25, 26], (2.12) serves as an important bridge connecting the aggregated micro-level temporal effects with the macro-level count of discrete firing events. In a vague sense, a_i and o_i are analogies to pre-activation and activation in the traditional ANNs, respectively, although they are not directly comparable. (2.12) allows for rate-coded error backpropagation on top of discrete spikes across the macro and micro levels.

Using (2.12), the macro-level rate-coded loss of (1.1) is rewritten as

$$E = \frac{1}{2}||\mathbf{o} - \mathbf{y}||_2^2 = \frac{1}{2}||g(\mathbf{a}) - \mathbf{y}||_2^2, \tag{2.13}$$

where $\mathbf{y}$, $\mathbf{o}$ and $\mathbf{a}$ are vectors specifying the desired firing counts (label vector), the actual firing counts, and the weighted sums of S-PSP of the output neurons, respectively. We now derive the gradient of E w.r.t w_{ij} at each layer of an SNN.

For the i_{th} neuron in the output layer m, we have

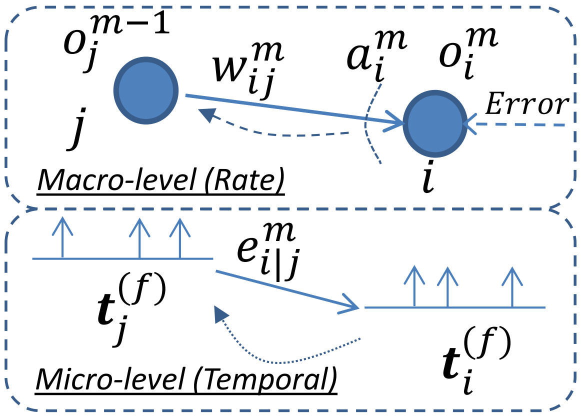

Figure 2.3: Macro/micro backpropagation in the output layer.

$$\frac{\partial E}{\partial w_{ij}} = \underbrace{\frac{\partial E}{\partial a_i^m}}_{\text{macro-level bp}} \times \underbrace{\frac{\partial a_i^m}{\partial w_{ij}}}_{\text{micro-level bp}}, \tag{2.14}$$

where variables associated with neurons in the layer m have m as the superscript. As shown in Fig. 2.3, the first term of (2.14) represents the macro-level backpropagation of the rate-coded error with the second term being the micro-level error backpropagation. From (2.13), the macro-level error backpropagation is given by

$$\delta_i^m = \frac{\partial E}{\partial a_i^m} = (o_i^m - y_i^m)\, g'(a_i^m) = \frac{o_i^m - y_i^m}{\nu}. \tag{2.15}$$

Similar to the conventional backpropagation, we use δ_i^m to denote the back propagated error. According to (2.11) and (2.10), a_i^m can be unwrapped as

$$a_i^m = \sum_{j=1}^{r^{m-1}} w_{ij}\ e_{i|j}^m = \sum_{j=1}^{r^{m-1}} w_{ij}\ f(o_j^{m-1}, o_i^m, \mathbf{t}_j^{(f)}, \mathbf{t}_i^{(f)}), \tag{2.16}$$

where r^{m-1} is the number of neurons in the $(m-1)_{th}$ layer. Differentiating (2.16) and making use of (2.12) leads to the micro-level error propagation based on the total post-synaptic potential (T-PSP) a_i^m

$$\frac{\partial a_i^m}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left(\sum_{j=1}^{r^{m-1}} w_{ij}\ e_{i|j}^m \right) = e_{i|j}^m + \sum_{l=1}^{r^{m-1}} w_{il} \frac{\partial e_{i|l}^m}{\partial o_i^m}\ \frac{\partial o_i^m}{\partial w_{ij}} = e_{i|j}^m + \frac{e_{i|j}^m}{\nu} \sum_{l=1}^{r^{m-1}} w_{il} \frac{\partial e_{i|l}^m}{\partial o_i^m}. \tag{2.17}$$

Although the network is feed-forward, there are non-linear interactions between S-PSPs. The second term of (2.17) captures the hidden dependency of the S-PSPs on the post-synaptic firing count o_i^m.

For the i_{th} neuron in the hidden layer k, we have

$$\frac{\partial E}{\partial w_{ij}} = \underbrace{\frac{\partial E}{\partial a_i^k}}_{\text{macro-level bp}} \times \underbrace{\frac{\partial a_i^k}{\partial w_{ij}}}_{\text{micro-level bp}} = \delta_i^k\ \frac{\partial a_i^k}{\partial w_{ij}}. \tag{2.18}$$

The macro-level error backpropagation at a hidden layer is much more involved as in Fig. 2.4

$$\delta_i^k = \frac{\partial E}{\partial a_i^k} = \sum_{l=1}^{r^{k+1}} \frac{\partial E}{\partial a_l^{k+1}}\ \frac{\partial a_l^{k+1}}{\partial a_i^k} = \sum_{l=1}^{r^{k+1}} \delta_l^{k+1}\ \frac{\partial a_l^{k+1}}{\partial a_i^k}. \tag{2.19}$$

According to (2.11) , (2.10) and (2.12), we unwrap a_l^{k+1} and get

$$a_l^{k+1} = \sum_{p=1}^{r^k} w_{lp}\ e_{l|p}^{k+1} = \sum_{p=1}^{r^k} w_{lp}\ f(o_p^k, o_l^{k+1}, \mathbf{t}_p^{(f)}, \mathbf{t}_l^{(f)}) = \sum_{p=1}^{r^k} w_{lp}\ f(g(a_p^k), o_l^{k+1}, \mathbf{t}_p^{(f)}, \mathbf{t}_l^{(f)}). \tag{2.20}$$

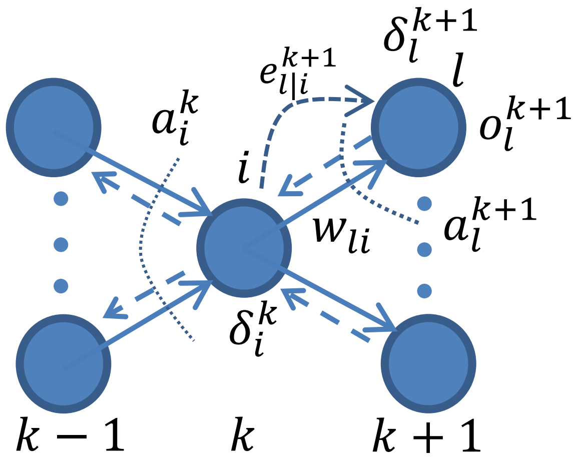

Figure 2.4: Macro-level backpropagation at a hidden layer.

Therefore, $\frac{\partial a_l^{k+1}}{\partial a_i^k}$ becomes

$$\frac{\partial a_l^{k+1}}{\partial a_i^k} = w_{li} \frac{\partial e_{l|i}^{k+1}}{\partial o_i^k} \frac{\partial o_i^k}{\partial a_i^k} = w_{li} \frac{\partial e_{l|i}^{k+1}}{\partial o_i^k} g'(a_i^k) = \frac{w_{li}}{\nu} \frac{\partial e_{l|i}^{k+1}}{\partial o_i^k}, \tag{2.21}$$

where the dependency of $e_{l|i}^{k+1}$ on the pre-synaptic firing count o_i^k is considered but the one on the firing timings are ignored, which is supported by the decoupled S-PSP model in (2.25). Plugging (2.21) into (2.19), we have

$$\delta_i^k = \frac{1}{\nu} \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{li} \frac{\partial e_{l|i}^{k+1}}{\partial o_i^k}. \tag{2.22}$$

The micro-stage backpropagation at hidden layers is identical to that at the output layer, i.e. (2.17). Finally, we obtain the derivative of E with respect to w_{ij} as follows

$$\frac{\partial E}{\partial w_{ij}} = \delta_i^k e_{i|j}^k \left(1 + \frac{1}{\nu} \sum_{l=1}^{r^{k-1}} w_{il} \frac{\partial e_{i|l}^k}{\partial o_i^k}\right), \tag{2.23}$$

where

$$\delta_i^k = \begin{cases} \frac{o_i^m - y_i^m}{\nu} & \text{for output layer,} \\ \frac{1}{\nu} \sum_{l=1}^{r^{k+1}} \delta_l^{k+1} w_{li} \frac{\partial e_{l|i}^{k+1}}{\partial o_i^k} & \text{for hidden layers.} \end{cases} \tag{2.24}$$

Unlike [25, 26], here decomposing the rate-coded error backpropagation into the macro and micro levels enables computation of the gradient of the actual loss function with respect to the tunable weights, leading to highly competitive performances. Our HM2-BP algorithm can introduce/remove multiple spikes by one update, greatly improving learning efficiency in comparison with SpikeProp [33]. To complete the derivation of HM2-BP, derivatives in the forms of $\frac{\partial e_{i|j}^k}{\partial o_i^k}$ and $\frac{e_{i|j}^k}{\partial o_j^k}$ as needed in (2.17) and (2.22) are yet to be estimated, which is non-trivial as shall be presented in Section 2.1.3.

2.1.3 Decoupled Micro-Level Model for S-PSP

The derivatives of the S-PSP $e_{i|j}^k$ with respect to the pre and post-synaptic neuron firing counts are key components in our HM2-BP rule. According to (2.9), the S-PSP $e_{i|j}^k$ is dependent on both rate and temporal information of the pre and post-synaptic spikes. The firing counts of pre and post-synaptic neurons (i.e., the rate information) are represented by the two nested summations in (2.9). The exact firing timing information determines the (normalized) post-synaptic potential ϵ of each pre/post-synaptic spike train pair as seen from (2.8). The rate and temporal information of spike trains are strongly coupled together, making the exact computation of $\frac{\partial e_{i|j}^k}{\partial o_i^k}$ and $\frac{e_{i|j}^k}{\partial o_j^k}$ challenging.

To address this difficulty, we propose a decoupled model for $e_{i|j}^k$ to untangle the rate and temporal effects. The model is motivated by the observation that $e_{i|j}^k$ is linear in both o_j^k and o_i^k in the limit of high firing counts. For finite firing rates, we decompose $e_{i|j}^k$ into an asymptotic rate-dependent effect using the product of o_j^k and o_i^k and a correction factor $\hat{\alpha}$ accounting for temporal correlations between the pre and post-synaptic spike trains

$$e_{i|j}^k = \hat{\alpha}(\mathbf{t}_j^{(f)}, \mathbf{t}_i^{(f)}) o_j^k \, o_i^k. \tag{2.25}$$

$\hat{\alpha}$ is a function of exact spike timing. Since the SNN is trained incrementally with small weight updates set by a well-controlled learning rate, $\hat{\alpha}$ does not change substantially by one training iteration. Therefore, we approximate $\hat{\alpha}$ by using the values of $e_{i|j}^k$, o_j^k, and o_i^k available before the next training update by

$$\hat{\alpha}(\mathbf{t}_j^{(f)}, \mathbf{t}_i^{(f)}) \approx \frac{e_{i|j}^k}{o_j^k o_i^k}.$$

With the micro-level temporal effect considered by $\hat{\alpha}$, we estimate the two derivatives by

$$\frac{\partial e_{i|j}^k}{\partial o_i^k} \approx \hat{\alpha}\ o_j^k = \frac{e_{i|j}^k}{o_i^k}, \quad \frac{\partial e_{i|j}^k}{\partial o_j^k} \approx \hat{\alpha}\ o_i^k = \frac{e_{i|j}^k}{o_j^k}.$$

Our hybrid training method follows the typical backpropagation methodology. First of all, a forward pass is performed by analytically simulating the LIF model (2.3) layer by layer. Then the firing counts of the output layer are compared with the desirable firing levels to compute the macro-level error. After that, the error in the output layer is propagated backwards at both the macro and micro levels to determine the gradient. Finally, an optimization method (e.g. Adam [72]) is used to update the network parameters given the computed gradient.

2.2 Experiments and Results

2.2.1 Experimental Settings and Datasets

The weights of the experimented SNNs are randomly initialized by using the uniform distribution $U[-a, a]$, where a is 1 for fully connected layers and 0.5 for convolutional layers. We use fixed firing thresholds in the range of 5 to 20 depending on the layer. We adopt the exponential weight regularization scheme in [25] and introduce the lateral inhibition in the output layer to speed up training convergence [25], which slightly modifies the gradient computation for the output layer (see **Appendix A**). We use Adam [72] as the optimizer and its parameters are set according to the original Adam paper. We impose greater sample weights for incorrectly recognized data points during the training as a supplement to the Adam optimizer.

The MNIST handwritten digit dataset [35] consists of 60k samples for training and 10k for testing, each of which is a 28×28 grayscale image. We convert each pixel value of a MNIST image into a spike train using Poisson sampling based on which the probability of spike generation is proportional to the pixel intensity. The N-MNIST dataset [36] is a neuromorphic version of the

MNIST dataset generated by tilting a Dynamic Version Sensor (DVS) [73] in front of static digit images on a computer monitor. The movement induced pixel intensity changes at each location are encoded as spike trains. Since the intensity can either increase or decrease, two kinds of ON- and OFF-events spike events are recorded. Due to the relative shifts of each image, an image size of 34×34 is produced. Each sample of the N-MNIST is a spatio-temporal pattern with $34 \times 34 \times 2$ spike sequences lasting for $300ms$. We reduce the time resolution of the N-MNIST samples by 600x to speed up simulation.

2.2.2 Fully Connected SNNs for the Static MNIST

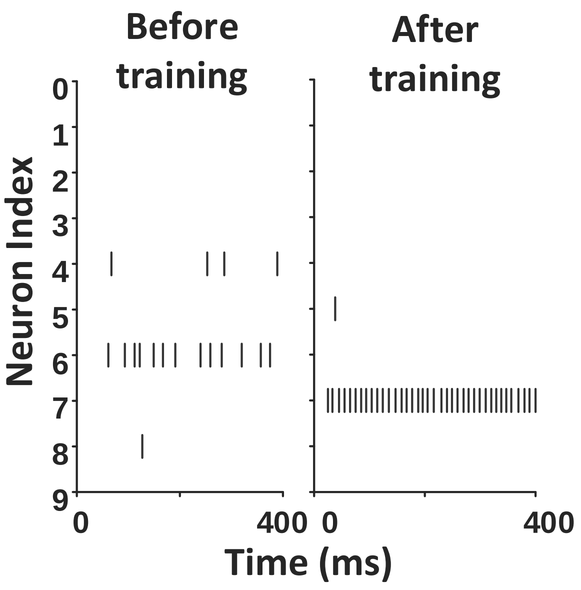

Figure 2.5: The spike raster plot of the output layer for the inference on digit 7 before and after the training.

Using Poisson sampling, we encode each 28×28 image of the MNIST dataset into a 2D $784 \times L$ binary matrix, where $L = 400ms$ is the duration of each spike sequence, and a 1 in the matrix represents a spike. The simulation time step is set to be $1ms$. No pre-processing or data augmentation is done in our experiments. Fig. 2.5 shows the output layer spike patterns of the inference on the digit 7 before and after training a $784 - 800 - 10$ network. No specific firing patterns can be found initially, but after training the corresponding 7_{th} neuron fires most actively while other neurons remain silent, suggesting the correct classification.

Table 2.1 compares the performance of SNNs trained by the proposed HM2-BP rule with other algorithms. HM2-BP achieves 98.93% test accuracy, outperforming STBP [26], which is the best previously reported algorithm for fully-connected SNNs. The proposed rule also achieves the best accuracy earlier than STBP (100 epochs v.s. 200 epochs). We attribute the overall improvement to the hybrid macro-micro processing that handles the temporal effects and discontinuities at two levels in a way such that explicit back-propagation of the rate-coded error becomes possible and practical. Notice that our SNN also performs better than the ANNs trained with Dropout or Drop-connect [74, 75].

Table 2.1: Comparison of different SNN models on MNIST

Model	Hidden layers structure	Accuracy	Epochs
Standard ANN [76]	800-800	98.4%	>100
Dropout ANN [74]	1024-1024-1024	98.75%	-
Drop-connect ANN [75]	800-800	98.88%	1020
Spiking MLP (converted*) [30]	500-500	94.09%	50
Spiking MLP (converted*) [29]	500-200	98.37%	160
Spiking MLP (converted*) [27]	1200-1200	98.64%	50
Spiking MLP [31]	300-300	97.80%	50
Spiking MLP [25]	800	98.71%[a]	200
Spiking MLP (STBP) [26]	800	98.89%	200
Spiking MLP (this work)	800	**98.93%**/98.81%	**100**/50

We only compare SNNs without any pre-processing (i.e., data augmentation) except for [30].
* means the model is converted from an ANN. [a] [25] achieves 98.88% with hidden layers of 300-300.

2.2.3 Fully Connected SNNs for N-MNIST

The simulation time step is $0.55ms$ for N-MNIST. Table 2.2 compares the results obtained by different models on N-MNIST. The first two results are obtained by the conventional CNNs with the frame-based method, which accumulates spike events over short time intervals as snapshots and recognizes digits based on sequences of snapshot images. The relative poor performances of the first two models may be attributed to the fact that the frame-based representations tend to be blurry and do not fully exploit spatio-temporal patterns of the input. The two non-spiking LSTM models, which are trained directly on spike inputs, do not perform too well, suggesting that LSTMs may be incapable of dealing with asynchronous and sparse spatio-temporal spikes. The SNN trained by our proposed approach naturally processes spatio-temporal spike patterns, achieving the start-of-the-art accuracy of 98.88%, outperforming the previous best ANN (97.38%) and SNN (98.78%) with significantly less training epochs required.

Table 2.2: Comparison of different models on N-MNIST

Model	Hidden layers structure	Accuracy	Epochs
Non-spiking CNN [77]	-	95.30%	-
Non-spiking CNN [78]	-	98.30%	15-20
Non-spiking LSTM [77]	-	97.05%	-
Non-spiking Phased-LSTM [77]	-	97.38%	-
Spiking CNN (converted*) [78]	-	95.72%	15-20
Spiking MLP [79]	10000	92.87%	-
Spiking MLP [25]	800	98.74%	200
Spiking MLP (STBP) [26]	800	98.78%	200
Spiking MLP (this work)	800	**98.88%**/98.77%	**60**/15

Only structures of SNNs are shown for clarity.* means the SNN model is converted from an ANN.

2.2.4 Spiking Convolution Network for the Static MNIST

We construct a spiking CNN consisting of two 5×5 convolutional layers with a stride of 1, each followed by a 2×2 pooling layer, and one fully connected hidden layer. The neurons in the pooling layer are simply LIF neurons, each of which connects to 2×2 neurons in the preceding convolutional layer with a fixed weight of 0.25. Similar to [25, 26], we use elastic distortion [76] for data augmentation. As shown in Table 2.3, our proposed method achieves an accuracy of 99.49%, surpassing the best previously reported performance [26] with the same model complexity after 190 epochs.

Table 2.3: Comparison of different spiking CNNs on MNIST

Model	Network structure	Accuracy
Spiking CNN (converted[a]) [27]	28×28-12C5-P2-64C5-P2-10	99.12%
Spiking CNN (converted[b]) [28]	-	92.70%[c]
Spiking CNN (converted[a]) [32]	-	99.44%
Spiking CNN [25]	28×28-20C5-P2-50C5-P2-200-10	99.31%
Spiking CNN (STBP) [26]	28×28-15C5-P2-40C5-P2-300-10	99.42%
Spiking CNN (this work)	28×28-15C5-P2-40C5-P2-300-10	**99.49%**

[a] converted from a trained ANN. [b] converted from a trained probabilistic model with binary weights.

[c] performance of a single spiking CNN. 99.42% obtained for ensemble learning of 64 spiking CNNs.

2.2.5 Training Complexity Comparison and Implementation

Unlike [26], our hybrid method does not unwrap the gradient computation in the time domain, roughly making it $O(N_T)$ times more efficient than [26], where N_T is the number of time points in each input example. The proposed method can be easily implemented.

2.3 Summary and Discussions

In this chapter, we present a novel hybrid macro/micro level error backpropagation scheme to train deep SNNs directly based on spiking activities. The spiking timings are exactly captured in the spike-train level post-synaptic potentials (S-PSP) at the microscopic level. The rate-coded error is defined and efficiently computed and back-propagated across both the macroscopic and microscopic levels. We further propose a decoupled S-PSP model to assist gradient computation at the micro-level. In contrast to the previous methods, our hybrid approach directly computes the gradient of the rate-coded loss function with respect to tunable parameters. We demonstrate the best performances for both fully connected and convolutional SNNs over the static MNIST and dynamic N-MNIST datasets, outperforming the best previously reported SNN training techniques.

Furthermore, the proposed approach also achieves competitive performances better than those of the conventional deep learning models when dealing with asynchronous spiking streams.

The performances achieved by the proposed BP method may be attributed to the fact that it addresses key challenges of SNN training in terms of scalability, handling of temporal effects, and gradient computation of loss functions with inherent discontinuities. Coping with these difficulties through error backpropagation at both the macro and micro levels provides a unique perspective to training of SNNs. More specifically, orchestrating the information flow based on a combination of temporal effects and firing rate behaviors across the two levels in an interactive manner allows for the definition of the rate-coded loss function at the macro level, and backpropagation of errors from the macro level to the micro level, and back to the macro level. This paradigm provides a practical solution to the difficulties brought by discontinuities inherent in an SNN while capturing the micro-level timing information via S-PSP. As such, both rate and temporal information in the SNN is exploited during the training process, leading to the state-of-the-art performances. We expect this work would help move the community forward towards enabling high-performance spiking neural networks and neuromorphic computing.

3. ADAPTIVE LIQUID STATE MACHINE ARCHITECTURES WITH EFFICIENT RESERVOIR TUNING*

3.1 Activity-based Probabilistic STDP (AP-STDP) for Efficient Reservoir Computing

While the standard LSM serves as a good trade-off between the tractable training of recurrent SNNs and exploiting the computation power. The randomly fixed recurrent reservoir that generated in a generic way may not be efficacious for specific applications. In spite that the spike-timing dependent plasticity (STDP) is a promising unsupervised algorithm for tuning reservoirs, the ongoing small-valued updates and high-resolution realization of the STDP rule inevitably demands high overhead when it comes to hardware implementation. Furthermore, the synaptic weight saturation is a serious issue in learning. Without a specific stop-learning mechanism, the continuous weight modifications can quickly saturate a synapse and prevent it from learning the new experience. Therefore, the STDP rule must be carefully designed for cost-effectiveness and addressing synaptic saturation issue.

To address the above challenges, this work proposes a novel activity-based probabilistic STDP (AP-STDP) rule to tune plastic reservoirs. The proposed rule achieves good learning performance with low synaptic weight resolutions by incorporating a probabilistic weight update process. Furthermore, AP-STDP prevents memory saturation by introducing activity-level based weight tuning with a stop-learning condition. By performing principal component analysis of the reservoir dynamics, we demonstrate that AP-STDP gives rise to more effective internal representations of input samples compared to other simpler STDP rules. We use the spoken English letters from the widely adopted TI46 Speech Corpus [80] as a real-world speech recognition benchmark to test the performance of liquid state machines tuned with AP-STDP. It is demonstrated that the proposed

*©2016 IEEE. Reprinted, with permission, from Yingyezhe Jin and Peng Li, "AP-STDP: A novel self-organizing mechanism for efficient reservoir computing", *Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, July 2016. ©2016 IEEE. Reprinted, with permission, from Yingyezhe Jin, Yu Liu and Peng Li, "SSO-LSM: A Sparse and Self-Organizing architecture for Liquid State Machine based neural processors", *Proceedings of the 2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*. IEEE, July 2016.

AP-STDP outperforms other conventional STDP rules and can produce a performance that is better than some of the best reported recognition performances obtained using fixed reservoirs [10].

The rest of the section is organized as follows. Section 3.1.1 provides a brief description of the background and motivations of this work. Section 3.1.2 presents our proposed AP-STDP rule. In Section 3.1.3, the PCA analysis of reservoir dynamics is introduced. The network settings and benchmark are described in Section 3.1.4. The experimental results are reported in Section 3.1.5. Finally, Section 3.1.6 concludes this work.

3.1.1 Standard STDP Rules

We briefly discuss the standard STDP rules and their limitations pertaining to self-organizing reservoirs.

STDP is a local unsupervised Hebbian learning mechanism realizing synaptic plasticity based on the respective firing timing orders of the presynaptic and postsynaptic neurons. The synapse w_{ij} from neuron j to neuron i is potentiated if a causal order (i.e., pre fires before post) is observed, or depressed if the postsynaptic neuron fires before the presynaptic neuron. The change in weight depends on the temporal difference $\Delta t = t_{post} - t_{pre}$ between the specific pair of pre- and postsynaptic spikes:

$$\begin{aligned} \Delta w^{+} &= F_{+}(w) \cdot e^{-\frac{|\Delta t|}{\tau_{+}}} \quad \textit{if}\, \Delta t > 0 \\ \Delta w^{-} &= F_{-}(w) \cdot e^{-\frac{|\Delta t|}{\tau_{-}}} \quad \textit{if}\, \Delta t < 0, \end{aligned} \tag{3.1}$$

where Δw^{+} and Δw^{-} represent the weight modification induced by long-term potentiation (LTP) and long-term depression (LTD), and $F_{\pm}(w)$ describes the dependency of the update on the current weight value. If $F_{\pm}(w) = A_{\pm}$ is fixed, it is called an additive STDP rule. If $F_{\pm}$ is proportional to the current weight w, it is called a multiplicative STDP rule. Additive STDP rules are usually applied to the reservoir because they have been shown to generate good self-organizing behaviors [81].

A typical additive STDP curve is plotted in Fig 3.1(a). And there are generally two pairing rules for the implementation of STDP: all-pairing and nearest-neighbor (shown in Fig. 3.1(b)). In the first scheme, synaptic updates are triggered by all possible pre-post spike pairs before and at the current time t. In nearest-neighbor pairing, instead, at each firing time, a pre-synaptic (post-synaptic) spike is only paired with the closed preceding post-synaptic (pre-synaptic) spike. As a common practice, excitatory synapses in the reservoir are usually assumed to be plastic and are tuned with STDP while inhibitory synapses are fixed.

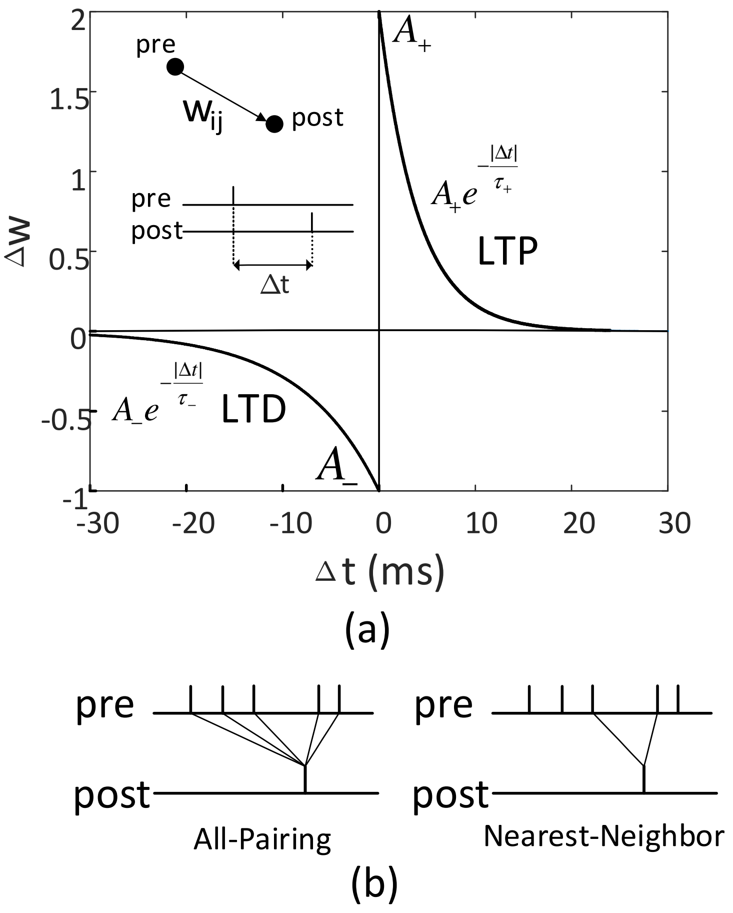

Figure 3.1: (a) A typical additive STDP curve. (b) Two pairing schemes. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 IEEE.

No matter which pairing rule is adopted, realizing (3.1) can produce weight updates that are arbitrarily small as the temporal difference Δt increases. The fact that a high synaptic resolution is needed to accommodate small updates and the number of such updates can be very large leading to a high level of inefficiency. On the other hand, simply reducing hardware overhead by adopting a low resolution can lead to learning performance degradations. Another disadvantage of this standard STDP rule is that persistent firing activities can eventually saturate the storage capacity of a given synapse and push the weight to the upper/lower limit, preventing the reservoir from adapting to subsequent input samples and degrading learning performance.

3.1.2 The Activity-based Probabilistic STDP

We first propose a simple probabilistic STDP rule for reservoir computing targeting low synaptic weight resolutions. Using the probabilistic rule as a starting point, we further propose an Activity-based Probabilistic STDP (AP-STDP) rule incorporating a stop-learning condition based upon the postsynaptic neural firing activity level. AD-STDP addresses both the resolution and memory saturation challenges discussed in Section 3.1.1.

The Probabilistic STDP

Inspired by two stochastic learning rules developed under contexts different from reservoir computing [82, 83], we propose a simple probabilistic STDP rule for reservoir tuning. The proposed probabilistic rule first computes the weight change $\Delta w^+/\Delta w^-$ according to (3.1). Instead of directly applying the weight change, the rule probabilistically commits a fixed amount of weight update with a probability that is proportional to the magnitude of $\Delta w^+/\Delta w^-$:

$$\begin{aligned} w &\leftarrow w + \Delta W \; \textit{with}\; p \propto |\Delta w^+| \quad \textit{if}\; \Delta t > 0 \\ w &\leftarrow w - \Delta W \; \textit{with}\; p \propto |\Delta w^-| \quad \textit{if}\; \Delta t < 0, \end{aligned} \tag{3.2}$$

where ΔW is the fixed weight update, representing the resolution of synaptic weights, i.e., a large ΔW leads to a low resolution and a lower hardware overhead. Furthermore, since the total number of weight updates is limited by the probabilities, the update process of this rule is much more efficient than that of the standard STDP. It is worthwhile mentioning that the probabilistic update of synaptic weights can be easily realized in hardware by using efficient random number generator (RNG) primitives.

In terms of performance, it has been argued that continuously fast weight updates of synapses with a limited number of states (e.g. due to a low synaptic resolution) can result in bad memory performance. This manifests itself in such a way that the most recent experiences are represented and learned by the synapses better than the older ones [84, 85]. The proposed probabilistic STDP addresses the above problem by slowing down the learning procedure and helping maximally utilize the synaptic storage capacity [47]. As a result, it more evenly distributes the memory across the network so as to well represent both the new and old experiences and allow for retrieval them at a later time. However, the issue of synaptic memory saturation remains unresolved.

The Activity-based Probabilistic STDP

Although the probabilistic STDP rule helps to slow down the learning for better memory performance, it has no stop-learning condition imposed to tackle the issue of synaptic memory saturation. In the unsupervised learning process of STDP rules, new input samples are fed into the reservoir and synaptic weights are tuned to capture the internal patterns of the inputs. The reservoir will not be able to learn from new stimuli once most of its synapses are over-potentiated or over-depressed (i.e., the synaptic memory is saturated). Incorporating a stop-learning condition provides an effective way to prevent memory saturation [47], which is particularly desirable for synapses with a finite number of states.

Conceptually, we may deactivate LTP when over-potentiation of synapses happens and deactivate LTD when over-depression takes place. This can be done for each synapse by monitoring the firing activity of the postsynaptic neuron. For example, if the postsynaptic neuron is overly active and has fired a lot of spikes, we could "turn off" the LTP of the STDP rule because the afferent synapses of this neuron may have been over-potentiated by this time. Similarly, if the postsynaptic neuron is inactive, we might "switch off" the LTD of the STDP rule to stop/prevent the synapses from being over-depressed. The above mechanism regulates the network dynamics and helps produce more orderly self-organizing behaviors in the reservoir which are crucial to good performance.

While the instantaneous firing frequency acts as a direct measure for the activity level of a neuron, a measure at a longer timescale may correlate better with the average firing level induced by both the new and old inputs. Motivated by [47], we adopt the internal calcium concentration of a biological neuron as an indicator for the firing activity within a specified time window in the proposed AP-STDP rule. The calcium variable $c(t)$ follows the first order dynamics with a large time constant and is a function of the postsynaptic neuron activity:

$$\frac{dc(t)}{dt} = -\frac{c(t)}{\tau_c} + \sum_i \delta(t - t_i), \tag{3.3}$$

where τ_c is the time constant and the summation is over all postsynaptic spikes arriving at time t_i.

We now discuss the idea of the stop-learning condition based on the calcium concentration. First of all, a threshold of calcium variable c_θ is defined for determining whether the neuron is active or inactive. Suppose a postsynaptic neuron is active and it has fired many spikes. Its corresponding calcium concentration must be very high. Our stop-learning condition basically enables the LTP of the corresponding synapses only when c is within a specific range. Otherwise, it deactivates LTP to avoid over-potentiation. Similarly, if a postsynaptic neuron is inactive and its calcium concentration is too low, implying that afferent synapses are possibly over-depressed, LTD would be disabled. More specifically, the proposed stop-learning condition is combined with the probabilistic STDP rule presented in the last subsection in the following form:

$$\begin{aligned} &w \leftarrow w + \Delta W \textit{with}\ p \propto |\Delta w^+| \ \ \textit{if}\ \Delta t > 0\ \&\& \\ &\qquad\qquad\qquad\qquad\qquad c_\theta < c < c_\theta + \Delta c \\ &w \leftarrow w - \Delta W \textit{with}\ p \propto |\Delta w^-| \ \ \textit{if}\ \Delta t < 0\ \&\& \\ &\qquad\qquad\qquad\qquad\qquad c_\theta > c > c_\theta - \Delta c, \end{aligned} \tag{3.4}$$

where c_θ is the threshold for determining the postsynaptic neuron firing activity, and Δc is a margin that is introduced to realize the stop-learning condition.

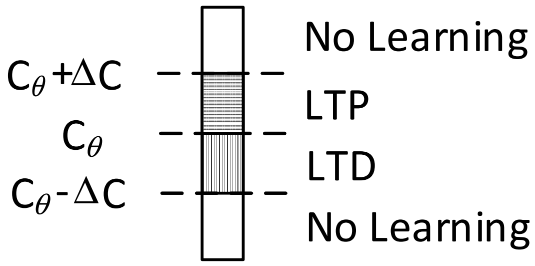

Figure 3.2: The AP-STDP rule with the stop-learning condition. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 IEEE.

As shown in Fig. 3.2, we allow a synapse to be potentiated if and only if the calcium concentration c of the postsynaptic neuron is in the range of $[c_\theta, c_\theta + \Delta c]$. Similarly, a synapse is depressed if and only if the calcium concentration falls in the range of $[c_\theta - \Delta c, c_\theta]$. No long-term modification is induced if the calcium level of the postsynaptic neuron is too low or too high. This regulatory mechanism protects the synaptic memory against modifications triggered by the ongoing spontaneous activity, prevents the synapses from over-potientation or over-depression, and allows for full exploration of all input samples in the learning process.

By introducing the stop learning rule based on the calcium level, we correlate the postsynaptic neuron firing activity with the synapse plasticity to avoid saturation. Note that since LTP and LTD are only allowed when the calcium concentration falls within the respective limits, no global positive feedback effect exists. In addition, the calcium concentration follows the first order dynamics of (3.3), and hence its value can increase or decrease depending on the balance between post-synaptic firing activities and the leakage. As a result, the imposed stop learning rule allows the synaptic weight to change freely without holding it at a specific high or low value.

Since the nearest-neighbor pairing scheme can be done more easily than the all-pairing scheme and the difference in performance between the two schemes is subtle, in this section, we only focus on nearest-neighbor pairing for our AP-STDP rule to reduce the implementation overhead.

3.1.3 Internal Representation Ability of Self-Organizing Reservoirs

To shed light on how the network dynamics may be impacted by the applied STDP mechanisms, hence leading to different learning performances, we perform principal component analysis (PCA) on the reservoir responses induced by various input samples. We define the network state at a given time t as a binary vector $\mathbf{s}(\mathbf{t}) \in \{0,1\}^{N^r}$, where N^r is the total number of reservoir neurons, and $s_i(t)$ is 1 if and only if the i^{th} reservoir neuron fires at time t. The network state $\mathbf{s}(\mathbf{t})$ specifies the reservoir response at t. The reservoir response matrix at time t is defined as:

$$\mathbf{R}(\mathbf{t}) = \{\mathbf{s^0}(\mathbf{t}), \mathbf{s^1}(\mathbf{t}), \cdots, \mathbf{s^j}(\mathbf{t}), \cdots, \mathbf{s^N}(\mathbf{t})\}, \tag{3.5}$$

where j is the index of the input samples, and N is the number of the input samples considered. The j-th column vector of $\mathbf{R}(\mathbf{t})$, or $\mathbf{s^j}(\mathbf{t})$, represents the network state at time t given the j^{th} input. The defined reservoir response matrix $\mathbf{R}(\mathbf{t})$ is a snapshot of the complex reservoir dynamics at time t considering all input samples (shown in Fig. 3.3). By analyzing the response matrix $\mathbf{R}(\mathbf{t})$, we can better understand how well the input samples are represented by the network dynamics.

Applying PCA to the response matrix $R(t)$ allows us to visualize the reservoir responses in the projection space expanded by the first few, say three, principal components (PCs). For the reservoir with weak internal representation capability, the projected responses of different class labels are expected to overlap with each other. In contrast, the formation of tight intra-class clusters with small or no inter-class overlaps is indicative of effective internal representation. We further evaluate the network dynamics by calculating the amount of variance explained by the first several PCs. The greater the variance that is explained by the first several PCs, the more orderly the network dynamics is, suggesting that the internal structures of the input samples can be better captured by the network response. The experimental results of PCA on the reservoir responses are reported in Section 3.1.5.

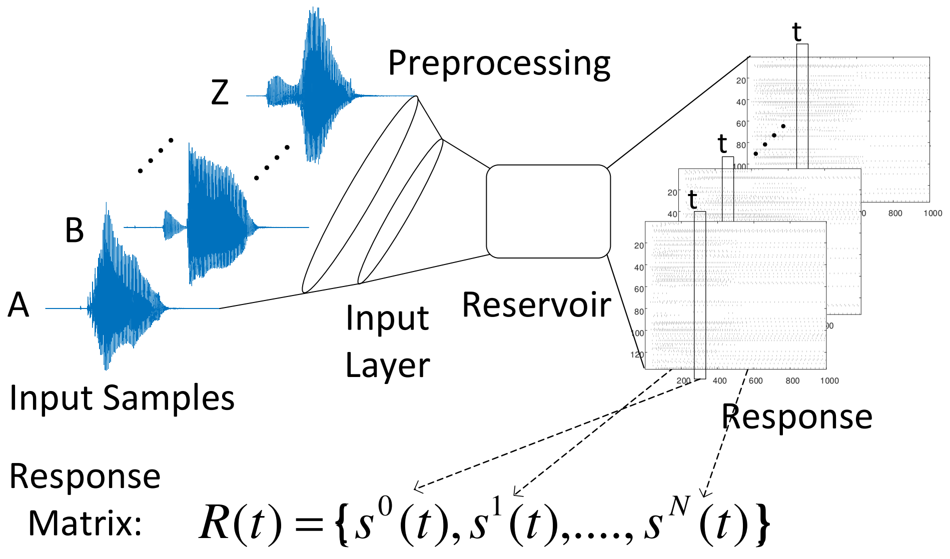

Figure 3.3: Extract the response matrix $R(t)$ from the reservoir responses. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 IEEE.

3.1.4 Experimental Settings and Benchmark

The reservoirs of two liquid state machines are set up using the approach described in [10], giving rise to a recurrent network of 135 and 90 reservoir neurons on a 3D grid, respectively. 80% of the reservoir neurons are excitatory while the rest of them are inhibitory. The connectivity between any two neurons is constructed randomly under a probabilistic distribution function such that the wiring probability of two neurons drops exponentially with the distance between them:

$$P(i,j) = k \cdot e^{-\frac{D(i,j)}{r^2}} \quad (i \neq j), \tag{3.6}$$

where $D(i,j)$ is the Euclidean distance between neuron i and neuron j, and r and k are two control parameters chosen as suggested in [10]. We adopt the discrete LIF neuronal model and the second-order synaptic model described in [10].

The parameters of all STDP rules described in this work are shown in Table 3.1. The maximum synaptic weight W_{max} is set to 8.0. The initial weights of excitatory synapses are set to ΔW while inhibitory synaptic weights are initialized to $-\Delta W$. We set the bit-width of reservoir synaptic weights to 4 bits. For comparison purposes, we have tuned parameters of the probabilistic STDP rule such that the total number and amount of weight updates are roughly the same for both the AP-STDP and probabilistic STDP rules.

Table 3.1: Parameter settings of the standard STDP, probabilistic STDP and AP-STDP Rules. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 IEEE.

Parameter	**Value**
A_+	8.0
A_-	4.0
τ_+	2.0
τ_-	4.0
ΔW	1.0
c_θ	5.0
Δc	3.0
τ_c	64.0

The adopted benchmark is a subset of the TI46 speech corpus [80]. This benchmark contains 10 utterances of each English letter from "A" to "Z", which were recorded from a single speaker. There are 260 samples in this benchmark. The time domain speech signals are preprocessed by Lyon's passive ear model [86], and encoded into 83 spike trains using the BSA algorithm [87]. Each input spike train generated in the preprocessing stage is sent to 32 randomly selected reservoir neurons with a fixed weight randomly chosen to be 2 or -2. The readout layer is fully connected to the reservoir with plastic synapses trained using the bio-inspired supervised learning algorithm proposed in [10].

Before training the readout layer, all speech samples are presented to each plastic reservoir one by one while a STDP rule is applied to tune the reservoir synapses. The process is repeated for a sufficient number of iterations till the reservoir synaptic weights converge. Then, the readout layer is trained with the learning algorithm described by [10]. We adopt a 5-fold cross validation scheme to test the recognition performance for each LSM network setting by randomly dividing all speech samples into 5 groups. The recognition decision is made right after each testing speech sample is presented. At this time, the readout neuron with the highest firing frequency is chosen as the winner whose class label is deemed to be the classification decision.

3.1.5 Experimental Results

Using the experimental setups described in Section 3.1.4, we compare five reservoir tuning settings, which are abbreviated in Table 3.2, for two reservoir sizes, namely 135 and 90 neurons, respectively.

Table 3.2: Reservoirs Tuning Methods. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 IEEE.

Abbreviation	**STDP Rule and Pairing Scheme Used**
Static	Randomly Generated Fixed Reservoir
AAP	Standard Additive STDP w/ All-Pairing
ANN	Standard Additive STDP w/ Nearest-Neighbor
PAAP	Probabilistic STDP w/ All-Pairing
PANN	Probabilistic STDP w/ Nearest-Neighbor
Proposed	Activity-based Probabilistic STDP w/ Nearest-Neighbor

Principal Component Analysis for Reservoir Dynamics

We first analyze reservoir dynamics using PCA. For this, we collect the reservoir response matrix $\mathbf{R}(\mathbf{t})$ at a randomly selected time $t_0 = 434\ ms$ and perform PCA to the reservoir responses for both static and plastic reservoirs. We visualize the projected responses in the projection space spanned by the first three PCs in Fig. 3.4. The static reservoir and the plastic reservoirs tuned using the standard STDP create visible overlaps across different speech classes without forming good intra-class clusters. This again indicates that a randomly generated reservoir may not be effective for a specific application. Fig. 3.4 also suggests that simply adopting the standard STDP does not

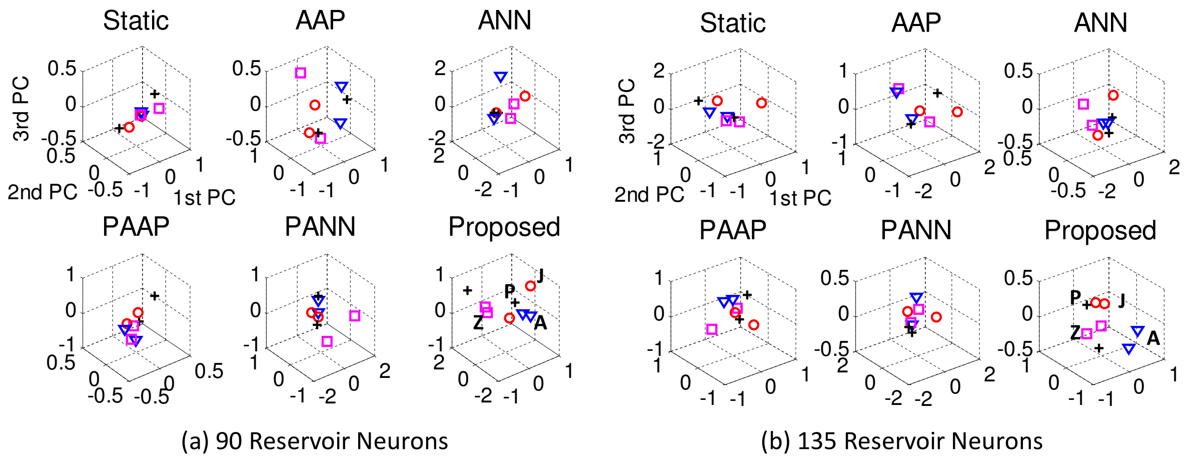

Figure 3.4: Visualization of the reservoir responses in the PCs space for different spoken letters. For simplicity, we visualize the responses of input speech samples with four different class labels (letters). The four class labels 'A', 'J', 'P' and 'Z' are marked as '$\bigtriangledown$', circle '$\circ$', cross '+' and square '$\square$', respectively. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 IEEE.

produce the desired self-organizing behaviors when the reservoir synapses have a low resolution.

Although intra-class clusters are partially formed in some cases with the probabilistic STDP rule, overlaps across different classes still exist. This suggests that the resulting poor input representation power may be attributed to the occurrence of synaptic memory saturation due to the lack of stop-learning rules. This conclusion is supported by the results of the proposed AP-STDP rule, which produces compact intra-class clusters and also well separates different classes.

Further analysis of the variance explained by the first several PCs offers additional insights on the internal representation effectiveness of different reservoirs as reported in Table 3.3. We use the static reservoir case as a baseline reference and visualize the change in the explained variance due to the adoption of different STDP rules in Fig. 3.5. As shown in Table 3.3 and Fig. 3.5, the standard and probabilistic STDP rules either make no significant improvements over the static baseline or even underperform it, reaffirming the potential weaknesses of these rules.

In the case of AP-STDP, there is a greater amount of variance explained by the first several PCs compared to the static baseline and other STDP rules. This is consistently the case for the

Table 3.3: Amount of variance explained by the first several PCs for different reservoir tuning methods. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 IEEE.

	Principal Components					
	135 Reservoir Neurons			90 Reservoir Neurons		
# of PCs	5	20	65	5	20	50
Static	21.8%	55.4%	94.2%	23.5%	65.3%	96.6%
AAP	20.3%	55.1%	94.6%	23.3%	65.8%	96.6%
ANN	18.7%	52.4%	93.2%	24.1%	67.1%	96.5%
PAAP	17.8%	51.6%	92.8%	23.0%	64.8%	96.2%
PANN	19.6%	54.0%	93.6%	24.3%	68.4%	96.0%
Proposed	**21.5%**	**57.2%**	**94.4%**	**28.4%**	**73.9%**	**97.8%**

reservoirs with 135 neurons and 90 neurons. As seen in Fig. 3.5, up to 2% and 6% more variance can be explained with AP-STDP compared to the static reservoir. These results suggest that the network dynamics induced by AP-STDP have an improved internal representational structure.

Recognition Performances

Table 3.4: Recognition performances of the LSMs with different reservoirs. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 IEEE.

Reservoir Tuning	**135 Reservoir Neurons**	**90 Reservoir Neurons**
Static	92.3%	89.6%
AAP	92.3%	88.4%
ANN	90.4%	89.6%
PAAP	93.5%	88.1%
PANN	92.3%	89.2%
Proposed	**94.2%**	**92.3%**

We use the adopted benchmark described in Section 3.1.4 to test the LSM recognition rates as reported in Table 3.4. We also plot the performance boosts over the static baseline achieved by the plastic reservoirs in Fig. 3.6. To the best knowledge of the authors, the best reported performance

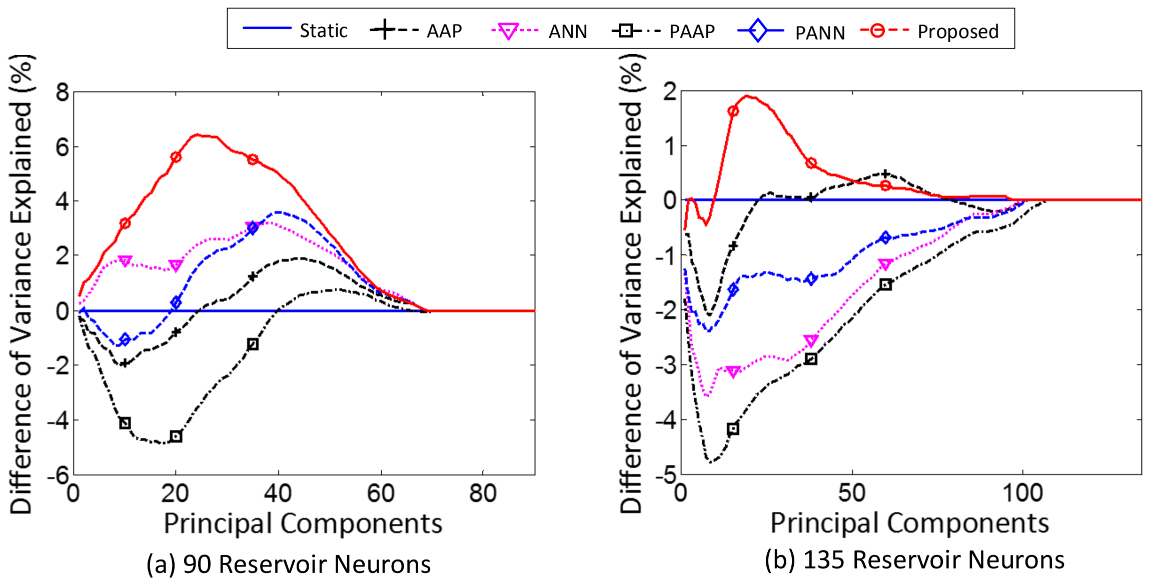

Figure 3.5: The differences between the plastic reservoirs and the static reservoir (baseline reference) in terms of the variances explained by the first several PCs. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 IEEE.

on the same benchmark achieved by a static LSM with 135 reservoir neurons is 92.3% [10]. The recognition performance of our static reservoir with 135 neurons achieves the same recognition rate of 92.3%. When the reservoir size is reduced to 90 neurons, the recognition rate of our static reservoir becomes 89.6%.

In comparison with the static baseline, the standard STDP rule degrades the performance as shown in Table 3.4 and Fig. 3.6. The improvements of the probabilistic STDP rule over the static reservoir are not consistent. It in fact leads to performance degradation in some cases. As shown in Table 3.4, the performance of AP-STDP is superior than other STDP rules. AP-STDP boosts the performance by 1.9% compared to the best reported performance obtained under a static reservoir for the reservoir size of 135 neurons. AP-STDP produces a good recognition rate of 92.3% when the reservoir has only 90 neurons. The performance boost over the static baseline is 2.7% in this case.

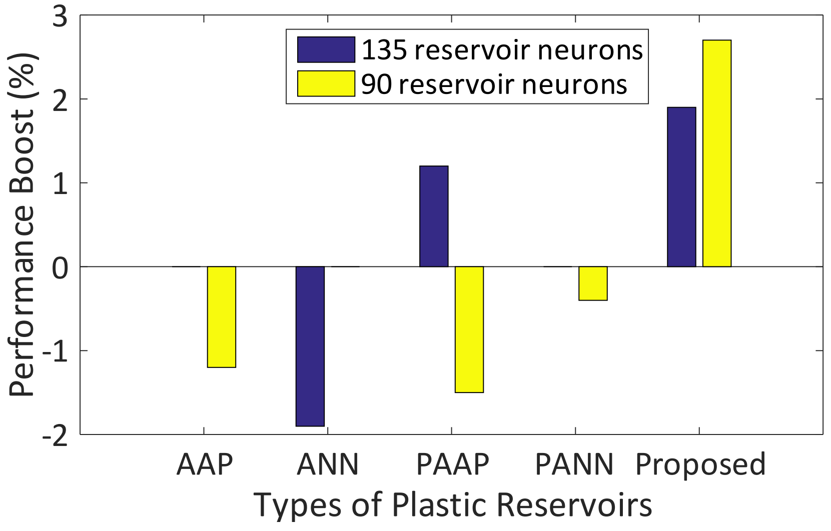

Figure 3.6: The performance boosts over the static reservoir achieved by different plastic reservoirs. The proposed AP-STDP significantly boosts the performance for both the 90-neuron and 135-neuron reservoirs. AAP, ANN, and PANN lead to close-to-zero performance boosts in some cases. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 IEEE.

3.1.6 Summary

In this section, we have proposed a novel activity-based probabilistic STDP (AP-STDP) rule as a promising self-organizing approach to construct plastic recurrent reservoirs in the context of the liquid state machine. Through a probabilistic update mechanism, AP-STDP achieves good learning performance and facilities efficient reservoir tuning at low synaptic weight resolutions. Furthermore, AP-STDP addresses the issue of synaptic memory saturation by imposing a stop-learning condition based an activity measure. AP-STDP is shown to outperform all other studied STDP rules based on the principle component analysis of the network dynamics and realistic performance benchmarking using speech recognition.

3.2 An Efficient, Sparse and Self-Organizing LSM (SSO-LSM) Learning Architecture

The aim of this work is to enable efficient adaptive LSM neural processors by proposing a novel Sparse and Self-Organizing LSM (SSO-LSM) architecture. The first key element of the SSO-LSM architecture is the exploration of a low-overhead hardware-friendly Spike-Timing Dependent Plastic (STDP) mechanism for efficient on-chip reservoir tuning, which is motivated by the following considerations. While training recurrent networks (e.g. reservoirs) is in general very challenging, STDP can supplement the training of readout synapses, and further boost learning performance through reservoir tuning [37, 39, 41, 88]. The simplicity and unsupervised nature of STDP is specially amenable to hardware realization. Equally important, STDP can produce attractive self-organizing behaviors in the reservoir that naturally lead to a sparser recurrent network, which we explore as a great opportunity for runtime energy reduction in SSO-LSM.

Nevertheless, realizing STDP in a digital hardware architecture presents interesting challenges, particularly when extremely low bit resolutions are targeted for low hardware and energy overhead. As we will show in this work, straightforward logic implementation can lead to large overhead and/or degraded learning performance boost. To this end, we propose a novel data-driven design flow to optimize the digital implementation of STDP for efficient hardware realization.

The second key ingredient of the proposed SSO-LSM architecture is the integration of an online readout synapse reconfiguration scheme. Under the context of the liquid state machine, typically a large number of readout synapses with a sufficient resolution is needed to achieve good learning performance. As such, the readout synapses contribute significantly to the overall energy dissipation of an LSM processor. The proposed reconfiguration scheme sparsifies readout synapses during runtime by monitoring the variances of firing activities of reservoir neurons. This leads to noticeable energy reduction without any significant degradation of performance.

Using the spoken English letters adopted from the TI46 speech corpus as a benchmark, we demonstrate that the SSO-LSM architecture boosts the average learning performance rather significantly by 2.0% while reducing energy dissipation by 25% compared to a baseline LSM design with little extra hardware overhead on a Xilinx Virtex-6 FPGA.

The rest of this section is organized as follows. Section 3.2.1 provides a brief description of the background and motivations of this work. Section 3.2.2 presents our proposed design flow for simple hardware STDP and readout sparsification method. In Section 3.2.5, the proposed SSO-LSM architecture is introduced. The experimental results are reported in Section 3.2.6. Finally, Section 3.2.7 summarizes this work.

3.2.1 Motivation for STDP-based Reservoir Tuning in Hardware

STDP is a local unsupervised Hebbian learning mechanism realizing synaptic plasticity based on the respective firing timing orders of the presynaptic and postsynaptic neuron [23, 24]. The synapse w_{ij} from neuron j to neuron i is potentiated if a causal order (i.e., the presynaptic neuron fires before the postsynaptic neuron) is observed, or depressed otherwise. The weight update depends on the temporal difference $\Delta t = t_{post} - t_{pre}$ between each pair of pre and post-synaptic spikes:

$$\begin{aligned} \Delta w^+ &= A_+(w) \cdot e^{-\frac{|\Delta t|}{\tau_+}} \quad \textit{if}\; \Delta t > 0 \\ \Delta w^- &= A_-(w) \cdot e^{-\frac{|\Delta t|}{\tau_-}} \quad \textit{if}\; \Delta t < 0, \end{aligned} \tag{3.7}$$

where Δw^+ and Δw^- are the weight modification induced by long-term potentiation (LTP) and long-term depression (LTD), and $A_\pm(w)$ determines the strength of LTP/LTD, respectively. A typical STDP curve is plotted in Fig. 3.1(a).

In addition to the potential boost of performance, tuning a reservoir using STDP can lead to a refined sparser structure of the reservoir, which we explore to lower the energy dissipation of the hardware processor. To see this, we induce the self-organizing behavior in a reservoir by applying the standard STDP and show the equilibrium synaptic weight distribution of the reservoir in Fig. 3.7(b). Here, as a common practice, excitatory synapses in the reservoir are assumed to be plastic and tuned with STDP while inhibitory synapses are fixed. Clearly, the resulting bimodal weight distribution suggests that a significant number of zero-valued or low-valued synapses may be powered off to save power without dramatically impacting learning performance.

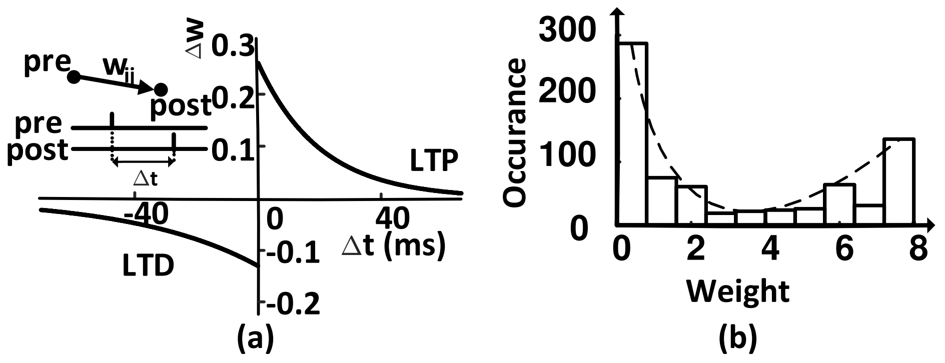

Figure 3.7: (a) A typical STDP characteristics, and (b) an equilibrium synaptic weight distribution induced by the STDP rule in a reservoir. Reprinted with permission from Yingyezhe Jin, Yu Liu and Peng Li ©2016 IEEE.

Now, here comes the important question of how to realize a given STDP in digital logic to minimize hardware overhead and maximize learning performance boost. A straightforward realization of STDP in high resolution can closely reflect the desired continuous STDP characteristics in hardware, and hence produce good performance. However, doing so can lead to an inhibitory cost as STDP shall be applied to all neurons in a reservoir. Furthermore, the realization of (3.7) with a high bit resolution produces diminishingly small weight updates as the temporal difference Δt increases, and the number of such updates can be very large. The combined effects of the two result in many synaptic events with small weight update values, jeopardizing the runtime energy efficiency of the neural processor. On the other hand, simply reducing hardware overhead by using a low resolution can lead to an immediate performance hit.

3.2.2 STDP Reservoir Tuning in SSO-LSM

Achieving good learning performance using STDP realized in extremely low bit resolutions presents interesting challenges. To eliminate small-valued weight updates so as to enhance processing efficiency, we first determine a STDP activation time window by setting an upper limit Δt_{limit} for the spike timing difference Δt. No synaptic update event is triggered when $\Delta t > \Delta t_{limit}$. To balance between potentiation and depression, we force the areas under the LTD and LTP portions of the STDP curve to be identical. With this, we would still need to retain the good performance of a well-designed continuous STDP rule by properly discretizing both synaptic weight and the continuous STDP.

Naive Digital STDP Realization

Intuitively, one may implement a B-bit STDP by uniformly discretizing weight value in a targeted range into 2^B levels: $\{w_1^d, w_2^d, \cdots, w_{2^B}^d\}$, and similarly discretize the weight change Δw^d of the continuous STDP rule within the activation window. A synaptic update with a spike timing difference Δt triggers the following process to determine the discretized new weight w_{new}^d: add the discretized weight change Δw^d from the discretized STDP curve to the current (old) discretized weight value w_{old}^d, and round the sum to produce w_{new}^d:

$$\begin{aligned} w_{new}^d &= f_{round}(w_{old}^d + \Delta w^d(\Delta t)) \\ &= f_{round}(f_{round}(w_{old}^c) + \Delta w^d(\Delta t)), \end{aligned} \tag{3.8}$$

where $f_{round}(\cdot)$ rounds its argument to its nearest discretized weight level, and w_{old}^c (w_{new}^c) represents the current (new) continuous-valued weight if synaptic weights and STDP were implemented in real numbers.

A careful examination of the above process reveals *two key limitations*. First, using an adder to perform each add operation (in parallel) can introduce large hardware overhead. Second, the computation of w_{new}^d in (3.8) suffers from two types of rounding error: discretizaion of the continuous weight w^c and quantization of the continuous weight update Δw^c. Through a specific example, Fig. 3.8(a) illustrates that the above process can produce a very large overall quantization error of 1.7 with respect to w_{new}^c under low bit resolutions.

3.2.3 Data-Driven STDP Implementation

To address the above limitations, our key idea is to discretize synaptic weight and the STDP scheme to match realistic synaptic data so as to minimize the aggregated discretization error over a large set of STDP updates. The offline data-driven design flow of Fig. 3.9 finds the optimal

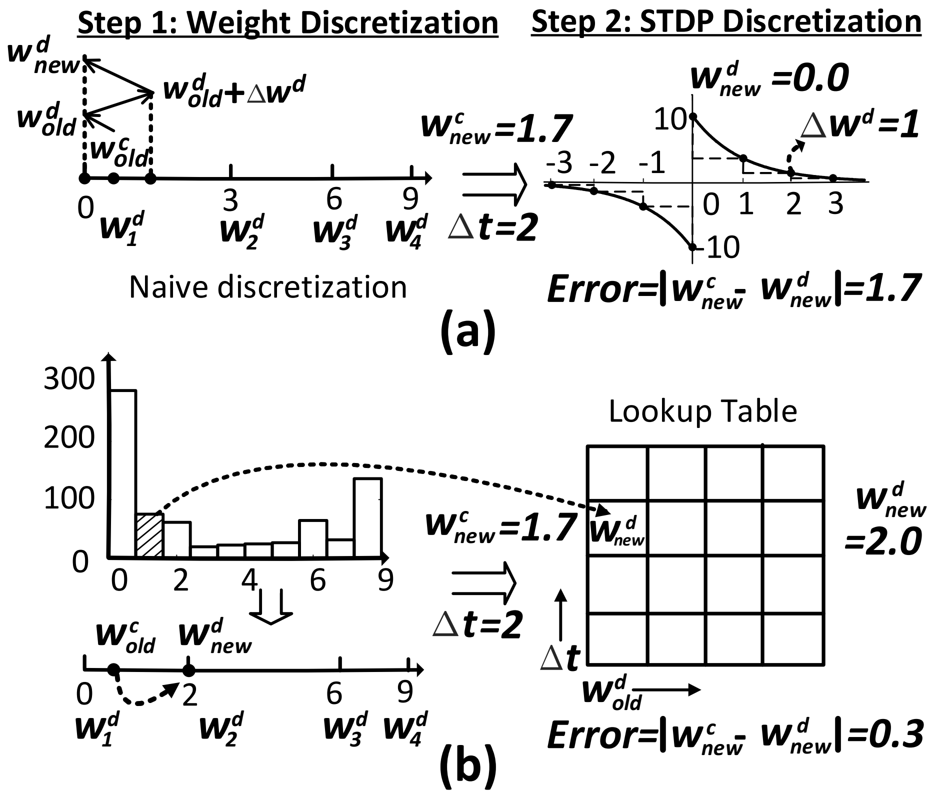

Figure 3.8: Comparison between the naive (a) and the proposed data-driven STDP realization (b). Reprinted with permission from Yingyezhe Jin, Yu Liu and Peng Li ©2016 IEEE.

discretization schemes for synaptic weights, and finally the STDP rule.

Step 0 simulates the reservoir using typical inputs and profiles continuous STDP events by collecting a set of N combinations of three values: $\{\Delta t_k, w^c_{old,k}, w^c_{new,k}\}$, $k = [1...N]$.

Step 1 finds 2^B optimal non-uniform integer discretized weight levels $\overrightarrow{W^d_i}$ for the data by solving an optimization problem:

$$\begin{aligned} \underset{w_i^d, 1\leq i \leq 2^B}{\text{minimize}} \quad & \sum_k f_{rnd\_err}(w^c_k, \overrightarrow{W^d_i}) \\ \text{subject to} \quad & \overrightarrow{W^d} = [w^d_1, w^d_2, \cdots, w^d_{2^B}]^T, \\ & w^d_i \in [w_{min}, w_{max}], \forall i \in [1, \cdots, 2^B], \end{aligned} \tag{3.9}$$

where $f_{rnd\_err}(w_k^c, \overrightarrow{W_i^d})$ is the squared rounding error for the k-th collected continuous weight w_k^c:

$$f_{rnd\_err}(w_k^c, \overrightarrow{W^d}) = \min\{(w_k^c - w_j^d)^2 : j \in [1, \cdots, 2^B]\}. \tag{3.10}$$

This optimization problem presents no challenge and can be even solved by exhaustive search since the design space is bounded by a very low bit resolution of B. Weight data at the equilibrium can be used in the above optimization problem to best approximate the converged weight distribution.

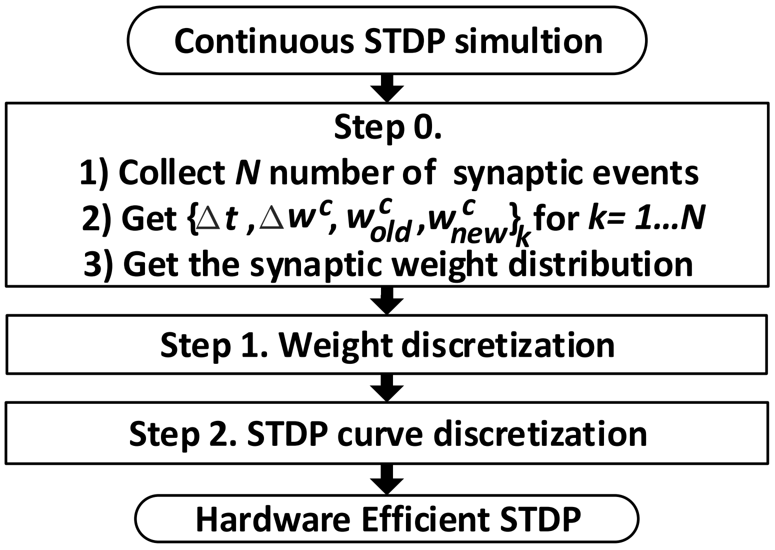
Continuous STDP simultion
Step 0.
1) Collect N number of synaptic events
2) Get {Δt ,Δw^c, w^c_old, w^c_new}_k for k= 1…N
3) Get the synaptic weight distribution
Step 1. Weight discretization
Step 2. STDP curve discretization
Hardware Efficient STDP

Figure 3.9: The proposed data-driven offline STDP design flow. Reprinted with permission from Yingyezhe Jin, Yu Liu and Peng Li ©2016 IEEE.

Step 2 finds the optimal discretized STDP rule based on the optimal weight discretization of Step 1. For each k-th synaptic update, our key idea is to map from the combination of discretized $\{\Delta t_k, w_{old,k}^d\}$ directly to the discretized new weight value $w_{new,k}^d$ using an optimized small lookup table (LUT). This LUT is indexed by Δt_k and $w_{old,k}^c$, where the spike timing difference Δt_k is already discretized by the chosen network emulation time step. In comparison to the naive approach, the proposed technique reduces hardware overhead by eliminating the add operation, and merges the multiple processing steps of the naive approach into a single LUT operation, thereby avoiding incurring multiple rounding errors. Fig. 3.8(b) illustrates the proposed STDP update process using the same example of Fig. 3.8(a), and shows that our approach produces a much smaller overall discretization error of 0.3. Each LUT entry is determined by minimizing an discretization error metric defined with respect to the continuous STDP data collected in Step 0.

Specially, the entry L_{ij} at the i-th row and j-th column of the LUT sets the mapped new weight value for all continuous STDP updates that fall into the corresponding bin Bin_{ij} defined by: $\Delta t \in [\Delta t^{ij,low}, \Delta t^{ij,high}]$ and $w^c_{old} \in [w^{d,low}_{old}, w^{d,high}_{old}]$, where $\Delta t^{ij,high/low}$ and $w^{d,high/low}_{old}$ are the boundaries of Bin_{ij}. We find the optimal value for L_{ij} by solving an optimization problem:

$$\begin{aligned} \underset{\boldsymbol{L_{ij}}}{\text{minimize}} \quad & \sum_k (w^c_{new,k} - L_{ij})^2 \\ \text{subject to} \quad & (\Delta t_k, w^c_{old,k}) \in Bin_{ij}, \\ & L_{ij} \in \{w^d_1, w^d_2, \cdots, w^d_{2^B}\}. \end{aligned} \tag{3.11}$$

Essentially, the above optimal solution minimizes the summed squared error for all continuous STDP updates that fall into the corresponding LUT entry. Again, this optimization problem can be trivially solved offline due to the small search space.

3.2.4 Online Sparsification of Readout Synapses

The connections between the reservoir and the readout are very dense and with high resolution to ensure good learning performance. Instead of randomly pruning the readout connectivity which might result in significant performance penalty, we propose a novel online readout reconfiguration approach based on variances of firing activities with little impact on performance. The key observation is that the variance of the firing activity of each reservoir neuron across different input samples correlates with its contribution to the distinguishability of different patterns. Hence, readout synapses projected from low-variance reservoir neurons can be powered off to save energy. Implementing this idea in hardware entails efficient monitoring of variance of firing activity, which we describe very briefly. Instead of computing the true variance, which is costly in hardware, we sum up the absolute differences in firing counts of consecutive input samples:

$$\delta = \sum_{i=2}^{n} |c_i - c_{i-1}|, \tag{3.12}$$

where $\{c_i\}_{i=1\cdots n}$ is the firing count sequence for n input samples of a neuron.

3.2.5 The SSO-LSM Architecture

In the proposed SSO-LSM architecture, the reservoir and readout layer are realized by a Reservoir Unit (RU) and Training Unit (TU) respectively, as depicted in Fig 3.10 for FPGA implementation. The digital neurons in RU are called Liquid Elements (LEs) and the neurons inside TU are Output Elements (OEs). The external input spikes connect to targeted liquid neurons through a crossbar connection interface. All LEs receive and process the input spikes in parallel. The spikes generated from LEs are buffered in a 135-bit register called R_{Spike}. Then, the spike outputs are sent back to LEs following certain connectivity pattern, which is realized by another crossbar interface. At the same time, the spikes from R_{Spike} also propagate to TU as the liquid response, where all OEs collect and process the spikes from TU simultaneously. Both LE and OE adopt the leaky integrated-and-fire(LIF) model while they leverage different learning schemes and architecture. We adopt the architecture for OE and the synaptic response and integration unit in LE from [17]. The weights of the plastic synapses associated with each OE are stored in its private block RAM (BRAM).

To implement the online readout reconfiguration scheme mentioned in Section 3.2.4, we use three firing counters (i.e., FC 1, FC 2 and FC 3) in the LE as illustrated in Fig. 3.11(a). FC 1 and FC 2 store the firing counts for the reservoir neuron regarding to the previous and current input sample, respectively. When the input spike trains come to the end, the absolute difference of FC 1 and FC 2 is added up to FC 3, and FC 2 is set to be the value of FC1 and FC1 is then reset. After all samples have been fed in, the value of FC 3 becomes the targeted approximated variance of the firing activities across all inputs. If the variance is less than a certain threshold, the connections from this neuron to the readout layer are powered off.

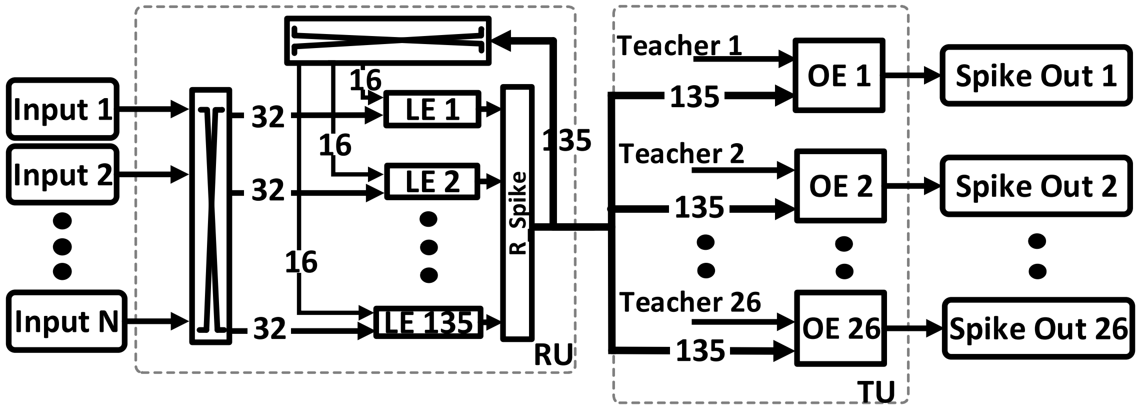

Figure 3.10: An exemplary implementation of the SSO-LSM architecture with 135 digit liquid neurons (LEs) and 26 output neurons (OEs). Each LE connects to up to 32 external inputs and up to 16 internal LEs. Reprinted with permission from Yingyezhe Jin, Yu Liu and Peng Li ©2016 IEEE.

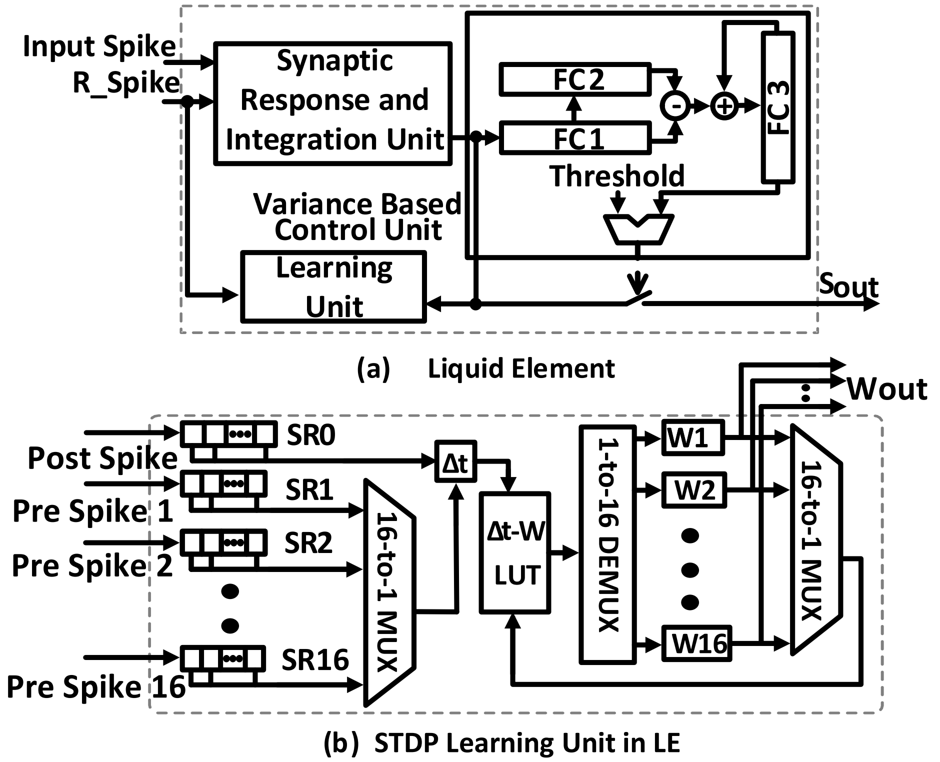

Figure 3.11: (a) The architecture of the LE, and (b) the architecture of STDP learning unit in LE. Reprinted with permission from Yingyezhe Jin, Yu Liu and Peng Li ©2016 IEEE.

The learning unit in LE implementing the proposed hardware STDP mechanism is shown in Fig. 3.11(b). We adopt the logic-level STDP implementation proposed in [89], and further simplify it. Each reservoir synapse connected to the reservoir neuron has its associated weight register. The computation of Δt for each pre and post synaptic spike pair and the update of the associated synaptic weight are executed in serial. For each post-synaptic neuron, shift registers SR1 to SR16 are used to keep track of the firing events from its 16 pre-synaptic neurons while SR0 is for recording the firing event of this post-synaptic neuron itself. Once there is a firing event, the fired spike is injected into registers from the MSB and shifted one bit to the right at every time step of the system. The width of pre-synaptic shift registers (i.e., SR1 to SR16) and post-synaptic shift registers (i.e., SR0) represents the time window Δt_{limit} for the LTP and LTD curve, respectively, as mentioned in Section 3.2.2. By examining the relative position of the spikes in the shift registers, we can compute the temporal difference Δt between pre and post-synaptic spike pairs. For example, when there is an '1' appears at the MSB of SR0, meaning the postsynaptic neuron just fires, we check its presynaptic shift registers to see if any presynaptic spike event can be paired with it. If there is one presynaptic spike to be paired, then Δt is the location of this presynaptic spike in the shift register. Similarly, we can capture the post-before-pre firing order if there is a spike in the MSB of the presynaptic shift register. With Δt, the new synaptic weight is given by the lookup table implementing the proposed hardware STDP rule.

3.2.6 Experimental Results

Several digital LSMs are simulated in software to fully assess the performance boost and sparsity resulting from the proposed STDP rule and readout synapse sparsification over a large design space in which various reservoir sizes and readout synaptic resolutions are considered. We report the hardware overhead and energy consumption of the SSO-LSM architecture with 135 reservoir neurons implemented on a Xilinx Virtex-6 FPGA. The software simulation setup and the supervised readout synapse learning rule are adopted from [10]. The adopted application benchmark is a subset of the TI46 speech corpus [80], containing 10 utterances of each English letter from "A" to "Z" recorded from a single speaker. There are 260 samples in this benchmark. The time domain speech signals are preprocessed by Lyon's passive ear model [86], and encoded into 78 spike trains using the BSA algorithm [87].

Performance Boost

By manually optimizing several key STDP rule parameters over a large design space, we obtain an optimized continuous STDP rule with an average performance boost of 1.9%. Following the proposed offline STDP design flow of Section 3.2.3, we use the continuous STDP rule to collect a set of profiled continuous STDP update data. To realize a low-cost hardware-based STDP, we discretize the reservoir synaptic weights using 2-bit resolution and only consider weight updates with $|\Delta t| \leq 3$. The application of our offline optimization flow leads to the optimal weight discretization levels and STDP lookup table shown in Table 3.5.

Table 3.5: Optimized weight discretization and STDP lookup table. Reprinted with permission from Yingyezhe Jin, Yu Liu and Peng Li ©2016 IEEE.

	$w_1^d = 0$	$w_2^d = 2$	$w_3^d = 6$	$w_4^d = 8$
$\Delta t = -3$	w_1^d	w_2^d	w_3^d	w_4^d
$\Delta t = -2$	w_1^d	w_1^d	w_2^d	w_3^d
$\Delta t = -1$	w_1^d	w_1^d	w_1^d	w_2^d
$\Delta t = 0$	w_1^d	w_2^d	w_3^d	w_4^d
$\Delta t = 1$	w_3^d	w_4^d	w_4^d	w_4^d
$\Delta t = 2$	w_2^d	w_3^d	w_4^d	w_4^d
$\Delta t = 3$	w_1^d	w_2^d	w_3^d	w_4^d

The recognition performances of several digital LSMs without STDP reservoir tuning over a large design space are reported in Table 3.6. In comparison, we show the performance boosts resulted from the proposed hardware STDP rule in Table 3.7. We also compare the averaged performance achieved for the LSM designs in Table 3.4 by the proposed hardware rule, the naive approach of Section 3.2.2, and six LUT based rules (C1 to C6) with randomly set LUT entry values in Fig. 3.12. The proposed STDP produces an average performance boost of 2%, superior than all other STDP rules.

Table 3.6: Performance of the baseline LSMs without STDP. Reprinted with permission from Yingyezhe Jin, Yu Liu and Peng Li ©2016 IEEE.

	Resolution of readout synapses					
Size	10	9	8	7	6	5
135	90%	90%	90.4%	88.8%	89.2%	89.2%
90	87.7%	86.5%	85.8%	86.9%	84.2%	84.2%
72	84.2%	83.1%	83.5%	82.7%	82.7%	80.4%
63	86.9%	86.2%	88.1%	85.9%	85.8%	82.7%
45	80.8%	78.8%	79.2%	80.8%	78.5%	72.7%

Table 3.7: Performance boost of the LSMs with the hardware STDP. Reprinted with permission from Yingyezhe Jin, Yu Liu and Peng Li ©2016 IEEE.

	Resolution of readout synapses					
Size	10	9	8	7	6	5
135	1.9%	1.9%	2.7%	3.9%	2.7%	1.9%
90	-0.8%	0.8%	2.3%	1.2%	2.3%	1.5%
72	1.9%	4.2%	4.2%	4.2%	3.1%	2.3%
63	1.5%	2.7%	0.4%	1.1%	0.4%	-1.5%
45	1.5%	3.1%	2.3%	0.8%	3.1%	1.5%

Sparsity of the SSO-LSM Architecture

First, we examine the sparsity of the reservoir due to the proposed hardware STDP rule. We report the percentages of reservoir synapses with a zero-weight value after applying the proposed STDP rule for different reservoir sizes in Table 3.8. As shown in Table 3.8, the proposed hardware rule can zero out up to 29.2% of the reservoir synapses, reaffirming its effectiveness in reducing the complexity of the LSM for better efficiency.

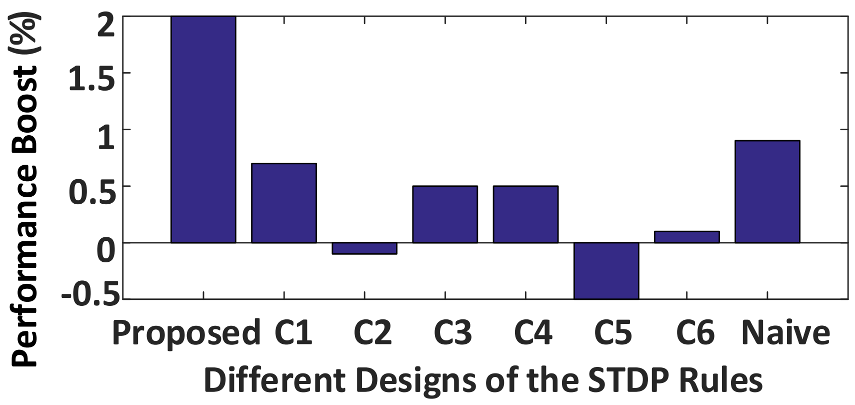

Figure 3.12: The average performance boosts over the considered design space achieved by different STDP rules. Reprinted with permission from Yingyezhe Jin, Yu Liu and Peng Li ©2016 IEEE.

Table 3.8: The reservoir synapse reductions of the proposed hardware STDP rule under different reservoir sizes. Reprinted with permission from Yingyezhe Jin, Yu Liu and Peng Li ©2016 IEEE.

Size	**135**	**90**	**72**	**63**	**45**
Reduction	27.2%	28.9%	28.7%	29.2%	27.5%

With the proposed hardware STDP applied, we measure the average recognition performance over the same design space considered before under different readout synapse sparsification levels in Fig. 3.13. As seen in Fig. 3.13, up to 20% of readout synapses projected from reservoir neurons with low firing variance can be powered off without any significant degradation of performance. These results suggest that a refined and efficient readout layer can be obtained with the proposed online readout reconfiguration approach.

Hardware Cost and Energy Dissipation of the SSO-LSM Architecture

We implement our proposed SSO-LSM architecture on a Xilinx Virtex-6 FPGA and measure the energy consumption of each building block using the Xilinx Power Analyzer. Table 3.9 shows the hardware cost and energy consumption of the baseline LSM architecture without the proposed

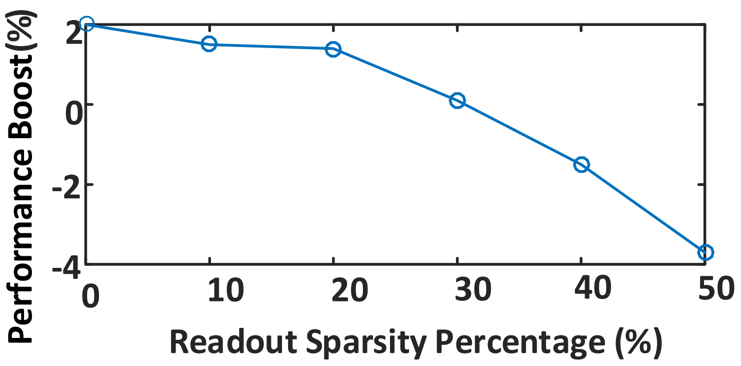

Figure 3.13: The average performance boosts achieved by the proposed hardware STDP rule under different readout sparsification levels. Reprinted with permission from Yingyezhe Jin, Yu Liu and Peng Li ©2016 IEEE.

hardware STDP tuning and readout synapse sparsification, and compares the baseline with the proposed SSO-LSM architecture. The readout synapses sparsification level is set to 20% for the SSO-LSM architecture, which boosts the recognition performance noticeably by 1.4% over the baseline. Furthermore, the proposed SSO-LSM architecture reduces the energy consumption of the RU and TU by 36% and 8%, respectively. It also significantly reduces the overall energy consumption by 25% with 4% additional logic overhead (LUTs) and 22% additional registers compared to the baseline LSM, as shown in the last column of Table 3.9.

3.2.7 Summary

In this section, we have proposed a novel design approach for efficient hardware-based STDP reservoir tuning and an online readout reconfiguration scheme to enable the sparse and self-organizing LSM (SSO-LSM) architecture for liquid state machine based neural processors. Facilitated by the proposed offline data-driven STDP optimization flow, the SSO-LSM architecture achieves efficient reservoir tuning at low synaptic weight resolutions and delivers good learning performance. Furthermore, sparse readout layers can be obtained by the presented firing variance based reconfiguration approach with little performance degradation. The SSO-LSM architecture is shown to outperform the baseline LSM architecture in terms of learning performance and energy efficiency

Table 3.9: Comparison between the baseline LSM and the proposed SSO-LSM architecture in terms of hardware cost and energy consumption. The percentages of hardware resource usages with respect to the total available on-chip FPGA resources are also reported. Reprinted with permission from Yingyezhe Jin, Yu Liu and Peng Li ©2016 IEEE.

		Baseline LSM	**SSO-LSM**	**Normalized Cost & Energy**
Cost (Usage %)	Slice FFs	13,336 (4%)	16,338 (5%)	1.22
	Slice LUTs	52,686 (34%)	55,013 (36%)	1.04
	BRAMs	26*18KB (3.1%)	26*18KB (3.1%)	1.00
Energy (J) @50MHz	RU	23.9	15.35	0.64
	TU	14.42	13.27	0.92
	Overall	38.32	28.62	0.75

on an FPGA platform using speech recognition as a benchmark.

4. READOUT LEARNING AND SPARSIFICATION OF LIQUID STATE MACHINES[*]

The recent work of [10] introduces a biologically-inspired spike-dependent readout training approach with the advantages being local and amenable to VLSI implementation. However, the key limitation of this approach is that good performance is typically guaranteed only with full connectivity between the reservoir and readout, which contributes significantly to the overall network complexity and is also costly from a hardware point of view.

The unsupervised learning of STDP is able to locally tune SNNs and sparsify the connectivity based on the temporal correlation. Towards supervised algorithms, several works have explored the idea of combining supervision and STDP [11, 50, 51] for precisely timed spike reproduction and decision making, but no successful results are demonstrated for real-world tasks.

In this chapter, we present the unifying calcium-modulated supervised STDP approach to achieve the two competing objectives of improved learning performance and readout sparsity of LSMs. The presented approach employs a two-step methodology: training for sparsification followed by training for learning performance. Both of the steps belong to the same biologically inspired supervised STDP framework. In the unifying framework, the calcium-modulated learning algorithm based on supervised STDP (*CaL-*S^2*TDP*) is proposed to achieve improved performance for readout training. The calcium-modulated sparsification algorithm based on supervised STDP (*CaS-*S^2*TDP*), is for readout synapse sparsification where a high-degree of sparsity is produced without significant degradation of learning performance.

Using the spoken English letters from the TI46 Speech Corpus [80] as a real-world speech recognition benchmark, we demonstrate that *CaL-*S^2*TDP* significantly improves the recognition rate by up to 25% over a reference supervised STDP rule. Compared to the competitive non-STDP spike-based learning rule in [10], *CaL-*S^2*TDP* improves the recognition rate by up to 2.7%. The seamless integration of the two proposed algorithms can prune out up to 30% of readout synapses

[*]©2017 IEEE. Reprinted, with permission, from Yingyezhe Jin and Peng Li, "Calcium-modulated supervised spike-timing-dependent plasticity for readout training and sparsification of the liquid state machine", *Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, May 2017.

without causing significant performance degradation.

4.1 Existing STDP Rules and their limitations

The standard STDP is a local unsupervised Hebbian learning mechanism realizing synaptic plasticity based on the relative firing timing orders of the presynaptic and postsynaptic neurons [23, 24]. The long-term potentiation (LTP) of the synapse w_{ij} occurs if the presynaptic neuron j fires before the postsynaptic neuron i. A presynaptic spike that follows postsynaptic spike produces long-term depression (LTD) of the synapse. The amount of synaptic modification depends on the temporal difference $\Delta t = t_{\text{post}} - t_{\text{pre}}$ between each pair of pre- and postsynaptic spikes:

$$\begin{aligned} \Delta w^{+} &= A_{+}(w) \cdot e^{-\frac{|\Delta t|}{\tau_{+}}} \quad \text{if } \Delta t > 0 \\ \Delta w^{-} &= A_{-}(w) \cdot e^{-\frac{|\Delta t|}{\tau_{-}}} \quad \text{if } \Delta t < 0, \end{aligned} \tag{4.1}$$

where Δw^{+} and Δw^{-} represent the weight change induced by LTP and LTD, $\tau_{\pm}$ are the time constants, and $A_{\pm}(w)$ determine the strength of LTP/LTD, respectively. A typical STDP characteristics is plotted in Fig. 4.1.

While the standard unsupervised STDP can be relatively straightforwardly applied to spiking neural networks, the lack of supervision disqualifies it as a choice for readout training. Although supervised STDP mechanisms such as [11, 50, 51] have been explored, so far no success has been demonstrated towards applying them to real-life applications. As discussed in Section 1.3, these approaches may also suffer from synaptic weight saturation and inefficiency for hardware realization.

4.2 Proposed Deterministic Supervised STDP without Calcium Modulation

Working towards deriving the proposed *CaL-S^{2}TDP* algorithm, we first present a simpler STDP algorithm, dubbed *D-S^{2}TDP*. *D-S^{2}TDP* performs deterministic weight updates, has all essential components of *CaL-S^{2}TDP*, but lacks probabilistic weight updates and calcium modulation of *CaL-S^{2}TDP*. *D-S^{2}TDP* also serves as a reference of comparison for *CaL-S^{2}TDP* in our experimen-

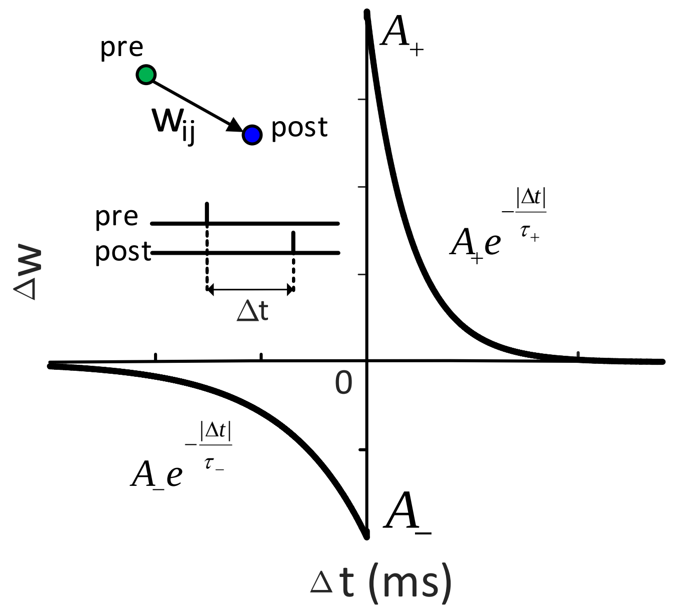

Figure 4.1: A typical STDP characteristics. Reprinted with permission from Yingyezhe Jin and Peng Li ©2017 IEEE.

tal study.

4.2.1 Mathematical Interpretation for Supervised STDP

As a common practice, let us assume that the classification decision is decoded by choosing the class label of the readout neuron with the highest firing activity (frequency) in the readout layer. Therefore, a supervised training algorithm shall: 1) maximize the firing rate of the readout neuron whose class label corresponds to the presented input sample, referred to as "desired neuron"; and 2) minimize the firing rates of all other readout neurons, referred to as "undesired neurons". We argue that both 1) and 2) can be achieved by solving the following optimization problem:

$$\begin{aligned} \underset{f_j^i}{\text{maximize}} \quad & \sum_{i=1}^{N}(f_{c(i)}^i(X_i, W) - \sum_{j \neq c(i)}^{C} f_j^i(X_i, W)) \\ \text{subject to} \quad & f_j^i \geq 0, \end{aligned} \tag{4.2}$$

where N is total number of training samples, C is the number of input classes, X_i is the i_{th} input temporal sample, $c(i)$ is the class label for X_i, f_j^i is the firing frequency of the j_{th} readout neuron under the i_{th} input, and W is the vector of all readout synaptic weights. Here, we maximize the difference in summed firing rate between the desired neuron and all undesired neurons so as to minimize the classification error over the N training samples. However, solving (4.2) in a mathematically exact manner is a formidable task.

4.2.2 Proposed *D-S²TDP* Algorithm

Instead, we take a more feasible biologically-inspired approach with respect to (4.2) as shown in Fig. 4.2 (a). The proposed *D-S²TDP* algorithm forces the desired neuron to fire at an elevated level via the positive current injected by a classification teacher (CT) signal. Each afferent synapse of the desired neuron is further mediated by a standard STDP rule. To suppress undesired neurons, we employ a novel depressive STDP rule for their afferent synapses.

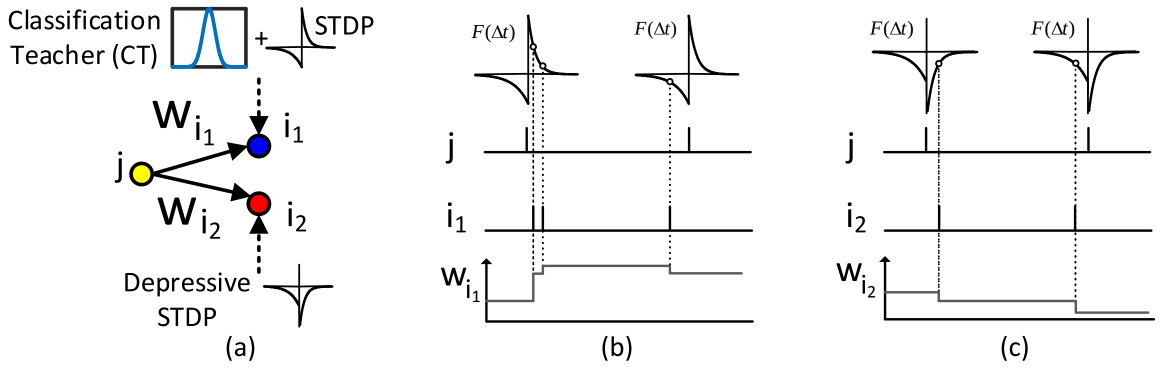

Figure 4.2: (a) Proposed *D-S²TDP* algorithm. The neuron i_1 is the desired neuron modulated by a classification teacher (CT) and the standard STDP. The neuron i_2 is an undesired one modulated by the depressive STDP for temporal "anti-learning"; (b) and (c) The pre-before-post timing order leads to LTP (LTD) for afferent synapses of the desired (undesired) neuron. The post-before-pre timing order results in depression for both neurons. Reprinted with permission from Yingyezhe Jin and Peng Li ©2017 IEEE.

To see how *D-S²TDP* serves the basic needs of this work, we recall that the standard STDP rule used for afferent synapses of the desired neuron conducts synaptic modification locally and renders the postsynaptic (desired) neuron sensitive to temporal presynaptic firing patterns. Since the causal order (i.e., pre-before-post) of spike pairs leads to synaptic potentiation, the STDP correlates the presynaptic firing events with the modulated postsynaptic firing patterns in a way such that the desired neuron is more likely to fire in presence of presynaptic firing events. As illustrated in Fig. 4.2(b), when modulated by the CT signal, the desired neuron i_1 emits two spikes after a presynaptic spike, resulting in potentiation of w_{i_1} and making itself more likely to respond to future firings of the presynaptic neuron j. Therefore, we can maximize the firing frequency of the desired neuron by potentiating the synapses that contribute to its firing. The presence of CT also makes the above process robust by initializing STDP-based LTP/LTD even with low initial presynaptic weight values.

We depress plastic synapses that may evoke firing of undesired neurons. As depicted in Fig. 4.2(c) when the undesired postsynaptic neuron i_2 fires, the pre-before-post spike pattern induces depression to w_{i_2} as determined by the novel depressive STDP rule. As such, we prevent the undesired neurons from learning from training samples of a different input class. Note also that for both desired and undesired neurons, the depression invoked by the anti-causal (i.e., post-before-pre) timing order enables competition among plastic synapses such that a sparse structure can be learned [53].

4.3 Proposed *CaL-S^{2}TDP* Training Algorithm

While *D-S^{2}TDP* possesses several key ingredients towards effective readout training, we address its limitations, i.e. poor memory retention, synaptic weight saturation and poor weight update efficiency for hardware implementation by extending it to the proposed *CaL-S^{2}TDP* algorithm.

Continuous rapid updates of synaptic weights of a limited number of states (e.g. due to a finite synaptic resolution) can result in bad memory retention. This manifests itself in such a way that the most recent experiences are represented and learned by the synapses better than the older ones [84, 85]. We adopt the probabilistic weight update scheme in [88] to slow down the learning process to better utilize network memory capacity. Furthermore, probabilistic updates reduce the committed number of weight updates, leading to improved hardware execution efficiency.

Furthermore, synaptic memory saturation needs to be addressed. Without any stop-learning mechanism, readout synapses are tuned by supervised STDP while continuously extracting temporal information out of the on-going reservoir firing activities. Excessive training can render each readout neuron unresponsive to new stimuli once most of its afferent synapses are over-potentiated or over-depressed, i.e., the synaptic weights are driven to the maximum/minimum value.

4.3.1 Postsynaptic Calcium Concentration

Ideally, we may deactivate potentiation of a synapse when its postsynaptic neuron is very active, which suggests that this synapse has been already over-potentiated. Similarly, we shall deactivate synaptic depression when the postsynaptic neuron becomes very inactive. Inspired by [47], we make use of the internal calcium concentration of a postsynaptic neuron as an indicator of its averaged firing level induced by new and old inputs over a long timescale. The calcium concentration $c(t)$ is a function of the postsynaptic neuron activity and modeled using a first-order dynamics:

$$\frac{dc(t)}{dt} = -\frac{c(t)}{\tau_c} + \sum_i \delta(t - t_i), \tag{4.3}$$

where τ_c is the time constant and the summation is over all postsynaptic spikes arriving at time t_i.

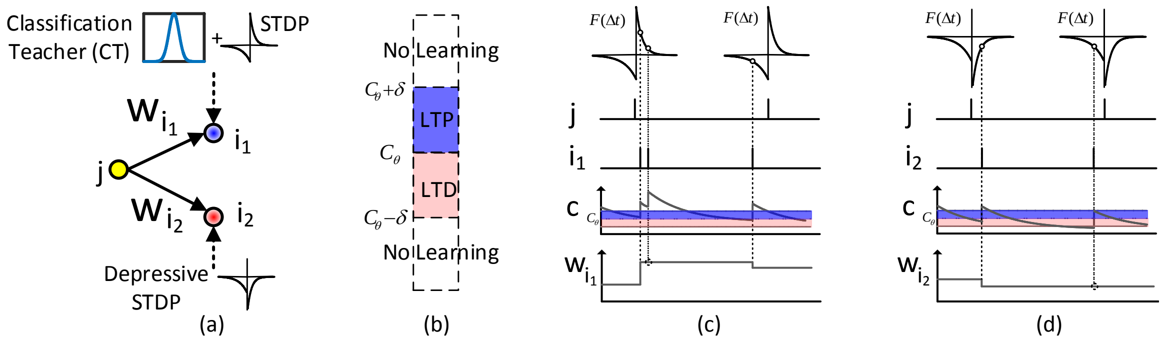

Figure 4.3: (a) Proposed *CaL-S²TDP* training algorithm with probabilistic weight updates. The desired neuron i_1 is expected to be active because of CT and the unwanted neuron i_2 is inactive due to the depressive STDP; (b) Stop-learning mechanisms; (c) and (d) The training of desired and undesired synapses. Different from Fig. 4.2(b) and (c), the LTP or LTD takes place only when the postsynaptic calcium level c falls into the specified range. Reprinted with permission from Yingyezhe Jin and Peng Li ©2017 IEEE.

4.3.2 Stop-learning for Desired Neuron

We now discuss how to implement a stop-learning mechanism for the desired neuron. First, a threshold c_θ of calcium variable is defined to distinguish active neurons from inactive ones. We then introduce a margin δ and allow potentiation when $c < c_\theta + \delta$. Analogously, depression is allowed when $c > c_\theta - \delta$. Following the principle of Hebbian learning, we further impose a lower bound of c for activating potentiation and an upper bound of c for activating depression. Combining the stop-learning mechanism and probabilistic weight updates gives the *CaL-S²TDP* algorithm:

$$\begin{aligned} w \leftarrow w + \Delta W \ \textit{w/prob.} \propto |\Delta w^+| \ & \text{ if } \Delta t > 0 \ \&\& \\ & c_\theta < c < c_\theta + \delta \\ w \leftarrow w - \Delta W \ \textit{w/prob.} \propto |\Delta w^-| \ & \text{ if } \Delta t < 0 \ \&\& \\ & c_\theta > c > c_\theta - \delta, \end{aligned} \tag{4.4}$$

where $\Delta w^+/\Delta w^-$ are the weight changes computed based on the employed the STDP rule, and determine the probabilities for committing a fixed weight update of $\pm\Delta W$ for LTP and LTD, respectively. As in Fig. 4.3(b), a synapse can be potentiated when the calcium concentration c of the desired neuron falls into $[c_\theta, c_\theta + \delta]$ and depressed when c is in $[c_\theta - \delta, c_\theta]$.

4.3.3 Stop-learning for Undesired Neurons

Since the depressive STDP is employed for the afferent synapses of undesired neurons, only the second equation in (3.4) is activated. The *CaL-S²TDP* algorithm is further illustrated in Fig. 4.3(c) and (d) where no long-term modification is induced if the calcium level is too low or too high, different from *D-S²TDP* as shown in Fig. 4.2(b) and (c).

4.4 Proposed *CaS-S²TDP* Sparsification Algorithm

The plastic readout synapses in an LSM often need to be very dense, e.g. forming a full-connectivity between the reservoir and readout, and have high resolution to guarantee good learning performance. This leads to two potential problems: over-fitting due to high model complexity, and large overhead for hardware implementation. On the other hand, randomly pruning readout connections can easily degrade performance significantly.

4.4.1 Readout Synapse Sparsification

The starting point for the proposed *CaS-S^{2}TDP* sparsification algorithm is the recognition of the fact that different from supervised classification for which neurons are instructed to learn certain firing patterns, the objective of sparsification is to allow sufficient competition among plastic readout synapses. We further recognize that synapses mediated by standard STDP characteristics compete for control of the timing of postsynaptic action potentials. As a result, some synapses to a postsynaptic neuron are strengthened while others are weakened [53]. When properly explored, the above process can lead to a bimodal weight distribution out of which many zero-valued or small-valued synapses can be pruned out. As a common practice, only excitatory plastic synapses are sparsified.

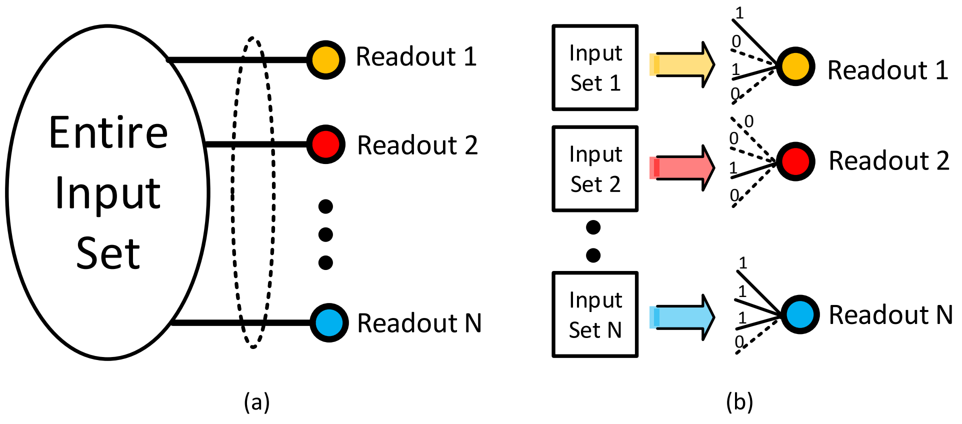

Figure 4.4: (a) The non-optimal sparsification based on the entire input set; (b) Optimized sparsification with the corresponding subset of inputs for each readout neuron. Reprinted with permission from Yingyezhe Jin and Peng Li ©2017 IEEE.

In order to make use of the above ideas for real-life multi-class classification tasks, we make additional important observations. To guarantee good performance, sparsification shall be not be performed blindly, instead, it must take into the spatio-temporal structures embedded in the training samples such that the discovered sparse patterns fit well with the data, and hence do not lead to significant performance loss. This suggests to enable a standard STDP for introducing competitions among the afferent synapses of each readout neuron over the entire training data set, as shown in Fig. 4.4(a). However, a closer examination reveals that since each readout neuron is associated with a specific class label, it is not necessary to instruct each readout neuron to learn the sparse structure of the entire input data. A more optimal approach is to constrain the finding of sparsity within of the subset of training data of the corresponding input class for each readout neuron as shown in Fig. 4.4(b). This leads to the maximum sparsity.

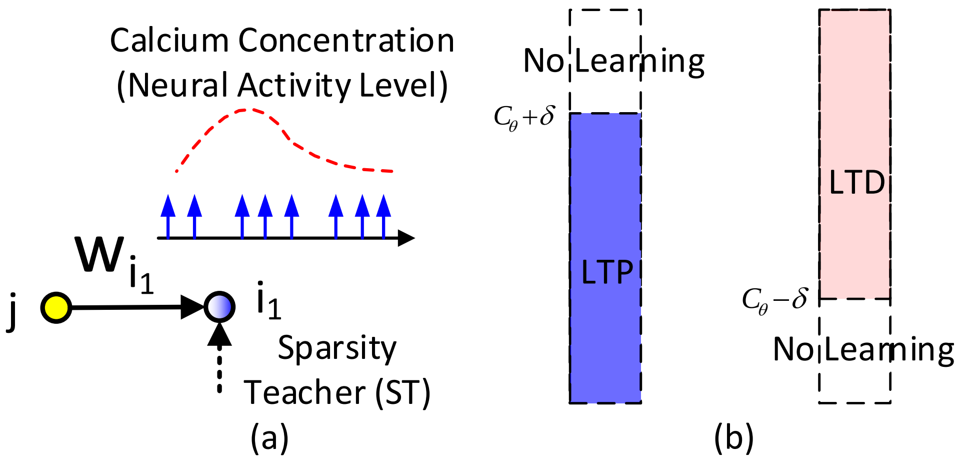

Figure 4.5: (a) The *CaS-S^{2}TDP* sparsification algorithm. The activity level of the selected readout neuron i_1 is boosted by the sparsity teacher (ST). (b) stop learning for readout synapse sparsification. Reprinted with permission from Yingyezhe Jin and Peng Li ©2017 IEEE.

Akin to the *CaL-S^{2}TDP* training algorithm, we make readout sparsification robust by introducing an external sparsity teacher (ST) (Fig. 4.5(a)), whose job is to reliably bring up the firing activity of each readout neuron to start synaptic competition modulated by the STDP. To maintain good learning performance, we also introduce a stop-learning mechanism. As shown in Fig. 4.5(b), this stop-learning mechanism is more relaxed with a wider calcium range for both LTP and LTD to avoid undesirable bias in calcium regulation and maximize sparsity. The resulting *CaS-S^{2}TDP* sparsification algorithm is summarized as:

$$\begin{aligned}
w \leftarrow w + \Delta W \; w/\,prob. \propto |\Delta w^{+}| \text{ if } \Delta t > 0 \;\&\& \\
c < c_\theta + \delta \\
w \leftarrow w - \Delta W \; w/\,prob. \propto |\Delta w^{-}| \text{ if } \Delta t < 0 \;\&\& \\
c_\theta - \delta < c.
\end{aligned} \tag{4.5}$$

4.5 Combined Sparsification and Training

The integration of the proposed sparsification and classification algorithms is summarized in Fig. 4.6. First, *CaS-S^{2}TDP* is applied to sparsify readout synapses. After removing the synapses of a zero-valued weight, the remaining synaptic weights are used as a starting point for training of the readout based on *CaL-S^{2}TDP*.

Because of the self-organizing behavior introduced by the STDP, the proposed *CaS-S^{2}TDP* sparsification algorithm learns to capture the spatio-temporal structures of the input spikes. Therefore, unlike blind synapse pruning, the proposed approach makes it possible to pass the discovered sparsity from the sparsification stage to the training phase, and produce good learning performance in the end.

4.6 Experimental Settings and Benchmark

Using the approach described in [10], two LSMs with 135 and 90 reservoir neurons are set up on a 3D grid, respectively. 80% of the reservoir neurons are excitatory while the rest of them are inhibitory. Since the adopted benchmark is spoken English letter recognition, there are 26 neurons in the readout layer, which is fully connected to the reservoir through plastic synapses. Furthermore, we adopt the discrete LIF neuronal model and the second-order synaptic model described in [10].

The parameters of the STDP algorithms described in this work are selected by exploring the

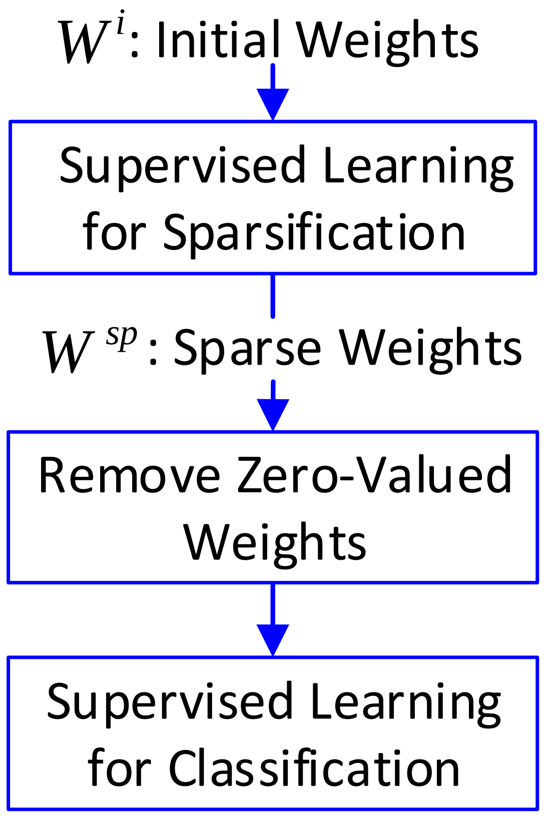
W^i: Initial Weights
Supervised Learning for Sparsification
W^{sp}: Sparse Weights
Remove Zero-Valued Weights
Supervised Learning for Classification

Figure 4.6: Two-step sparsification and classification. Reprinted with permission from Yingyezhe Jin and Peng Li ©2017 IEEE.

design space to a certain degree and we summarize the chosen ones in Table 4.1. The maximum readout synaptic weight $W_{\max}$ is set to 8.0. The initial weights of excitatory readout plastic synapses are set to 1.0 while inhibitory synaptic weights are initialized to be a random value between 0 and -$W_{\max}$. We set the bit-width of readout synaptic weights to 10 bits.

The adopted benchmark is a subset of the TI46 speech corpus [80], which contains 10 utterances of each English letter from “A” to “Z”. The speech samples were recorded from a single speaker. 260 samples are in this benchmark. The time domain speech signals are preprocessed by Lyon’s passive ear model [86], and encoded into 78 spike trains using the BSA algorithm [87]. Each input spike train generated in the preprocessing stage is sent to 32 randomly selected reservoir neurons with a fixed weight randomly chosen to be 2 or -2.

Table 4.1: Parameter settings of the proposed STDP algorithms. Reprinted with permission from Yingyezhe Jin and Peng Li ©2017 IEEE.

Parameter	**Value**
A_+	3.0
A_-	1.5
τ_+	4.0
τ_-	8.0
ΔW	0.016
c_θ	5.0
δ	2.0
τ_c	64.0

During the supervised readout sparsification phase, all speech samples are presented to the reservoir one by one while the *CaS-S*^{2}*TDP* algorithm is only applied to tune the plastic synapses of the corresponding readout neuron while other readout neurons are isolated. The process is repeated for a sufficient number of iterations until the distribution of the readout synaptic weights reaches to the steady-state. Then, we permanently remove zero-valued plastic weights and train the readout layer with the proposed *CaL-S*^{2}*TDP* algorithm for final training. A 5-fold cross validation scheme is adopted to test the recognition performance for each LSM by randomly dividing entire speech samples into 5 groups. The recognition decision is made right after each testing speech sample is presented. At this time, the class label of the readout neuron with the highest firing rate is regarded as the classification decision. The readout layer is trained for 500 iterations in order to get decent learning performance.

4.7 Experimental Results

Using the experimental setups described in Section 4.6, we compare the learning performance of *CaL-S*^{2}*TDP* to both the simpler *D-S*^{2}*TDP* algorithm of Section 4.2 and the competitive non-STDP spike-dependent algorithm of [10]. We also compare the proposed *CaS-S*^{2}*TDP* based sparsification algorithm with random and variance-based pruning [90] for both of which the algorithm of [10] is used to train the readout.

4.7.1 Classification Performance of *CaL-S^{2}TDP*

We use the adopted benchmark described in Section 4.6 to test the LSM recognition rates with three different readout learning algorithms and the results are shown in Table 4.2. The standard deviations are obtained from five experiments. Here, the proposed readout sparsification is not performed before the application of *CaL-S^{2}TDP*. It turns out that *D-S^{2}TDP* produces very low recognition rates under both reservoir sizes, indicating that *D-S^{2}TDP* is ineffective for the readout learning when the synapses have a finite resolution. In Fig. 4.7, we visualize the distribution of the readout plastic weights obtained after running the first ten training iterations. For *D-S^{2}TDP*, a considerable number of plastic readout weights quickly reach to the highest or lowest weight value, which is a direct sign of synaptic memory saturation. Fig. 4.7(a) and (c) also suggest that the poor performance of *D-S^{2}TDP* may be attributed to the occurrence of synaptic weight saturation, resulting from the lack of stop-learning mechanisms.

Table 4.2: Recognition rates of the LSMs with different readout training algorithms. Reprinted with permission from Yingyezhe Jin and Peng Li ©2017 IEEE.

Reservoir Size	**[10]**	***D-S^{2}TDP***	***CaL-S^{2}TDP***
135	$92.3 \pm 0.4\%$	$68.8 \pm 0.1\%$	**93.8** $\pm 0.5\%$
90	$89.6 \pm 0.5\%$	$67.3 \pm 0.4\%$	**92.3** $\pm 0.4\%$

The proposed *CaL-S^{2}TDP* algorithms achieves good learning performances of 93.8% with 135 reservoir neurons and 92.3% with 90 reservoir neurons, respectively. Equipped with the probabilistic update and stop learning conditions, *CaL-S^{2}TDP* significantly outperforms the simpler *D-S^{2}TDP* by 25% in terms of recognition rate for both reservoir sizes. The dominance of the proposed *CaL-S^{2}TDP* algorithm can be further explained by the weight distribution in Fig. 4.7 (b) and (d), where less plastic weights are saturated compared to the simpler algorithm.

To the best knowledge of the authors, the best reported performance on the same benchmark

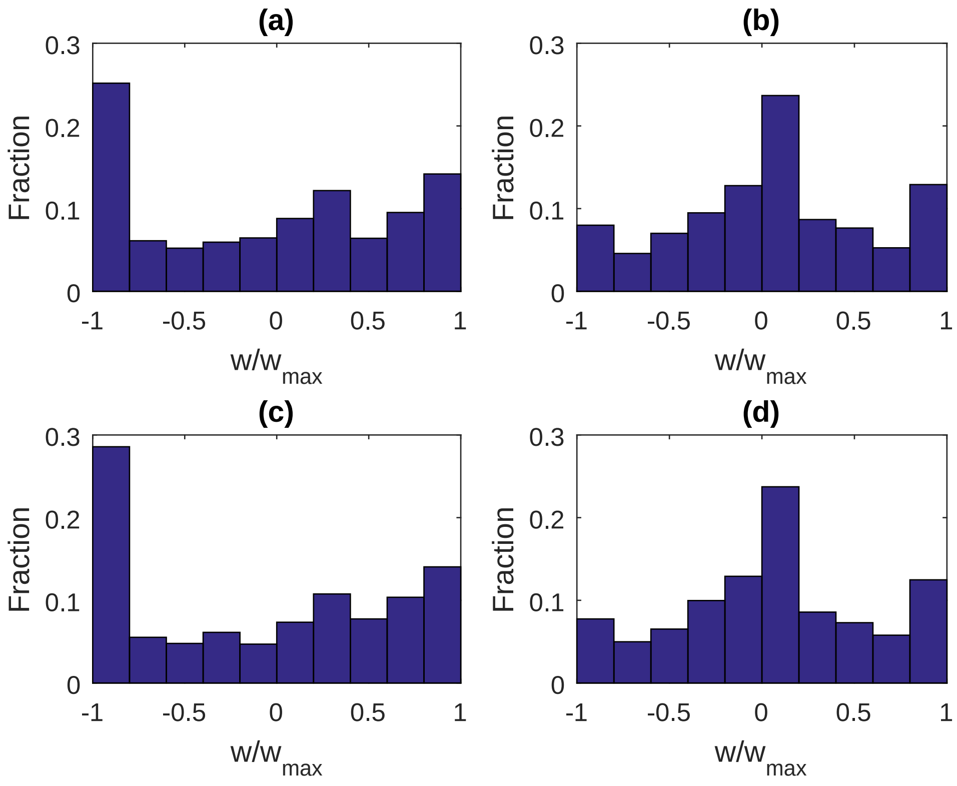

Figure 4.7: The distribution of readout synaptic weights after running first few training iterations. (a) and (c): the distribution obtained under *D-S^2TDP* with 90 and 135 neurons in the reservoir, respectively; (b) and (d): the distribution under *CaL-S^2TDP* with the same two reservoir sizes. Reprinted with permission from Yingyezhe Jin and Peng Li ©2017 IEEE.

achieved by the standard LSM with 135 reservoir neurons is 92.3% [10]. In comparison to this algorithm, the proposed supervised STDP algorithm *CaL-S^{2}TDP* outperforms it by 1.5% with the reservoir size of 135 neurons. *CaL-S^{2}TDP* also produces a good recognition rate of 92.3% when the size of the reservoir is reduced to 90 neurons, achieving a performance boost of 2.7% in this case.

4.7.2 Sparsity Obtained by *CaS-S^{2}TDP*

We examine the sparsity of the readout due to the proposed *CaS-S^{2}TDP* based sparsification scheme. After training the readout based on the proposed two-step sparsification and classification, we report the percentages of zero-valued readout synapses and the final learning performances in Table 4.3. We implement the random pruning policy and variance-based pruning of [90] and train the readout with the bio-inspired algorithm [10] for comparison. The obtained sparsity as well as learning performances are also shown in Table 4.3. Using the random pruning policy as a baseline, we plot the performance boosts achieved by the variance-based and the proposed approach in Fig. 4.8.

Table 4.3: Recognition performances with three sparsification methods. Reprinted with permission from Yingyezhe Jin and Peng Li ©2017 IEEE.

	Recognition Performance		
Sparsity	135 Reservoir Neurons		
%	Random	Variance	Proposed
10%	$90.7 \pm 0.6\%$	$91.5 \pm 0.4\%$	$92.7 \pm 0.5\%$
18%	$90.4 \pm 0.5\%$	$90.4 \pm 0.8\%$	$91.9 \pm 0.4\%$
30%	$83.8 \pm 1.0\%$	$85.0 \pm 1.3\%$	$90.7 \pm 0.4\%$
Sparsity	90 Reservoir Neurons		
%	Random	Variance	Proposed
10%	$86.9 \pm 1.0\%$	$87.7 \pm 0.7\%$	$91.5 \pm 0.4\%$
18%	$85.7 \pm 1.2\%$	$86.9 \pm 0.8\%$	$90.7 \pm 0.4\%$
30%	$89.2 \pm 0.8\%$	$89.6 \pm 0.8\%$	$91.1 \pm 0.4\%$

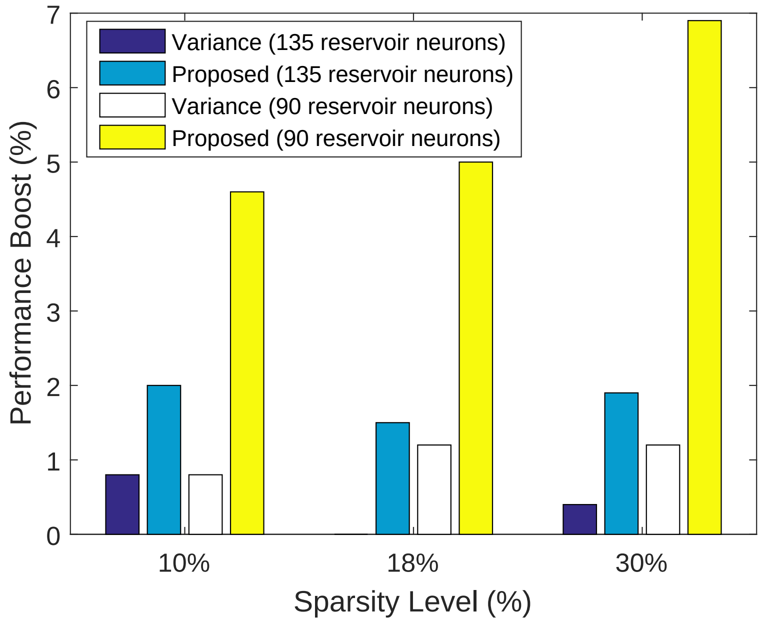

Figure 4.8: The performance boosts over the random pruning policy achieved by two different readout sparsification approaches. The proposed *CaS-S²TDP* algorithm significantly boosts performance compared to the random baseline and outperforms the variance-based approach. Reprinted with permission from Yingyezhe Jin and Peng Li ©2017 IEEE.

As shown in Table 4.3, randomly removing the readout synapses can lead to apparent performance degradation. The variance-based policy performs lightly better than the random baseline but the improvement is not significant. In comparison to the above two approaches, the proposed *CaS-S²TDP* algorithm delivers a decent learning performance under all considered levels of sparsity for different reservoir sizes as shown in Table 4.3. Importantly, the proposed approach substantially improves the effectiveness of readout sparsification compared to the random baseline. As shown in Fig. 4.8, the STDP-based approach is superior than the variance-based rule. The proposed approach can boost the random baseline performance up to 6.9% whereas the maximum boost for variance-based approach is only 1.2%.

4.8 Summary

In this chapter, we have proposed a novel calcium-modulated supervised STDP approach for both classification and sparsification, targeting efficient readout training in the context of the liquid state machine. Via a classification teacher signal, the proposed depressive STDP and probabilistic weight updates, *CaL-S^{2}TDP* robustly delivers good learning performance with finite weight resolutions. *CaL-S^{2}TDP* addresses the issue of synaptic memory saturation by imposing an stop learning condition modulated by the postsynaptic calcium concentration. Sparse readout layers can be obtained by the presented *CaS-S^{2}TDP* readout sparsification approach with little performance gradation. Using speech recognition as a realistic benchmark, we have shown that *CaL-S^{2}TDP* outperforms all other studied bio-plausible supervised training algorithms and *CaS-S^{2}TDP* based readout sparsification mechanism is superior over all other investigated approaches.

5. PERFORMANCE AND ROBUSTNESS OF DIGITAL LIQUID STATE MACHINES[1]

This chapter presents a systematic examination of performance and robustness issues of liquid state machines (LSMs), focusing specifically on speech recognition and the targeted digital VLSI implementation. In this chapter, we perform a systematic design space exploration of the LSMs proposed in [10] and show that it is possible to attain good recognition performance while noticeably reducing design complexity. More specifically, we show that recognition performance can be traded off favorably for a potentially significant reduction in reservoir size, synaptic weight and membrane voltage resolutions. It shall be noted that these three key network design parameters have a significant impact on the silicon area and power overhead of the VLSI implementation. To shed a deeper light on how these design parameters influence the internal dynamics of the network and finally recognition performance, we use several theoretical measures to characterize the computational power of the LSM as a function of the design parameters. We correlate these theoretical measures with the corresponding real-life speech recognition performance by using the widely adopted TI46 speech corpus [91] as a benchmark. Finally, to evaluate the robustness of the hardware-based LSM, a key design concern for VLSI implementation in modern CMOS technologies, we model various manufacturing and noise induced failure and error mechanisms and show the presented LSMs are in general tolerant to failures and errors.

5.1 Background

5.1.1 Speech Recognition Using the Liquid State Machine

The LSM based speech recognition can be constructed as depicted in Fig. 5.1 [15, 10]. Speech signals are first preprocessed by the Lyon passive ear model [92] then encoded into spike trains by the BSA algorithm [15, 93], and fed into a group of randomly selected neurons in the reservoir.

Input signals are processed in two steps. The first step takes place in the reservoir, where an

[1]©2016 Elsevier B.V. Reprinted, with permission, from Yingyezhe Jin and Peng Li, "Performance and robustness of bio-Inspired digital Liquid State machines: A Case study of speech recognition", *Neurocomputing*, Volume: 226, Feb. 2017, Pages 145 - 160.

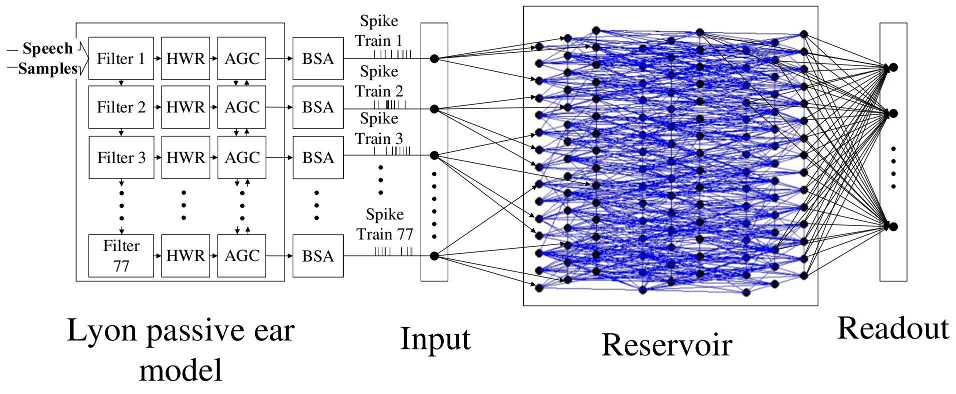

Figure 5.1: The LSM-based speech recognition system. 77 channels of spike trains preprocessed by the Lyon passive ear model and BSA algorithm are used as input to the reservoir. Then the reservoir projects the input spikes by a nonlinear transformation to the readout for further processing. Finally, the readout is trained to classify different input signals by their temporal responses in the reservoir. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

incoming spike train $u(t)$ gets mixed and mapped to the responses of the reservoir, represented by a higher dimensional transient state, rendering complex patterns more likely to be separable [94]. In the second step, the responses of the reservoir are projected to the readout through plastic synapses. For each readout neuron at time t, the net current it receives from the reservoir is given by:

$$I_o(t) = \sum_i w_{oi} \cdot f_i(t) = \sum_i w_{oi} \cdot f_i[(u(t)], \tag{5.1}$$

where $f_i(t)$ is the response of the i^{th} neuron in the reservoir, and w_{oi} is the synaptic weight between the i^{th} reservoir neuron and the readout neuron. The integrated net current over [0, T] is:

$$\int_0^T I_o(t) = \sum_i w_{oi} \cdot \int_0^T f_i(t) = \sum_i w_{oi} \cdot \int_0^T f_i[u(t)]. \tag{5.2}$$

As the integrated net current to each readout neuron is a linear combination of integrated outputs coming from all reservoir neurons, each readout neuron can be treated as a linear classifier of the responses of the reservoir, which can be considered falling into a feature space. Ideally, only reservoir responses produced by input signals from the same class are expected to activate the correspondent readout neuron. Therefore, conceptually in the feature space, the hyperplane defined by all w_{oi}'s separates these inputs from others. Finally, the task of speech recognition is a problem of solving all these linear classification problems by tuning these synaptic weights between the reservoir and readout layer. This can be achieved by applying a learning algorithm.

5.1.2 Learning Algorithm

Hebb's postulate, which claims that neurons fire together wire together, was proposed and widely accepted [95]. In particular, if the firing activity of neuron i tends to excite/inhibit the firing activity of neuron j, the synapse connected from i to j will be potientiated/depressed. Based on this principle, a number of biologically plausible learning algorithms can be used for training the readout layer of an LSM. In the literature, temporal encoding has been adopted in several learning rules, e.g. ReSuMe [11], I-Learning [58], tempotron [55] and SPAN [96]. Firing rate encoding has also been adopted for developing an abstract learning rule [56].

The focus of this chapter is not to study a specific learning algorithm under the context of LSMs, but to examine the performance and robustness of LSMs when a typical biologically plausible learning rule is adopted. In other words, the emphasis of this work is placed upon the key dynamical and network characteristics of the liquid state machine rather than a behavior of a given learning rule.For this purpose, we have opted to use a hardware-friendly learning rule [10], which is motivated by the abstract rule of [56]. We succinctly describe the key features of this adopted rule below.

The adopted rule is based on the principle of Hebbian learning, under which the goal of the learning process is to modulate the activity of the readout neurons according to the desired level, and then tune the weights of plastic synapses correspondingly. More precisely, when a certain readout neuron is expected to fire actively, we drive its firing activity to a high level, with the help of certain teacher signals that implement supervised learning; while at the same time, we inhibit the firing activities of other readout neurons.

To implement the adopted learning rule, we define the calcium variables c_r and c_d to indicate the actual and desired firing activity of a postsynaptic neuron, respectively. To distinguish highly active neurons from inactive ones, a threshold c_θ of the calcium variables is imposed - if the calcium level is higher (lower) than the threshold, the neuron is considered to be at a high (low) activity. The update of plastic synapses only happens when a presynaptic neuron fires and the actual calcium concentration c_r of the postsynaptic neuron is higher (or lower) than c_θ:

$$\begin{cases} w_{ij} \to w_{ij} + \Delta w \text{ with prob. } p_+ \\ \quad \text{iff } neuron_j \text{ fires \& } c_\theta < c_r < c_\theta + \Delta c \\ w_{ij} \to w_{ij} - \Delta w \text{ with prob. } p_- \\ \quad \text{iff } neuron_j \text{ fires \& } c_\theta - \Delta c < c_r < c_\theta, \end{cases} \tag{5.3}$$

where w_{ij} is the weight of the plastic synapse from neuron j to neuron i, Δw is the potentiation/depression granularity, and the parameter Δc is used for good generalization performance. Potentiation or depression of synapses happens with the probability of p_+ or p_-, respectively. The potentiation/depression of synapses is only activated when c_r is in the specific ranges specified by c_θ and Δc. This mechanism is used to avoid saturation of the plastic weights. This learning process is visualized in Fig. 5.2.

Teacher signals are introduced to the readout layer to implement supervised learning. A teacher signal injects a large positive or negative current to the corresponding neuron for the purpose of modulating its real calcium concentration c_r to the desired level of calcium concentration c_d. More specifically, when the samples from a certain input speech class are presented, the readout neuron corresponding to this class is expected to be highly active (i.e. c_d is high). Therefore, its firing activity and calcium concentration are both driven to a high level by the teacher signals, whereas other readout neurons are supposed to be inactive with a low c_d. Consequently, these neurons are driven to the highly inactive region where c_r is low by their teacher signals. The combined use of the learning rule (shown in Equ. 5.3) and the teacher signals potentiates or suppresses the plastic synapses in a way that leads towards separation of different input classes.

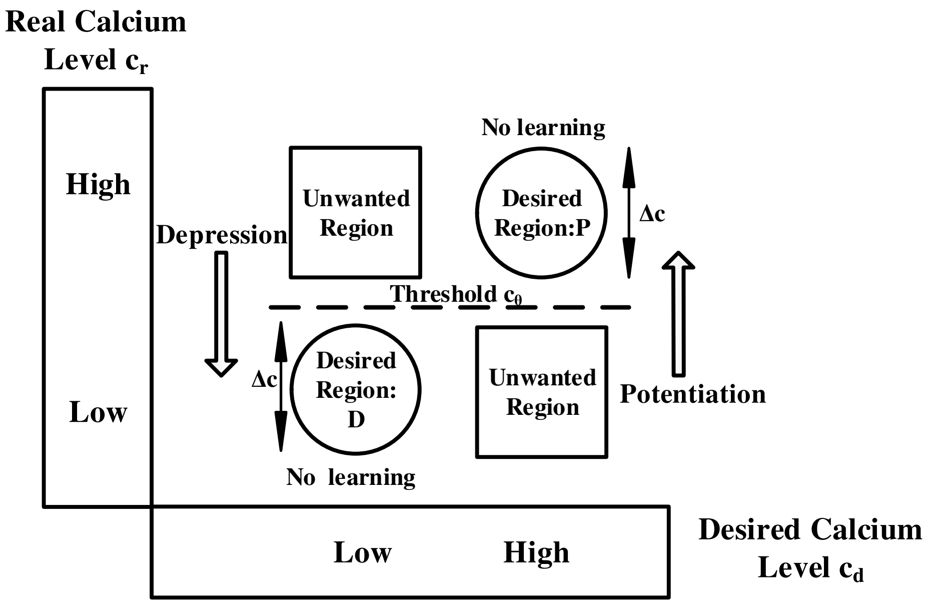

Figure 5.2: Learning process of the LSM. Four regions in the diagram show how different combinations of c_d and c_r of the postsynaptic neuron determine the synaptic plasticity (c_d and c_r are only defined for readout neurons). The two arrows represent depression and potentiation implemented by the teacher signals, driving the activity of a neuron to the desired region (highly active or inactive), where the corresponding synaptic weights are tuned. More precisely, potentiation of the corresponding synapses happens in the desired region marked by "P" and depression happens in the desired region marked by "D". Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

5.1.3 Digitized Simulation Models

In terms of simulation of LSMs, we adopt the digitized leaky-integrate-and-fire (LIF) model for neurons and second order response model for synapses adopted from [10]. The dynamics of the membrane voltage of a neuron can be described by the following equation:

$$V_m^n = V_m^{n-1} - \frac{V_m^{n-1}}{\tau_m} + I_{syn}R_{syn} + I_tR_t, \tag{5.4}$$

where the superscript of V is the index of the time step, V_m and τ_m are the membrane voltage and the time constant of its first-order dynamics, respectively, I_{syn} models synaptic input current, I_t models the input current from the teacher signal, and R_{syn} and R_t model the synaptic resistance and the input resistance associated with the teacher signal. If the membrane voltage (V_m) of a neuron reaches or exceeds the threshold voltage V_{th}, the neuron fires and its membrane voltage is reset to the resting potential V_{rest}. There is an absolute refractory period τ_{refrac} associated with each spike, during which a fired neuron cannot fire again.

The calcium level of a neuron is used to model its firing activity to trigger the learning rule. The dynamics of the calcium level is modeled as:

$$C^n = C^{n-1} - \frac{C^{n-1}}{\tau_c} + \sum_i \delta_{T^n,T_i}. \tag{5.5}$$

Here τ_c is the time constant of this first-order model, i is the index of the spike emitted from this neuron, and $\delta_{x,y}$ is the Kronecker delta whose value is 1 if $x = y$, and 0 otherwise. T_i is the time when the neuron transmits its i_{th} spike and T^n is the simulation time.

The synaptic current I_{syn} to each neuron is modeled as:

$$I_{syn} = \sum_i \sum_j W_i \cdot Syn(T^n, T_{ij} + D_{ij}), \tag{5.6}$$

where i and j are indices of the presynaptic neurons and the spikes, respectively. Specifically, W_i represents the weight of the synapse that connects to the i^{th} presynaptic neuron. T_{ij} is the firing time of the j^{th} spike emitted from the i^{th} presynaptic neuron, and D_{ij} is the corresponding synaptic propagation delay. $Syn(\cdot)$ is the digitized second-ordered dynamic response of a synapse to an incoming spike:

$$\begin{aligned} &Syn(T^n, T_{ij} + D_{ij}) = \\ &\frac{1}{\tau_{s1} - \tau_{s2}} \cdot e^{-\frac{T^n - T_{ij} - D_{ij}}{\tau_{s1}}} \cdot S(T^n - T_{ij} - D_{ij}) \\ &- \frac{1}{\tau_{s1} - \tau_{s2}} \cdot e^{-\frac{T^n - T_{ij} - D_{ij}}{\tau_{s2}}} \cdot S(T^n - T_{ij} - D_{ij}). \end{aligned} \tag{5.7}$$

τ_{s1} and τ_{s2} are two time constants of the model. $S(\cdot)$ is the unit step function. The term $\frac{1}{\tau_{s1} - \tau_{s2}}$ normalizes the response function such that the integrated response of each spike is normalized to one.

The setup of the parameters in the neuron and synpatic model can be found in Section 5.3.

5.1.4 Adopted Speech Benchmarks

In order to benchmark the performance of our LSMs for speech recognition, three subsets of TI46 speech corpus [91] are used. The speech samples in TI46 were collected in a low noise sound isolation booth using an Electro-Voice RE-16 Dynamic Cardiod microphone at $12.5KHz$ sample rate.[2]

The first benchmark is widely used in testing the performance of reservoir computing based speech recognition [97, 15, 98, 67]. It contains isolated word utterances of 5 different speakers. 10 different utterances of each word from “zero” to “nine” are recorded for each speaker. Thus, this benchmark contains 500 speech samples. This is the main benchmark used to study the performance and robustness of the LSMs in this chapter. The second benchmark includes 1,000 utterances of isolated digits in the training set of the TI46 speech corpus. This large subset contains speech samples from 10 speakers. For each speaker in the subset, there are 10 recorded samples of each spoken digit. The third subset contains 10 utterances of each English letter from “A” to “Z”, which were recorded from a single speaker. There are 260 samples in the third benchmark.

These speech samples are preprocessed by Lyon’s ear model, which consists of three prepassing stages: a band-pass filter bank, a half wave rectifier with automatic gain control [92], and BSA, an algorithm converting time domain input signals into spike trains [93, 15]. The average input spike rates of different spoken digits in the first benchmark are illustrated in Table 5.1. Each reported rate is the average rate of different recordings of the same digit. To visualize these speech samples, we show several representative input spike trains of the words “zero”, “three”, “six” and “nine” with the corresponding reservoir responses in Fig. 5.3. The resolutions of synaptic weights and membrane voltages in the reservoir are 10 bits and 16 bits, respectively. Other detailed parameter settings can be found in Section 5.3.

It is worth noting that although the spike rates of input spikes remain roughly the same for

[2]More information of TI 46 is available from `https://catalog.ldc.upenn.edu/LDC93S9`.

different utterances, it can be observed from Fig. 5.3 that the reservoir is able to produce responses with distinctive spatio-temporal characteristics in response to different input speech samples. It can be expected that the mapping from the space of input spikes to the higher-dimensional space of reservoir responses contributes to differentiability across different speech classes.

Table 5.1: Average input spike rates for different words in the first benchmark. Each spike rate is the average rate of different recordings of the same digit. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

Digit	0	1	2	3	4	5	6	7	8	9
Spike rate (kHz)	9.0	7.0	8.5	7.7	7.2	8.2	7.5	7.5	5.7	7.6

5.1.5 Network and Training Setup

The spiking network and training are set up according to [10]. As illustrated in Fig. 5.1, the reservoir has a grid structure. 20% of neurons in the reservoir are randomly chosen to be inhibitory while the rest are excitatory. The connectivity in the reservoir is constructed randomly under a distribution such that the wiring probability of any two neurons (N_i and N_j) drops exponentially in the distance between them [54]:

$$P_{connect}(N_i, N_j) = k \cdot e^{-\frac{D^2(N_i,N_j)}{r^2}} \; (i \neq j), \tag{5.8}$$

where $D(N_i, N_j)$ is the Euclidean distance between these two neurons, r is chosen to control the exponential decay of the probability, and k is a constant depending on the neuron type. The parameters are chosen according to the values suggested by [10].

In the network, each input spike train generated in the preprocessing stage is sent to four randomly chosen reservoir neurons through synapses with fixed weights randomly chosen to be W_{max} or W_{min}, where W_{max} and W_{min} are maximum and minimum synaptic weights used in

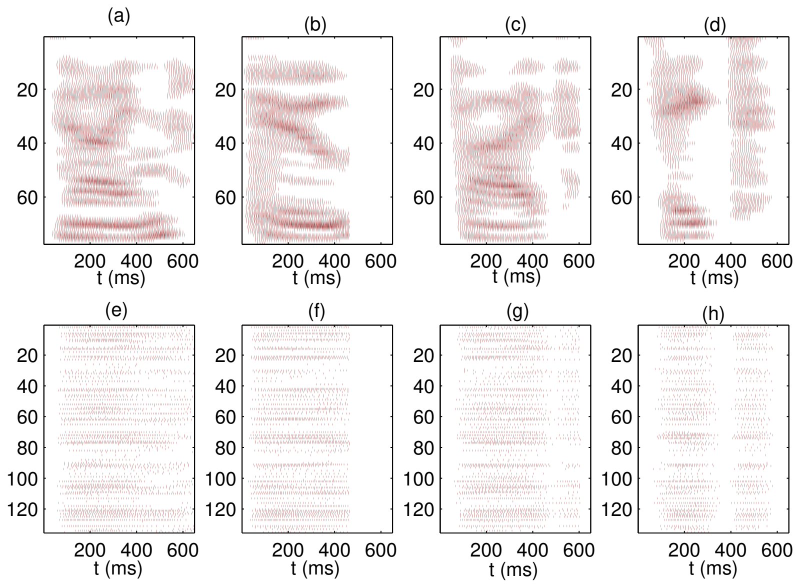

Figure 5.3: (a)-(d) are the input spike trains of the utterances “zero”, “three”, “six” and “nine”, with the y-axis showing the indices of the input channels. (e) - (h) show the corresponding neuron activities in the reservoir for the words “zero”, “three”, “six’ and “nine”, respectively. The y-axis represents the indices of the reservoir neurons. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

the simulation, respectively. Reservoir neurons are fully connected to each readout neuron by plastic synapses, whose weights are randomly initialized between W_{max} and W_{min}. The plastic synapses are trained by the adopted learning algorithm. The synaptic weights in the reservoir are fixed according to the neuron type. The detailed parameter settings of synaptic connections can be found in Section 5.3.

To test the performance of the various LSM designs considered in this chapter, we adopt a 5-fold cross validation scheme to determine the speech recognition rate. In this setup, all speech samples are randomly divided into five groups. Based on these samples, a fixed LSM is trained and tested for five times with different training and testing datasets. For the i_{th} ($i = 1, 2, 3, 4, 5$) time, the i_{th} group is used for testing and the remaining data for training. The recognition decision is made after each testing speech sample is played. At this time, the readout neuron that has fired most frequently is the winner and its associated class label is deemed to be the classification decision of the LSM. Finally, the five classification rates obtained in the testing stage are averaged as final performance measure.

5.1.6 Performance of the Base-line LSM

We set up a baseline LSM as a reference for the presented design space exploration. There are 135 neurons in this baseline LSM and the resolutions for membrane voltages and synaptic weights are set to be 16 bits and 10 bits, respectively. The detail of other parameter settings will be discussed in Section 4. The best recognition rate of this LSM is 99.2% based upon the first benchmark described in Section 5.1.4.

5.2 Theoretical Measures of Computational Performance

To gain insights into the LSM network dynamics and its relation to learning performance, we adopt three theoretical measures of computational power to analyze the presented LSMs. First, we measure the "fading memory" of the reservoir of a given LSM [54, 97], characterizing how well the reservoir "memorizes" temporal input patterns. From a dynamic system point of view, we examine the operating regime of an LSM and quantify its distance to the so-called edge between order and chaos [99]. Finally, from a task-oriented point of view, we analyze the LSMs in terms of their separation property and generalization capability [100].

5.2.1 Fading Memory

First of all, we theoretically estimate how well the dynamics in the reservoir helps to "memorize" different input patterns by measuring its fading memory[3] (Fig. 5.4).

By observing the responses in the reservoir, we intuitively approximate the fading memory. As proposed in [54], one way to empirically measure fading memory is to count the number of firing neurons and calculate the duration of firing activity in the reservoir after injecting random temporal signals into the LSM. Long lasting and strong firing activity is usually desirable because it implies the strong memory capacity possessed by the reservoir.

[3]see [54] for a detailed definition.

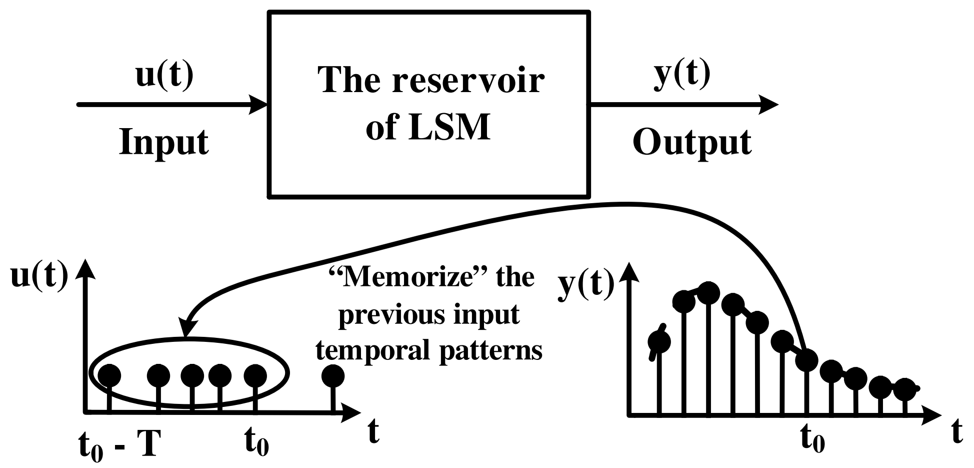

Figure 5.4: Fading memory of an LSM. The temporal input stream $u(t)$ is transformed by the reservoir into a high-dimensional signal $y(t)$, which holds the information about the recent history of the input $u(t)$ ($[t_0 - T, t_0]$). Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

5.2.2 Edge of Chaos

Second, we theoretically analyze an LSM from a dynamic system point of view. Related studies [101, 102] suggest that dynamic systems operating near the phase transition between order and chaos (i.e. "edge of chaos") possess a good amount of computational power. To determine the ordered and chaotic regimes for discrete dynamical systems driven by online inputs, [99] proposed to track the evolution of state difference resulting from two close initial states while the system is driven by the same input. The state difference of a chaotic system is highly amplified while that of an ordered system vanishes quickly. One can quantitatively analyze the phase transition by Lyapunov exponents [103]. We look for the exponent λ that is determined by

$$\delta_{\Delta T} \approx \delta_0 \cdot e^{\lambda \Delta T}, \delta_0 \to 1, \Delta T \gg 1. \tag{5.9}$$

Here δ_0 represents the initial state difference at time 0 and $\delta_{\Delta T}$ is state separation at time ΔT. As depicted in Fig. 5.5, $\lambda > 0$ suggests that the system is chaotic while $\lambda < 0$ indicates an ordered system. The dynamical system sits on the transition boundary if its λ is equal to 0. In our

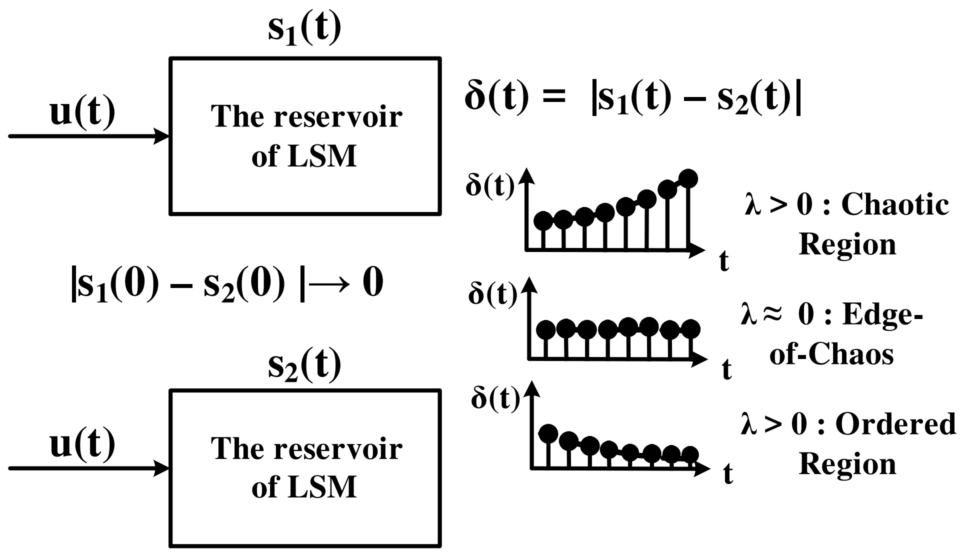

Figure 5.5: Edge of chaos of an LSM. With the same input $u(t)$ and two initially close states ($s_1(t)$ and $s_2(t)$), the difference between two states is recorded and measured as the dynamics of the LSM evolves. The Lyapunov exponent λ theoretically reveals whether or not the system is on the phase transition boundary. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

measurement, we define the state of the LSM as a binary vector $s(t) = [s_1(t), s_2(t), \cdots, s_n(t)]$, with s_i setting to one when the i_{th} reservoir neuron fires at time t. The Hamming distance between two states is defined as the state difference.

5.2.3 Separation and Generalization

Third, as illustrated in Fig. 5.6, we investigate the theoretical computational power by quantitatively analyzing two essential properties, i.e. separation and generalization, of an LSM to characterize its performance from an application perspective [100]. Separation of the reservoir reflects the kernel-quality of the neural circuit and generalization measures how well the reservoir can generalize a learned function to new input streams.

As suggested in [100], the separation and generalization properties of the reservoir are estimated by computing the rank of an $n \times m$ matrix M, where n is the number of state variables of the reservoir, m is number of inputs, and each column of M is the state vector $x_{u_i}(t_0)$ under the incoming input stream u_i at a fixed time point t_0. To measure separation property in our case,

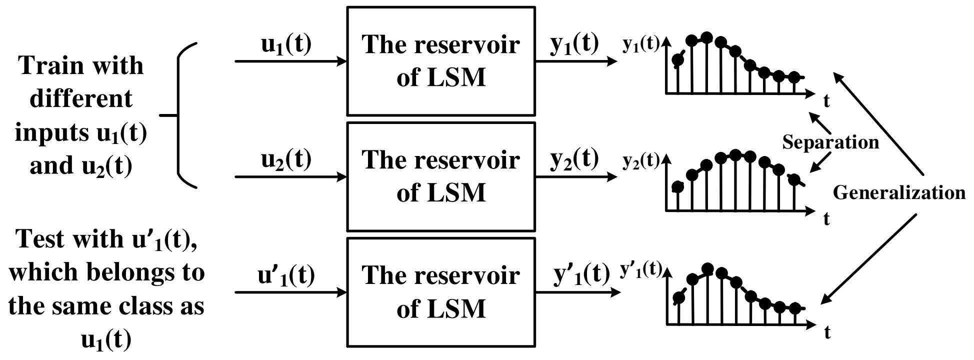

Figure 5.6: Generalization and separation of an LSM. When an LSM is trained with two different inputs ($u_1(t)$ and $u_2(t)$), the outputs of the reservoir ($y_1(t)$ and $y_2(t)$) are expected to be distinct because of the separation property. While tested with the input $u_1'(t)$ which belongs to the same class as $u_1(t)$, the output of the reservoir $y_1'(t)$ should be similar to $y_1(t)$ because of good generalization. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

randomly generated input streams are used; while for approximating the generalization capability, application-specific speech signals are used. According to [100], a large difference between r_S (the rank estimating the separation capability) and r_G (the rank representing the generalization capability) is usually a good indicator of strong computational power with respect to the specific task at hand.

5.3 Performance of LSMs and Its Dependencies on Key Model/Design Parameters

The performance of an LSM immediately depends on several network design parameters and these parameters in turn greatly determine the resulting hardware implementation cost. When it comes to the application of speech recognition, one major design choice that needs to be made is what level of precision should be maintained to guarantee the good performance of LSMs when emulating the behaviors of neurons and synapses. This question is meaningful both from a biological modeling and an engineering point of view. First, nervous systems in nature exhibit trial-to-trial variability [104] in the present of intrinsic noise, therefore it is not necessary to have extremely high precision in modeling the dynamics of neurons and synapses. Second, from an engineering perspective, it is critical to choose an appropriate level of precision for the targeted application because excessive precision leads to unnecessary increase in implementation overhead.

To achieve the above goal, we conduct behavior-level simulations to study the performance of LSMs with a broad range of design parameters. The considered parameters, including ones for the digitized LIF neuron model, synaptic model, and learning rule, are summarized in Tables 5.2 - 5.4. Some of the parameters have been adopted from [10]. In Table 5.2, two extra parameters, V_{max} and V_{min}, are imposed as the upper bound and lower bound upon the membrane voltage V_m, for the purpose of discreteness. The same discreteness is also applied to synaptic weights and calcium concentrations. In Table 5.3, E and I indicate excitatory and inhibitory neurons, respectively. $E \rightarrow I$ denotes connections from excitatory presynaptic neurons to inhibitory postsynaptic neurons. In Table 5.4, the strength of the teacher signal I_t is set to be $\frac{V_{th}}{R_t}$ for potentiation, and $-\frac{3V_{th}}{4R_t}$ for depression.

Table 5.2: Parameters of the neuron model. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

Parameter	Value
Threshold voltage V_{th}	$20\ mV$
Resting potential V_{rest}	$0\ mV$
Time constant τ_m	$32\ ms$
Time constant τ_c	$64\ ms$
Refractory period τ_{refrac}	$2\ ms$
Upper bound of membrane voltage V_{max}	$32\ mV$
Lower bound of membrane voltage V_{min}	$-32\ mV$
Granularity of membrane votlage δV	$\frac{V_{max}-V_{min}}{2^{n_{mem-bit}}}$ [a]

[a] $n_{mem-bit}$: the resolution of the membrane voltage.

To attain the simulated recognition performance of each sample point of the design parameter space, five randomly generated LSMs are trained and tested for speech recognition. To optimize the performance, we train the LSMs for multiple iterations. For each generated LSM, the best recognition rate is computed after multiple training iterations and we average five obtained best recognition rates of the LSMs as the reported recognition rate. The standard deviation (SD) of the

Table 5.3: Parameters of the synaptic model. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

Parameter	Value	
Fixed weights in the reservoir	type	value
	$E \to E$	3
	$E \to I$	6
	$I \to E$	-2
	$I \to I$	-2
Upper bound of synaptic weights W_{max}	$E/I \to E/I$	8
Lower bound of synaptic weights W_{min}	$E/I \to E/I$	-8
Time constant τ_{s1} of the second-order synaptic dynamics	$E \to E/I$	4
	$I \to E/I$	8
Time constant τ_{s2} of the second-order synaptic dynamics	$E \to E/I$	4
	$I \to E/I$	2
Synaptic propagation delay D_{ij}	1	
Synaptic resistance R_{syn}, R_t	$1\ \Omega$	
Granularity of synaptic weights δW	$\frac{W_{max}-W_{min}}{2^{n_{syn-bit}}}$ [b]	

[b] $n_{syn-bit}$: the resolution of the synaptic weight.

five best recognition rates is measured to report variation of recognition performance.

To comprehensively study the relation between a broad range of parameters and performance, we investigate from three aspects: resolution of the neuron model, resolution of the synaptic model and size of the reservoir. As mentioned in Section 5.1.6, we use the base-line LSM as the reference design (see Table 5.5 for the key parameter settings) and apply the first adopted benchmark for conducting the performance study. In addition to simulation, we also theoretically characterize the computational power of the targeted LSMs under various parameter settings.

5.3.1 Resolution of Membrane Voltage and Calcium Level

First of all, we examine how the precision of membrane voltage and calcium level of the neurons can influence recognition performance. While reservoir and readout neurons have different roles in the LSM, we separately analyze these two types of neurons. The performance of the LSM with different resolution settings is plotted in Fig. 5.7 and Fig. 5.8. As mentioned in Section 5.3,

Table 5.4: Parameters used in the learning rule. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

Parameter	Value
Granularity of calcium level δc	$2^{n_{cal-bit}-4}$ [c]
Upper bound of calcium level c_{max}	$16 \times \delta c$
Lower bound of calcium level c_{min}	0
Threshold of calcium level c_θ	$5 \times \delta c$
Generalization parameter Δc	$3 \times \delta c$
Teacher signal strength I_t	$\frac{V_{th}}{R_t}$ or $-\frac{3V_{th}}{4R_t}$
Learning probability p_+, p_-	$\frac{0.004}{2^{n_{syn-bit}-4}}$
Potentiation/depression granularity ΔW	δW

[c] $n_{cal-bit}$: the resolution of the calcium level.

Table 5.5: Key design parameters of the reference design. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

Design Parameter	Neuron Type	Resolution /Value
Calcium Level	Readout[4]	14 bits
Membrane Voltage	Reservoir	16 bits
	Readout	16 bits
Synaptic Weight	Reservoir	10 bits
	Readout	10 bits
Size of Reservoir	N.A.	135 neurons

each plotted point in Fig. 5.7 and Fig. 5.8 is the averaged recognition rate of the five recognition rates obtained by training and testing five randomly generated LSMs with different random seeds for the generation of random connections inside the reservoir. Each error bar in the figure represents the standard deviation (SD) of the five recognition rates with its length being $2 \times SD$.

The curve with circles in Fig. 5.7 shows the simulated recognition rates of LSMs with a decreasing resolution of membrane voltage for reservoir neurons while the other design parameters are fixed according to Table 5.5. The simulation results suggest that the recognition performance only degrades slightly when the precision of membrane voltage for reservoir neurons is reduced

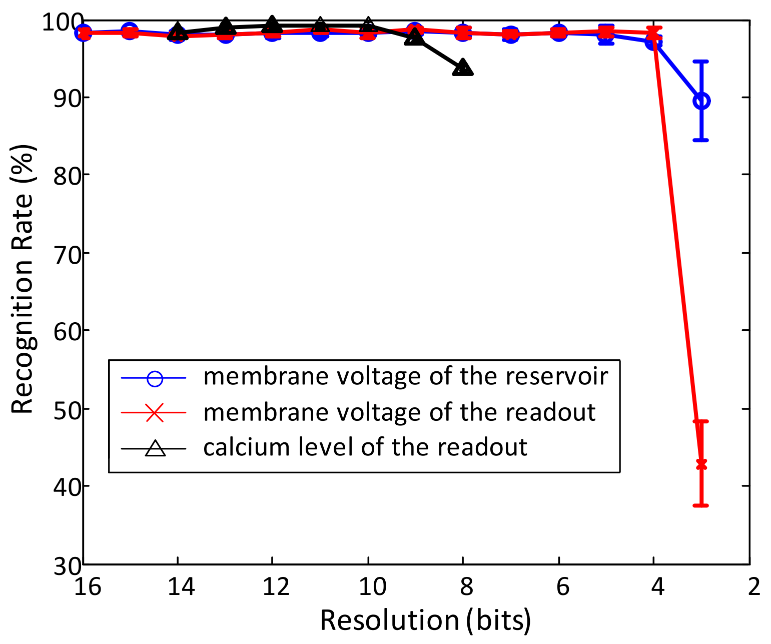

Figure 5.7: Performance of the LSMs drops as a function of the decreasing bit-resolutions of the membrane voltage and calcium level. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

down to 6 bits. This phenomenon may be understood by noticing that under a low membrane voltage resolution, fixed and recurrent connections in the reservoir may still be able to propagate the firing activities initialized by a few neurons to create rich dynamics in the network. But the resolution cannot be too low (e.g. below 3 bits or 4 bits) because the firing activity of the reservoir will not reflect the critical information of speech samples well under a very low resolution. Similarly, a low resolution for the membrane voltage of readout neurons will not cause much performance degradation. As shown in the curve marked with "x" in Fig. 5.7, LSMs start to perform poorly only when the resolution drops below 3 bits, where the average recognition performance is about 40%. The result suggests that the activation of the correct winning readout neuron for a given input speech is not very sensitive to the bit resolution used for the membrane voltage of readout neurons.

To have a more comprehensive study of performance sensitivity to membrane voltage resolutions, we test the performance under different combinations of resolution settings for the reservoir and readout (shown in Fig. 5.8). The performance is not sensitive to wide range variations of

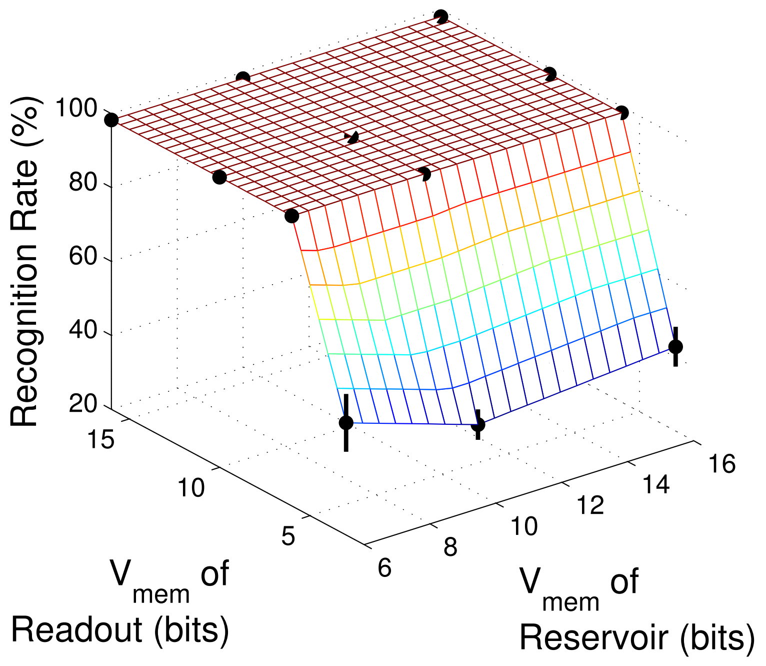

Figure 5.8: Performance of the LSMs with different combinations of reservoir and readout membrane voltage resolutions. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

membrane voltage resolutions, particularly for the case of the resolution of reservoir neurons.

The calcium level, used in the learning algorithm, is another important parameter associated with readout neurons. The curve denoted by triangles in Fig. 5.7 manifests that the recognition performance degrades noticeably as the resolution of the calcium level is lower than 10 bits. This suggests that the calcium concentration plays an important role in learning and hence a fine resolution may be needed to accurately tune the plastic synaptic weights of readout neurons to achieve good performance. We also examine various combinations of the calcium level threshold (c_θ) and generalization parameter (c_Δ), which are two parameters of the learning rule (see Table 5.4), to investigate their impacts on the performance. As shown in Fig. 5.9, the choices of the two learning parameters within the considered range have no significant impact on the performance.

In addition to the simulated recognition performance presented above, we further use the three theoretical measures of computational power mentioned before to characterize the performance

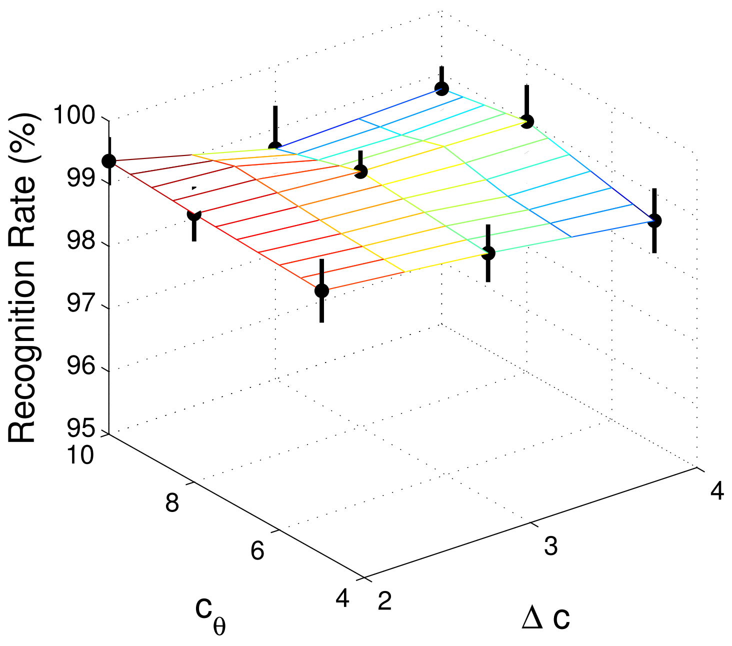

Figure 5.9: Performance of the LSMs with different combinations of parameters in the learning rule. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

impacts of the resolution of membrane voltage. As will be seen, these theoretical measures provide largely consistent performance evaluations of the LSMs while offering additional understanding of network dynamics and associated computing power. Note that the three theoretical measures aim at examining the computational power resulting from the dynamics created in the reservoir, which focus on the following analysis.

Fading Memory

To measure the fading memory, we inject 77 random temporal spike trains ending at $23ms$ into the reservoir and record the responses of the reservoir (i.e. the number of firing neurons and the duration of the firing activities). The impacts of lowering precision of reservoir neurons' membrane voltage on fading memory are shown in Fig. 5.10. The results clearly suggest that the reservoir has large fading memory even when the resolution of its membrane voltage is reduced to 6 bits. The observation is in agreement with Fig. 5.7 and once again implies that the implementation of

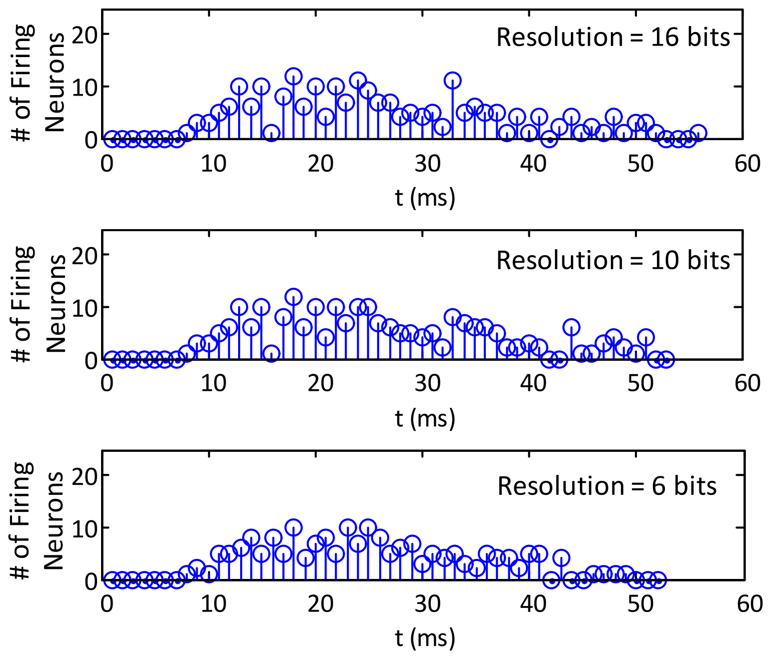

Figure 5.10: Firing activities (fading memory) in the reservoir when the resolution of membrane voltage for the reservoir is reduced. Three reservoirs with descending resolutions are tested by injecting 77 random input spike trains that are all ended at $23ms$. A lowered resolution does not necessarily lead to reduced fading memory. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

reservoir neurons can be simplified to lower hardware overhead without any significant degradation of performance.

Edge-of-Chaos

Applying an approach similar to that is adopted in [105], we introduce an infinitesimal initial state difference to characterize the operating regime of an LSM by using a pair of two slightly different inputs. Two such input pairs are used in order to eliminate the randomness introduced by the choice of specific inputs. For the first input pair, the only difference between the two input spike trains results from one missing spike at $24ms$ from one spike train. For the second input pair, the difference between two spike trains is due to one missing spike at $42ms$ from one of the input spike trains. The two slight different inputs are only used to introduce an initial state difference and after that a future state difference is used to compute the Lyapunov exponent. By independently feeding the two different inputs to an LSM, the resulting state difference $\delta_{\Delta t}$ is observed at a future time point. To be specific, $\delta_{\Delta t}$ is measured at $\Delta t = 300ms$ across the two slightly different input streams, where Δt is the elapsed time from when the initial state difference occurs to the observation time point. Then the Lyapunov exponent λ is determined by [105]

$$\lambda = \frac{1}{\Delta t}\ln(\frac{\delta_{\Delta t}}{\delta_{ini}}), \tag{5.10}$$

where δ_{ini} is the initial state difference introduced by the two slight different inputs.

Fig. 5.11 shows how the state separation temporally evolves due to a small input difference for three LSMs with a descending membrane voltage resolution. It is obvious that although the precision is reduced, the state difference remains roughly at the same level. In other words, the reservoir's membrane voltage resolution might not significantly influence its dynamics, which is further confirmed by the calculated Lyapunov exponent λ shown in Table 5.6.

As can be seen from Table 5.6, the Lyapunov exponents of the three LSMs are relatively small and close to zero, indicating that they operate in a region that is close to the transition boundary, which is consistent to the corresponding good recognition performance. In addition, the calculated

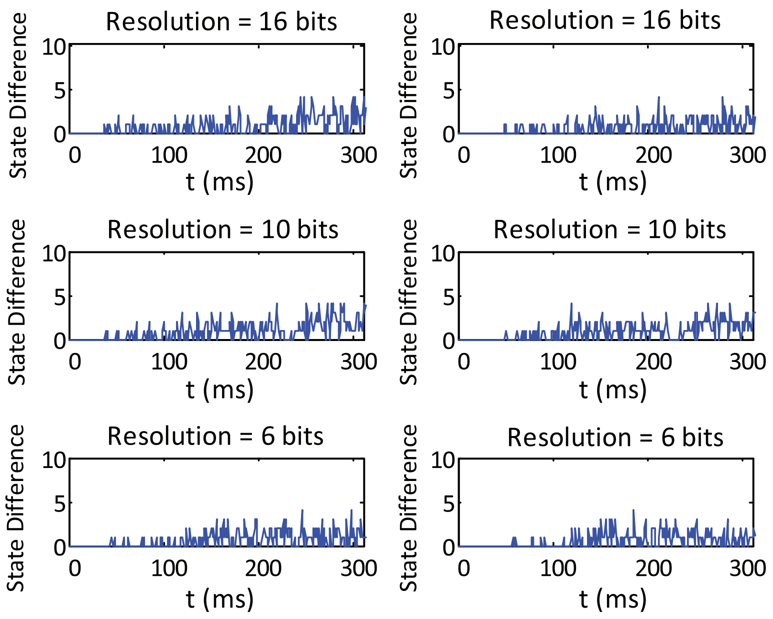

Figure 5.11: State separation under different resolution settings for membrane voltage in the reservoir. The temporal evolution of the Hamming distance between the two resulting states $x_u(t)$ and $x_v(t)$ is shown. For the sub-figures on the left and right sides, the input u differs from input v due to only one missing spike at time $24ms$ and $42ms$, respectively. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

Lyapunov exponents suggest that a low resolution of membrane voltage may be sufficient for achieving good recognition performance.

Separation and Generalization

We use the method for estimating the separation and generalization capabilities mentioned in Section 5.2.3. After applying 500 randomly generated input streams and 500 application specific speech signals, we measure the ranks r_G and r_S respectively of the matrix M at five fixed time points from $394ms$ to $399ms$ and report the maximum obtained ranks for estimation of separation and generation. The rank difference Δ_{SG} between r_S and r_G is calculated as shown in Table 5.7. As shown in Table 5.7, interestingly, in the range which is considered for the membrane voltage resolution of the reservoir, lowering the resolution does not affect the computational capability very much. The approximated computational ability reconfirms that the good classification performance

Table 5.6: Lyapunov exponents of LSMs with various resolutions of reservoir neurons' membrane voltage. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

LSM	Resolution	λ_1	λ_2	Recognition Rate (%)
1	16 bits	0.36	0.30	98.16%
2	10 bits	0.32	0.28	98.24%
3	6 bits	0.22	0.26	98.28%

Table 5.7: Estimated separation and generalization capabilities of the LSM as a function of reservoir neurons' membrane voltage resolutions. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

LSM	Resolution	r_S	r_G	Δ_{SG}	Recognition Rate (%)
1	16 bits	135	101	34	98.16%
2	10 bits	135	102	33	98.24%
3	6 bits	135	99	36	98.28%

might be achieved given the low resolution of the reservoir neurons' membrane voltage.

5.3.2 Resolution of Synaptic Weights

Here we study how the resolution for synaptic weights affects the overall performance. Since synapses within the reservoir have fixed efficacy while synapses connected to the readout are plastic, we consider them separately when changing the resolution.

First, we take a close look at the synapses in the reservoir. The curve with circles in Fig. 5.12 shows the recognition rates of LSMs with different resolutions of synaptic weights for the reservoir synapses while the resolutions for other parameters are fixed according to Table 5.5. Clearly, reducing the resolution of fixed synaptic weights has little impact on performance. This observation may be understood by recalling that no synaptic weight adaption takes place in the reservoir and the main functionality of the reservoir is to create rich dynamics to map the input to a higher dimensional space. As a result, low-resolution or even binary synapses may be sufficient for the

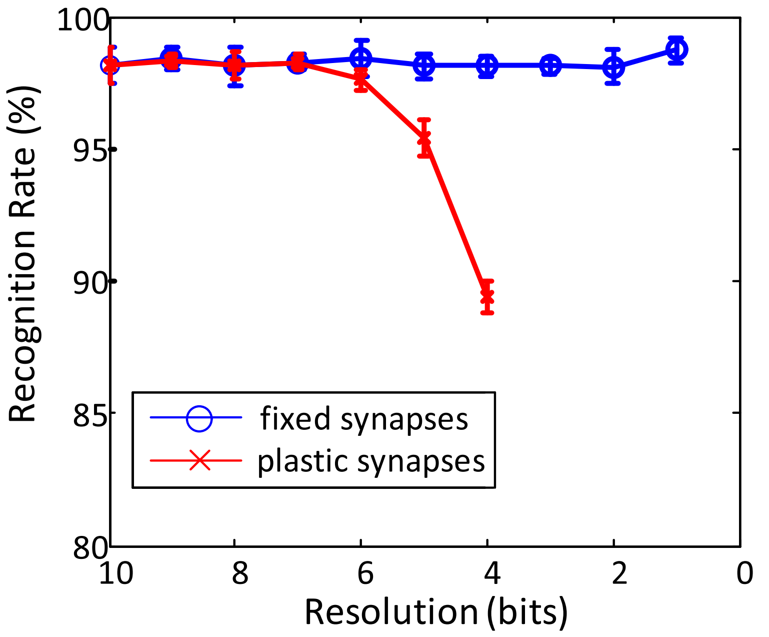

Figure 5.12: Performance of the LSMs degrades as a function of the reduced bit-resolutions of the synaptic weights. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

reservoir.

However, in terms of the plastic synapses between the reservoir and the readout, fine resolution is desirable because synaptic weights with high precision guarantee the accuracy of adjusting the location and orientation of the hyperplane implemented by the linear readout. We examine how the precision of plastic synaptic weights influences performance experimentally. In the simulations, 10-bit synapses are used in the reservoir. The performance degradation of LSMs with different resolution settings for plastic synapses can be seen in Fig. 5.12. We conclude that 8-bit resolution is needed for efficacy of plastic synapses because further reduction will cause a fairly large performance loss.

We vary the resolutions of fixed and plastic synaptic weights together to obtain a complete picture of how synaptic weight resolutions can affect performance. The simulation results shown in Fig. 5.13 reconfirm that the resolution of fixed synapses has a limited effect on the performance while the plastic synapses do immediately affect the overall recognition rates.

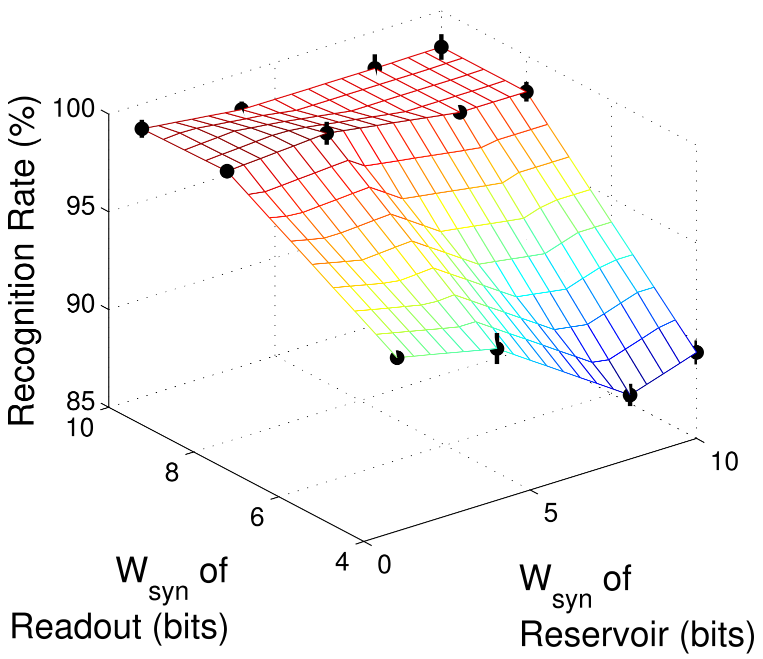

Figure 5.13: Performance of the LSMs with different combinations of resolutions of fixed and plastic synaptic weights. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

In the following, we use the three theoretical measures of computational power to correlate with the above simulated recognition performance. The same experimental setup is used here as what is mentioned in Section 5.3.1.

Fading Memory

After injecting 77 random spike trains into the reservoir and counting the number of fired neurons and measuring the duration of the response, we obtain the fading memory of the reservoir shown in Fig. 5.14. As depicted in Fig. 5.14, the synaptic resolution in the reservoir has limited influence on fading memory, which explains why binary synapses can be used in the reservoir for reducing the complexity of LSMs.

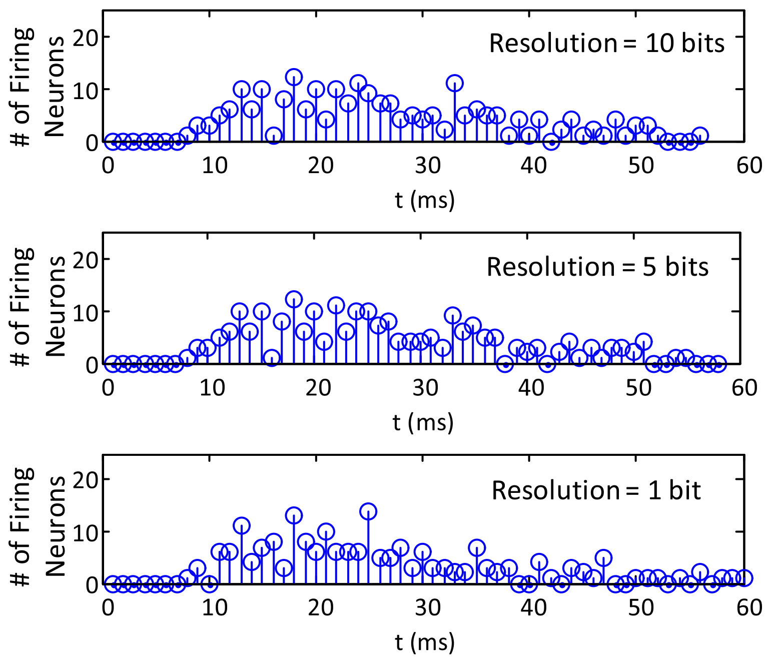

Figure 5.14: Firing activity in the reservoir when the resolution of synaptic weights for fixed synapses gets reduced. Similarly to the results shown in Fig. 5.10, reduction of synaptic weights for the reservoir does not affect fading memory significantly. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

Edge-of-Chaos

Two slightly different input streams are injected into the same reservoir and Fig. 5.15 shows how state divergence temporally evolves due to a small input difference for three LSMs with a descending resolution of the fixed reservoir synapses. By lowering the precision of synaptic weights, we observe that the state difference decreases slightly. The characteristics of the reservoir dynamics are reflected by the Lyapunov exponents λ shown in Table 5.8.

The computed Lyapunov exponents in Table 5.8 are relatively small and close to zero, suggesting that the corresponding dynamics of the all three reservoirs are close to the "edge-of-chaos", and thus good performance can be achieved. And the Lyapunov exponent of the LSM with binary reservoir synapses is even closer to zero, indicating that its dynamics is closer to the transition boundary. In other words, by reducing the resolution of synaptic weights, it may be possible for us to move the dynamics of the reservoir from the chaotic region towards the ordered region, and

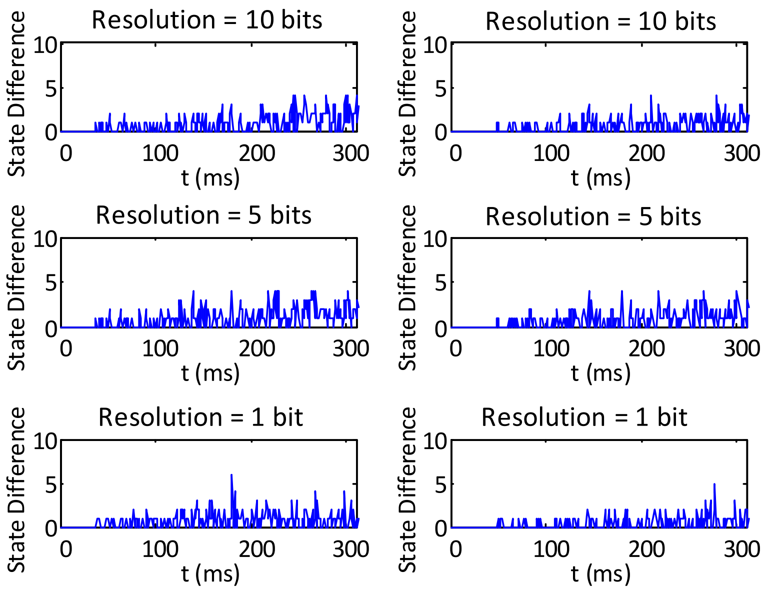

Figure 5.15: State separation under different resolutions for synaptic weights in the reservoir. Similar to Fig. 5.11, the temporal evolution of the Hamming distance between the two resulting states is shown. For the sub-figures on the left and right sides, the input u differs from input v due to only one missing spike at time $24ms$ and $42ms$, respectively. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

hence having the system operating at the transition boundary.

Separation and Generalization

Randomly generated input streams and application specific speech signals are applied to the reservoir separately and we measure the ranks r_S and r_G respectively of the matrix M as mentioned in Section 5.2.3. As seen in Table 5.9, the rank difference Δ_{SG} almost remains the same when lowering the resolution of the fixed synaptic weights, which suggests that low resolution does not have a significant impact upon the computational capability. And therefore, good recognition performance can be achieved under low resolution of reservoir's synaptic weights.

In conclusion, low resolution of synaptic weights may be adequate to attain good computational capability of the reservoir, and thus guarantees good separation and generalization. In terms of the readout layer, however, high resolution synapses are required. It is worth mentioning that by applying a coarser synaptic resolution, we can altogether lower the resolution of membrane voltage

Table 5.8: Lyapunov exponents of LSMs with various resolutions of the reservoir synapses. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

LSM	Resolution	λ_1	λ_2	Recognition Rate (%)
1	10 bits	0.36	0.30	98.16%
2	5 bits	0.38	0.36	98.40%
3	1 bits	0.10	0.18	98.72%

Table 5.9: Estimated separation and generalization capabilities of the LSM as a function of fixed synaptic resolutions. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

LSM	Resolution	r_S	r_G	Δ_{SG}	Recognition Rate (%)
1	10 bits	135	101	34	98.16%
2	5 bits	135	102	33	98.40%
3	1 bits	134	98	36	98.72%

for the reservoir and readout neurons. Finally in terms of neurons in both the reservoir and readout, 6-bit resolution for membrane voltage and 10-bit resolution for calcium concentration are sufficient to guarantee good performance.

5.3.3 Size of the Reservoir

Another way to reduce the implementation cost is to cut down the size of the reservoir. To examine this reduction's impact on performance, we randomly remove a certain percentage of neurons from the reservoir. The original reservoir contains 135 neurons. The percentage of the removed neurons is varied from 1% to 90% and the resulting recognition rates are plotted in Fig. 5.16. Here we use binary synapses for the reservoir, 8-bit synaptic resolution for the readout, 6-bit membrane voltage resolution for both reservoir and readout neurons, and 10-bit calcium concentration resolution for the readout. In the simulation, each time an LSM is trained and tested with randomly chosen neurons being removed from the reservoir. The simulation results suggest that it is

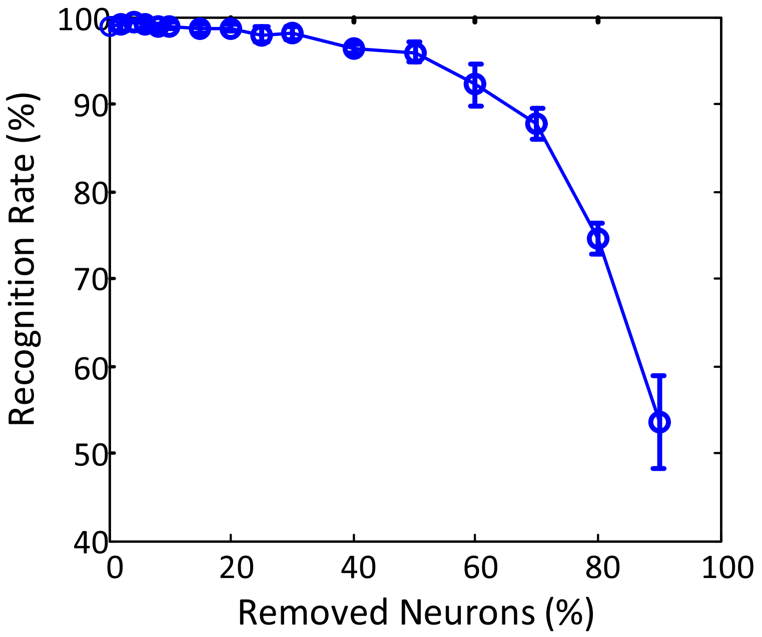

Figure 5.16: Performance degrades as the percentage of removed reservoir neurons increases. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

possible to get reasonably high performance without 30% of neurons but the performance begins to degrade noticeably if more neurons get removed. In order to shed light in a theoretical perspective, we again extract the three measures of computational power for these LSMs with the same experimental setup.

Fading Memory

77 random spike trains are injected into the reservoir for measuring fading memory. In terms of the reservoir size, although fading memory is slightly weakened when squeezing the reservoir, removing too many neurons yields a performance penalty since fading memory will quickly die out or even vanish (shown in Fig. 5.17), making the reservoir incapable of generating rich dynamics. In this case, good performance can still be obtained given that 40 neurons (30% of the original size) get removed from the reservoir because fading memory is well preserved.

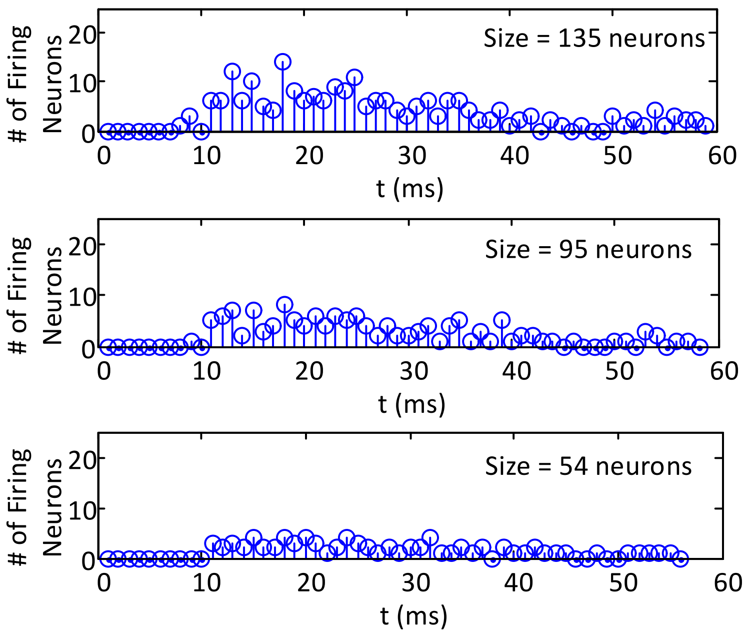

Figure 5.17: Firing activity in the reservoir when its size gets reduced. Compared to the results shown in Fig. 5.10 and Fig. 5.14, the fading memory stays nearly the same after removing 30% neurons from the reservoir. Both the magnitude and duration of the responses are largely reduced if more neurons are removed. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

Edge-of-Chaos

Two slightly different input signals are injected into the reservoir and Fig. 5.18 shows how state separation temporally evolves due to a small input difference for three LSMs with descending reservoir sizes. It is clear that by reducing the size of the reservoir, the state difference gradually decreases; in other words, we might be able to change the dynamics of the reservoir from the chaotic region to the ordered region and phase transition happens around the point where 30% neurons get removed, which is further confirmed by the calculated Lyapunov exponent λ shown in Table 5.10. The Lyapunov exponent of the second LSM in the Table 5.10 is reasonably close to zero, indicating that the dynamics at this point lies very close to the edge of chaos, which theoretically explains why decent performance can be obtained with this size. Furthermore, in this particular case, LSMs will not function very well if the dynamics is in the ordered region.

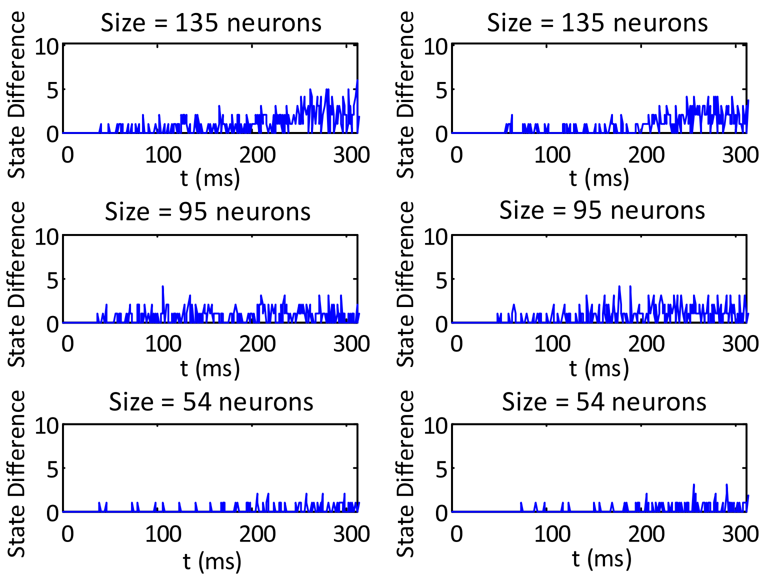

Figure 5.18: State separation for the reservoir with different sizes. Similar to Fig. 5.11, the temporal evolution of the Hamming distance between the resulting states is shown. For the sub-figures on the left and right sides, the input u differs from input v due to only one missing spike at time $24ms$ and $42ms$, respectively. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

Separation and Generalization

The ranks r_S and r_G of the matrix M (see Section 5.2.3 for details) are obtained by feeding randomly generated input streams and application specific speech signals into the reservoir, respectively. As seen in Table 5.11, although the first LSM is shown to possess powerful separation and generalization capabilities and has very good performance, the second LSM, whose Δ_{SG} is not the largest, also achieves good recognition performance. This interesting phenomenon can be understood by noting that multiple dynamic regions can provide the LSM with good performance [100]. Nonetheless, further reduction beyond this point is reconfirmed to cause performance degradation.

5.3.4 Summary of Performance Study

Finally, we summarize the relation between a broad range of key network parameters and the recognition rates with different levels of performance sensitivity in Table 5.12. Clearly, there

Table 5.10: Lyapunov exponents of LSMs with various reservoir sizes. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

LSM	Reservoir Size	λ_1	λ_2	Recognition Rate (%)
1	135 neurons	0.10	0.18	98.92%
2	95 neurons	-0.05	0.00	98.16%
3	54 neurons	-0.40	-0.32	92.24%

Table 5.11: Estimated separation and generalization capabilities of the LSM as a function of the reservoir size. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

LSM	Reservoir Size	r_S	r_G	Δ_{SG}	Recognition Rate (%)
1	135 neurons	135	98	37	98.92%
2	95 neurons	95	69	26	98.16%
3	54 neurons	54	36	18	92.24%

exists a large design space in which various network design parameters can be explored to trade off between hardware overhead and performance. In particular, our experimental study presented here demonstrates the possibility of reducing the network complexity, hence implementation overhead, without incurring any significant degradation of performance. For instance, with the resolutions and the reservoir size getting reduced to the suggested values in Table 5.12, the recognition rate can still reach up to 98.16%.

The adopted theoretical measures of computational power are normally consistent with the real-world performance. It is worth mentioning that the correlation between the theoretical measures and performance might not be straightforward sometimes, because it is found that the LSM can have numerous dynamical regimes with rich computational power for real-world applications.

5.4 Robustness of LSMs with Respect to Catastrophic Failures and Random Errors

The resilience of digital VLSI circuit has been one of the greatest design challenges in the past decades due to the scaling of IC manufacturing technologies and aggressive sizing of transis-

Table 5.12: Key network parameters and their corresponding performance sensitivity. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

Specifications	Type	Suggested setting	Range	Best/Worst Performance	Sensitivity
Calcium Level	Readout	10 bits	$14-8$ bits	99.16%/ 93.76%	Low/ Medium
Membrane Voltage	Reservoir	6 bits	$16-4$ bits	98.52%/ 97.2%	Low
	Readout	6 bits	$16-4$ bits	98.68%/ 98.12%	Low
Synaptic Weight	Reservoir	1 bit	$10-1$ bits	98.72%/ 98.08%	Low
	Readout	8 bits	$10-4$ bits	98.36%/ 89.4%	High
Size of Reservoir	N.A.	95 neurons	$135-54$ neurons	98.92%/ 92.24%	Low/ Medium

tors. Modern integrated circuits (both analog and digital) are susceptible to a very wide range of failure mechanisms. Therefore, it is worthwhile to examine the robustness of a given LSM when implementing it using highly-scaled modern VLSI technologies for which process variations (e.g. variations in transistors and interconnect parameters) and manufacturing defects (e.g. stuck-at-0 and stuck-at-1 faults) may introduce unavoidable parameter fluctuations, and cause various levels of performance variation or even permanent failures [106, 107]. In addition, modern VLSI chips are prone to errors in operation, which may result from environmental effects (e.g. temperature variation and random power supply noises) and soft errors (e.g. single-event upsets due to cosmic rays and crosstalk noise), potentially causing transient errors and rendering a VLSI-based LSM to fail in numerous ways [108, 109].

Note that the above failure mechanisms may render rather different types of failure behavior. Catastrophic manufacturing defects may cause permanent failures of certain circuit blocks. Statistical manufacturing process variations may lead to increased circuit delay, hence producing timing errors under certain inputs. While many different inputs are applied to a given logic circuit block, the input dependency of timing errors may be viewed as adding random errors to the output of the circuit block. Environmental effects, in particular, power supply noise can lead to timing failures and hence errors in digital circuits. Since power supply noise has a significant random component, the resulting errors may be modeled as random both temporally and in terms of occurrence probability. Soft errors may lead to erroneous computations of certain output bits and also have

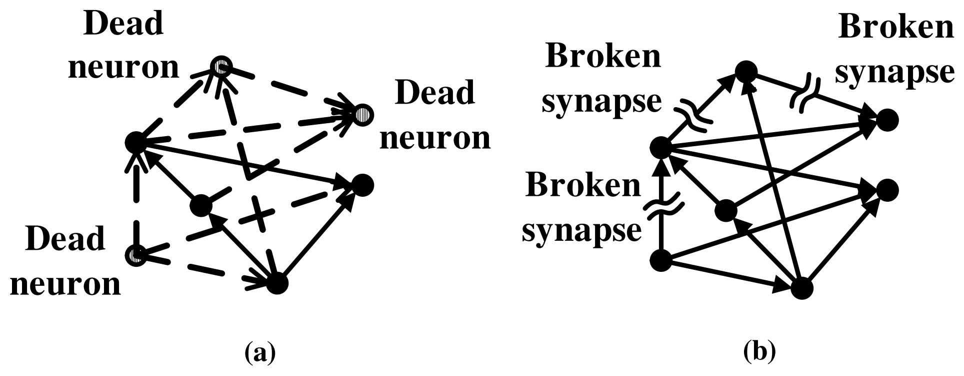

Figure 5.19: Modeling broken synapses and dead neurons in the LSM. (a): dead neurons; (b): broken synapses. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

a strong random component. These failures are highly process and design dependent and an accurate failure analysis is feasible only for a given the design of the targeted integrated circuit and the adopted manufacturing process. Therefore, given the scope of this work, we only model these failure mechanisms with certain abstraction and assess the general robustness of the proposed LSM rather its particular hardware implementations.

As mentioned above, the failures cited above have two main effects in a digital circuit: 1) catastrophic failures that cause certain circuit blocks to fail completely, and 2) random errors in terms of both occurrence and magnitude. First of all, we model the first effect as broken synapses and dead neurons. In the simulation, we assume catastrophic failures may result in permanent malfunction of certain synapses or neurons, that is, some of them fail to respond to the coming spikes and become completely nonfuctional (Fig. 5.19). This modeling approach effectively removes such broken synapses or dead neurons from the network. To provide an insight about how LSMs perform when subjected to random errors, we consider to introduce a random error probability for key arithmetic blocks (i.e. adders, shifters and comparators) in the network. Furthermore, once an error occurs, a normal distribution is used to model the amount of error produced relative to the correct value.

Similar to the experimental settings in Section 5.3, in this experimental study, five LSMs are

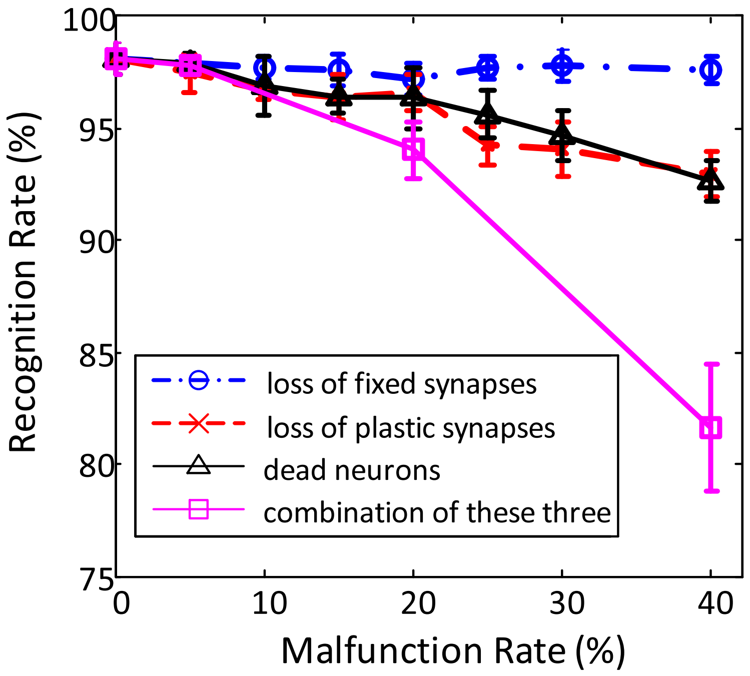

Figure 5.20: Performance degradation as a function of malfunction rates. The malfunction rates are the percentages of broken synapses or dead neurons. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

generated with random reservoir connections. For each LSM, we perform the 5-fold cross validation mentioned in Section 5.1.5 to attain the recognition rate at each targeted failure/error level. The five obtained recognition rates are averaged as the reported rate and the corresponding standard deviation (shown as the error bar with its length being $2 \times SD$) is plotted.

5.4.1 Catastrophic Failures

To acquire a quantitative understanding of robustness, we model the effect of catastrophic failures as causing the critical blocks (i.e. synapses and neurons) of the LSM to be non-functional. In the simulation, this effect is equivalent to removing a certain amount of synapses or neurons from the network.

Broken Synapses

After a certain number of broken synapses being deleted from the LSMs, we measure the recognition rates at different levels of severity of catastrophic failures and plot the results in Fig. 5.20. Note that even at a fairly large failure level (such as 20%), the LSM can still achieve pretty good performance (around 97%) under the cases where fixed or plastic synapses are broken, respectively. It implies that LSMs are robust to potential "broken synapses" caused by process variations and manufacturing defects. Furthermore, the classification performance is more sensitive to failures of plastic synapses than those of fixed synapses. This can be understood again by noting that the former play a key role in classification conducted by the linear readout neurons. Therefore, plastic synapses shall be implemented with more robust circuit-level techniques.

Dead Neurons

We show the recognition rates of the LSM with different percentages of dead reservoir neurons in Fig. 5.20. As can be observed, the recognition performance of the LSM is more sensitive to dead reservoir neurons than broken reservoir synapses. A possible explanation for this discrepancy is that reservoir neurons are fundamental processing/computing elements in the network. Broken reservoir synapses may not necessarily knock out any neurons from the reservoir while the existence of dead neurons certainly does. Hence, necessary steps of preventing a large number of neurons to fail are important for ensuring good performance. However, thanks to intrinsic resilience and redundancy presented in the LSM, the recognition rate is still about 96% even with 20% of neurons stopping functioning.

Combination of Broken Synapses and Dead Neurons

Furthermore, we look at the compound effect of having simultaneous broken synapses and dead neurons in Fig. 5.20. With 5% combined malfunction rate (5% of fixed and plastic synapses are broken and 5% of reservoir neurons stop functioning), only about 1% performance drop is observed. With 20% combined malfunction rate, the performance can still reach up to 94%.

In conclusion, it is clear that the catastrophic failures do have some impacts on performance.

Although the examined failure rates are fairly large, the learning performance does not degrade a lot, which reveals the good robustness of this neuromorphic system.

5.4.2 Random Errors

To perform this robustness study of the LSMs under random errors, we perturb outputs of crucial arithmetic blocks (i.e. adders, shifters and comparators) with errors. We assume that the error probability for each adder and shifter is 0.1 for simplicity. Once an error happens, a normal distribution is used to model the amount of error introduced relative to the correct value for adders and shifters. For each comparator, we consider the probability of generating erroneous output because a comparator only outputs “0” or “1”.

Error in Adding Operations

To consider a somewhat stressed error scenario, we assume all adders used to implement the neurons and synapses in the LSM are subjected to an error probability P_{err} of 10%. When an error occurs, a normal distribution is used to model the magnitude of error as described before. As the amount of injected error increases, we observe a significant performance degradation in Fig. 5.21. For example, the LSM largely fails when the amount of error is larger than 20%.

We further study how the network performance may depend on adding errors occurring in parts of the LSM. As shown in Fig. 5.21, when the reservoir is subjected to the error, the performance only degrades by no more than 2% within the considered range. Hence, the recognition performance appears to be insensitive to errors in the reservoir. In comparison, the performance is much more sensitive to errors in the readout as performance drops down to 86% in the worse case. This phenomenon may be understood by noting that additions are needed for both simulating the neurodynamics of the readout neurons and implementing the learning rule. The former ultimately determines the firing activity of a reservoir neuron which is a basis for interpreting recognition decision; while the latter is responsible for tuning the plastic synaptic weights according to the firing activities in the reservoir, playing a critical role in adjusting the corresponding hyperplanes implemented by the linear readout neurons. However, given that the LSM still performs well un-

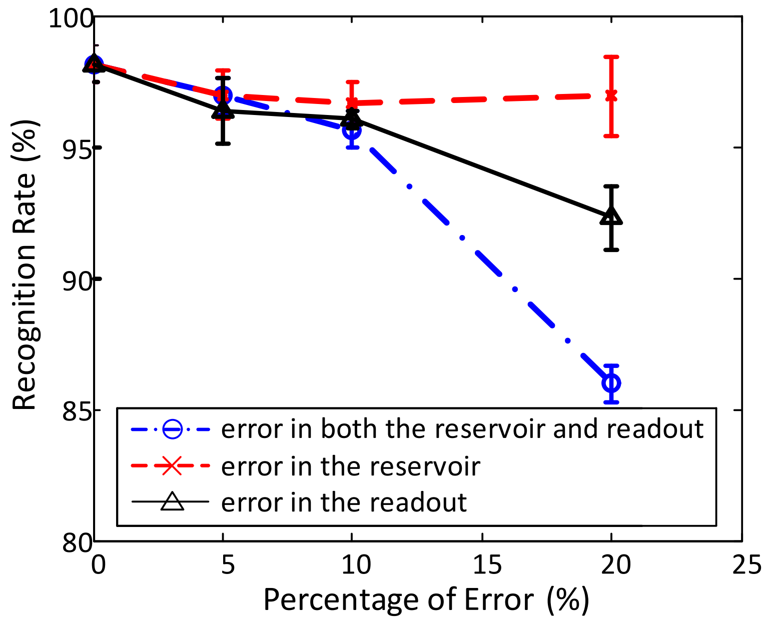

Figure 5.21: Performance degradation as a function of the amount of error in adders. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

der a large amount of error (e.g. 10%), a fairly noisy environment indeed, we do observe good robustness of the network even with respect to errors in the readout.

Error in Shifting Operations

In this work, shifters are used to simplify multiplication and division operations [10]. Similar to modeling output errors in adders, we introduce random errors to the outputs of the shifters in the LSM with the error probability of 0.1 and the error amount modeled by a normal distribution. The resulting performance as a function of the amount of shifting error is shown in Fig. 5.22. As the shifters suffer from more error, the performance degrades very gracefully. Even 20% of error, the performance degradation is less than 2%. Thus we can conclude that LSMs are insensitive to error from shifters. This argument can be further supported by Fig. 5.22 where we separately consider errors in the reservoir and readout. No significant performance penalty can be seen. It is evident that the network performance is less sensitive to errors in shifters compared with adders. This interesting phenomenon may be understood by noting that multiplication or division is only

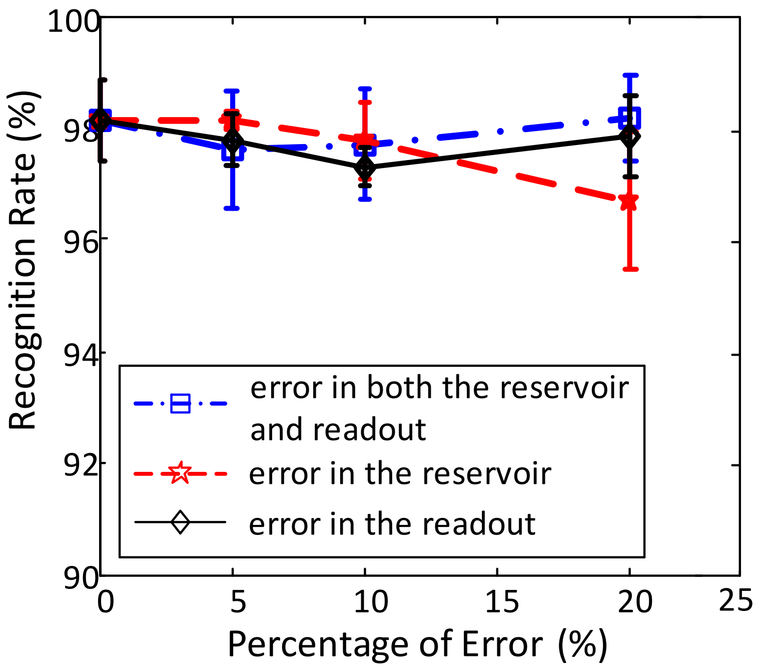

Figure 5.22: Performance degrades gracefully with an ascending amount of shifting error. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

used when calculating dynamics of neurons and synapses while addition is not only responsible for the computation of dynamics, but also the update of plastic synaptic weights.

Error in Comparison Operations

The way to model random errors in comparators is slightly different from the previous cases. Since comparators only output “0” or “1”, we consider the probability of generating the erroneous outputs. As shown in Fig. 5.23, the performances at three error probability levels (5%, 10% and 20%) are measured. Since the worst performance penalty is nearly 90%, we conclude that error in comparators does affect the performance. By further perturbing comparators in the reservoir and the readout (Fig. 5.23), it appears that the performance is sensitive to errors in both the reservoir and the readout. The reason might be that comparison is involved in determining the firing activities of neurons and the adaptation of plastic synaptic weights. Any error in comparison might give rise to unpredictable behaviors of neurons and synapses. For example, an error-prone comparator

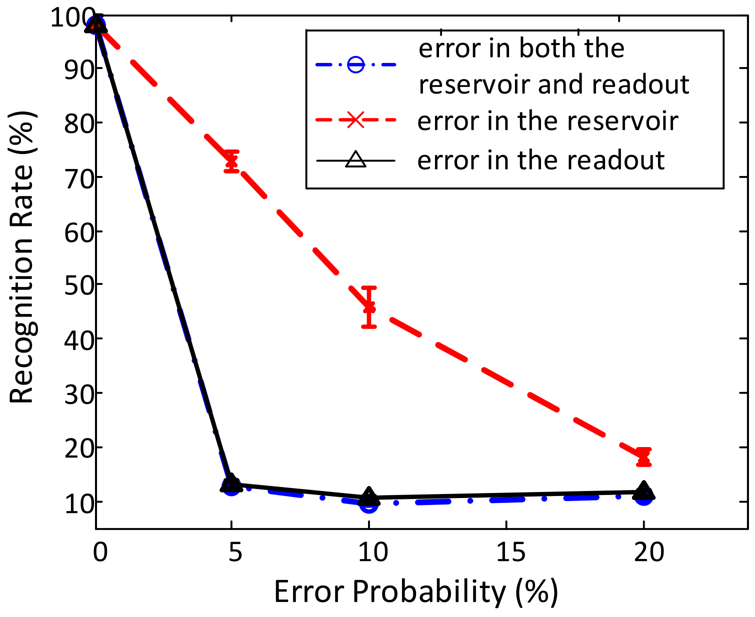

Figure 5.23: Performance drops dramatically with an increasing error probability. The impact of comparing error from different parts of the LSM is shown. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

can render a neuron fire a spike though its membrane voltage is less than the threshold. Similarly, an error-prone comparator in a plastic synapse might mistakenly initiate weight adaption with the learning rule being violated. Therefore, the LSM can be very sensitive to errors in comparison.

5.4.3 Summary of Robustness Study

Table 5.13 summarizes two types of catastrophic failure and three types of random error with their impacts on performance. In general, the reservoir is much more insensitive to failure and error compared to the readout, therefore the readout layer should be the main target of fault and noise tolerance for hardware implementation. Furthermore, error effects associated with comparators appear to have a high impact on performance. Hence, the comparators may be designed with a high-level of robustness. Besides, the recognition performance is sensitive to addition error that happens in the readout, and thus the adders in the readout should also be designed with a high-level of robustness. The other arithmetic operations have been shown to be less critical for the overall

Table 5.13: Typical types of failure and error with their impacts on performance. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

Failure/ Error	Type	Range	Worst degradation	Sensitivity
Catastrophic failures in reservoir	Dead neurons	0% – 40%	5.48%	Low/Medium
	Broken synapses	0% – 40%	0.96%	Low
Catastrophic failures in readout	Broken synapses	0% – 40%	5.2%	Low/Medium
Error in reservoir	Error in adders	0% – 20%	1.52%	Low
	Error in shifters	0% – 20%	1.44%	Low
	Error in comparators	0% – 20%	79.94%	High
Error in readout	Error in adders	0% – 20%	5.88%	Low/Medium
	Error in shifters	0% – 20%	0.84%	Low
	Error in comparators	0% – 20%	87.68%	High

performance and hence a less robust implementation may be explored to gain benefits in area and energy consumption. Generally speaking, the studied LSMs appear to be robust to various types of catastrophic failure and random error. This is very appealing and can be leveraged for efficient hardware implementation while maintaining a good level of robustness.

5.5 Performance Study of LSM on Other Subsets of TI46

Table 5.14: Design parameters of different levels of design complexity. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

Design Specifications	Type	Complexity Level 3	Complexity Level 2	Complexity Level 1
Calcium Level	Readout	14 bits	10 bits	10 bits
Membrane Voltage	Reservoir	16 bits	6 bits	6 bits
	Readout	16 bits	6 bits	6 bits
Synaptic Weight	Reservoir	10 bits	1 bit	1 bit
	Readout	10 bits	8 bits	8 bits
Size of Reservoir	N.A.	135 neurons	95 neurons	54 neurons

To provide a more complete understanding on the trade-offs between design cost and performance, we introduce two additional benchmarks, which are the second and third benchmark

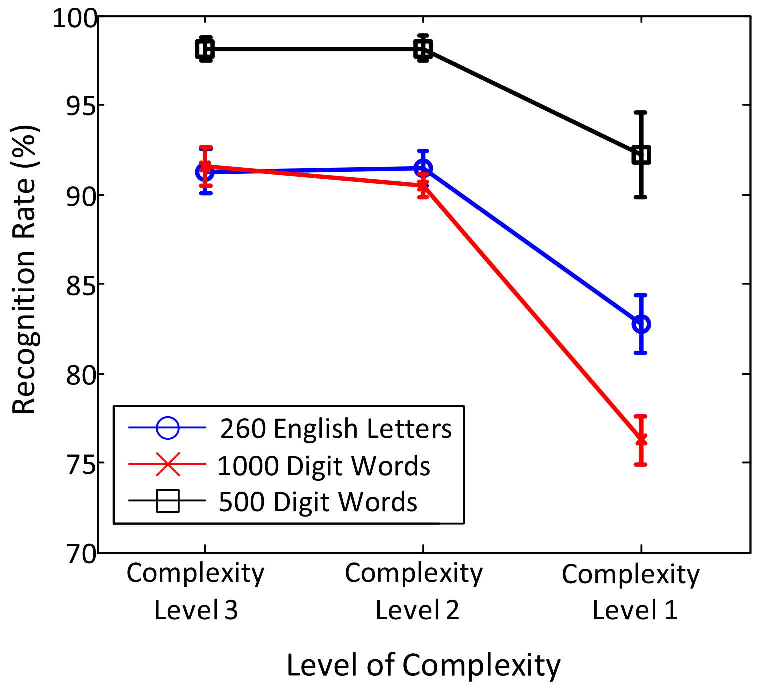

Figure 5.24: Classification performance of the LSM on the three adopted benchmarks decreases as a function of design complexity. A reasonably good performance can be attained with reduced complexity. Reprinted with permission from Yingyezhe Jin and Peng Li ©2016 Elsevier B.V.

described in Section 5.1.4, and three levels of implementation complexity with design parameters shown in Table 5.14. The speech signals of the second and third benchmark are preprocessed by the Lyon passive ear model [92] then encoded into 83 spike trains by the BSA algorithm [93, 15], and fed into a group of randomly selected neurons in the reservoir. And other experimental settings are the same as described in Section 4. We test the performance of the LSM on these two additional subsets of the TI46 speech corpus, and the recognition rates of the LSMs on the three different benchmarks adopted in this chapter are depicted in Fig. 5.24. In Table 5.14, the complexity level 3 is the original setting in [10] and the complexity level 2 is the suggested setting of this chapter (see Table 5.12). It is clear from Fig. 5.24 that the recognition rate degrades pretty gracefully as the design complexity decreases for all three benchmarks. This suggests that within a reasonably wide range, performance and design overhead can be rather nicely traded off with each other, providing a rather good flexibility in achieving the overall design objectives.

5.6 Summary and Discussions

This chapter presents a comprehensive performance and robustness study of bio-inspired digital liquid state machines for speech recognition. By examining a broad range of key network design parameters and using real-world meaningful benchmarks, we shed light on the relationship between design parameters and performance. We show that good performance can be maintained while reducing the resolutions and reservoir size, both of which have immediate impacts on hardware implementation overhead. To gain deep insights into the computational capability of LSMs, we adopt three theoretical measures to illustrate the relation between performance and the design parameters and the results generally agree with the simulated performance. By applying the theoretical measures, we also notice that multiple regimes can provide the LSM with sufficiently rich dynamics for separation of different input samples. To provide practical suggestions for future hardware implementation, we study the impacts of failure and error mechanisms introduced by process variations and environmental effects upon the recognition performance, showing that LSMs are fairly robust.

We have several main findings. First, in general, the implementation of the reservoir does not appear to be critical and require a high level of precision and robustness as long as it is capable of creating rich dynamics. One exception is the implementation of comparators which directly impact the firing activities of the reservoir. On the contrary, robust and accurate arithmetic units (comparators and adders especially) are desirable for the readout because they are critical parts of the LSM. These insights are particularly useful in practice as they offer insightful guidance for circuit implementation such that a good level of performance and robustness may be maintained while avoiding unnecessary overdesign.

6. CONCLUSION AND FUTURE WORK

6.1 Conclusion

This dissertation presents the architectures and training algorithms for embracing high performance and energy efficient deep spiking neuron networks. The issues of learning performance, efficiency and hardware overhead are addressed when designing architectures and algorithms. We summarize the major contributions of this dissertation as follows.

Training feedforward SNNs for reaching the comparable performance level of deep ANNs has been a grand challenge in the neuromorphic computing community because of the complex temporal dynamics and non-differentiable discrete spike activities of spiking neurons. In chapter 2, we present a hybrid macro/micro level backpropagation (HM2-BP) algorithm for training deep SNNs which operates directly on discrete spike events. In our HM2-BP, the spike-train level post-synaptic potentials (S-PSP) at the micro level exactly capture the spike timing information. And the rate-coded error is defined and efficiently computed and back-propagated across both the macro and micro levels. Furthermore, we propose a decoupled S-PSP model to assist gradient computation at the micro-level. Compared with the previous algorithms, our hybrid approach directly computes the error gradient of the rate-coded objective function with respect to the network tuning parameters. The best performances of both fully connected and convolutional SNNs are demonstrated over the static MNIST and dynamic N-MNIST datasets, outperforming the current state-of-the-art results of SNN training techniques. We also show that the proposed approach achieves better performance than those of traditional deep models when dealing with asynchronous spike streams.

While the reservoir of the standard LSM is usually fixed for relaxing the overall training difficulty, the fixed reservoir might not be an effective filter for specific applications, which provides the opportunity of optimization for potential learning performance boost. In chapter 3, we propose a novel activity-based probabilistic STDP (AP-STDP) rule as a promising self-organizing approach to construct plastic recurrent reservoirs. The issue of synaptic memory saturation is tackled by the proposed stop-learning mechanism controlled by the activity measure. The principle component analysis of the network dynamics is conducted to show that the proposed rule can generate an improved internal representation compared with all other studied STDP approaches. The AP-STDP achieves a learning performance boost at up to 2.7% at low synaptic weight resolutions. Furthermore, we propose a self-organizing LSM architecture with a hardware-optimized STDP for reservoir tuning and an online readout reconfiguration scheme for sparsity. The proposed architecture improves the average learning performance of a baseline LSM design by 2% while reducing its energy consumption by 25% with little extra overhead.

The dense connectivity of the readout presents a bottleneck for the improvement of overall efficiency of the resulted LSM based neural processor. In chapter 4, we propose a unifying calcium-modulated supervised STDP for both training and sparsification of readout synapses. For the first time, the improved learning performance and sparsity, which are naturally competing objectives, are achieved under an integrated algorithmic framework. The algorithm follows a two-step approach where the readout is trained for sparsity first by *CaS-S^{2}TDP* and then trained for improved learning performance by *CaL-S^{2}TDP* with a stop-learning mechanism. We have demonstrated that the proposed unifying algorithm outperforms a competitive non-STDP spike dependent algorithm by up to 2.7%. And it can prune out the readout synapses by up to 30% without inducing significant performance degradation.

The liquid state machine is a promising biologically inspired computational framework of SNNs, which exploits the computation capability of spiking neurons with minimal training effort. In chapter 5, we present a comprehensive performance and robustness study of liquid state machines for the purpose of guiding its hardware realization. The relation between a large space of design parameters and learning performance is studied with insightful theoretical measures to provide guidance for trade-offs in future hardware implementation of SNNs. We show that the hardware implementation overhead can be minimized by decreasing the resolutions and reservoir size while maintaining the good performance. From a circuit perspective, we investigate the impacts of the modeled process variation and noise schemes upon the learning performance, showing that LSMs are fairly error-resilient. With this systematic study, we are able to offer practical suggestions and guidances for future hardware realization of the energy efficient LSM based neural processor.

In conclusion, with all the contributions in this research, we expect this dissertation would help move the community forward towards high-performance spiking neural networks and energy efficient neuromorphic computing.

6.2 Future Work

There are multiple potential directions that lead to further exploration and expansion of the existing work. For example, we can combine the both feedforward and recurrent SNNs together for explore more interesting architectures. Besides, the proposed backpropagation approach can be extended to train recurrent spiking neuron networks. Instead of adopting the rate-coded error, temporal-coded error might be used to exploit the temporal coding capability of SNNs and facilitate the fast response and thus the classification decision can be made without seeing the entire spike sequence. Moreover, it would be interesting to study the corresponding hardware implementation of the HM2-BP with the comprehensive consideration of various design concerns. In this section, we focus on two main potential directions from 1) architectural and algorithmic and 2) hardware design perspectives.

6.2.1 Combination of feedforward and recurrent SNNs

It would be interesting to see the combination of the recurrent reservoir with multiple feedforward SNNs shown in Fig. 6.1 for targeting at more challenging benchmarks, such as the TI46 speech corpus with multiple speakers. The proposed backpropagation can be used to train the feedforward connections while the recurrent connections can be tuned by the proposed bio-inspired STDP algorithm. This would enable us to gain further insight of architectural and training exploration for SNNs.

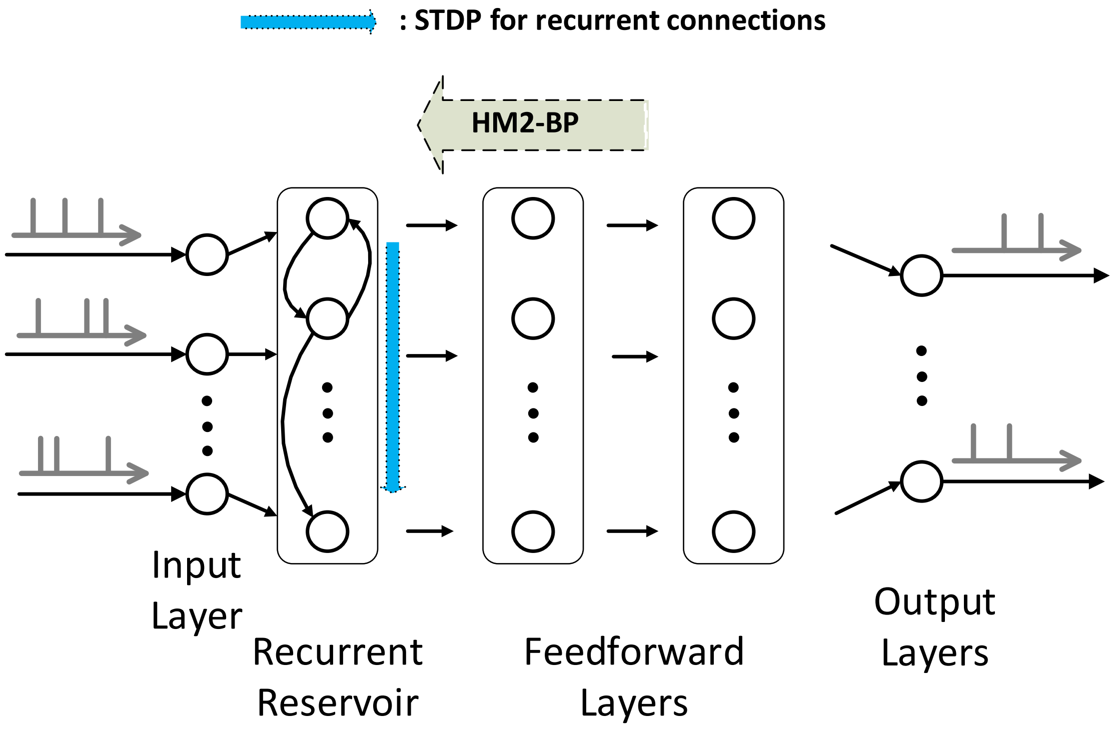

Figure 6.1: A deep SNN with a recurrent layer and multiple feedforward layers.

6.2.2 Extension of the HM2-BP algorithm

It is well known that the recurrent connection is difficult to train [22]. To the best knowledge of the author, tuning the recurrent reservoir using supervision has never been demonstrated before. Therefore, it would be interesting to extend the proposed HM2-BP for training the recurrent spiking neuron networks as shown in Fig. 6.2. The idea of backpropagation through time [34] might be a starting point by which we can be inspired to further propose backpropagation algorithm to take consideration of timing dependency generated by the recurrent connections. However, in order to fully address the complex temporal dependency resulted from the recurrent connectivity, the timing characteristics of the spiking neuron network shall be more carefully investigated. To better capture and tackle the timing dependency, the objective function might need to be changed accordingly to explicitly incorporate spike timings at the macro-level.

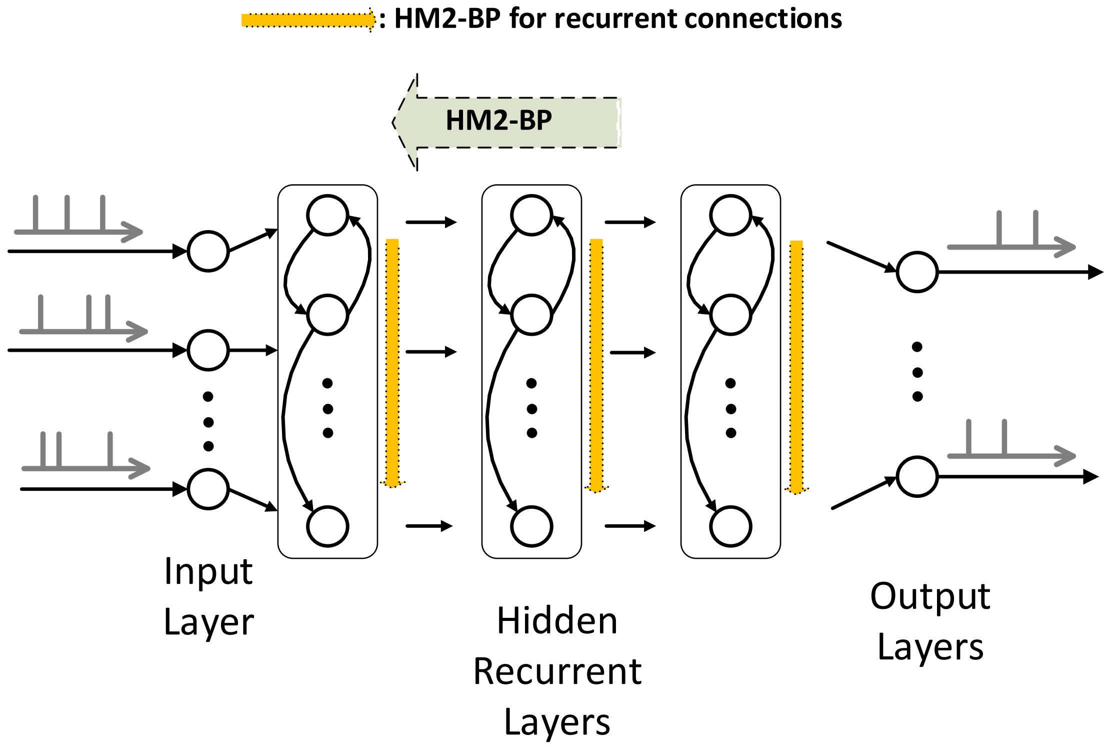

Figure 6.2: Extension of the HM2-BP for training deep SNNs with recurrent hidden layers.

6.2.3 Architecture and algorithm co-design for deep SNNs

Another immediate extension of the current work is to design the corresponding neuromorphic hardware with on-chip learning for backpropagation to enable the high performance SNN-based VLSI system. Although the competitive performance can be achieved by the proposed HM2-BP, it is might be fairly costly to be directly implemented in the hardware due to the complex computation involved in S-PSPs and high-resolution weights that are required in backpropagation. It can be a promising direction for co-designing the HM2-BP based training algorithms and the hardware architecture with on-chip training capability to obtain a good trade-off between the high performance, efficiency and hardware cost as illustrated in Fig. 6.3.

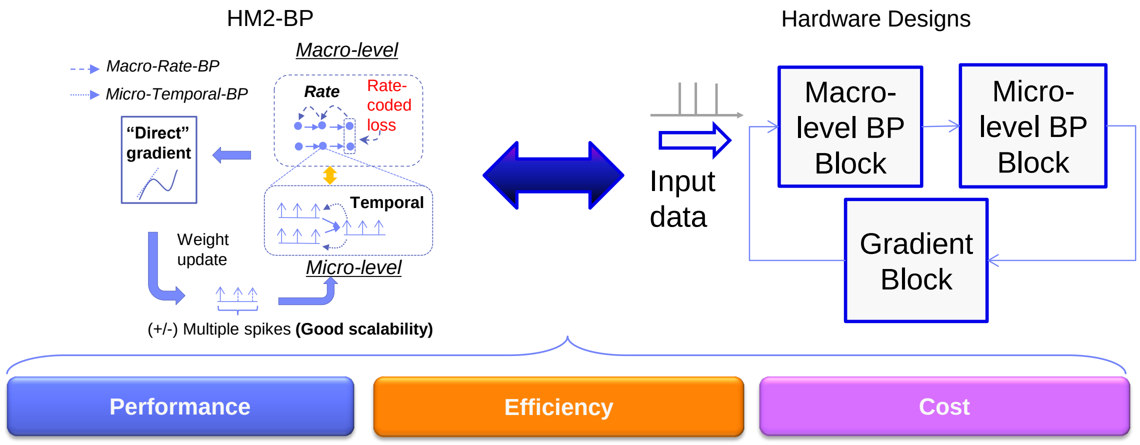

Figure 6.3: The co-design of the neuromorphic hardware and the HM2-BP algorithm for the system with online training capability.

REFERENCES

[1] L. A. Urry, M. L. Cain, S. A. Wasserman, P. V. Minorsky, R. Jackson, and J. B. Reece, *Campbell biology in focus*. Benjamin Cummings, 2015.

[2] E. M. Izhikevich and G. M. Edelman, "Large-scale model of mammalian thalamocortical systems," *Proceedings of the national academy of sciences*, vol. 105, no. 9, pp. 3593–3598, 2008.

[3] W. Maass, "Networks of spiking neurons: the third generation of neural network models," *Neural networks*, vol. 10, no. 9, pp. 1659–1671, 1997.

[4] G. Indiveri, E. Chicca, and R. Douglas, "A vlsi array of low-power spiking neurons and bistable synapses with spike-timing dependent plasticity," *IEEE transactions on neural networks*, vol. 17, no. 1, pp. 211–221, 2006.

[5] S. Mitra, S. Fusi, and G. Indiveri, "Real-time classification of complex patterns using spike-based learning in neuromorphic vlsi," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 3, no. 1, pp. 32–42, 2009.

[6] 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 *Custom Integrated Circuits Conference (CICC), 2011 IEEE*, pp. 1–4, IEEE, 2011.

[7] 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.

[8] 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.

[9] 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.

[10] 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, pp. 2635–2649, Nov. 2015.

[11] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting," *Neural computation*, vol. 22, no. 2, pp. 467–510, 2010.

[12] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, "Span: Spike pattern association neuron for learning spatio-temporal spike patterns," *International Journal of Neural Systems*, vol. 22, no. 04, p. 1250012, 2012.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

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

[15] 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.

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

[17] Q. Wang, Y. Jin, and P. Li, "General-purpose LSM learning processor architecture and theoretically guided design space exploration," in *Biomedical Circuits and Systems Conference (BioCAS), 2015 IEEE*, pp. 1–4, IEEE, 2015.

[18] S. Roy, A. Banerjee, and A. Basu, "Liquid state machine with dendritically enhanced readout for low-power, neuromorphic VLSI implementations," *Biomedical Circuits and Systems, IEEE Transactions on*, vol. 8, no. 5, pp. 681–695, 2014.

[19] 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)*, pp. 361–364, May 2016.

[20] E. Goodman and D. Ventura, "Effectively using recurrently-connected spiking neural networks," in *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, vol. 3, pp. 1542–1547, IEEE, 2005.

[21] D. Norton and D. Ventura, "Improving liquid state machines through iterative refinement of the reservoir," *Neurocomputing*, vol. 73, no. 16, pp. 2893–2904, 2010.

[22] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *Neural Networks, IEEE Transactions on*, vol. 5, no. 2, pp. 157–166, 1994.

[23] G.-q. Bi and M.-m. Poo, "Synaptic modification by correlated activity: Hebb's postulate revisited," *Annual review of neuroscience*, vol. 24, no. 1, pp. 139–166, 2001.

[24] A. Morrison, M. Diesmann, and W. Gerstner, "Phenomenological models of synaptic plasticity based on spike timing," *Biological cybernetics*, vol. 98, no. 6, pp. 459–478, 2008.

[25] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers in neuroscience*, vol. 10, p. 508, 2016.

[26] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *arXiv preprint arXiv:1706.02609*, 2017.

[27] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, “Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing,” in *Neural Networks (IJCNN), 2015 International Joint Conference on*, pp. 1–8, IEEE, 2015.

[28] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, “Backpropagation for energy-efficient neuromorphic computing,” in *Advances in Neural Information Processing Systems*, pp. 1117–1125, 2015.

[29] E. Hunsberger and C. Eliasmith, “Spiking deep networks with lif neurons,” *arXiv preprint arXiv:1510.08829*, 2015.

[30] P. O’Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer, “Real-time classification and sensor fusion with a spiking deep belief network,” *Frontiers in neuroscience*, vol. 7, p. 178, 2013.

[31] P. O’Connor and M. Welling, “Deep spiking networks,” *arXiv preprint arXiv:1602.08323*, 2016.

[32] B. Rueckauer, Y. Hu, I.-A. Lungu, M. Pfeiffer, and S.-C. Liu, “Conversion of continuous-valued deep networks to efficient event-driven networks for image classification,” *Frontiers in neuroscience*, vol. 11, p. 682, 2017.

[33] S. M. Bohte, J. N. Kok, and H. La Poutre, “Error-backpropagation in temporally encoded networks of spiking neurons,” *Neurocomputing*, vol. 48, no. 1-4, pp. 17–37, 2002.

[34] P. J. Werbos, “Backpropagation through time: what it does and how to do it,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[36] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, “Converting static image datasets to spiking neuromorphic datasets using saccades,” *Frontiers in neuroscience*, vol. 9, p. 437, 2015.

[37] D. Norton and D. Ventura, “Preparing more effective liquid state machines using hebbian learning,” in *Neural Networks, 2006. IJCNN’06. International Joint Conference on*, pp. 4243–4248, IEEE, 2006.

[38] H. Paugam-Moisy, R. Martinez, and S. Bengio, “Delay learning and polychronization for reservoir computing,” *Neurocomputing*, vol. 71, no. 7, pp. 1143–1158, 2008.

[39] A. Lazar, G. Pipa, and J. Triesch, “Sorn: a self-organizing recurrent neural network,” *Frontiers in computational neuroscience*, vol. 3, 2009.

[40] S. V. Notley and A. Grüning, “Improved spike-timed mappings using a tri-phasic spike timing-dependent plasticity rule,” in *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pp. 1–6, IEEE, 2012.

[41] F. Xue, Z. Hou, and X. Li, “Computational capability of liquid state machines with spike-timing-dependent plasticity,” *Neurocomputing*, vol. 122, pp. 324–329, 2013.

[42] J. Yin, Y. Meng, and Y. Jin, “A developmental approach to structural self-organization in reservoir computing,” *Autonomous Mental Development, IEEE Transactions on*, vol. 4, no. 4, pp. 273–289, 2012.

[43] 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.

[44] N. Kasabov, N. M. Scott, E. Tu, S. Marks, N. Sengupta, E. Capecci, M. Othman, M. G. Doborjeh, N. Murli, R. Hartono, *et al.*, “Evolving spatio-temporal data machines based on the neucube neuromorphic framework: design methodology and selected applications,” *Neural Networks*, vol. 78, pp. 1–14, 2016.

[45] C. C. Petersen, R. C. Malenka, R. A. Nicoll, and J. J. Hopfield, “All-or-none potentiation at ca3-ca1 synapses,” *Proceedings of the National Academy of Sciences*, vol. 95, no. 8, pp. 4732–4737, 1998.

[46] D. H. O'Connor, G. M. Wittenberg, and S. S.-H. Wang, "Graded bidirectional synaptic plasticity is composed of switch-like unitary events," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 27, pp. 9679–9684, 2005.

[47] J. M. Brader, W. Senn, and S. Fusi, "Learning real-world stimuli in a neural network with spike-driven synaptic dynamics," *Neural computation*, vol. 19, no. 11, pp. 2881–2912, 2007.

[48] B. Schrauwen, M. DHaene, D. Verstraeten, and J. V. Campenhout, "Compact hardware liquid state machines on fpga for real-time speech recognition," *Neural Networks*, vol. 21, no. 2, pp. 511–523, 2008.

[49] 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 *Internatioal Symposium of Circuits and Systems (ISCAS), 2016 IEEE*, pp. 361–364, IEEE, 2016.

[50] J.-P. Pfister, T. Toyoizumi, D. Barber, and W. Gerstner, "Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning," *Neural computation*, vol. 18, no. 6, pp. 1318–1348, 2006.

[51] J.-M. P. Franosch, S. Urban, and J. L. van Hemmen, "Supervised spike-timing-dependent plasticity: A spatiotemporal neuronal learning rule for function approximation and decisions," *Neural computation*, vol. 25, no. 12, pp. 3113–3130, 2013.

[52] N. Caporale and Y. Dan, "Spike Timing-Dependent Plasticity: a hebbian learning rule," *Annu. Rev. Neurosci.*, vol. 31, pp. 25–46, 2008.

[53] S. Song, K. D. Miller, and L. F. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[54] 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.

[55] R. Gütig and H. Sompolinsky, “The tempotron: a neuron that learns spike timing–based decisions,” *Nature neuroscience*, vol. 9, no. 3, pp. 420–428, 2006.

[56] J. M. Brader, W. Senn, and S. Fusi, “Learning real-world stimuli in a neural network with spike-driven synaptic dynamics,” *Neural computation*, vol. 19, no. 11, pp. 2881–2912, 2007.

[57] S. Ghosh-Dastidar and H. Adeli, “A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection,” *Neural Networks*, vol. 22, no. 10, pp. 1419–1431, 2009.

[58] R. V. Florian, “The chronotron: a neuron that learns to fire temporally precise spike patterns,” *PloS one*, vol. 7, no. 8, p. e40233, 2012.

[59] M. Rubinov, O. Sporns, J.-P. Thivierge, and M. Breakspear, “Neurobiologically realistic determinants of self-organized criticality in networks of spiking neurons,” *PLoS computational biology*, vol. 7, no. 6, p. e1002038, 2011.

[60] H. Hazan and L. M. Manevitz, “Topological constraints and robustness in liquid state machines,” *Expert Systems With Applications*, vol. 39, no. 2, pp. 1597–1606, 2012.

[61] B. Glackin, T. M. McGinnity, L. P. Maguire, Q. Wu, and A. Belatreche, “A novel approach for the implementation of large scale spiking neural networks on fpga hardware,” in *Computational Intelligence and Bioinspired Systems*, pp. 552–563, Springer, 2005.

[62] J. Schemmel, A. Grubl, K. Meier, and E. Mueller, “Implementing synaptic plasticity in a VLSI spiking neural network model,” in *Neural Networks, 2006. IJCNN’06. International Joint Conference on*, pp. 1–6, IEEE, 2006.

[63] A. Afifi, A. Ayatollahi, and F. Raissi, “Implementation of biologically plausible spiking neural network models on the memristor crossbar-based cmos/nano circuits,” in *Circuit Theory and Design, 2009. ECCTD 2009. European Conference on*, pp. 563–566, IEEE, 2009.

[64] Y. Kim, Y. Zhang, and P. Li, “A digital neuromorphic VLSI architecture with memristor crossbar synaptic array for machine learning,” in *SOC Conference (SOCC), 2012 IEEE International*, pp. 328–333, IEEE, 2012.

[65] W. Maass, T. Natschlager, and H. Markram, “A model for real-time computation in generic neural microcircuits,” *Advances in neural information processing systems*, pp. 229–236, 2003.

[66] H. Burgsteiner, M. Kröll, A. Leopold, and G. Steinbauer, “Movement prediction from real-world images using a liquid state machine,” *Applied Intelligence*, vol. 26, no. 2, pp. 99–109, 2007.

[67] A. Ghani, T. M. McGinnity, L. P. Maguire, and J. Harkin, “Neuro-inspired speech recognition with recurrent spiking neurons,” in *Artificial Neural Networks-ICANN 2008*, pp. 513–522, Springer, 2008.

[68] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, ACM, 2008.

[69] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

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

[71] W. Gerstner and W. M. Kistler, *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002.

[72] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[73] P. Lichtsteiner, C. Posch, and T. Delbruck, “A 128 $\times$128 120 db 15μs latency asynchronous temporal contrast vision sensor,” *IEEE journal of solid-state circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[74] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[75] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International Conference on Machine Learning*, pp. 1058–1066, 2013.

[76] P. Y. Simard, D. Steinkraus, J. C. Platt, *et al.*, "Best practices for convolutional neural networks applied to visual document analysis.," in *ICDAR*, vol. 3, pp. 958–962, 2003.

[77] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased lstm: Accelerating recurrent network training for long or event-based sequences," in *Advances in Neural Information Processing Systems*, pp. 3882–3890, 2016.

[78] D. Neil and S.-C. Liu, "Effective sensor fusion with event-based sensors and deep network architectures," in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, pp. 2282–2285, IEEE, 2016.

[79] G. K. Cohen, G. Orchard, S.-H. Leng, J. Tapson, R. B. Benosman, and A. Van Schaik, "Skimming digits: neuromorphic classification of spike-encoded images," *Frontiers in neuroscience*, vol. 10, p. 184, 2016.

[80] L. D. Consortium, *The TI46 Speech Corpus*, 2014 (accessed June 10, 2014). `http://catalog.ldc.upenn.edu/LDC93S9`.

[81] E. M. Izhikevich, J. A. Gally, and G. M. Edelman, "Spike-timing dynamics of neuronal groups," *Cerebral cortex*, vol. 14, no. 8, pp. 933–944, 2004.

[82] P. A. Appleby and T. A. Elliott, "Synaptic and temporal ensemble interpretation of spike-timing-dependent plasticity," *Neural computation*, vol. 17, no. 11, pp. 2316–2336, 2005.

[83] 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 *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pp. 1–10, IEEE, 2013.

[84] D. J. Amit and S. Fusi, "Constraints on learning in dynamic synapses," *Network: Computation in Neural Systems*, vol. 3, no. 4, pp. 443–464, 1992.

[85] D. J. Amit and S. Fusi, "Learning in neural networks with material synapses," *Neural Computation*, vol. 6, no. 5, pp. 957–982, 1994.

[86] R. F. Lyon, "A computational model of filtering, detection, and compression in the cochlea," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, vol. 7, pp. 1282–1285, IEEE, 1982.

[87] B. Schrauwen and J. Van Campenhout, "Bsa, a fast and accurate spike train encoding scheme," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 4, pp. 2825–2830, IEEE Piscataway, NJ, 2003.

[88] Y. Jin and P. Li, "AP-STDP: A novel self-organizing mechanism for efficient reservoir computing," in *Neural Networks, 2015. IJCNN'15. Proceedings. 2015 IEEE International Joint Conference on*, vol. 3, pp. 1–4, IEEE, 2015.

[89] A. Cassidy, A. G. Andreou, and J. Georgiou, "A combinational digital logic approach to STDP," in *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pp. 673–676, IEEE, 2011.

[90] Y. Jin, Y. Liu, and P. Li, "Sso-lsm: A sparse and self-organizing architecture for liquid state machine based neural processors," in *Nanoscale Architectures (NANOARCH), 2016 IEEE/ACM International Symposium on*, pp. 55–60, IEEE, 2016.

[91] G. R. Doddington and T. B. Schalk, "Computers: Speech recognition: Turning theory to practice: New ics have brought the requisite computer power to speech technology; an evaluation of equipment shows where it stands today.," *Spectrum, IEEE*, vol. 18, no. 9, pp. 26–32, 1981.

[92] R. F. Lyon, "A computational model of filtering, detection, and compression in the cochlea," in *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'82.*, vol. 7, pp. 1282–1285, IEEE, 1982.

[93] B. Schrauwen and J. Van Campenhout, "Bsa, a fast and accurate spike train encoding scheme," in *Proceedings of the international joint conference on neural networks*, vol. 4, pp. 2825–2830, IEEE Piscataway, NJ, 2003.

[94] T. M. Cover, "Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition," *Electronic Computers, IEEE Transactions on*, no. 3, pp. 326–334, 1965.

[95] D. O. Hebb, *The organization of behavior: A neuropsychological theory*. Wiley, 1949.

[96] A. Mohemmed, S. Schliebs, S. Matsuda, and N. Kasabov, "Span: Spike pattern association neuron for learning spatio-temporal spike patterns," *International journal of neural systems*, vol. 22, no. 04, 2012.

[97] W. Maass, T. Natschläger, and H. Markram, "Fading memory and kernel properties of generic cortical microcircuit models," *Journal of Physiology-Paris*, vol. 98, no. 4, pp. 315–330, 2004.

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

[99] 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.

[100] 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.

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

[102] S. A. Kauffman, *The origins of order: Self-organization and selection in evolution*. Oxford university press, 1993.

[103] B. Luque and R. V. Solé, "Lyapunov exponents in random boolean networks," *Physica A: Statistical Mechanics and its Applications*, vol. 284, no. 1, pp. 33–45, 2000.

[104] A. A. Faisal, L. P. Selen, and D. M. Wolpert, "Noise in the nervous system," *Nature Reviews Neuroscience*, vol. 9, no. 4, pp. 292–303, 2008.

[105] B. Schrauwen, L. Büsing, and R. Legenstein, "On computational power and the order-chaos phase transition in reservoir computing," in *22nd Annual conference on Neural Information Processing Systems (NIPS 2008)*, vol. 21, pp. 1425–1432, NIPS Foundation, 2009.

[106] P. Li, F. Liu, X. Li, L. T. Pileggi, and S. R. Nassif, "Modeling interconnect variability using efficient parametric model order reduction," in *Proceedings of the conference on Design, Automation and Test in Europe-Volume 2*, pp. 958–963, IEEE Computer Society, 2005.

[107] M. Orshansky, S. Nassif, and D. Boning, *Design for manufacturability and statistical design: a constructive approach*. Springer Science & Business Media, 2007.

[108] T. Karnik and P. Hazucha, "Characterization of soft errors caused by single event upsets in cmos processes," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, pp. 128–143, April 2004.

[109] H. Chang and S. S. Sapatnekar, "Statistical timing analysis under spatial correlations," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 24, no. 9, pp. 1467–1482, 2005.

APPENDIX A

GRADIENT COMPUTATION FOR THE OUTPUT LAYER WITH LATERAL INHIBITION

Without loss of generality, it is assumed that lateral inhibition exists between every pair of two output neurons. It is also assumed that the weights for lateral inhibition are all fixed at the constant value of w_0. As shown in Fig. A-1, the total post-synaptic potential (T-PSP) a_i^m for the neuron i at the output layer m can be unwrapped as

$$a_i^m = \sum_{j=1}^{r^{m-1}} w_{ij}\; e_{i|j}^m + \sum_{l \neq i}^{r^m} w_0\; e_{i|l}^m,$$

where the second term describing the lateral inhibition effects between neurons in the same output layer.

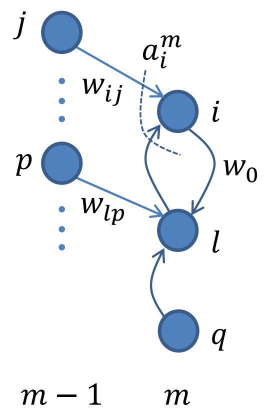

Figure A-1: The output layer with lateral inhibition.

The derivative of a_i^m with respect to w_{ij} is

$$\begin{aligned} \frac{\partial a_i^m}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}} \left(\sum_{j=1}^{r^{m-1}} w_{ij}\; e_{i|j}^m \right) + \frac{\partial}{\partial w_{ij}} \left(\sum_{l \neq i}^{r^m} w_0\; e_{i|l}^m \right) \\ &= e_{i|j}^m \left(1 + \frac{1}{\nu} \sum_{h=1}^{r^{m-1}} w_{ih} \frac{\partial e_{i|h}^m}{\partial o_i^m} \right) + \sum_{l \neq i}^{r^m} w_0 \frac{\partial e_{i|l}^m}{\partial o_l^m} g'(a_l^m) \frac{\partial a_l^m}{\partial w_{ij}}. \end{aligned} \tag{A-1}$$

Similarly for the neuron l of the output layer m, a_l^m is given as

$$a_l^m = \sum_{p=1}^{r^{m-1}} w_{lp}\; e_{l|p}^m + \sum_{q \neq l}^{r^m} w_0\; e_{l|q}^m.$$

Therefore, $\frac{\partial a_l^m}{\partial w_{ij}}$ is

$$\frac{\partial a_l^m}{\partial w_{ij}} = w_0 \frac{\partial e_{l|i}^m}{\partial o_i^m} g'(a_i^m) \frac{\partial a_i^m}{\partial w_{ij}}. \tag{A-2}$$

Plugging (A-2) back into (A-1) and solving for $\frac{\partial a_i^m}{\partial w_{ij}}$ leads to

$$\frac{\partial a_i^m}{\partial w_{ij}} = \gamma \ e_{i|j}^m \left(1 + \frac{1}{\nu} \sum_{h=1}^{r^{m-1}} w_{ih} \frac{\partial e_{i|h}^m}{\partial o_i^m}\right), \tag{A-3}$$

where

$$\gamma = \frac{1}{1 - \frac{w_0^2}{\nu^2} \sum_{l \neq i}^{r^m} \frac{\partial e_{i|l}^m}{\partial o_l^m} \frac{\partial e_{l|i}^m}{\partial o_i^m}}.$$

(A-3) is identical to (2.17) except that the factor γ is introduced to capture the effect of lateral inhibition.

For the output layer with lateral inhibition, the term $\frac{\partial E}{\partial a_i^m}$ of the macro-level backpropagation defined in (2.14) is the same as the one without lateral inhibition.