TEMPORAL CODING AND LEARNING IN SPIKING NEURAL NETWORKS

YU QIANG

NATIONAL UNIVERSITY OF SINGAPORE

2014

TEMPORAL CODING AND LEARNING IN SPIKING NEURAL NETWORKS

YU QIANG

(B.Eng., HARBIN INSTITUTE OF TECHNOLOGY)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2014

DECLARATION

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Yu Qiang

YU Qiang

31 July 2014

Acknowledgements

Looking back to my time as a PhD student, I would say it is challenging but exciting. Based on my experience, learning is important over education, especially for being an independent researcher. The PhD career is full of difficulties and challenges. To overcome these, fortunately, I received valuable helps from others. Therefore, I would like to take this opportunity to thank those who gave me supports and guidance during my hard times.

I would like to take this time to thank National University of Singapore (NUS) and Institute for Infocomm Research (I2R) for all of the funding they were able to provide to me in order to make this thesis possible.

The first person I would like to thank is my PhD supervisor, Associate Professor TAN Kay Chen, for introducing me to the front-edge research area of theoretical neuroscience. I remember at the beginning of my study when I was frustrated about those unexpected negative results, he encouraged me with kindness but not blame. He said “this is normal and this is what a ‘research’ is!”. Besides, he also helped me to get used to the life in the university, which is the basis for a better academic life. I learned much from him, not only skills for research, but also other skills for being a mature man. Thanks for his encouragement, valuable supervision and great patience.

Another important person I would like to thank is Dr. TANG Huajin, for his professional guidance in my research. His motivation and advice helped me a lot. He always puts the student’s work to high priority. Whenever I walked to his door for a discussion, he would stop his work and turn around to discuss the results. For every manuscript I sent to him, he edited it sentence by sentence, and taught me how to write a scientific paper with proper English.

I would also like to thank Professor LI Haizhou, Dr. YU Haoyong, ZHAO Bo and Jonathan Dennis for their valuable ideas during our cooperations. I would also like to express my gratitude to Associate Professor Abdullah Al Mamun and Assistant Professor Shih-Cheng YEN for their suggestions during my qualification exam, and for taking time to read my work carefully.

It was also a pleasure to work with all the people in the lab. My great thanks also goes to my seniors who shared their experience with me: Shim Vui Ann, Tan Chin Hiong, Cheu Eng Yeow, Hu Jun, Yu Jiali, Yuan Miaolong, Tian bo and Shi Ji Yu. I would like to thank people who make my university life memorable and enjoyable: Gee Sen Bong, Lim Pin, Arrchana, Willson, Qiu Xin, Zhang Chong and Sim Kuan. I would also like to express my gratitude to the lab officers, HengWei and Sara, for their continuous assistance in the Control and Simulation lab.

Last but not least, thanks to my family for their selfless love, patience and understanding they had for me throughout my PhD study. This thesis would not be possible without the ensemble of these causes.

YU Qiang

30/July/2014

Contents

Summary

Neurons in the nervous systems transmit information through action potentials (or called as spikes). It is still mysterious that how neurons with spiking features give rise to powerful cognitive functions of the brain. This thesis presents detailed investigation on information processing and cognitive computing in spiking neural networks (SNNs), trying to reveal and utilize mechanisms how the biological systems might operate. Temporal coding and learning are two major concerns in SNNs, with coding describing how information is carried by spikes and with learning presenting how neurons learn the spike patterns. The focus of this thesis varies from a neuronal level to a system level, including topics of spike-based learning in single and multilayer neural networks, sensory coding, system modeling, as well as applied development of visual and auditory processing systems. The temporal learning rules proposed in this thesis show possible ways to utilize spiking neurons to process spike patterns. The systems consisting of spiking neurons are successfully applied to different cognitive tasks such as item recognition, sequence recognition and memory.

Firstly, a consistent system considering both the temporal coding and learning is preliminarily developed to perform various recognition tasks. The whole system contains three basic functional parts: encoding, learning and readout. It shows that such a network of spiking neurons under a temporal framework can effectively and efficiently perform various classification tasks. The results suggest that the temporal learning rule combined with a proper encoding method can provide basic classification abilities of spiking neurons on different classification tasks. This system is successfully applied to learning patterns of either discrete values or continuous values. This integrated system also provides a general structure that could be flexibly extended or modified according to various requirements, as long as the basic functional parts inspired from the biology do not change.

Motivated by recent findings in biological systems, a more complex system is constructed in a feedforward structure to process real-world stimuli from a view point of rapid computation. The external stimuli are sparsely represented after the encoding structure, and the representations have some properties of selectivity and invariance. With a proper encoding scheme, the SNNs can be applied to both visual and auditory processing. This system is important in the light of recent trends in combining both the coding and learning in a systematic level to perform cognitive computations.

Then, a new temporal learning rule, named as the precise-spike-driven (PSD) synaptic plasticity rule, is developed for learning hetero-association of spatiotemporal spike patterns. Various properties of the PSD rule are investigated through an extensive experimental analysis. The PSD rule is advantageous in that it is not limited to performing classification, but it is also able to memorize patterns by firing desired spikes at precise time. The PSD rule is efficient, simple, and yet biologically plausible. The PSD rule is then applied in a spiking neural network system for sequence recognition. It shows that different functional subsystems can consistently cooperate within a temporal framework for detecting and recognizing a specific sequence. The results indicate that different spiking neural networks can be combined together as long as a proper coding scheme is used for the communications between each other.

Finally, temporal learning rules in multilayer spiking neural networks are investigated. As extensions of single-layer learning rules, the multilayer PSD rule (MutPSD) and multilayer tempotron rule (MutTmptr) are developed. The multilayer learning is fulfilled through the construction of causal connections. Correlated neurons are connected through fine tuned weights. The MutTmptr rule converges faster, while the MutPSD rule gives better generalization ability. The proposed multilayer rules provide an efficient and biologically plausible mechanism, describing how synapses in the multilayer networks are adjusted to facilitate the learning.

List of Tables

List of Figures

Chapter 1

Introduction

Since the emergence of the first digital computer, people are set free from heavy computing works. Computers can process a large amount of data with high precision and speed. However, compared to the brain, the computer still cannot approach a comparable performance considering cognitive functions such as perception, recognition and memory. For example, it is easy for human to recognize the face of a person, read papers and communicate with others, but hard for computers. Mechanisms that utilized by the brain for such powerful cognitive functions still remain unclear. Neural networks are developed for providing a brain-like information processing and cognitive computing. Theoretical analysis on neural networks could offer a key approach to reveal the secret of the brain. The subsequent sections provide detailed background information, as well as the objectives and the challenges of this thesis.

1.1 Background

The computational power of the brain has attracted many researchers to reveal its mystery in order to understand how it works and to design human-like intelligent systems. The human brain is constructed with around 100 billion highly interconnected neurons. These neurons transmit information between each other to perform cognitive functions. Modeling neural networks facilitates investigation of information processing and cognitive computing in the brain from a mathematical point of view. Artificial neural networks (ANNs), or simply called neural networks, are the earliest work for modeling the computational ability of the brain. The research on ANNs has achieved a great deal in both theories and engineering applications. Typically, an ANN is constructed with neurons which have real-valued inputs and outputs.

However, biological neurons in the brain utilize spikes (or called as action potentials) for information transmission between each other. This phenomenon of the 'spiking' nature of neurons has been known since the first experiments conducted by Adrian in the 1920s [1]. Neurons will send out short pulses of energy (spikes) as signals, if they have received enough input from other neurons. Based on this mechanism, spiking neurons are developed with a same capability of processing spikes as biological neurons. Thus, spiking neural networks (SNNs) are more biologically plausible than ANNs since the concept of spikes, rather than real values, is considered in the computation. SNNs are widely studied in recent years, but questions of how information is represented by spikes and how the neurons process these spikes are still unclear. These two questions demand further studies on neural coding and learning in SNNs.

Spikes are believed to be the principal feature in the information processing of neural systems, though the neural coding mechanism remains unclear. In 1920s, Adrian also found that sensory neurons fire spikes at a rate monotonically increasing with the intensity of stimulus. This observation led to the widespread adoption of the hypothesis of a rate coding, where neurons communicate purely through their firing rates. Recently, an increasing body of evidence shows that the precise timing of individual spikes also plays an important role [2]. This finding supports the hypothesis of a temporal coding, where the precise timing of spikes, rather than the rate, is used for encoding information. Within a 'temporal coding' framework, temporal learning describes how neurons process precise-timing spikes. Further research on temporal coding and temporal learning would provide a better understanding of the biological systems, and would also explore potential abilities of SNNs for information processing and cognitive computing. Moreover, beyond independently studying the temporal coding and learning, it would be more important and useful to consider both in a consistent system.

1.2 Spiking Neurons

The rough concept of how neurons work is understood: neurons send out short pulses of electrical energy as signals, if they have received enough of these themselves. This principal mechanism has been modeled into various mathematical models for computer use. These models are built under the inspiration of how real neurons work in the brain.

1.2.1 Biological Background

A neuron is an electrically excitable cell that processes and transmits information by electrical and chemical signaling. Chemical signaling occurs via synapses, specialized connections with other cells. Neurons form neural networks through connecting with each other.

Computers communicate with bits; neurons use spikes. Incoming signals change the membrane potential of the neuron and when it reaches above a certain value the neuron sends out an action potential (spike).

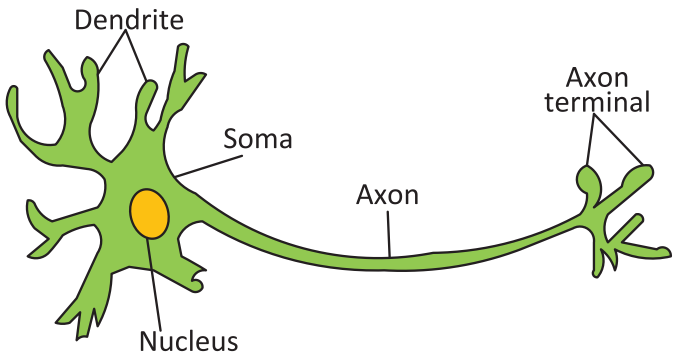

Figure 1.1: Structure of a typical neuron. A neuron typically possesses a soma, dendrites and an axon. The neuron receives inputs via dendrites and sends output through the axon.

As is shown in Figure 1.1, a typical neuron possesses a cell body (often called soma), dendrites, and an axon. The dendrites serve as the inputs of the neuron and the axon acts as the output. The neuron collects information through its dendrites and sends out the reaction through the axon.

Spikes cannot cross the gap between one neuron and the other. Connections between neurons are formed via cellular interfaces, so called synapses. An incoming pre-synaptic action potential triggers the release of neurotransmitter chemicals in vesicles. These neurotransmitters cross the synaptic gap and bind to receptors on the dendritic side of the synapse. Then a post-synaptic potential will be generated [3,4].

The type of synapse and the amount of released neurotransmitter determine the type and strength of the post-synaptic potential. The membrane potential would be increased by excitatory post-synaptic potential (EPSP) or decreased by inhibitory post-synaptic potential (IPSP). Real neurons only use one type of neurotransmitter in all their outgoing synapses. This makes the neuron either be excitatory or inhibitory [3].

1.2.2 Generations of Neuron Models

From the conceptual point of view, all neuron models share the following common features:

1. **Multiple inputs and single output:** The neuron receives many inputs and produces a single output signal.

2. **Different types of inputs:** The output activities of neurons are characterized by at least one state variable that usually corresponding to the membrane potential. An input from the excitatory/inhibitory synapses will increase/decrease the membrane potential.

Based on these conceptual features, various neuron models are developed. Artificial neural networks are already becoming a fairly old technique within computer science. The first ideas and models are over fifty years old. The first generation of artificial neuron is the one with McCulloch-Pitts threshold. These neurons can only give digital output. Neurons of the second generation do not use a threshold function to compute their output signals, but a continuous activation function, making them suitable for analog input and output [5]. Typical examples of neural networks consisting of these neurons are feedforward and recurrent neural networks. They are more powerful than their first generation [6].

Neuron models of the first two generations do not employ the individual pulses. The third generation of neuron models raises the level of biological realism by using individual spikes. This allows incorporating spatiotemporal information in communication and computation, like real neurons do.

1.2.3 Spiking Neuron Models

For the reasons of greater computational power and more biological plausibility, spiking neurons are widely studied in recent years. As the third generation of neuron models, spiking neurons increase the level of realism in a neural simulation.

Spiking neurons have an inherent notion of time that makes them seemingly particularly suited for processing temporal input data [7]. Their nonlinear reaction to input provides them with strong computational qualities, theoretically requiring just small networks for complex tasks.

Leaky Integrate-and-Fire Neuron (LIF)

The leaky integrate-and-fire neuron [4] is the most widely used and best-known model of threshold-fire neurons. The membrane potential of the neuron $V_m(t)$ is dynamically changing over time, as:

$$\tau_m \frac{dV_m}{dt} = -V_m + I(t) \tag{1.1}$$

where τ_m is the membrane time constant in which voltage 'leaks' away. A bigger τ_m can result in a slower decaying process of $V_m(t)$. $I(t)$ is the input current which is a weighted sum from all incoming spikes.

Once a spike arrives, it is multiplied by corresponding synaptic efficacy factor to form the post-synaptic potential that changes the potential of the neuron. When the membrane potential crosses a certain threshold value, the neuron will elicit a spike; after which the membrane potential goes back to a reset value and holds there for a refractory period. Within the refractory time, the neuron is not allowed to fire.

From both the conceptual and computational points of view, the LIF model is relatively simple comparing to other spiking neuron models. An advantage of the model is that it is relatively easy to integrate it in hardware, achieving a very fast operation. Various generalizations of the LIF model have been developed. One popular generalization of the LIF model is the Spike Response Model (SRM), where a kernel approach is used in neuron's dynamics. The SRM is widely used due to its simplicity in analysis.

Hodgkin-Huxley Model (HH) and Izhikevich Model (IM)

The Hodgkin-Huxley (HH) model was based on experimental observations with the large neurons found in squid [8]. It is by far the most detailed and complex neuron model. However, this model is less suited for simulations of large networks since the realism of neuron model comes at a large computational cost.

The Izhikevich model (IM) was proposed in [9]. By choosing different parameter values in the dynamic equations, the neuron model can function differently, like bursting or single spiking.

1.3 Neural Codes

The world around us is extremely dynamic, that everything changes continuously over time. The information of the external world goes into our brain through the sensory systems. Determining how neuronal activity represents sensory information is central for understanding perception. Besides, understanding the representation of external stimuli in the brain directly determines what kind of information mechanism should be utilized in the neural network.

Neurons are remarkable among the cells of the body in their ability to propagate signals rapidly over large distances. They do this by generating characteristic electrical pulses called action potentials or, more simply, spikes that can travel down nerve fibers. Sensory neurons change their activities by firing sequences of action potentials in various temporal patterns, with the presence of external sensory stimuli, such as light, sound, taste, smell and touch.

It is known that information about the stimulus is encoded in this pattern of action potentials and transmitted into and around the brain.

Although action potentials can vary somewhat in duration, amplitude and shape, they are typically treated as identical stereotyped events in neural coding studies. Action potentials are all very similar. In addition, neurons in the brain work together, rather than individually, to transfer the information.

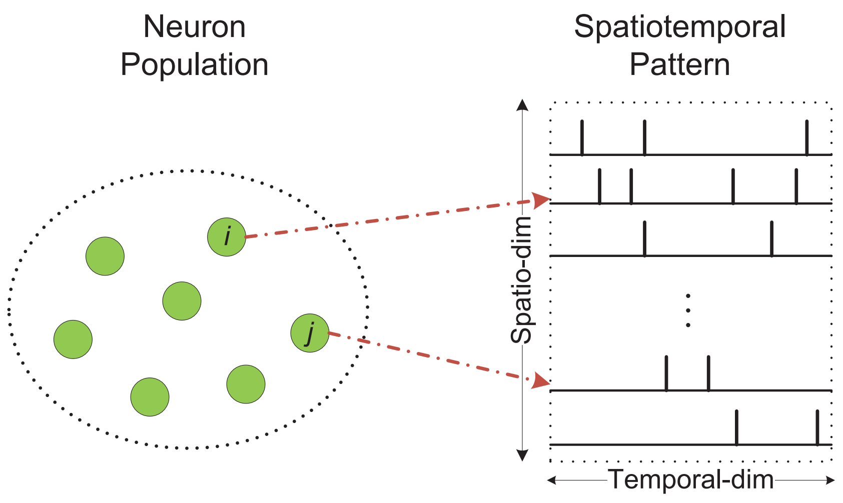

Figure 1.2: A typical spatiotemporal spike pattern. A group of neurons (Neuron Group) works together to transfer the information, with each neuron firing a spike train in time. All spike trains from the group form a pattern with both spatio- and temporal-dimension information. This is called spatiotemporal spike pattern. The vertical lines denote spikes.

Figure 1.2 shows a typical spatiotemporal spike pattern. This pattern contains both spatial and temporal information of a neuron group. Each neuron fires a spike train within a time period. The spike trains of the whole neuron group form the spatiotemporal pattern. The spiking neurons inherently aim to process and produce this kind of spatiotemporal spike patterns.

The question is still not clear that how this kind of spike trains could convey information of the external stimuli. A spike train may contain information based on different coding schemes. In motor neurons, for example, the strength at which an innervated muscle is flexed depends solely on the ‘firing rate’, the average number of spikes per unit time (a ‘rate code’). At the other end, a complex ‘temporal code’ is based on the precise timing of single spikes. They may be locked to an external stimulus such as in the auditory system or be generated intrinsically by the neural circuitry [10].

Whether neurons use the rate code or the temporal code is a topic of intense debate within the neuroscience community, even though there is no clear definition of what these terms mean. The followings further present a detailed overview of the rate code and the temporal code.

1.3.1 Rate Code

Rate code is a traditional coding scheme, assuming that most, if not all, information about the stimulus is contained in the firing rate of the neuron. Because the sequence of action potentials generated by a given stimulus varies from trial to trial, neuronal responses are treated statistically or probabilistically. They may be characterized by firing rates, rather than by specific spike sequences. In most sensory systems, the firing rate increases, generally non-linearly, with increasing stimulus intensity [3]. Any information possibly encoded in the temporal structure of the spike train is ignored. Consequently, the rate code is inefficient but highly robust with respect to input noise.

Before encoding external information into firing rates, precise calculation of the firing rates is required. In fact, the term ‘firing rate’ has a few different definitions, which refer to different averaging procedures, such as an average over time or an average over several repetitions of experiment. For most cases in the coding scheme, it considers the spike count within an encoding window [11]. The encoding window is defined as the temporal window that contains the response patterns that are considered as the basic information-carrying units of the code. The hypothesis of the rate code receives support from the ubiquitous correlation of firing rates with sensory variables [1].

1.3.2 Temporal Code

When precise spike timing or high-frequency firing-rate fluctuations are found to carry information, the neural code is often identified as a temporal code [12]. A number of studies have found that the temporal resolution of the neural code is on a millisecond time scale, indicating that precise spike timing is a significant element in neural coding [13, 14].

Neurons, in the retina [15, 16], the lateral geniculate nucleus (LGN) [17] and the visual cortex [14, 18] as well as in many other sensory systems [19, 20], are observed to precisely respond to the stimulus on a millisecond timescale. These experiments support hypothesis of the temporal code, in which precise timings of spikes are taken into account for conveying information.

Like real neurons, communication is based on individually timed pulses. The temporal code is potentially much more powerful for encoding information with respect to the rate code. It is possible to multiplex much more information into a single stream of individual pulses than you can transmit using just the average firing rates of a neuron. For example, the auditory system can combine the information of amplitude and frequency very efficiently over one single channel [21].

Another advantage of the temporal code is speed. Neurons can be made to react to single spikes, allowing for extremely fast binary calculation. The human brain, for example, can recognize faces in as little as 100 ms [22,23].

There are several kinds of temporal code that have been proposed, like latency code, interspike intervals code and phase of firing code [11]. Latency code is a specific form of temporal code, that encoding information in the timing of response relative to the encoding window, which is usually defined with respect to stimulus onset. The latency of a spike is determined by the external stimuli. A stronger input could result in an earlier spike. In the interspike intervals code, the temporally encoded information is carried by the relative time between spikes, rather than by their absolute time with respect to stimulus onset. In the phase of firing code, information is encoded by the relative timing of spikes regarding to the phase of subthreshold membrane oscillations [11,24].

1.3.3 Temporal Code V.S. Rate Code

In the rate code, a higher sensory variable corresponds to a higher firing rate. Although there are few doubts as to the relevance of this firing rate code, it neglects the extra information embedded in the temporal structure.

Recent studies have shown neurons in the vertebrate retina fire with remarkable temporal precision. In addition, temporal patterns in spatiotemporal spikes can carry more information than the rate-based code [25–27]. Thus, temporal code serves as an important component in neural system.

Since the temporal code is more biologically plausible and computationally powerful, a temporal framework is considered throughout this study.

1.4 Temporal Learning

Learning is a process to acquire new knowledge or modify existing knowledge. Researchers have gone a long way to explore the secret of learning mechanisms in the brain. In neuroscience, the learning process is found to be related to synaptic plasticity, where the synaptic weights are adjusted along the learning.

In 1949, Donald Hebb introduced a basic mechanism that explained the adaptation of neurons in the brain during the learning process [28]. It is called the Hebbian learning rule, where a change in the strength of a connection is a function of the pre- and post-synaptic neural activities. When neuron A repeatedly participates in firing neuron B, the synaptic weight from A to B will be increased.

The Hebbian mechanism has been the primary basis for learning rules in spiking neural networks, though detailed processes of the learning occurring in biological systems are still unclear. According to the schemes on how information is encoded with spikes, learning rules in spiking neural networks can be generally assorted into two categories: rate learning and temporal learning.

The rate learning algorithms, such as the spike-driven synaptic plasticity rule [29, 30], are developed for processing spikes presented in a rate-based framework, where mean firing rates of the spikes are used for carrying information. However, since the rate learning algorithms are formulated in a rate-based framework, this group of rules cannot be applied to process precise-time spike patterns.

To process spatiotemporal spike patterns with a temporal framework, the temporal learning rule is developed. This kind of learning rule can be used to process information that is encoded with a temporal code, where precise timing of spikes acts as the information carrier. Development of the temporal learning rule is imperative considering an increasing body of evidence supporting the temporal code.

Among various temporal rules, several rules have been widely studied, including: spike-timing-dependent plasticity (STDP) [31, 32], the tempotron rule [33], the SpikeProp rule [34], the SPAN rule [35], the Chronotron rule [36] and the ReSuMe rule [37].

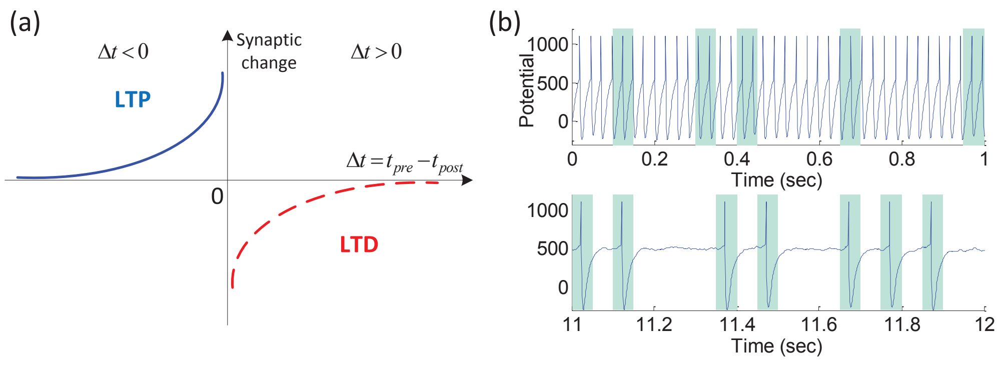

Figure 1.3: Spike-Timing-Dependent Plasticity (STDP). (a) is a typical asymmetric learning window of STDP. Pre-synaptic spike firing before post-synaptic spike will cause long-term potentiation (LTP). Long-term depression (LTD) occurs if the order of these two spikes is reversed. (b) shows the ability of STDP to learn and detect repeating patterns that embedded in continuous spike trains. Shaded areas denote the embedded repeating patterns, and the blue line shows the potential trace of the neuron. Along the learning with STDP, the neuron gradually detects the target pattern by firing a spike.

STDP is one of the most commonly and experimentally studied rules in recent years. STDP is in agreement with Hebbs postulate because it reinforces the connections with the pre-synaptic neurons that fired slightly before the postsynaptic neuron, which are those that 'took part in firing it'. STDP describes the learning process depending on the precise spike timing, which is more biologically plausible. The STDP modification rule is shown as the following equation:

$$\Delta w_{ij} = \begin{cases} A^{+} \cdot exp(\frac{\Delta t}{\tau^{+}}) & , \Delta t \leqslant 0 \\ -A^{-} \cdot exp(-\frac{\Delta t}{\tau^{-}}) & , \Delta t > 0 \end{cases} \tag{1.2}$$

where Δt denotes the time difference between the pre- and post-synaptic spikes. A^{+}, A^{-} and τ^{+}, τ^{-} are parameters of learning rates and time constants, respectively.

As is shown in Figure 1.3(a), if pre-synaptic spike fire before the post-synpatic spike, long-term potentiation (LTP) will happen. Long-term depression (LTD) occurs when the firing order is reversed.

Figure 1.3(b) shows that neurons equipped with STDP can automatically find the repeating pattern which is embedded in a background. The neuron will emit a spike at the presence of this pattern [38–40].

However, STDP characterizes synaptic changes solely in terms of the temporal contiguity of the pre-synaptic spike and the post-synaptic potential or spike. This is not enough for learning spatiotemporal patterns since it would cause silent response sometimes.

The tempotron rule [33] is one such temporal learning rule where neurons are trained to discriminate between two classes of spatiotemporal patterns. This learning rule is based on a gradient descent approach. In the tempotron rule, the synaptic plasticity is governed by the temporal contiguity of pre-synaptic spike, post-synaptic depolarization and a supervisory signal. The neurons could be trained to successfully distinguish two classes by firing a spike or by remaining quiescent.

The tempotron rule is an efficient rule for the classification of spatiotemporal patterns. However, the neurons do not learn to fire at precise time. Since the tempotron rule mainly aims at decision-making tasks, it cannot support the same coding scheme used in both the input and output spikes. The time of the output spike seems to be arbitrary, and does not carry information. To support the same coding scheme through the input and output, a learning rule is needed to let the neuron not only fire but also fire at the specified time. In addition, the tempotron rule is designed for a specific neuron model, which might limit its usage on other spiking neuron models.

By contrast, the SpikeProp rule [34] can train neurons to perform a spatiotemporal classification by emitting single spikes at the desired firing time. The SpikeProp rule is a supervised learning rule for SNNs that based on gradient descent approach. The major limitation is that the SpikeProp rule and its extension in [41] do not allow multiple spikes in the output spike train. To solve this problem, several other temporal learning rules, such as the SPAN rule, the Chronotron rule and the ReSuMe rule, have been developed to train neurons to produce multiple output spikes in response to a spatiotemporal stimulus.

In both the SPAN rule and the Chronotron E-learning rule, the synaptic weights are modified according to a gradient descent approach in an error landscape. The error function in the Chronotron rule is based on the Victor & Purpura distance [42] in which the distance between two spike trains is defined as the minimum cost required to transform one into the other, while in the SPAN rule the error function is based on a metric similar to the van Rossum metric [43] where spike trains are converted into continuous time series for evaluating the difference. These arithmetic calculations can easily reveal why and how networks with spiking neurons can be trained, but the arithmetic-based rules are not a good choice for designing networks with biological plausibility. The biological plausibility of error calculation is at least questionable.

From the perspective of increased biological plausibility, the Chronotron I-learning rule and the ReSuMe rule are considered. The I-learning rule is heuristically defined in [36] where synaptic changes depend on the synaptic currents. According to the I-learning rule, its development seems to be based on a particular case of the Spike Response Model [4], which might also limit its usage on other spiking neuron models or at least is not clearly demonstrated. Moreover, those synapses with zero initial weights will never be updated according to the I-learning rule. This will inevitably lead to information loss from those afferent neurons.

In view of the two aspects presented above, i.e., the biological plausibility and the computational efficiency, one major purpose of this study was to combine the two aspects for a new temporal learning rule and develop a comprehensive research framework within a system where information is carried by precise-timing spikes.

1.5 Objectives and Contributions

Even though many attempts have been devoted to exploring mechanisms used in the brain, a majority of facts about spiking neurons for information processing and cognitive computing still remain unclear. The research gaps for current studies on SNNs are summarized below:

1. Temporal coding and temporal learning are two of the major areas in SNNs. Various mechanisms are proposed based on inspirations from biological observations. However, most studies on these two areas are independent. There are few studies considering both the coding and the learning in a consistent system [30, 34, 44–46].

2. Over the rate-based learning algorithms, the temporal learning algorithms are developed for processing precise-timing spikes. However, these temporal learning algorithms focus more on the aspects of either arithmetic or biological plausibility. Either side of these two aspects would not be a good choice considering both the computational efficiency and the biological plausibility.

3. Currently, there are few studies on the practical applications of SNNs [30, 34, 45–47]. Most studies only focus on theoretical explorations of SNNs.

4. Learning mechanisms for building causal connections have not been clearly investigated.

The main aim of this study is to explore and develop cognitive computations with spiking neurons under a temporal framework. The specific objectives of this research are:

1. To develop an integrated consistent system of spiking neurons, where both the coding and the learning are considered from a systematic level.
2. To develop a new temporal learning algorithm that is both simple for computation and also biologically plausible.
3. To investigate various properties of the proposed algorithm, such as memory capacity, robustness to noise and generality to different neuron models, etc.
4. To investigate the ability of the proposed SNNs applying to different cognitive tasks, such as image recognition, sound recognition and sequence recognition, etc.
5. To investigate the temporal learning in multilayer spiking neural networks.

The significance of this study is two-fold. On one hand, such models proposed in this study may contribute to a better understanding of the mechanisms by which the real brains operate. On the other hand, the computational models inspired from biology are interesting in their own right, and could provide meaningful techniques for developing real-world applications.

This thesis is restricted to computer simulations for exploring cognitive computations of spiking neurons. There is no intention to perform experiments on biological systems since this is beyond the scope of this study. The computations of spiking neurons in this study are considered in a temporal framework rather than a rate-based framework. This is because mounting evidence shows that precise timing of individual spikes plays an important role.

In addition, the temporal framework offers significant computational advantages than the rate-based framework.

1.6 Outline of the Thesis

In the area of theoretical neuroscience, the general target is to provide a quantitative basis for describing what nervous systems do, understanding how they function, and uncovering the general principles by which they operate. It is a challenging area since multidisciplinary knowledges are required for building models. Investigating spike-based computation serves as a main focus for conducting the research work of this study. To further specify the research scope, the temporal framework is considered in this study. In order to achieve the aforementioned objectives, a general system structure has been devised. Further investigations on individual functional parts of the system are conducted. The organization of the remaining chapters of this thesis is as follows.

Chapter 2 presents a brain-inspired spiking neural network system with simple temporal encoding and learning. With a biologically plausible supervised learning rule, the system is applied to various pattern recognition tasks. The proposed approach is also benchmarked with the nonlinearly separable task.

In Chapter 3, more complex and biologically plausible system structures are developed based on the one proposed in Chapter 2. The encoding system provides different levels of robustness, and enables the spiking neural networks to process real-world stimuli, such as images and sounds. Detailed investigations on the encoding and learning are also provided.

In Chapter 4, a novel learning rule, namely Precise-Spike-Driven (PSD) synaptic plasticity, is proposed for training the neuron to associate spatiotemporal spike patterns. The PSD rule is simple, efficient, and biologically plausible. Various properties of this rule are investigated.

Chapter 5 presents the application of the PSD rule on sequence recognition. In addition, the classification ability of the PSD rule is investigated and benchmarked against other learning rules.

In Chapter 6, the learning in multilayer spiking neural networks is investigated. Causal connections are built to facilitate the learning. Several tasks are used to analyze the learning performance of the multilayer network.

Finally, Chapter 7 presents the conclusions of this thesis and some future directions.

Chapter 2

A Brain-Inspired Spiking Neural Network Model with Temporal Encoding and Learning

Neural coding and learning are important components in cognitive memory systems, by processing the sensory inputs and distinguishing different patterns to provide higher level brain functions such as memory storage and retrieval. Benefiting from biological relevance, this chapter presents a spiking neural network of leaky integrate-and-fire (LIF) neurons for pattern recognition. A biologically plausible supervised synaptic learning rule is used so that neurons can efficiently make a decision. The whole system contains encoding, learning and readout. Utilizing the temporal coding and learning, networks of spiking neurons can effectively and efficiently perform various classification tasks. The proposed system can learn patterns of either discrete values or continuous values through different encoding schemes.

2.1 Introduction

The great computational power of biological systems has drawn increasing attention from researchers. Although the detailed information processing involved in memory is still unclear, observed biological processes have inspired many computational models operating at power efficiencies close to biological systems. Pattern recognition is the ability to identify objects in the environment. As is a necessary step in all cognitive processes including memory, it is better to consider pattern recognition from brain-inspired models which could potentially provide great computational power.

In order to approach biological neural networks, the artificial neural networks (ANNs) are developed as simplified approximations in terms of structure and function. Since early neurons of the McCulloch-Pitt neuron in 1940s and the perceptron in 1950s [48], referred as the first generation neuron models, ANNs have been evolving towards more neural-realistic models. Different from the first generation neurons in which step-function threshold is used, the second generation neurons use continuous activation functions (like a sigmoid or radial basis function) as threshold for output determination [49].

The first two generations are referred as traditional neuron models. Studies on biological systems disclose that neurons communicate with each other through action potentials. As the third generation neuron model, spiking neurons raise the level of biological realism by utilizing spikes. The spiking neurons dealing with precise timing spikes improve the traditional neural models on both the aspects of accuracy and computational power [50]. Among different kinds of spiking neuron models, the leaky integrate-and-fire (LIF) model is the most widely used spiking neuron model [30, 33, 34, 40, 46, 47, 51] due to its simplicity and computational effectiveness.

Encoding is the first step in creating a memory, which considers how information is represented in the brain. Although results remains unclear, there are strong reasons to believe that it is optimal using pulses to encode the information for transmission [52]. The inputs to a spiking neuron are discrete spike times. Rate coding and temporal coding are two basic and widely studied schemes of encoding information in these spikes. In the rate coding the average firing rate within a time window is considered, while for the temporal coding the precise timings of spikes are considered [11]. Neurons, in the retina [16,23], the lateral geniculate nucleus (LGN) [17] and the visual cortex [14] as well as in many other sensory systems, are observed to precisely respond to stimuli on a millisecond timescale [13]. Temporal patterns can carry more information than rate-based patterns [25–27]. The capability of encoding information in the timing of single spikes to compute and learn realistic data is demonstrated in [53]. The scheme of utilizing single spikes to transfer information could potentially be beneficial for efficient pulse-stream very large scale integration (VLSI) implementations.

Many algorithms for spiking neural networks (SNNs) have been proposed. Based on arithmetic calculations, the SpikeProp [34, 53] was proposed for training SNNs, similar in concept to the backpropagation algorithm developed for traditional neural networks [54]. Others use bio-inspired algorithms, such as spike timing dependent plasticity (STDP) [31,55–57], the spike-driven synaptic plasticity [30], and the tempotron rule [33]. Although the arithmetic calculations can easily reveal why and how networks can be trained, the arithmetic-based rules are not a good choice building networks with a biological performance. STDP is found to be able to learn distinct patterns in an unsupervised way [40], and it characterizes synaptic changes solely in terms of the temporal contiguity of presynaptic spikes and postsynaptic potentials or spikes. In the spike-driven synaptic plasticity [30], a rate coding is used. The learning process is supervised and stochastic, in which a teacher signal steers the output neuron to a desired firing rate. Being different with spike-driven synaptic plasticity, the tempotron learning rule [33] is efficient to learn spiking patterns where information is embedded in precise timing spikes.

Although SNNs show promising capability in playing a similar performance as living brains due to their more faithful similarity to biological neural networks, the big challenge of dealing with SNNs is reading data into and out of them, which requires proper encoding and decoding methods [58]. Some existing SNNs for pattern recognition (as in [30, 59]) are based on the rate coding. Different from these SNNs, we focus more on the temporal coding which could potentially carry the same information efficiently using less number of spikes than the rate coding. This could largely facilitate the computing speed.

In this chapter, we build a bio-inspired model of SNNs containing encoding, learning and readout. Neural coding and learning are the main considerations in this chapter, since they are important components in cognitive memory system by processing the sensory inputs and distinguishing different patterns to allow for higher level brain functions such as memory storage and retrieval [60]. Inspired by the local receptive fields of biological neurons, the encoding neuron integrates information from its receptive field and represents the encoded information through precise timing of spikes. The timing scale of spikes is on a millisecond level which is consistent with biological experimental observations. The readout part uses a simple binary presentation to represent fired or non-fired state of the output neuron.

The main contribution of this chapter lies in the approaches of designing SNNs for pattern recognition. Pattern recognition helps to identify and sort information for further processing in brain systems. A new coming pattern is recognized upon paying attention and similarity to previously learned patterns which obtained through weight modification. Recognition memory is formed and stored in synaptic strengths. Inspired by biology, spiking neurons are employed for computation in this chapter. The proposed functional system contains encoding, learning and readout parts. We demonstrate that, utilizing the temporal coding and learning, networks of spiking neurons can effectively and efficiently perform various classification tasks.

The rest of this chapter is organized as follows. Section 2.2 presents the architecture of the spiking neural network. Section 2.3 describes the temporal learning rule we used in our approaches. The relationship between this rule and well-studied STDP is also introduced. Section 2.4 shows the ability of the network to learn different patterns of neural activities (discrete-valued vectors). Section 2.5 shows the SNN for learning continuous input variables. We use the well-known Iris dataset problem to benchmark our approach against several existing methods. Finally, we end up with discussions in Section 2.6, followed by conclusions in the last section.

2.2 The Spiking Neural Network

In this section, we describe the whole system architecture of spiking neurons for obtaining recognition memory. The system composes of 3 functional parts: the encoding part, the learning part and the readout part (see Figure 2.1). A stimulus consists of several components. The components are partially connected to encoding neurons to generate encoded spiking information. The encoding neurons are fully connected to learning neurons.

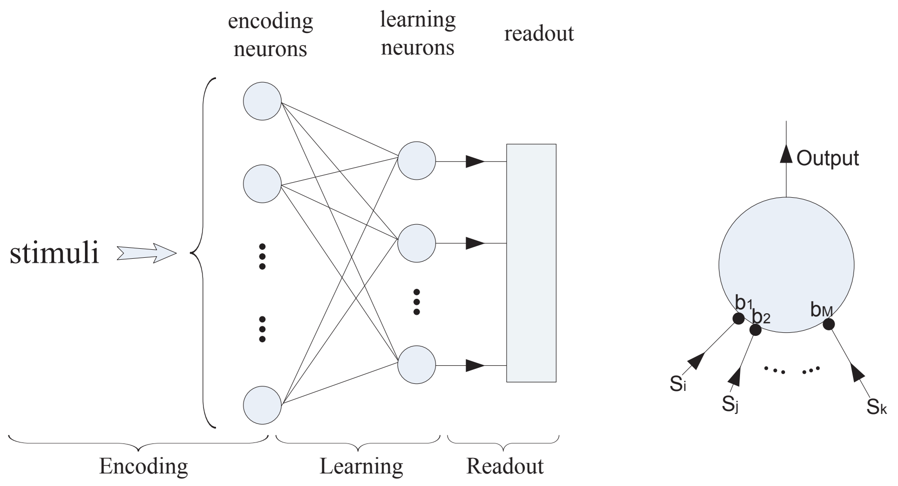

Figure 2.1: Architecture for pattern recognition. Left: A schematic of the system architecture. Right: Encoding neuron model. It has one output and M input points connected to part of the stimulus. It performs a mapping function that converts a value string to a temporal spike.

Each part plays a different functional role in the system: the encoding layer generates a set of specific activity patterns that represent various attributes of external stimuli; the learning layer tunes the neurons' weights making sure particular neurons can respond to certain patterns correctly; the readout part extracts information about the stimulus from a given neural response. Through this architecture, the problem of getting data into and out of the spiking neural network is solved, and the task of pattern recognition could be fulfilled.

2.2.1 Encoding

The encoding part aims to generate spike patterns that represent the input stimuli. The temporal encoding is used over the rate-based encoding when patterns within the encoding window provide information about the stimulus that cannot be obtained from spike count. The latency code [11] is a simple example of temporal encoding. It encodes information in the timing of response relative to the encoding window, which is usually defined with respect to the stimulus onset. The single spike latencies are used to encode stimulus information in our system. Within the encoding window, each input neuron fires only once.

Each encoding neuron has M input points (Figure 2.1) which are selected from components of the stimulus. It performs a specific function to convert the input points into latencies within the encoding window. For example, if the stimulus is composed of binary values (0 or 1), the function of the encoding neuron is to convert the binary strings into temporal patterns of discrete spikes. The encoding time window is chosen to be hundreds of milliseconds, consistent with biological observations.

2.2.2 Learning

The learning part of the network is composed of one layer of tempotrons [33]. The encoding neurons are fully connected to the learning neurons. The number of synapses to a learning neuron is equal to the number of encoding neurons (N_{en}) according to the structure. As used in the perceptron [61] and tempotron [33] learning rules, the ratio of the number of random patterns that a neuron can correctly classify over the number of its synapses, is used to measure the memory load. The maximum number of randomly generated patterns that a tempotron can learn is roughly 3 times the number of its synapses [33]. Therefore, as long as the number of patterns does not exceed the critical load value, the network can perform the task well. If there are too many patterns, the number of encoding neurons should be increased correspondingly. The learning neurons generate action potentials when the internal neuron state crosses a firing threshold. The tempotron rule is used to train neurons to react at a desired firing state when presented to incoming stimuli.

2.2.3 Readout

The readout part aims to extract information about the stimulus from responses of the learning neurons. In this part, we can use a binary sequence to represent a certain class of patterns for the reason that each learning neuron can only discriminate two groups. Each learning neuron responds to a stimulus by firing (1) or not firing (0). So, the total N learning neurons as the output can represent a maximum number of 2^N classes of patterns. The number of learning neurons is determined by the number of classes in the recognition task. For example, four readout could be sufficient for a group of patterns containing 16 classes.

2.3 Temporal Learning Rule

Temporal learning rules aim to deal with information encoded by precise-timing spikes. One of the most commonly studied rules is spike-timing-dependent plasticity (STDP) which has emerged in recent years as experimentally most studied form of synaptic plasticity (see [31, 55–57, 62] for reviews). According to STDP, the plasticity depends on the intervals between pre- and post-synaptic spikes. The basic mechanisms of plasticity found in STDP are the long term potentiation (LTP) and the long term depression (LTD). However, STDP characterizes synaptic changes solely in terms of the temporal contiguity of presynaptic spikes and postsynaptic potentials or spikes. In addition, to get convergence of learning with STDP, a suitable balance of many parameters is needed [57]. In [33], the tempotron learning rule is presented. In this rule, the synaptic plasticity is governed by the temporal contiguity of pre-synaptic spike and post-synaptic depolarization, and a supervisory signal. The tempotron can make appropriate decision under the supervisory signal by tuning fewer parameters than STDP. Moreover, the tempotron rule also uses mechanisms of LTP and LTD to fulfill synaptic plasticity as in STDP. Because of the discrete nature of spikes, the evaluation of neural dynamics in our study is performed on a time step of $dt = 1\ ms$.

The neuron model used here is a leaky integrate-and-fire (LIF) neuron driven by exponential decaying synaptic currents generated by its synaptic afferents. The potential of the neuron is a weighted sum of postsynaptic potentials (PSPs) from all incoming spikes:

$$V(t) = \sum_i w_i \sum_j K(t - t_i^j) + V_{rest} \tag{2.1}$$

where w_i and t_i^j are the synaptic efficacy and the $j\text{-}th$ firing time of the $i\text{-}th$ afferent. V_{rest} is the rest potential of the neuron. K denotes a normalized PSP kernel:

$$K(t - t_i^j) = V_0 \cdot (\exp(\frac{-(t - t_i^j)}{\tau_m}) - \exp(\frac{-(t - t_i^j)}{\tau_s})) \tag{2.2}$$

where τ_m and τ_s denote decay time constants of membrane integration and synaptic currents. We choose $\tau_m = 4\tau_s = 15ms$ in the following sections. V_0 normalizes PSP so that the maximum value of the kernel is 1. $K(t - t_i^j)$ is a causal filter that only considers spikes $t_i^j \leq t$. The kernel function is shown in Figure 2.2. Each afferent spike will cause a change on the potential of post-synaptic neuron. The height of the PSP is modulated by the synaptic efficacy w_i to get effective post-synaptic potential. The final potential of the post-synaptic neuron is a summation over all afferents.

In the classification task, each input pattern belongs to one of two classes (which are labeled by P^+ and P^-). One neuron can discriminate these patterns by firing or not. When a P^+ pattern is presented to the neuron, it should fire a spike; when a P^- pattern is presented to the neuron, it should keep silent by not firing. The neuron learns patterns by changing its synaptic efficacies (w_i) whenever there is an error. If the neuron fails to fire in response to a P^+ pattern, this is denoted as a P^+ error. If the neuron erroneously fires a spike in response

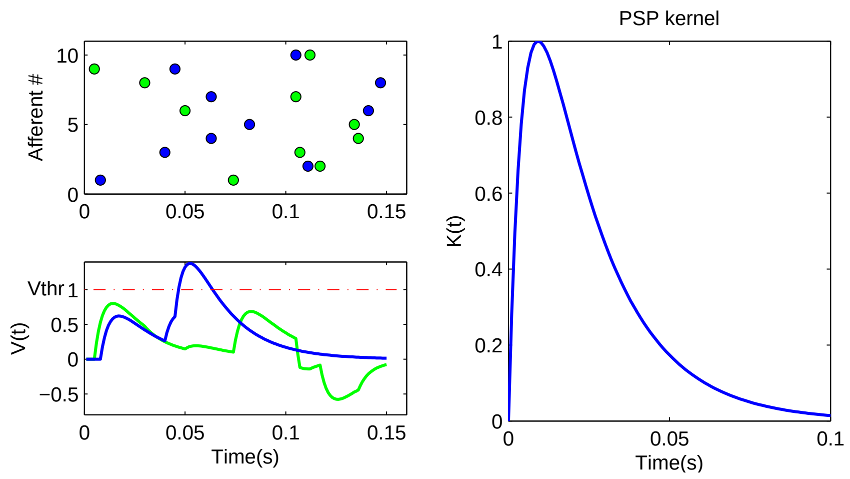

Figure 2.2: Dynamics of the tempotron response. Left top: examples of spiking patterns. There are two patterns (blue and green) and each spike from an input afferent is denoted by a dot. The Y axis show input identification number. Left bottom: neural potential traces. Each color of lines corresponds to the same color patterns on the left top. In this neuron model, the potential boundaries at the threshold and the rest potential are ignored. Right: normalized PSP kernel.

to a P^- pattern, it is denoted as a P^- error. Depending on the type of error, the learning rule is as:

$$\Delta w_i = \begin{cases} \lambda_+ \sum_{t_i^j < t_{max}} K(t_{max} - t_i^j), & \text{if } P^+ \text{ error;} \\ -\lambda_- \sum_{t_i^j < t_{max}} K(t_{max} - t_i^j), & \text{if } P^- \text{ error;} \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

where t_{max} denotes the time at which the neuron reaches its maximum potential value in the time domain. $\lambda > 0$ is a constant representing the learning rate (we set $\lambda_+ = \lambda_- = \lambda = 0.005$ here). It denotes the maximum change on synaptic efficacies.

The tempotron updates its weights whenever it fails to respond as the same desired state as the instructor. It means that, within the presentation time of a pattern (T), the neuron will perform weight modification as long as its firing state violates the instructor. Such a method requires the supervisory signal to evaluate neuron's responding state at each time step. A trial updating method could also be adopted where the synaptic weights are modified at the end of each pattern presentation. This method only requires the instructor to make evaluation at the end of pattern presentation. Considering efficiency and biological realism, we adopt the dynamic updating method. Whenever an error occurs, the neuron will immediately update its weights. Each spike firing prior to t_{max} will result in a change on the corresponding synapse. The shape of the learning window follows kernel K and the changing amount of the weight depends on the time difference between t_i^j and t_{max}. If we only consider single spike coding and the latest spike updating, the learning rule will be simplified as:

$$\Delta w_i = \begin{cases} \lambda_+ K(t_{max} - t_i)\Theta(t_{max} - t_i), & \text{if } P^+ \text{ error;} \\ -\lambda_- K(t_{max} - t_i)\Theta(t_{max} - t_i), & \text{if } P^- \text{ error;} \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

where t_i is the spike time of the i-th afferent, and $\Theta(\cdot)$ is a heaviside function. This simplified rule is used in the rest of this chapter.

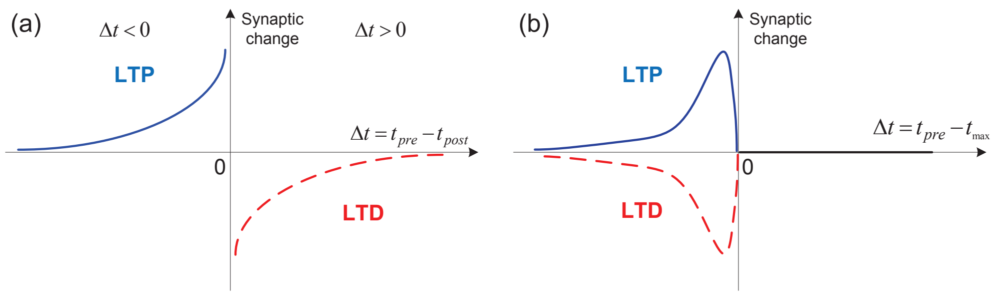

Figure 2.3: Learning windows of different rules. The blue lines denote the LTP process and the dashed red lines denote the LTD process. (a) is the learning window for STDP; (b) is for tempotron rule.

From a biological perspective, a training algorithm should adapt synaptic weights based on the states of pre and post-synaptic neurons to keep with

Hebbian theory. Normally, a longer time difference will result in a little weight change while a shorter time difference results in a larger change, as like processes in biological systems. In the tempotron rule, two STDP-like windows are used to adjust the synaptic weights (see Figure 2.3). The LTP window is used to increase the synaptic weights and the LTD window is for depressing the weights, whenever the neuron fails to respond in a desired output.

The learning process uses a supervisory signal. Although so far there is no strong experimental confirmation of the supervisory signal, an increasing body of evidence shows that this kind of learning is also exploited by the brain [63]. The most documented evidence for this type of rule comes from studies on the cerebellum and the cerebellar cortex [64, 65]. In addition, there is evidence that the supervisory signals are provided to the learning modules by sensory feedback [66] or other supervisory neural structures in the brain [65]. In the tempotron, the supervisory signal is only for determining the polarity of synaptic changes. Classical error feedback is a possible way to implement this control of polarity. A neuromodulator released by the supervisory system can induce the control of adaptation. This control occurs for several neuromodulatory pathways, such as dopamine and acetylcholine [33, 67, 68]. In addition, the gating role of the supervisory signal has strong biological resonance such as voltage-gated calcium channels and NMDA receptors. Their involvements in the induction of long term plasticity are well established [69, 70].

2.4 Learning Patterns of Neural Activities

Many ways of encoding memory patterns in neural networks have been studied. The memory patterns encoded in synaptic weights can be taken to be binary vectors, as well as they can also be taken to be drawn from a distribution with several discrete activity values or from a continuous distribution [60]. In Hopfield network [71], memory patterns are expressed through the activities of neurons, where the states of the neurons have binary values (+1 for active neuron and -1 for inactive neuron). In some other networks, non-binary coding schemes [72] are also introduced.

In the previous section, the ability of the tempotron to separate temporal patterns is introduced. However, the following questions arise: can this method be used to recognize memory patterns mentioned above in this section? If so, how can it perform the task?

The patterns are n-dimensional vectors and the value of each element in the vector refers to neuron's activity which can be drawn from several discrete values. The coding schemes used here are same as that in Treves and Rolls [73]. The activity η of each neuron follows a probability distribution function $p(\eta)$:

$$p(\eta) = \begin{cases} (1-c)\delta(\eta-\eta_0) + c\delta(\eta-\eta_1), & \text{(binary)} \\ (1-\frac{4c}{3})\delta(\eta-\eta_0) + c\delta(\eta-\eta_1) + \frac{c}{3}\delta(\eta-\eta_2), & \text{(ternary)} \\ (1-2c)\delta(\eta) + 4ce^{-2\eta}, & \text{(exponential)} \end{cases} \tag{2.5}$$

where $\delta(x)$ is the Dirac's function: $\delta(x) = 1 \; (x = 0), 0$ (otherwise). c is the coding level which is defined as the mean level of the network activity [60, 73].

As explorations for the ability of the tempotron to classify different patterns of activities, we use binary and ternary patterns as stimuli. The ternary patterns represent a simple non-binary structure. We also use variable coding levels to see the performance. The pattern vectors are generated according to Equation (2.5).

The activity values we choose for binary patterns are $\eta_0 = 0$ and $\eta_1 = 1$, and for ternary patterns are $\eta_0 = 0$, $\eta_1 = 1$ and $\eta_2 = 2$. Some examples of binary and ternary patterns are shown in Figure 2.4.

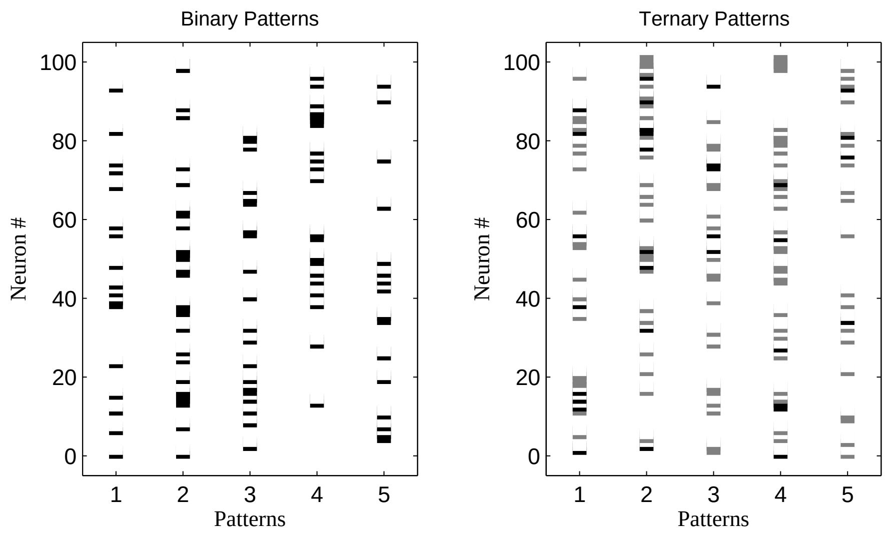

Figure 2.4: Examples of binary and ternary patterns with c=0.2. The neural activities are shown in gray scale (the maximal activity value is shown in black, and the minimal activity value is in white). There are 5 patterns in each sub-figure and only the activities of 100 neurons are included.

Pattern is stored in an n-dimensional vector with discrete values of activity. We use the system architecture of spiking neurons to classify pattern vectors (see Figure 2.1). The layer of encoding neurons performs a function converting the pattern vector into temporal pattern for the tempotron to classify. We only use one learning neuron to test the ability of the tempotron learning two groups of patterns.

To test the performance, we generate 100 memory patterns with 1024 elements, and the activity value of each element is randomly drawn according to the probability distribution function presented in Equation (2.5). Then we randomly assign half of patterns to one group and others to another group. We also use different coding levels ($c = 0.2$ and $c = 0.5$) in our simulation. In order to control the encoding time window in a scale around $250\ ms$, we set the number of input points of the encoding neuron to 8 and 5 for binary and ternary patterns, respectively. Each element of the pattern vector is connected to only one encoding neuron and the connections between the pattern vector and encoding neurons are in order. For example, in binary patterns, the first 8 elements connect to the first encoding neuron and the second 8 elements connect to the second and the last 8 connect to the last encoding neuron. The encoding neuron in this case acts as a converter that translates a binary or ternary string into a spike timing.

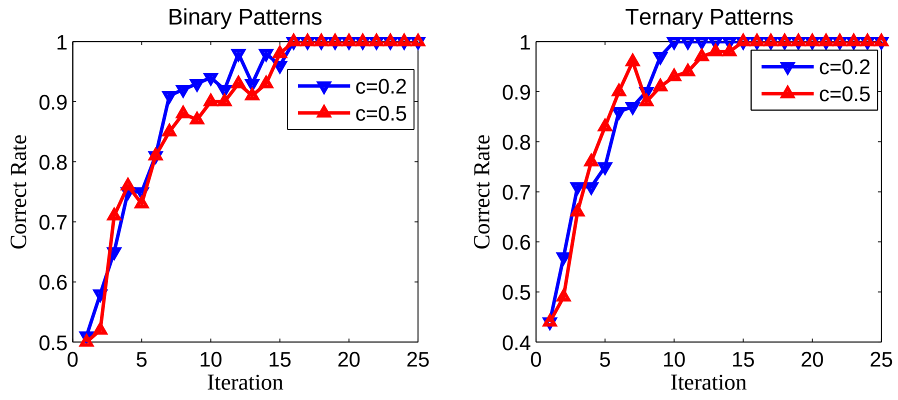

Figure 2.5: Classification results for different patterns of activities. The pattern is 1024-dimensional vector. The total number of patterns is 100 (each class has 50) in each simulation.

From Figure 2.5, we see the tempotron can successfully learn different patterns of activities that are presented in discrete values. After several learning iterations, the neuron can correctly classify discrete-valued patterns under different coding levels. Thus, we successfully investigate a method for spiking neurons to perform classification on discrete-valued patterns.

2.5 Learning Patterns of Continuous Input Variables

In this section, we conduct experiments with our spiking neural network on classifying patterns with continuous variables. We use the Iris dataset to benchmark our approach against several existing methods.

2.5.1 Encoding Continuous Variables into Spike Times

To encode the continuous variables into spike times on a precision of millisecond level, we employ a similar approach as in [34] based on arrays of receptive fields. As a result, each input variable is represented by a group of neurons with graded and overlapping sensitivity profiles. This approach is biologically plausible and well studied method for representing real-valued parameters [34, 74].

We adopt the same encoding setup as in [34, 75], where each input dimension is encoded by an array of one-dimensional Gaussian receptive fields. For a variable x in a range $[x_{min}, x_{max}]$, n neurons with different Gaussian receptive fields are used to encode. The center and width of the $i\text{-}th$ neuron are set to $\mu_i = x_{min} + (2 \cdot i - 3)/2 \cdot (x_{max} - x_{min})/(n - 2)$ and $\sigma_i =$

$1/1.5 \cdot (x_{max} - x_{min})/(n-2)$, respectively. The activation values of the n neurons encoding the variable x are calculated. Highly activated neurons will fire early and less activated neurons will fire later or not fire.

Through this temporal encoding approach, two important benefits could be obtained. Firstly, a sparse coding, allowing for efficient simulation as in [76], is achieved through a small set of significantly activated neurons. Secondly, an optimal number of neurons could be roughly obtained for each independently encoded variable [34].

2.5.2 Experiments on the Iris Dataset

The three-class Iris dataset is used to benchmark our approach since it is perhaps one of the best known databases to be found in the recognition literature. The different three classes represent the different species of the Iris plant, including Iris Setosa Canadenisis (*Class 1*), Iris Veriscolor (*Class 2*) and Iris Virginica (*Class 3*). The dataset contains 150 samples, 50 for each class. Each sample has 4 input variables: sepal length, sepal width, petal length and petal width. The latter two classes are not linearly separable from each other.

To encode these data, we firstly normalize the 4 variables into a same range. Each input variable is encoded by $n = 12$ neurons with Gaussian receptive fields. Thus, for each input pattern, the 48 activation values between 0 and 1 can be calculated. We ignore activation values below 0.1 since they are too weak to stimulate a spike. These activation values are then linearly converted to delay times, associating $t = 0$ with activation value 1 and later times up to $t = 100\ ms$ with lower activation values. The spike times are rounded to $dt = 1$

ms precision. The dataset is split into two sets and classified using two-fold cross-validation.

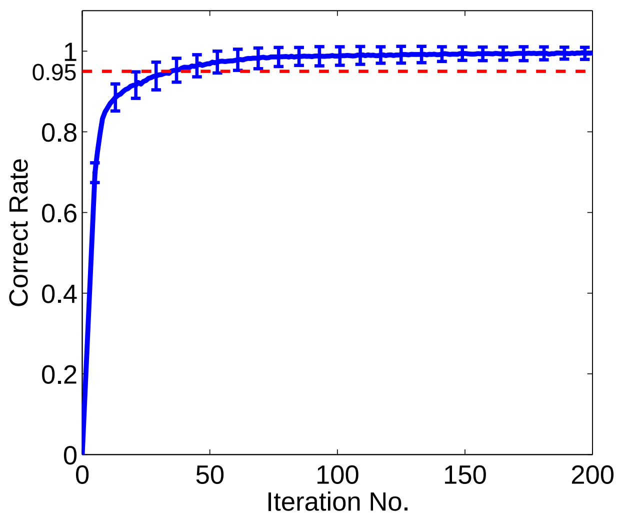

Figure 2.6: Correct rate for classifying *Class 2* from the other classes vs. iterations for training. The plot is averaged over 100 runs.

Before the multi-class problem, it is necessary to investigate the performance of a single spiking neuron to classify two classes. Through this process, a proper stopping criteria for the purpose of training the network could be chosen. We simulate one spiking neuron to separate *Class 2* from the other classes. We set the maximum number of iterations for training to be $Max_{iter} = 200$. According to Figure 2.6, the neuron can rapidly reach a high accuracy (0.95) within tens of training iterations, and it stabilizes at high accuracy for further training.

For balancing between a high simulation speed and a high accuracy, it is reasonable to choose a lower Max_{iter} according to Figure 2.6. We set $Max_{iter} = 100$ for the multi-class problem. After each training period, the neuron will fall into two cases. We refer *Case1* as that the neuron successfully

Table 2.1: Comparison of different training algorithms: results for Iris dataset

Algorithm	Inputs	Hidden	Outputs	Iterations	Training	Testing
MatlabBP	50	10	3	2.6×10^6	$98.2\% \pm 0.9$	$95.5\% \pm 2.0$
MatlabLM	50	10	3	3750	$99.0\% \pm 0.1$	$95.7\% \pm 0.1$
SpikeProp [34]	50	10	3	1000	$97.4\% \pm 0.1$	$96.1\% \pm 0.1$
SpikeProp [50]	17	8	3	37	$\geq 95\%$	92.7%
SWAT	16	208	3	500	$95.5\% \pm 0.6$	$95.3\% \pm 3.6$
Tem	48	-	3	less than 100	$99.63\% \pm 0.81$	$92.55\% \pm 3.3$
Tem*Case1*	48	-	3	less than 100	100%	$93.09\% \pm 2.94$
Tem*Case2*	48	-	3	less than 100	$98.9\% \pm 1.06$	$91.49\% \pm 3.74$

separates all samples in the training set before Max_{iter} reaches, and *Case2* as that the neuron still cannot separate all samples at the end of maximum number of training iterations.

For the three-class Iris problem, we employ only one neuron per output class, and the output neuron with the strongest activation state represents the class association. The training is stopped either when the Max_{iter} reaches or when the neuron successfully separates all training samples before Max_{iter} reaches. After 100 runs of training and testing, the averaged classification accuracy for the training set is 96.63% and for testing set is 92.55%. Among the 100 runs of training, we find there are 66 runs in which all the three neurons are trained in *Case1*, and 34 runs where at least one neuron is trained in *Case2*. We refer these two situations as Tem*Case1* and Tem*Case2* respectively. Interestingly, in the case of Tem*Case1*, the classification accuracy for both the training set and testing set is improved, reaching 100% and 93.09%, respectively.

Table 2.1 presents the results of our approach against several existing algorithms for the Iris dataset. MatlabBP and MatlabLM, representing traditional artificial neural network, are built-in functions of Matlab that implement the backpropagation and Levenberg-Marquardt training algorithms. SpikeProp [34, 50] and SWAT [46], as spiking neural networks applied on Iris dataset, are also used to benchmark our approach. The SpikeProp, as the first supervised training algorithm for SNNs, was an adaptation of gradient-descent-based-error-backpropagation method [34]. An efficient SpikeProp for Iris classification was presented in [50], where less synaptic weights were used.

As is shown in Table 2.1, the training accuracy of our approach slightly surpasses other approaches, and the testing accuracy is comparable and acceptable. Some of the state-of-the-art approaches such as [77, 78] that come from hybrid-system approach, can even result in a higher accuracy (normally over 96%). However, it is extremely time consuming to train if genetic algorithms are used in the hybrid system. Although the classification accuracy of our approach does not beat other approaches at this moment, the ability to use a biologically plausible SNN to do the task is highlighted. With a comparable and acceptable classification accuracy, our approach is more efficient and effective than other methods as listed in Table 2.1. It can perform the task comparably well with less neurons (without a layer of hidden neurons) and with less number of learning iterations (within 100). This preliminary approach with biologically plausible SNN demonstrates the great computational power inherited from biology. Continued improvements on this approach could be explored to perform better than conventional machine-learning algorithms.

2.6 Discussion

In this section, we discuss several considerations regarding biological relevance that benefit the procession in our approach.

The computational power of biological neurons attracts the community to develop models for computation on action potentials. The average firing rate of neuron is generally assumed to be the coding scheme, with the success in neural network modeling and the substantial electrophysiological support. However, there has been increasing number of reports showing that precise timings of action potentials carry significant information. In addition, it has been demonstrated that the temporal coding with precise timings can carry more information than the rate coding scheme. The usage of time-to-first-spike coding facilitate the computational speed in SNNs. For machine learning purposes, efficient implementations of SNNs can be obtained by the event nature of spikes.

The encoding scheme used in this chapter is inspired from receptive fields of biological neurons. Each neuron receives a partial information from external stimuli. The encoding window of the temporal patterns is chosen to be on a scale of hundreds of milliseconds, which matches the biological evidence [11,13,26]. Although the encoding window could be flexible by scaling up or down, a choice from biology could make this approach consistent and compatible with other bio-inspired models in the case of combination.

Besides biological plausibility, another benefit of a biologically inspired spiking system would be that it offers the possibility of real-time learning systems. Biological neural networks need to respond in real time to real-world stimuli. The needs for fast reactive systems normally shadow classical computing approaches which have mostly focused on off-line problems. Thus, the responding speed is another reason for the choice of spiking neural networks.

2.7 Conclusion

This chapter presents an architecture of spiking neurons to approach pattern recognition on various classification tasks such as recognition of neural activities and continuous input variables. The recognition memory is formed through weights modification during the learning process. A new pattern could be recognized through matching what the network has learnt. Since the temporal encoding and learning in this chapter are believed to be inherited from the biological neural systems, the biological plausibility of the approach is a main aspect in this study considering machine learning as a main target. A general SNN system for pattern recognition is proposed, where it contains encoding, learning and readout. Our approach is benchmarked using the Iris dataset problem, and the results highlight the capability of our approach to classify nonlinearly-separable data effectively and efficiently.

Chapter 3

Rapid Feedforward Computation by Temporal Encoding and Learning with Spiking Neurons

The previous chapter presents a general structure of spiking neural network for pattern recognition, showing that the SNN has the ability to learn different patterns of activities and continuous input variables [79]. However, considering some real-world stimuli (such as images and sounds), a more complex and proper encoding is required for the network to process them.

In this chapter, a more complex and biologically plausible system is developed from an extension on the previous simple system. As we know, primates perform remarkably well in cognitive tasks such as pattern recognition. Motivated from recent findings in biological systems, a unified and consistent feedforward system network with a proper encoding scheme and supervised temporal rules is built for processing real-world stimuli. The temporal rules are used for processing the spatiotemporal patterns. To utilize these rules on images or sounds, a proper encoding method and a unified computational model with consistent and efficient learning rule are required. Through encoding, external stimuli are converted into sparse representations which also have properties of invariance. These temporal patterns are then learned through biologically derived algorithms in the learning layer, followed by the final decision presented through the readout layer. The performance of the model is also analyzed and discussed.

3.1 Introduction

Primates are remarkably good at cognitive skills such as pattern recognition. Despite decades of engineering effort, the performance of the biological visual system still outperforms the best computer vision systems. Pattern recognition is a general task that assigns an output value to a given input pattern. It is an information-reduction process which aims to classify patterns based on a priori knowledge or statistical information extracted from the patterns. Typical applications of pattern recognition includes automatic speech recognition, handwritten postal codes recognition, face recognition and gesture recognition. There are several conventional methods to implement pattern recognition, such as maximum entropy classifier, Naive Bayes classifier, decision trees, support vector machines (SVM) and perceptrons. We refer these methods as traditional rules since they are less biologically plausible compared to spiking time involved rules described later. Compared to human brain, these methods are far from reaching comparable recognition. Humans can easily discriminate different categories within a very short time. This motivates us to investigate computational models for rapid and robust pattern recognition from a biological point of view. At the same time, inspired by biological findings, researchers have come up with different theories of encoding and learning. In order to bridge the gap between those independent studies, a unified systematic model with consistent rules is desired.

A simple feedforward architecture might account for rapid recognition as reported recently [22]. Anatomical back projections abundantly appear almost every area in the visual cortex, which puts the feedforward architecture under debate. However, the observation of a quick response appeared in inferotemporal cortex (IT) [80] most directly supports the hypothesis of the feedforward structure. The activity of neurons in monkey IT appears quite soon (around 100 ms) after stimulus onset [81]. For the purpose of rapid recognition, a core feedforward architecture might be a reasonable theory of visual computation.

How information is represented in the brain still remains unclear. However, there are strong reasons to believe that using pulses is the optimal way to encode the information for transmission [52]. Increasing number of observations show that neurons in the brain precisely response to a stimulus. This support the hypothesis of the temporal coding.

There are many temporal learning rules proposed for processing spatiotemporal patterns, including both supervised and unsupervised rules. As opposed to the unsupervised rule, a supervised one could potentially facilitate the learning speed with the help of an instructor signal. Although so far there is no strong experimental confirmation of the supervisory signal, an increasing body of evidence shows that this kind of learning is also exploited by the brain [63].

Learning schemes focusing on processing spatiotemporal spikes in a supervised manner have been widely studied. With proper encoding methods, these schemes could be applied to image categorization. In [30], the spike-driven synaptic plasticity mechanism is used to learn patterns encoded by mean firing rates. A rate coding is used to encode images for categorization. The learning process is supervised and stochastic, in which a teacher signal steers the output neuron to a desired firing rate. According to this algorithm, synaptic weights are modified upon the arrival of pre-synaptic spikes, considering the state of post-synaptic neuron's potential and its recent firing activity. One of the major limitations of this algorithm is that it could not be used to learn patterns presented in the form of precise timing spikes. Different from the spike-driven synaptic plasticity, the tempotron learning rule [33] is efficient to learn spike patterns in which information is embedded in precise timing spikes as well as in mean firing rates. This learning rule modifies the synaptic weights such that a trained neuron fires once for patterns of corresponding category and keeps silent for patterns of other categories. The ReSuMe learning rule [37, 47] is also a supervised rule in which the trained neuron can fire at desired times when corresponding spatiotemporal patterns are presented. It has been demonstrated that the tempotron rule and the ReSuMe rule are equivalent under certain conditions [82].

Although spiking neural networks (SNNs) show promising capabilities in achieving a performance similar to living brains due to their more faithful similarity to biological neural networks, one of the main challenges of dealing with SNNs is getting data into and out of them, which requires proper encoding and decoding methods. The temporal learning algorithms are based on spatiotemporal spike patterns. However, the problem remains how to represent real-world stimuli (like images) by spatiotemporal spikes for further computation in the spiking network. To deal with these problems, a unified systematic model, with consistent encoding, learning and readout, is required.

The main contribution of this chapter lies in the design of a unified systematic model of spiking neural network for solving pattern recognition problems. To the best of our knowledge, this is the first work in which complex classification task is solved through combination of biologically plausible encoding and supervised temporal learning. The system contains consistent encoding, learning and readout parts. Through the network, we fill the gap between real-world problem (image encoding) and theoretical studies of different learning algorithms for spatiotemporal patterns. Finally, our approach suggests a plausibility proof for a class of feedforward models of rapid and robust recognition in the brain.

3.2 The Spiking Neural Network

In this section, the feedforward computational model for pattern recognition is described. The model composes of 3 functional parts: the encoding part, the learning part and the readout part. This structure is similar to the one in

Figure 2.1, but with a more complex structure for encoding and readout.

Considering the encoding, the latency code is a simple example of temporal coding. It encodes information in the timing of response relative to the encoding window, which is usually defined with respect to the stimulus onset. The external stimuli would trigger neurons to fire several spikes in different times. From biological observations, visual system can analyze a new complex scene in less than 150 ms [23,83]. This period of time is impressive for information processing considering billions of neurons involved. This suggests that neurons exchange only one or few spikes. In addition, it is shown that subsequent brain region may learn more and earlier about the stimuli from the time of first spike than from the firing rate [23].

Therefore, we use single spike code as the encoding mechanism. Within the encoding window, each input neuron fires only once. This code is simple and efficient, and the capability of encoding information in the timing of single spikes to compute and learn realistic data has been shown in [53]. Compared to rate coding as used in [30], this single spike coding would potentially facilitate computing speed since less spikes are involved in the computation.

Our single spike coding is similar to the rank order coding in [84, 85] but taking into consideration of the precise latency of the spikes. In the rank order coding, the rank order of neurons' activations is used to represent the information. This coding scheme is still under research. Taking the actual neurons' activations into consideration but not their rank orders, our proposed encoding method could convey more information than the rank order coding. Since this coding utilizes only a single spike to transmit information, it could also potentially be beneficial for efficient very large scale integration (VLSI) implementations.

In the learning layer, supervised rules are used since they could improve the learning speed with the help of the instructor signal. In this chapter, we investigate the tempotron rule and the ReSuMe rule.

The aim of the readout part is to extract information about the stimulus from the responses of learning neurons. As an example, we could use a binary sequence to represent a certain class of patterns in the case that each learning neuron can only discriminate two groups. Each learning neuron responds to a stimulus by firing (1) or not firing (0). Thus, the total N learning neurons as the output can represent a maximum number of 2^N classes of patterns.

A more suitable scheme for readout would be using population response. In this scheme, several groups are used and each group, containing several neurons, is one particular representation of the external stimuli. Different groups compete with each other by a voting scheme in which the group with the most amount of firing neurons would be the winner. This scheme is more compatible with the real brain since the information is presented by the cooperation of a group of neurons rather than one single neuron [86].

3.3 Single-Spike Temporal Coding

We have mentioned the function of the encoding layer is to convert stimulus into spatiotemporal spikes. In this section, we illustrate our encoding model of single-spike temporal coding, which is inspired from biological agents.

The retina is a particular interesting sensory area to study neural information processing, since its general structure and functional organization are remarkably well known. It is widely believed that information transmitted from retina to brain codes the intensity of the visual stimuli at every place in visual field. The ganglion cells (GCs) collect the information from their receptive fields which could best drive spiking responses [87]. In addition, different ganglion cells might have overlapped centers of receptive fields [88]. A simple encoding model of retina is described in [84] and is used in [85]. The GCs are used as the first layer in our model to collect information from original stimuli.

Focusing on emulating the processing in visual cortex, a realistic model (HMAX) for recognition has been proposed in [89] and widely studied [22, 90, 91]. It is a hierarchical system that closely follows the organization of visual cortex. The HMAX performs remarkably well with natural images by using alternate simple cells (S) and complex cells (C). Simple cells (S) gain their selectivity from a linear sum operation, while complex cells (C) gain invariance through a nonlinear max pooling operation. Like the HMAX model, in order to obtain an invariant encoding model to some extent, a complex cells (CCs) layer is used in our model. In the brain, equivalents of CCs may be in V1 and V4 (see [92] for more details).

In our model (see Figure 3.1), the image information (intensity) is transmitted to GCs through photo-receptors. Each GC linearly integrates at its soma the information from its receptive field. Their receptive fields are overlapping and their scales are generally distributed non-uniformly over the visual field. DoG (difference of gaussian) filters are used in the GCs layer since

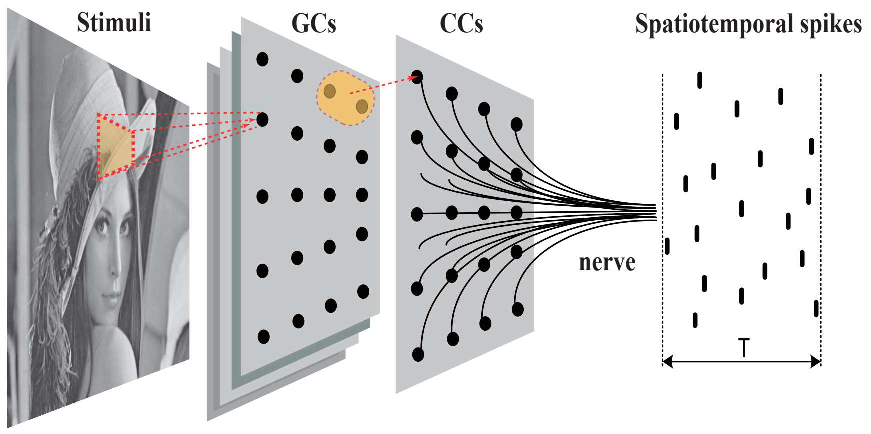

Figure 3.1: Architecture of the visual encoding model. A gray-scale image (as the stimuli) is presented to the encoding layer. The photo-receptors transmit the image information analogically and linearly to the corresponding ganglion cells (GCs). Each ganglion cell collects information from its receptive field (an example shown as the red dashed box). There are several layers of GCs and each has a different scale of receptive field. The complex cells (CCs) collect information from a local position of GCs and a MAX operation among these GCs determines the activation value of CC unit. Each CC neuron would fire a spike according to their activations. These spikes are transmitted to the next layer as the spatiotemporal pattern in particular time window (T).

this filter is believed to mimic how neural processing in the retina of the eye extracts details from external stimuli [93, 94]. Several different scales of DoG would construct different GCs images. The CCs unit would operate a nonlinear max pooling to obtain an amount of invariance. Max pooling over the two polarities, different scales and different local positions provides contrast reverse invariance, scale invariance and position invariance, respectively. Biophysically plausible implementations of the MAX operation have been proposed in [95], and biological evidences of neuron performing MAX-like behavior have been found in a subclass of complex cells in V1 [96] and cells in V4 [97].

The activation value of CC unit would trigger a firing spike. Strongly activated CCs will fire earlier, whereas weakly activated will fire later or not at all. The activation of the GC is computed through the dot product as:

$$GC_i := < I, \phi_i > = \sum_{l \in R_i} I(l) \cdot \phi_i(l) \tag{3.1}$$

where $I(l)$ is the luminance of pixel l which is sensed by the photo-receptor. R_i is the receptive field region of neuron i. ϕ_i is the weight of the filter.

The GCs compute the local contrast intensities at different spatial scales and for two different polarities: ON- and OFF-center filters. We use the simple DoG as our filter where the surround has three times the width of the center. The DoG has the form as:

$$DoG_{\{s,l_c\}}(l) = G_{\sigma(s)}(l - l_c) - G_{3 \cdot \sigma(s)}(l - l_c) \tag{3.2}$$

$$G_{\sigma(s)}(l) = \frac{1}{2\pi \cdot \sigma(s)^2} \cdot exp(-\frac{\|l\|^2}{2 \cdot \sigma(s)^2}) \tag{3.3}$$

where $G_{\sigma(s)}$ is the 2D Gaussian function with variance $\sigma(s)$ which depends on the scale s. l_c is the center position of the filter.

An example of the DoG filter is shown in Figure 3.2. An OFF-center filter is simply an inverted version of an ON-center receptive field. All the filters are sum-normalized to zero and square-normalized to one so that when there is no contrast change in the image the neuron's activation would be zero and when the image is same with the filter the neuron's activation would be 1. Therefore, all the activations of the GCs are scaled to the same range ([-1, 1]).

The CCs max over different polarities according to their absolute values at same scale and same position. Through this max operation, the model gain a contrast reverse invariance (Figure 3.3a). From the property of the polar filters, only one could be positive activated for a given image. Similarly, the

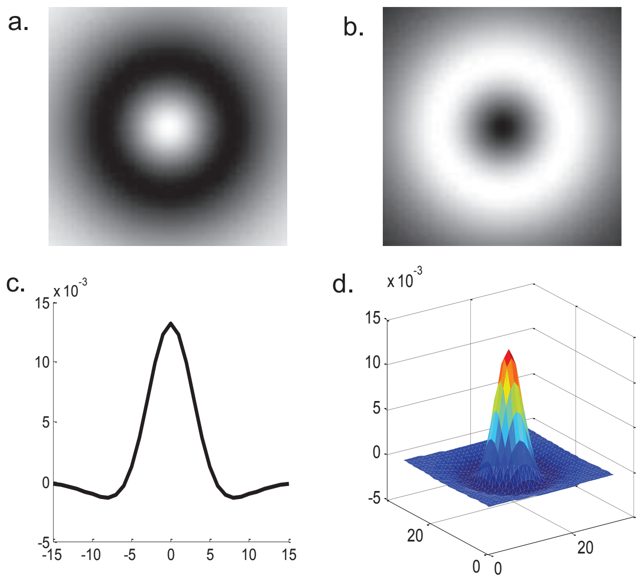

Figure 3.2: Linear filters in retina. (a) is an image of the ON-center DoG filter, whereas (b) is an image of the OFF-Center filter. (c) is the one-dimensional show of the DoG weights and (d) is the 2-dimensional show.

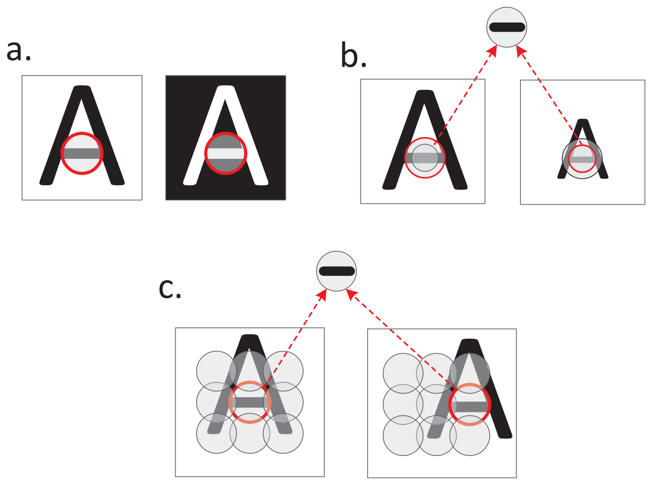

Figure 3.3: Illustration of invariance gained from max pooling operation. (a) the contrast reverse invariance by max pooling over polarities. (b) the scale invariance by max pooling over different scales. (c) the local position invariance by max pooling over local positions. The red circle denotes the maximally activated one.

scale invariance is increased by max pooling over different scales at the same position (Figure 3.3b). Finally, the position invariance is increased by pooling over different local positions (Figure 3.3c). The dimension of images is reduced since only the max activated value in a local position is preserved.

Figure 3.4 shows the basic processing procedures in different encoding layers. Through the encoding, the original image is sparsely presented in the CCs (Figure 3.4f).

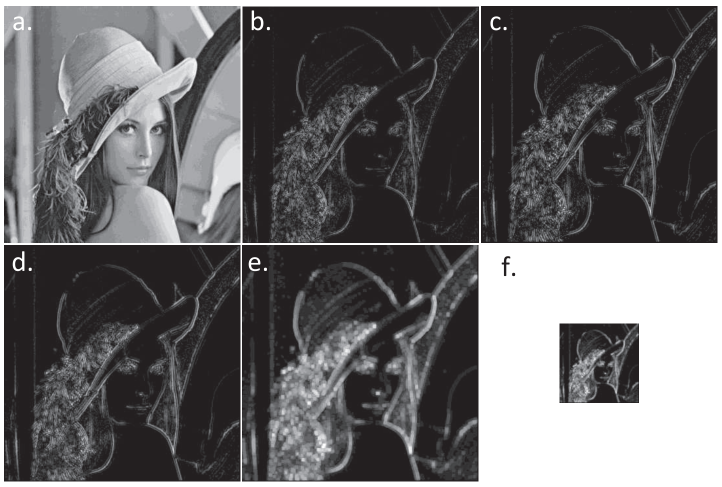

Figure 3.4: Illustration of the processing results in different encoding procedures. (a) is the original external stimulus. (b) and (c) are the processing results in layer GCs with different scales. (d), (e) and (f) are the processes in the CCs layer. (d) is the result of max pooling over different scales. (e) is max pooling over different local positions. (f) is the sub-sample from (e).

The final activations of CCs are used to produce spikes. Strongly activated neurons would fire earlier, whereas weakly activated ones would fire later or not at all. The spike latencies are then linearly mapped into a predefined encoding time window. These spatiotemporal spikes are transmitted to the next layer for computation.

In our encoding scheme, we consider the actual values of neurons' activations to generate spikes but not the rank order of these activations as used in [84, 85]. This could carry more information than the rank order coding which only considers the rank order of different activations and ignores the exact differences between different activations. For example, there are 3 neurons (n_1,n_2 and n_3) having their activations (C_1, C_2 and C_3) in the range of [0,1]. Pattern P_1 is represented by ($C_1 = 0.1$, $C_2 = 0.3$ and $C_3 = 0.9$); pattern P_2 is represented by ($C_1 = 0.29$, $C_2 = 0.3$ and $C_3 = 0.32$). In rank order coding, it will treat P_1 and P_2 as same patterns since the rank orders are same. For our encoding, in contrast, P_1 and P_2 would be treated as totally different patterns. In addition, the rank order coding would be very sensitive to the noise since the encoding time of one neuron depends on other neurons' rank. For example, if the least activated value is changed to a max activated value because of a disturbance, the rank of all the other neurons would be changed. However in our proposed algorithm only the information of the disturbed neuron would be affected.

3.4 Temporal Learning Rule

Temporal learning rule aims at dealing with information encoded by precise timing spikes. In this section, we consider supervised mechanisms like the tempotron rule and the ReSuMe rule that could be used for training neurons to discriminate between different spike patterns. Whether a LTP or LTD process occurs depends on the supervisory signal and the neuron's activity. This kind of supervisory signal can facilitate the learning speed compared to the unsupervised method.

3.4.1 The Tempotron Rule

In Chapter 2, the tempotron rule is introduced in detail including neuron dynamics and plasticity. For the reason of simplicity and clearance, we briefly introduce it here again.

In binary classification problem, each input pattern presented to the neuron belongs to one of two classes (which are labeled by P^+ and P^-). One neuron can make decision by firing or not. When a P^+ pattern is presented to the neuron, it should elicit a spike; when a P^- pattern is presented, it should keep silent by not firing. The tempotron rule modifies the synaptic weights (w_i) whenever there is an error. This rule performs like gradient-descent rule that minimizes a cost function as:

$$C = \begin{cases} V_{thr} - V_{t_{max}}, & \text{if the presented pattern is } P^+; \\ V_{t_{max}} - V_{thr}, & \text{if the presented pattern is } P^-. \end{cases} \tag{3.4}$$

where $V_{t_{max}}$ is the maximal value of the post-synaptic potential V.

Applying the gradient descent method to minimize the cost leads to the tempotron learning rule (refer to Chapter 2 for more details).

3.4.2 The ReSuMe Rule

The ReSuMe described in [47] is a supervised method that aims to produce desired spike trains in response to the given input sequence. According to this rule, the synaptic weights are modified according to the following equation:

$$\frac{d\omega_i(t)}{dt} = \lambda[S^d(t) - S^{out}(t)][a + \int_0^\infty W(s)S^i(t-s)ds] \tag{3.5}$$

where λ is the learning rate, a is a constant, W is a learning window with a exponential form ($W(s) = Ae^{-s/\tau_E}$). $S^d(t)$, $S^{out}(t)$ and $S^i(t)$ are the target, post- and pre-synaptic spike trains, respectively. Although the shape of learning window is not restricted to exponential form, this shape can result in a better performance of convergence [98]. The spike trains have the following form:

$$S(t) = \sum_{f=1}^{n} \delta(t - t^f) \tag{3.6}$$

where t^f denotes the moment of the f-th spike in the train, n denotes the total number of spikes in the train, $\delta(x)$ is the impulse function $\delta(x) = 1$ if $x = 0$ (or 0 otherwise).

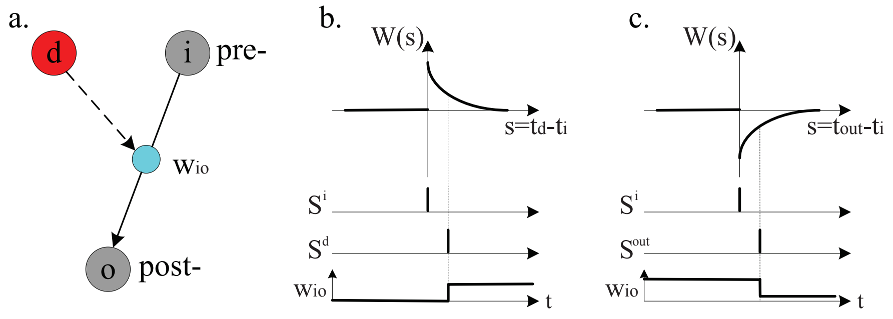

Figure 3.5: Illustration of the ReSuMe learning rule. (a) demonstrates that the synaptic plasticity depends on the correlation between the pre- and postsynaptic firing times, and on the correlation between pre- and desired firing times. (b) demonstrates that the synaptic weight is potentiated whenever a desired spike is observed. (c) shows that the synaptic weight is depressed whenever the trained neuron fires. This figure is revised from [37].

Figure 3.5 illustrates the ReSuMe learning rule. The synaptic efficacy depends not only on the correlation between the pre-synaptic and post-synaptic firing times but also on the correlation between the pre-synaptic and desired firing times. A desired spike would result in synaptic potentiation, and a post-synaptic spike would result in synaptic depression.

After a learning trial, the total synaptic change is:

$$\begin{aligned}\Delta\omega_i =&\lambda a(n^d - n^{out}) + \lambda\Sigma_{t^d}\Sigma_{t_i\leq t^d}W(t^d - t_i) \\ &- \lambda\Sigma_{t^{out}}\Sigma_{t_i\leq t^{out}}W(t^{out} - t_i)\end{aligned} \tag{3.7}$$

where n^d and n^{out} are the number of spikes from the desired and the actual output spike trains respectively. t_i is the pre-synaptic spike time.

The ReSuMe rule could be used for both the batch learning and the online learning.

3.4.3 The Tempotron-like ReSuMe Rule

As proposed in [82], the tempotron learning rule is a particular case of ReSuMe rule under certain conditions. The rule discussed here is a connection between the tempotron rule and the ReSuMe rule.

Considering to apply ReSuMe to the tempotron setup, the combined rule can be approached. The neuron is only allowed to fire once or not. After a spike is emitted, the neuron shunts all its incoming spikes immediately. If there is only one spike, regardless of its time, it is reasonable to consider the neuron firing at t_{max}. This learning rule follows [82]:

$$\Delta w_i = \begin{cases} \lambda a + \lambda\sum_{t_i\leq t_{max}} W(t_{max} - t_i), & \text{if } n^d = 1, n^{out} = 0; \\ -\lambda a - \lambda\sum_{t_i< t_{out}} W(t_{out} - t_i), & \text{if } n^d = 0, n^{out} = 1; \\ 0, & \text{if } n^d = n^{out}. \end{cases} \tag{3.8}$$

When $a = 0$ and $W(s) = K(s)$, the combined rule is equivalent to the tempotron learning rule. This implicates that the tempotron rule is a particular case of the ReSuMe rule.

3.5 Simulation Results

In this section, several simulations are performed to test the performance of the network and different learning rules.

3.5.1 The Data Set and The Classification Problem

The stimuli from real world typically have a complex statistical structure. It is quite different from idealized case of random patterns often considered. In the real world, the stimuli hold large variability in a given class and have a high level of correlation between members of different classes. The data set we considered here is the MNIST digits (see Figure 3.6).

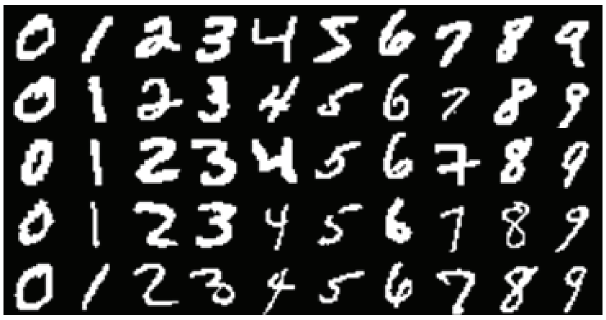

Figure 3.6: Examples of handwritten digits from MNIST dataset.

The MNIST data set contains a large number of examples of hand-written digits, which consists of ten classes (digits 0 to 9) of examples and each example is an image of 28 $\times$ 28 pixels. The MNIST data set is available from http://yann.lecun.com/exdb/mnist, where many classification results from different methods are also listed. All images from this data set are gray-scale.

3.5.2 Encoding Images

Each image is presented to the encoding layer, and is then converted into spatiotemporal pattern. We use the coding strategy discussed previously through which the output is sparse, as is observed in biological agents [99].

For simplicity of applying the encoding algorithm to the data set, we distribute GCs with different receptive fields all over the image (each pixel). The image size in GCs is same as the input image. Considering examples of 28-by-28 images, we choose two scales for the filters ($\sigma = 1$ for 5×5 pixels as scale 1, and $\sigma = 2$ for 7×7 pixels as scale 2). The CCs layer performs the max pooling operation on the previous GCs layer. For local position operation we choose 6×6 pixels and we set the overlap pixels to be 3 in one axis (x or y) for sub-sampling operation. A detailed process of max operation is described in [89].

The application of all these processes produces a set of analog values, corresponding to the activation levels of our CCs unit. The strongly activated cell will fire earlier, whereas the weakly activated will fire later or not at all. The spike latencies are linearly mapped into a predefined encoding time window (100 ms in this study). The activation values are linearly converted to delay times, associating $t = 0$ with activation value 1 and later times up to $100\ ms$ with lower activation values. The neurons with activation value of 0 (or below a chosen small value) will not fire due to the weak activation.

An illustration of encoding an image is shown in Figure 3.4. Our scheme is to extract the basic information and encode it to a spatiotemporal spike pattern. Through the whole encoding structure, a sparse representation of the original incoming image is finally obtained. Using this sparse representation to generate the spike pattern would, to some extent, be compatible with biological observations in retina.

3.5.3 Choosing Among Temporal Learning Rules

In the tempotron rule, we specify the following parameters. The ratio between the membrane and the synaptic constants is fixed at $\tau_m/\tau_s = 4$. The threshold V_{thr} is set to 1 and V_{rest} is set to 0. We use $\tau_m = 10\ ms$ and $\lambda = 0.002$.

For comparison purpose, in the ReSuMe we use the similar neuron model as the one in the tempotron rule. However, the difference is that when the neuron emits a spike, its potential is reset to a rest value (0 here) and is hold there for a refractory period (3 ms here).

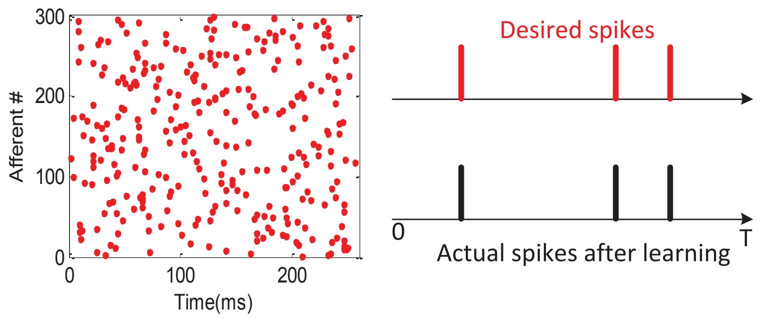

Figure 3.7: Illustration of the suitability of ReSuMe rule for the chosen neuron model. The input pattern contains 300 afferent synapses and each fires once only. These spikes are generated randomly with uniform distribution. For the desired spike, 3 random spiking times are chosen.

Since the ReSuMe rule is based only on the spiking times, it could work independently on the used spiking neuron models [47]. To verify the suitability of this rule for our chosen neuron model, we generate a spike pattern and force the neuron to respond at desired times. We choose 300 afferent synapses and each fires only once in the time window. The timing of each spike is generated randomly with uniform distribution between 0 and T. After learning, the neuron could perform as the desired way (see Figure 3.7).

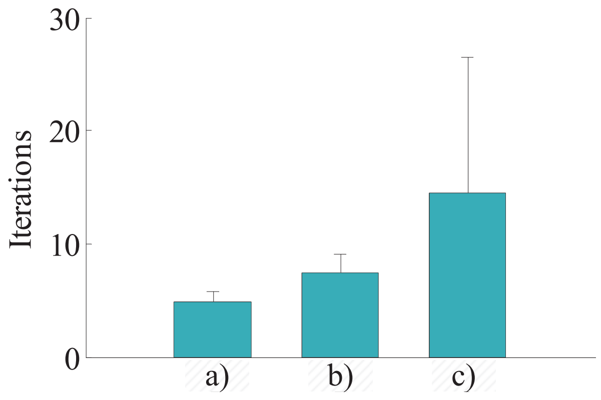

Figure 3.8: The number of iterations needed for the correct classification of spike patterns, through different learning rules. (a) is the tempotron learning rule. (b) is the tempotron-like ReSuMe rule. (c) is the ReSuMe rule in which if the neuron fires, it should spike at a desired time. Over 100 experiments with different initial conditions, the averages (4.95, 7.36 and 14.48) and standard deviations (0.8454, 1.7438 and 12.014) are obtained for (a),(b) and (c), respectively.

To compare the learning speed of different learning rules, we generate 30 spatiotemporal patterns and each pattern contains 120 afferent synapses. The spiking times are generated randomly with a uniform distribution between 0 and T. We randomly choose 3 patterns as one category that is needed to be discriminated from others. We record the minimum times of iterations for different rules to learn these patterns correctly. We perform this experiment for 100 times and the results are shown in Figure 3.8.

According to Figure 3.8, there is no significant difference of learning speed between the tempotron rule and tempotron-like ReSuMe rule. This is because the only difference between these two rules is the kernel windows which have a similar effect on the synaptic change. However, compared to the ReSuMe rule, the tempotron rule is much faster (about 3 times as the ReSuMe rule). Besides this, the learning speed of the ReSuMe varies significantly for different initial conditions (such as the number of patterns, the initial weights and the learning rate). For the sake of fast recognition, we choose the tempotron rule as our learning rule.

3.5.4 The Properties of Tempotron Rule

Since the tempotron rule is chosen, a test on its properties is needed.

Capacity

As is used for perceptron [61], the ratio of the number of random patterns (N_p) that correctly classified by the neuron over the number of its synapses (N_{in}), $\alpha = N_p/N_{in}$, is used to measure the load of the neuron. An important characteristic of neuron's capacity is the maximum load that it can learn. As studied in [33], the maximum recognition load of a tempotron can reach 3 approximately, which means that the number of patterns the neuron can learn could roughly approach to 3 times the number of synapses connected to it.

For our chosen neuron, a test on its load is shown in Figure 3.9. We set $N_{in} = 100$ and generate different number of spike patterns within a fixed time window ($T = 100\ ms$). Each afferent fires only once and the spiking time is randomly chosen from uniform distribution within T. The mean number of cycles of pattern presentations for error-free classification is shown versus the load (α). Although a more robust estimation of the load is feasible by

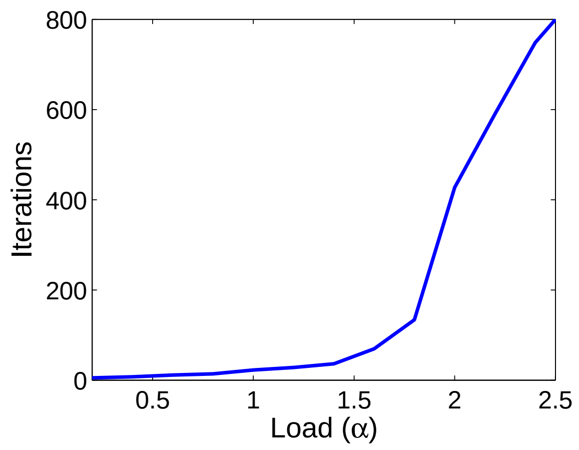

Figure 3.9: The mean number of iterations of pattern presentations for error-free classification versus neuron load. The patterns are randomly generated within the fixed time window (100ms). The number of synapses is 100. Data are averaged over 20 runs.

allowing a small percentage of false alarms, the rigorous condition of error-free classification is useful to testify the neuron's ability of classifying all assigned patterns successfully.

According to Figure 3.9, the neuron could successfully learn the patterns within several tens of iterations if the load is not very high (below 1.5), but the number of iterations would increase sharply when the load is over 1.5. This means that under a higher load the neuron needs more time to learn the patterns or the learning process could never converge.

This load test, to some extent, could guarantee the learning convergence when the tempotron neuron is applied to our chosen recognition task. In our task, there are only ten categories and patterns in each category share some common features. Compared to the randomly generated patterns, the neuron's capacity might be sufficient to learn these real-world stimuli.

Robustness

In some cases, the external noise might change the encoded spike patterns more or less. The tempotron rule should hold some level of robustness to tolerate the noise. To assess the robustness of the learning rule, we trained the neuron with a number of patterns ($\alpha = 1$). Then we tested the performance of the neuron when facing with jittered versions of previous learned patterns. The jittered pattern was generated by adding a Gaussian noise to all spike times of a template pattern. The robust performance of the neuron is shown in Figure 3.10.

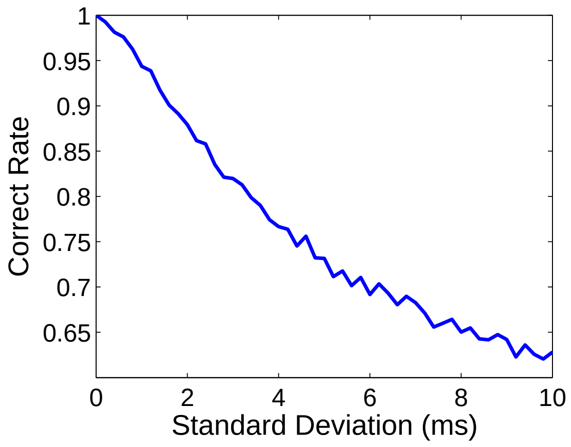

Figure 3.10: The mean correct rate of classification on jittered spike patterns. The jittered pattern is generated by adding Gaussian noise with standard deviation to all spike times of a template pattern.

According to Figure 3.10, the performance of correct recognition decreases with increasing jitter. Within a limited jitter range (0-3 ms), the performance stays in a relatively high level (over 0.8). This indicates the learning rule is robust to the presence of temporal noise to some extent.

3.5.5 Recognition Performance

The combined system is applied to recognize different patterns. To see the ability of our system network on the recognition task, we use a small data set from the MNIST (50 digits and 5 for each category). And we choose four neurons as the readout. We call this readout as the fully distributed scheme with no redundancy (each neuron codes for one bit). After several iterations of training, the network can recognize all the patterns in this data set. Here, we take the recognition results of several digits as an example (Figure 3.11). If the potential of the learning neuron crosses the threshold, namely it fires, the value of this neuron is considered as 1, otherwise it is 0. In Figure 3.11, when image "0" shows up to the network, none of the learning neurons fire, so the result is $[0000]$. For image "3", the result is $[0011]$, and for "9" it's $[1001]$. This indicates that the tempotron rule applied in our model could recognize different classes of images successfully.

However, using only four neurons as the readout in a binary format might be very sensitive to changes of input images, especially considering the real-world stimuli in which samples hold large variability in a given class and overlap with members in different classes. If only one neuron misclassified the incoming pattern while others correctly responded, the pattern was still wrongly classified. Researchers have found that neighboring neurons have similar response properties. Depending on this, neural groups are used for assembly computing [86].

Thus, we use several grouped pools as our readout. We firstly consider a

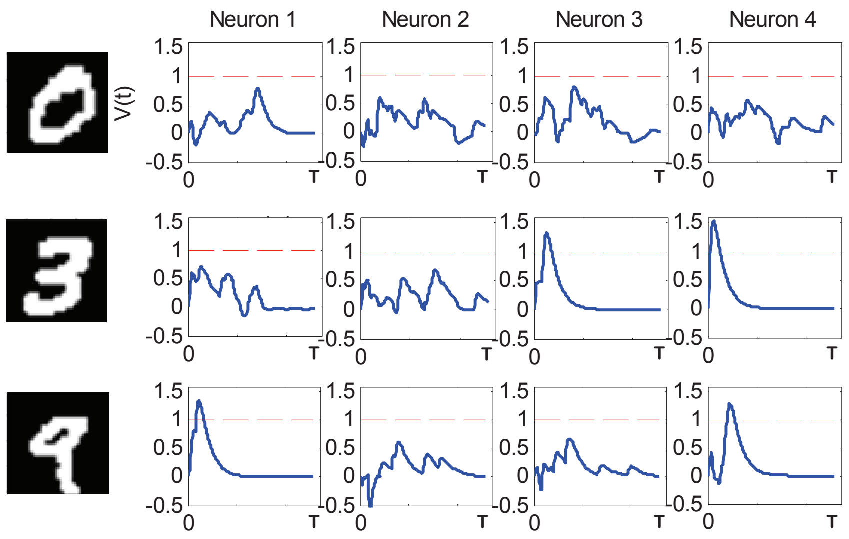

Figure 3.11: Recognition results of digits by Tempotron learning rule. Here shows 4 learning neurons (Neuron 1 to 4) and 3 images. The neuron responds to an image by firing (1) or not (0). The results for “0”, “3” and “9” are [0000], [0011] and[1001], respectively.

distributed code with redundancy: 4 pools of 20 neurons each. Each pool codes for one binary feature as in Figure 3.11. A voting system decides if the binary feature is 0 or 1 based on the voting majority in this pool. Then we consider a localist scheme with redundancy, where each pool of 20 neurons codes for only one category. For an incoming stimulus, it is classified into a category according to the pool which has the most amount of voting neurons fired. If two or more pools have the same maximal firing number, the incoming stimulus is classified as unknown pattern.

These two schemes of readout with redundancy are used. For cross-validation, we choose 500 digits (50 images for each category) as our training set and randomly choose other 100 images from the MNIST data set as the testing set. In the training phase, each neuron is trained with a sub-training set chosen from the training set. This sub-training set consists of examples randomly chosen from the corresponding category and also other categories. After training, the performance is tested on both training set and testing set. The correct rate on the testing set is around 50% for the distributed code with no redundancy, and is around 79% for the localist code with redundancy. For distributed code, although the robustness for coding one bit feature is improved comparing to single neuron code, it is still not comparable to the localist one. This is due to that in the distributed code the final decision highly depends on correct reaction of each pool, but in the localist code it only depends on a correct major voting of one corresponding pool. Thus, in the localist scheme, the robustness is not only due to the redundancy but also to the localist aspect. This localist scheme is considered in our following experiments.

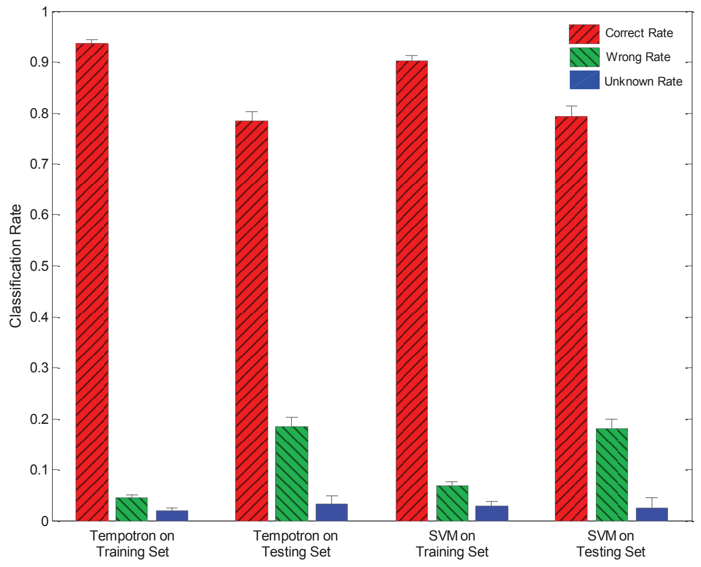

Figure 3.12: The classification performance of tempotron and SVM. The system is trained 40 times each for tempotron and SVM. After each training time, the generalization is performed on both the training and testing set. The averages and standard deviations are plotted.

To make a comparison with the benchmark machine learning method,

Table 3.1: The classification performance of tempotron and SVM on MNIST

Percentage(%)	Tempotron Rule		SVM	
	Training	Testing	Training	Testing
Correct Rate	93.67 ± 0.67	78.5 ± 1.85	90.24 ± 0.98	79.33 ± 2.03
Wrong Rate	4.48 ± 0.58	18.35 ± 1.85	6.88 ± 0.78	18.15 ± 1.69
Unknown Rate	1.86 ± 0.61	3.15 ± 1.64	2.89 ± 0.86	2.53 ± 2.04

SVMs are chosen to perform the classification on the CCs activation values. Since SVM also has a binary decision behavior, we set the same classification condition on training and testing as for tempotron. The performances of both the tempotron and SVM on the training set and testing set are shown in Figure 3.12. The corresponding recognition rates are shown in Table 3.1.

According to Figure 3.12, our network with the tempotron rule performs at a high correct rate (around 93.7%) on the training set and at an acceptable correct rate (around 79%) on the testing set, especially considering the small data set (500 images) used for training. Comparing with SVM under the same condition of our encoding model, the performances of spiking neurons are better than SVM for the training set and comparable to SVM for the testing set. From a biological point of view, our system attempts to perform robust and rapid recognition with a brain-like architecture.

To investigate the states of the spiking neurons in one grouped pool after learning, a picture of the average weights is shown in Figure 3.13. According to Figure 3.13, the grouped neurons, cooperating together, roughly grab a general and basic feature of the learned category. Taking digit 0 as an example, the center weights are mostly inhibited since these neurons are rarely activated by the incoming stimulus 0 through our encoding model.

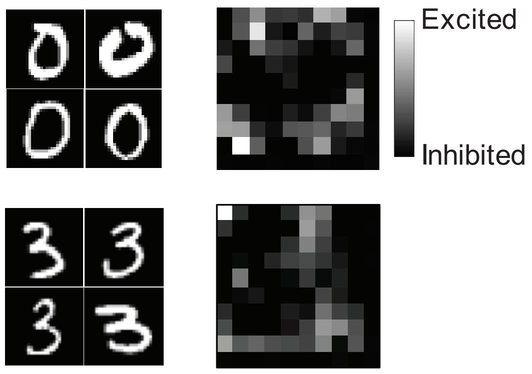

Figure 3.13: Average weights of the spiking neurons in the pool representing digit 0 and 3. Left: Image samples of digit 0 and 3 from MNIST are listed. Right: The picture of the average weight of the spiking neurons in corresponding group. Inhibited afferents are plotted black, while excited ones are plotted gray-scale according to their weights.

3.6 Discussion

Discussions on the proposed system are given as follows.

Encoding Benefits from Biology

Through the layers of GCs and CCs the external stimuli are sparsely represented in the activation values of CCs units. These activation values are used to generate spiking patterns in a time domain. It already has been shown that coding schemes based on the firing rates are unlikely to be efficient enough for fast information processing [84, 100]. Considering the rapid processing in the brain and billions of neurons involved, a temporal code which uses single spikes is, in principle, capable of carrying substantial information about the external stimuli [23] and facilitating the computational speed. In several sensory systems, shorter latencies of spikes result from stronger stimulation [101, 102]. In our encoding layer, the strongly activated neurons would fire earlier, whereas the weakly activated neurons would fire later or not at all. The chosen encoding window of the temporal patterns is on a scale of hundreds of milliseconds, which matches the biologically experimental results as mentioned in [11, 13, 26]. In addition, our encoding is efficient and the spiking output is sparse as observed in biological retinas [99, 103].

Types of Synapses

The types of synapses are determined by the signs of their efficacies, with positive values corresponding to excitatory synapses and negative values to inhibitory synapses. Although this model is far from biological realism, it is proved to be a useful computational approach [47]. In the neuron model, the sign of synapse could change by learning. The learning also works when the signs of synapses are not allowed to change, but the capacity is reduced. For a practical usage for multiple-class problem, changing sign is allowed in the neuron model. This can be realized by altering the balance between excitatory and inhibitory pathways [33].

Schemes of Readout

Using a binary version of readout, the network is shown to be capable to finish a simple recognition task on a small data set. However, this kind of readout would be very sensitive to each neuron's performance in the readout. If only one neuron misclassifies the pattern while others do a correct classification, the final readout would also be wrong since it depends on all the neurons in a binary form. Using grouped pools could effectively compensate this. In nervous systems such as visual cortical areas [104] and hippocampus [105], information is commonly expressed through populations or clusters of cells rather than through single cell [106]. This strategy is robust since damage to a single cell will not have a catastrophic effect on the whole population. Through learning, neurons in the same group try to find the common features discriminating that category, and through voting, the most active group would be chosen. Another meaningful aspect of the readout is that there is an unknown decision. Since some samples in one category are quite similar to other categories (for example the digit “5” in the second row of Figure 3.6), it is reasonable to label them as unknown rather than wrong. A further processing could be done for these unknown samples.

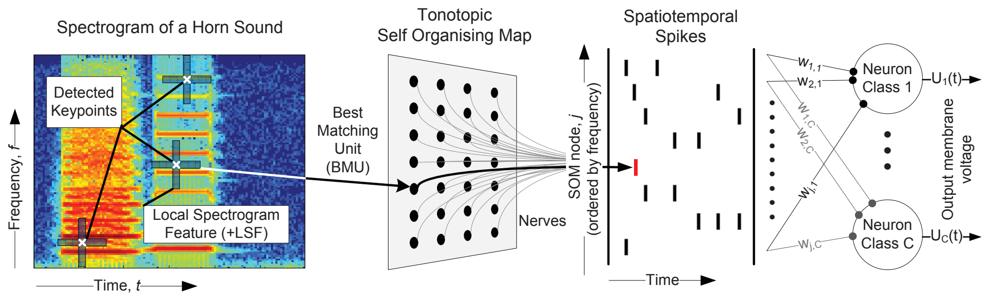

Figure 3.14: The proposed LSF-SNN system for sound recognition. Firstly the keypoints are detected and the corresponding LSFs are extracted. Then, the SOM map is used to produce the output spatiotemporal spike patterns. These patterns are then learnt by the tempotrons for recognition.

Extension of the Network for Robust Sound Recognition

In addition to the recognition on images, we also proposed a SNN for recognizing sounds. The general structure remains the same, where functional parts of encoding, learning and readout are involved. The major difference of the two systems is the encoding part. With a proper encoding scheme for sounds, the SNN can perform the recognition well. We propose a novel approach based on the temporal coding of Local Spectrogram Features [44], which generates spikes that are used to train the following neurons. The general structure for sound recognition is shown in Figure 3.14. Our experiments demonstrate the robust performance of this system across a variety of noise conditions, such that it is able to outperform the conventional frame-based baseline methods. More details can be found in [44].

3.7 Conclusion

A systematic computational model by using consistent temporal encoding, learning and readout has been presented to explore brain-based computation especially in the regime of pattern recognition. It is a preliminary attempt to perform rapid and robust pattern recognition from a biological point of view. The schemes used in this model are efficient and biologically plausible. The external stimuli are sparsely represented after our encoding and the representations have properties of selectivity and invariance. Through the network, the temporal learning rules can be applied to processing real-world stimuli.

Chapter 4

Precise-Spike-Driven Synaptic Plasticity

This chapter proposes a new temporal learning rule, namely the Precise-Spike-Driven (PSD) Synaptic Plasticity, for processing and memorizing spatiotemporal patterns. PSD is a supervised learning rule that is analytically derived from the traditional Widrow-Hoff rule and can be used to train neurons to associate an input spatiotemporal spike pattern with a desired spike train. Synaptic adaptation is driven by the error between the desired and the actual output spikes, with positive errors causing long-term potentiation and negative errors causing long-term depression. The amount of modification is proportional to an eligibility trace that is triggered by afferent spikes. The PSD rule is both computationally efficient and biologically plausible. The properties of this learning rule are investigated extensively through experimental simulations, including its learning performance, its generality to different neuron models, its robustness against noisy conditions, its memory capacity, and the effects of its learning parameters.

4.1 Introduction

With the same capability of processing spikes as biological neural systems, spiking neural networks (SNNs) [4, 107, 108] are more biologically realistic and computationally powerful than the traditional artificial neural networks (ANNs). Spikes are believed to be the principal feature in the information processing of neural systems, though the neural coding mechanism, i.e., how information is encoded in spikes still remains unclear. The temporal codes describe one possibility, where information is conveyed through precise times of spikes. However, the complexity of processing temporal codes [33, 109] might limit their usage in SNNs, which demands the development of efficient learning algorithms.

Supervised learning was proposed as a successful concept of information processing [110]. Neurons are driven to respond at desired states under a supervisory signal, and an increasing body of evidence shows that this kind of learning is exploited by the brain [63–66]. Supervised mechanism has been widely used to develop various learning algorithms for processing spatiotemporal spike patterns in SNNs [30, 33–37, 45].

SpikeProb [34] is one of the first supervised learning algorithms for processing precise spatiotemporal patterns in SNNs. However, in its original form, SpikeProb cannot learn to reproduce a multi-spike train. The tempotron rule [33], another gradient descent approach that is evaluated to be efficient for binary temporal classification tasks, cannot output multiple spikes either. As the tempotron is designed mainly for pattern recognition, it is unable to produce precise spikes. To produce a desired spike train, several learning algorithms have been proposed such as ReSuMe [37,47], Chronotron [36] and SPAN [35]. These three learning rules are all capable of training a neuron to generate a desired spike train in response to an input stimulus. In the Chronotron, two learning rules are introduced. One is analytically-derived (E-learning) and another one is heuristically-defined (I-learning). The I-learning rule is more biologically plausible but comes with less memory capacity than the E-learning rule. The performance of the I-learning rule depends on the weight initialization, where initial zero values can cause information loss from the corresponding afferent neurons. The E-learning rule and the SPAN rule are both based on an error function of the difference between the actual output spike train and the desired spike train. Their applicability is therefore limited to the tractable error evaluation, which might be unavailable in actual biological networks and inefficient from a computational point of view. These arithmetic-based rules can reveal explicitly how SNNs can be trained but the biological plausibility of the error calculation is somewhat questionable.

In this chapter, we propose an alternative learning mechanism called Precise-Spike-Driven (PSD) synaptic plasticity, that is able to learn the association between precise spike patterns. Similar to ReSuMe [37] and SPAN [35], the PSD rule is derived from the Widrow-Hoff (WH) rule but based on a different interpretation. The PSD rule is derived analytically based on converting the spike trains into analog signals by applying the spike convolution method. Such an approach is rarely reported in the existing learning rule studies [35]. Synaptic adaptation in the PSD is driven by the error between the desired and the actual output spikes, with positive errors causing long-term potentiation (LTP) and negative errors causing long-term depression (LTD). The amount of adaptation depends on an eligibility trace determined by the afferent spikes. Without complex error calculation, the PSD rule provides an efficient way for processing spatiotemporal patterns. We show that the PSD rule inherits the advantageous properties of both arithmetic-based and biologically realistic rules, being simple and efficient for computation, and yet biologically plausible. Furthermore, the PSD is an independent plasticity rule that can be applied to different neuron models. This straightforward interpretation of the WH rule also provides a possible direction for further exploitation of the rich theory of ANNs, and minimizes the gap between the learning algorithms of SNNs and the traditional ANNs.

Various properties of the PSD rule are investigated through an extensive experimental analysis. In the first experiment, the basic concepts of the PSD rule are demonstrated, and its learning ability on hetero-association of spatiotemporal spike pattern is investigated. In the second experiment, the PSD rule is shown to be applicable to different neuron models. Thereafter, experiments are conducted to analyze the learning rule regarding its robustness against noisy conditions, its memory capacity, effects of the learning parameters and its classification performance. Finally, a detailed discussion about the PSD rule and several related algorithms is presented.

4.2 Methods

In this section, we begin by presenting the spiking neuron models. We then describe the PSD rule for learning hetero-association between the input spatiotemporal spike patterns and the desired spike trains.

4.2.1 Spiking Neuron Model

The leaky integrate-and-fire (LIF) model is firstly considered. The dynamics of each neuron evolves according to the following equation:

$$\tau_m \frac{dV_m}{dt} = -(V_m - E) + (I_{ns} + I_{syn}) \cdot R_m \tag{4.1}$$

where V_m is the membrane potential, $\tau_m = R_m C_m$ is the membrane time constant, $R_m = 1\ M\Omega$ and $C_m = 10\ nF$ are the membrane resistance and capacitance, respectively, E is the resting potential, I_{ns} and I_{syn} are the background current noise and synaptic current, respectively. When V_m exceeds a constant threshold V_{thr}, the neuron is said to fire, and V_m is reset to V_{reset} for a refractory period t_{ref}. We set $E = V_{reset} = 0\ mV$ and $V_{thr} = E + 18\ mV$ for clarity, but any other values as $E = -60\ mV$ will result in equivalent dynamics as long as the relationships among E, V_{reset} and V_{thr} are kept.

For the post-synaptic neuron, we model the input synaptic current as:

$$I_{syn}(t) = \sum_i w_i I_{PSC}^i(t) \tag{4.2}$$

where w_i is the synaptic efficacy of the i-th afferent neuron, and I_{PSC}^i is the un-weighted postsynaptic current from the corresponding afferent.

$$I_{PSC}^i(t) = \sum_{t^j} K(t - t^j) H(t - t^j) \tag{4.3}$$

where t^j is the time of the j-th spike emitted from the i-th afferent neuron, $H(t)$ refers to the Heaviside function, K denotes a normalized kernel and we choose it as:

$$K(t-t^j) = V_0 \cdot \big(\exp(\frac{-(t-t^j)}{\tau_s}) - \exp(\frac{-(t-t^j)}{\tau_f})\big) \tag{4.4}$$

where V_0 is a normalization factor such that the maximum value of the kernel is 1, τ_s and τ_f are the slow and fast decay constants respectively, and their ratio is fixed at $\tau_s/\tau_f = 4$.

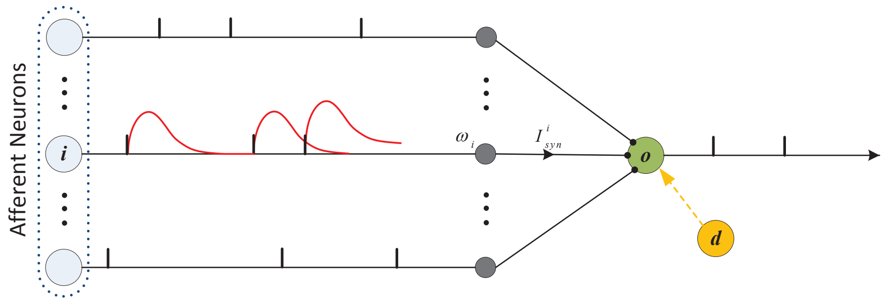

Figure 4.1: Illustration of the neuron structure. The afferent neurons are connected to the post-synaptic neuron through synapses. Each emitted spike from afferent neurons will trigger a post-synaptic current (PSC). The membrane potential of the post-synaptic neuron is a weighted sum of all incoming PSCs from all afferent neurons. The yellow neuron denotes the instructor which is used for learning.

Figure 4.1 illustrates the neuron structure. Each spike from the afferent neuron will result in a post-synaptic current (PSC). The membrane potential of the post-synaptic neuron is a weighted sum of all incoming PSCs over all afferent neurons.

In addition to the LIF model, we also investigate the flexibility of the PSD rule to different neuron models. For this, we use the IM model [9], where the dynamics of the IM model is described as:

$$\begin{cases} dV_m/dt = 0.04V_m^2 + 5V_m + 140 - U + I_{syn} + I_{ns} \\ dU/dt = a(bV_m - U) \\ \text{if } V_m \geq 30\, mV, \\ \text{then } V_m \leftarrow c, U \leftarrow U + d \end{cases} \tag{4.5}$$

where V_m again represents the membrane potential. U is the membrane recovery variable. The synaptic current (I_{syn}) is in the same form as described before, and I_{ns} again represents the background noise. The parameters $a = 0.02$, $b = 0.2$, $c = -65$ and $d = 8$ are chosen such that the neuron exhibits a regular spiking behavior which is the most typical behavior observed in cortex [9].

For computational efficiency, the LIF model is used in the following studies, unless otherwise stated.

4.2.2 PSD Learning Rule

In this section we describe in detail the PSD learning rule. Note that the spiking neuron models were developed from the traditional neuron models. In a similar way, we develop the learning rule for spiking neurons from traditional algorithms. Inspired by [35], we derive the proposed rule from the common WH rule. The WH rule is described as:

$$\Delta w_i = \eta x_i (y_d - y_o) \tag{4.6}$$

where η is a positive constant referring to the learning rate, x_i, y_d and y_o refer to the input, the desired output and the actual output, respectively.

Note that because the WH rule was introduced for the traditional neuron models such as perceptron, the variables in the WH rule are regarded as real-valued vectors. In the case of spiking neurons, the input and output signals are described by the timing of spikes. Therefore, a direct implementation of the WH rule does not work for spiking neurons. This motivates the development of the PSD rule.

A spike train is defined as a sequence of impulses triggered by a particular neuron at its firing time. A spike train is expressed in the form of:

$$s(t) = \Sigma_f \delta(t - t^f) \tag{4.7}$$

where t^f is the $f\text{-}th$ firing time, and $\delta(x)$ is the Dirac function: $\delta(x) = 1$ (if $x = 0$) or 0 (otherwise). Thus, the input, the desired output and the actual output of the spiking neuron are described as:

$$\begin{cases} s_i(t) = \Sigma_f \delta(t - t_i^f) \\ s_d(t) = \Sigma_g \delta(t - t_d^g) \\ s_o(t) = \Sigma_h \delta(t - t_o^h) \end{cases} \tag{4.8}$$

The products of Dirac functions are mathematically problematic. To solve this difficulty, we apply an approach called spike convolution. Unlike the method used in [35], which needs a complex error evaluation and requires spike convolution on all the spike trains of the input, the desired output and the actual output, we only convolve the input spike trains.

$$\tilde{s_i}(t) = s_i(t) * \kappa(t) \tag{4.9}$$

where $\kappa(t)$ is the convolving kernel, which we choose to be the same as Equation (4.4). In this case, the convolved signal is in the same form as I_{PSC}

in Equation (4.3). Thus, we use I_{PSC} as the eligibility trace for the weight adaptation. The learning rule becomes:

$$\frac{dw_i(t)}{dt} = \eta[s_d(t) - s_o(t)]I^i_{PSC}(t) \tag{4.10}$$

Equation (4.10) formulates an online learning rule. The dynamics of this learning rule is illustrated in Figure 4.2. It can be seen that the polarity of the synaptic changes depends on three cases: (1) a positive error (corresponding to a miss of the spike) where the neuron does not spike at the desired time, (2) a zero error (corresponding to a hit) where the neuron spikes at the desired time, and (3) a negative error (corresponding to a false-alarm) where the neuron spikes when it is not supposed to.

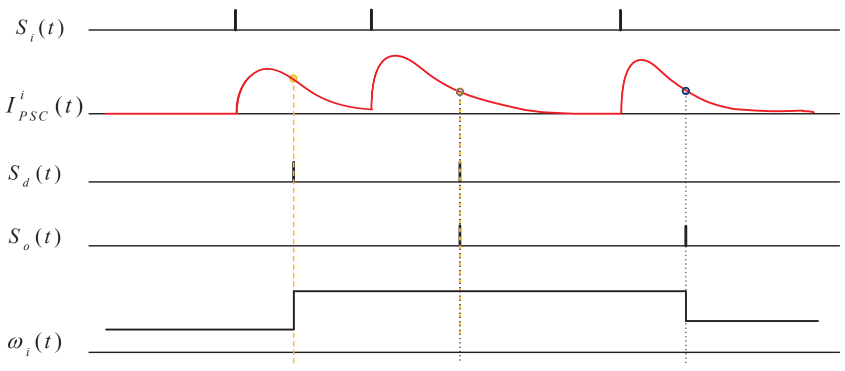

Figure 4.2: Demonstration of the weight adaptation in PSD. $S_i(t)$ is the presynaptic spike train. $S_d(t)$ and $S_o(t)$ are the desired and the actual postsynaptic spike train, respectively. $I^i_{PSC}(t)$ is the postsynaptic current and can be referred to as the eligibility trace for the adaptation of $w_i(t)$. A positive error, where the neuron does not spike at the desired time, causes synaptic potentiation. A negative error, where the neuron spikes when it is not supposed to, results in synaptic depression. The amount of adaptation is proportional to the postsynaptic current. There will be no modification when the actual output spike fires exactly at the desired time.

Thus, the weight adaptation is triggered by the error between the desired and the actual output spikes, with positive errors causing long-term potentiation and negative errors causing long-term depression. No synaptic change will occur if the actual output spike fires at the desired time. The amount of synaptic changes is determined by the current $I^i_{PSC}(t)$.

With the PSD learning rule, each of the variables involved has its own physical meaning. Moreover, the weight adaptation only depends on the current states. This is different from rules involving STDP, where both the pre- and post-synaptic spiking times are stored and used for adaptation.

By integrating Equation (4.10), we get:

$$\begin{aligned}\Delta w_i &= \eta \int_0^\infty [s_d(t) - s_o(t)] I^i_{PSC}(t) dt \qquad (4.11)\\ &= \eta \Big[\sum_g \sum_f K(t^g_d - t^f_i) H(t^g_d - t^f_i) - \sum_h \sum_f K(t^h_o - t^f_i) H(t^h_o - t^f_i) \Big]\end{aligned}$$

This equation could be used for trial learning where the weight modification is performed at the end of the pattern presentation.

In order to measure the distance between two spike trains, we use the van Rossum metric [111] but with a different filter function as described in Equation (4.4). This filter is used to compensate for the discontinuity of the original filter function. The distance can be written as:

$$Dist = \frac{1}{\tau} \int_0^\infty [f(t) - g(t)]^2 dt \qquad (4.12)$$

where τ is a free parameter (we set $\tau = 10\ ms$ here), $f(t)$ and $g(t)$ are filtered signals of the two spike trains that are considered for distance measurement.

Noteworthily, this distance parameter $Dist$ is not involved in the PSD learning rule, but is used for measuring and analyzing the performance of the learning rule, which reflects the dissimilarity between the desired and the actual spike trains. In the following experiments, different values of $Dist$ are used for analysis depending on the problems. For single-spike and multi-spike target trains, we set $Dist$ to be 0.2 and 0.5, respectively, corresponding to an average time difference of around $2.5\ ms$ for each pair of the actual and desired spikes. Smaller $Dist$ can be used if exact association is the main focus, e.g., $Dist = 0.06$ corresponds to a time difference about $0.6\ ms$, where no obvious dissimilarity can be seen between the two spike trains.

4.3 Results

In this section, several experiments are presented to demonstrate the characteristics of the PSD rule. The basic concepts of the PSD rule are first examined, by demonstrating its ability to associate a spatiotemporal spike pattern with a target spike train. Furthermore, we show that the PSD has desirable properties, such as generality to different neuron models, robustness against noise and learning capacity. The effects of the parameters on the learning are also investigated. Then, the application of the proposed algorithm to the classification of spike patterns is also shown.

4.3.1 Association of Single-Spike and Multi-Spike Patterns

This experiment is devised to demonstrate the ability of the proposed PSD rule for learning a spatiotemporal spike pattern. The neuron is trained to reproduce spikes that fire at the same spiking time of a target train.

Experiment setup

The neuron is connected with n afferent neurons, and each fires a single spike within the time interval of $(0,\ T)$. Each spike is randomly generated with a uniform distribution. We set $n = 1000$, $T = 200\ ms$ here. To avoid a single synapse dominating the firing of the neuron, we limit the weight below $w_{max} = 6\ nA$. The initial synaptic weights are drawn randomly from a normal distribution with mean value of $0.5\ nA$ and a standard deviation of $0.2\ nA$. For the learning parameters, we set $\eta = 0.01 w_{max}$ and $\tau_s = 10\ ms$. The target spike train can be randomly generated, but for simplicity, we specify it as $[40,\ 80,\ 120,\ 160]\ ms$ to evenly distribute the spikes over the whole time interval T.

Learning process

Figure 4.3 illustrates a typical run of the learning. Initially, the neuron is observed to fire at any arbitrary time and with a firing rate different from the target train, resulting in a large distance value. The actual output spike train is quite different from the target train at the beginning. During the learning process, the neuron gradually learns to produce spikes at the target time, and that is also reflected by the decreasing distance. After finishing the first 10 epochs of learning, both the firing rate and the firing time of the output spikes match those in the target spike train. The dynamics of neuron's membrane potential is also shown in Figure 4.3. Whenever the membrane potential exceeds the threshold, a spike is emitted and the potential is kept at reset level for a refractory period. The detailed mathematical description governing this behavior was presented

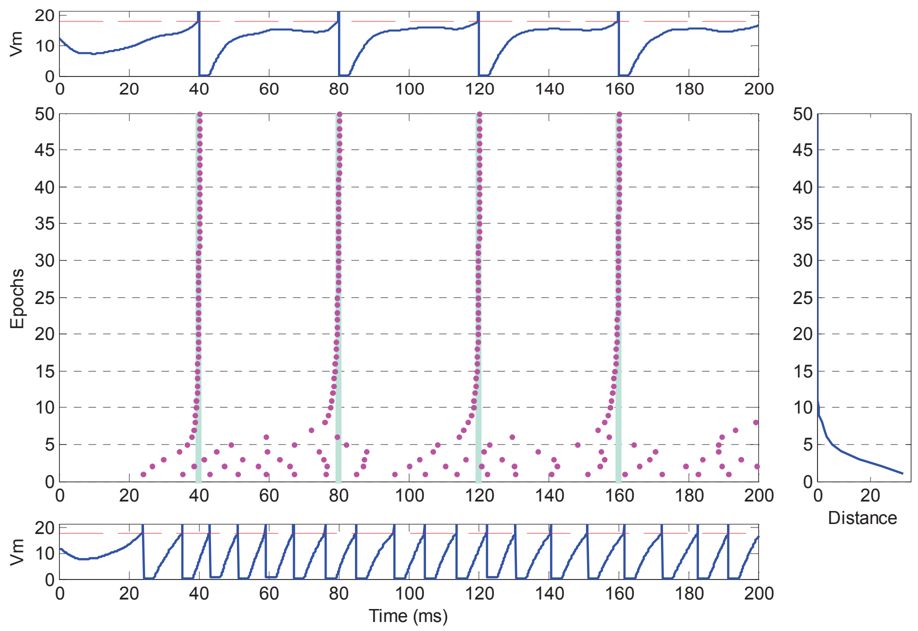

Figure 4.3: Illustration of the temporal sequence learning of a typical run. The neuron is connected with $n = 1000$ synapses, and is trained to reproduce spikes at the target time (denoted as light blue bars in the middle). The bottom and top show the dynamics of the neuron's potential before and after learning, respectively. The dashed red lines denote the firing threshold. In the middle, each spike is denoted as a dot. The right figure shows the spike distance between the actual output spike train and the target spike train.

previously in the section on the Spiking Neuron Model.

This experiment shows the feasibility of the PSD rule to train the neuron to reproduce a desired spike train. After several learning epochs, the neuron can successfully spike at the target time. In other words, the proposed rule is able to train the neuron to associate the input spatiotemporal pattern with a desired output spike train within several training epochs. The information of the input pattern is stored by a specified spike train.

Causal weight distribution

We further examine how the PSD rule drives the synaptic weights and the evolution of the distance between the actual and the target spike trains. In order to guarantee statistical significance, the task described in Figure 4.3 is repeated 100 times. Each time is referred to as one run. At the initial point of each run, different random weights are used for training.

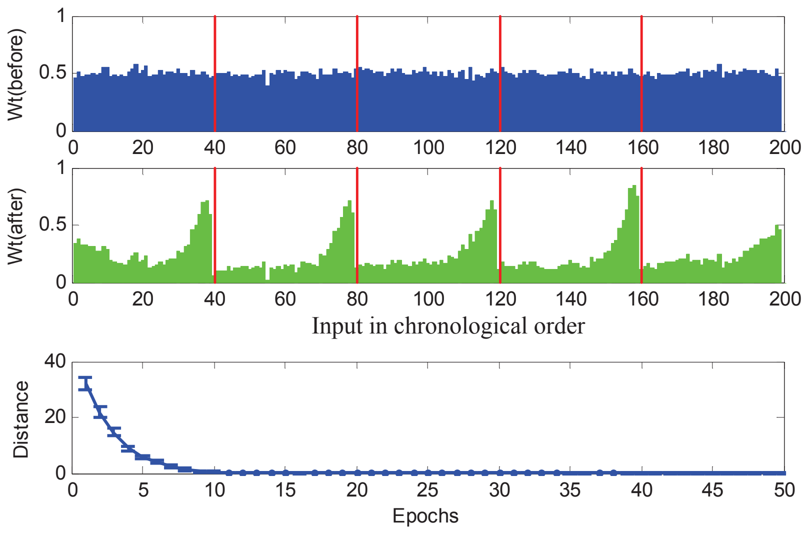

Figure 4.4: Effect of the learning on synaptic weights and the evolution of distance along the learning process. The top and the middle show the averaged weights before and after learning, respectively. The height of each bar in the figure reflects the corresponding synaptic strength. All the afferent neurons are chronologically sorted according to their spike time. The target spikes are overlayed on the weights figure according to their time, and are denoted as red lines. The bottom shows the averaged distance between the actual spike train and the desired spike train along the learning process. All the data are averaged over 100 runs.

As can be seen from Figure 4.4, the initial weights are normally distributed around $0.5\ nA$, which reflects the fact that there are no significant differences among the input synapses. This initial distribution of weights is expected due to the experimental setup. After learning, a causal connectivity is established. According to the learning rule, the synapses that fire temporally close to the time of the target spikes are potentiated. Those synapses that result in undesired output spikes are depressed. This temporal causality is clearly reflected on the distribution of weights after learning (Figure 4.4). Among those causal synapses, the one with a closer spiking time to the desired time normally has a relatively higher synaptic strength. The synapses firing far from the desired time will have lower causal effects. Additionally, the evolution of distance along the learning shows that the PSD rule successfully trains the neuron to reproduce the desired spikes in around ten epochs. The results also validate the efficiency of the PSD learning rule in accomplishing the single association task.

Adaptive learning performance

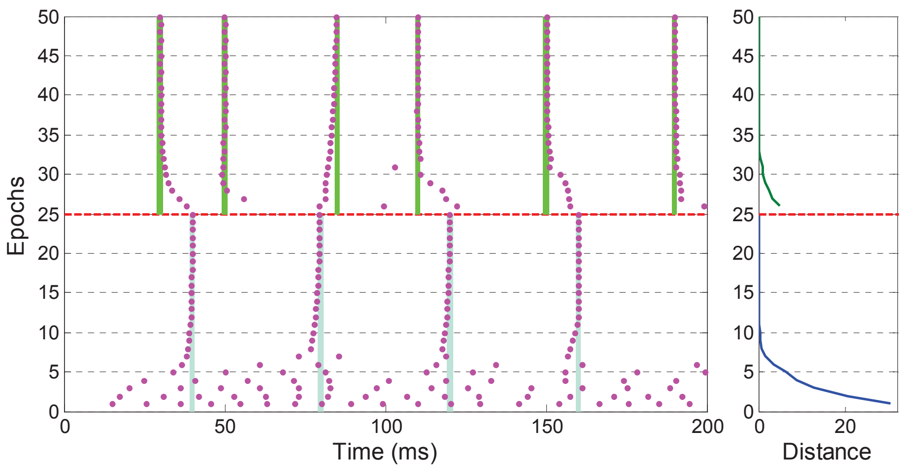

Figure 4.5: Illustration of the adaptive learning of the changed target trains. Each dot denotes a spike. At the beginning, the neuron is trained to learn one target (denoted by the light blue bars). After 25 epochs of learning (the dashed red line), the target is changed to another randomly generated train (denoted by the green bars). The right figure shows the distance between the actual output spike train and the target spike train along the learning process.

At the beginning, the neuron is trained to learn a target train as in the previous tasks. After one successful learning, the target spike train is changed to another arbitrarily generated train, where the precise spike time and the firing rate are different from the previous target. We discover that, with the PSD learning rule, we successfully train the neuron to learn the new target within several epochs. As shown in Figure 4.5, during learning, the neuron gradually adapts its firing status from the old target to the new target.

Learning multiple spikes

In the scenario considered above, all afferent neurons are supposed to fire only once during the entire time window. The applicability of the PSD rule is not limited to this single spike code. We further illustrate the case where each synaptic input transmits multiple spikes during the time window. We again use the same setup as above, but each synaptic input is now generated by a homogeneous Poisson process with a random rate ranging from $5 - 25\ Hz$. Multiple spikes increase the difficulty of the learning since these spikes interfere with the local learning processes [47].

As shown in Figure 4.6, the learning although slower, is again successful. The interference of local learning processes results in fluctuations of the output spikes around the target time. In the subsequent learning epochs, the neuron gradually converges to spiking at the target time. This experiment demonstrates that the PSD rule deals with multiple spikes quite well. Compared to multiple spikes, the single spike code is simple for analysis and efficient for computation. Thus, for simplicity, we use the single spike code in the following experiments where each afferent neuron fires only once during the time window.

These experiments clearly demonstrate that the PSD rule is capable of training the neuron to fire at the desired time. The causal connectivity is established after learning with this rule. In the following sections, some more challenging learning scenarios are taken into consideration to further investigate

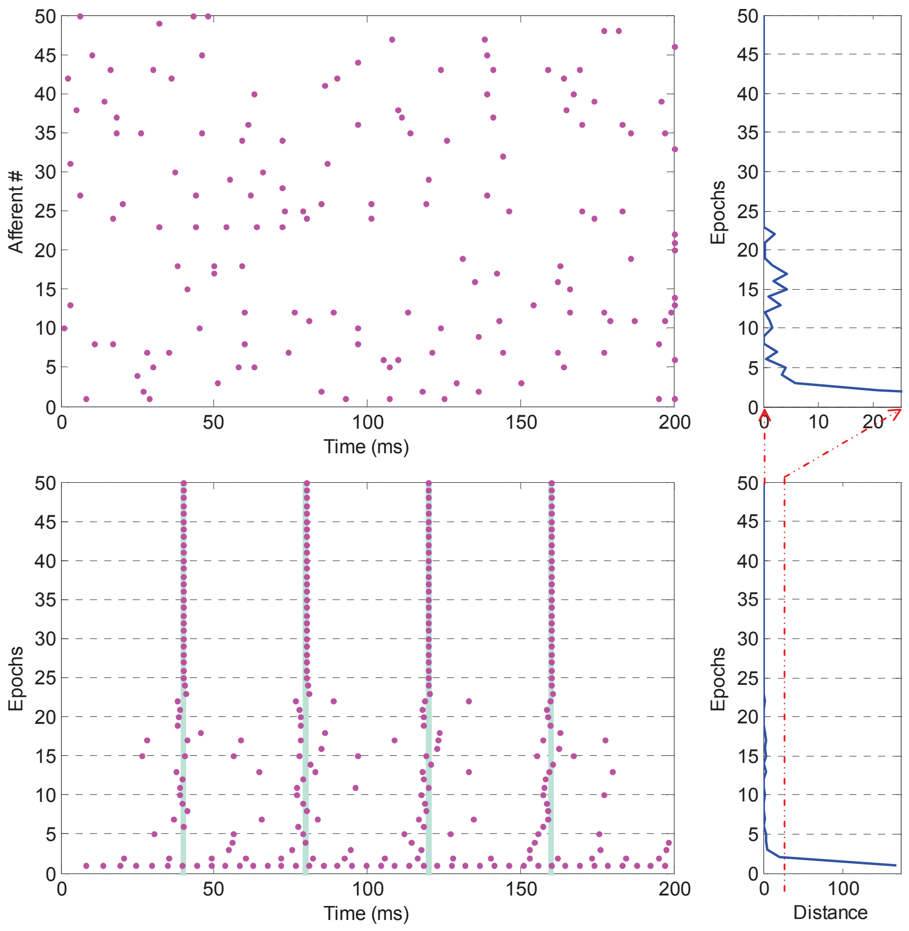

Figure 4.6: Illustration of a typical run for learning multi-spike pattern. Each dot denotes a spike. The top left shows the input spikes from the first 50 afferent neurons out of 1000. Each synaptic input is generated by a homogeneous Poisson process with a random rate from $5 - 25\ Hz$. The bottom left shows the neuron's output spikes. The right column shows the distance between the actual output spike train and the target spike train along learning.

the properties of the PSD rule.

4.3.2 Generality to Different Neuron Models

We carry out this experiment to demonstrate that the PSD learning rule is independent of the neuron model. In this experiment, we only compare the results of learning association for the LIF and IM neuron models that were described previously. For a fair comparison, both neurons are connected to the same afferent neurons, and they are trained to reproduce the same target spike train. The setup for generating the input spatiotemporal patterns is the same as the experiment in Figure 4.5. The connection setup is illustrated in Figure 4.7. Except for the neuron dynamics described in Equation (4.1) and Equation (4.5) respectively, all the other parameters are the same for the two neurons.

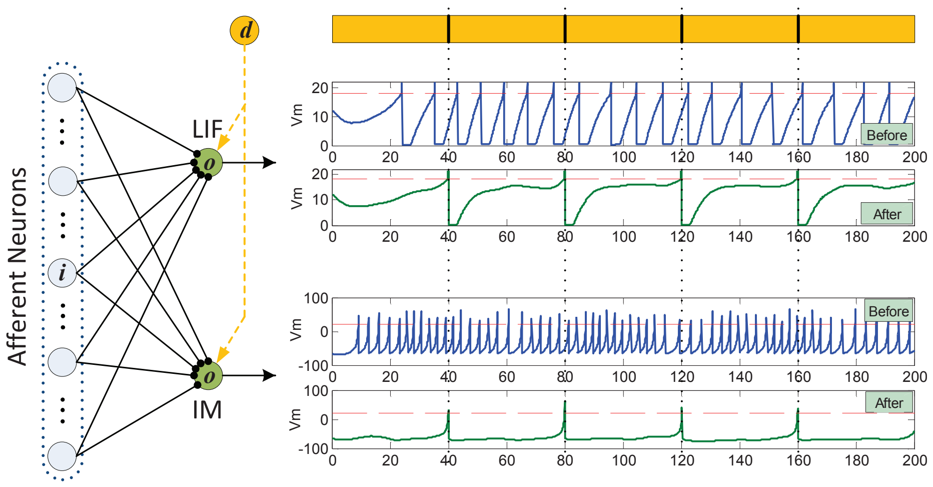

Figure 4.7: Learning with different spiking neuron models. The LIF and IM neuron models are considered. The left panel shows the connection setup of the experiment. Both the two neurons are connected to the same $n = 1000$ afferent neurons, and are trained to reproduce target spikes (denoted by the yellow parts). The right panel shows the dynamics of neurons' potential before and after learning. The dashed red lines denote the firing threshold.

The dynamic difference between the two types of spiking neuron models is clearly demonstrated in Figure 4.7. Although the neuron models are different, both of the neurons can be trained to successfully reproduce the target spike train with the proposed PSD learning rule. It is seen that the two neurons fire at arbitrary time before learning, while after learning they fire spikes at the desired time.

In the PSD rule, synaptic adaptation is triggered by both the desired spikes and the actual output spikes. The amount of updating depends on the presynaptic spikes firing before the triggering spikes. That is to say, the weight adaptation of our rule is based on the correlation between the spiking time only. This suggests the PSD has the generality to work with various neuron models, a capability similar to that of the ReSuMe rule [47].

4.3.3 Robustness to Noise

In previous experiments, we only consider the simple case where the neuron is trained to learn a single pattern under noise-free condition. However, the reliability of the neuron response could be significantly affected by noise. In this experiment, two noisy cases are considered: stimuli noise and background noise.

Experiment setup

In this experiment, a single LIF neuron with $n = 500$ afferent neurons is tested. Initially, a set of 10 spike patterns are randomly generated as in previous experiments. These 10 spike patterns are fixed as the templates. The neuron is trained for 400 epochs to associate all patterns in the training set with a desired spike train (the same train as is used before). Two training scenarios are considered in this experiment, i.e., deterministic training (in the noise-free condition) and noisy training. In the testing phase, a total number of 200 noise patterns are used. Each template is used to construct 20 testing patterns. We determine the association to be correct, if the distance between the output spike train and the desired spike train is lower than a specified level (0.5 is used here).

Input jittering noise

In the case of input jittering noise, a Gaussian jitter with a standard deviation (σ_{Inp}) is added to each input spike to generate the noise patterns. The strength of the jitter is controlled by the standard deviation of the Gaussian. The top row in Figure 4.8 shows the learning performance. In the deterministic training, the neuron is trained purely with the initial templates. In the noisy training, a noise level of $3\ ms$ is used. Different levels of noise are used in the testing phase to evaluate the generalization ability. For the deterministic training, the output stabilizes quickly and can exactly converge to the desired spike train within tens of learning epochs. However, the generalization accuracy decreases quickly with the increasing jitter strength. In the scenario of noisy training, although the training error cannot become zero, a better generalization ability is obtained. The neuron can successfully reproduce the desired spike train with a relatively high accuracy when the noise strength is not higher than the one used in the training. In conclusion, the neuron is less sensitive to the noise if the noisy training is performed.

Background current noise

In this case, the background current noise (I_{ns}) is considered as the noise source. The mean value of I_{ns} is assumed zero, and the strength of the noise is determined by its variance ($\sigma_{I_{ns}}$). A strength of $10\ nA$ noise is used in the noisy training. We report the results in the bottom row of Figure 4.8. Similar results are obtained as with the first case. Although the output can quickly converge to zero error in the deterministic training, the generalization performance is quite

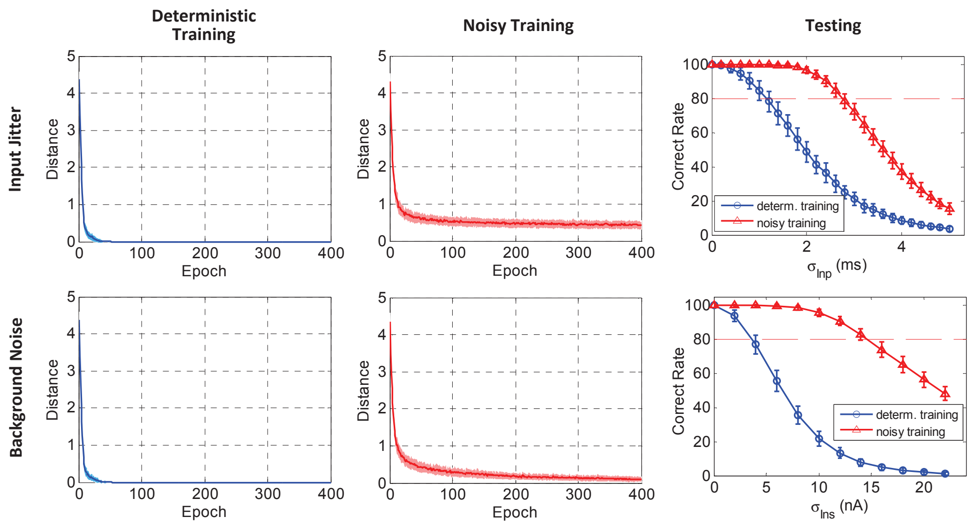

Figure 4.8: Robustness of the learning rule against jittering noise of input stimuli and background noise. The top row presents the case where the noise comes from the input spike jitters. The bottom row presents the case of background noise. The neuron is trained under noise-free conditions (denoted as deterministic training), or is trained under noisy conditions (denoted as noisy training). In the training phase (left two columns), the neuron is trained for 400 epochs. Along the training process, the average distance between the actual output spike train and the desired spike train is shown. The standard deviation is denoted by the shaded area. In the testing phase (right column), the generalization accuracies of the trained neuron on different levels of noise patterns are presented. Both the average value and the standard deviation are shown. All the data are averaged over 100 runs.

sensitive to the noise. The association accuracy drops quickly when the noise strength increases. When the neuron is trained with noise patterns, it becomes less sensitive to the noise. A relatively high accuracy can be obtained with a noise level up to $14\ nA$.

This experiment shows that the neuron trained under noise-free conditions will be significantly affected by noise in the testing phase. Such an influence of noise on the timing accuracy and reliability of the neuron response has been considered in many studies [33, 35, 36, 47, 112, 113]. Under the noisy training, the trained neuron demonstrates high robustness against the noise. The noisy training enables the neuron to reproduce desired spikes more reliably and precisely.

4.3.4 Learning Capacity

As used for the perceptron [61] and tempotron [33,45] learning rules, the ratio of the number of random patterns (p) that a neuron can correctly classify over the number of its synapses (n), $\alpha = p/n$, is used to measure the memory load. An important characteristic of a neuron's capacity is the maximum load that it can learn. In this experiment, the memory capacity of the PSD rule is investigated.

Experiment setup

We devise an experiment that has a similar setup to that in [35]. A number of p patterns are randomly generated in the same process as previous experiments, where each pattern contains n spike trains and each train has a single spike. The patterns are randomly and evenly assigned to c different categories. Here we choose $c = 4$ for this experiment. A single LIF neuron is trained to memorize all patterns correctly in a maximum number of 500 training epochs. The neuron is trained to emit a single spike at a specified time for patterns from each category. The desired spikes for the 4 generated categories are set to the time of 40, 80, 120 and 160 ms, respectively. A pattern is considered to have been correctly memorized by the neuron if the distance between the actual spike train and the desired train is below 0.2. The learning process is considered a failure if the number of training epochs reaches the maximum number.

Maximum load factor

Figure 4.9 shows the results of the experiment for the case of 500, 750 and 1000 afferent neurons, respectively. All the data are averaged over 100 runs. In each run, different initial weights are used. As seen from Figure 4.9, the number of epochs required for the training increases slightly as the number of patterns increases when the load is not too high, but a sharp increase of learning epochs occurs after a certain high load. This suggests that the task becomes tougher with an increasing load. It is also noted that a larger number of synapses leads to a bigger memory capacity for the same neuron. It is reported that the maximum load factors for 500, 750 and 1000 synapses are 0.144, 0.133 and 0.124, respectively.

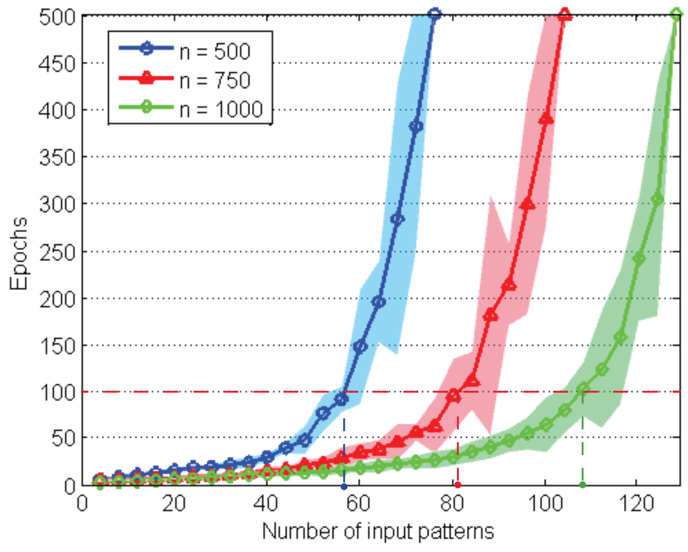

Figure 4.9: The memory capacity of the PSD rule with different numbers of synapses. The neuron is trained to memorize all patterns correctly in a maximum number of 500 epochs. The reaching points of 500 epochs are regarded as failure of the learning. The marked lines denote average learning epochs and the shaded areas show the standard deviation. The dashed line at 100 epochs is used for evaluating the efficient load α_e described in the main text. All the data are averaged over 100 runs.

Efficient load factor

Besides the maximum load factor, we heuristically define another factor, the efficient load α_e. The neuron can learn patterns efficiently with a relatively high load when the number of patterns does not exceed a certain value (p_e). The efficient load factor is denote as $\alpha_e = p_e/n$. When the load is below α_e, the neuron can reliably memorize all patterns with a small number of training epochs. There are different ways to define α_e. We show two possible ways. One is to derive the definition from a mathematical calculation such as $(dEpochs/dp)_{p_e} = \delta$, where δ is a specified value (for example $\delta = 0.5$). A simpler method is where a specified number of training epochs is used. The corresponding number of patterns that can be correctly learnt is considered as p_e. For simplicity, we use the latter as an example for demonstration and the specified number of epochs is set to 100. As seen from Figure 4.9, the efficient load factors for 500, 750 and 1000 synapses are 0.112, 0.109 and 0.108, respectively. Surprisingly, these efficient load factors seem to all be around a stable value which only changes slightly across different numbers of synapses. This fixed value of efficient load factor for different values of n indicates that the number of patterns that a neuron can efficiently memorize grows linearly with the number of afferent synapses. It is worth noting that the concept of efficient load factor α_e provides an important guideline for choosing the load of patterns when a reliable and efficient training is required.

4.3.5 Effects of Learning Parameters

Two of the major parameters involved in the PSD learning rule are the learning rate η and the decay constant τ_s. In this section, we aim to investigate the effects of these parameters on the learning process.

Small τ_s results in strong causal weight distribution

As a decay constant, τ_s is an important parameter involved in the postsynaptic current. It determines how long a presynaptic spike will still have causal effect on the postsynaptic neuron. In the phase of synaptic adaptation, τ_s also determines the magnitude of modification on the synaptic weights at the time of a triggering spike. Thus, τ_s will affect the distribution of weights after the training. To look into this effect, we conduct an experiment with a similar setup as in Figure 4.4 but with different values of τ_s. Here we choose $\tau_s = 3$, 10 and 30 ms. As can be seen from Figure 4.10, a smaller τ_s (3 ms) can result in a very uneven distribution with only a few synapses being given relatively higher weights. A flat distribution is obtained with an increasing τ_s. This is because τ_s determines how long the causal effect of an afferent spike will sustain. A smaller τ_s means that only the nearer neighbors are involved in generating the desired spikes, hence resulting in a smaller number of causal synapses. With a smaller number of causal synapses, a higher synaptic strength will be required to generate spikes at the desired time. On the other hand, with a larger τ_s, a wider range of causal neighbors can contribute to generating the desired spikes, and therefore a lower synaptic strength will be sufficient. The synaptic strength and distribution for different values of τ_s are obtained as in Figure 4.10.

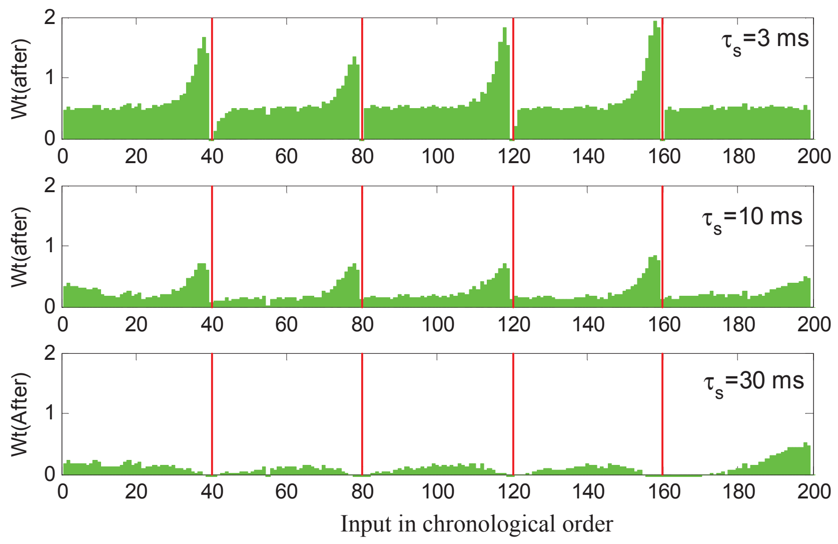

Figure 4.10: Effect of decay constant τ_s on the distribution of weights. The averaged weights after learning are shown. The height of each bar reflects the synaptic strength. The afferent neurons are chronologically sorted according to their spike time. The target spikes are overlayed and denoted as red lines. Cases of $\tau_s = 3$, 10 and 30 ms are depicted. All the data are averaged over 100 runs.

Effects of both η and τ_s on the learning

We further conduct another experiment to evaluate the effects of both η and τ_s on the learning. In this experiment, a single LIF neuron with $n = 500$ afferent neurons is considered. The neuron is trained to correctly memorize a set of 10 spike patterns randomly generated over a time window of 200 ms. The neuron is trained in a maximum number of 500 epochs to correctly associate all these patterns with a desired spike train of [40, 80, 120, 160] ms. We denote that a pattern is correctly memorized if the distance between the output spike train and the desired spike train is below 0.06. If the number of training epochs exceeds 500, we regard it as a failure. We conduct an exhaustive search over a wide range of η and τ_s. Figure 4.11 shows how η and τ_s jointly affect the learning performance, which can be used as a guidance to select the learning parameters.

With a fixed τ_s, a larger η results in a faster learning speed (shown in Figure 4.11, right panel), but when η is increased above a critical value (e.g., 0.1 for $\tau_s = 30$ ms in our experiments), the learning will slow down or even fail. For small η, a larger τ_s leads to a faster learning, however, for large η, a larger τ_s has the opposite effect. As a consequence, when τ_s is set in a suitable range (e.g., [5,15] ms), a wide range of η can result in a fast learning speed (e.g., below 100 epochs).

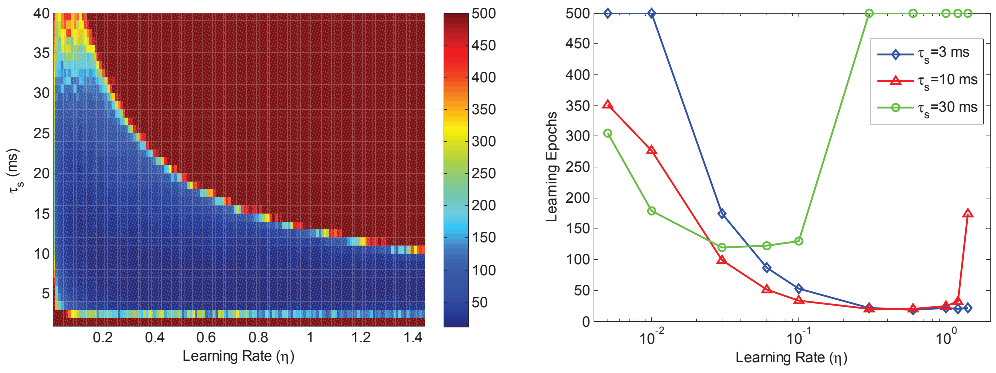

Figure 4.11: Effects of η and τ_s on the learning. The neuron is trained in a maximum number of 500 epochs to correctly memorize a set of 10 spike patterns. The average learning epochs are recorded for each pair of η and τ_s. The reaching points of 500 epochs are regarded as failure of the learning. The left shows an exhaustive investigation of a wide range of η and τ_s, and the data are averaged over 30 runs. A small number of learning parameters are examined in the right figure, and the data are averaged over 100 runs.

4.3.6 Classification of Spatiotemporal Patterns

In this experiment, the ability of the proposed PSD rule for classifying spatiotemporal patterns is investigated by using a multi-category classification task. The setup of this experiment is similar to that in [35]. Three random spike patterns representing three categories are generated in a similar fashion to that in the previous experiments, and they are fixed as the templates. A

Gaussian jitter with a standard deviation of $3\ ms$ is used to generate training and testing patterns. The training set and the testing set contain 3×25 and 3×100 samples, respectively. Three neurons are trained to classify these three categories, with each neuron representing one category. Different neurons for each category can be specified to fire different spike trains. However, for simplicity, all the neurons in this experiment are trained to fire the same spike train ($[40,\ 80,\ 120,\ 160]\ ms$). The experiment is repeated 100 times, with each run having different initial conditions.

After training, classification is performed on both the training and the testing set. In the classification task, we propose two decision-making criteria: absolute confidence and relative confidence. With the absolute confidence criterion, only if the distance between the desired spike train and the actual output spike train of the corresponding neuron is smaller than a specified value (0.5 is used here), then the input pattern will be regarded as being correctly classified. As for the relative confidence criterion, a scheme of competition is used. The incoming pattern will be labeled by the winning neuron that produces the closest spike train to its desired spike train.

Figure 4.12 shows the average classification accuracy for each category under the two proposed decision criteria. From the absolute confidence criterion, we see that the neuron successfully classifies the training set with an average accuracy of 99.65%. The average accuracy for the testing set is 77.11%. Noteworthily, under the relative confidence, both the average accuracies for the training and the testing set reach 100%. The performance for the classification task is therefore significantly improved by the relative confidence decision

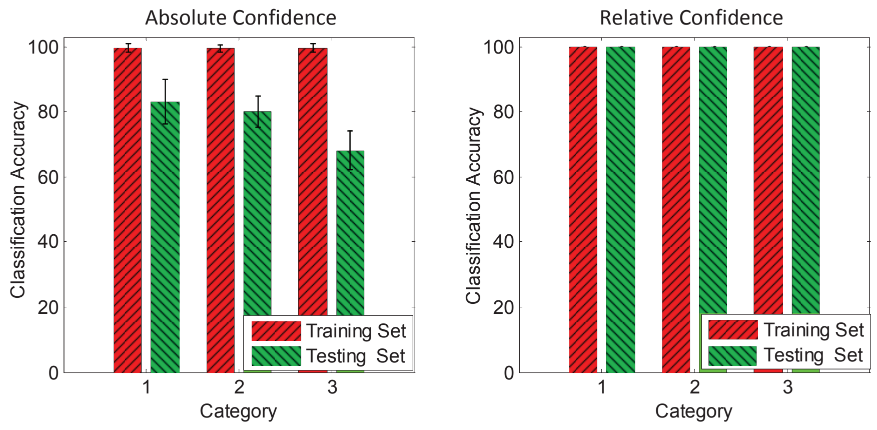

Figure 4.12: The average accuracies for the classification of spatiotemporal patterns. There are 3 categories to be classified. The average accuracies are represented by shaded bars. Two types of criteria for making decision are proposed and investigated. The left is the absolute confidence criterion, and the right is the relative confidence criterion. All the data are averaged over 100 runs.

Table 4.1: Multi-Category Classification of Spatiotemporal Patterns

Accuracy (%)	Category 1		Category 2		Category 3	
	Training	Testing	Training	Testing	Training	Testing
Absolute Confidence	99.6 ±1.21	83.15 ±6.79	99.68 ±1.09	80.06 ±4.73	99.68 ±1.23	68.12 ±6.09
Relative Confidence	100	100	100	100	100	100
Tempotron	100	99.65 ±1.21	100	99.74 ±1.01	100	99.61 ±1.0

making criterion. With the absolute confidence criterion, the trained neuron strives to find a good match with the memorized patterns. However, with the relative confidence criterion, the trained neuron attempts to find the most likely category through competition.

For the classification of spatiotemporal patterns, the tempotron is an efficient rule [33] in training LIF neurons to distinguish two classes of patterns by firing one spike or by keeping quiescent. We use the tempotron rule to benchmark the PSD rule in the classification of spatiotemporal patterns. The tempotron rule is applied to perform the same classification task as above. The classification accuracies are shown in Table 4.1. As can be seen from Table 4.1, our proposed rule with the relative confidence criterion has a comparable performance to the tempotron rule. Moreover, the PSD rule is advantageous in that it is not limited to performing classification, but it is also able to memorize patterns by firing desired spikes at precise time.

4.4 Discussion and Conclusion

The PSD rule is proposed for the association and recognition of spatiotemporal spike patterns. In summary, the PSD rule transforms the input spike trains into analog signals by convolving the spikes with a kernel function. By using a kernel function, the analog signals are presented in the simple form of synaptic currents. It is biologically plausible because it allows us to interpret the signals with physical meaning. Synaptic adaptation is driven by the error between the desired and the actual output spikes, with positive errors causing LTP and negative errors causing LTD. The amount of synaptic adaptation is determined by the transformed signal of the input spikes (postsynaptic currents here) at the time of modification occurrence. When the actual spike train is the same as the desired spike train, the adaptation of the weights will be terminated.

There is a supervisory signal involved in the PSD rule. The most documented evidence for supervised rules comes from studies of the cerebellum and the cerebellar cortex [64, 65]. It is shown that supervisory signals are provided to the learning modules by sensory feedback [66] or other supervisory neural structures in the brain [65]. A neuromodulator released by the supervisory system can induce the control of the adaptation. This control occurs for several neuromodulatory pathways, such as dopamine and acetylcholine [67,68]. Experimental evidence shows that N-methyl-D-aspartate (NMDA) receptors are critically involved in the processes of LTP and LTD [114–116]. After opening the NMDA channels, the resulting Ca^{2+} entry then activates the biochemistry of potentiation which leads to LTP [116]. Suppression of NMDA receptors by spike-mediated calcium entry may be a necessary step in the induction of LTD [32, 116]. The synaptic modification can be implemented through a supervisory control of opening or suppression of these NMDA channels.

The PSD rule is simple and efficient in synaptic adaptation. Utilizing the postsynaptic current as the eligibility trace for weight adaptation is a simple and efficient choice. The same signals of postsynaptic currents are also used in the synaptic adaptation as in the neuron dynamics, unlike the learning rules such as [34, 35, 47] where different sources of signals were used. Thus, the number of signal sources involved in the learning is reduced, which will directly benefit the computation. Secondly, unlike the arithmetic-based rules [34–36], where a complex error calculation is required for the synaptic adaptation, the PSD rule is based on a simple form of spike error between the actual and the desired spikes. The synaptic adaptation is driven by these precise spikes without complex error calculation. As a matter of fact, the weight modification only depends on currently available information (shown as Figure 4.2). Additionally, due to the ability of the PSD rule to operate online, it is suitable for real-time applications. According to the PSD rule, different kernels, such as the exponential kernel and α kernel, can also be used in convolving the spikes to provide different eligibility traces.

The PSD rule is designed for processing spatiotemporal patterns, where the exact time of each spike is used for information transmission. The PSD rule is unsuitable for learning patterns under the rate code because this rule is designed to process precise-timing spikes by its nature. The rate code uses the spike count but not the precise time to convey information. Like other spatiotemporal mapping algorithms, including ReSuMe [37], Chronotron [36] and SPAN [35], the PSD rule cannot guarantee successful learning of an arbitrary spatiotemporal spike pattern. A sufficient number of input spikes around the desired time are required for establishing causal connections. In other words, the temporal range covered by the desired spikes should be covered by the input spikes.

In most of the experiments, a single spike code is used for afferent neurons, where each input neuron only fires a single spike during the entire time window. This single spike code is chosen for various reasons but more than one spike is also allowed for the PSD rule. Firstly, a single spike code is simple for analysis and efficient for computation. Secondly, there is strong biological evidence supporting the single spike code. The PSD rule is also suitable for multi-spike train (results shown in Figure 4.6). When the number of spikes from each afferent neuron is not high enough, the neuron can produce the desired spike train after several epochs. When the number of spikes increases, the learning becomes slower and more difficult to converge. Additionally, the biological plausibility of an encoding scheme that can use multiple spikes to code information is still unclear.

Chapter 5

A Spiking Neural Network System for Robust Sequence Recognition

This chapter proposes a biologically plausible network architecture with spiking neurons for sequence recognition. This architecture is a unified and consistent system with functional parts of sensory encoding, learning and decoding. This system is the first attempt that helps to reveal the systematic neural mechanisms considering both the upstream and the downstream neurons together. The whole system is consistently combined in a temporal framework, where the precise timing of spikes is considered for information processing and cognitive computing. Experimental results show that our system can properly perform the sequence recognition task with the integration of all three functional parts. The recognition scheme is robust to noisy sensory inputs and it is also invariant to changes in the intervals between input stimuli within a certain range. The classification ability of the temporal learning rule used in our system is investigated through two benchmark tasks including an XOR task and an optical character recognition (OCR) task. Our temporal learning rule outperforms other two benchmark rules that are widely used for classification. Our results also demonstrate the computational power of spiking neurons over perceptrons for processing spatiotemporal patterns.

5.1 Introduction

As one of the cognitive abilities, sequence recognition refers to the ability to detect and recognize the temporal order of discrete elements occurring in sequence. Such sequence decoding operations are required for processing temporally complex stimuli such as speech where important information is embedded in patterns over time. However, the biophysical mechanisms by which neural circuits detect and recognize sequences of external stimuli are poorly understood.

Sequence information processing is a general problem that the brain needs to solve. Several approaches with the design of traditional artificial neural network structures [117, 118] have been considered and implemented for processing temporal information. The functionality of the brain for sequence recognition is mimicked through the artificial structures. However, these neural structures do not consider the building units of spiking neurons. Recognizing sequences of external stimuli with spiking features in the brain still remains an open question. Numerous studies have put efforts separately to computational mechanisms with spiking neurons, where some focus on neural representations of the external information [11] while others focus on the internal procession of either upstream or downstream neurons [33–36, 47, 119–122]. Relatively few proposals exist for recognizing the sequence of incoming stimuli from a systematic level of view. Thus, a structure based on spiking neural networks is demanded. Such a spiking neural system for sequence recognition should contain several functional parts including neural coding, learning and decoding. With these functional parts integrating with each other, the system could process information from levels of upstream encoding neurons to levels of downstream decoding neurons.

Among several different temporal learning rules, without complex error calculation, the PSD rule is simple and efficient from the computational point of view, and yet biologically plausible [119]. In the classification of spatiotemporal patterns, the PSD rule can even outperform the efficient tempotron rule [119]. Moreover, the PSD rule is not limited to the classification, but can also train the neuron to associate the spatiotemporal spike patterns with the desired spike trains.

Recently, a new decoding scheme with spiking neurons has been proposed to describe how downstream neurons with dendritic bistable plateau potentials can perform the decoding of spike sequences [120, 121]. The transition dynamics of this downstream decoding network is demonstrated to be equivalent to that of a finite state machine (FSM). This decoding scheme has the same computational power as the FSM. It is capable of recognizing an arbitrary number of spike sequences [121]. However, as a part of a whole system, this decoding only describes the behavior of the downstream neurons. How the upstream neurons behave and communicate with the downstream neurons remains unclear.

In this chapter, a unified and consistent system with spiking neurons is proposed for sequence recognition. To the best of our knowledge, this is the first attempt to consider a spiking system for sequence recognition with functional parts of sensory coding, learning and decoding. This work helps to reveal the systematic neural mechanisms considering all the processes of sensory coding, learning and downstream decoding. Such a system bridges the gap between these independently studied processes. The system is integrated in a consistent scheme by processing precise-timing spikes, where temporal coding and learning are involved. The sensory coding describes how external information is converted into neural signals. Through learning, the neurons adapt their synaptic efficacies for processing the input neural signals. The decoding describes how the output neurons extract information from the neural responses. The sequence recognition of the proposed biologically plausible system is realized through the combination of item recognition and sequence order recognition. Identifying the input stimuli is required before recognizing the sequence order. The recognition scheme is robust to noisy sensory input and it is also insensitive to changes in the intervals between input stimuli within a certain range. The experiments present spiking neural networks as a paradigm which can be used for recognizing sequences of incoming stimuli.

The rest of this chapter is organized as follows. In section 5.2, detailed descriptions are presented about the methods used in our integrated system, including the sensory encoding method, the temporal learning rule and the spike sequence decoding method. Section 5.3 shows the performances of our approach through numerical simulations. Detailed investigation and analysis on different parts of the system are presented firstly. The classification ability of the temporal learning rule is initially investigated using the XOR benchmark task. Then a practical optical character recognition (OCR) task is applied to investigate the functionality of our system on item recognition. The performance of the spike sequence decoding system is investigated by using a synthetic sequence of spikes. Finally, the ability of the whole system on sequence recognition is demonstrated. Discussions about our system are presented in section 5.4, followed by a conclusion in section 5.5.

5.2 The Integrated Network for Sequence Recognition

In this section, the whole system for sequence recognition is described, as well as the corresponding schemes used in different parts. The systematic model contains three functional parts including sensory encoding, learning and decoding (see Figure 5.1). The encoding neurons are used to generate spatiotemporal spike patterns that represent the external stimuli. The learning neurons focus on recognizing each input stimulus, and we call this recognition process as item recognition. The decoding neurons are used for recognizing the sequence order of the input stimuli based on the output of previous item recognition, and we call this recognition process as spike sequence recognition. In the learning layer, we use the PSD rule to train the neurons for item recognition since this rule is simple and efficient. Detailed descriptions about the PSD rule could be referred in Chapter 4.

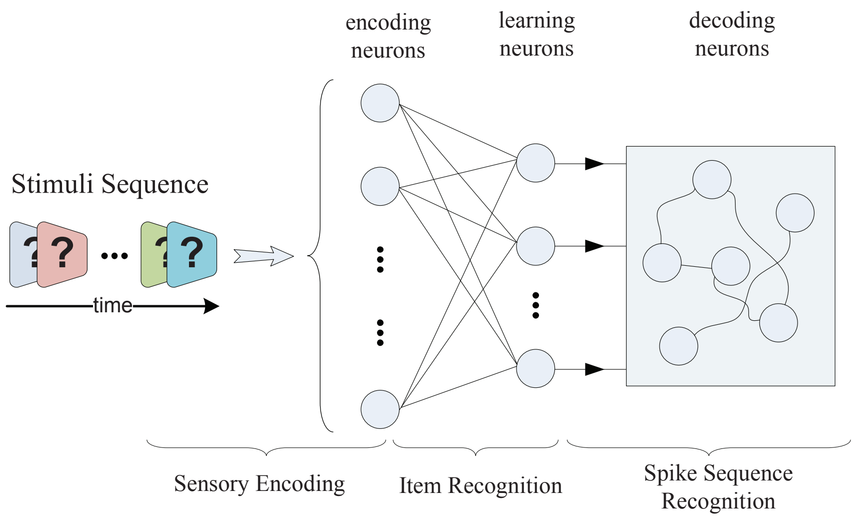

Figure 5.1: Illustration of the system for sequence recognition. The system contains three functional parts which are used for sensory encoding, item recognition and spike sequence recognition, respectively. The encoding neurons convert the external stimuli to spatiotemporal spike patterns. The learning neurons would recognize the content of each input item based on the corresponding spatiotemporal spike pattern. The sequence order of the input stimuli would be recognized through the decoding neurons.

5.2.1 Neural Encoding Method

An increasing body of evidence shows that action potentials are related to the phases of the intrinsic subthreshold membrane potential oscillations ($SMOs$) [123–125]. These observations support the hypothesis of a phase code [24, 113, 126]. Such a coding method can encode and retain information with high spatial and temporal selectivity [24]. Following the coding methods presented in [24, 113], we propose a new simple phase encoding method. Our encoding mechanism is presented in Figure 5.2.

Each encoding unit contains a positive neuron (Pos), a negative neuron (Neg) and an output neuron (E_{out}). Each encoding unit is connected to an

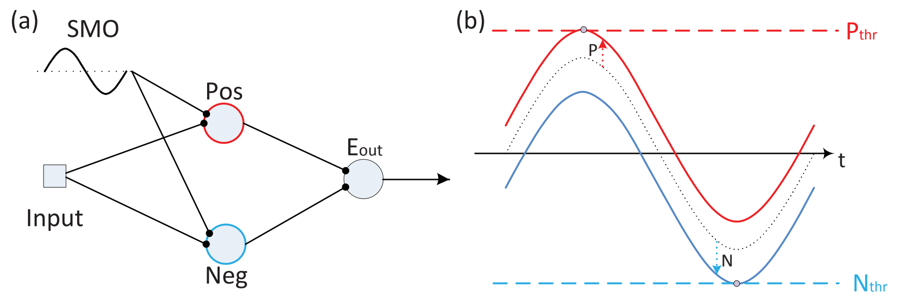

Figure 5.2: Illustration of the phase encoding method. (a) shows the structure of an encoding unit. Each encoding unit contains a positive neuron (Pos), a negative neuron (Neg) and an output neuron (E_{out}). The encoding unit receives signals from an input and a subthreshold membrane potential oscillation (SMO). (b) shows the dynamics of the encoding. A positive (negative) input will drive the membrane potential upwards (downwards) from the SMO. Whenever the membrane potential crosses the threshold (P_{thr} or N_{thr}), the neuron (Pos or Neg) will fire. The firing of either the Pos neuron or the Neg neuron will immediately trigger the firing of the E_{out} neuron.

input signal and a SMO. A positive (negative) input will cause an upward (downward) shift from the SMO. The firing of either the Pos neuron or the Neg neuron will immediately cause the firing of the E_{out} neuron. The SMO for the i-th encoding unit is described as:

$$SMO_i = M\cos(\omega t + \phi_i) \tag{5.1}$$

where M is the magnitude of the SMO, ω is the phase angular velocity and ϕ_i is the initial phase. ϕ_i is defined as:

$$\phi_i = \phi_0 + (i-1)\cdot\Delta\phi \tag{5.2}$$

where ϕ_0 is the reference phase and $\Delta\phi$ is the phase difference between nearby encoding units. We set $\Delta\phi = 2\pi/N_{en}$ where N_{en} is the number of encoding units.

5.2.2 The Sequence Decoding Method

In this part, we describe the sequence decoding method used for the decoding neurons in our system. A network of neurons with dendritic bistable plateau potentials can be used to recognize spike sequences [121]. Based on this idea, we build our decoding system as presented in Figure 5.3. This decoding network can recognize a specific sequence order of the spike inputs from the excitatory input neurons.

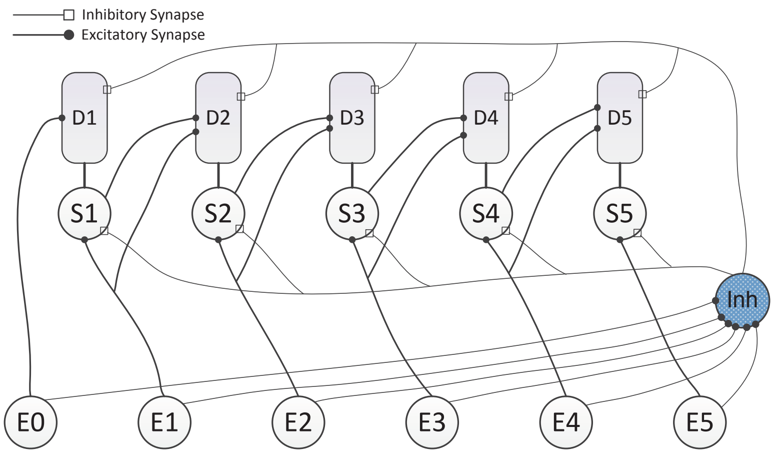

Figure 5.3: The neural structure for spike sequence recognition. E0-5 denote the excitatory input neurons. S1-5 and D1-5 denote the soma and the dendrite respectively. Inh denotes the inhibitory neuron.

The dynamics of the membrane potential of the soma is described as:

$$\tau_{sm}\frac{dV_{sm}}{dt} = -\left(V_{sm} - E_r\right) + g_{ds}(V_{dr} - V_{sm}) + I_s + I_A + I_{ns} \tag{5.3}$$

where V_{sm} and V_{dr} denote the potential of the soma and the dendrite respectively; $\tau_{sm} = 20\ ms$ is the membrane time constant; $E_r = -70\ mV$ is the resting membrane potential; $g_{ds} = 0.35$ is the conductance from the dendrite to the soma; I_s is the synaptic current on the soma; I_A is the A-type potassium current; I_{ns} is a background current, and is set to zero here.

The A-type potassium current [127, 128] is activated near the resting potential and inactivated at more depolarized potentials. I_A in the soma is given by:

$$I_A = -g_A \cdot a_\infty \cdot V_{sm}^3 \cdot b(t) \cdot (V_{sm} - E_K) \tag{5.4}$$

where $g_A = 10$ is the conductance; $E_K = -90\ mV$ is the reversal potential of the potassium current; a_∞ and $b(t)$ are the activation and inactivation variables respectively, and they are given by:

$$\begin{cases} a_\infty = \frac{1}{1+exp\left(-(V_{sm}+70)/5\right)} \\ \tau_A \frac{db}{dt} = -b + \frac{1}{1+exp\left((V_{sm}+80)/6\right)} \end{cases} \tag{5.5}$$

where $\tau_A = 5\ ms$ is a time constant.

The synaptic current on the soma is given by:

$$I_s = -g_{As} \cdot (V_{sm} - E_E) - g_{Gs} \cdot (V_{sm} - E_I) \tag{5.6}$$

where g_{As} and g_{Gs} are the alpha-amino-3-hydroxy-5-methyl-4-isoxazolepropionic acid (AMPA) and gamma-amino-butyric-acid (GABA) synaptic conductances respectively. The AMPA and GABA synaptic conductances mediate synaptic excitation and inhibition respectively. $E_E = 0\ mV$ and $E_I = -75\ mV$ are the reversal potential of excitatory and inhibitory synapses respectively.

The dynamics of the membrane potential of the dendrite is described as:

$$\tau_{dr} \frac{dV_{dr}}{dt} = -(V_{dr} - E_r) + g_{sd} \cdot (V_{sm} - V_{dr}) + I_{dr} \tag{5.7}$$

where $\tau_{dr} = 10\ ms$ is the time constant of the dendrite; $g_{sd} = 0.05$ is the conductance from the soma to the dendrite; I_{dr} is the synaptic current on the dendrite, and is given by:

$$\begin{aligned} I_{dr} = & - g_{Ad} \cdot (V_{dr} - E_E) - g_{Gd} \cdot (V_{dr} - E_I) \\ & - \frac{g_{Nd} \cdot V_{dr}}{1 + exp\big(- (V_{dr} + 30)/5 \big)} \end{aligned} \tag{5.8}$$

where g_{Ad} and g_{Gd} are the AMPA and GABA synaptic conductances respectively; g_{Nd} is the N-methyl-D-aspartate (NMDA) synaptic conductance that is responsible for the transient bistable plateau potential.

An incoming spike arrives at a synapse with strength G will cause changes on synaptic conductances g: $g \to g{+}G$. On the dendrite, a spike to an excitatory synapse will cause $g_{Ad} \to g_{Ad} + G$ and $g_{Nd} \to g_{Nd} + 5G$. Without incoming spikes, all the synaptic conductances will decay exponentially. The decay time constants for both the AMPA and GABA conductances are $5\ ms$. For the NMDA conductance, the decay time constant is $150\ ms$. g_{Nd} is not allowed to exceed 10 due to a saturation.

The inhibitory neuron is modeled as a single compartment quadratic LIF neuron [120, 121]. It can respond with a short latency (here $2\ ms$) to an excitatory spike input. The details of the inhibitory neuron model are described in [120].

5.3 Numerical Simulations

In this section, several experiments are presented to demonstrate the characteristics of our model. Through simulations, we investigate the abilities of our system mainly for item recognition and sequence recognition. A correct recognition on the input items is an essential step for further recognizing a specific sequence.

Firstly, in section 5.3.1, the exclusive OR (XOR) problem is used to preliminarily analyze the classification ability of the temporal learning rule on spike patterns. In section 5.3.2, a set of optical characters with images of digits 0-9 is used for analysis, and the performance of the item recognition system is individually investigated and analyzed with these digits under different noisy conditions. Section 5.3.3 shows the performance of our spike sequence decoding system where the downstream neurons could recognize a specific spike sequence. Finally, in section 5.3.4, the performance of the whole system on recognizing both the items and the sequence order is presented.

5.3.1 Learning Performance Analysis of the PSD Rule

The XOR problem is a linearly nonseparable task, and it is a benchmark widely used for investigating the classification ability of SNNs recently [34, 129–131]. Thus, we also use the XOR problem to investigate the performance of the PSD rule firstly. Different from approaches in [34, 129, 130] where a simple output with only one single spike is used, we apply the PSD rule to represent the categories by the associated target trains with multiple precise-timing spikes.

Similar to the setup in [47, 131], we randomly generate two homogeneous poisson spike trains with a firing rate of $50\ Hz$ in a time window of $200\ ms$. These two spike trains represent 0 or 1 respectively, and they are used to form the four inputs of the XOR problem: (0, 0), (0, 1), (1, 0) and (1, 1) (see Figure 5.4(a)). We also employ the concept of reservoir computing with a network of Liquid State Machine (LSM) like in [47, 131, 132]. The LSM

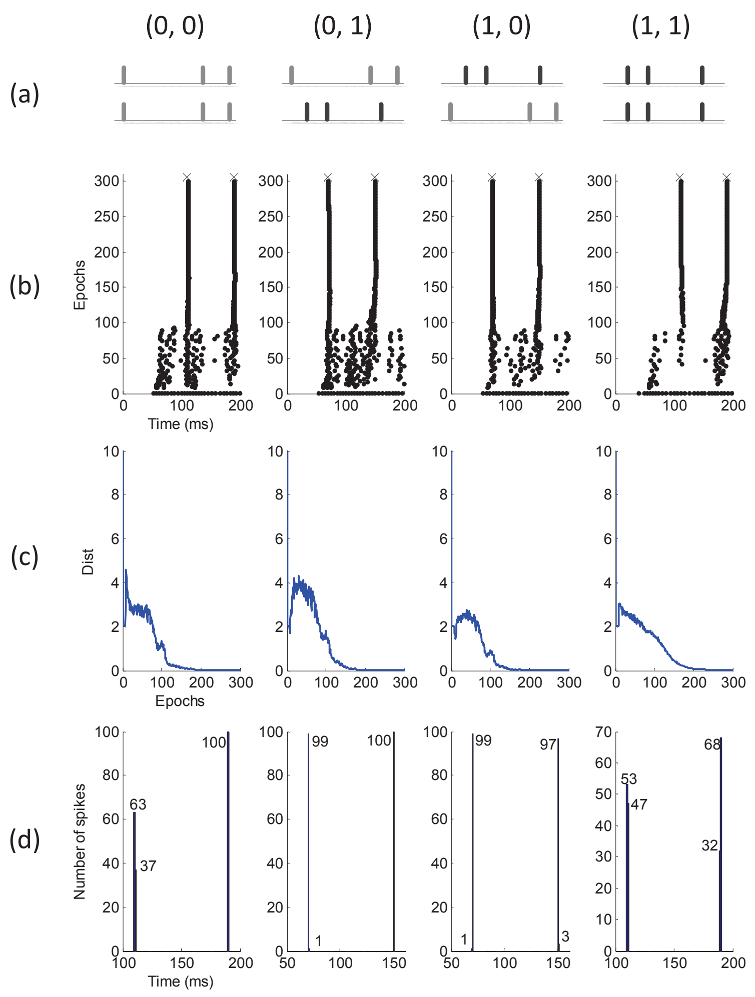

Figure 5.4: The performance of the PSD rule on the XOR task. (a) is an illustration of the four inputs of the XOR task. (b) shows the output spike signals for each of the four input patterns during learning in a typical run. '×' denotes the desired spike time. (c) and (d) are the results of the output neuron after 100 runs. (c) is the average spike distance between the desired and the actual output spike trains. The average spike distance for each input pattern is presented. (d) is the spike histogram showing the distribution of the actual output spikes.

uses spiking neurons connected by dynamic synapses to project the inputs to a higher-dimensional feature space, which can facilitate the classification. The network used in this experiment consists of two input neurons, a noise-free reservoir with 500 LIF neurons and one readout neuron.

We specify a target spike train for each category. For inputs of (0, 0) and (1, 1), the output neuron is trained to spike at $[110,\ 190]$ ms, while for (0, 1) and (1, 0), it is trained to fire another target train of $[70,\ 150]$ ms. The initial synaptic weights of the output neuron are randomly drawn from a normal distribution with a mean value of 0.5 nA and a standard deviation of 0.2 nA. This initial condition of synaptic weights is also used for other experiments in this chapter. These synaptic weights are adjusted by the PSD rule with a set of learning parameters $\eta = 0.01$ and $\tau_s = 10$ ms. The results are averaged over 100 runs.

Figure 5.4(b) shows the results of a typical run, with the actual output spikes for each of the four input patterns during the learning. At the beginning, both the firing rates and the precise timings of the output spike trains are different from those of the target spike trains. After tens of learning epochs, the readout neuron can gradually learn to fire the target spike trains according to different input patterns. After hundreds of learning epochs, the readout neuron stabilizes at the target spike trains. This phenomenon can be also seen from the spike distance between the actual and the target spike trains (see Figure 5.4(c)). A larger spike distance occurs at the beginning due to the initial conditions, followed by a gradually decreasing spike distance along the learning, and it finally converges to zero. Figure 5.4(d) shows the distribution of the actual output spikes corresponding to the four input patterns. From these histograms, we can see our approach with the PSD rule obtains better performance than that in [131]. Firstly, there are no undesired extra spikes or missing desired spikes in our approach. In the 100 runs of experiments, the trained neuron fires exactly 100 spikes around each desired time. Secondly, the actual output spikes are precisely and reliably close to the desired time. The maximum error of spike time is around $1\ ms$. Thus, the learning success rate of our approach is higher than that in [131].

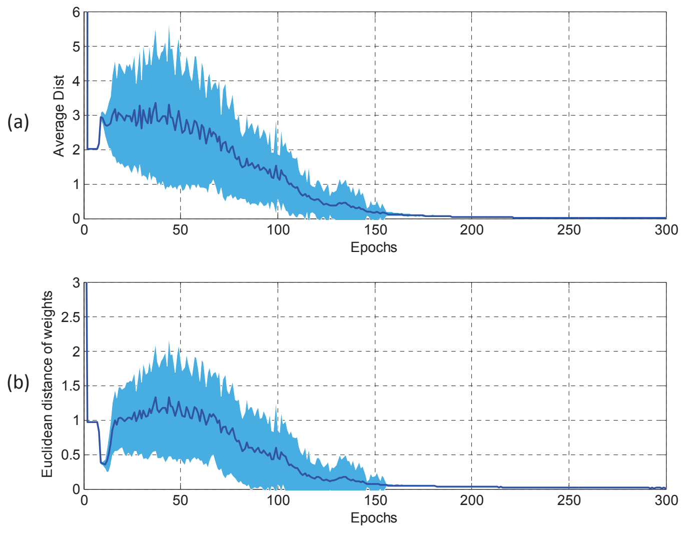

Figure 5.5: The convergent performance. (a) shows the average spike distance over all the four input patterns. (b) is the Euclidean distance between the weights before and after each learning epoch. All the results are averaged over 100 runs.

Figure 5.5 shows the convergent performance during the learning process. The average spike distance over all four input patterns is presented as well as the Euclidean distance between the weights before and after each learning epoch. As can be seen from Figure 5.5, irregular distances occur at the first several learning epochs because of the random initial conditions. After that, the distances gradually decrease and converge to zero. The zero spike distance corresponds to the readout neuron firing exactly the target spike train, and the zero weight distance implies that there are no more changes occurring on the weights. These two distance graphs also show the ability of the PSD rule to modify the weights in order to produce the desired output spikes. Either of these two types of distance can be used as a stopping criterion for the learning process.

This experiment with the XOR problem demonstrates the ability of the PSD rule for classifying spatiotemporal patterns. Detailed investigations on performance of item recognition and sequence order recognition of our system are presented at following.

5.3.2 Item Recognition

In this section, we consider the performance of our system on the item recognition. A set of optical characters with images of digits 0-9 is used. Each image has a size of 20 $\times$ 20 black/white (B/W) pixels, and each would be destroyed by a reversal noise where each pixel is randomly reversed with a probability denoted as the noise level. Some clean and noisy samples are demonstrated in Figure 5.6.

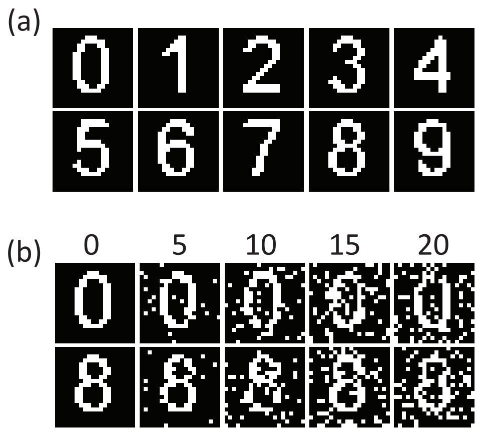

Figure 5.6: Illustration of the OCR samples. (a) shows the template images. (b) shows image samples with different levels of reversal noise.

The phase encoding method illustrated in Figure 5.2 is used to convert the digit images into spatiotemporal spike patterns. Each pixel acts as an input to each encoding unit, with a W/B pixel causing a positive/negative shift from the SMO. Through a fine tuning of the values of M, *P*/*N* and P_{thr}/N_{thr}, we set the encoded spikes to occur at peaks of the $SMOs$. The number of encoding units is equal to the number of pixels which is 400 here. We set the oscillation period of the $SMOs$ to be 200 ms which corresponds to a frequency of 5 Hz.

We select 10 learning neurons trained by the PSD rule, with each learning neuron corresponding to one category. The learning parameters in the PSD rule are set to be $\eta = 0.06$ and $\tau_s = 10\ ms$. All the learning neurons are trained to fire a target spike train with the corresponding category. The target spike train is set to be evenly distributed over the time window T_{max} (200 ms here) with a specified number of spikes n. The firing time of the i-th target spike: $t_i = i/(n+1) \cdot T_{max}$, $i = 1,\ 2...n$. We choose $n = 4$ by default, otherwise will be stated. In item recognition, a relative confidence criterion is used [119] with the PSD rule, where the incoming pattern is represented by the neuron that fires the most closest spike train to its target spike train.

In this section, two noisy scenarios are considered: (1) spike jitter noise where a Gaussian jitter with a standard deviation (denoted as the jitter strength) is added into each encoded spike; (2) reversal noise (as illustrated in Figure 5.6(b)) where each pixel is randomly reversed with a probability denoted as the noise level.

Spike jitter noise

In this scenario, the templates of the digit images are firstly encoded into spatiotemporal spike patterns. After that, jitter noises are added to generate noisy patterns. The learning neurons are trained for 100 epochs with a jitter strength of 2 ms. In each learning epoch, a training set of 100 patterns, with 10 for each category, is generated. After training, a jitter range of 0-8 ms is used to investigate the generalization ability. The number of the testing patterns for each jitter strength is set to 200. The PSD rule is applied with different numbers of target spikes (n =1, 2, 4, 6, 8, 10). All the results are averaged over 100 runs.

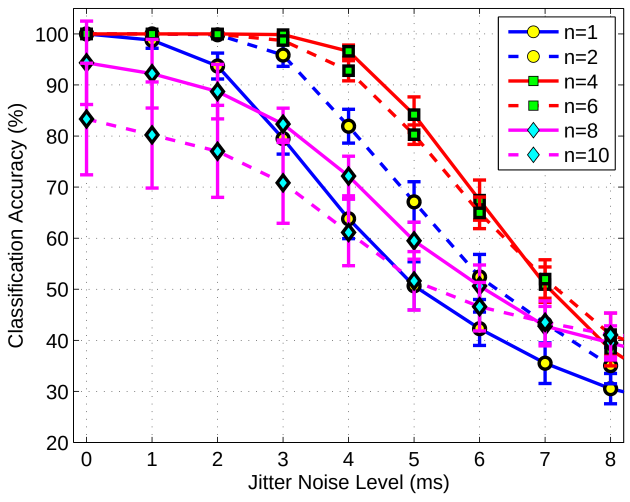

Figure 5.7: The performance of the PSD rule with different numbers of target spikes under the case of jitter noise.

Figure 5.7 shows the effects of the number of the target spikes on the learning performance of the PSD rule. As can be seen from Figure 5.7, when n is low (e.g. 1, 2), the recognition performance is also relatively low. An increasing number of the target spikes can improve the recognition performance significantly (see $n = 1, 2 \rightarrow n = 4, 6$). However, a further increase in the number of target spikes ($n = 6 \rightarrow n = 8, 10$) would reduce the recognition performance. The reasons for this phenomenon are due to the local temporal features associated with each target spike. For small number of target spikes, the neurons make decision based on a relatively less number of temporal features. This small number of features only covers a part range of the whole time window, which inevitably leads to a lower performance compared to a more number of spikes. However, when the number of spikes continues increasing, an interference of local learning processes [47] occurs and increases the difficulty of the learning. Thus, a higher number of spikes normally cannot lead to a better performance due to the interference.

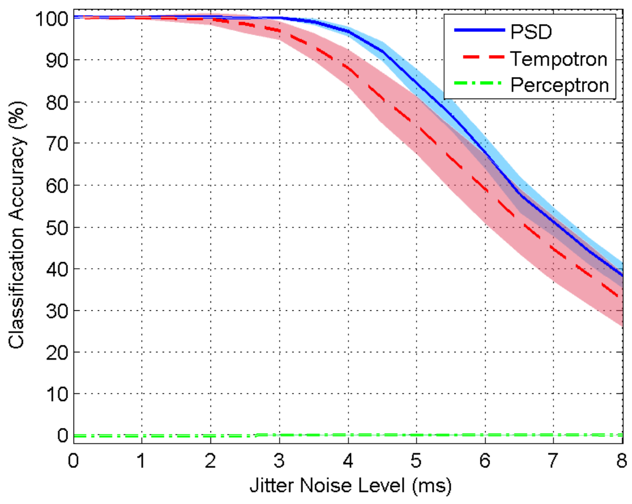

Figure 5.8: The performance of different rules under the case of jitter noise. The PSD rule uses $n = 4$ target spikes. The PSD rule outperforms the other two rules in the considered task.

Figure 5.8 shows the performance of different learning rules for the same classification task. We use a similar approach for the perceptron rule as in [131, 132], where the spatiotemporal spike patterns are transformed into continuous states by a low-pass filter. The target spike trains are separated into bins of size t_{smp}, with $t_{smp} = 2\ ms$ being the sampling time. The target vectors for the perceptron contain values of 0 and 1, with 1 (or 0) corresponding to those bins that contain (or not contain) a target spike in the bin. The input vectors for the perceptron are sampled from the continuous states with t_{smp}. The input pattern will be classified by the winning perceptron that has the closest output vector to the target vector.

As can be seen from Figure 5.8, the PSD rule outperforms both the tempotron rule and the perceptron rule. The inferior performance of the perceptron rule can be explained. The complexity of the classification for the perceptron rule depends on the dimensionality of the feature space and the number of input vectors for decisions. A value of $t_{smp} = 2\ ms$ will generate 100 input vectors for each input pattern. These 100 points in 400-dimensional space are to be classified into 1 or 0. This can increase the difficulty for the perceptron rule, let alone considering a large number of input patterns from different categories. Without separating the time window into bins, the spiking neurons by their nature are more powerful than the traditional neurons such as the perceptron. Both the PSD rule and the tempotron rule are better than the perceptron rule. The PSD rule is better than the tempotron rule since the PSD rule makes a decision based on a combination of several local temporal features over the entire time window, but the tempotron rule only makes a decision by firing one spike or not based on one local temporal feature.

Reversal noise

In this scenario, the reversal noise is used for generating noisy patterns as illustrated in Figure 5.6(b). The learning neurons are trained for 100 epochs with a reversal noise level randomly drawn from the range of 0-10$\%$ in each learning epoch. Meanwhile, a training set of 100 noisy patterns, with 10 for each category, is generated for each learning epoch. After training, another number of 100 noisy patterns are generated and used to test the generalization ability.

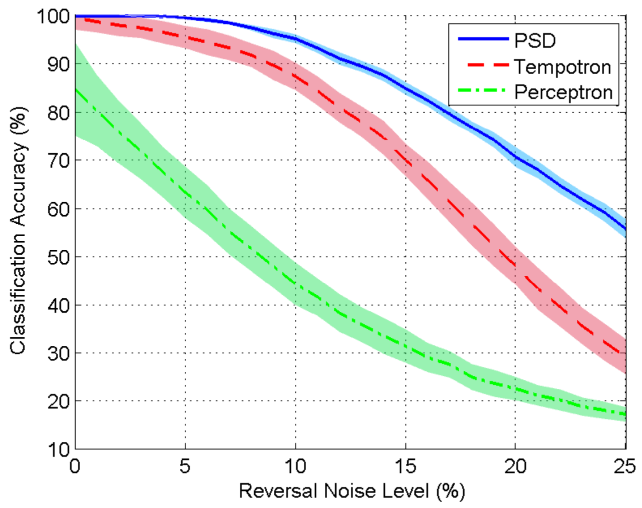

Figure 5.9: The performance of different rules under the case of reversal noise. The PSD rule uses $n = 4$ target spikes. The PSD rule outperforms the other two rules even when the noise level is high.

As can be seen from Figure 5.9, the performances of all the three rules decrease with the increasing noise level. The performance of the PSD rule again outperforms the other two rules as in the previous scenario. Spiking neurons trained by the PSD rule can obtain a high classification accuracy (around 85%) even when the reversal noise reaches a high level (15%). The performance of the perceptron rule in this scenario is much better than that in the previous scenario. This is because of the type of the noise. The performance of the perceptron rule is quite susceptible to the changes in state vectors. Every spike of the input spatiotemporal spike patterns in the case of spike jitter noise suffers a change, while in the case of reversal noise, a change only occurs with a probability of the reversal noise level. This is to say, the elements in a filtered state vector have a less chance to change under the reversal noise than that under the jitter noise. Thus, the performance of the perceptron rule under the reversal noise is better than that under the jitter noise.

These results demonstrate our item recognition is robust to different noisy sensory inputs. A reliable recognition on the incoming item is essential for further sequence recognition.

5.3.3 Spike Sequence Decoding

In this section, we investigate the performance of our decoding system for spike sequence recognition. The structure of this decoding system is presented in Figure 5.3. This decoding structure can recognize a specific sequence of E0, E1...E5. We denote the synaptic connections as: E0→D1 (G_0), E1-5→S1-5 (G_1), E1-5→D2-5 (G_2), S1-5→D2-5 (G_3), Inh→D1-5 (G_4), and Inh→S1-5 (G_5). We set $G_0 = 5$, $G_1 = 2.5$, $G_2 = G_3 = 3$, $G_4 = 5$, $G_5 = 6$. We generate a spike input feeding into our decoding system, with Figure 5.10 showing a 200 ms interval between nearby spikes and 230 ms for Figure 5.11.

As can be seen from Figure 5.10, the decoding system successfully recognizes the sequence through a firing from S5. A strong, excitatory input to the dendrite can make its potential go to a plateau potential that is transiently stable for a time. The plateau potential of the dendrite then drives the potential of the soma to a high depolarized state. Without the plateau potential of the dendrite, the potential of the soma stays near the resting potential. We refer the high depolarized state of the soma as the UP state, and the state near the resting potential as the DOWN state. Two conditions are required to make a soma fire:

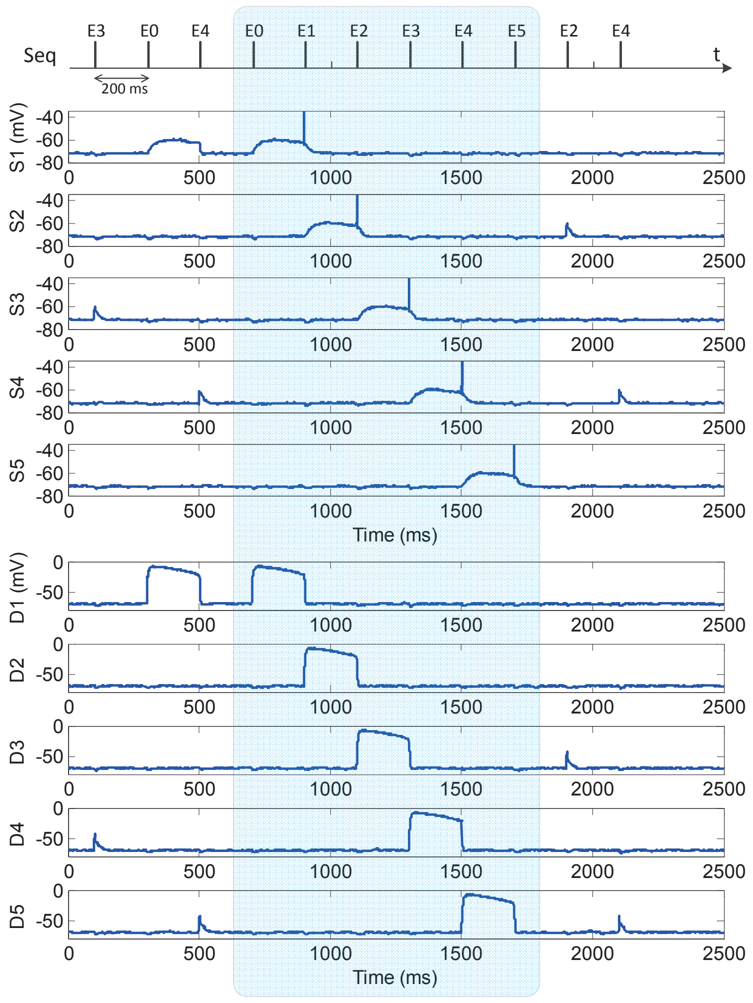

Figure 5.10: A reliable response of the spike sequence decoding system. A synthetic spike sequence is used as the input (denoted as 'Seq'). The target sequence pattern of E0, E1...E5 is highlighted by the shaded area. The potentials of the somas ($S1-5$) and the dendrites ($D1-5$) are shown. The interval spike time in the input sequence is 200 ms. The neurons can be successfully activated to fire when the target sequence presents.

(1) the potential of the soma sustains in the UP state (2) when an excitatory spike input comes to this soma.

Under the experimental setup of our decoding system, the UP state of the soma can sustain for a period around 225 ms, during which the soma can reliably fire a desired spike when corresponding excitatory neuron fires. We refer this period as the reliable period. When the time interval between spikes is

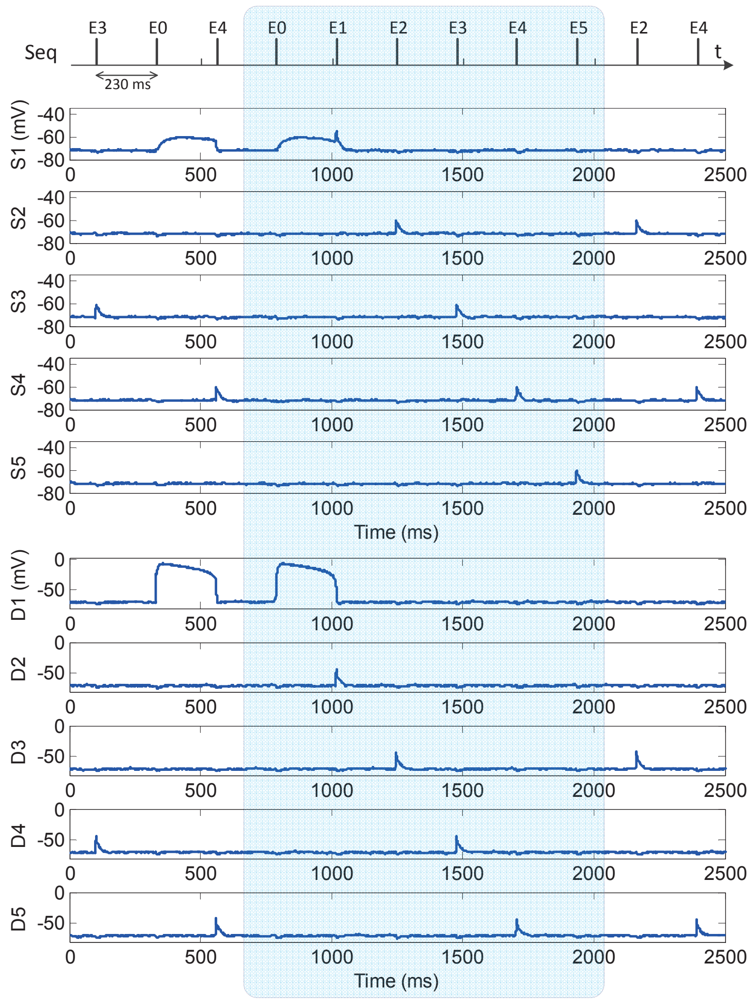

Figure 5.11: An unreliable response of the spike sequence decoding system. The interval spike time in the input sequence is 230 ms. When the interval time is over a certain range (225 ms for this experimental setup), the neurons cannot be activated to fire even when the target sequence presents. This is because that the potential of the soma cannot sustain in the UP state for a such long interval.

shorter than the reliable period, the decoding system can perform the recognition well (see Figure 5.10). When the time interval between input spikes is longer than the reliable period, the UP state of a soma no longer sustains at a reliably high state. This leads to that a corresponding excitatory input spike no longer reliably drives a spike on the soma (see Figure 5.11).

The experimental results indicate that our spike sequence decoding system is invariant to changes in the intervals between input spikes within a certain range.

5.3.4 Sequence Recognition System

In this section, the performance of the proposed whole system is investigated. The sensory encoding, temporal learning and spike sequence decoding are consistently combined together for sequence recognition. We perform the experiment with the previous digit set used in section 5.3.2.

These optical digits are used to form a sequence pattern, with each digit image in the sequence being destroyed by a reversal noise level of 15%. We can specify a target sequence through building connections between the output neurons of the item recognition network and the excitatory input neurons of the spike sequence decoding network. For the reason of simplicity, we specify a target sequence order of digits as: 012345. Thus, the learning neurons corresponding to the categories in this target sequence are connected to the excitatory input neurons in the sequence decoding network one by one. Each digit image is presented for $200\ ms$. Additionally, the interval between two successive images is not allowed to exceed $25\ ms$, guaranteeing a reliable performance of the spike decoding system, which is further explained in the following part.

We construct a sequence pattern of 6 segments, with 6 images for each segment. Every image in this sequence is randomly chosen from the 10 categories. Then the target sequence of 012345 is embedded into this sequence, with a probability of $1/3$ replacing each initial segment in the sequence. After this, we feed the whole sequence to our system. The target of our system is to detect and recognize the target sequence embedded in the whole sequence.

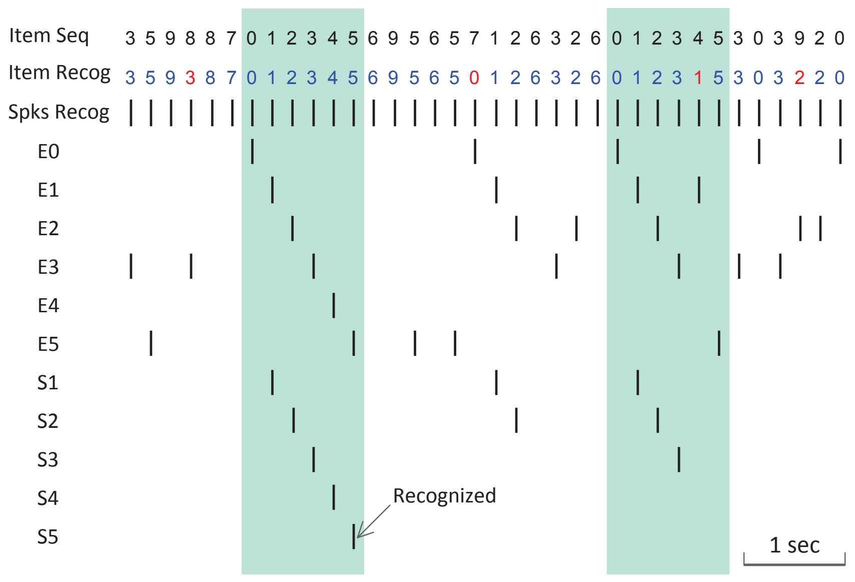

Figure 5.12: The performance of the combined sequence recognition system. An image sequence input is fed into the sequence recognition system. Each image suffers a reversal noise of 15%. The target of this system is to detect and recognize a specified target sequence of 012345 (the shaded areas). 'Item Seq' denotes the input sequence of the images. 'Item Recog' is the output results of the learning neurons, with the blue/red color representing correct/incorrect recognition. Each output of the learning neurons results a spike in the corresponding excitatory input neurons of the spike decoding network ('Spks E'). S1-5 denote the spike output of neurons in the sequence decoding network.

Figure 5.12 shows the performance of our system for sequence recognition. An important step before recognizing the sequence order is to correctly recognize each input item. Only after knowing what is what, a recognition on the sequence order can be applied. The detected target sequence is represented by the firing of S5. As can be seen from Figure 5.12, the first target sequence is successfully recognize through the sequential firing of S1-5, while the second target sequence is not correctly recognized due to a failure recognition on image '4'.

In addition, we conduct another experiment, where one item in the target sequence is semi-blind. This semi-blind item is conditioned to a specific range.

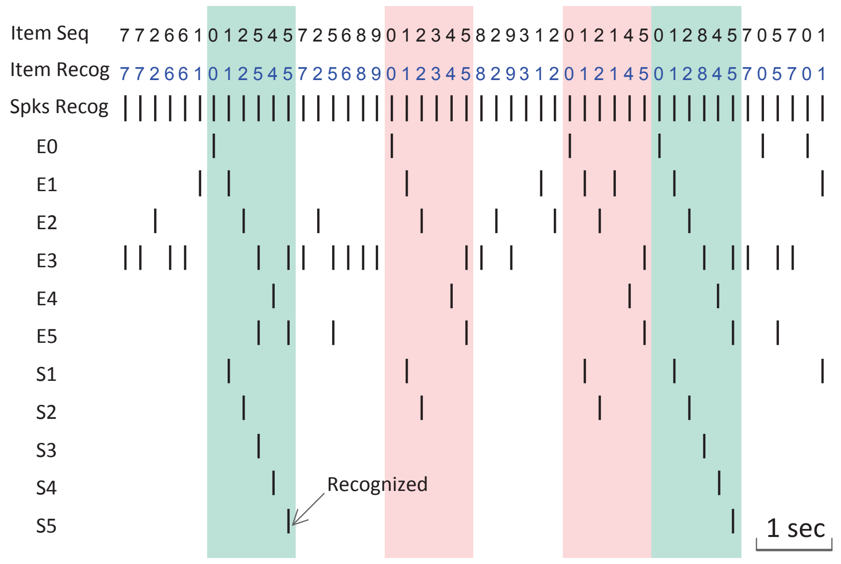

Figure 5.13: Performance on a target sequence with one semi-blind item. The input sequence is considered in a noise-free condition. The target of this system is to detect and recognize a specified target sequence of 012?45 where '?' is from a specific range of 5-9 (illustrated in the shaded light-cyan areas). The shaded light-pink areas show some interference sequence patterns where '?' is not chosen from the allowed range.

We specify a target sequence of 012?45, where '?' is restricted to the range of 5-9. Other digits of '?' being out of this specific range lead to non-target sequences. For the sake of simplicity, this experiment is conducted in a noise-free condition. We randomly construct an input sequence with 48 items, and then embed the target sequences, as well as some interference sequences, into the input sequence. In order to detect and recognize the semi-blind target sequences, we reconstruct the connections between the output neurons of the item recognition network and the excitatory input neurons of the spike sequence decoding network, with all learning neurons for digit 5-9 connecting to E3 in Figure 5.3. Other connections are not changed. As can be seen from Figure 5.13, the semi-blind target sequences are successfully recognized, and those interference sequences are also successfully declined. Our system successfully recognizes the target sequence of 012?45 with '?' only belonging to 5-9.

These experiments show that our system with spiking neurons can perform the sequence recognition well, even under some noisy conditions. Item recognition is an essential step for a successful recognition of the target sequence. The step before recognizing the sequence order is to recognize what are the items in the input sequence. A failure recognition of the item in the target sequence would directly affect the further recognition on the sequence order.

5.4 Discussions

In this chapter, a biologically plausible system with spiking neurons is presented for sequence recognition. Discussions based on the simulation results are as follows.

5.4.1 Temporal Learning Rules and Spiking Neurons

The PSD rule [119], proposed in the concept of processing and memorizing spatiotemporal spike patterns, is applied in our system for item recognition. In the PSD rule, the synaptic adaptation is driven by the precise timing of the actual and the target spikes. Without a complex error calculation, the PSD rule is simple and beneficial for computation [119]. According to the classification tasks considered in this chapter, the PSD rule outperforms both the tempotron rule [33,45] and the perceptron rule [131,132].

The computational power of the spiking neurons over the traditional neurons (perceptrons) is reflected by the better performance of both the PSD rule and the tempotron rule than the perceptron rule (see Figure 5.8 and Figure 5.9). This is because that the spiking neurons, by their nature, are designed for processing in a time domain with a complex evolving dynamics on the membrane potential. A major difference between the perceptrons and the spiking neurons is this dynamic membrane potential. The perceptrons calculate current states in a static manner that only based on the current inputs, while the spiking neurons evolve current states in a dynamic manner that not only based on the current inputs but also the past states. Additionally, due to the ability of the spiking neurons to operate online, it can benefit the computation of a sequential procession with time elapsing.

Between the two temporal learning rules for spiking neurons, the performance of the PSD rule is better than the tempotron rule. The decisions made by the neurons under the PSD rule are based on a combination of several local temporal features over the whole time window. By contrast, the tempotron rule trains a neuron to make a decision only based on one local temporal feature if the neuron is supposed to fire a spike. A decision based on several local temporal features would result in a better performance than that only based on one local temporal feature. In addition, the PSD rule is not limited to a classification task, but it can also train a neuron to associate spatiotemporal patterns with the specified desired spike trains.

5.4.2 Spike Sequence Decoding Network

Our spike sequence decoding network is biologically realistic that can behave like FSM to recognize spike sequences [120, 121]. The functionality of this network is achieved through transitions between the UP and DOWN states of neurons. Transitions between bistable membrane potentials are widely observed through various experiments in cortical pyramidal neurons *in vivo* [133, 134]. The transitions between the states are controlled by feedforward excitation, lateral excitation and feedforward inhibition. The neurons enter the UP state if their dendrites have a plateau potential. The neurons will return to the DOWN state from the UP state when enough long time elapses without excitatory input spikes. In addition, the recognition is robust to time warping of the sequence. The recognition is intact as long as the interval between input spikes lies in a specific range which can be quite broad (see Figure 5.10 and Figure 5.11). Invariance to time warping is beneficial for tasks like speech recognition [7, 135].

5.4.3 Potential Applications in Authentication

Our system provides a general structure for sequence recognition. With proper encoding mechanisms, this system could also be applied to acoustic, tactual and olfactory signals in addition to visual signals. The processes of the item recognition and the sequence order recognition in our system could be used for user authentication to access approval. It provides a double-phase checking scheme for gaining access. Only if both the items and also their orders are correct, the person would be allowed to access.

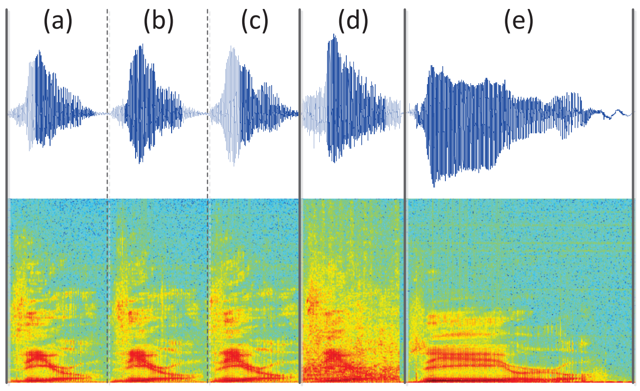

Figure 5.14: Voice samples of digit ‘Zero’. (a), (b) and (c) are samples spoken by a person in clean conditions with a similar manner for each recording. (d) is a sample under a 5dB noise and (e) is a warped sample spoken in a different manner. The top panel and the bottom panel show the sound waves and the corresponding spectrograms respectively.

We preliminarily applied these concepts to the speech task with our previously proposed encoding scheme [44] for sounds. The voices of ten digits were considered. It is still a very challenging task for spiking neurons to process audio signals due to variations of speed, pitch, tone and volume. Our system could be successful in the case where words are spoken in a similar manner such as samples (a)-(c) in Figure 5.14, but it would be failed if the voice is changed significantly like (d) and (e) in Figure 5.14. Further study is required for speech recognition with spiking neurons, and further results would be presented in our next stage.

5.5 Conclusion

In this chapter, a biologically plausible network is proposed for sequence recognition. This is the first attempt to solve the sequence recognition with the network of spiking neurons by considering both the upstream and the downstream neurons together. The system is consistently integrated with functional parts of sensory encoding, learning and decoding. The system operates in a temporal framework, where the precise timing of spikes is considered for information processing and cognitive computing. The recognition performance of the system is robust to different noisy sensory inputs, and it is also invariant to changes in the intervals between input stimuli within a certain range.

Chapter 6

Temporal Learning in Multilayer Spiking Neural Networks Through Construction of Causal Connections

This chapter presents a new supervised temporal learning rule for multilayer spiking neural networks. We present and analyze the mechanisms utilized in the network for the construction of causal connections. Synaptic efficacies are finely tuned for resulting in a desired post-synaptic firing status. Both the PSD rule and the tempotron rule are extended to multiple layers, leading to new rules of multilayer PSD (MutPSD) and multilayer tempotron (MutTmptr). The algorithms are applied successfully to classic linearly non-separable benchmarks like the XOR and the Iris problems.

6.1 Introduction

In biological nervous systems, neurons communicate with others through action potentials (spikes). To emulate this phenomenon, spiking neurons are introduced to process spike information. Due to the spiking feature, the spiking neurons are more biologically plausible and computationally powerful than traditional neuron models like perceptron.

Information could be carried by spikes either in a rate-based form or a precise spike-based form. Increasing evidence shows that individual spikes with precise time play a significant role in transmitting information. Neurons can learn more and faster from the spike-based code than the rate-based code.

Considering the spatiotemporal spike patterns, many learning rules have been proposed to understand how neurons process the information. Most of temporal learning methods, such as tempotron [33], ReSuMe [37], Chronotron [36], SPAN [35] and PSD [119], only focus on the learning of single spiking neurons or single-layer SNNs. These learning rules are biologically plausible to some extent. However, the real nervous systems are extremely complex network with a large number of neurons interconnecting with each other. Investigations on the level of single neurons or single-layer networks might be insufficient to simulate the cognitive functions of the brain. Therefore, research on multilayer SNNs is demanded.

Some gradient-descent-based learning rules such as SpikeProp [34] and its extensions [50, 129] are proposed to train the network with hidden neurons to output a target spike train. The derivations of these rules are based on the explicit dynamics of the SRM model, which limit the applicability of these rules to other neuron models. The same problem is also involved in another gradient-descent-based rule proposed in [75]. Although the gradient-descent-based rules are effective, they lack biological explanation. The complex error calculation involved in the learning is at least questionable. In [130], an extension of the ReSuMe rule is proposed for multilayer SNNs, where the weights are updated according to STDP and anti-STDP processes. This ReSuMe-based multilayer learning rule requires back propagation of the network error. When the number of layers increases, the evaluation of the network error will become more complex. Again, such a complex error evaluation is also debatable considering the real nervous systems.

In this chapter, we propose a new supervised learning rule for multilayer spiking neural networks. This rule is an extension of the PSD rule introduced in Chapter 4. Without complex error evaluation, the learning is simple and efficient, and yet biologically plausible. In addition, we also proposed a multilayer learning for the tempotron rule. Through our multilayer learning, causal connections are constructed between layers of spiking neurons.

The rest of this chapter is organized as follows. In section 6.2, the proposed learning rules for multilayer SNNs are presented, including multilayer PSD (MutPSD) rule and multilayer tempotron (MutTmptr) rule. Heuristic discussions about our multilayer learning rules are presented in section 6.3. Section 6.4 presents the simulation results. Construction of the causal connections is firstly demonstrated. The properties and power of the MutPSD and MutTmptr rules are showcased by linearly non-separable tasks including the XOR problem and the Iris dataset task. Finally, discussions of the multilayer learning rules, as well as the conclusion, are presented in section 6.5.

6.2 Multilayer Learning rules

In this section, we describe the learning schemes in the feedforward multilayer spiking neural networks. Firstly, the neuron model used in this chapter is introduced. Then, the multilayer PSD (MutPSD) rule is described, followed by the introduction of multilayer tempotron (MutTmptr) rule.

6.2.1 Spiking Neuron Model

For the sake of simplicity, our neuron model consists of a leaky integrate-and-fire neuron driven by synaptic currents generated by its afferents. The potential of the neuron is a weighted sum of postsynaptic currents (PSCs) from all incoming spikes:

$$V(t) = \sum_i w_i I^i_{PSC}(t) + V_{rest} \tag{6.1}$$

where w_i and I^i_{PSC} are the synaptic efficacy and the PSC of the i-th afferent. V_{rest} is the rest potential of the neuron. The dynamics of the I^i_{PSC} is as follow:

$$I^i_{PSC}(t) = \sum_{t^j} K(t-t^j)H(t-t^j) \tag{6.2}$$

where t^j is the time of the j-th spike emitted from the i-th afferent neuron, $H(t)$ refers to the Heaviside function, K denotes a normalized kernel and we choose it as:

$$K(t-t^j) = V_0 \cdot \big(\exp(\frac{-(t-t^j)}{\tau_s}) - \exp(\frac{-(t-t^j)}{\tau_f}) \big) \tag{6.3}$$

where V_0 is a normalization factor such that the maximum value of the kernel is 1, τ_s and τ_f are the slow and fast decay constants respectively, and their ratio is fixed at $\tau_s/\tau_f = 4$.

For the neurons in the hidden layers, we utilize a fire-and-shutdown scheme as in [33]. This can guarantee a single spike scheme in the hidden neurons if the neurons receive enough strong stimulus. Increasing experimental evidence suggests that neural systems use exact time of single spikes to transmit information [23, 83, 136]. Visual system can analyze a new complex scene in less than 150 ms [23, 83]. This period of time is impressive for information processing considering billions of neurons involved. This suggests that neurons exchange only one or few spikes. In the tactile system, it is shown that the time of the first spike contains important information about the external stimuli [136]. In addition to the biological plausibility, first spikes also serve as an efficient way to transmit information. Subsequent brain region may learn more and earlier about the stimuli from the time of the first spikes [23]. The benefits of the first spike suit the role of hidden neurons acting as the information transmitter between the input and output neurons.

6.2.2 Multilayer PSD Rule

The PSD rule [119] for single neurons or single layer network is described as:

$$\frac{dw_i(t)}{dt} = \eta[s_d(t) - s_o(t)]I^i_{PSC}(t) \tag{6.4}$$

The polarity of the synaptic changes depends on three cases: (1) a positive error (corresponding to a miss of the spike) where the neuron does not spike at the desired time, (2) a zero error (corresponding to a hit) where the neuron spikes at the desired time, and (3) a negative error (corresponding to a false-alarm) where the neuron spikes when it is not supposed to.

In the single-layer PSD, only the direction of synaptic modification is used. The amount of modification depends on the current input PSC. Based on this idea, a multilayer PSD (MutPSD) rule can be developed. The instructor signals that only containing directions of modifications are back propagated to all synapses in the multilayer feedforward network, while the amount of synaptic change depends on the corresponding PSC received by each synapse.

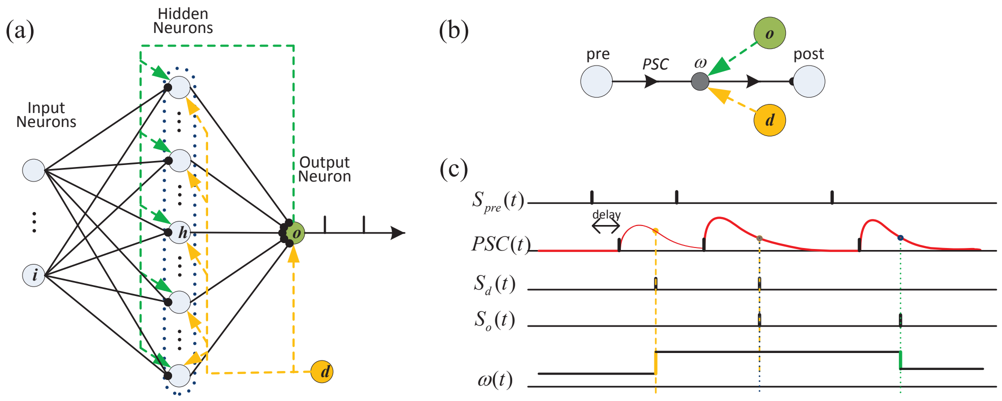

Figure 6.1: Structure and plasticity of multilayer PSD. (a) is the structure of the multilayer network where input neurons are connected to the output neuron through hidden neurons. (b) shows the synaptic structure. The synaptic plasticity in the multilayer network is driven by the desired signal (d) and the actual output signal (o). (c) demonstrates the scheme for synaptic plasticity. A desired spike will result in potentiation, while an actual output spike will lead to depression. The amount of synaptic modification depends on the PSC signal.

Figure 6.1(a) shows the multilayer structure. For the reason of simplicity, one layer of hidden neurons is considered, but the algorithm can be extended to networks with more hidden layers similarly. The instructor signals are used to guide the synaptic modification direction of all synapses. Considering the synaptic delays d, the MutPSD rule can be described as:

$$\begin{aligned}\Delta w_i &= \eta \int_0^\infty [s_d(t) - s_o(t)] I^i_{PSC}(t-d)dt \\ &= \eta \Big[\sum_g I^i_{PSC}(t^g_d - d) - \sum_h I^i_{PSC}(t^h_o - d) \Big]\end{aligned} \tag{6.5}$$

where t^g_d and t^h_o denotes the g-th desired spike and the h-th actual output spike, respectively. The synaptic structure is shown in Figure 6.1(b).

The dynamics of synaptic plasticity is demonstrated in Figure 6.1(c). Similar to the single-layer PSD, the weight adaptation in the MutPSD is triggered by the error between the desired and the actual output spikes, with positive errors causing long-term potentiation (LTP) and negative errors causing long-term depression (LTD). No synaptic change will occur if the actual output spike fires at the desired time. The amount of synaptic changes is determined by the signal I^i_{PSC}.

6.2.3 Multilayer Tempotron Rule

The tempotron learning rule [33] was introduced to train a single neuron to discriminate between spatiotemporal spike patterns. Neurons are trained to distinguish between two classes by firing at least one spike or remaining quiescent. Whenever a neuron failed to fire a spike corresponding to a positive pattern, LTP will occur; if the neuron fired a spike to a negative pattern, LTD will happen.

The tempotron rule and the PSD rule are similar to some extent. In both rules, the instructor signals are used to guide the direction of the synaptic modification, either potentiation or depression. The amount of synaptic change

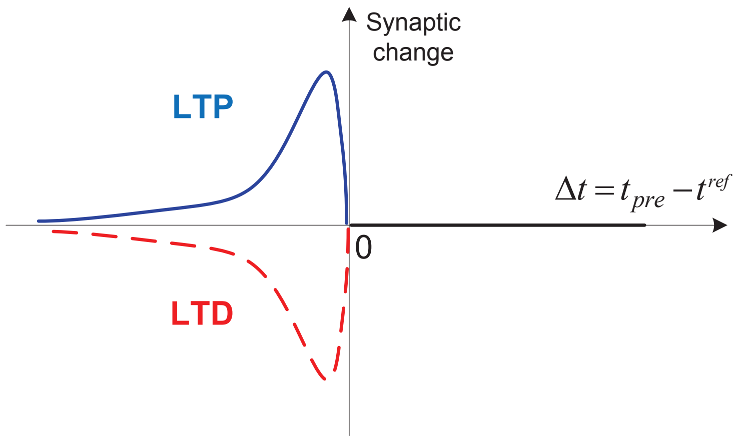

Figure 6.2: Similarity between the PSD rule and the tempotron rule on learning windows. The amount of synaptic change depends on the time difference Δt between the afferent spikes t_{pre} and the reference time t^{ref}.

depends on the time difference between the afferent spikes and the reference time t^{ref}. Figure 6.2 shows the learning windows. In the tempotron rule, t_{max} is the reference time for updating synaptic weights. In the PSD rule, t^{ref} refers to t_d or t_o. In the tempotron rule, it refers to t_{max}. In both the tempotron rule and the PSD rule, only the pre-synaptic spikes that precede the reference time can induce the change of synaptic weights, resulting in a construction of causal connections.

Based on the similarity with the PSD rule, a multilayer tempotron (MutTmptr) rule can be developed as an extension of the single layer tempotron. The synaptic plasticity for MutTmptr is described as follow:

$$\Delta w_i = \begin{cases} \eta \sum_{t_i<t_{max}} K(t_{max} - t_i - d), & \text{if } P^+ \text{ error;} \\ -\eta \sum_{t_i<t_{max}} K(t_{max} - t_i - d), & \text{if } P^- \text{ error;} \\ 0, & \text{otherwise.} \end{cases} \tag{6.6}$$

where t_{max} denotes the time at which the neuron reaches its maximum potential value in the time domain, and d denotes the synaptic time delay. The above equation is equivalent to the follow:

$$\Delta w_i = \begin{cases} \eta \cdot I_{PSC}^i(t_{max} - t_i - d), & \text{if } P^+ \text{ error;} \\ -\eta \cdot I_{PSC}^i(t_{max} - t_i - d), & \text{if } P^- \text{ error;} \\ 0, & \text{otherwise.} \end{cases} \tag{6.7}$$

where I_{PSC}^i denotes the post-synaptic current (PSC) of the corresponding synapse.

The instructor signal, containing the modification direction, is back propagated to all synapses in the multilayer network. The amount of synaptic change depends on the PSC signal. Equation 6.5 and Equation 6.7 are used to conduct the following simulations.

6.3 Heuristic Discussion on the Multilayer Learning Rules

In this section, we use a simple three-layer network (see Figure 6.1) to analyze the process of the synaptic modification in our MutPSD and MutTmptr rules. For simplicity, neurons are connected through synapses without delays. Synaptic change between the input and the hidden neurons is denoted as Δw_{ih}. Δw_{ho} refers to the change between the hidden and the output neurons.

The following cases would occur along the learning.

1. The output neuron fires a spike at t_o in MutPSD or fires a spike to negative patterns in MutTmptr:

   The LTD process will occur. The depression is back propagated to all synapses, resulting in $\Delta w_{ih} < 0$ and $\Delta w_{ho} < 0$. The excitatory synapses will become less excitatory and the inhibitory synapses will become more inhibitory. This could eliminate the wrong spike of the output neuron. A decrease in w_{ho} could cause the decrease in the potential of the output neuron, thus the spike could be eliminated. The decrease in w_{ih} would result in a silent response of the hidden neuron. Without the stimulating signal (spikes) from the hidden neuron, the output neuron could become silent as desired.

2. The output neuron fails to fire a spike at t_d in MutPSD or keeps silent to a positive pattern in MutTmptr:

   The LTP process will occur. Similar to the depression process, the potentiation is back propagated to all synapses, resulting in $\Delta w_{ih} > 0$ and $\Delta w_{ho} > 0$. The excitatory synapses will become more excitatory and the inhibitory synapses will become less inhibitory. As a result, the potential of the output neuron could be increased, leading to a spike correspondingly.

3. The output neuron reacts correctly as desired:

   In the MutPSD rule, this means the output neuron only fires at the time of t_d. In the MutTmptr rule, it means the output neuron fires at least one spike to positive patterns and keeps silent to negative patterns. If the output neuron responds as desired, no synaptic modification occurs.

The instructor signal guides the direction of the synaptic modification, leading the output neuron to a desired response if such a solution exists.

6.4 Simulation Results

In this section, several simulation experiments are conducted to demonstrate the capabilities of the algorithm. Firstly, through the association of spatiotemporal spike pattern by the MutPSD rule, we demonstrate how the causal connections are constructed. Both the MutPSD and the MutTmptr rules are then applied to classic benchmarks, including the XOR problem and the Iris dataset.

6.4.1 Construction of Causal Connections

In order to demonstrate the construction of causal connections, the MutPSD rule is used to train the neuron to associate the input spatiotemporal spike pattern with a desired spike train.

Technical Details

We construct the network in the structure of $50 \times 100 \times 1$, without the synaptic delay. The input spatiotemporal spike pattern connects with the network through the input neurons. The spatiotemporal spike pattern is designed in a single-spike manner, where each input neuron only fires once within a time window of $30\,ms$. The output neuron is trained, within a max number of training epochs (150), to fire a desired spike train of [10, 20, 30] ms. The initial weights are uniformly distributed in the range of [0, 0.5]. We set $\eta = 0.01$ and $\tau_s = 7\,ms$. The learning is considered converged when each of the actual output spike approaches to the corresponding desired spike within a precision of $0.1\,ms$.

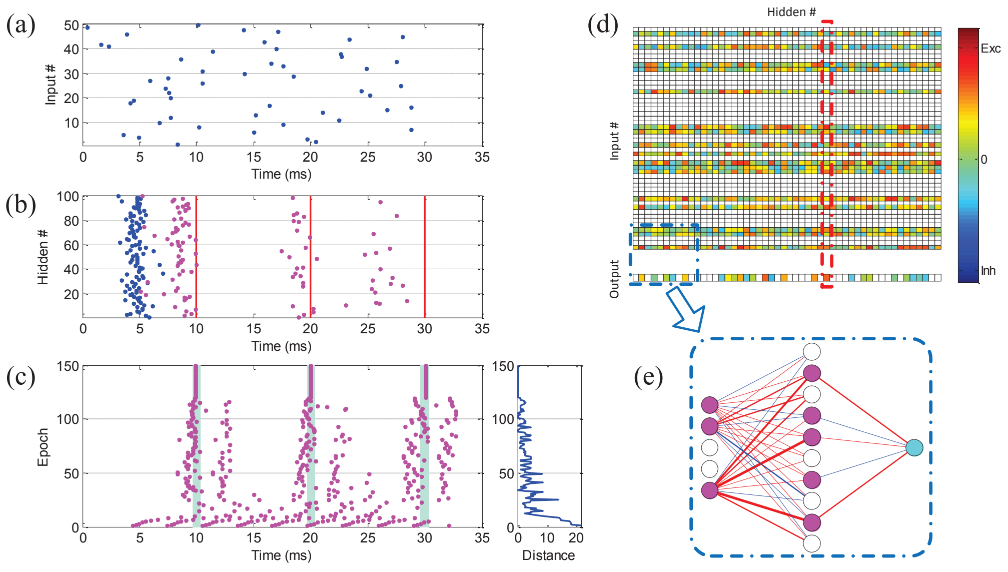

Figure 6.3: Construction of causal connections. The multilayer network is trained to output a desired spike train associating with the input pattern. (a) is the demonstration of the input spatiotemporal spike pattern. (b) shows the spikes of the hidden neurons before learning (blue) and after learning (magenta). Vertical red lines denote the target time. (c) shows the actual output spikes along the learning epochs. The spike distance between the desired and actual output spike trains is also shown. Shaded bars denote the desired spike time. (d) demonstrates the weight matrix of the network that relating to the target time of 10 ms. The intensity reflects the weight value, and white boxes mean the corresponding connections do not fire before this target time. (e) demonstrates a connection view of the corresponding part in (d). Shaded neurons mean that they fired before 10 ms, thus they are wired up to construct causal connections. The weight strength is denoted by the line width.

Analysis of the Learning

Figure 6.3(a) shows the input spatiotemporal spike pattern. The network is trained to associate this spike pattern with the desired spike train. As is shown in Figure 6.3(c), the output neuron gradually learns to fire at the desired times. At the begin, both the firing rate and the precise time of the output spikes are different from those of the desired spikes. Along the learning, the output neuron can successfully fire the desired spikes. This can also be reflected through the spike distance graph, where a small distance denotes a big similarity between the desired and the actual output spike trains.

Figure 6.3(b) shows the firing behavior of the hidden neurons. Before learning, the spikes of the hidden neurons are far away from the desired time, thus it is difficult for these hidden spikes causing desired output spikes. After learning, a sufficient number of hidden spikes appear before each desired spike. These hidden spikes are necessary for resulting in spikes at the desired times. We denote those pre-synaptic spikes that resulting in a post-synaptic spike as the causal spikes. Another necessary factor for causing desired spikes is that the synaptic weights corresponding to the causal spikes should be fine tuned.

Figure 6.3(b) demonstrates one necessary factor with respect to the causal spikes. The other necessary factor regarding to the weights are shown in Figure 6.3(d)(e). For simplicity, only the causal connections for firing a target spike at $10\ ms$ are shown. Figure 6.3(d) shows the weight matrix of the network. For example, the red rectangle reflects the weights relating to a specific hidden neuron, with upper figure showing connections from input neurons to this hidden neuron, and lower figure representing the weight from this hidden neuron to the output neuron. In the weight matrix figure, the white boxes mean the corresponding connections do not have causal spikes. Figure 6.3(e) shows the connection structure corresponding to the part in Figure 6.3(d). Neurons without causal spikes do not have effect on the desired spikes. As can be seen from the figure, causal neurons are connected with fine tuned weights, including both the excitatory and inhibitory synapses.

Table 6.1: XOR Problem Description for Multilayer SNNs

XOR input	Encoded Spike Input (ms)			MutPSD Output (ms)	MutTmptr Output
(0, 0)	0	0	0	16	Fire
(0, 1)	0	6	0	10	Silent
(1, 0)	6	0	0	10	Silent
(1, 1)	6	6	0	16	Fire

6.4.2 The XOR Benchmark

The XOR problem is a linearly nonseparable task, and it is a classic benchmark problem widely used for investigating the classification ability of spiking neural networks recently [34, 129–131]. Thus, we also use the XOR task to investigate the ability of our MutPSD and MutTmptr rules in this section. Detailed experimental setup and results are presented as follows.

Technical Details

Similar to [34], the input and output patterns for the XOR task are encoded into spikes (as can be seen in Table 6.1). The XOR input of 0/1 is directly converted to the spike input of 0/6 ms. In addition to these two inputs, a third neuron with an input spike at 0 ms is used to serve as the time reference. Without this time reference, pattern (0, 0) and (1, 1) would be identical in the view of spikes, thus the network would be unable to distinguish them.

We choose the network structure as $3 \times 5 \times 1$. Additionally, multiple sub connections (mSub) with different delays were used. We set mSub = 15, with delays ranging from 0 to 12 ms. The network was trained with $\eta = 0.01$ and $\tau_s = 7\ ms$, otherwise will stated. The network was simulated with a time window of 30 ms and a time step of 0.1 ms.

Demonstration of the Learning

The capabilities of both the MutPSD and the MutTmptr rules on the XOR task are demonstrated here. In the MutPSD rule, the output neuron is required to fire desired spikes with a precision of 0.2 ms corresponding to different input patterns. In the MutTmptr rule, instead of firing precisely, the output neuron is only required to correctly fire or not fire corresponding to an input pattern.

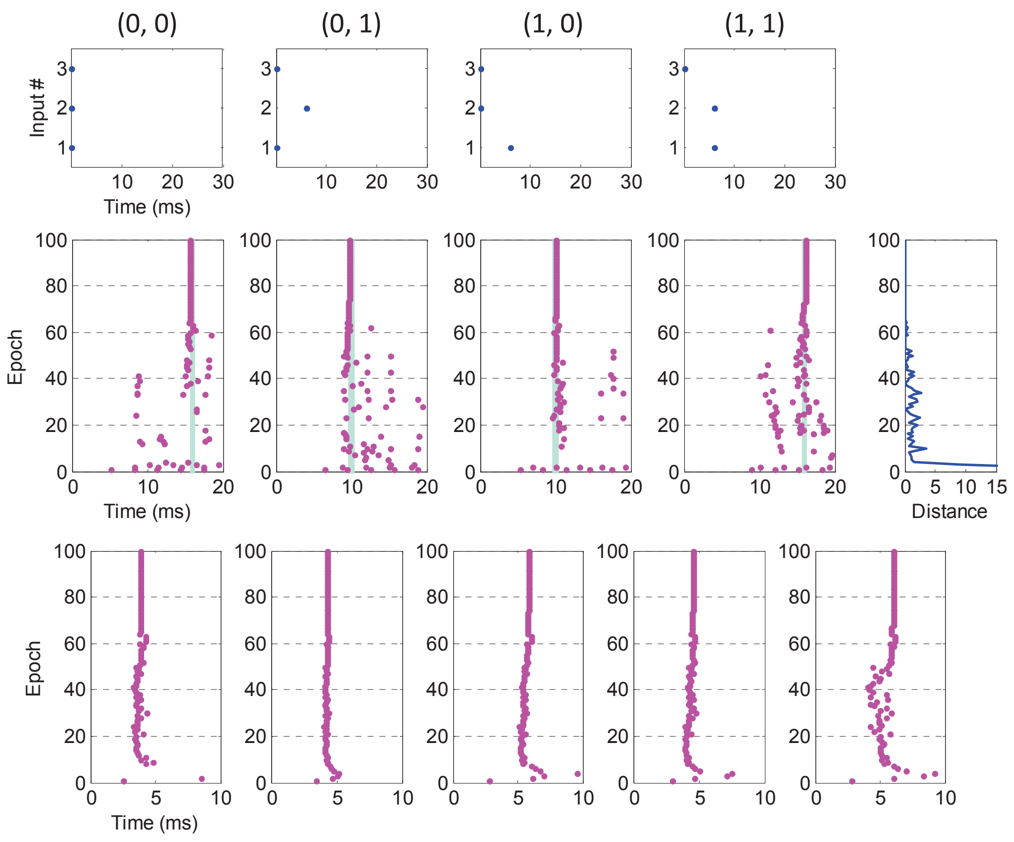

Figure 6.4: Demonstration of the MutPSD rule for the XOR task. The top row shows the four spike input patterns of the XOR task. The middle row shows the actual output spikes according to each input pattern. The shaded bars denote the desired spike time. The average spike distance is also shown on the right. The bottom row shows the output spikes of the hidden neurons for the input pattern of (0, 0).

Figure 6.4 demonstrates the MutPSD rule can successfully train the network to learn the XOR task. As is shown in Chapter 5, the single-layer PSD cannot directly learn this task, unless a reservoir network is used to enrich the dimension of the input space. In our MutPSD rule, a small number of hidden neurons with adjustable weights are sufficient for the XOR task. Along the learning, the output neuron gradually learns to fire desired spikes corresponding to different input patterns. The spike distance between the desired and the actual output spikes decreases gradually. The synaptic efficacies of the hidden neurons are also modified along the learning, which is reflected from the adjustment of their spike times. The adjusted spike time of the hidden neurons can facilitate the output neuron to fire desired spikes. These hidden spikes serve as the stimulating sources for the output neuron.

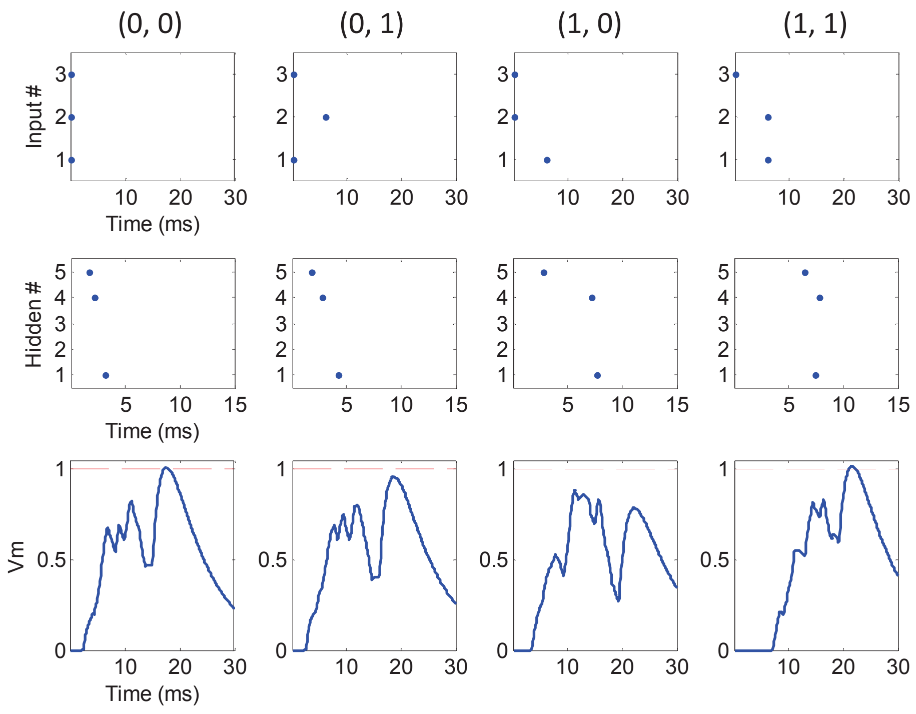

Figure 6.5: Demonstration of the MutTmptr rule for the XOR task. The top row shows the four spike input patterns of the XOR task. The middle row shows the actual output spikes of the hidden neurons according to each input pattern. The bottom row shows the membrane potentials of the output neuron for each corresponding input pattern.

As can be seen from Figure 6.5, the MutTmptr rule can also train the network to perform the XOR task well, with the output neuron firing a spike

Table 6.2: Convergent results for the XOR problem.

	Precision of convergence (ms)	No. of epochs for convergence	Successful rate (%)
Bohte [34]	0.71	250	-
McKennoch [137]	1.0	127	-
Sporea [130]	1.0	137	98
MutPSD	1.0	86	100
MutTmptr	-	37	100

for patterns of (0, 0) and (1, 1), and keeping silent for (0, 1) and (1, 0). The hidden neurons fire differently for each input pattern. Again, these spikes from the hidden neurons serve as the stimulating sources for the output neuron. Noteworthily, although 5 hidden neurons are chosen for the XOR task, only a small number of these hidden neurons (#1, #4, #5) are utilized. Therefore, our multilayer learning rule can effectively select a sufficient number of resources that are enough to fulfill the task.

Convergence of the Learning

In order to investigate the convergent performance of our multilayer learning rules, the previous demonstration experiment is conducted for 50 runs. For the MutPSD rule, a precision of 1 ms is used as in other studies [130, 137]. The average results are reported in Table 6.2.

Table 6.2 shows our multilayer learning rules are more efficient for the XOR task. To train the output neuron to spike precisely corresponding to different patterns, our MutPSD rule has a faster convergent speed compared to other rules. A higher successful rate of runs is also obtained compared to that in [130]. In addition, with less number of learning parameters, our MutPSD rule is simpler compared to multilayer ReSuMe rule in [130]. Regardless of firing precisely, the MutTmptr rule converges even faster. This is expected since only a response of fire or not fire is considered for the MutTmptr rule. Such a binary response can simplify the learning compared to those rules for precise response in time.

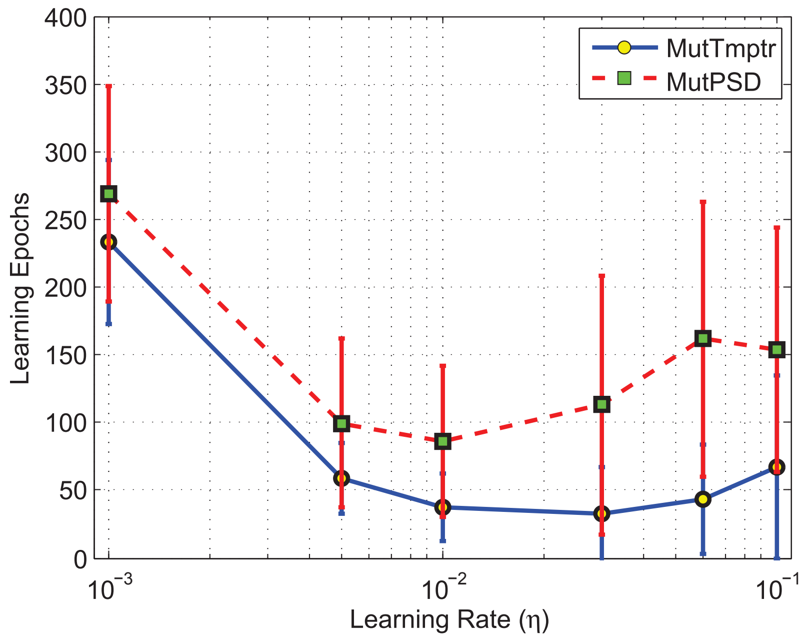

Figure 6.6: Effect of the learning rate on the convergence of the XOR task. Results are averaged over 50 runs.

Figure 6.6 shows the effect of the learning rate η on our multilayer learning rules. As can be seen in this figure, a smaller η results in a slower learning speed. The learning becomes faster with an increasing η. However, a further increase in η cannot benefit the learning. The successful rate of runs can be decreased with a larger η (results are not shown here). Additionally, the learning speed of the MutTmptr rule is always faster than that of the MutPSD rule. As discussed before, this is because the MutTmptr rule only needs to train the neuron to have a binary response of fire or not, regardless of the precise time of the response.

6.4.3 The Iris Benchmark

In oder to investigate the recognition performance of our multilayer rules, the classic Iris benchmark task is considered. The dataset consists of three classes of Iris flowers, with one class being linearly separable from the other two classes, and two classes being nonlinearly separable with each other. Each class contains 50 samples and each sample is represented by 4 variables.

Technical Details

To encode the variables of Iris, we use the same population encoding scheme as in [34, 75], where each feature is encoded separately by an array of Gaussian functions with different centers. For a variable x in a range $[x_{min}, x_{max}]$, n neurons with different Gaussian receptive fields are used to encode. The center and width of the i-th neuron are set to $\mu_i = x_{min} + (2 \cdot i - 3)/2 \cdot (x_{max} - x_{min})/(n - 2)$ and $\sigma_i = 1/1.5 \cdot (x_{max} - x_{min})/(n - 2)$, respectively. Each feature is encoded as n (set as 5) spike times between 0 and 10 ms. Thus, the total number of input neurons is $4 \times 5 + 1 = 21$. The number of hidden neurons is selected as 8. The number of sub connections is set to 5, and each synapse has a synaptic delay between 0 and 10 ms. Three networks of $21 \times 8 \times 1$ are constructed with each network for one class. The upper limit of training epochs is set to 300. For the MutPSD rule, each network is trained to fire a desired train of [15, 25] ms corresponding to the correct input class, and to keep silent for other classes. In the MutTmptr rule, each network is trained to fire a spike for the positive class, and to keep silent for other classes.

Analysis of the Learning

We use a winner-take-all scheme for the readout. For the MutPSD rule, the network with closest spike distance dominates the class of the input pattern. For the MutTmptr rule, two different winner-take-all readout schemes are investigated. One regards to the fire status (denoted as MutTmptr_Fire), and the other one regards to the maximum potential (denoted as MutTmptr_Vmax).

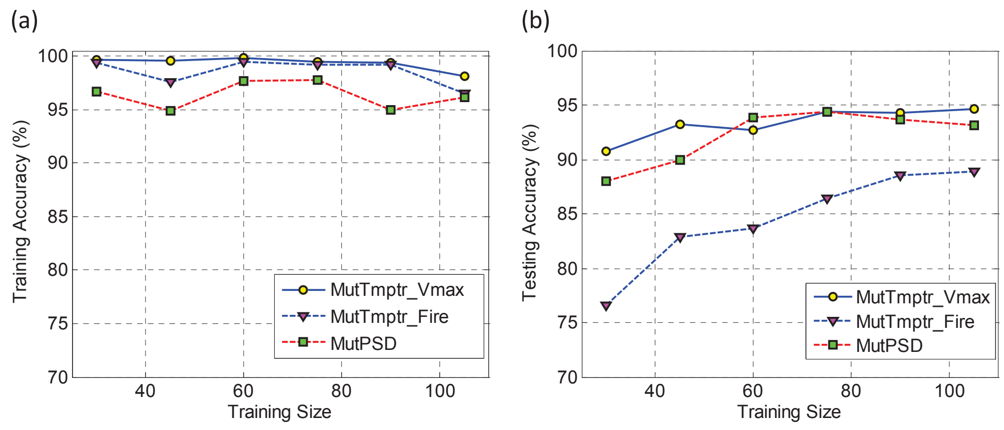

Figure 6.7: Performance of multilayer learning rules on the Iris task. (a) and (b) show the training and testing accuracy, respectively. Results are averaged over 10 runs.

As can be seen from Figure 6.7, the MutTmptr rule can learn the training set better than the MutPSD rule, while the MutPSD rule has a better generalization performance. It can be seen from Figure 6.7(b), the testing accuracy tends to increase with the increasing number of samples used for training. If only output spikes are considered for the readout, the MutPSD rule performs better than the MutTmptr_Fire rule. This is because the MutPSD rule makes decision based on a combination of several local temporal features, but the MutTmptr rule only uses single temporal feature for the decision. In addition, the MutTmptr rule requires all the three nets to response correctly for a correct decision. This is another factor affects its performance. If we use the maximum potential for the decision, the performance is improved significantly (see MutTmptr_Vmax in Figure 6.7(b)).

6.5 Discussion and Conclusion

In this chapter, we proposed two learning rules for multilayer SNNs, namely the multilayer PSD rule (MutPSD) and the multilayer tempotron rule (MutTmptr). These two rules are similar, where a supervisor signal, containing the synaptic modification direction, is back propagated to the synapses in the network. Without complex error evaluation as in [34, 75, 130], our multilayer learning rules are simpler and more efficient. In addition, it is not biologically plausible for the neurons to back propagated a calculated error, or it is at least questionable. A global neuromodulatory signal, determining the polarity of the synaptic changes, would be more feasible [33].

The amount of synaptic change depends on the pre-synaptic currents. This scheme, combined with the supervisor signal, can help to construct the causal connections between neurons. Correlated neurons are connected with fine tuned weights, resulting in a desired response at the output neuron. The hidden neurons serve as the information transmitter between the input and output neurons.

The MutTmptr rule has a faster convergent speed than the MutPSD rule. This is because the MutTmptr rule only trains the network to respond correctly with a binary status, either fire or not. For the MutPSD rule, the precise spike time of the output neuron is also considered. This makes the learning more difficult than the MutTmptr rule. However, the MutPSD rule has a better generalization ability compared to the MutTmptr rule. This is due to that, the MutPSD makes a decision based on a combination of several local temporal features, while the MutTmptr uses only a single local temporal feature for a decision.

In summary, both the MutPSD and the MutTmptr rules are simple, efficient and yet biologically plausible. We demonstrate the mechanisms that how the causal connections are constructed in the multilayer spiking neural networks. The performances of our multilayer learning rules are investigated through the two classic benchmark tasks, that is the XOR task and the Iris dataset problem. The MutTmptr rule can provide a faster learning speed, while the MutPSD rule gives a better generalization ability.

Chapter 7

Conclusions

In this chapter, the main results of this thesis are summarized, and some possible future directions are also provided.

7.1 Summary of Contributions

In Chapter 2, a consistent system considering both the temporal coding and learning was preliminarily developed to perform various recognition tasks. The whole system was constructed by three basic functional parts: encoding, learning and readout. It was found that such a network of spiking neurons under a temporal framework can effectively and efficiently perform various classification tasks. The results suggest that the temporal learning rule combined with a proper encoding method can provide basic classification abilities of spiking neurons on different classification tasks. This proposed system can learn patterns of either discrete values or continuous values through different encoding schemes. It is likely that an effective and efficient learning could be attributed to a suitable encoding where encoded spatiotemporal patterns from different categories are easily separable. This system is important since it integrates both the coding and the learning on a systematic level. The integrated system also provides a general structure that could be flexibly extended or modified according to various requirements, as long as the basic functional parts inspired from the biology do not change. This can significantly benefit future studies on a systematic level.

In Chapter 3, a more complex and biologically plausible system was developed from an extension of the previous system introduced in Chapter 2. Motivated by recent findings in biological systems, this system was constructed in a feedforward structure to process real-world stimuli from a view point of rapid computation. It was found that the external stimuli are sparsely represented after the encoding structure, and the representations have some properties of selectivity and invariance. These properties of the encoding structure could be attributed to simple cells and complex cells alternately used in HMAX which is a hierarchical system that closely follows the organization of visual cortex. The simple cells gain their selectivity from a linear sum operation, while the complex cells obtain invariance through a nonlinear max pooling operation. The properties of selectivity and invariance in the encoding structure can facilitate further procession in downstream neurons. It was also found that the tempotron rule is a proper choice among different temporal learning rules when the learning speed is mainly considered. Compared to the ReSuMe rule, the tempotron rule can accomplish a recognition task faster and more reliably. The results also showed that a robust response from the readout layer can be obtained through groups of neurons rather than single neurons. These results suggest that grouped pools of neurons are more reliable and biologically realistic than single neurons considering a decision-making task in the readout layer. A possible explanation is that the readout would be very sensitive to each neuron's response if single neurons are considered. The final decision would be wrong even if only one neuron misclassifies an input pattern. The significance of this study is that it would provide a possible direction of applying spiking neurons into real-world recognition tasks. It is also important in the light of recent trends in combining both the coding and learning on a systematic level to perform cognitive computations.

In Chapter 4, a new temporal learning rule, named as the precise-spike-driven (PSD) synaptic plasticity rule, was developed for learning hetero-association of spatiotemporal spike patterns. Various properties of the PSD rule were also investigated through an extensive experimental analysis. It was found that the PSD rule could successfully train neurons to associate a sparse spatiotemporal pattern with a desired spike train. This is due to the successful establishment of the causal connections along the learning. Sufficient synaptic strength is required for those afferent neurons firing around a desired spike time to stimulate this desired spike. It was also found that the PSD rule can also perform classification of spatiotemporal spike patterns. The PSD rule with the relative confidence criterion has a comparable performance to the tempotron rule. Moreover, the PSD rule is advantageous in that it is not limited to performing classification, but it is also able to memorize patterns by firing desired spikes at precise time. The contribution of this study is that a temporal learning rule is developed for spiking neurons from both the view points of simplicity and biological plausibility.

In Chapter 5, a spiking neural network system for sequence recognition was developed. The PSD rule was applied and further investigated for practical applications in this study. It was found that different functional subsystems can consistently cooperate within a temporal framework for detecting and recognizing a specific sequence. The results indicate that different spiking neural networks can be combined together as long as a proper coding scheme is used for the communications between networks. This study is significant since it provides a possible explanation of mechanisms that might be used in the brain for sequence recognition.

In Chapter 6, two temporal learning rules were proposed for multilayer spiking neural networks, namely the multilayer PSD rule (MutPSD) and the multilayer tempotron rule (MutTmptr). These two multilayer rules are extensions of the single layer PSD and tempotron rules. The multilayer learning is fulfilled through the construction of causal connections. Correlated neurons are connected through fine tuned weights. It was found that the MutTmptr rule converges faster, while the MutPSD rule gives better generalization ability. The fast convergent speed of the MutTmptr rule is due to the binary response of either fire or not. The good generalization ability of the MutPSD rule could be attributed to the combination of several local temporal features for a decision. The significance of this study is that it provides an efficient and biologically plausible mechanism, describing how synapses in the multilayer network are adjusted to facilitate the learning.

7.2 Future Work

It is still one of the greatest challenges facing science today to understand the brain due to the limitations of current technology. Instead of directly performing experiments on biological systems, this thesis was restricted to computer simulations to explore the cognitive abilities of spiking neurons. Due to the nature of this approach, the applicability would not be exactly suitable to the real brain. The modeling assumptions used in this study are based on the recent experimental findings, which may restrict the biological plausibility of the model to a certain degree. Future experimental findings from the neuroscience could further benefit our understandings about the brain. Further research is therefore needed to develop new models considering these new findings. The computers could then perform cognitive computations more like the brain, which could further benefit the area of artificial intelligence.

This thesis did not consider computations under a rate-based framework. This is because mounting evidence shows that precise timing of individual spikes plays an important role. The temporal framework also offers significant computational advantages over the rate-based framework. Since it is indisputable that the rate coding is also used in the functioning of the brain, it would be interesting and valuable to explore computations under a framework considering both the rate coding and the temporal coding.

Come to cognitive functions, the best man-made computer still cannot even give a comparable performance to the brain. Such cognitions of the brain actually rely on both the neuronal and the systematic levels. It would be

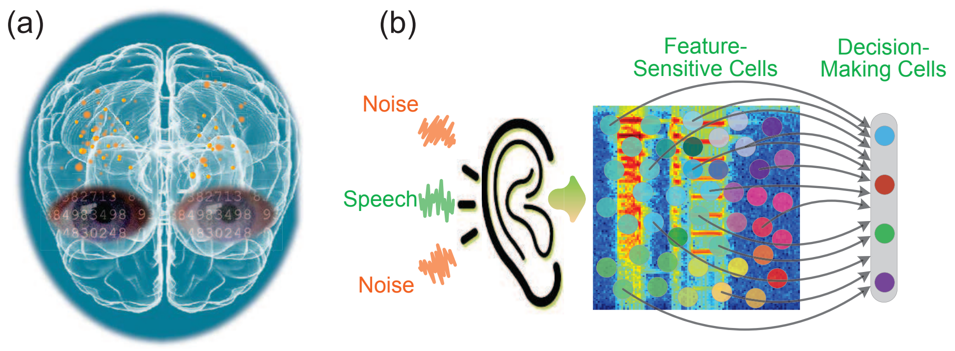

Figure 7.1: Sensory systems for cognitions. (a) and (b) demonstrate a visual and auditory system, respectively.

interesting and valuable to further investigate how spiking features of a neuron could enrich the computational power, and how system with layers of spiking neurons could process information for cognitive functions. Sensory systems share a similar general system structure with functional parts of encoding, learning and decoding. Such a structure preliminarily describes the building blocks required for an intelligent system. One of the long-range goals is to develop an intelligent cognitive system with spiking neurons, and to utilize it on practical tasks such as visual or auditory processing (see Figure 7.1). To accomplish this, cooperation between both computational and experimental approaches would be required.

Bibliography

[1] E. Adrian, *The Basis of Sensation: The Action of the Sense Organs*. W. W. Norton, New York, 1928.

[2] R. VanRullen, R. Guyonneau, and S. J. Thorpe, "Spike times make sense," *Trends in Neurosciences*, vol. 28, no. 1, pp. 1–4, 2005.

[3] E. R. Kandel, J. H. Schwartz, T. M. Jessell, *et al.*, *Principles of neural science*, vol. 4. McGraw-Hill New York, 2000.

[4] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 1 ed., 2002.

[5] J. Vreeken, "Spiking neural networks, an introduction," *Institute for Information and Computing Sciences, Utrecht University Technical Report UU-CS-2003-008*, 2002.

[6] W. Maass, G. Schnitger, and E. D. Sontag, "On the computational power of sigmoid versus boolean threshold circuits," in *Foundations of Computer Science, 1991. Proceedings., 32nd Annual Symposium on*, pp. 767–776, IEEE, 1991.

[7] J. J. Hopfield and C. D. Brody, "What is a moment? Transient synchrony as a collective mechanism for spatiotemporal integration," *Proceedings of the National Academy of Sciences*, vol. 98, no. 3, pp. 1282–1287, 2001.

[8] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve.," *The Journal of physiology*, vol. 117, no. 4, pp. 500–544, 1952.

[9] E. M. Izhikevich, "Simple model of spiking neurons.," *IEEE Transactions on Neural Networks*, vol. 14, no. 6, pp. 1569–1572, 2003.

[10] W. Gerstner, A. K. Kreiter, H. Markram, and A. V. Herz, "Neural codes: firing rates and beyond," *Proceedings of the National Academy of Sciences*, vol. 94, no. 24, pp. 12740–12741, 1997.

[11] S. Panzeri, N. Brunel, N. K. Logothetis, and C. Kayser, "Sensory neural codes using multiplexed temporal scales," *Trends in Neurosciences*, vol. 33, no. 3, pp. 111–120, 2010.

[12] P. Dayan and L. Abbott, “Theoretical neuroscience: computational and mathematical modeling of neural systems,” *Journal of Cognitive Neuroscience*, vol. 15, no. 1, pp. 154–155, 2003.

[13] D. A. Butts, C. Weng, J. Jin, C.-I. Yeh, N. A. Lesica, J.-M. Alonso, and G. B. Stanley, “Temporal precision in the neural code and the timescales of natural vision,” *Nature*, vol. 449, no. 7158, pp. 92–95, 2007.

[14] W. Bair and C. Koch, “Temporal precision of spike trains in extrastriate cortex of the behaving macaque monkey.,” *Neural Computation*, vol. 8, no. 6, pp. 1185–1202, 1996.

[15] M. J. Berry and M. Meister, “Refractoriness and neural precision.,” *The Journal of Neuroscience*, vol. 18, no. 6, pp. 2200–2211, 1998.

[16] V. J. Uzzell and E. J. Chichilnisky, “Precision of spike trains in primate retinal ganglion cells.,” *Journal of Neurophysiology*, vol. 92, no. 2, pp. 780–789, 2004.

[17] P. Reinagel and R. C. Reid, “Temporal coding of visual information in the thalamus,” *The Journal of Neuroscience*, vol. 20, no. 14, pp. 5392–5400, 2000.

[18] Z. F. Mainen and T. J. Sejnowski, “Reliability of spike timing in neocortical neurons,” *Science*, vol. 268, no. 5216, pp. 1503–1506, 1995.

[19] F. Gabbiani, W. Metzner, R. Wessel, and C. Koch, “From stimulus encoding to feature extraction in weakly electric fish,” *Nature*, vol. 384, no. 6609, pp. 564–567, 1996.

[20] M. Wehr and A. M. Zador, “Balanced inhibition underlies tuning and sharpens spike timing in auditory cortex,” *Nature*, vol. 426, no. 6965, pp. 442–446, 2003.

[21] W. Maass and C. M. Bishop, *Pulsed neural networks.* MIT press, 2001.

[22] T. Serre, A. Oliva, and T. Poggio, “A feedforward architecture accounts for rapid categorization,” *Proceedings of the National Academy of Sciences*, vol. 104, no. 15, pp. 6424–6429, 2007.

[23] T. Gollisch and M. Meister, “Rapid neural coding in the retina with relative spike latencies.,” *Science*, vol. 319, no. 5866, pp. 1108–1111, 2008.

[24] Z. Nadasdy, “Information encoding and reconstruction from the phase of action potentials,” *Frontiers in Systems Neuroscience*, vol. 3, p. 6, 2009.

[25] R. Kempter, W. Gerstner, and J. L. van Hemmen, “Spike-Based Compared to Rate-Based Hebbian Learning.,” in *NIPS’98*, pp. 125–131, 1998.

[26] A. Borst and F. E. Theunissen, “Information theory and neural coding.,” *Nature Neuroscience*, vol. 2, no. 11, pp. 947–957, 1999.

[27] J. J. Hopfield, “Pattern recognition computation using action potential timing for stimulus representation,” *Nature*, vol. 376, no. 6535, pp. 33–36, 1995.

[28] D. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. Taylor & Francis Group, 2002.

[29] S. Fusi, “Spike-driven synaptic plasticity for learning correlated patterns of mean firing rates,” *Reviews in the Neurosciences*, vol. 14, no. 1-2, pp. 73–84, 2003.

[30] 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.

[31] G. Q. Bi and M. M. Poo, “Synaptic modification by correlated activity: Hebb’s postulate revisited,” *Annual Review of Neuroscience*, vol. 24, pp. 139–166, 2001.

[32] R. C. Froemke, M.-m. Poo, and Y. Dan, “Spike-timing-dependent synaptic plasticity depends on dendritic location,” *Nature*, vol. 434, no. 7030, pp. 221–225, 2005.

[33] 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.

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

[35] 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.

[36] R. V. Florian, “The Chronotron: A Neuron that Learns to Fire Temporally Precise Spike Patterns,” *PLoS One*, vol. 7, no. 8, p. e40233, 2012.

[37] F. Ponulak, “ReSuMe–new supervised learning method for spiking neural networks,” tech. rep., Institute of Control and Information Engineering, Poznoń University of Technology, 2005.

[38] R. Guyonneau, R. van Rullen, and S. J. Thorpe, “Neurons Tune to the Earliest Spikes Through STDP,” *Neural Computation*, vol. 17, no. 4, pp. 859–879, 2005.

[39] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Spike timing dependent plasticity finds the start of repeating patterns in continuous spike trains," *PloS One*, vol. 3, no. 1, p. e1377, 2008.

[40] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Competitive stdp-based spike pattern learning," *Neural Computation*, vol. 21, no. 5, pp. 1259–1276, 2009.

[41] O. Booij *et al.*, "A gradient descent rule for spiking neurons emitting multiple spikes," *Information Processing Letters*, vol. 95, no. 6, pp. 552–558, 2005.

[42] J. D. Victor and K. P. Purpura, "Metric-space analysis of spike trains: theory, algorithms and application," *Network: Computation in Neural Systems*, vol. 8, no. 2, pp. 127–164, 1997.

[43] M. C. Van Rossum, G. Bi, and G. Turrigiano, "Stable Hebbian learning from spike timing-dependent plasticity," *The Journal of Neuroscience*, vol. 20, no. 23, pp. 8812–8821, 2000.

[44] J. Dennis, Q. Yu, H. Tang, H. D. Tran, and H. Li, "Temporal coding of local spectrogram features for robust sound recognition," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 803–807, 2013.

[45] Q. Yu, H. Tang, K. C. Tan, and H. Li, "Rapid feedforward computation by temporal encoding and learning with spiking neurons," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 10, pp. 1539–1552, 2013.

[46] J. J. Wade, L. J. McDaid, J. A. Santos, and H. M. Sayers, "SWAT: A Spiking Neural Network Training Algorithm for Classification Problems," *IEEE Transactions on Neural Networks*, vol. 21, no. 11, pp. 1817–1830, 2010.

[47] F. Ponulak and A. J. Kasinski, "Supervised Learning in Spiking Neural Networks with ReSuMe: Sequence Learning, Classification, and Spike Shifting.," *Neural Computation*, vol. 22, no. 2, pp. 467–510, 2010.

[48] H. Adeli and S. L. Hung, *Machine learning – Neural networks, genetic algorithms, and fuzzy sets.* NY: John Wiley and Sons, 1995.

[49] W. Maass, "Lower bounds for the computational power of networks of spiking neurons," *Neural Computation*, vol. 8, no. 1, pp. 1–40, 1996.

[50] S. Ghosh-Dastidar and H. Adeli, "Improved spiking neural networks for EEG classification and epilepsy and seizure detection," *Integrated Computer-Aided Engineering*, vol. 14, no. 3, pp. 187–212, 2007.

[51] S. G. Wysoski, L. Benuskova, and N. Kasabov, "Fast and adaptive network of spiking neurons for multi-view visual pattern recognition," *Neurocomputing*, vol. 71, no. 13, pp. 2563–2575, 2008.

[52] P. X. Tsukada M., "The spatiotemporal learning rule and its efficiency in separating spatiotemporal patterns," *Biological Cybernetics*, vol. 92, pp. 139–146, 2005.

[53] S. M. Bohte, E. M. Bohte, H. L. Poutr, and J. N. Kok, "Unsupervised Clustering with Spiking Neurons by Sparse Temporal Coding and Multi-Layer RBF Networks," *IEEE Transactions on Neural Networks*, vol. 13, pp. 426–435, 2002.

[54] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," tech. rep., DTIC Document, 1985.

[55] R. Kempter, W. Gerstner, and J. L. van Hemmen, "Hebbian learning and spiking neurons," *Physical Review E*, vol. 59, no. 4, pp. 4498–4514, 1999.

[56] S. Song, K. D. Miller, and L. F. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature Neuroscience*, vol. 3, pp. 919–926, 2000.

[57] R. Legenstein, C. Naeger, and W. Maass, "What Can a Neuron Learn with Spike-Timing-Dependent Plasticity?," *Neural Computation*, vol. 17, pp. 2337–2382, 2005.

[58] C. Johnson and G. K. Venayagamoorthy, "Encoding real values into polychronous spiking networks," in *IJCNN*, pp. 1–7, 2010.

[59] S. Mitra, S. Fusi, and G. Indiveri, "Real-Time Classification of Complex Patterns Using Spike-Based Learning in Neuromorphic VLSI," vol. 3, no. 1, pp. 32–42, 2008.

[60] H. Tang, H. Li, and R. Yan, "Memory dynamics in attractor networks with saliency weights," *Neural Computation*, vol. 22, no. 7, pp. 1899–1926, 2010.

[61] E. Gardner, "The space of interactions in neural networks models," *Journal of Physics*, vol. A21, pp. 257–270, 1988.

[62] R. C. Froemke and Y. Dan, "Spike-timing-dependent synaptic modification induced by natural spike trains," *Nature*, vol. 416, no. 6879, pp. 433–438, 2002.

[63] E. I. Knudsen, "Supervised learning in the brain," *Journal of Neuroscience*, vol. 14, no. 7, pp. 3985–3997, 1994.

[64] W. T. Thach, "On the specific role of the cerebellum in motor learning and cognition: clues from PET activation and lesion studies in man," *Behavioral and Brain Sciences*, vol. 19, no. 3, pp. 411–431, 1996.

[65] M. Ito, "Mechanisms of motor learning in the cerebellum," *Brain Research*, vol. 886, no. 1-2, pp. 237–245, 2000.

[66] M. R. Carey, J. F. Medina, and S. G. Lisberger, "Instructive signals for motor learning from visual cortical area MT," *Nature Neuroscience*, vol. 8, no. 6, pp. 813–819, 2005.

[67] R. C. Foehring and N. M. Lorenzon, "Neuromodulation, development and synaptic plasticity.," *Canadian Journal of Experimental Psychology/Revue canadienne de psychologie expérimentale*, vol. 53, no. 1, pp. 45–61, 1999.

[68] J. K. Seamans, C. R. Yang, *et al.*, "The principal features and mechanisms of dopamine modulation in the prefrontal cortex.," *Progress in Neurobiology*, vol. 74, no. 1, pp. 1–57, 2004.

[69] M. Randic, M. Jiang, and R. Cerne, "Long-term potentiation and long-term depression of primary afferent neurotransmission in the rat spinal cord," *The Journal of Neuroscience*, vol. 13, no. 12, pp. 5228–5241, 1993.

[70] C. Hansel, A. Artola, and W. Singer, "Relation Between Dendritic Ca2+ Levels and the Polarity of Synaptic Long-term Modifications in Rat Visual Cortex Neurons," *European Journal of Neuroscience*, vol. 9, no. 11, pp. 2309–2322, 2006.

[71] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the National Academy of Sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[72] A. Treves and E. T. Rolls, "What determines the capacity of autoassociative memories in the brain?," *Network: Computation in Neural Systems*, vol. 2, no. 4, pp. 371–397, 1991.

[73] A. Treves, "Graded-response neurons and information encoding in autoassociative memory," *Physical Review A*, vol. 42, no. 4, pp. 2418–2430, 1990.

[74] C. W. Eurich and S. D. Wilke, "Multi-Dimensional Encoding Strategy of Spiking Neurons," *Neural Computation*, vol. 12, pp. 1519–1529, 2000.

[75] Y. Xu, X. Zeng, L. Han, and J. Yang, "A supervised multi-spike learning algorithm based on gradient descent for spiking neural networks," *Neural Networks*, vol. 43, pp. 99–113, 2013.

[76] A. Delorme, J. Gautrais, R. van Rullen, and S. Thorpe, "SpikeNET: A Simulator For Modeling Large Networks of Integrate and Fire Neurons," *Neurocomputing*, vol. 24, pp. 26–27, 1999.

[77] K. C. Tan, E. J. Teoh, Q. Yu, and K. C. Goh, "A hybrid evolutionary algorithm for attribute selection in data mining," *Expert Systems with Applications*, vol. 36, no. 4, pp. 8616–8630, 2009.

[78] M. Fallahnezhad, M. H. Moradi, and S. Zaferanlouei, "A hybrid higher order neural classifier for handling classification problems," *Expert Systems with Applications*, vol. 38, no. 1, pp. 386–393, 2011.

[79] Q. Yu, H. Tang, K. C. Tan, and H. Yu, “A brain-inspired spiking neural network model with temporal encoding and learning,” *Neurocomputing*, vol. 138, pp. 3–13, 2014.

[80] D. I. Perrett, J. K. Hietanen, M. W. Oram, and P. J. Benson, “Organization and functions of cells responsive to faces in the temporal cortex,” *Philosophical Transactions of the Royal Society of London, Series B*, vol. 335, pp. 23–30, 1992.

[81] C. P. Hung, G. Kreiman, T. Poggio, and J. J. DiCarlo, “Fast readout of object identity from macaque inferior temporal cortex,” *Science*, vol. 310, no. 5749, pp. 863–866, 2005.

[82] R. V. Florian, “Tempotron-Like Learning with ReSuMe,” in *Proceedings of the 18th international conference on Artificial Neural Networks, Part II*, ICANN ’08, (Berlin, Heidelberg), pp. 368–375, Springer-Verlag, 2008.

[83] S. Thorpe, D. Fize, and C. Marlot, “Speed of processing in the human visual system,” *Nature*, vol. 381, no. 6582, pp. 520–522, 1996.

[84] R. Van Rullen and S. J. Thorpe, “Rate coding versus temporal order coding: what the retinal ganglion cells tell the visual cortex.,” *Neural Computation*, vol. 13, no. 6, pp. 1255–1283, 2001.

[85] L. Perrinet, M. Samuelides, and S. J. Thorpe, “Coding static natural images using spiking event times: do neurons cooperate?,” *IEEE Transactions on Neural Networks*, vol. 15, no. 5, pp. 1164–1175, 2004.

[86] J. Ranhel, “Neural Assembly Computing,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 6, pp. 916–927, 2012.

[87] D. H. Hubel and T. N. Wiesel, “Receptive fields and functional architecture of monkey striate cortex.,” *The Journal of physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[88] Burkart and Fischer, “Overlap of receptive field centers and representation of the visual field in the cat’s optic tract,” *Vision Research*, vol. 13, no. 11, pp. 2113 – 2120, 1973.

[89] Riesenhuber and T. Poggio, “Hierarchical models of object recognition in cortex,” *Nature Neuroscience*, vol. 2, no. 11, pp. 1019–1025, 1999.

[90] T. Masquelier and S. J. Thorpe, “Unsupervised Learning of Visual Features through Spike Timing Dependent Plasticity,” *PLoS Computational Biology*, vol. 3, no. 2, 2007.

[91] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, “Robust object recognition with cortex-like mechanisms,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 411–426, 2007.

[92] T. Serre, M. Kouh, C. Cadieu, U. Knoblich, G. Kreiman, and T. Poggio, “A theory of object recognition: Computations and circuits in the feedforward path of the ventral stream in primate visual cortex,” in *AI Memo*, 2005.

[93] C. Enroth-Cugell and J. G. Robson, “The contrast sensitivity of retinal ganglion cells of the cat.,” *The Journal of Physiology*, vol. 187, no. 3, pp. 517–552, 1966.

[94] M. J. McMahon, O. S. Packer, and D. M. Dacey, “The classical receptive field surround of primate parasol ganglion cells is mediated primarily by a non-GABAergic pathway.,” *The Journal of Neuroscience*, vol. 24, no. 15, pp. 3736–3745, 2004.

[95] A. J. Yu, M. A. Giese, and T. Poggio, “Biophysiologically Plausible Implementations of the Maximum Operation,” *Neural Computation*, vol. 14, no. 12, pp. 2857–2881, 2002.

[96] I. Lampl, D. Ferster, T. Poggio, and M. Riesenhuber, “Intracellular measurements of spatial integration and the MAX operation in complex cells of the cat primary visual cortex,” *Journal of Neurophysiology*, vol. 92, no. 5, pp. 2704–2713, 2004.

[97] T. J. Gawne and J. M. Martin, “Responses of primate visual cortical neurons to stimuli presented by flash, saccade, blink, and external darkening,” *Journal of Neurophysiology*, vol. 88, no. 5, pp. 2178–2186, 2002.

[98] F. Ponulak, “Analysis of the resume learning process for spiking neural networks,” *Applied Mathematics and Computer Science*, vol. 18, no. 2, pp. 117–127, 2008.

[99] B. A. Olshausen and D. J. Field, “Sparse coding with an overcomplete basis set: a strategy employed by V1?,” *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[100] J. Gautrais and S. Thorpe, “Rate coding versus temporal order coding: a theoretical approach,” *Biosystems*, vol. 48, no. 1-3, pp. 57 – 65, 1998.

[101] D. S. Reich, F. Mechler, and J. D. Victor, “Independent and Redundant Information in Nearby Cortical Neurons,” *Science*, vol. 294, pp. 2566–2568, 2001.

[102] M. Greschner, A. Thiel, J. Kretzberg, and J. Ammermüller, “Complex spike-event pattern of transient ON-OFF retinal ganglion cells.,” *Journal of Neurophysiology*, vol. 96, no. 6, pp. 2845–2856, 2006.

[103] J. J. Hunt, M. R. Ibbotson, and G. J. Goodhill, “Sparse Coding on the Spot: Spontaneous Retinal Waves Suffice for Orientation Selectivity,” *Neural Computation*, vol. 24, no. 9, pp. 2422–2433, 2012.

[104] W. Usrey and R. Reid, “Synchronous activity in the visual system,” *Annual Review of Physiology*, vol. 61, no. 1, pp. 435–456, 1999.

[105] M. Wilson and B. McNaughton, “Dynamics of the hippocampal ensemble code for space,” *Science*, vol. 261, no. 5124, pp. 1055–1058, 1993.

[106] A. Pouget, T. Dyan, and R. Zemel, “Information processing with population codes,” *Nature Reviews Neuroscience*, vol. 1, pp. 125–132, 2000.

[107] S. Ghosh-Dastidar and H. Adeli, “Spiking neural networks,” *International Journal of Neural Systems*, vol. 19, no. 04, pp. 295–308, 2009.

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

[109] M. N. Shadlen and J. A. Movshon, “Synchrony unbound: review a critical evaluation of the temporal binding hypothesis,” *Neuron*, vol. 24, pp. 67–77, 1999.

[110] B. Widrow and M. Lehr, “30 years of adaptive neural networks: Perceptron, madaline, and backpropagation,” *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[111] M. Rossum, “A novel spike distance,” *Neural Computation*, vol. 13, no. 4, pp. 751–763, 2001.

[112] F. Rieke, D. Warland, Rob, and W. Bialek, *Spikes: Exploring the Neural Code*. Cambridge, MA: MIT Press, 1st ed., 1997.

[113] J. Hu, H. Tang, K. C. Tan, H. Li, and L. Shi, “A spike-timing-based integrated model for pattern recognition,” *Neural Computation*, vol. 25, no. 2, pp. 450–472, 2013.

[114] A. Artola, S. Bröcher, and W. Singer, “Different voltage-dependent thresholds for inducing long-term depressiona and long-term potentiation in slices of rat visual cortex,” *Nature*, vol. 347, pp. 69–72, 1990.

[115] A. Ngezahayo, M. Schachner, and A. Artola, “Synaptic activity modulates the induction of bidirectional synaptic changes in adult mouse hippocampus,” *The Journal of Neuroscience*, vol. 20, no. 7, pp. 2451–2458, 2000.

[116] J. Lisman and N. Spruston, “Postsynaptic depolarization requirements for LTP and LTD: a critique of spike timing-dependent plasticity,” *Nature Neuroscience*, vol. 8, no. 7, pp. 839–841, 2005.

[117] J. A. Starzyk and H. He, “Spatio-temporal memories for machine learning: A long-term memory organization.,” *IEEE Transactions on Neural Networks*, vol. 20, no. 5, pp. 768–780, 2009.

[118] V. A. Nguyen, J. A. Starzyk, W.-B. Goh, and D. Jachyra, "Neural network structure for spatio-temporal long-term memory.," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 6, pp. 971–983, 2012.

[119] Q. Yu, H. Tang, K. C. Tan, and H. Li, "Precise-spike-driven synaptic plasticity: Learning hetero-association of spatiotemporal spike patterns," *PLoS One*, vol. 8, no. 11, p. e78318, 2013.

[120] D. Z. Jin, "Spiking neural network for recognizing spatiotemporal sequences of spikes," *Physical Review E*, vol. 69, no. 2, p. 021905, 2004.

[121] D. Z. Jin, "Decoding spatiotemporal spike sequences via the finite state automata dynamics of spiking neural networks," *New Journal of Physics*, vol. 10, no. 1, p. 015010, 2008.

[122] S. Byrnes, A. N. Burkitt, D. B. Grayden, and H. Meffin, "Learning a sparse code for temporal sequences using STDP and sequence compression," *Neural Computation*, vol. 23, no. 10, pp. 2567–2598, 2011.

[123] R. R. Llinas, A. A. Grace, and Y. Yarom, "In vitro neurons in mammalian cortical layer 4 exhibit intrinsic oscillatory activity in the 10-to 50-Hz frequency range," *Proceedings of the National Academy of Sciences*, vol. 88, no. 3, pp. 897–901, 1991.

[124] J. Jacobs, M. J. Kahana, A. D. Ekstrom, and I. Fried, "Brain oscillations control timing of single-neuron activity in humans," *The Journal of Neuroscience*, vol. 27, no. 14, pp. 3839–3844, 2007.

[125] K. Koepsell, X. Wang, V. Vaingankar, Y. Wei, Q. Wang, D. L. Rathbun, W. M. Usrey, J. A. Hirsch, and F. T. Sommer, "Retinal oscillations carry visual information to cortex," *Frontiers in Systems Neuroscience*, vol. 3, p. 4, 2009.

[126] C. Kayser, M. A. Montemurro, N. K. Logothetis, and S. Panzeri, "Spike-phase coding boosts and stabilizes information carried by spatial and temporal spike patterns," *Neuron*, vol. 61, no. 4, pp. 597–608, 2009.

[127] N. Schoppa and G. Westbrook, "Regulation of synaptic timing in the olfactory bulb by an A-type potassium current," *Nature Neuroscience*, vol. 2, no. 12, pp. 1106–1113, 1999.

[128] O. Shriki, D. Hansel, and H. Sompolinsky, "Rate models for conductance based cortical neuronal networks," *Neural Computation*, vol. 15, no. 8, pp. 1809–1841, 2003.

[129] 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.

[130] I. Sporea and A. Grüning, "Supervised learning in multilayer spiking neural networks," *Neural Computation*, vol. 25, no. 2, pp. 473–509, 2013.

[131] Y. Xu, X. Zeng, and S. Zhong, "A new supervised learning algorithm for spiking neurons," *Neural Computation*, vol. 25, no. 6, pp. 1472–1511, 2013.

[132] 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.

[133] B. L. Lewis and P. O'Donnell, "Ventral tegmental area afferents to the prefrontal cortex maintain membrane potential 'up'states in pyramidal neurons via D1 dopamine receptors," *Cerebral Cortex*, vol. 10, no. 12, pp. 1168–1175, 2000.

[134] J. Anderson, I. Lampl, I. Reichova, M. Carandini, and D. Ferster, "Stimulus dependence of two-state fluctuations of membrane potential in cat visual cortex," *Nature Neuroscience*, vol. 3, no. 6, pp. 617–621, 2000.

[135] R. Gütig and H. Sompolinsky, "Time-warp–invariant neuronal processing," *PLoS Biology*, vol. 7, no. 7, p. e1000141, 2009.

[136] R. S. Johansson and I. Birznieks, "First spikes in ensembles of human tactile afferents code complex spatial fingertip events," *Nature Neuroscience*, vol. 7, no. 2, pp. 170–177, 2004.

[137] S. McKennoch, D. Liu, and L. G. Bushnell, "Fast modifications of the spikeprop algorithm," in *Neural Networks, 2006. IJCNN'06. International Joint Conference on*, pp. 3970–3977, IEEE, 2006.

Author's Publications

The publications that were published, accepted, and submitted during the course of the author are listed as follows.

Journals

1. **Q. Yu**, H. Tang, K. C. Tan, and H. Li, "Temporal Learning in Multilayer Spiking Neural Networks Through Construction of Causal Connections", to be submitted, 2014.

2. **Q. Yu**, R. Yan, H. Tang, K. C. Tan, and H. Li, "A Spiking Neural Network System for Robust Sequence Recognition", *IEEE Transactions on Neural Networks and Learning Systems*, resubmitted after revision, 2014.

3. **Q. Yu**, H. Tang, K. C. Tan, and H. Yu, "A Brain-inspired Spiking Neural Network Model with Temporal Encoding and Learning", *Neurocomputing*, vol. 138, pp. 3-13, 2014.

4. **Q. Yu**, H. Tang, K. C. Tan, and H. Li, "Precise-Spike-Driven Synaptic Plasticity: Learning Hetero-Association of Spatiotemporal Spike Patterns", *PLoS One*, vol. 8, no. 11, pp. e78318, 2013.

5. **Q. Yu**, H. Tang, K. C. Tan, and H. Li, "Rapid Feedforward Computation by Temporal Encoding and Learning with Spiking Neurons", *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 10, pp. 1539-1552, 2013.
   PS: This paper was selected as a featured article of TNNLS, and introduced in *IEEE Computational Intelligence Society*. It was also selected as one of research highlights, and highlighted in A∗STAR *Research*.

Conferences

1. **Q. Yu**, S. K. Goh, H. Tang, and K. C. Tan, "Application of Precise-Spike-Driven Rule in Spiking Neural Networks for Optical Character Recognition", *The 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES'2014)*, accepted, Nov. 10-12, 2014, Singapore.

2. B. Zhao, **Q. Yu**, R. Ding, S. Chen, and H. Tang, "Event-Driven Simulation of the Tempotron Spiking Neuron", *in IEEE Biomedical Circuits and Systems Conference (BioCAS)*, in press, Oct 22-24, 2014, Lausanne, Switzerland.

3. **Q. Yu**, H. Tang, and K. C. Tan, "A New Learning Rule for Classification of Spatiotemporal Spike Patterns", *in IEEE International Joint Conference on Neural Networks (IJCNN)*, in press, Jul 6-11, 2014, Beijing, China.

4. B. Zhao, **Q. Yu**, H. Yu, S. Chen, and H. Tang, "A Bio-inspired Feedforward System for Categorization of AER Motion Events", *in IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 9-12, Oct 31-Nov 2, 2013, Rotterdam, Netherlands.

5. J. Dennis, **Q. Yu**, H. Tang, H. D. Tran, and H. Li, "Temporal Coding of Local Spectrogram Features for Robust Sound Recognition", *in IEEE Acoustics, Speech and Signal Processing (ICASSP)*, pp. 803-807, May 26-31, 2013, Vancouver, Canada.
   PS: This paper was highlighted in *ScienceDaily Report* as "Audio Processing Computers Following the Brains Lead", Nov 6, 2013.

6. **Q. Yu**, K. C. Tan, and H. Tang, "Pattern Recognition Computation in A Spiking Neural Network with Temporal Encoding and Learning", *in IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 466-472, Jun 10-15, 2012, Brisbane, Australia.

7. H. Tang, **Q. Yu**, and K. C. Tan, "Learning Real-World Stimuli by Single-Spike Coding and Tempotron Rule", *in IEEE International Joint Conference on Neural Networks (IJCNN)*, pp. 466-472, Jun 10-15, 2012, Brisbane, Australia.

8. C. H. Tan, E. Y. Cheu, J. Hu, **Q. Yu**, and H. Tang, "Associative Memory Model of Hippocampus CA3 Using Spike Response Neurons", *in IEEE 18th International Conference on Neural Information Processing (ICONIP)*, pp. 493-500, Nov 14-17, 2011, Shanghai, China.