Energy Efficient Deep Spiking Recurrent Neural Networks: A Reservoir Computing-Based Approach

Kian Hamedani

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

Doctor of Philosophy
in
Electrical Engineering

Yang (Cindy) Yi, Chair
Lingjia Liu, Co-Chair
Dong. S. Ha
Jeffrey H. Reed
Ali R. Butt

May 12, 2020
Blacksburg, Virginia

Keywords: Recurrent Neural Network, Reservoir Computing, Spiking Neural Networks, Smart Grids, Spectrum Sensing, Adversarial Attacks.

Energy Efficient Deep Spiking Recurrent Neural Networks: A Reservoir Computing-Based Approach

Kian Hamedani

ABSTRACT

Recurrent neural networks (RNNs) have been widely used for supervised pattern recognition and exploring the underlying spatio-temporal correlation. However, due to the vanishing/exploding gradient problem, training a fully connected RNN in many cases is very difficult or even impossible. The difficulties of training traditional RNNs, led us to reservoir computing (RC) which recently attracted a lot of attention due to its simple training methods and fixed weights at its recurrent layer. There are three different categories of RC systems, namely, echo state networks (ESNs), liquid state machines (LSMs), and delayed feedback reservoirs (DFRs). In this dissertation a novel structure of RNNs which is inspired by dynamic delayed feedback loops is introduced. In the reservoir (recurrent) layer of DFR, only one neuron is required which makes DFRs extremely suitable for hardware implementations. The main motivation of this dissertation is to introduce an energy efficient, and easy to train RNN while this model achieves high performances in different tasks compared to the state-of-the-art. To improve the energy efficiency of our model, we propose to adopt spiking neurons as the information processing unit of DFR. Spiking neural networks (SNNs) are the most biologically plausible and energy efficient class of artificial neural networks (ANNs). The traditional analog ANNs have marginal similarity with the brain-like information processing. It is clear that the biological neurons communicate together through spikes. Therefore, artificial SNNs have been introduced to mimic the biological neurons. On the other hand, the hardware implementation of SNNs have shown to be extremely energy efficient. Towards achieving this overarching goal, this dissertation presents a spiking DFR (SDFR) with novel

encoding schemes, and defense mechanisms against adversarial attacks. To verify the effectiveness and performance of the SDFR, it is adopted in three different applications where there exists a significant Spatio-temporal correlations. These three applications are attack detection in smart grids, spectrum sensing of multi-input-multi-output(MIMO)-orthogonal frequency division multiplexing (OFDM) Dynamic Spectrum Sharing (DSS) systems, and video-based face recognition.

In this dissertation, the performance of SDFR is first verified in cyber attack detection in Smart grids. Smart grids are a new generation of power grids which guarantee a more reliable and efficient transmission and delivery of power to the costumers. A more reliable and efficient power generation and distribution can be realized through the integration of internet, telecommunication, and energy technologies. The convergence of different technologies, brings up opportunities, but the challenges are also inevitable. One of the major challenges that pose threat to the smart grids is cyber-attacks. A novel method is developed to detect false data injection (FDI) attacks in smart grids.

The second novel application of SDFR is the spectrum sensing of MIMO-OFDM DSS systems. DSS is being implemented in the fifth generation of wireless communication systems (5G) to improve the spectrum efficiency. In a MIMO-OFDM system, not all the subcarriers are utilized simultaneously by the primary user (PU). Therefore, it is essential to sense the idle frequency bands and assign them to the secondary user (SU). The effectiveness of SDFR in capturing the spatio-temporal correlation of MIMO-OFDM time-series and predicting the availability of frequency bands in the future time slots is studied as well.

In the third application, the SDFR is modified to be adopted in video-based face recognition. In this task, the SDFR is leveraged to recognize the identities of different subjects while they rotate their heads in different angles.

Another contribution of this dissertation is to propose a novel encoding scheme of spiking

neurons which is inspired by the cognitive studies of rats. For the first time, the multiplexing of multiple neural codes is introduced and it is shown that the robustness and resilience of the spiking neurons is increased against noisy data, and adversarial attacks, respectively. Adversarial attacks are small and imperceptible perturbations of the input data, which have shown to be able to fool deep learning (DL) models. So far, many adversarial attack and defense mechanisms have been introduced for DL models. Compromising the security and reliability of artificial intelligence (AI) systems is a major concern of government, industry and cyber-security researchers, in that insufficient protections can compromise the security and privacy of everyone in society. Finally, a defense mechanism to protect spiking neurons against adversarial attacks is introduced for the first time. In a nutshell, this dissertation presents a novel energy efficient deep spiking recurrent neural network which is inspired by delayed dynamic loops. The effectiveness of the introduced model is verified in several different applications. At the end, novel encoding and defense mechanisms are introduced which improve the robustness of the model against noise and adversarial attacks.

Energy Efficient Deep Spiking Recurrent Neural Networks: A Reservoir Computing-Based Approach

Kian Hamedani

General Audience Abstract

The ultimate goal of artificial intelligence (AI) is to mimic the human brain. Artificial neural networks (ANN) are an attempt to realize that goal. However, traditional ANNs are very far from mimicking biological neurons. It is well-known that biological neurons communicate with one another through signals in the format of spikes. Therefore, artificial spiking neural networks (SNNs) have been introduced which behave more similarly to biological neurons. Moreover, SNNs are very energy efficient which makes them a suitable choice for hardware implementation of ANNs (neuromporphic computing). Despite the many benefits that are brought about by SNNs, they are still behind traditional ANNs in terms of performance. Therefore, in this dissertation, a new structure of SNNs is introduced which outperforms the traditional ANNs in three different applications. This new structure is inspired by delayed dynamic loops which exist in biological brains. The main objective of this novel structure is to capture the spatio-temporal correlation which exists in time-series while the training overhead and power consumption is reduced.

Another contribution of this dissertation is to introduce novel encoding schemes for spiking neurons. It is clear that biological neurons leverage spikes, but the language that they use to communicate is not clear. Hence, the spikes require to be encoded in a certain language which is called neural spike encoding scheme. Inspired by the cognitive studies of rats, a novel encoding scheme is presented.

Lastly, it is shown that the introduced encoding scheme increases the robustness of SNNs against noisy data and adversarial attacks. AI models including SNNs have shown to be

vulnerable to adversarial attacks. Adversarial attacks are minor perturbations of the input data that can cause the AI model to misscalassify the data. For the first time, a defense mechanism is introduced which can protect SNNs against such attacks.

Acknowledgement

I would like to express my most sincere gratitude to my Ph.D. advisors Dr. Yang (Cindy) Yi and Dr. Lingjia Liu for their continuous support, and guidance during my Ph.D. studies. I have had their support, encouragement, and guidance since the first day I began my Ph.D. I will always be grateful to them for sharing their knowledge and experience with me which made my Ph.D. journey very productive and pleasant.

I would also like to express my deepest appreciation to the members of my Ph.D. advisory and exam committee: Professors Dong. S. Ha, Jeffrey H. Reed, and Ali R. Butt for their valuable comments and suggestions which significantly helped me towards improving my dissertation.

Words fail to express my appreciation to my loving wife, Kate. The completion of this dissertation would not have been possible without her nurturing and support. I thank my beloved parents,parents in law, and brothers for their uncoditional love , care, and sacrifice that they dedicated to me in my life.

Last but not the least, I would like to thank my labmates who I had the pleasure of working with both at VT and KU.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In the recent years, the major advancements of artificial intelligence (AI) and machine learning (ML) have been realized due to the significant progress in the field of artificial neural networks (ANNs). ANNs are the core information processing technology of many real world applications and have witnessed striking progress in the past few years. Neurons are the fundamental units of biological neurons and ANNs are mathematical representation of biological neurons. In fact ANNs are constituted of a network of neuron-like information processing units which are connected via synaptic weights. In general, there are two categories of ANNs, namely, feedforward neural networks (FNNs) and recurrent neural networks (RNNs) [3]. The choice of which category of ANN depends on the application. For the applications with static data where no temporal correlation exists FNNs are mainly used. On the other hand, RNNs are used for processing the dynamic data, e.g., time-series, video frames, and speech signal [4]. RNNs capture the temporal correlation via cyclic connections within the network of nodes [3]. Backpropogation (BP) is the most successful algorithm to train FNNs [5]. The

chain rule is used in BP algorithm to calculate the synaptic weights. The synaptic weights are then update based on the gradient of the loss function. There is no guarantee that the BP algorithm can achieve the global minimum due to the non-convex surface of the loss function [3]. Convolutional neural networks (CNNs) are a variant of FNNs which have been popular in computer vision [6]. CNNs are able to capture the local dependencies of visual information and leverage that information to improve the image classification [3]. RNNs are used to model the dynamics via recurrent connections among the nodes. In fact, the outputs of artificial neurons within RNNs are not only dependent on the current input, but they depend on the previous samples within time domain as well [7]. RNNs are developed to find a nonlinear mapping between a sequence of input and target data. The input sequence can be expressed as $[x^{(1)}, x^{(2)}, ..., x^{(T)}]$ where each temporal sample, $x^{(t)}$ is a vector of real valued features of the input data, and T is the temporal length of the input. RNNs have shown to be very effective to analyze the time-series with short length [8]. However, as the length of the input increases, RNNs suffer from vanishing and exploding gradient problem [9]. The vanishing and exploding gradient problem rises while the BP error is calculated over a long sequence. Figure 1.1 represents a simple example of RNNs with one input, output, and hidden recurrent node. The nodes which are used to connect the adjacent temporal samples are called recurrent nodes. The recurrent nodes may establish cycles too. The cycles are the recurrent connections with the length of one which in fact form a connection from a node to itself across the time. The current input $x(t)$ and the value of the previous hidden state $h(t-1)$ are received by the recurrent node at each time sample t. Consequently, the output $\hat{y(t)}$ at time t is estimated by the hidden state at time t. The necessary calculations to compute the hidden states and the output of a simple RNN as Figure 1.1 are presented as follows [10],

$$h(t) = f(W_{hx}x(t) + W_{hh}h(t-1) + b_h), \tag{1.1}$$

where f is a nonlinear activation function, W_{hx} is the set of synaptic weights between the input and the hidden layer, W_{hh} is the set of synaptic weights between the hidden recurrent layer and itself at consecutive temporal instances, and b_h is the set of bias parameters at the hidden layer which its role is to increase the capacity of RNN to learn from an offset. The output is estimated as [11],

$$\hat{y(t)} = softmax(W_{hy}h(t) + b_y), \tag{1.2}$$

where W_{hy} and b_y are the weight matrix connecting the hidden states to the output and the bias vector of the output layer, respectively. $softmax$ is a function that takes the real valued numbers as the input and it generates the probability distributions of the input through normalizing them using the exponential of the input numbers. In order to mimic the biological neurons, an activation function is required to map the input to the output. $Sigmoid$, $tangsig$, $relu$ and $softmax$ are the most well-known activation functions which have been introduced to model the biological neurons.

Assume that an input is passed to the network (Figure 1.1) at time τ and its corresponding error is calculated at time t. Tying the weights across the temporal samples, leads to the fixed weights of the recurrent hidden units. The vanishing/exploding gradient problem is visualized in Figure 1.2. In order to train RNNs, the recurrent hidden layer is unfolded through time and the algorithm which is used to train the synaptic weights is called BP through time (BPTT) [12]. The vanishing gradient occurs when $W_{hh} < 1$. As the recurrent unit is unfolded through time, W_{hh} is trained using the BPTT algorithm. However, as the chain rule is applied to BPTT, W_{hh} gets smaller and smaller until it converges to zero. Hence, for longer time series the vanishing gradient problem occurs which prevents RNNs from learning the nonlinear mapping between the input and output sequences.

On the other hand while $W_{hh} > 1$ and BPTT is applied on very long sequences, W_{hh} will

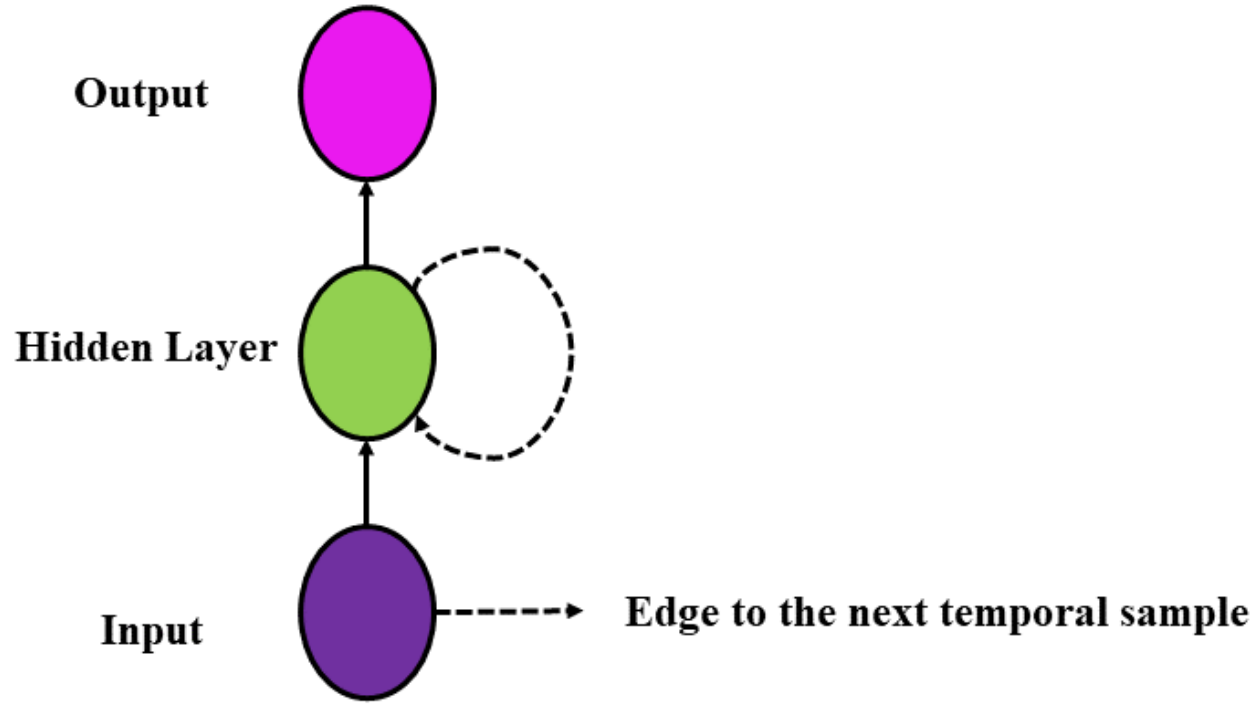

Figure 1.1. A simple RNN with one input node, one output node, and one hidden recurrent node.

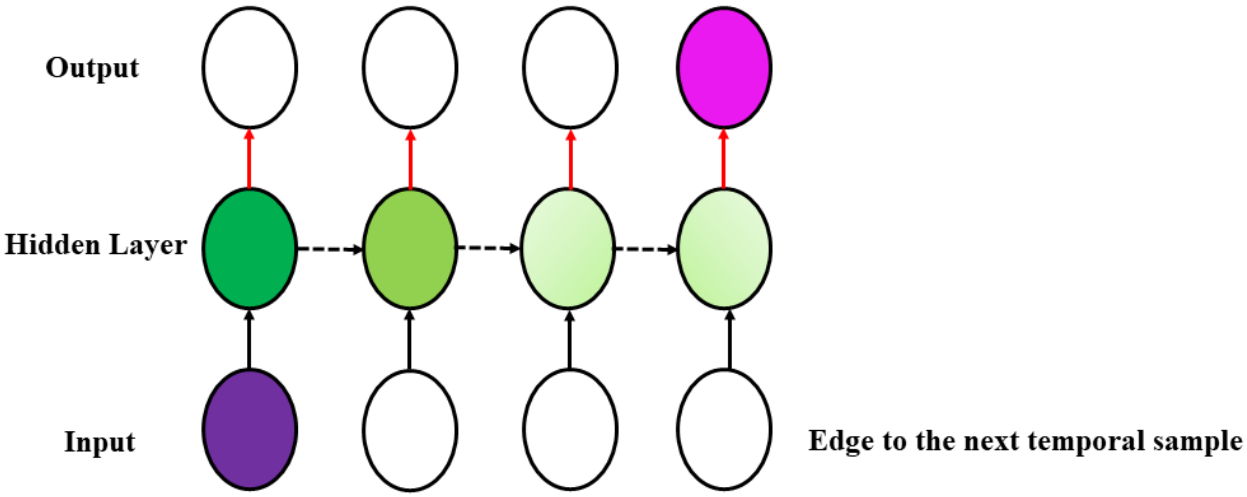

Figure 1.2. Unfolding the recurrent unit through time and Vanishing/Exploding Gradient

converge to very large values. This is called exploding gradient and similar to vanishing gradient is a major challenge for training RNNs [13]. Reservoir computing (RC) is a new class of RNNs which has recently been introduced to solve the vanishing/exploding gradient problem of RNNs [14–19]. The RC models are regarded as an extension of ANNs where the reservoirs are a set of neurons which are connected together via fixed weight [20,21]. The RC models have been extensively adopted for temporal/sequential information processing. The recurrent layer of an RC model is formed via a set of neurons in the reservoir layer where they are sparsely connected connected with some random recurrent weights [4]. In an RC model, the input sequential data is mapped from a low-dimensional space to a high-dimensional space which is formed as the reservoir layer. The weight which connect the reservoir neurons are fixed and do not require any training, hence, the vanishing/exploding gradient problem of RNNs can be resolved. The only weights in an RC model which require training are the readout weights which map the high-dimensional features of the reservoir layer to the output. Reducing the computational complexity of the training is another advantage of RC models as the input and recurrent wights are fixed and do not undergo any training. In many cases including temporal pattern classification, synthesize, and prediction, RC models have shown equal or even better performances than standard RNNs. Echo state networks (ESNs) and liquid state machine (LSMs) are the two main categories of RC models. Recently, inspired by the delayed dynamic loops, a new category of RC models has been introduced. In order to create effective reservoirs, some certain adjustments are required. Some of these adjustments are task dependent and some of them are independent of task. In ESNs a certain property named as echo state property needs to be satisfied. According to the echo state property, the spectral radius of the reservoir weights has to be smaller than 1. The spectral radius of the reservoir weights is defined as the maximum absolute eigenvalue of the reservoir weights. In that way, the reservoir nodes can form short-term fading memory which can map the data from low-dimensional space to high-dimensional space. The structure of an RC model

is presented in Figure 1.3. As it can be seen, RC models are constituted of three layers, namely, input, reservoir, and output layer.

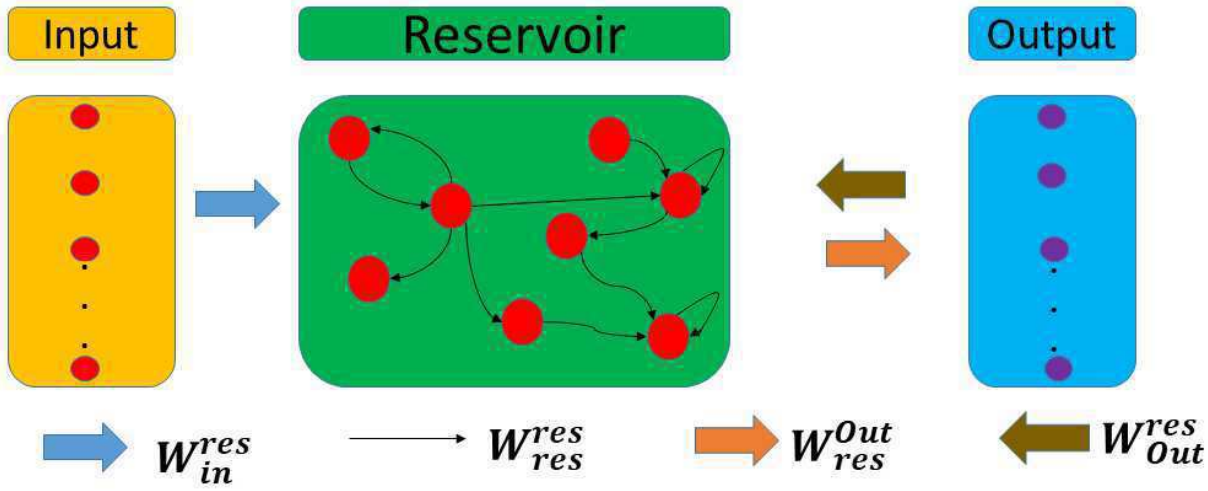

Figure 1.3. Structure of RC

1.1.1 Reservoir Computing

ESNs and LSMs were introduced in early 2000s as the first two major RC models. The main difference between RC models and standard RNNs is that the recurrent (reservoir) layer of RC models is fixed and does not require any training. The state of the reservoirs is expressed as,

$$h(n) = f(W_{in}^{res}x(n) + W_{res}^{res}h(n-1) + W_{out}^{res}y(n-1)), \tag{1.3}$$

where n denotes the discrete time, W_{in}^{res} is the matrix of weights which connect the input nodes to the reservoir layer, W_{res}^{res} is the matrix of the recurrent weights in the reservoir layer, and W_{out}^{res} is the matrix of the feedback weights which are used to incorporate the output of the previous time step in the current state. The output of RC is estimated as,

$$y(n) = W_{res}^{out}h(n), \tag{1.4}$$

where $y(n)$ is the estimated output at time step n, and W_{res}^{out} is the matrix of weights which maps the hidden high-dimensional reservoir states to the output. The W_{res}^{out} is the only set of weights in RC which undergoes a training step. In the recent years RC models have been adopted in many applications. The RC models have shown to be very effective in biomedical, audio, and visual signal processing and prediction. They have also been applied in security related problems including cryptography. The hardware implementation of RC models has drawn a remarkable attention due to the easy training and considerable reduction of computational complexity. The physical realization of RC models is a straightforward method to implement RNNs which is called neuromporphic computing [22, 23]. In order to make the hardware implementation of RC models effective and reliable, there are certain requirements which have to be satisfied. 1) High-dimensional behavior is necessary for RC models. The input data can be mapped from low-dimensional space to high-dimensional space due to the high-dimensional behavior of the reservoir layer. Mapping the data to high-dimensional space facilitates the classification tasks while the data is not linearly separable. On the other hand, the spatio-temporal correlations can be read out in prediction tasks. 2)An RC model has to be nonlinear, otherwise it is not capable of estimating the nonlinear mappings between the input and output data. 3) The neurons of the reservoir layer have to form fading (short-term) memory. In that way, the states of the reservoirs are only dependent on the recent past inputs and the far past inputs are faded and, hence, not taken into effect. 4) Lastly, separation is another property which has to be satisfied by RC models, i.e., they have to be insensitive to the minor fluctuations of the input. In other words, RC models have to be robust against noise.

Besides, ESN and LSM, there is another category of RC models which is inspired by delayed dynamical systems. Using a time-delayed dynamical system is another approach to generate

a high-diemsnional system. A time-delayed dynamical system is presented as,

$$\frac{dh(t)}{dt} = F\left(t, h(t), h(t-\tau)\right), \tag{1.5}$$

where τ is the delay of the dynamic system, F is a nonlinear function, and $h(t)$ is the state of the delay loop at time t. Depending on the parameters of the delayed dynamical system, the delayed dynamical system is capable of exhibiting considerable nonlinear behaviors including periodic and chaotic. The delayed based reservoir models require only one neuron as their information processing unit which makes them extremely suitable for implementation of neuromorphic circuits compared with network-based RC models. The focus of this dissertation is on the single-node delay based RC models.

1.1.2 Spiking Neural Networks

In ANNs the neurons are modeled using continuous valued activation function such as *sigmoid* and *tansig*. Yet, modeling the biological neurons with a continuous activation function is far from the reality. Biological neurons communicate together via discrete signals, namely, spikes. Therefore, to understand and model the computational mechanisms of brain, it is essential to introduce artificial neurons which mimic the spikes. Spiking neural networks (SNNs) are an attempt to represent a mathematical model of brain which is more biologically plausible than ANNs. Unlike ANNs which use continuous activation function, SNNs adopt activation functions which are event-driven and also sparse in time and space. Being sparse in time and space and also event driven, makes SNNs very energy efficient. In fact SNNs are the first choice of neuromorphic hardware realization of biological neurons. Despite, the several advantages of SNNs, they have not been fully adopted in real world applications. This is because SNNs are still behind ANNs in terms of performance in several

applications. The BP algorithm works well with continuous and differentiable activation functions. However, the activation function of SNNs is non-differentiable and discrete in time. This makes the training of SNNs very challenging and still an open problem. In recent years, several supervised and unsupervised training algorithms have been proposed for SNNs. The recent studies have shown that the gap between SNNs and ANNs in terms of accuracy is decreasing and even in some tasks has completely vanished. In general SNNs are better candidates for processing the spatio-temporal data. Due to the advantages of SNN, the motivation of this dissertation is to introduce a RC model which is based on delay dynamic loops where a spiking neuron is adopted as the single unit of information processing.

1.2 Contribution and Dissertation Structure

1.2.1 Chapter2

In chapter 2 of this dissertation, a recurrent structure of SNNs is introduced, which is inspired by delayed feedback reservoirs (DFRs). To verify the performance of the introduced structure it is used in a novel application which RC models have not been used before, i.e., single-period attack detection in smart grids. It is shown that the combination of temporal encoding, DFR, and a multilayer perceptron (MLP) as the output readout layer can achieve significant performance improvement over existing attack detection methods such as MLPs, support vector machines (SVM), and conventional state vector estimation (SVE) in terms of attack detection in smart grids.The introduced algorithms is als shown to be more robust than MLP and SVE in dealing with different variables such as the amplitude of the attack, attack types, and the number of compromised measurements in smart grids [24].

1.2.2 Chapter 3

SNNs have been widely used for supervised pattern recognition exploring the underlying spatiotemporal correlation. Meanwhile, spatio-temporal correlation manifests significantly between different components in a smart grid making the spiking neural network a desirable candidate for false data injection attack detection. In this chapter, a SNN based technique for dynamic cyber-attack detection in a smart grid is introduced. This is achieved through judiciously integrating spiking neurons with a special recurrent neural network called the delayed feedback reservoir computing. The inter-spike interval encoding is also explored in the precise-spike-driven (PSD) synaptic plasticity based training process. The simulation results suggest that the introduced method outperforms MLPs and can achieve a significantly better performance compared to the state-of-the-art techniques. Furthermore, our analysis indicates that the delay value in the delayed feedback reservoir will have a substantial impact on the overall system performance. Several encoding schemes including latency, latency phase and ISI are investigated to train the SNN using the PSD algorithm. The results suggest that the ISI encoding leads to the least training error and the best attack detection performance [25, 26].

1.2.3 Chapter 4

In this chapter, the introduced model is modified to be applied in a different application. This chapter introduces a novel spectrum sensing method for multiple-input-multiple-output - orthogonal frequency division multiplexing (MIMO-OFDM) systems in dynamic spectrum sharing (DSS) environments. The model is extended in time and space domains to capture the spatial and temporal correlations in DSS environments. Conditional Generative adversarial networks (cGANs) are introduced to tackle the data scarcity issue arose from applying

Ml based techniques to MIMO-OFDM systems where training data is limited. Simulation results suggest that the probability of detection of the introduced RC based spectrum sensing at the low signal-to-noise (SNR) regime is significantly higher than the state-of-the-art techniques and the computational complexity is also reduced compared with traditional RNNs. In this chapter the combination of spiking DFR (SDFR) and cGAN is introduced to to synthesize more training data while the training data is scarce. We investigate the quality of the synthesized data in different scenarios. To the best of our knowledge, this is the first time that the cGANs are used to synthesize MIMO-OFDM symbols for data augmentation. It is shown that the detection performance of our introduced spectrum sensing approach will significantly drop when the size of training data is limited and we resolve this issue by introducing a combined cGAN and SDFR platform.

1.2.4 Chapter 5

In this chapter two other major drawbacks of SNNs are addressed: 1) existing encoding mechanisms are not robust against noise; and 2) defense mechanisms against adversarial attacks are lacking. To address these issues, this chapter introduces an easy-to-train spiking recurrent structure and shows its capability in classifying time-series data. Furthermore, motivated by recent studies in cognitive science, a new multiplexing encoding mechanism is introduced and it is shown that it outperforms the state-of-the-art codes. To thwart adversarial attacks for the first time in the literature, we introduce an effective defense mechanism for SNNs. The efficacy of the introduced network and mechanisms are validated in two applications, namely, video-based face recognition and bad data detection in smart grids (a cyber-physical system). The results demonstrate the effectiveness of our mechanisms in terms of classification accuracy, robustness to noise, and resilience to adversarial attacks. For the first time, the idea of multiplexing two encoding schemes in artificial SNNs is intro-

duced. More specifically, inspired by recent studies in cognitive neuroscience, we multiplex the phase and temporal encoding of the neuron spikes. Under our SDFR model, we show that our encoding scheme achieves higher classification accuracy, is more robust to noise, and is more robust to adversarial attacks than existing encoding schemes. An effective defense mechanism for SNN against adversarial attacks is also presented in this chapter. At the end a game theoretic approach is presented to optimize the attack/defence strategy while the attack/defence budget is limited.

1.3 Related Work

1.3.1 The variants of RC models and their applications

To improve the performance of RC models, several modifications have been introduced. These modifications have been mainly performed on different aspects of RC models. In [27–29] new architectures of RC models have been developed where multiple reservoirs are adopted to improve the performance of RC systems. Evolving reservoirs, combining RC models with other features extraction techniques such as reinforcement learning, and untrained CNN layers are other recent attempts to improve the performance of RC models. The theories of information, dynamical systems, and statistics have been used to better understand the relationship between the dynamics of the reservoirs and their performance. These studies have shown that RC models can form fading memory and demonstrate high-dimensional behavior only if their parameters are tuned somehow that they can operate at the edge of chaos [30, 31]. Hence, a proper hyperparameter tuning of RC models is very essential.

Two approaches have been devised to extend the structure of single-node delay based reser-

voirs. In the first approach, the outputs of the ensemble of two delay based reservoirs are combined together and as a result, performance, and the robustness of delay based reservoirs are improved. The second approach relies on the circular concatenation of the two delay based reservoirs which leads to a longer delay line. So far RC models have been adopted in different applications. Spoken digit recognition, action recognition, digit recognition, chaotic time-series prediction, NARMA-10 time series prediction, wave generation, measuring the memory capacity, channel equalization, biomedical signal classification, and analyzing the financial data are just few examples that RC models have been successfully applied for. In this dissertation, besides introducing a new spiking structure of RC models, novel applications of RC models are presented as well.

1.3.2 Electronics RC

Developing energy efficient ML devices with low computational complexity has been the subject of many RC based neuromorphic circuits studies [32–36]. Single-node delay based RC systems have drawn more attention for neuromorphic circuits implementations because they consume less energy, are faster to train. Analog circuits and Field-Programmable Gate Array (FPGAs) have been used to implement the delay based reservoirs on hardware. The nonlinear single node along with the delay loop was can be built using analog circuits and the pre- and post-processing blocks are developed by digital components. Masking is an important pre-processing step while using delay based RC models. A model was proposed in [37] where digital-to-analog and analog-to-digital converters with 12 bits resolution were adopted as the interfaces between the digital and analog parts of the system. That model has shown to be very effective in spoken digit recognition, time-series prediction, and estimation of memory capacity. The reconfigurability of FPGAs has made them a popular choice for neuromorphic implementation of ANNs as the weights of ANNs are updated adaptive and

a lot of concurrent processing are performed. In the domain of RC, FPGAs have been used to implement the reservoir and readout layers. In [38] a single-node delay based reservoir using Boolean logic is implemented on FPGA. The invereter gates are paired and cascaded together to realize the delay loop and the Boolean logic component acts as an XOR gate. An external computer was used to collect the combination of the virtual nodes' states and use their linear combination to estimate the output. Very large scale integration circuits (VLSI) have also been used to implement RC circuits. To reduce the costs of hardware implementation, VLSI circuits have been used to implement the biologically inspired RC models, namely, LSMs. Memristive RC is another approach to realize the neuromporphic circuits. Memristives are elements which their resistance changes through time with respect to the input intensity. The adaptive variations of the memristives resistance makes them a proper choice to implemet the synaptic connections between neurons. Memristives based RC models are categorized in to two major groups: 1) neuromemristive reservoirs which consist both the neurons and synapse circuits, 2) memristive reservoirs which only consist synaptic circuits without any neurons.

1.3.3 SNNs: A Biologically Inspired Approach for Information Processing

SNNs are regarded as the third generation of ANNs and they have encouraged a lot of studies on biologically inspired pattern recognition. The discrete action potentials (spikes) that are fired by the biological neurons were the first motivation to develop SNNs. In this part the recent trends, advances and applications of SNNs are presented.

- **SNN Architecture.** Similar to ANNs, SNNs are also constituted of a network of artificial neurons which are connected via adjustable scalar weights. However, unlike

ANNs, in SNNs the artificial neurons are discrete in time. To process the analog inputs using SNNs, the first step is to encode the input using rate or some type of temporal encoding. In SNNs, similar to biological neurons, synaptic inputs are received from adjacent neurons. There are two types of dynamics in biological networks (similarly in SNNs), namely, action potential (AP) dynamics, and network level dynamics. The network level dynamics of SNNs are highly simplified compared to biological networks. The membrane potential of post-synaptic neurons is modulated via pre- synaptic neurons activity which leads to an AP (spike) when the membrane voltage exceeds a certain threshold value. The model which was introduced by Hodgkin and Huxley in 1952 was the first mathematical representation of biological neurons. The Hodgkin and Huxley model provides rich biological details. The Hodgkin and Huxley model is computationally expensive, hence, in the recent years other models such as the the leaky integrated-and-fire (LIF) neuron, spike response model (SRM), and Izhikevich neuron model have been introduced. Specifically, the LIF neuron has drawn a lot of attention.

- **Training Algorithms of SNNs.** The most challenging part of training SNNs is because the spiking neurons are non-differentiable and the BP algorithm cannot be used to train SNNs. In recent years many unsupervised and supervised training algorithms have been introduced for tarining SNNs. Many different type of learning rule for SNNs have been identified by neuroscientists and they are all under the same unified terminology, namely, spike-timing-dependent plasticity (STDP). The STDP implies that all learning rules of SNNs, supervised or unsupervised, are based on adjusting the synaptic weights through time. If a post-synaptic neuron fires a spike after the pre-synaptic neuron, then the connection weights are strengthened and, conversely. Strengthening of the weight connections is called ong-term potentiation (LTP) and weakening the wight connections is called long-term depression (LTD).

- **Applications of SNNs.** Deep neural networks (DNNs) are biologically inspired, end-to-end models which are trained via different optimization techniques to find the optimal mapping between the input and output and they have been adopted in many ML and AI applications. Studies on biological brains have shown that the information is processed through a multi-layer process. Therefore, the AI researchers have been trying models of ANNs which process the information in mutil-layers. Increasing the depth of ANNs has led to DNN which in some applications have shown equal or even better performances than brains. SNNs have also been applied in a variety of applications including, image, speech, and biomedical signals classification. Standard DNN models with continuous activation functions have achieved remarkable performances in different applications. SNNs are very energy efficient which makes them very popular for neuromorphic circuits. SNNs similar to DNNs have been applied in different applications in visual, speech, and medical signal analysis and classification. Some of these applications include processing the spatio-temporal brain data, path finding in robots, multi-object detection and classification in video streams, and speech recognition.

1.3.4 Novel Applications of Spiking Delayed Feedback Reservoir

As it was mentioned the main motivation of this dissertation is to introduce an energy efficient, and easy to train spiking recurrent neural network which is inspired by single-node delay based RC model. After developing this model, it is required to verify its performance in different applications. In this dissertation, our introduced model is applied in novel applications which neither of RC and SNN models have not been used before. In this part these novel applications are introduced and a survey of the related works of these applications is presented.

- **False Data Injection (FDI) Detection ins Smart Grids.** The SDFR is first applied in FDI detection in smart grids. Smart grids are new infrastructure that integrate energy with many different technologies, such as telecommunication, internet, and electronic devices. This convergence of different technologies brings up some opportunities and challenges as well. The main opportunity that smart grids provide is a bidirectional flow of electricity and information between power suppliers and costumers, which will result in a more efficient distribution of power. However, due to the integration of different technologies, the smart grids are more vulnerable to cyber-attacks [26, 39]. FDI attacks are known to be as one of the most exceedingly malicious cyber-security concerns in smart grids. Cyber security tries to maintain a reliable and secure communication between different components of the network, including communication networks and computer systems [40]. As a result of this secure communication, supervisory control and data acquisition (SCADA) system can have a better estimation of the network state. State estimation is a critical process in control system and due to this fact, the SCADA is usually a target for attackers. Injecting false data to the SCADA, manipulates the state estimation and it can cause economic gains for the attacker [40]. Approaches based on machine learning have been extensively used for FDI detection in smart grids [41]. Different machine learning methods including ANNs, support vector machines (SVM), and k-nearest neighbor (KNN) have been used for this purpose [41]. Approaches based on machine learning have yielded better performance than the traditional state vector estimation (SVE) in detecting the FDI in smart grids. Esmalifalak *et al* [42] have applied dimension reduction for mapping the data collected from the network and used both unsupervised and supervised machine learning algorithms to detect the stealth data injections in smart grids. In [43], the performance of SVM, and KNN in detection the false data in an IEEE- 30 bus system under balanced and imbalanced data scenarios is studied. Mohammadpourfard

et al [44] proposed an unsupervised anomaly detection method in smart grids which considers the effect of wind power generation and topology configuration. Zhao *et al* [45] have proposed a method based on short term state forecasting which considers a temporal correlation among measurements. Moslemi *et al* [46], proposed an approach based on maximum likelihood which is decentralized, and the near chordal sparsity of smart grids is considered in order to detect the FDI.

- **Spectrum Sensing of MIMO-OFDM Systems in DSS Environments.** Spectrum sensing of MIMO-OFDM systems is the second novel application which is introduced in this dissertation. MIMO- technology combined with OFDM has been adopted in many advanced wireless communication systems. The combination of MIMO and OFDM technologies improves the spectral efficiency, as the MIMO utilizes the spatial multiplexing gain and the OFDM avoids frequency selective fading. Besides, adding cyclic prefix (CP) to the OFDM symbols decreases inter channel interference (ICI), and inter symbol interference (ISI). However, in a MIMO-OFDM based wireless communication system not all the subcarriers are utilized simultaneously by the primary user (PU), and the spectrum utilization efficiency is low [47]. Dynamic spectrum sharing (DSS) in a MIMOOFDM system introduce a solution to resolve this problem where the under utilized subcarriers can be used by the secondary users (SUs). The SUs are allowed to transmit signals only on the subcarriers that are found idle and they should evacuate those bands as soon as the PU wants to use them. Therefore, it is fundamental for the SUs to perform spectrum sensing subsequently to identify spectrum holes accurately and the interference is minimized. So far, in the literature three major techniques have been introduced as the classical spectrum sensing methods. These methods are energy detection, matched filtering, and cyclo-stationary feature detection which suffer from several drawbacks including, low probability of detection at low SNRs, requiring accurate prior knowledge of the signal, and computational complexity,

respectively [47–49]. Due to the limitations of classical spectrum sensing techniques, machine learning (ML) approaches have drawn a lot of attention as they have several advantages over traditional spectrum sensing techniques. These advantages are: 1) the ML based spectrum sensing approaches are more adaptive and can learn the surrounding DSS environment (e.g., the fading channel) effectively; 2) the detection performance of the ML based spectrum sensing techniques are better as they can identify the decision boundaries [50–61]. In the recent years many supervised and unsupervised ML and deep learning (DL) techniques have been leveraged in spectrum sensing [62, 63]. However, most of the ML algorithms introduced for spectrum sensing are not able to capture the hidden spatio-temporal correlations of the received signals. Convolutional neural networks (CNN) [64], RNNs, deep Bayesian networks, stacked auto encoders (SAEs) [65–69] are among the most recent methods [70]. The DL based approaches which have been introduced so far face many challenges and issues. CNNs loose some information during the feature extraction phase as the neurons are partially connected together and they are also very computationally expensive. Deep Bayesian networks can achieve good performances while the training data is limited, but they are required to estimate the posterior distribution of the model's parameters [16, 71–76]. This is specially very challenging in the scenarios where the size of network is very large. SAEs have recently been adopted in spectrum sensing while there is limited labeled training data and showed good performance in a single-input-single-output (SISO)-OFDM system. However, this method is computationally very complex and requires several minutes of training ($\sim$30 minutes) [65, 77–79]. Applying SAEs in MIMO systems will even further increase their computational complexity. In a DSS scenario where the channel statistics vary rapidly, we need to choose a technique that is very fast to train. Due to the limits of the current DL based techniques, we introduce SDFR to overcome these challenges.

- **Video-Based Face Recognition.** Recognizing the identities of people using their face has been one of the most popular biometric tools in the past few decades. However, there are certain challenges that restrict the widespread usage of face recognition technologies. These challenges are mainly caused due to the variations of pose, illumination, and expression. Deep convolutional neural networks (DCNNs) have successfully addressed some of these challenges. However, DCNNs are only effective while the identity of the subjects is recognized only from one image. However, in scenarios where it is required to recognize the identity of the subjects from a series of frames, i.e., video-based face recognition, DCNNs fail to achieve high accuracy because they are only able to capture the spatial information and not the temporal. Therefore, in such scenarios a model like RNNs which captures the temporal correlation achieves better performances than DCNNs. The third application of SDFR in this dissertation is to recognize the faces of few subjects from videos while they rotate their heads in different angles.

1.3.5 State-Of-The-Art Encoding Schemes of SNNs

It is still unclear what method is employed by neurons to encode their information. This requires further investigation to improve the performance of SNNs. The major encoding methods employed by neurons include rate and temporal encoding [14]. Previously rate encoding approaches were more common in neuroscience literature. However, recent studies have shown that temporal encoding approaches have better performance compared to rate encoding approaches [15]. The precise time of spikes is required in temporal codes to encode the information. As one example, the neurons in the retina, the lateral geniculate nucleus, and the visual cortex respond within milliseconds (ms) to visual stimuli. Is has also been observed that the computational complexity of temporal encoding is less than rate encoding

[14]. Although the two encoding schemes have been shown to be effective in different scenarios and applications several questions remain unanswered. For example, is there an optimal encoding scheme? Can we combine two or more schemes to encode the spikes?. In this dissertation, these questions are answered and a novel encoding mechanism is introduced.

1.3.6 Adversarial Attacks on SNNs & Defence Mechanisms

Adversarial attacks are regarded as a serious threat to the DL models. DNNs are being leveraged in robotics, autonomous cars, and drones. Therefore, the robustness and resilience of DNNs against adversarial attacks needs to be improved. Adversarial attacks are carefully crafted small perturbations which are applied on the clean inputs. It has been shown that DNNs are extremely vulnerable to such attacks and they can cause wrong predictions with high confidence. Ensemble training, using random noise for implicit prior modeling, distillation, and salable training are just a few techniques to protect DNNs against adversarial attacks. SNNs have shown to be more robust than DNNs against adversarial attacks. This behavior could be due to: 1) SNNs are more biologically plausible than DNNs, hence, it is more difficult to fool them, 2) the inherent stochastic characteristics of SNNs and temporal dynamics of them could be another reason which leads to more robustness of SNNs against adversarial attacks compared to DNNs. However, SNN is not an exception of deep learning models, i.e., it is vulnerable to adversarial attacks; and 2) SNNs are more resilient than DNNs against adversarial attacks. Although the latter observation supports the resilience of SNNs against adversarial attacks, without a defense mechanism, SNNs are still vulnerable. There are two categories of adversarial attack, namely, white-box attacks, and black-box attacks. In white-box attacks, it is assumed that the adversary is aware of the parameters of the model which it is trying to craft the attack with respect to. On the other hand, in black-box attacks, the adversary has no access to the parameters of the model. Hence, black-box

attacks are more challenging to craft but they are more realistic. In this dissertation, for the first time a defense mechanism for spiking neurons against adversarial attacks is introduced.

1.4 Publications

Journal Papers

- **Hamedani, K**, Liu, L. and Yi, Y. (2020). MIMO-OFDM Spectrum Sensing Using Delayed Feedback Reservoir Computing. Submitted to IEEE Transactions on Wireless Communication.

- **Hamedani, K.**, Liu, L., Atat, R., Wu, J. and Yi, Y., 2018. Reservoir computing meets smart grids: Attack detection using delayed feedback networks. IEEE Transactions on Industrial Informatics, 14(2), pp.734-743.

- **Hamedani, K.**, Liu, L., Hu, S., Ashdown, J., Wu, J. and Yi, Y., 2019. Detecting Dynamic Attacks in Smart Grids Using Reservoir Computing: A Spiking Delayed Feedback Reservoir Based Approach. IEEE Transactions on Emerging Topics in Computational Intelligence.

- Zhao, C., **Hamedani, K.**, Li, J. and Yi, Y., 2017. Analog spike-timing-dependent resistive crossbar design for brain inspired computing. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 8(1), pp.38-50.

- Li, J., Liu, L., Zhao, C., **Hamedani, K.**, Atat, R. and Yi, Y., 2017. Enabling sustainable cyber physical security systems through neuromorphic computing. IEEE Transactions on Sustainable Computing, 3(2), pp.112-125 (published).

Book Chapter

- **Hamedani, K.**, Zhou, Z., Bai, K. and Liu, L., 2019. The Novel Applications of Deep Reservoir Computing in Cyber-Security and Wireless Communication. In Intelligent System and Computing. IntechOpen.

Conference Papers

- **Hamedani, K.**, Liu, L. and Yi, Y. (2020). Spiking Recurrent Neural Network with Novel Encoding and Defense Mechanisms. Submitted to ICML 2020.
- **Hamedani, K.**, Liu, L., Liu, S., He, H., and Yi, Y., 2020. Deep Spiking Delayed Feedback Reservoirs and Its Application in Spectrum Sensing of MIMO-OFDM Dynamic Spectrum Sharing. In Proceedings of the AAAI Conference on Artificial Intelligence.
- Bai, K., Li, J., **Hamedani K.** and Yi, Y., 2018, June. Enabling an new era of brain-inspired computing: Energy-efficient spiking neural network with ring topology. In 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC) (pp. 1-6). IEEE.
- Li, J., Zhao, C., **Hamedani, K.** and Yi, Y., 2017, May. Analog hardware implementation of spike-based delayed feedback reservoir computing system. In 2017 International Joint Conference on Neural Networks (IJCNN) (pp. 3439-3446). IEEE

Chapter 2

Reservoir Computing Meets Smart Grids: Attack Detection Using Delayed Feedback Networks

2.1 Introduction

Recently, Academic and industry show a lot of attention to the energy harvesting from renewable resources, such as wind and solar [24, 80], due to the recent advancements and increase in the world's power demand field [24]. Energy harvesting technologies can supply the smart grid elements by up to 80%. This includes sensors and smart meters, which will significantly lower the ongoing maintenance costs and battery replacement costs and of the smart grid networks [24]. Further, renewable energy significantly reduces the fossil fuel consumption that leads a sustainable and greener environment [24]. Typically,a wind turbine with a rotor of 1 meter under an 8 meters per second wind speed or a solar panel of size 121 centimeters (cm) by 53 cm in diameter can deliver approximately an electric power

of 100 watt (W) [81]. Although renewable energies and energy harvesting seem attractive for smart grids applications [24], they have several complications and disadvantages. These drawbacks must be addressed in order to gain the maximum capabilities of them [82]. Since the primary sources for harvesting energy from such networks are wind and solar, they are not reliable source of power for smart grids applications. This justifies the use of energy harvesters as a supplementary source of power to decrease the generation costs in power plants, fossil based systems and carbon emission [24, 82, 82–84].

We take the wind turbines as the main power generation source in this chapter. The generated power by the turbines supply smart grid networks. The use of first wind turbine goes back to 1887. The invented turbine could be able to generate a power of 12 kilowatts (KW) [24, 85, 86]. Since that date, advancements in technology allowed engineers to achieve higher wind to electrical conversion efficiency and lower cost per kilowatt that led to greater power generation [24].

Cyber-security is one of the essentials for guaranteeing the reliability of the smart grids [24]. False data injection (FDI) is the censorious among attainable cyber-attacks [24, 41]. Adversaries can launch these attacks by compromising smart meters to introduce malicious measurements.

If these measurements change the result of the state estimation, they can deviate the control algorithms of power grid that may result in fatal consequences such as blackouts in large geographic areas [24]. Therefore, attack detection is the most essential step for minimizing the damages resulting from the FDI. The efficiency and effectiveness of FDI detection can have a significant impact on the overall performance of smart grids. Feedforward neural networks have been applied on FDI detection but they did not yield good results because the spatio-temporal correlation of data is not considered in training [24, 41].

On the other hand, it is found in [24, 87] that recurrent neural networks (RNNs) are capable

of exploiting the underlying correlation within the data. It was shown that under fairly mild and general assumptions, RNNs are universal approximations of dynamic systems. However, training a fully connected RNN in many cases is very difficult or even impossible [24,88]. Due to the difficulty of training traditional RNNs, reservoir computing (RC) recently attracted a lot of attention due to its simple training methods [89,90]. Liquid state machine (LSM) [87] and echo state networks (ESN) [91] are two most popular RC systems. The difference between LSM and ESN is that, LSM uses spiking trains as the input which has to be encoded by temporal or other encoding schemes, on the other hand ESN deals with regular data that is not a spike [24, 87, 91]. In general, a typical RC system is composed of three different layers: the input layer, the reservoir, and the readout/output layer. The reservoir is mainly composed of randomly connected neurons where the weights of the connections between neurons stay unchanged during the training [24]. The readout/output layer uses a linear combination of the reservoirs to produce the desired output [91, 92]. It has been shown in [24, 91, 93] that RC systems achieve better performance than traditional RNNs in many applications.

It is observed that delayed feedback networks (DFNs) are also capable of acting as RC systems [24, 94]. The set of sparsely connected neurons (reservoirs) in LSM and ESN are replaced by a nonlinear node. This approach not only simplifies the structure of RC systems but also demonstrates a very significant computational efficiency [24, 94]. The parallelism that exists in many other structures of artificial neural networks may simply be changed by a nonlinear node in which the input is inserted into that node

Several schemes have been introduced to encode the neural information. Rate encoding and temporal encoding are the two most popular ones [2]. In rate encoding, a code consists of a number of spikes occurring in a time frame after the stimulus appears [95]. Temporal encoding is subdivided into three main groups: latency code, interspike intervals, and phase

of firing [96]. In latency code, the time in which the first spike occurs is used for encoding [95]. Interspike interval coding is another scheme that uses the intervals between different spikes for encoding [96, 97]. In the temporal encoding using phase of firing, the phase of the local field power (LFP) is used to encode the information [98]. Studies show that interspike interval encoding carries more information than rate encoding [99, 100]. Therefore, in this chapter, we use interspike interval temporal encoding as the encoder of our RC systems.

Equipped with the platform of analog spiking RC architecture, we will be able to conduct anomaly detection in cyber physical systems (CPS) efficiently and effectively using RC. To be specific, in this chapter, we show that by using DFNs and MLPs it is possible to efficiently and effectively detect attacks in smart grids. Compared to existing attack detection algorithms in smart grids, our introduced design shows a great deal of robustness with respect to various attack variations. The main contributions of our work are the following:

- First, to the best of our knowledge, this is the first work to introduce the concept of reservoir computing for attack detection in smart grids. It is shown through simulations that the RC-based attack detection performs better than existing approaches. Furthermore, the accuracy of the attack detection of the RC-based approach is insensitive to attack variations such as the magnitude of the attack and the number of compromised meters.

- Second, we modify the delayed feedback network so that it is able to take spike trains as the input. Note that spike encoding is more biologically plausible and very similar to the way that information is encoded in our brains. Several modifications are conducted on the existing DFN architecture in the literature: 1) A block is added to convert the spike train into analog signals before the nonlinear node and in the feedback loop. 2) The leaky-integrate and fire (LIF) neuron model is introduced as the nonlinear node in the DFN tailoring towards the input spike train.

- Third, a multi-layer preceptron is introduced as the readout layer that can deal with both non-linear data and classification tasks.

We will show that the average attack detection rate based on the accuracy metric for 10000 simulations will be above 99% . This chapter is organized as follows. Section 2.2 reviews the related works in smart grid security. Section 2.3 the proposed design will be described; in Section 2.4 the simulation results are presented and compares the results of the proposed algorithm with current existing methods and we will discuss why our proposed method outperforms the other methods in literature. Section 2.5 concludes the chapter.

2.2 Related Work

FDI problem in smart grids was first introduced in [101]. In [102] a summary of all the proposed methods for FDI detection and the advantages and disadvantages of each methods is presented. Tan *et al.* [103] present a survey of the recent data driven approaches in smart grid security. So far, many algorithms have been introduced for FDI in smart grids. Within these methods, the state vector estimation [101] is among the first introduced algorithms. Machine learning techniques have also been introduced to FDI detection of smart grids. To be specific, feedforward neural network, K-nearest neighbor, support vector machines, and sparse logistic regression have been applied to FDI detection recently [41]. However, most of these techniques rely on manually chosen meta-parameters/parameters for the corresponding model. Even though the feedforward neural network allows for certain autonomy, its performance is usually strictly suboptimal when dealing with correlated data. Machine learning approaches show better results than support vector estimation methods when applied on IEEE test systems [104]. The effectiveness of the Precision Measurement Units (PMUs) have been extensively investigated in order to improve the performance of state

vector estimation [105] , [106]. Extended distributed state estimation (EDSE) was studied by Cramer *et al.* [107]. EDSE uses graph partition algorithms to divide each power system to several subsystems and in each subsystem three main categories are considered for the buses: boundary bus, internal bus and adjacent bus. EDSE-based methods show better performance than the traditional state estimation methods. In [108] the compromised nodes are detected through the analysis of the existing relationship between the physical properties of the power system and FDI.

2.3 RC Design for Attack Detection in Smart Grids

2.3.1 Realizing RC using DFN

Traditional RC models, including ESN and LSM, are different from DFN in the reservoir layer [92]. The DFN consists of a nonlinear node. The output of the node, also called the state, is a shifted version in time which provides the states of other nodes, called the virtual nodes [109]. The structure of the DFN, used in this work, is shown in Figure 2.1. The temporal encoder of [2] is used as the first, i.e. the input, layer. The details of the temporal encoder is described in Section 2.3.2. We employ 10000 vectors as the measurements generated using MATPOWER 5.1 [110]. A random Gaussian vector, with a variance of 0.05, is used for simulating an attack on half of the measurements. We aggregate the combined attacked and non-attacked data in a single vector and apply the temporal encoder on the data. Further, the corresponding spiking train is generated for every sample in the vector. This procedure allows us to convert the measurement matrix, extracted from 57 buses, to the corresponding temporal code. Since we use several meters on the same bus, size of the measurement matrix, generated by MATPOWER, is equal to 137. Next, we apply the generated spikes on the nonlinear node of the DFN.

In order to implement spiking neural networks, we employ a leaky-integrate and fire (LIF) neuron as the input node of the reservoir layer [111]. The generated spikes are then converted to an analog current before feeding into the LIF neuron. In this way, for every analog current at the input of the LIF neuron, a corresponding spike train is generated.

Almost any system with dynamics incorporate delays. As an example, information transfer in the brain, from one neuron to another, involves delays. Delay differential equations are the popular mathematical models to represent the delayed systems [112]. The dynamics of such a system depend on both the current states as well as the previous ones. The significant characteristics of dynamic systems include high dimensionality and short-term memory which constitute the prerequisites for an RC system [113].

Delayed feedback RC systems exhibit practically similar performance as the traditional RC systems [94, 114, 115]. However, the difference of the delayed feedback reservoir system from the traditional reservoir is in a single nonlinear node and a delay loop. A training algorithm is employed to optimized the output of the reservoir. The optimization objective is to minimize the difference between the weighted sum of the state and a target output value. The nonlinear node directly receives the input data. Further, a masking process employed to compensate for the loss of parallelism. In the masking process, the input signals in the transient regime are scaled [94]. The signals at the output of masking process are then transferred to the nonlinear node for implementing the nonlinear mapping. The only trainable weights of the system are the output weights, similar to a traditional RC system.

We employ an analog hardware implementation of the delayed feedback system which is able to process spike-based signals directly. It is shown that analog implementations has the advantage of implicit real-time operation which allows for smaller design area and lower power [116–125]. Our proposed analog implementation is inspired by the delayed feedback reservoir. In our design, the need for peripheral components, such as analog-to-digital (ADC)

and digital-to-analog converters (DAC), for interfacing with analog signals, is alleviated. The spike train generated by the LIF neuron is shifted in time by 10 milliseconds (ms) which is used as the state of the second node in the reservoir. The process is repeated four times to arrive at a different state. Furthermore, information is encoded before the corresponding signals are fed to the nonlinear node.

Most common encoding strategies used in the literature of spiking neural networks, include *rate encoding* and *temporal encoding*. In rate encoding, information is represented by the number of spikes while other spike characteristics, such as amplitude and phase, are irrelevant. A temporal encoding scheme encodes information into inter-spike intervals. Temporal encoding provides a compact model and energy efficient processing since the analog signals are encoded into spike based information. In this work, we employ temporal encoding with an iterative structure to process data. In this scheme, the number of neurons and the number of spikes have an exponential relationship. As a result, less neurons would be needed to achieve the same number of spikes.

In our temporal encoder, only one neuron operates in the dynamic mode which results in significant savings in power consumption [126, 127]. Our proposed encoding scheme has been fabricated using 180 nm CMOS technology and a symmetric layout to maximize the die area utilization. Not only our design features an internal verification technique, but also an output temporal code, which results in high error-tolerance via exploiting the additional inspection spikes. In addition to high accuracy, the proposed neuron also achieves low power consumption compared to the state-of-the art designs [128]. Our implemented neuron is able to extract five different states for every sample in the measurement matrix. The extracted states are then employed for training a multi-layer perceptron (MLP) neural network. We use the time instances of the spikes, corresponding to the state of every sample, as the features in training the MLP. We use the binary 0 and 1 labels, corresponding to the non-attacked

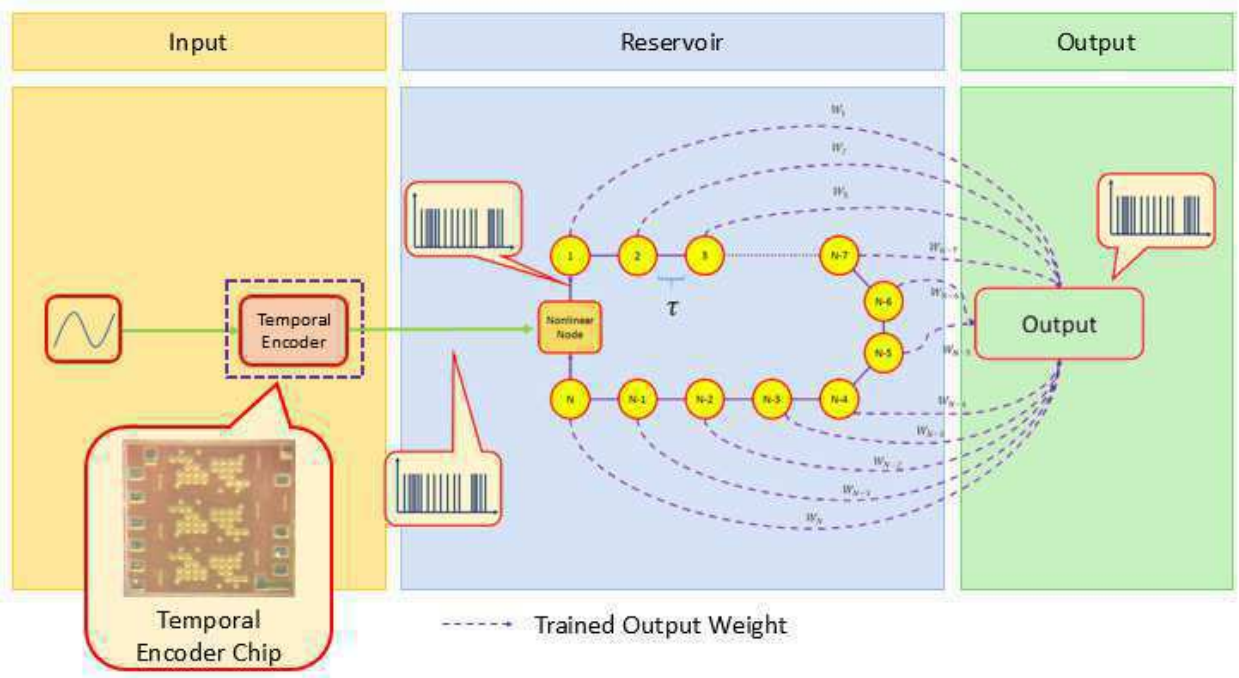

Figure 2.1. Hardware implementation of delayed feedback reservoir system [1]

and attacked samples, respectively, for training the reservoir. After training the MLP, the performance of the system is evaluated using the generated test data.

2.3.2 Temporal Encoder

The design of our temporal encoder is adapted based on the the encoder of [2]. The corresponding inter-spike intervals, for encoding data, is expressed as:

$$D_i = f\left(C_i, V_i\right) - f\left(C_{i-1}, V_{i-1}\right). \tag{2.1}$$

in which the function $f(X, Y)$ is:

$$f\left(C_i, V_i\right) = \left(C_{i+1}\right)\left[\beta\left(V_i - \gamma\right) + \theta\right], \tag{2.2}$$

In the above equations, the design parameters include the characteristics of the encoder, such as charging and refractory periods. More specifically, C_i and V_i represent membrane capacitance and the firing threshold voltage, respectively.

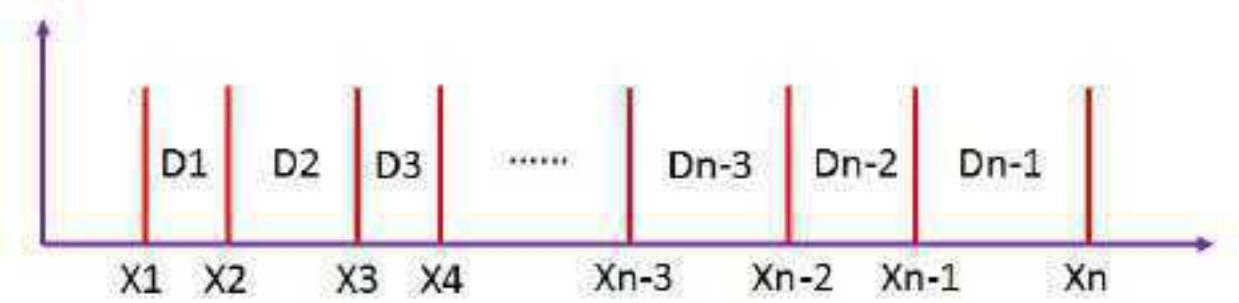

Figure 2.2. Interspike intervals [2].

Using the above temporal encoder, any sample in the measurement matrix is encoded in the inter-spike interval distances, D_i. There may be different number of intervals based on the number of neurons used in the temporal encoder for any sample. In this experiment, for the sake of simplicity, we choose the number of neurons in the encoder to be $N = 3$, which results in four different spikes, X_1 to X_4 or three intervals D_1 to D_3. Due to the following equation mentioned in [2], there is a relationship between the number of spikes produced and the number of neurons used in the temporal encoder:

$$S_N = 2^{N-1}, \tag{2.3}$$

where, S_N is the number of spikes produced by the temporal encoder and N is the number of neurons used in the encoder.

2.3.3 Smart Grid Attack Detection Formulation

Smart grids are used to make a reliable power transmission network and connection between consumers and generators. They are really vulnerable to cyber-attacks, and thus it is a very important and challenging task to provide a secure network of smart grids [129]. MATPOWER 5.1 can be used to produce the smart grids' measurement matrix [130]. MATPOWER allows the users to run the toolbox with different numbers of buses. In our experiment, the number of buses is set to 57 resulting in 137 different measurements. Note that it is pointed out in [41] that the parameter that really impacts smart grid attack detection is the number of compromised meters instead of the number of buses. The reason that we pick 57 is because this number is almost in the middle of the range of the number of the buses that is provided by MATPOWER. This configuration will result in 137 meters which is large enough for us to study the effect of different number of the attacked meters in that configuration [131].

The system model that is used to study the attack detection in smart grids is defined in [101]:

$$z = Hx + n. \tag{2.4}$$

The measurement vector which is the output of different meters on the buses is z; H is the state vector; x is the voltage phase of the buses; and n is the environment noise. When attack is present, an attack vector, a, is added to the measurement. Accordingly, the measurement, $\breve{z}$, becomes

$$\breve{z} = Hx + a + n. \tag{2.5}$$

We assume that the attack is a Gaussian random vector with 0.05 variance [131]. State vector estimation (SVE) is the first method introduced to perform attack detection for smart grids. This method consists of calculating a residual stated as ρ. If the value of ρ exceeds a

predefined threshold value, it is said that the z vector has been attacked and the meters are compromised [101].

$$\rho = \|\breve{z} - H\hat{x}\|_2^2, \tag{2.6}$$

where $\hat{x}$ is the vector estimated using SVE algorithm. $\hat{x}$ is then estimated as follows:

$$\hat{x} = (H^T \wedge H^{-1})H^T \wedge z. \tag{2.7}$$

The only action that has to be done in SVE algorithm is to estimate the $\hat{x}$, and to do so, $\wedge$ needs to be calculated, where $\wedge$ is defined as a diagonal matrix with its diagonal elements are the reciprocals of the variance of the measurements. For example, the j-th diagonal element of the $\wedge$ is equal to the reciprocal of the variance of the j-th element of the z. SVE method is a very simple method for implementation but it has many shortcomings with different attack situations [101].

In the case where $a = HC$, the attack is hidden (see Appendix A). It means that SVE is incapable of detecting the bad measurements [101]. In the case of hidden attacks, the residual value is less than the threshold and the attack cannot be detected by SVE. In order to perform a hidden attack, the cyber attacker has to have access to at least a specific number of attacks. In [101], it has been shown that it is not possible for the attacker to choose any arbitrary c and multiply it by H to perform the hidden attack. This means that in order to make a hidden attack, the attacker has to have access to at least k measurements, in which $k > m - n$ where m is the number of meters and n is the number of buses. In our system, $m = 137$ and $n = 57$. In this case, $m - n = 80$ meaning that with high chance the attack will be a hidden or stealth attack when there are more than 80 compromised measurements in our smart grid network. Under this scenario, the SVE becomes an inefficient attack detector.

2.3.4 Modeling the wind power generators in MATPOWER

MATPOWER can also be used to study renewable energy, especially wind powers [132]. There are six power generators for a 57-bus smart grid network. It is possible to substitute the power of these generators with the power obtained from wind power generators. Accordingly, (2.8) gives the power produced by a wind power generator as the source of energy [133]:

$$P_{\text{avail}} = 1/2\rho A v^3 C_P, \tag{2.8}$$

where P_{avail} is the power converted from wind; ρ is the air density which is assumed to be equal to 1.23 kg/m^3; A is the sweep area of the wind turbine blades; v is the speed of the wind; and C_P is the coefficient of the power. Albert Betz, a German physicist, has shown that the maximum value for the power coefficient is equal to 0.59. This is called Betz Limit or Betz Law. Based on that, the performance of a wind power generator cannot exceed 0.59 [134]. In this study the value of C_P is set to 0.4 while the area of the generator is set to 8495 m^2, six different values ranging from 0 to 12 m/s are used for the wind speed. These power values are inserted in the MATPOWER to produce H matrix [134].

2.3.5 Smart Grid Attack Detection using DFN and MLP

The FDI problem can also be formulated as a classification problem. So far, many machine learning algorithms have been suggested to deal with this problem [135]. To the best of our knowledge, this problem has never been studied from RC's point of view. We are the first to study this problem using RC methods. In the FDI problem we face two classes of data: attacked data and non attacked data, we can assign two different labels for these two classes and figure out the classification of data.

In this experiment, two different sets of data are used. The data which has been attacked by a hidden attack and the data which its measurements have been attacked by direct or non-hidden attack vectors. The experiments are performed on 1000 samples and the experiments are repeated 10 times. The first step is to encode z using the temporal encoder. Then, every spike train extracted from the temporal encoder is converted to an analog current. In [136], an equation was introduced to convert the spike trains to the analog current:

$$I^i = \sum_{t^j} K\left(t - t^j\right) \mathcal{H}\left(t - t^j\right), \tag{2.9}$$

where $\mathcal{H}$ is the Heaviside function; I^i is the analog current of the i-th sample in the z; and t^j is the time of occurrence of the j-th spike in the corresponding spike train of the i-th sample achieved from the temporal encoder [136]; and

$$K(t - t^j) = V_0 \cdot \left(\exp(-((t - t^j)/\tau_s)) - \exp(-((t - t^j)/\tau_f))\right), \tag{2.10}$$

where τ_s is set to 10 ms and τ_f to 2.5 ms. The values of τ_s and τ_f have to be chosen somehow that $\tau_s/\tau_f = 4$. V_0 is a normalization factor to make sure that the maximum value of kernel does not exceed one [136].

Up to now what we can generate analog current signals from (2.9) and (2.10) corresponding to the temporal codes extracted from the temporal encoder. The next step is to apply these current on the DFN to produce the corresponding states. As mentioned in Section 2.3.1, the nonlinear node of the DFN is chosen to be an LIF neuron. The analog current signals for the attacked samples and non-attacked samples were applied to the DFN. The output of the LIF neuron is shifted 10 ms in time to produce the state of the first virtual node. This process is repeated in four times until we obtain four virtual nodes. Note that the state of the fourth virtual node is shifted 40 ms compared to the nonlinear node. Then the state of the fourth

virtual node is multiplied by 0.8 which is the feedback gain, and is then added to the new incoming analog current. Now for both attacked and non-attacked samples, five different states are generated. Figure 2.3 shows the average total state of attacked and non-attacked samples.

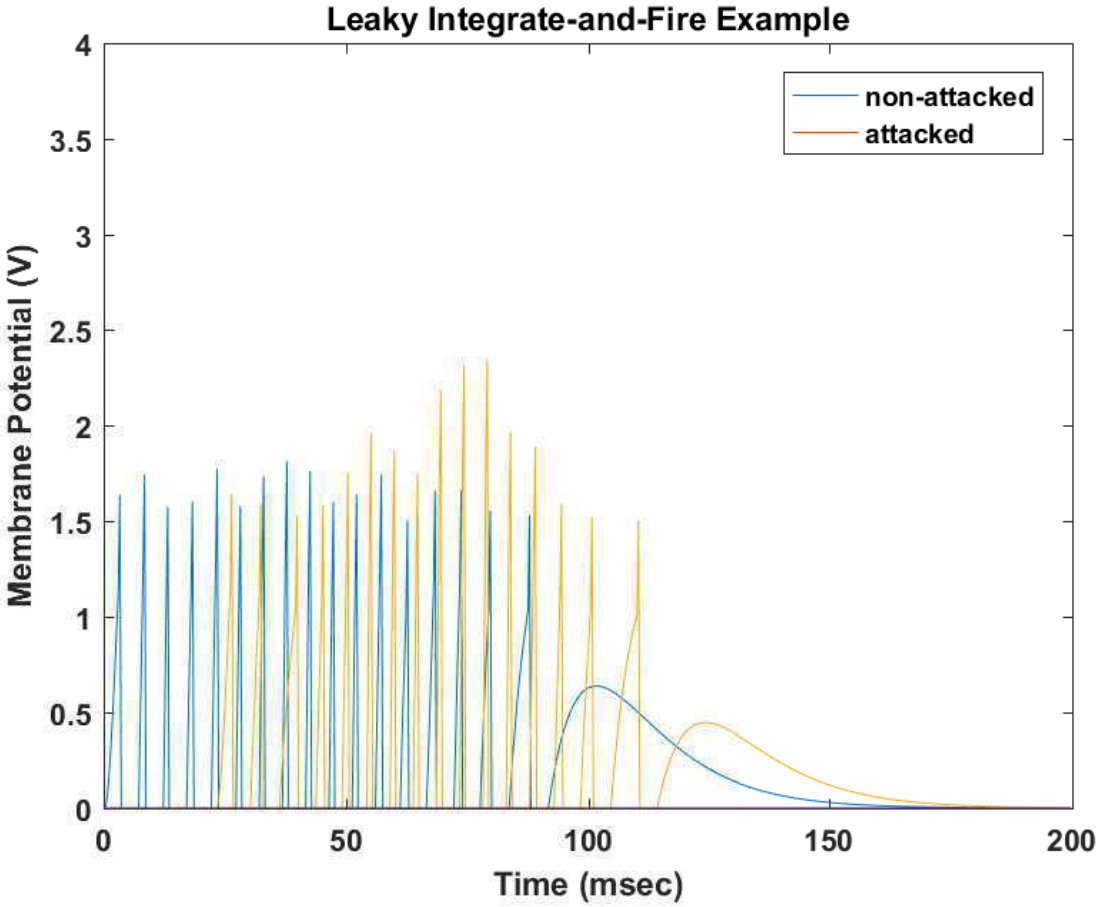

Figure 2.3. Average DFN states for attacked and non-Attacked Data.

From Figure 2.3, it is clear that the timing of the spikes for the average states of the two data classes are very different. It can be seen that average spikes produced for the attacked samples are more likely to fire at smaller times and the ones fired for non-attacked samples are more likely to fire at larger times. Furthermore, it is possible to use these timings as a feature to classify these two groups. Therefore, in the next step we utilize a MLP and train

the MLP with these features [137].

2.3.6 Training an MLP with the timing of spikes

As demonstrated in [94], the readout layer can be trained with a linear algorithm. In the introduced training algorithm in [94], a weight was assigned to the every state extracted from the DFN in a way that the desired output values can be estimated with the least possible error. The following expression provides a good summary of the training algorithm in [94]:

$$\hat{y}(k) = \sum_{i=1}^{N} w_i \times x\left[k\tau - \tau/N(N-i)\right], \tag{2.11}$$

where $\hat{y}(k)$ is the estimated output; w_i is the connection weight; x is the state vector; and N is the number of states. The above algorithm is linear and not iterative so it won't be very precise. Therefore, in our RC-based attack detector, we adopt an MLP for output estimation. The algorithm used for training the MLP is backpropagation. The label of y is set to 1 for training the samples being attacked and 0 for the samples not being attacked. The time in which attacks spikes happen for different states are saved in a vector and are used as features for training the MLP. The MLP is trained with two different hidden layers and one output layer. The desired output for the attacked sample is 1 otherwise it is 0. As it can be seen in Figure. 2.4 the MLP is trained after 82 iterations. Then the weights are saved to be applied on the test data to evaluate the performance of the system. In the next step, SVE algorithm, MLP, and Support Vector Machines (SVM) are applied on the samples to compare against the performance of our introduced RC-based attack detection strategy. We use Gaussian radial basis function as the kernel of the SVM classifier (see Appendix B for details) [131].

As it can be seen in Figure. 2.4, the training mean square error (MSE) reduces to 0.016

which indicates that MLP is capable to distinguish between the states spike timings of the attacked data and the non-attacked data. The learning rate is set to 0.01 and momentum factor to 0.5. If the output value is greater than 0.5, it is considered as 1, else it is considered as 0. In that sense, the training accuracy is almost 100%, meaning that almost 100% of the samples are classified accurately as attacked and non-attacked. In order to quantify the detection performance, the **accuracy** metric is defined in equation (2.12). We used 50% of the samples for training and the rest are saved for testing and validation. The results of applying the DFN algorithm are presented in Section 2.4. Figure 2.6 shows the block diagram of our RC-based attack detection algorithm.

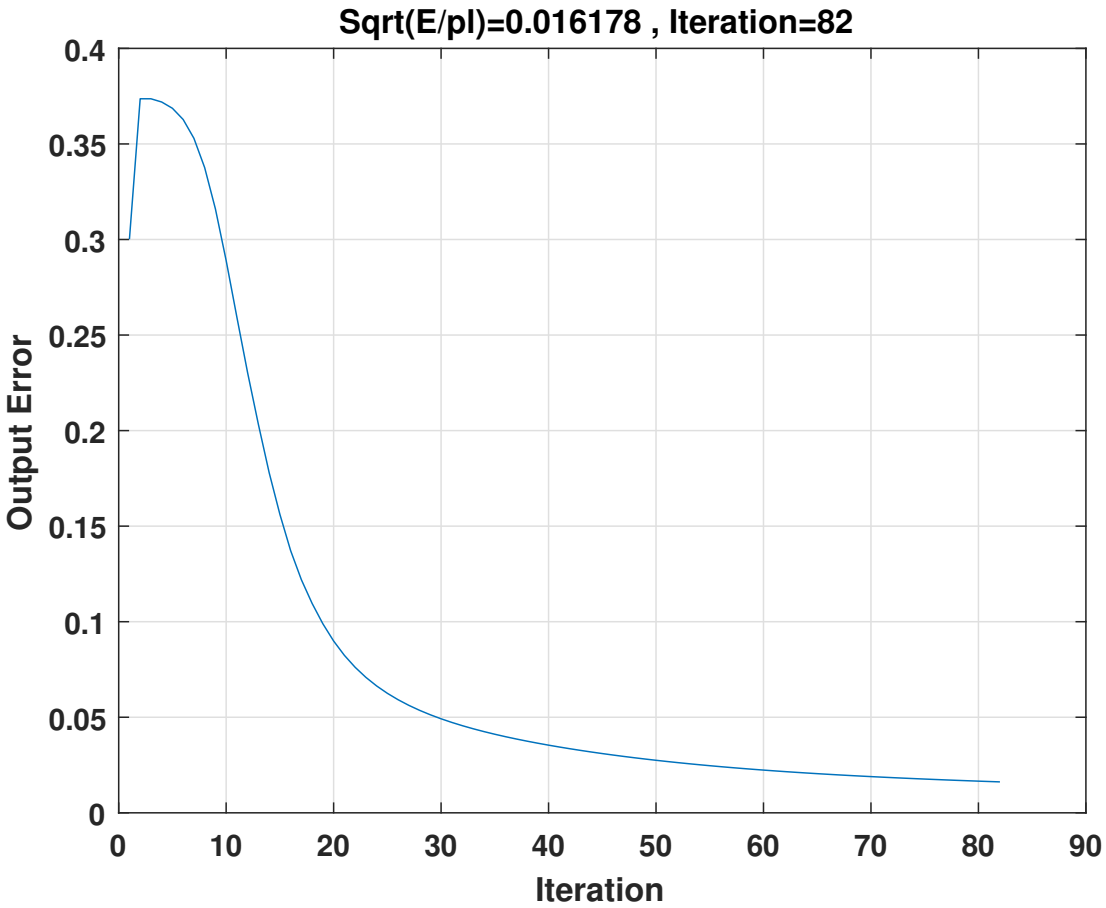

Figure 2.4. Error Plot for training an MLP with DFN states.

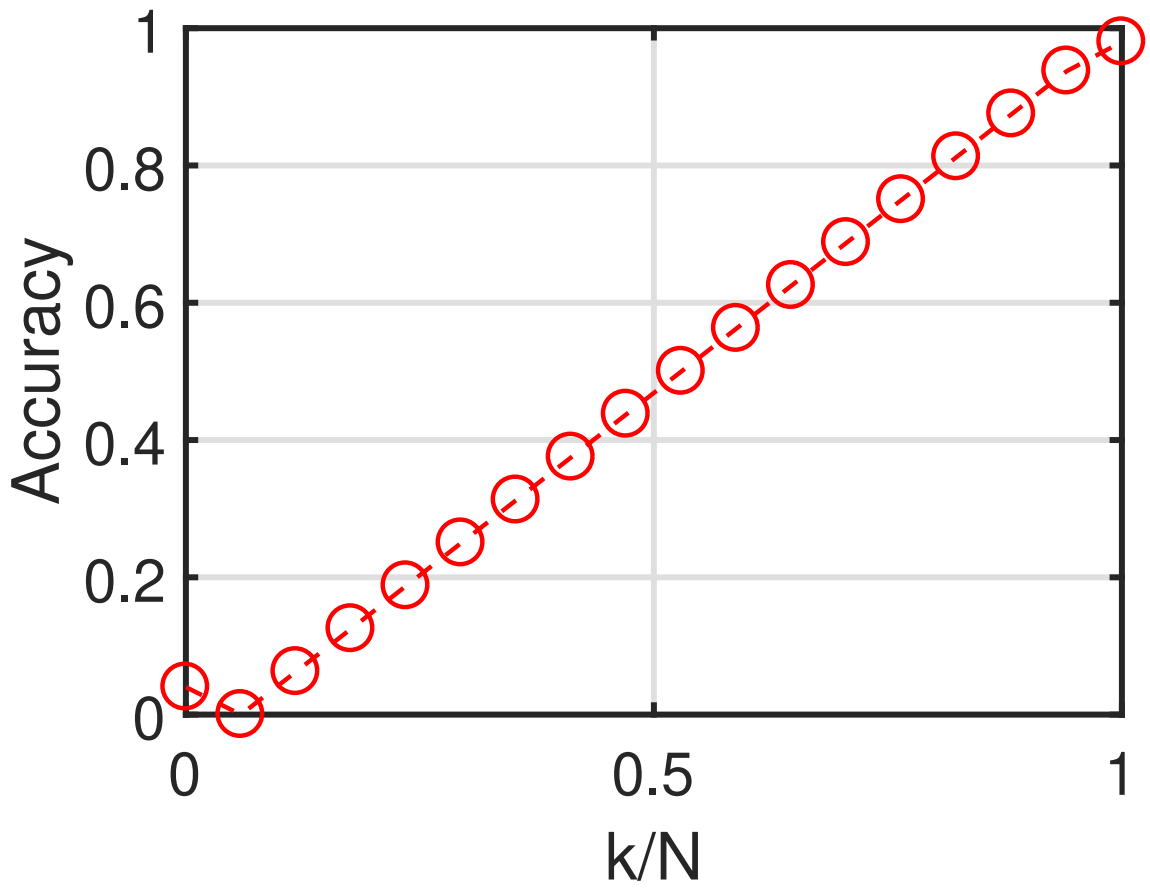

Figure 2.5. Accuracy of the SVE.

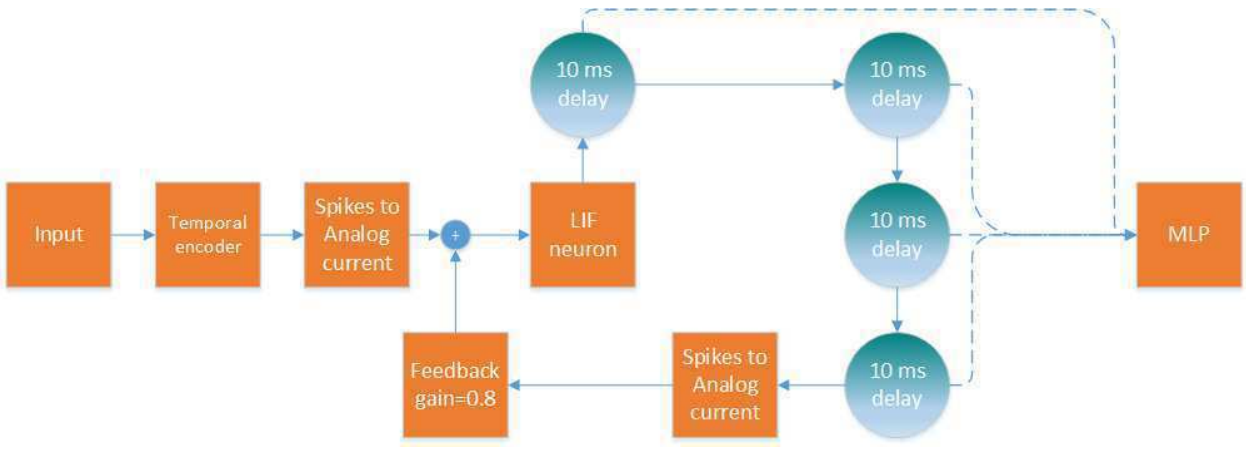

Figure 2.6. Block diagram of the proposed DFN+MLP system for attack detection.

2.3.7 State Vector Estimation

As mentioned in Section 2.3.3, $\rho = \|\breve{z} - H\hat{x}\|_2^2$ needs to be computed for SVE. If the value of ρ exceeds a predefined threshold value, it is said that an attack has occurred, non-attack is detected otherwise [101]. Accordingly, we can calculate the value of ρ when the measurement vector is attacked by the same attack vector mentioned in the previous section. The value of ρ achieved for attacked vectors with different number of compromised measurements is used to evaluate the performance of SVE. However, we show in the next section that there are some drawbacks with SVE. The performance metric used to evaluate the detection performance is the **accuracy** which is defined as

$$\text{Accuracy} = (\text{TP} + \text{TN}) \,/\, (\text{TP} + \text{TN} + \text{FP} + \text{FN})\,, \tag{2.12}$$

where TP, TN, FP and FN correspond to the number of true positive, true negative, false positive and false negative samples respectively.

The attack detection performance of SVE can be clearly seen in Figure 2.5. As seen in the figure, the accuracy of SVE is severely affected by the number of compromised measurements even when the attack is not hidden. This is due to the fact that the performance of SVE depends heavily on the residual value. When the number of compromised measurements is small, the accuracy of the SVE drops significantly. In Section 2.4, we will show that this issue can be completely resolved by the introduced RC-based DFN+MLP attack detector.

2.4 Performance Evaluation

As it was mentioned in Section 2.3.3 only 50% of the data is used for training and the rest is saved for test and validation. In this section, we will detail the performance evaluation of

the three aforementioned algorithms: RC-based method (DFN+MLP), MLP, and SVM. We have totally 5000 samples for testing and validation. Half of them are attacked and half of them are not. As the main evaluation results, Figures. 2.7 & 2.8 show the accuracy of the proposed method for the two types of attacks in smart grids, hidden and direct, as a function of the attack magnitude a. Three different values of the attack magnitude are used: $a = 0.1$, $a = 1$, and $a = 10$. Note that since SVE is not capable of detecting hidden attacks [101], we did not evaluate its performance in Figures. 2.7 & 2.8.

From the figures, we can clearly observe that the performance of both MLP and SVM are very sensitive to attack magnitudes as well as the number of attacked meters. Unlike SVE, both MLP and SVM can detect hidden attacks. However, their detection performances are very sensitive to attack parameters. For example, the accuracy of both MLP and SVM increases as the attack magnitude increases. This means that MLP and SVM can detect attacks accurately when attacks have large magnitudes. However, when attacks have small magnitudes, MLP and SVM will detect attacks with less certainty. To be specific, for the case of MLP, the accuracy is 100% when the magnitude of the attack is 10 and can be as low as 70% when the attack magnitude is 0.1. This is not very desirable for attack detection in smart grids where the attack magnitude can be arbitrary. For the RC-based DFN+MLP method, we can see that the variations of attack magnitude do not cause any significant change to the accuracy. To be specific, the accuracy variation due to the change in attack magnitude is very small for RC-based approach and the accuracy is close to 100% in all attack magnitudes. This clearly suggests that the attack detection performance of the RC-based approach is robust under different attack magnitudes. Figures. 2.7 & 2.8 also show the accuracy as a function of the number of compromised meters for different attack detection strategies. SVE is not capable of detecting hidden attacks, therefore, we did not evaluate its performance in Figures. 2.7 & 2.8. From the figures we can see that the introduced RC-based approach is much more robust than the MLP and the SVM method

under different number of compromised meters. Furthermore, comparing the two figures, we can observe that unlike existing detection strategies (SVE, MLP, and SVM) the RC-based DFN+MLP method provides uniform performance under different attack methods (direct and hidden). In this study 50 % of the samples are attacked and the rest not, which means we are dealing with a balanced data set and if the number of attacked and non-attacked samples are significantly different the data set is imbalanced [131]. The imbalanced data set is very likely to compromise the performance of the learning algorithm [138]. In such scenarios *F1* score is used to evaluate the performance of the learning algorithm [131, 139]. *F1* measure can handle the imbalanced data. In [131] the detection performance evaluation is studied for both balanced and imbalanced data set extracted from IEEE 30-bus system.

$$\mathrm{F1} = (2\mathrm{TP}) \,/\, (2\mathrm{TP} + \mathrm{FP} + \mathrm{FN})\,, \tag{2.13}$$

In that study [131] the performance plots , accuracy for balanced data set and *F1* measure for the imbalanced data set, do not show any meaningful difference.

2.5 Conclusion

In this chapter, we introduced a RC-based (DFN+MLP) attack detection strategy for smart grids.The introduced method constitutes of three main steps. The first step is encoding the measurement vector with temporal encoder and converting the produced spikes to their corresponding analog currents. In the second step, these analog currents are applied on an LIF neuron and shifted in time to produce the sates of virtual nodes. The output of the fourth virtual node is multiplied by a feedback gain and added to the new incoming data in order to preserve the recurrent nature of the DFN. The spiking times of these states are used to train an MLP for classification. Simulation results have shown that this algorithm can

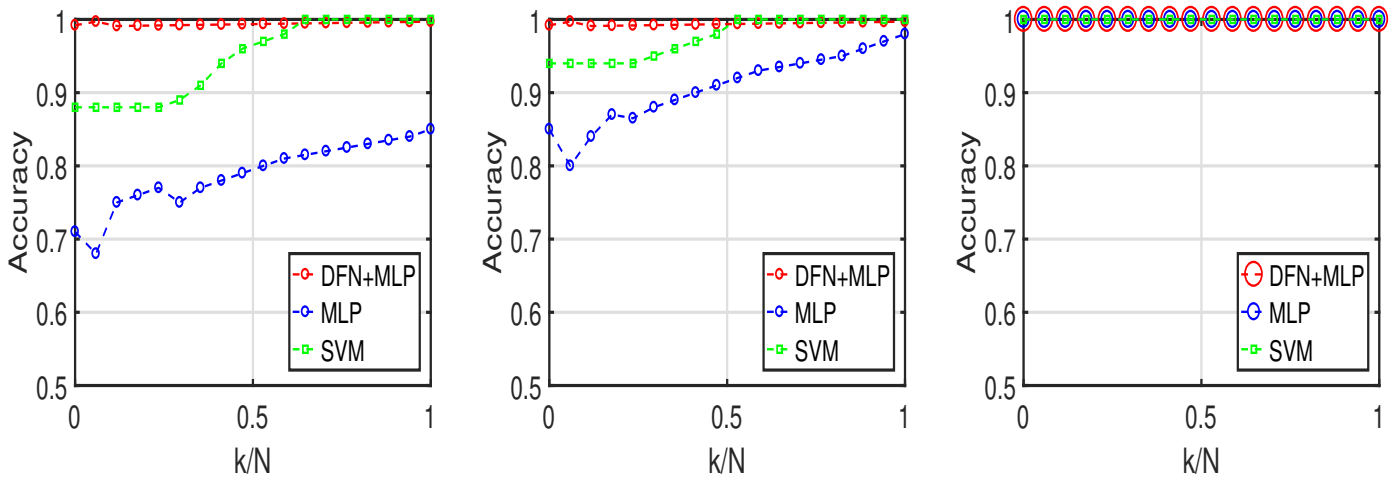

Figure 2.7. Accuracy of direct attack detection for three different methods, a=0.1,1,10.

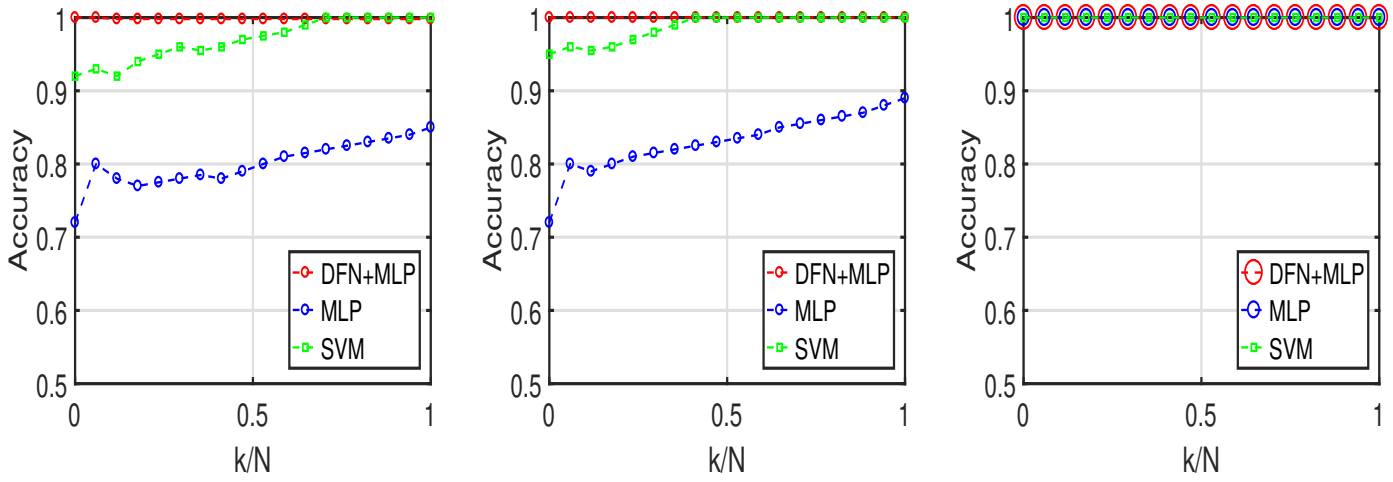

Figure 2.8. Accuracy of hidden attack detection for three different methods, a=0.1,1,10.

robustly detect attacks under different attack variations such as magnitudes and the number of compromised meters compared to existing methods such as SVE, MLP, and SVM. It is also important to note that this work is the first effort to solve FDI problems in smart grids through RC. The proposed model can be applied on any classification task that there is spatio-temporal correlation between the samples of the data set. In our next work we will show that we have been able to apply this model successfully on face recognition task from video frames. Since there are spatio-temporal correlations among the meters in smart grids, RC-based attack detection can take full advantage of this spatio-temporal correlation yielding a better performance compared to existing solutions.

Chapter 3

Detecting Dynamic Attacks in Smart Grids using Reservoir Computing: A Spiking Delayed Feedback Reservoir-Based Approach

Smart grids are power grids that have been modernized for intelligent transmission and distribution that improve the system reliability, security, and efficiency. However, smart grids introduce new vulnerabilities in security unless properly controlled. Cyber-attacks or incidents arise from various sources with different motivations, ranging from pranks to terrorism. With the enormous amount of data that is constantly flowing through the network, it becomes a challenging task for analysts to monitor the extensive security-related information that is being exchanged for anomaly detection [140].

The false data injection (FDI) problem in smart grids was first introduced in [141]. In general, there are two main categories of FDI attacks in smart grids: single-period or opportunistic

attacks and multi-period or dynamic attacks [142]. In single-period scenario, attacks are performed simultaneously, and the attacker waits until there is an opportunity to perform the attack with a high chance of success. Most of the existing work on smart grid cyber security focused on this case. In dynamic attack scenarios, the adversary manipulates the state of the network gradually and through time toward the desired state. The dynamic attacks are generally more difficult to detect because the variations to the normal state take place gradually and slowly. A comprehensive summary of existing FDI detection algorithms, and the pros and cons of each algorithm were presented in [143]. State vector estimation (SVE) was the first algorithm that was proposed to tackle the FDI problem in smart grids [141].

Machine learning based approaches has also been introduced for FDI detection in smart grids [144] where artificial neural networks (ANNs), support vector machines (SVMs), and k-nearest neighbor (KNN) are investigated for this purpose. In general, machine learning based approaches have shown better performance than SVE in detecting the FDI in smart grids. Both supervised and unsupervised learning algorithms are utilized to detect the stealth data injections in smart grids [42]. In [145], the performance of SVM and KNN in detecting the false data in an IEEE-30 bus system under balanced and imbalanced data scenarios are studied. A deep learning-based method is exploited in In [146] to capture the behavior of the FDI attacks features. These features are used to detect the FDI in real-time with high accuracy. Unfortunately, all these work focus on the single-period attack and neglect the spatio-temporal correlation of the measurement in smart grids.

In our previous work in [24], a reservoir computing (RC) based approach is introduced to detect the FDI taking advantage of the underlying spatio-temporal correlation. RC is a class of recurrent neural networks (RNNs) that is more easily trained compared with the traditional RNNs [20]. The two well-known RC models, echo state network (ESN) and liquid state machine (LSM), employ the strength of RNN as their reservoir or liquid in

which the synaptic connection within these layers are not trained [20]. Recently, another RC model, called the delayed feedback reservoir computing (DFR), is constructed through a single nonlinear neural node with dynamic delay loop. It is shown in [147, 148] that DFR has improved performance compared to other RC models. Our previous work in [24] utilizes DFR to conduct single-period attack detection. In this chapter, we will extend the work to the challenging case of dynamic attack.

There are several different mathematical models for artificial neurons in a neural network. Spiking neural networks (SNNs) are one example of these models that have been highly used during past decades to solve pattern recognition related problems. Spikes are thought to be the main signal format that neurons use in the brain to communicate with each other. Therefore, SNNs are more biologically plausible compared with traditional artificial neural networks (ANNs) [136]. The other advantage of SNNs is that SNNs are much more energy efficient and are better for hardware implementations [149].

Several studies have shown that the artificial SNNs can be very energy efficient as they are using spiking models of neurons. For example, the TrueNorth chip which is capable of running 1 million neurons with 256 million synapses consumes only 70 milliWatts (mW) [149]. This makes SNN a suitable choice for hardware implementations of artificial neurons.

On the other hand, the approach that neuron is employed to encode the information is still unclear and deserves further investigations. There have been various encoding schemes, which can be categorized into two major classes, namely, rate encoding and temporal encoding [150]. Rate coding schemes used to be more popular in the literature. However, recent research works have shown the superiority of temporal encoding schemes over rate encoding schemes [151]. In the temporal code the exact time of the spike is used for encoding the neural information. Several experimental results have shown that the exact time of spikes is used by neurons to encode and convey the information. For example, the neurons in the

retina, the lateral geniculate nucleus and the visual cortex respond to the stimuli with milliseconds (ms) precision. The temporal encoding approaches are also more computationally efficient than rate encoding approaches as it has been studied in [150].

In [136] an algorithm called precise-spike-driven (PSD) synaptic plasticity was introduced to learn the hetero-association which exists in spatio-temporal spike patterns. Synaptic plasticity is a concept in neuroscience and computer science where the strength or the amplitude of the weights that connect neurons together can be adjusted. Essentially, synaptic plasticity specifies the influence of the activation or firing of one neuron on other neurons. The PSD synaptic plasticity is a rule that is used to adjust the weights of a SNN in order to learn the spatio-temporal patterns. Different from most of supervised SNNs' training algorithms that employ rate codes, PSD uses temporal codes as the encoding scheme.

In this chapter, the PSD algorithm is introduced to train SNNs to detect the multi-period attack in a smart grid and find the optimal encoding scheme of spikes. Multi-Period or dynamic attacks are a special type of attacks, where the adversary manipulates the state of the smart grid network gradually and through time toward the desired state. The optimal encoding scheme that is identified in the training of the PSD algorithm will be used for training the spiking DFR+MLP.

The main contributions of this chapter are of the following,

1. Dynamic attack detection in smart grids are studied for the first time using recurrent neural networks. Furthermore, our work explores SNNs based on the DFR structure to effectively capture the spatio-temporal association existing among spikes patterns.

2. Several encoding schemes including latency, latency-phase and ISI are investigated to train the SNN using the PSD algorithm. Our results suggest that the ISI encoding leads to the least training error and the best attack detection performance.

3. Spiking neurons are integrated with DFR to take the advantage of both methods. DFRs can capture the spatio-temporal correlation between different components of the smart grid and map the data to a higher dimensional space to make it easier for the MLP to learn the spatio-temporal patterns of attack.

4. The effects of different delay values in DFR on the attack detection performance are investigated and the chaos behavior of the introduced algorithm is analyzed. Our results suggest that a dynamic delay system has to work at the edge of chaos in order to show high-dimensional behavior and delay is the parameter that specifies the level of chaotic behavior of such a dynamic system.

The rest of the chapter is organized in the following: Section 3.1 presents different types of attacks in smart grids and corresponding mathematical modelings. Section 3.2 describes the PSD algorithm and its training procedure. The RC approaches for high dimensional mapping of data with the focus on DFR is discussed in Section 3.3. In Section 3.4, training methods for the SNNs, DFRs, and MLPs using different temporal encoding schemes are presented under dynamic attacks. Section 3.5 presents performance results and provides intuitions behind the results. Section 3.6 contains the conclusion. Table 3.1 provides the list of acronyms used in the chapter.

3.1 Problem Formulation

There are potentially two different targets to attack in smart grid systems, the state of measurements and the topology of the smart grid network [142]. The measurement of each meter in a smart grid system is determined by the state of the system, a linear function and

Table 3.1: Acronyms and Their Descriptions

Acronym	Description
RC	Reservoir Computing
RNN	Recurrent Neural Network
ESN	Echo State Network
LSM	Liquid State Machine
ESN	Echo State Network
DFR	Delayed Feedback Reservoir
SNN	Spiking Neural Network
ANN	Artificial Neural Network
PSD	Precise Spike Driven
MLP	Multi Layer Perceptron
ISI	Inter Spike Interval
FDI	False Data Injection
SVE	State Vector Estimation
SVM	Support Vector Machine
KNN	K-Nearest Neighbor
SCADA	Supervisory Control and Data Acquisition
LIF	Leaky-Integrate-and-Fire
WH	Widrow-Huff
PSC	Post Synaptic Current
TTFS	Time to First Spike
SMO	Subthreshold Membrane Potential Oscillation
FLOP	Floating-Point Operations

the environment noise, and can be expressed as follows,

$$z = Hx + n, \tag{3.1}$$

where z is the measurement vector consisting the real parts of the line flows and bus injections; H is a linear function; x is the state vector, and n is the environment noise [142]. In the case where an attack exists, equation 3.1 can be written as,

$$\begin{aligned} \check{z} &= z + a, \\ \check{z} &= Hx + n + a, \end{aligned} \tag{3.2}$$

where a is the attack vector. The attack mentioned in equation 3.2 is called the observable attack. It is possible for the adversaries to perform another type of attack which is called the unobservable or hidden attack that is more difficultly detected especially for traditional attack detection methods such as SVE [145, 152–154]. In hidden attacks, it is assumed that the adversary has access to the H matrix, and the adversary can hide its desired state in the H matrix to manipulate the measurement vector z. The hidden attacks are more challenging to detect as they are unobservable by the Supervisory Control and Data Acquisition (SCADA) center of the smart grid. In this chapter, we consider hidden dynamic attacks. The hidden attack is defined as $a = Hc$, and equation 3.2 is reformulated as follows,

$$\begin{aligned} \check{z} &= Hx + n + Hc \\ \check{z} &= H(x + c) + n, \end{aligned} \tag{3.3}$$

where c is the desired state of the adversary, where the attacker wants to drift the normal state of the smart grid toward its desired state by hiding it in the H matrix.

Dynamic attacks are performed in a way that the state of the smart grid system will gradually be manipulated toward the state desired by the adversary [142]. In order to design this attack, the attack vector has to be defined as a function of time, because unlike single-period attack that the adversary performs the attack opportunistically, in dynamic attacks the attack is performed gradually and through time. The dependency of the dynamic attack on time makes the magnitude of the attacks variations lower, and are more challenging to be detected. In single-period attacks the variations of the attacks magnitude are sudden and abrupt, and are more easily detected. The formulation of dynamic attack used in this chapter is as follows,

$$\check{z}(t) = Hx(t) + n + a(t). \tag{3.4}$$

The dynamic attack $a(t)$ is time dependent, and we also assume that the adversary has access to H matrix. Thus the attack can be performed as hidden or unobservable. In hidden attacks the attack $a(t)$ can be expressed as $a(t) = Hc(t)$, and $c(t)$ is defined as follows,

$$c(t) = Acos(2\pi f_c t) \times N(0,1), \tag{3.5}$$

where A is the magnitude of attack, cos is the cosine function, f_c is the frequency of the attack and is set to 1 in this study, in order to perform a dynamic attack that can gradually manipulate the state of smart grid, $N(0,1)$ is a normally distributed vector with zero mean and variance of 1. Substituting (5) into (4) gives us the following equation,

$$\check{z}(t) = H(x + Acos(2\pi f_c t) \times N(0,1)) + n. \tag{3.6}$$

The motivation of this chapter, is to detect the presence or absence of hidden dynamic attacks on smart grids meters. To detect this, we need to have smart grid measurements data that can be compromised by hidden dynamic attacks. In [155] providing that the adversary is aware of the attack detector's parameters, the attack detection probability is minimized. However, in this study we assume that the adversary is not aware of the attack detector's parameters which in our case are the parameters of the DFR and MLP. Therefore, the adversary cannot perform any optimization on the attack before launching it to minimize the attack detection probability. However, the adversary can adjust the magnitude of the attack to make its detection probability smaller. As we will show in Section 3.5, while the attack magnitude is lower, the attack detection is more challenging and vice versa.

The dynamic attack model that we use to compromise the smart grid's measurements is expressed in (6). The MATPOWER toolbox and IEEE 14-bus test system are used for the simulation of the smart grid's measurements [110]. In the IEEE 14-bus smart grid test bed,

there are totally 34 different meters that can be compromised by the adversary. Our model assumes that the amount of access the adversary has to the system can vary. In our model the number of meters in the system that the adversary is able to compromise ranges from 0 to 34. We generate totally 10000 samples for training our model, and 10000 samples for testing and validation, using the MATPOWER toolbox and IEEE 14-bus test bed.

3.2 Precise spike driven synaptic plasticity

3.2.1 Neuron Model

The PSD algorithm is capable of learning the spatio-temporal correlation that exists in different data sets. There exists a significant spatio-temporal correlation among the different components of a smart grid [141]. Therefore, PSD is a good candidate for extracting this spatio-temporal correlation and using it to facilitate the FDI attack detection. In this section, we discuss the neuron model, the PSD learning rule, and its training procedure. So far, several models for spiking neurons have been proposed in order to mimic the behavior of biological neurons. The Leaky-Integrate-and-Fire (LIF) and the Hodgkin-Huxley are two well-known models for artificial spiking neurons which are used commonly [156]. The LIF neuron has been used more often than other models due to its simplicity and ease of hardware implementation [151]. When a stimulating current is applied on a neuron, the neuron starts to fire as soon as its membrane voltage exceeds a certain threshold. In the LIF neuron, the relationship between the stimulus and the membrane voltage is expressed as,

$$\tau_m \frac{dv_m}{dt} = -(V_m - E) + (I_{noise} + I_s)R_m, \tag{3.7}$$

where V_m is the membrane voltage, $\tau_m = R_m C_m$ is the time constant of the neuron, R_m and C_m correspond to the resistance and the capacitance of the membrane respectively, E represents the resting voltage, I_{noise} is the background noise and I_s is the presynaptic or stimulus current [136]. R_m is set to 1 mega *ohms* and C_m= 10nF.

Equation 3.9 formulates the relationship between the voltage membrane and the presynaptic current. Equation 3.10 formulates the relationship between the postsynaptic current produced for each LIF neuron and presynaptic currents. The presynaptic current of a neuron is a weighted summation of postsynaptic currents for afferent neurons [136, 157].

$$I_s(t) = \sum_{j=1} w_i I^i_{PSC}(t), \tag{3.8}$$

where w_i is the weight of the current coming from the i-th afferent neuron and; I^i_{PSC} corresponds to the postsynaptic current of the i-th afferent neuron [136, 157]. It is shown in [136] that $I^i_{PSC}(t)$ can be presented as below,

$$I^i_{PSC}(t) = \sum_{t^j} K(t - t^j) H(t - t^j), \tag{3.9}$$

where H corresponds to the Heaviside function, t^j is the time in which a spike occurs, and k is defined as below,

$$\kappa(t - t^j) = V_0.(exp(-\frac{t - t^j}{\tau_s}) - exp(-\frac{t - t^j}{\tau_f})), \tag{3.10}$$

where $\frac{\tau_s}{\tau_f}$ has to be set to 4. In our case, τ_s and τ_f are set to 10 and 2.5, respectively. Moreover, V_0 is called the normalizing factor which helps assure that the magnitude of the kernel will remain less than 1.

3.2.2 PSD Learning Algorithm

The PSD algorithm is inspired by the traditional Widrow-Huff (WH) rule defined in the following equation [158],

$$\Delta w_i = \eta u_i (y_d - y_o), \tag{3.11}$$

where η is a positive number called the learning rate which specifies the rate of the training, u_i corresponds to the input, y_d is the desired output and y_o is the output produced by the network, Δw_i specifies the weight update value.

The traditional WH rule cannot be easily applied on training SNNs because the WH model was originally proposed to handle continuous signals. For spikes, the input and outputs are just defined as the time in which spikes occur. Due to this reason, the updated formulation presented in equation 3.11 cannot be directly applied on SNNs.

Assuming that each neuron can fire several spikes, an impulse function can be used to formulate the spike trains that fired by each neuron,

$$S(t) = \sum_j \delta(t - t^j), \tag{3.12}$$

where δ is the Dirac or impulse function, t^j is the time when the $j - th$ spike occurs. The input, output and desired spike trains can also be defined ,

$$\begin{cases} S_{in} = \sum_j \delta(t - t^j) \\ S_{out} = \sum_i \delta(t - t^i) \\ S_{des} = \sum_f \delta(t - t^f), \end{cases} \tag{3.13}$$

where S_{in}, S_{out} and S_{des} correspond to the input, output and desired spike trains respectively

[136].

The Dirac function in the input, output and desired output makes it difficult to use the traditional WH rule. In order to resolve that issue, a concept called spike convolution is used. Convolving a kernel with different spike trains converts spike trains to real-valued functions. In this way, the WH rule can be applied to the spike trains as well.

$$\breve{s}(t) = s(t) * \kappa(t), \tag{3.14}$$

where $\kappa(t)$ is the kernel function and is derived from equation 3.12. Based on this derivation, the convolved spike has the same form as I_{PSC} and the only difference is that I_{PSC} is the weighted sum of the afferent spiking neurons. $S_t(i)$ is the presynaptic spike train, and $\breve{S}_t(i)$ is the convolved version of $S_t(i)$ where it is convolved with the kernel function expressed in equation 3.10. The convolution of $S_t(i)$ with the kernel function, will transform $S_t(i)$ from spikes to an analog current signal. It is essential to convert the presynaptic spike trains to the analog current, otherwise the postsynaptic neurons will not be able to process the information coming from presynaptic neurons. Figure 3.1 demonstrates a post synaptic current produced for random input and random output spikes. It can be seen that it is continuous and can be used for training the SNN using WH training rule.

After the above modifications, the updated WH learning rule that is suitable for training is expressed,

$$\frac{dw_i}{dt} = \eta(S_{des}(t) - S_o(t))I^i_{PSC}(t). \tag{3.15}$$

The above equation indicates that by applying the kernel introduced in two discrete spike trains using equation 3.14, we can calculate the derivative of weights that corresponds to the weight update in each iteration.

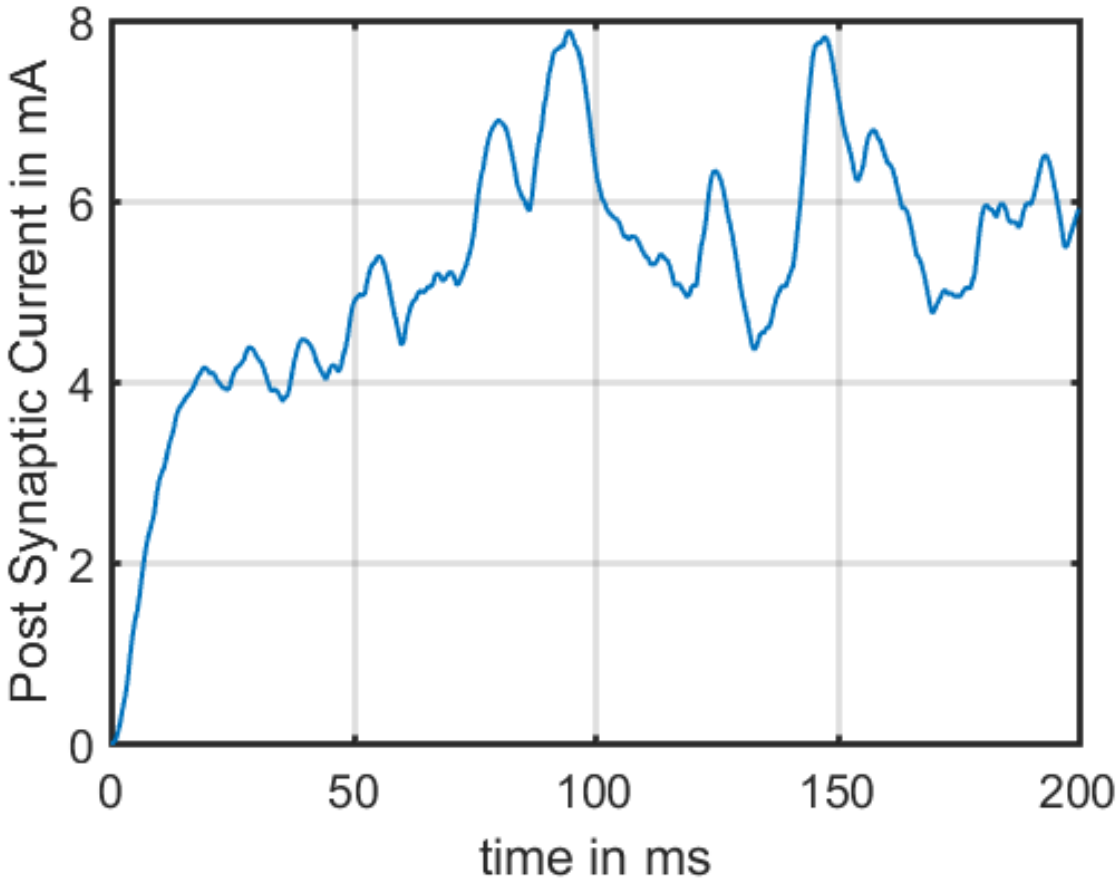

Figure 3.1. Post Synaptic Current after Spike Convolution.

3.2.3 Error Function

The Rossum metric was used in the PSD algorithm for determining the error. This metric calculates the distance between the desired output and the output produced by the network and is defined as below [136],

$$Dist = \frac{1}{\tau_s} \int_0^{+\infty} [h(t) - g(t)]^2, \tag{3.16}$$

where $h(t)$ is the filtered version of the output and $g(t)$ is the filtered version of the desired output. In our , we consider two different spikes trains as the labels of two different classes, i.e., attacked measurements and not-attacked measurements. In this chapter, the desired output means the label of the related input signal. In the learning process, the $Dist$ function does not affect the updating of weight values. However, the Dist is a measure that indicates

when we have a good training.

3.3 RC and Spiking DFR

In this section, we first introduce the structure of the RC, and then explain how this can be modified to spiking DFR.

3.3.1 Design and Structure

In the past, neuron function was modeled as a linear integration of multiple synaptic inputs followed by a threshold nonlinearity. However, recent neural information-processing research suggests that single-neuron function is actually much more complicated [159]. These data suggest that the spatial arrangement and the cable-filtering properties of biological synapses and dendrites make the spatial-temporal characteristics highly nonlinear. In addition, these temporal and spatial nonlinear integration processes enable the dynamics of certain neurons to transit between superlinear and sublinear regimes. Such superlinear excitatory inputs can transform the neurons into a two-layer neural network with the potential to classify linearly non-separable functions. This single-neuron nonlinear-processing property has attracted widespread research interests, not only in neuroscience but also in the ANN [160].

The architecture of the reservoir is based on the RNN. Unlike the RNNs, the connections within the reservoir are not trained whereby randomly synaptic weights are assigned. The input connections serve as the scaling of the input signal and transfer the scaled signal to the reservoir. Within the reservoir, nodes are connected in a random manner whereby the nonlinear mapping takes place. The two well-known reservoir computing models, echo state network (ESN) and liquid state machine (LSM), employ the strength of RNN as the

reservoir or liquid in which the synaptic connection within these layers are not trained. By only training the output weights, the complexity of training process has greatly reduced resulting in less computation power and complexity.

The node states can be expressed as,

$$s(t) = f[W_{res}^{res}.s(t-1) + W_{in}^{res}.x(t-1)], \tag{3.17}$$

where $s(t)$ is the node state at time t, $x(t-1)$ is the input, and W_{res}^{res} and W_{in}^{res} represent the randomly generated reservoir and input connection weights. The output, $y(t)$, can be expressed in terms of input and weight connections,

$$\hat{y} = W_{res}^{out}s(t) + W_{in}^{out}.x(t-1) + W_{bias}^{out}, \tag{3.18}$$

where W_{res}^{out} are the output weights from reservoir, W_{in}^{out} are the feedback weights from output to reservoir, and W_{bias}^{out} is the set of weights for training bias value.

The neuron-like nodes in the reservoir possess the functionality of nonlinear mapping whereby the biological neuron's behavior is achieved. Any reservoir computing systems should possess two properties, 1) high dimensionality, and 2) short-term memory. The RC is computationally inexpensive to train, and its reservoir implementation is highly flexible. These advantages make the RC especially suitable for emerging unconventional computing paradigms.

The DFR model utilizes a single neuron and a delayed feedback to create reservoirs with a ring topology. Figure 3.2 illustrates the topology of our introduced spiking DFR. There are several blocks in this structure. The incoming smart grids measurements first have to be encoded. The encoded data is then converted to the analog current. This current is next applied to the nonlinear node of the DFR, where in this structure is a LIF neuron. The spikes generated by the LIF neuron go through a delay loop, and for each measurement

of the smart gird its corresponding spike train is generated. This process is repeated until the corresponding spike trains of all the measurements are generated. In the next step, the interspike intervals of the spikes trains are used as the features for training the readout layer, where in our case is a MLP. The MLP is trained with the features extracted in the reservoir layer, and there are two different labels corresponding to the compromised measurements and not-compromised measurements.

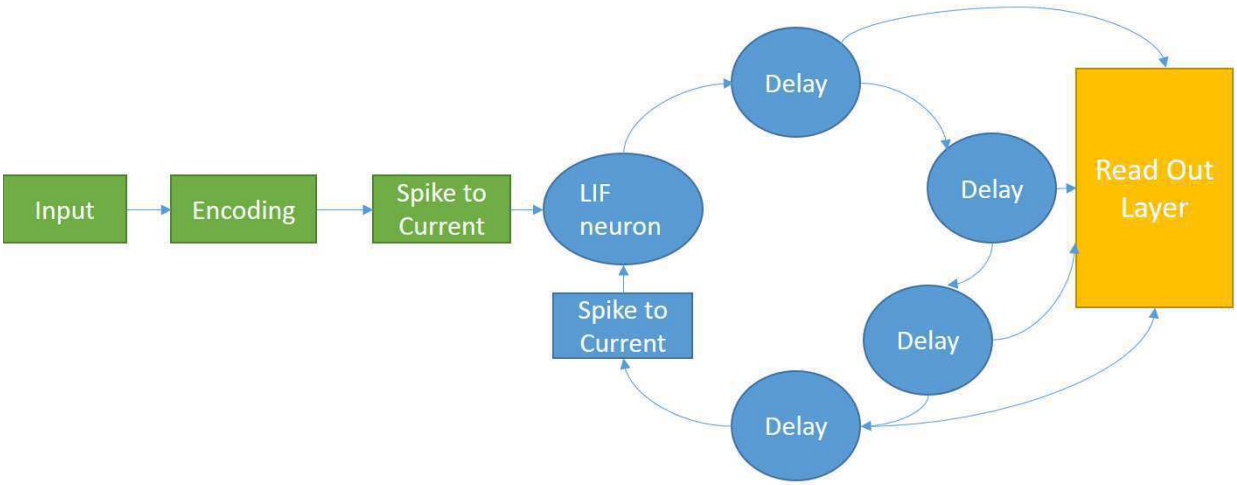

Figure 3.2. Spiking Delayed Feedback Reservoir Computing.

The governing equation for DFR is expressed in as,

$$\dot{x} = -x(t) + F(x(t-\tau), I(t), \theta), \tag{3.19}$$

where f is a nonlinear differentiable function, also known as the nonlinear mapping, τ is the delay, $x(t)$ is the states of DFR, and $I(t)$ is the input signal along with a masking scheme that injects into the DFR. Within the delay loop, the total delay time, τ, is divided into N equidistant delay units which can be expressed as equation 3.20,

$$\tau = N\theta, \tag{3.20}$$

where θ is the time interval between virtual nodes. Different from the RC, due to the ring topology of DFR, the number of nonlinear nodes is drastically reduced. The input will be injected directly into the nonlinear node whereby the nonlinear mapping takes place. Similar to the traditional RC, the output weight connections are the only trained weights [159].

3.3.2 High dimensional behavior of DFR

The reason that DFR has drawn a lot of attention in machine learning is that it is capable of mapping data to a higher dimensional space which makes the data linearly separable [159]. The question that arises is , "How it could be possible that a simple delay would map the data to a high dimensional space". The chaos theory answers this question very well. The Lyapunov analysis of chaos systems shows that the delay can map the low dimensional data to a higher dimensional space [161]. Several studies have shown that the Lyapunov dimension of a chaos system directly corresponds to the delay of the loop [161]. In this chapter, we will show that there is a potential to combine DFR and SNN. In that way, we can benefit from both of them. Applying DFR on the data will map it to a higher dimensional space that makes the linear separation of the attacked and not-attacked easier. On the other hand, if we can use the SNN trained by the PSD rule, then it is possible to take advantage of the spatiotemporal correlation that exists in different components of smart grids.

Based on several studies, DFR has to be at the edge of the chaotic region in order to show high dimensional behavior. In Section 3.5, we will examine the behavior of the DFR for several delays and we will choose the value that shows more chaotic behavior, as the delay used in our simulations. As it can be seen in Figure 3.3, the data on the left side is not easily linearly separable but when it gets mapped to a higher dimensional space, it is much more easily separable. That is the main motivation behind using DFR in this chapter.

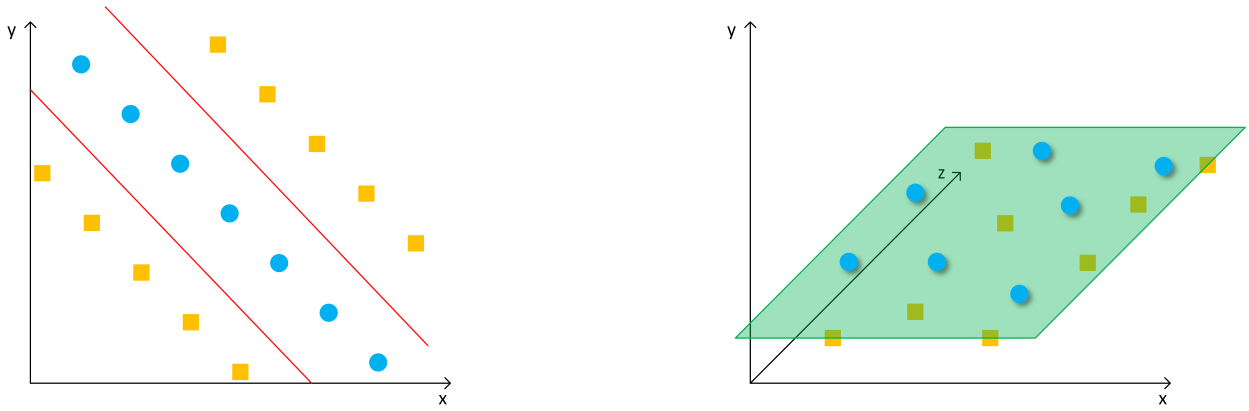

Figure 3.3. High dimensional mapping of data using DFR.

3.4 Training PSD For FDI Attack Detection

In this section, we introduce different spike encoding schemes and describe how to leverage them for encoding the smart grids' measurements. We then use the encoded measurement for training the SNN using the PSD algorithm. Encoding the information and learning are the two major components of each SNN [156]. In this chapter, three different encoding schemes are explored to encode the measurements produced by the smart grid simulations. That is, latency, latency-phase, and ISI are respectively used for training SNN using the PSD.

3.4.1 Latency encoding

In latency or time to first spike (TTFS) encoding, a stimulus is encoded as the time in which the first spike fires with respect to the stimulus, i.e., the latency between the onset of the stimulus and the first spike observed [157]. Latency code is formulated as below in [156],

$$Latency = t_i = t_{max} - ln(\alpha.s_i + 1), \tag{3.21}$$

where t_{max} is the maximum length of the encoding window; α is the scaling factor; s_i is the intensity of each pixel. In our case, t_{max} is set to 200 ms, s_i is replaced by smart grid measurements and α is set to -1 when the measurement is a negative number and 1 otherwise. Equation 3.21 is inspired by this fact that stimulus with stronger intensity leads to shorter response time and stimulus with weaker intensity leads to a longer latency for the neuron to fire [162].

3.4.2 Latency-Phase encoding

It has been shown in [157] that combining or multiplexing different encoding approaches can yield higher information capacity. There is also other evidence showing that multiplexing of different encoding schemes happens in the brain and that is a reason why a brain is a powerful tool in discriminating different objects. Inspired by this background, Nadasdy [163] proposed a mechanism for combining latency and phase codes. The relative timing of a spike with respect to the intrinsic subthreshold membrane potential oscillation (SMOs) is the feature that defines the phase code of a spike [163]. SMOs can simply be defined as below for the measurement of the *i-th* meter in the smart grid,

$$SMO_i = Acos(2\pi ft + \phi_i) \quad i = 1, 2, ..., L, \tag{3.22}$$

where A is the magnitude of SMO and in this chapter, it is set to 1, f is the oscillation frequency that in our case is set to 0.05, L is the total number of encoding units that in this chapter is equal to 34 which represents the total number of meters in the smart grid design that we used, ϕ_i is the initial phase of the i-th meter and is presented as the following equation,

$$\phi_i = \phi_0 + (i-1)\Delta\phi, \tag{3.23}$$

where ϕ_0 corresponds to the reference phase, and $\Delta\phi$ is the phase difference between adjacent smart grid meters. In [136, 156], $\Delta\phi$ is set to $2\pi/N_{en}$ and N_{en} is the total number of pixels.

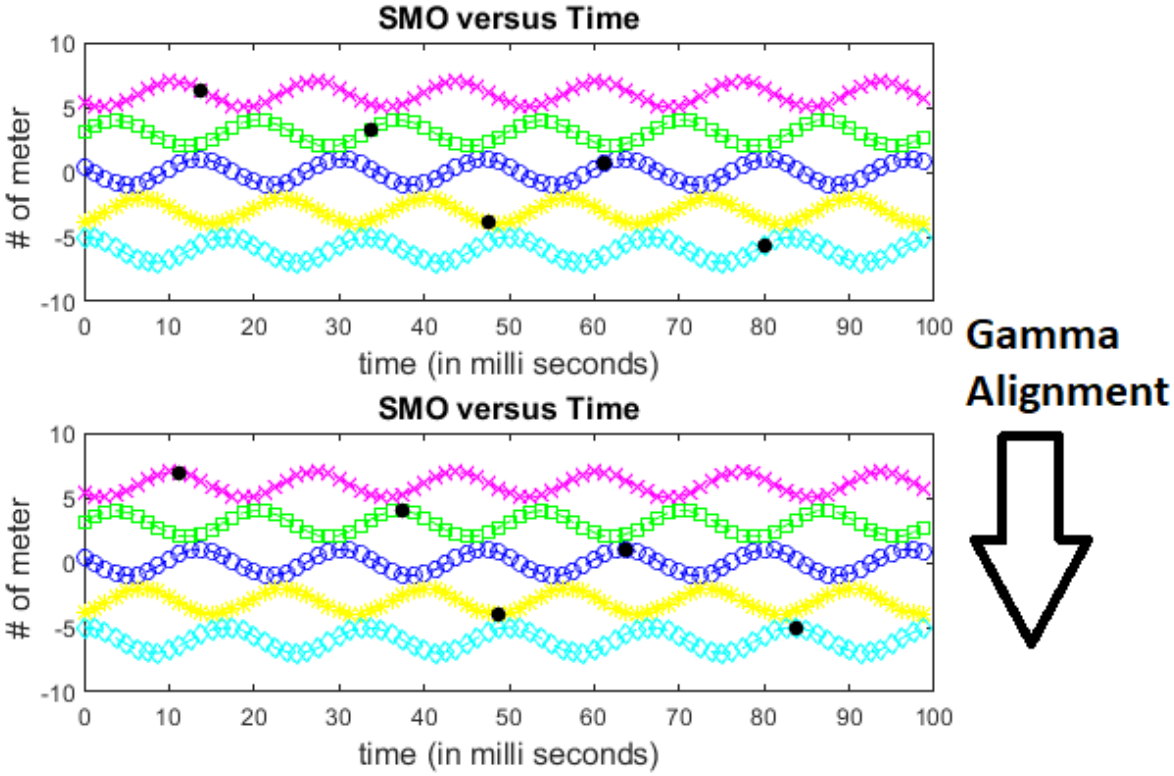

Figure 3.4. Aligning latency code with SMO of each meter using Gamma alignment.

Gamma alignment is the process that multiplexes the latency code with phase code. In this process, the latency code gets synchronized with the corresponding SMO of each meter as shown in Figure 3.4. In fact, the latency codes are drifted using this method so that they are located at the peak of their corresponding SMO for each meter. In this way, we can combine phase and latency code and probably get better results in training the SNN using PSD code.

3.4.3 ISI encoding

ISI encodes the relative distance between the spikes, i.e., instead of using the absolute timing of each spike, the relative distance of each spike with respect to other ones is used for encoding the stimulus. Figure 3.5, demonstrates an example of ISI.

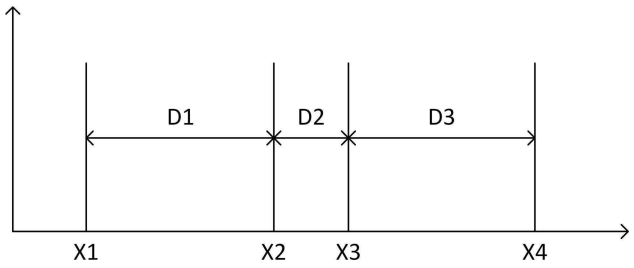

Figure 3.5. The interval between spikes in ISI encoding.

$$D_i = f(C_i, V_i) - f(C_{i-1}, V_{i-1}), \tag{3.24}$$

where $f(C_i, V_i)$ is defined as below,

$$f(C_i, V_i) = (C_{i+1})[\beta(V_i - \gamma) + \theta], \tag{3.25}$$

where C_i,V_i are the parameters of the LIF neuron performing the encoding. C_i,V_i corresponds to the capacitance of the membrane and the threshold [150, 151]. The number of the spikes produced by the encoding unit is set as 4 for the sake of the simplicity. This means that for the measurement of each meter, there are totally three interspike intervals. The extracted intervals will be used later to train the SNN with PSD.

3.5 Performance and Analysis

3.5.1 Using SNN for Dynamic Attack Detection

In our simulations, we assume that the number of compromised meters can vary from 0 to 34. Figure 3.6 shows the average training error for the attacked data. As it can be seen, TTFS multiplexed by phase can improve the training performance. This agrees with our initial hypothesis that multiplexing different codes results in better classification. However, the training performance with ISI is even better than the TTFS multiplexed phase. Figure 3.6 demonstrates the average training error for different numbers of compromised meters. As it can be seen in Figure 3.7, the training error of PSDS using ISI is better than both other methods, and the performance of the TTFS multiplexed by phase encoding is better than simple TTFS. All the simulations are performed over 2000 iterations, and the error is defined as the distance between the generated spike train and the spike train specified as the label for each attack. In this chapter, 10 different neurons are used at the output layer, and for each number of compromised meters, there exists one spike train as the label. The SNN trained with the PSD algorithm will associate the pattern of each spatiotemporal attack depending on the number of compromised meters with a spike train in the output. It means that if an attack is performed that only compromises one meter, the first neuron has to fire a spike train at 20ms, 60ms and 100ms and the other neurons should not fire any spikes. The error for SNN is defined as the distance between the desired spike train and the output spike train as defined in equation 3.16.

As seen in Figure 3.7, when the number of compromised meters increases, the training error decreases. In our simulations, we set the learning rate, η, equal to 0.01. The weights are initialized with a normal distribution of mean and variance equal to 0.5 and 0.2 respectively. The output neurons are supposed to fire three spikes at three different times for the samples

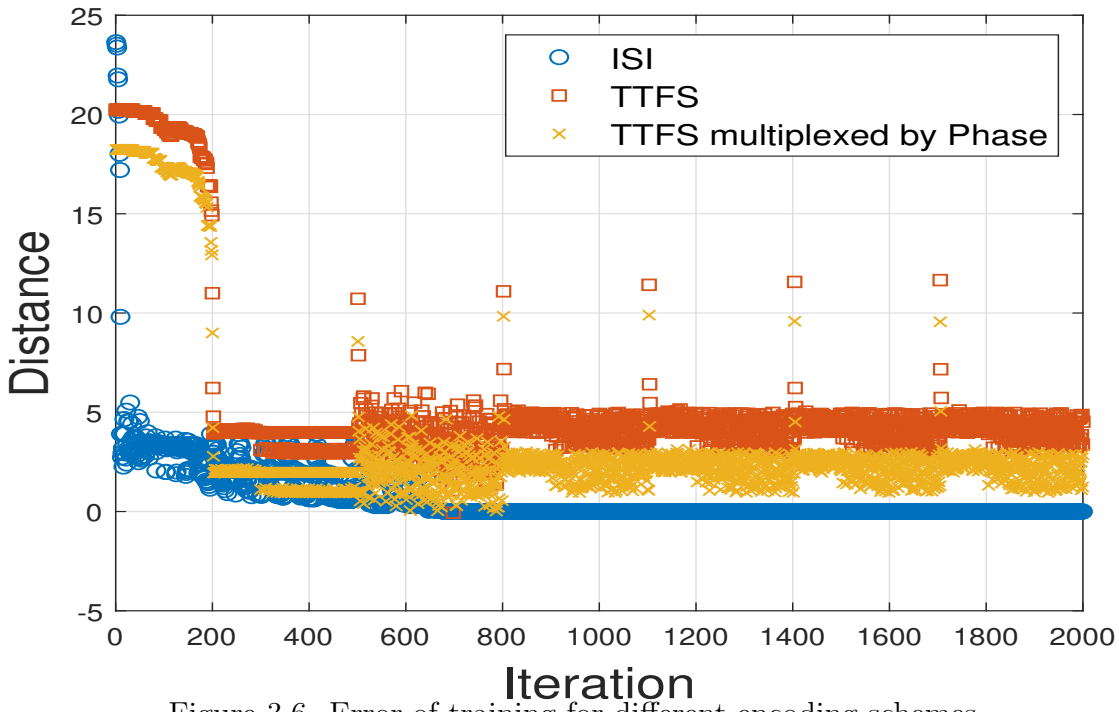

Figure 3.6. Error of training for different encoding schemes.

of each class of data. These results indicate that SNNs, especially if being trained by ISI encoding, have a good potential for supporting FDI attack detection. The drawback of ISI encoding is that ISI produces multiple spikes per meter, as opposed to only a single spike per meter for the latency encoding. This makes it computationally more challenging, and the training takes longer.

The performance metric used to evaluate the detection performance is the **accuracy** which is defined as

$$\text{Accuracy} = (\text{TP} + \text{TN}) \,/\, (\text{TP} + \text{TN} + \text{FP} + \text{FN}) \,, \tag{3.26}$$

where TP, TN, FP and FN correspond to the number of true positive, true negative, false positive and false negative samples respectively. As it can be seen in Figure 3.8, ISI encoding achieves higher attack detection rate based on the accuracy metric defined in equation 3.26. For testing, some other data that are not used for the training are applied on the weights

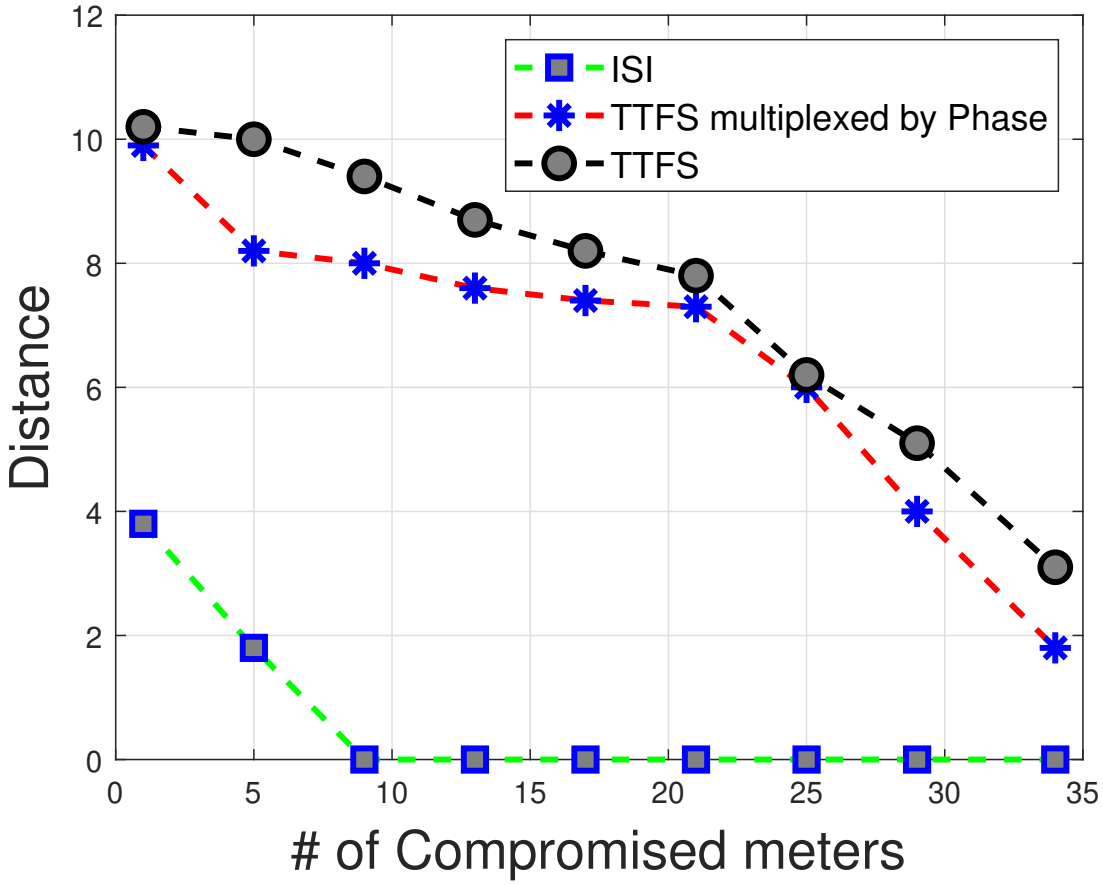

Figure 3.7. Error of training for different encoding schemes & different number of compromised meters.

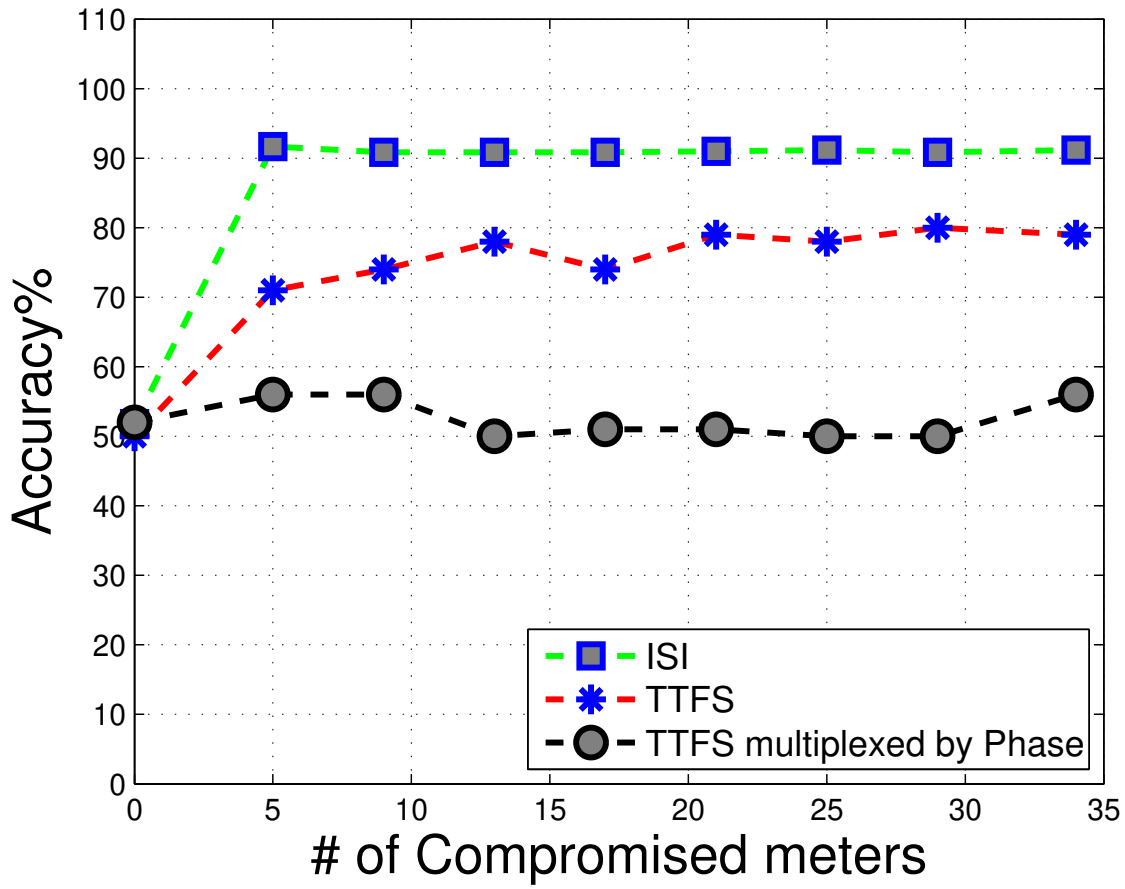

Figure 3.8. Test results.

achieved from the training phase. The test results also confirm that ISI is a better encoding scheme. Therefore, multiplexing can improve the results of TTFS encoding but it cannot outperform the ISI.

3.5.2 Attack Detection Using DFR

As it was mentioned in Section 3.4.1, RC is a type of recurrent neural networks that is easier to train, and is capable of classifying linearly non-separable data by mapping them to higher dimensional space. In this section we use spiking DFR to detecting the dynamic attacks. The data that we use are unbalanced. It means that the ratio of the compromised and uncompromised samples is not equal. In our data set, 20% of the samples are compromised and 80% are not. The performance metrics for evaluation are ***accuracy***, and ***F1***. ***Accuracy***

was defined in Section 3.5.1, and ***F1*** is defined,

$$\mathrm{F1} = 2 \times \frac{Precision \times Recall}{Precision + Recall}, \tag{3.27}$$

where $Precision = \frac{TP}{TP+FP}$, and $Recall = \frac{TP}{TP+FN}$.

Figures 3.10 & 3.11, demonstrate that the combination of the spiking neurons, DFR,

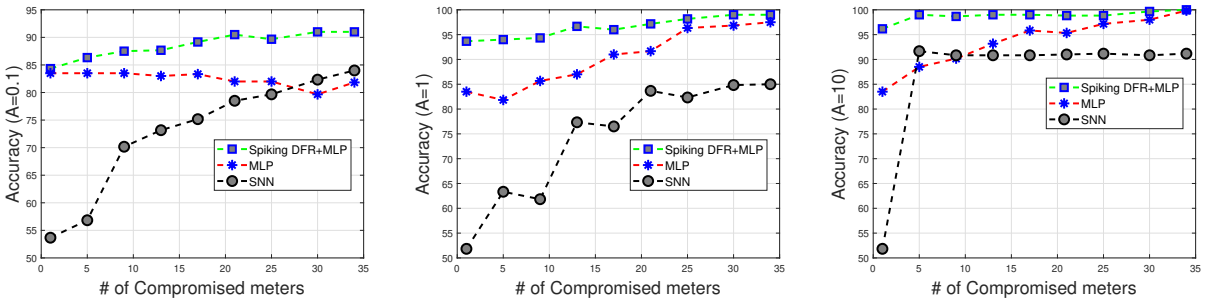

Figure 3.9. Accuracy of attack detection for three different methods and magnitude of attacks, A=0.1,1,10.

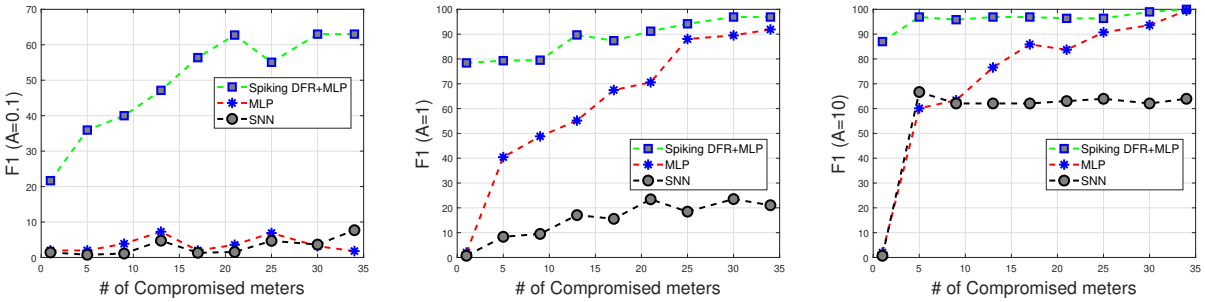

Figure 3.10. F1 of attack detection for three different methods and magnitude of attacks, A=0.1,1,10.

and MLP achieves a higher performance than the MLP and SNN, based on both evaluation metrics. As it can be seen, as the magnitude of the attack, A, increases the performance also gets better. It is due to the fact that DFR can mimic the RNNs and captures the spatiotemporal correlation between the meters of the smart grids. In the DFR+MLP, the MLP is used as the readout layer of the DFR. The structure of the spiking DFR is shown in

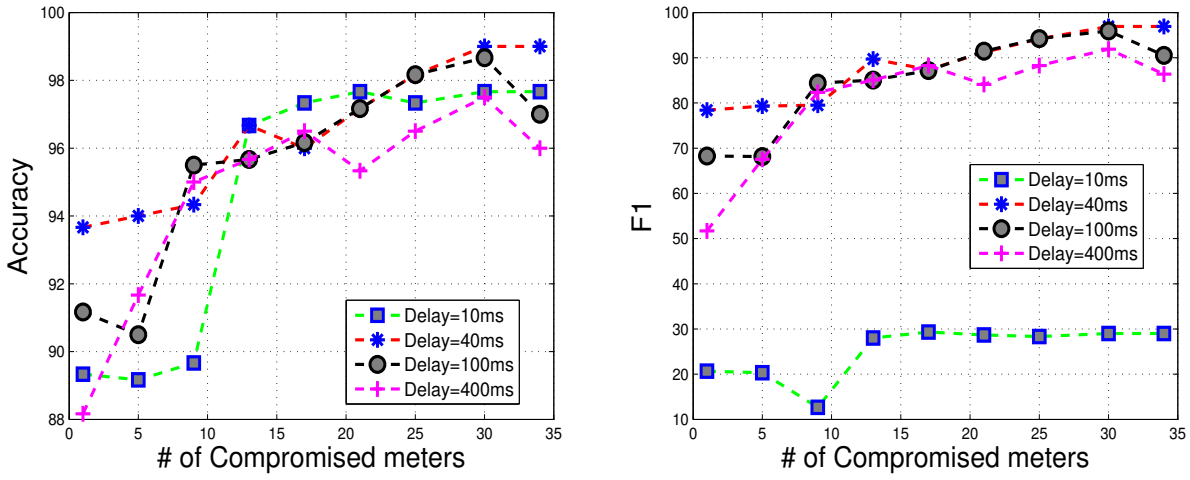

Figure 3.11. Effects of different values of the delay on the performance when the A=1 .

Figure 3.2. The input is the data that is collected. In our simulations we use MATPOWER for generating the data, and 20% of the total data is compromised by dynamic attack defined in (6). The nonlinear node of the DFR is chosen as a LIF neuron, which is a model of spiking neurons. The encoded input needs to be converted to the current that can be applied to the LIF neuron. Equation 3.9 is used for converting the encoded input to the current. This current is then applied to the LIF neuron, and for each current the corresponding spike train is generated. These spike trains are delayed in time, and then are again converted to the current that can be added to the corresponding current of the next input. This process is repeated until all the samples are processed and the corresponding spike train of each input samples is generated. These spike trains are composed of several spikes, and each spike occurs at a different time. The occurrence frequency of these spikes are used as the features for training the readout layer. In this chapter, the MLP is the readout layer. The input of the MLP is the occurrence time of these spikes and the output of the MLP is 0 or 1, depending on the class of the data. If the data are attacked, then the output of the MLP is 1, otherwise is 0. After the training is accomplished, the data that have not been previously seen by the spiking DFR is used for testing.

Totally 10000 samples are used for training and 10000 are used for testing, where 20% of the samples are compromised and 80% of them are not. The two different ratios of compromised and uncompromised samples, make the data unbalanced. The same dataset are also used for training the MLP and SNN. In Figures 3.10 & 3.11, the performance of these three methods, DFR+MLP, MLP, and SNN with respect to to the number of compromised meters and the magnitude of the attack is plotted. When the magnitude of the attack increases, the performance gets better regarding both performance metrics. The spiking DFR+MLP, achieves the best performance. For low values of magnitude of attacks, both MLP and SNN achieve very low performance rates. However, DFR+MLP can detect the attacks with much higher accuracy even for low values of magnitude of attack. Our simulation results demonstrate that the average **accuracy** of attack detection for all different magnitude of attacks and number of compromised measurements is increased up to **94.6%** when spiking neurons, DFR, and MLP are combined together in a single platform. In the simplest case where only SNNs are used, the average **accuracy** is **77.92%**, which shows that our introduced method can improve the average **accuracy** about **17%**. The enhancement brought about by our method regarding ***F1*** measure is even more significant. The average ***F1*** achieved by combining spiking neurons, DFR, and MLP is **78%**. However, the ***F1*** is only **25%** where we just use SNN for dynamic attack detection. This result implies that our introduced algorithm can improve ***F1*** measure about **53%** In Section 3.5.1, the performance of the SNN was presented while the magnitude of the attack was high, i.e., $A = 1$. However, as it is shown in Figures 3.10 & 3.11, the SNN achieves very low performance rates when the magnitude of the attack is low. Specially, $F1$ metrics demonstrates that in situations that the magnitude of the dynamic attack is very low the MLP and SNN are both incapable of detecting the attacks. The main reason is that the MLP and the SNN are not able to capture the spatiotemporal correlation of the data and map the data to higher dimensional space respectively. However, the spiking DFR+MLP is able to capture the spatiotemporal

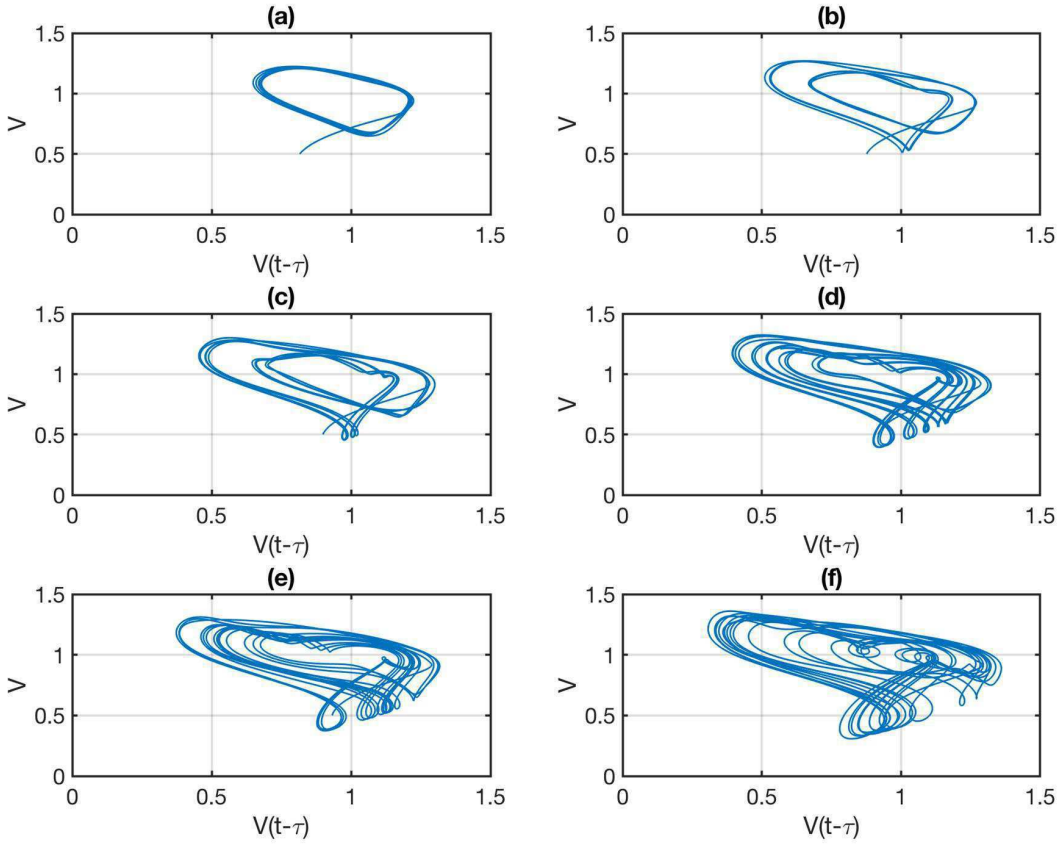

Figure 3.12. Phase portraits of dynamic systems.

correlation existing in the data due to its recurrent structure, and it also maps the data to a higher dimensional space due to the fact that it is able to act at the edge of chaos [164].

3.5.3 Effect of Delay on the Performance

Figure 3.12. shows the performance of the DFR+MLP when the delay value is changing. When the delay value is set to 40ms, the performance achieves higher rates regarding both **accuracy** and **F1**. However, when the delay is set to 10ms, the lowest performances are obtained. This is due to the fact that proper values of the delay need to be chosen so that the spiking DFR+MLP can work at the edge of chaos. Varying the time delay, the phase portraits are illustrated in Figure 3.12. As delay increases, the dynamic behavior varies from ordered to the edge of chaos to completely chaotic. Studies in [161] suggest that a dynamic system can show a high dimensionless behavior if its delay is tuned in a

way that the dynamic system can operate at the edge of chaos. To further examine the dynamic behavior, the solution to the DDE equation is investigated. The dynamic behavior of the nonlinear function is modeled by the DDE with varied delay time, as demonstrated in Figure 3.12. The phase portrait is a representation of the solutions tracing the path of each particular solution. It is a graphical tool to visualize how the solutions of a given system of differential equations would behave in the long run. Phase portrait tracks the dynamic behavior and demonstrates the chaotic or periodic behavior of the system.

3.5.4 Comparison With Classical Algorithms

The SVE is the first method for attack detection in smart grids [141]. This method is based on calculating the chi-squared residual stated as ρ. If the calculated value of ρ is greater than a predefined threshold value, then the measurement vector, z, is classified as attacked. The residual value, ρ, is defined as,

$$\rho = ||\tilde{z} - H\hat{x}||_2^2, \tag{3.28}$$

where $\hat{x}$ is the estimated state of the smart grid, and can be calculated as follows,

$$\hat{x} = (H^T W H^{-1}) H^T W z, \tag{3.29}$$

where W is a diagonal matrix and its diagonal elements are the reciprocal of the measurements variance. The ith diagonal element of W corresponds to the reciprocal of the variance of the ith meter. The SVE is very easily implemented. However, it faces a major shortcoming while dealing with hidden attacks. In the case of hidden attacks, as we will show and prove in (30), the residual value is always less than the threshold value. Therefore, the SVE

cannot be useful for hidden attacks.

$$\begin{aligned}\rho &= ||\check{z} - H\hat{x}||_2^2 \\ &= ||z + a - H((H^T W H^{-1}) H^T W z)||_2^2 \\ &= ||z - Hx + Hc - H((H^T W H^{-1}) H^T W H c)||_2^2 \\ &= ||z - Hx + Hc - HC||_2^2 \\ &= ||z - Hx||_2^2 \leq \tau\end{aligned} \tag{3.30}$$

As it can be seen in (30), when the attack is hidden the residual value, ρ, is always smaller than the threshold value, τ. Therefore, the hidden attack cannot be detected by SVE. Since, in this chapter we also consider the attack to be hidden, it means that the SVE is not a good choice for us.

3.5.5 Complexity Analysis

In this section, the complexity of our approach is analyzed and compared with other methods. The computational complexity of the spiking DFR based dynamic attack detection is associated with calculating the reservoirs and updating the weights of the readout layer. In the spiking DFR based approach, the only weights that require training, are the weights of the readout layer. However, in RNNs the weights of all the neurons in different layers have to be trained. The training of all the layers of RNNs make them much more computationally complex than RC based structures. The total number of floating- point operations (FLOPs) can be used for measuring the complexity. It has been shown in [69] that the computational complexity of RC based learning approaches, correspond to the training time. Therefore, in this section we will demonstrate the training time of our approach and compare it with the training times of the MLP, and SNN. In Table 3.2, the training times of spiking DFR+MLP,

MLP, and SNN are presented. The comparisons are done among the training time of these three algorithms, which imply their computational complexities as well.

Table 3.2: Computational Complexity Analysis

Algorithm	Training Time
Spiking DFR+MLP	16.69 s
MLP	3.2 s
SNN	90 s

As shown in Table 3.2, the SNN algorithm has the highest computational complexity. In Figure 3.6, it can also be observed that it takes a lot of iterations for the SNN algorithm to converge. The spiking DFR+MLP, and MLP are the second and third computationally complex algorithms respectively. As it can be seen in Figure 3.2, the spiking DFR+MLP has some building blocks that make it more computationally complex than MLP. Our introduced algorithm contains some blocks, e.g. spike to current, reservoir block, and MLP as the readout layer. These blocks make our algorithm more computationally complex than a simple MLP. However, this extra overhead is not that much significant as shown in Table 3.2.

3.6 Conclusion

This is the first chapter exploring spiking DFRs for dynamic cyber-attack detection in a smart grid. Our method integrates DFR and MLP to detect FDI dynamic attack in a smart grid. Different temporal encoding schemes are investigated to encode the measurements obtained from the smart grid. Simulation results suggest that ISI yields the best training performance because it conveys more information than other encoding schemes. On the other hand, DFR can be used to map the data to a higher dimensional space making it easier to classify the compromised data from uncompromised data. Simulation results also

show that combining DFR with MLP can outperform other methods such as MLP and SNN. Furthermore, the chaotic dynamic behavior can be observed in DFR responses when the delay is changed suggesting the DFR performance is sensitive to the delay value.

Chapter 4

MIMO-OFDM Spectrum Sensing using Delayed Feedback Reservoir Computing

4.1 Introduction

Multiple-input-multiple-output (MIMO) technology combined with orthogonal frequency division multiplexing (OFDM) has been adopted in many advanced wireless communication systems. The combination of MIMO and OFDM technologies improves the spectral efficiency, as the MIMO utilizes the spatial multiplexing gain and the OFDM avoids frequency selective fading. Besides, adding cyclic prefix (CP) to the OFDM symbols decreases inter channel interference (ICI), and inter symbol interference (ISI). However, in a MIMO-OFDM based wireless communication system not all the subcarriers are utilized simultaneously by the primary user (PU), and the spectrum utilization efficiency is low [47, 165–169]. Dynamic spectrum sharing (DSS) in a MIMO-OFDM system introduce a solution to resolve

this problem where the under utilized subcarriers can be used by the secondary users (SUs).

The SUs are allowed to transmit signals only on the subcarriers that are found idle and they should evacuate those bands as soon as the PU wants to use them. Therefore, it is fundamental for the SUs to perform spectrum sensing subsequently to identify spectrum holes accurately and the interference is minimized. The performance of the spectrum sensing can be significantly degraded due to fading wireless channels, and low signal-to-noise (SNR) ratios. In addition, sensing the OFDM symbols faces more challenges including noise uncertainty, timing delay and carrier frequency offset [170]. Despite, numerous efforts these challenges still remain unsolved. In order to minimize the interference, for a given SNR, the probability of detection (P_d) has to be close to one while the probability of false alarm (P_f) is close to zero.

So far, in the literature three major techniques have been introduced as the classical spectrum sensing methods. These methods are energy detection, matched filtering, and cyclostationary feature detection which suffer from several drawbacks including, low probability of detection at low SNRs, requiring accurate prior knowledge of the signal, and computational complexity, respectively [47–49]. Due to the limitations of classical spectrum sensing techniques, machine learning (ML) approaches have drawn a lot of attention as they have several advantages over traditional spectrum sensing techniques. These advantages are: 1) the ML based spectrum sensing approaches are more adaptive and can learn the surrounding DSS environment (e.g., the fading channel) effectively; 2) the detection performance of the ML based spectrum sensing techniques are better as they can identify the decision boundaries [50, 51].

Recurrent neural networks (RNNs) have been introduced to capture the temporal correlation of the RF received signals and increase the P_d for a low value of P_f [171]. However, due to the vanishing gradients, the traditional RNNs are very challenging or even impossible to

train. Reservoir computing (RC) is a new generation of RNNs that is much easier to train, and in many cases have shown equal or even better performances than the traditional RNNs. There are three different types of RC systems, echo state networks (ESN), delayed feedback reservoirs (DFR), and liquid state machines (LSM) [20]. In this chapter we focus on DFRs because they have low computational complexity. For several reasons, we introduce to adopt spiking DFR (SDFR) for spectrum sensing in a MIMO-OFDM system. The neurons in our brains communicate together using spikes and artificial spiking neural networks (SNNs) are the most energy efficient and biologically plausible model of artificial neurons which make them a suitable choice for hardware implementations. As an example, we can mention to Truenorth which is a spiking neural network chip developed by IBM in 2014, and consumes only 70 milliWatts (mW) while performs 46 billions of synaptic operations in a second [172]. Although it is clear that biological neurons leverage spikes to communicate, the encoding mechanism which is adopted to encode the neural information is not clear yet. There are evidences which show that temporal encoding is the most likely encoding scheme used by biological neurons [157]. Temporal encoding schemes are also divided to different categories. Latency or time to the first spike (TTFS) and interspike-interval encoding are the two most well-known temporal encoding schemes in biological neurons. In this chapter we introduce the mathematical representation of these two schemes and identify the optimal encoding in terms of detection accuracy. On the other hand, training the DFR is composed of an unsupervised feature extraction followed by a supervised training. It will be discussed in Section 4.2 that the unsupervised feature extraction step of the DFR does not require any training which makes DFR very suitable for practical scenarios where the training time should be very low. Moreover, none of the ML based methods in the literature have been adopted in the spectrum sensing of MIMO systems. MIMO is a core technology in many advanced wireless communication systems. Therefore, it is very important to introduce a spectrum sensing method which can be adopted in MIMO systems as well.

As discussed earlier, ML based spectrum sensing approaches provide several benefits. However, these approaches face multiple challenges as well. One of the major challenges that the ML based spectrum sensing techniques suffer from is the scarcity of labeled data [173]. In order to achieve high performances, the ML based techniques are required to use sufficient and expressive data for training. Collecting sufficient and expressive data for spectrum sensing is very expensive and time consuming. Therefore, we introduce to leverage generative adversarial networks (GANs) combined with SDFRs to generate synthetic samples using a small number of real samples [174]. The GANs were initially introduced to synthesize fake images which look very similar to the real images. In recent years, the GANs have been applied in other domains as well. Data collection process is both expensive and time consuming and we will show that GANs combined with SDFR are very efficient and fast to train in spectrum sensing while there is limited training data. The main contributions of this chapter are the followings:

- First, we introduce an energy efficient and easy to train hybrid training scheme using spiking neurons and DFR combined with multi-layer perceptrons (MLPs) to identify the busy/idle subcarrier in a MIMO-OFDM system. We incorporate the temporal correlation in the PU activity behavior using spectrum sensing dataset developed by RWTH Aachen University Static Spectrum Occupancy Measurement Campaign [175]. In most of the spectrum sensing literature, the activity of the PUs follows a Markov chain or a random trend [68]. However, none of these two assumptions are realistic. We use the Aachen university PU activity behavior to model the occupancy trend of the PU in this chapter. RWTH Aachen University Static Spectrum Occupancy Measurement Campaign has conducted several experiments to measure the PU activity in different frequency bands and time slots. Their experiments have shown that in each frequency band, a significant temporal correlation exists in the PU activity in different

time slots. In this chapter, the occupancy of each subcarrier is modeled based on its corresponding frequency occupancy model which is extracted from RWTH Aachen university spectrum occupancy database. Then, we will show that our introduced scheme outperforms the state-of-the-art techniques in terms of detection accuracy and computational complexity. This is the first time that the hybrid SDFR and MLP platform is being introduced for spectrum sensing.

- Second the optimal encoding scheme to encode the spikes is introduced. It is clear that in our brains, neuron communicate together using spikes. However, the mechanism which is used to encode the information is not clear. We will formulate two temporal encoding schemes namely, latency and ISI and identify the optimal encoding mechanism in terms of detection accuracy.

- Third, our model is extended to a stacked deep SDFR in space domain to capture the spatial correlations which exist while there are multiple SUs in a cooperative DSS environment. The results indicate that the deep extension of SDFR in space is very effective to exploit the spatial correlations between multiple SUs.

- Fourth, the combination of SDFR and conditional GANs (cGANs) are introduced to synthesize more training data while the training data is scarce. We investigate the quality of the synthesized data in different scenarios. To the best of our knowledge, this is the first time that the cGANs are used to synthesize MIMO-OFDM symbols for data augmentation. It is shown that the detection performance of our introduced spectrum sensing approach will significantly drop when the size of training data is limited and we resolve this issue by introducing a combined cGAN and SDFR platform.

The organization of this chapter is as follows. In Section 4.2, the system model of the MIMO-OFDM spectrum sensing and detailed description of SDFR based spectrum sensing are

presented. Section 4.3, describes our approach for using cGANs to synthesize MIMO-OFDM symbols to enlarge the training dataset. Section 4.4 presents the results and performance evaluation. Section 4.5, concludes the chapter.

4.2 Problem Formulation & System Model

In OFDM multicarrier transmission, $L-$point inverse discrete Fourier transform (IDFT) is applied to modulate the PU symbols. In this chapter we use QPSK for modulating the data bits. Subsequently, the CP is added to the symbols and the symbols are transmitted over a fading Rayleigh channel. In this chapter, we set the mean, and the variance of the Rayleigh channel equal to 0, and 1, respectively. Equation 4.1 represents the k^{th} QPSK symbol generated by the PU after it passes through the IDFT,

$$s(t - kT_i) = \sum_{l=0}^{L-1} S_{k,l} e^{\frac{j2\pi l(t-kT_i)}{T_i}} e^{j2\pi f_i(t-kT_i)} \tag{4.1}$$

where $S_{k,l}$ is the PU symbol that is modulated on the l^{th} subcarier, f_i is the QPSK carrier frequency, and T_i is the QPSK symbol period. This symbol is transmitted over a fading Rayleigh channel and the received signal is down converted to the baseband and then passes through the $P-$point discrete Fourier transform (DFT). The n^{th} OFDM symbol can be written as follows,

$$y(t - nT_s) = e^{-j2\pi f_s(t-nT_s)} \sum_{m=0}^{M-1} h_m s(t - kT_i - mT_s), \tag{4.2}$$

where h_m is the Rayleigh channel fading coefficient, T_s is the OFDM symbol period, and f_s is the OFDM carrier frequency. The signal transmitted on the p^{th} subcarrier is as follows [176]:

$$Y_p(n) = \sum_{l=0}^{L-1} X_{k,l} H_l e^{j2\pi(-kf_iT_i + nf_sT_s)} e^{j\pi\beta_{l,p}(P-1)} \times \frac{sin(\pi\beta_{l,p}P)}{sin(\pi\beta_{l,p})} \qquad 1 \leq p \leq P, \tag{4.3}$$

where $H_l = \sum_{m=0}^{M-1} h_m e^{-j2\pi m\left(f_iT_s + l\frac{T_s}{T_i}\right)}$, $\beta_{l,p} = \left(\frac{k}{T_i} + f_i - f_s\right)\frac{T_s}{M} - \frac{m}{M}$, and M is the number of subcarriers. Therefore, the received signal at the SU which is transmitted on the p^{th} subcarrier can be written as,

$$R_p(n) = Y_p(n) + N_p(n) \tag{4.4}$$

where $N_p(n)$ is the DFT of complex additive white Gaussian noise (AWGN) with zero mean and unit variance. The main objective of this chapter is to determine the presence or absence of the signal on the p^{th} subcarrier. The presence and absence of the received signal are denoted as two hypothesis, $H1$ and $H0$ respectively. Therefore, the received signal is as follows,

$$R_p(n) = \begin{cases} N_p(n) & H_0 \\ Y_p(n) + N_p(n) & H_1 \end{cases} \qquad n = 1, ..., N \tag{4.5}$$

where N is the number of OFDM received symbols. In the energy detection based spectrum sensing, the decision statistics of each subcarrier is formed based on the average received energy of N symbols and is expressed as follows,

$$E_p = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} \left|N_p(n)\right|^2 & H_0 \\ \frac{1}{N}\sum_{n=1}^{N} \left|Y_p(n) + N_p(n)\right|^2 & H_1 \end{cases} \tag{4.6}$$

The decision statistics (E_p) is compared with a threshold value, if E_p is larger than the threshold value then the subcarrier is considered as busy, otherwise the subcarrier is denoted

as idle. The threshold value is calculated based on the given probability of false alarm (P_f). The ideal case is to have a high probability of detection for each subcarrier (P_d^p) while P_f^p has a low value. The P_d^p, and P_f^p are defined as [177],

$$\begin{aligned} P_f^p &= \Pr(E_p > \epsilon^p | H_0) \\ &= Q\left(\left(\frac{\epsilon}{\sigma_n^2} - 1\right)\sqrt{N}\right) \\ P_d^p &= \Pr(E_p > \epsilon^p | H_1) \\ &= Q\left(\left(\frac{\epsilon}{\sigma_n^2} - \gamma^p - 1\right)\frac{\sqrt{N}}{\gamma^p + 1}\right) \end{aligned} \tag{4.7}$$

where ϵ^p is the energy detection threshold for subcarrier p^{th}, γ^p is the signal-to-noise ratio (not in dB) of the p^{th} subcarrier, N is the number of OFDM received symbols, $Q(.)$ is the complementary function of a standard Gaussian distribution, and σ_n^2 is the noise variance which in this chapter is assumed to be 1. The ϵ^p is calculated based on the given P_f^p. So far, the received signals, and decision statistics of a single antenna SU have been identified. However, the receiver of SU might have multiple antennas. The received signal of the p^{th} subcarrier at the j^{th} antenna is,

$$R_p^t(n) = Y_p^t(n) + N_p^t(n) \qquad t = 1, ..., T, \tag{4.8}$$

where T is the number of SU antennas. The decision statistics (E_p) are as follows while we assume there are multiple antennas at the SU,

$$E_p = \begin{cases} \frac{1}{NT}\sum_{t=1}^{T}\sum_{n=1}^{N}\left|N_p^t(n)\right|^2 & H_0 \\ \frac{1}{NT}\sum_{t=1}^{T}\sum_{n=1}^{N}\left|Y_p(n) + N_p^t(n)\right|^2 & H_1 \end{cases} \tag{4.9}$$

The P_f^p and P_d^p for a MIMO-SU are written as follows,

$$\begin{aligned} P_f^p &= Q\left(\left(\frac{\epsilon}{\sigma_n^2}-1\right)\sqrt{TN}\right) \\ P_d^p &= Q\left(\left(\frac{\epsilon}{\sigma_n^2}-\gamma^p-1\right)\frac{\sqrt{TN}}{\gamma^p+1}\right). \end{aligned} \tag{4.10}$$

4.2.1 Delayed Feedback Reservoirs for Spectrum Sensing

In almost every system delay exists, inevitably in the neurons of our braina as well. The delay can affect the performance of the neurons. Therefore, it is very essential to study the delay effects on the information processing neurons. In delayed dynamic systems, the state of the system not only depends on the current time but also on the previous time samples. Delayed differential equations (DDEs) are leveraged to express the mathematical formulation of delayed dynamic systems [147],

$$\frac{dx(t)}{dt} = f\left[x(t), x\left(t-\tau\right)\right] \tag{4.11}$$

where f is a nonlinear function, and τ is the delay value. It is shown in [147] that the delayed feedback loops can perform as reservoirs which means they can form short term memories. As it was mentioned in Section 4.1, RC is a new class of RNNs which is capable of capturing the spatio-temporal correlations in the data and also map the data from low dimensional space to high dimensional space which makes the classification task of linearly non-separable date easier. The RC systems are composed of three different layers: input, reservoir, and output layer. In the input and reservoir layers, all the weights are fixed, but the output layer requires training. The reservoir layer acts as the recurrent layer of the system, and captures the temporal correlation of the input data. The extracted temporal features are then used for training the output weights.

The structure of a delayed based RC scheme is depicted in Figure 4.1. As it can be seen in

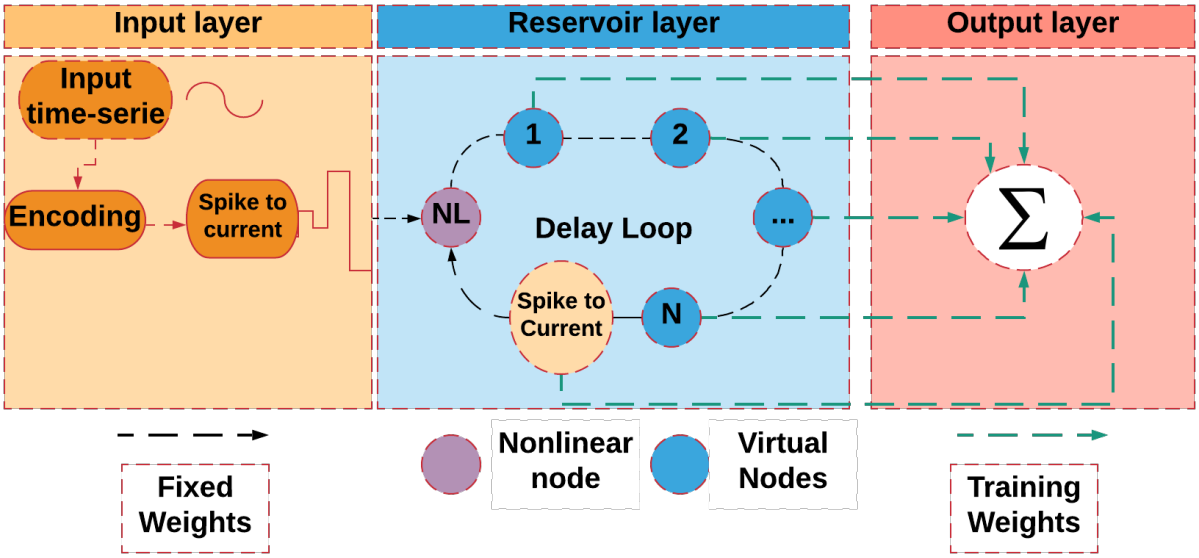

Figure 4.1. Spiking DFR Structure.

Figure 4.1, the delay loop is composed of a nonlinear node and a set of virtual nodes. The nonlinear node (NL) that we use in this chapter is a mathematical representation of spiking neurons. This model is called leaky-integrate and fire (LIF) neuron which is defined as,

$$\tau_{RC}\frac{dv(t)}{dt} = -v(t) + I(t), \tag{4.12}$$

where τ_{RC}, $v(t)$, and $I(t)$ are the time constant, membrane voltage and input current of LIF neuron, respectively. The LIF neuron generates a spike as soon as the membrane voltage exceeds a certain threshold value. The spike trains which are generated by LIF neuron go through a delay loop which is composed of virtual nodes. The virtual nodes are separated by θ which is a fixed delay value. Delayed dynamic systems can form short term memory and demonstrate high dimensional behavior only if they operate at the edge of chaotic region. Therefor, the number of the virtual nodes and the value of θ are very important parameters of SDFR which have to be tuned properly. To introduce the recurrence in this model, the

output of the last virtual node is multiplied by a feedback gain (g) and then it is added to the next input sample.

To process the input signals using spiking neurons, we first need to encode the signals. It was mentioned in Section 4.1 that the temporal encoding is the most optimal encoding which has been introduced in the literature so far. Therefore, in this chapter we formulate two different categories of temporal encoding schemes and evaluate the performance of each one in MIMO-OFDM spectrum sensing. At first, we introduce the mathematical representation of latency encoding. The latency encoding is defined in terms of the latency between the input onset and occurrence of the first spike. The intuition behind the latency encoding is that for input with higher intensity the latency is shorter and conversely. Therefore, the latency encoding is defined as follows,

$$t_i = T_{max} - ln(\eta.I_i + 1), \tag{4.13}$$

where t_i is the latency which is shown in Figure 4.2(a), T_{max} is encoding window time, I_i is the intensity of the input, and η is a scaling factor.

In ISI encoding, the information is encoded with respect to the relative distance between different spikes which exist in a spike train. Figure 4.2(b) demonstrates an example of ISI encoding scheme where D_1, D_2, and D_3 are the relative distances between the spikes. We leverage a nonlinear neuron model for ISI encoding. Equation (4.14) shows the ISI encoding strategy [36],

$$D_i = G\left(r_i, C_i, V_i\right) - G\left(r_i, C_{i-1}, V_{i-1}\right), \tag{4.14}$$

where D_i is the time distance between two consecutive spikes; G is the nonlinear ISI encoding neuron; C_i and V_i are the capacitance and threshold voltage of the i_{th} encoding neuron, respectively. The number of the ISI encoding neurons is a hyperparameter that requires

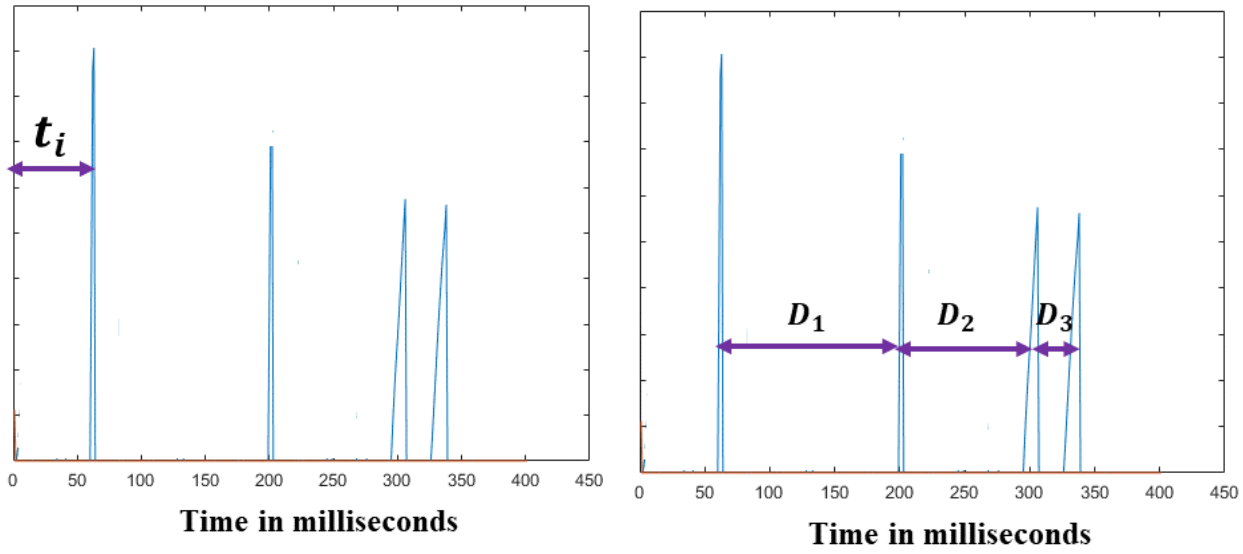

Figure 4.2. a. latency encoding, b. ISI encoding

tuning. In this chapter, for the sake of computational simplicity, the number of ISI encoding neurons are set to 3. The relationship between the number of ISI encoding neurons and the number of spikes can be expressed as $N_S = 2^{N-1}$ where N_S is the number of spikes. As it can be seen in Figure 4.1, after encoding the input signal there is a block which converts the encoded information to analog currents. This is inspired by the information processing in our brains. To process the encoded information by the adjacent information processing neurons, it is required to be converted to an analog current. Equation 4.15 is adopted to convert the encoded information to its corresponding analog current,

$$I_{PSAC}(t) = \sum_{t^j} \kappa(t - t^j) H(t - t^j), \tag{4.15}$$

where $I_{PSAC}(t)$ is the postsynaptic analog current; H is the Heaviside function; t^j is the time of spiking; and $\kappa(t - t_j)$ is a kernel function and is defined in equation (4.16),

$$\kappa\left(t - t^j\right) = V_0 \times \left(\exp\left(-\frac{t - t^j}{\tau_s}\right) - \exp\left(-\frac{t - t^j}{\tau_f}\right)\right), \tag{4.16}$$

where V_0, τ_m, and τ_s correspond to the normalization factor, slow decay, and fast decay constants, respectively. For each temporally encoded received signal, its corresponding analog current is generated using equation 4.15 and then it goes through the delay reservoir loop. In order to mimic a recurrent structure, the train of spikes fired by the LIF neuron for a given input time sample is: 1) shifted in time (delayed) by value of τ; 2) converted to an analog current using equation 4.15; 3) multiplied by a feedback gain (g); and 4) added to the current of the next time sample of the time-series data.

The final layer is the output layer which is the only layer of the introduced model that undergoes a training. The hidden states that are extracted in the reservoir layer are now used to estimate the final outputs. The hidden reservoir states are expressed as,

$$h_t = f[g.h_{t-1} + W_{in}^{res} x_t], \tag{4.17}$$

where h_t is the hidden state at time t, f is a nonlinear function which in this chapter is a LIF neuron, g is the feedback gain, W_{in}^{res} is the set of randomly generated input weights that are fixed and require no training, and x_t is the input at time t. The desired output is estimated using the hidden reservoir states as,

$$\hat{y}_t = W_{res}^{out} h_t + W_{bias}^{out}, \tag{4.18}$$

where $\hat{y}_t$ is the estimated output at time t, and W_{res}^{out} is the set of weights which map the hidden reservoir states to the output, and W_{bias}^{out} is the bias training weights. The goal of the output layer is to minimize the difference between the estimated output and the desired output ($y(t)$) for each time sample, t. The mean square error (MSE) is the most common

objective function and is defined as,

$$MSE = \frac{1}{T_{train}} \sum_{t=1}^{T_{train}} ||y_t - \hat{y}_t||^2, \tag{4.19}$$

where T_{train} is the total time length of the training data. Using MSE as the objective function makes the training time very long. Therefore, we introduce to leverage a MLP with cross-entropy as the objective function to estimate the output layer weights (W_{res}^{out}). Cross-entropy is defined as follows,

$$\zeta = cross - entropy = \sum_{t=1}^{T_{train}} [y_t \log(\hat{y}_t) + (1 - y_t) \log(1 - \hat{y}_t)]. \tag{4.20}$$

Let $\omega = [W_{res}^{out}, W_{bias}^{out}]$ define the set of weights that minimize the cross-entropy,

$$\omega = \underset{\omega}{\text{argmin}}\ \zeta. \tag{4.21}$$

To achieve the optimal ω, we train a MLP with 20 neurons in the hidden layer and a *sigmoid* function as the activation function and gradient descent algorithm,

$$\begin{aligned} W_{res}^{out}(n+1) &\leftarrow W_{res}^{out}(n) - \alpha \frac{\partial \zeta(Y, \hat{Y})}{\partial W_{res}^{out}} \\ W_{bias}^{out}(n+1) &\leftarrow W_{bias}^{out}(n) - \alpha \frac{\partial \zeta(Y, \hat{Y})}{\partial W_{bias}^{out}} \end{aligned} \tag{4.22}$$

where $W_{res}^{out}(n)$, $W_{bias}^{out}(n)$ denote the output and bias weights at iteration n. $Y = [y_1, y_2, ..., y_{T_{train}}]$, $\hat{Y} = [\hat{y_1}, \hat{y_2}, ..., \hat{y_{T}}_{train}]$, and α is the learning rate.

4.2.2 Stacked Deep Spiking DFRs

The introduced model in Section 4.2.1 is only capable to capture the temporal correlations. However, in the cooperative spectrum sensing scenarios where there are multiple SUs

in different locations the initial model cannot completely exploit the spatial information. Therefore, it is necessary to extend our introduced model in space domain as well. Figure 4.3, depicts the stacked spiking DFR (SSDFR) model which is extended in space domain to perform cooperative spectrum sensing where there are multiple SUs.

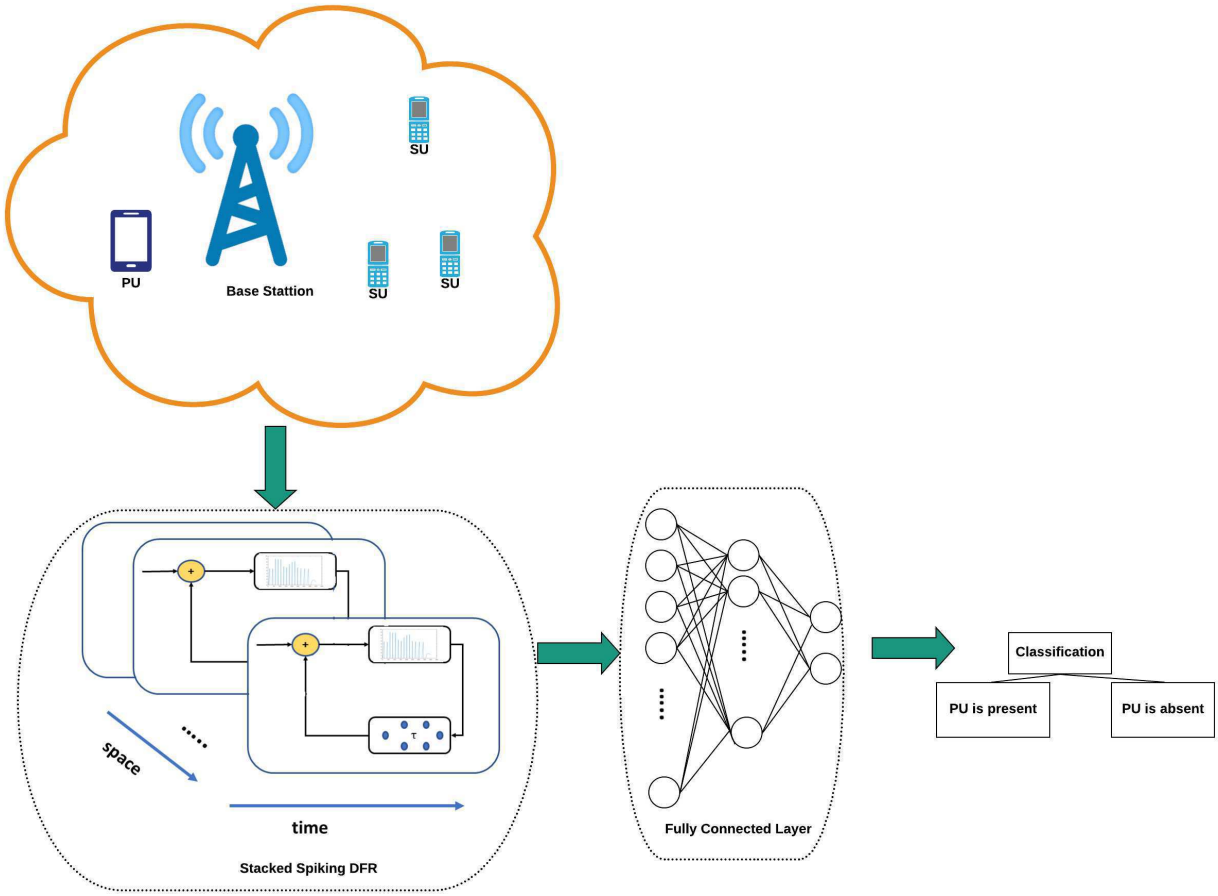

Figure 4.3. Structure of SSDFR.

SSDFRs are adopted to extract the hidden spatial features of the received MIMO-OFDM signals. There are two major steps in training the SSDFR: unsupervised pretraining, and classification. In the unsupervised pretraining phase, the SSDFR network is partitioned into

independent SDFRs and the combined spatial-temporal hidden features are calculated as,

$$h_t^l = f[g.h_{t-1}^l + \sum_{\substack{p=1 \\ p \neq l}}^{P} W_{l,p}^{spatial} h_{t-1}^p + W_{in}^{res}.x_t^l], \tag{4.23}$$

where h_t^l is the hidden state of the lth SDFR at time t, $P = L - 1$ and L is the total number of SDFRs, $W_{l,p}^{spatial}$ is the connection weight between the pth and lth SSDFRs, and x_t^l is the input of the lth SDFR. In the SSDFR model, the output is estimated as,

$$\hat{y_t} = W_{res}^{out} H_t + W_{bias}^{out}, \tag{4.24}$$

where $H_t = [h_t^1, h_t^2 ... h_t^L]$. W_{res}^{out} and W_{bias}^{out} are estimated using equation 4.22 and a fully connected MLP with 20 neurons in its hidden layer. The number of neurons in the hidden layer significantly affects the performance. To identify the optimal number of neurons, we did hyperparameter tuning. We will show in Section 4.4 that the introduced SSDFR model is very successful to capture the spatial correlation between multiple SUs and the detection performance is improved compared against other techniques without causing much computational overhead.

4.3 Synthesizing MIMO-OFDM Symbols Using GAN

The GANs are composed of two competing deep neural networks (DNN) that play a min-max game together. The first DNN is called a generator network which its task is to map the noise to the MIMO-OFDM symbols. The second DNN is a discriminator which tries to discriminate between the fake samples which are synthesized by the generator and the real samples. These two DNNs play a min-max game until both networks achieve a steady state where none of their loss functions changes significantly anymore. The objective of

the generator is to synthesize the fake MIMO-OFDM symbols such that the discriminator cannot distinguish whether they are real or fake. The output of the discriminator is 0 or 1 depending on whether the input of the discriminator is real or fake. However, at the steady state the output of the discriminator for either of the fake or real classes should be equal to 0.5. This means that at the steady state the generator is synthesizing samples that are very similar to the real samples. The generator and the discriminator play a min-max game which can be mathematically formulated as follows,

$$\begin{aligned} L_D &= \max_{\mathbf{D}} E_{x\sim p_x(x)}\left[\log\left(D(x)\right)\right] + E_{z\sim p_z(z)}\left[\log\left(1-D\left(G(z)\right)\right)\right] \\ L_G &= \max_{\mathbf{G}} E_{z\sim p_z(z)}[\log(D(G(z)))]. \end{aligned} \tag{4.25}$$

The first version of GAN introduced in [178] was not subject to any conditions. However, in some scenarios such as supervised learning the synthesized data by the GAN has to be subject to some conditions such as the labels of the data. In these scenarios another version of GANs which is called conditional GAN (cGAN) was introduced [179]. In the cGANs the fake data is generated with respect to its corresponding condition. In our case this condition can be defined as the state of the subcarrier. The fake data is synthesized with respect to its corresponding label which is 0 for idle state and 1 for busy state. The formulation of cGAN is expressed as follows,

$$\begin{aligned} L_D &= \max_{\mathbf{D}} E_{x\sim p_x(x)}\left[\log\left(D(x|c)\right)\right] + E_{z\sim p_z(z)}\left[\log\left(1-D\left(G(z|c)\right)\right)\right] \\ L_G &= \max_{\mathbf{G}} E_{z\sim p_z(z)}[\log(D(G(z|c)))] \end{aligned} \tag{4.26}$$

where c is the corresponding label of the subcarrier. In this chapter we use cGAN to synthesize fake data to improve the performance of introduced algorithm while the training data is limited. We assume that we have only 25 training samples available for training and use this limited data to train the cGAN to compensate the performance loss. In Figure 4.4, a

MIMO-OFDM system supplemented with cGAN and SDFR as the spectrum sensing platform is depicted where cGAN is used to enlarge the size of the training data. The S/P block

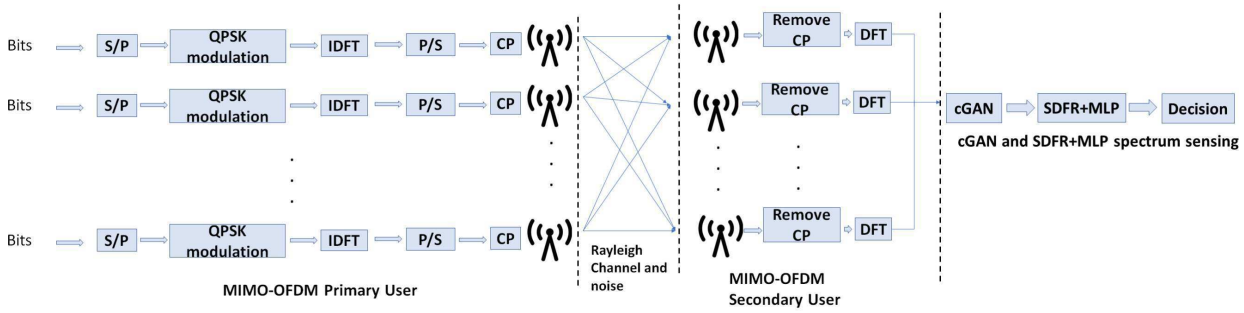

Figure 4.4. System Model of MIMO-OFDM spectrum sensing using GAN and SDFR+MLP.

in Figure 4.3, is a serial to parallel block which performs the subcarrier mapping of input bits and pilot carriers. After passing through the S/P block, the input bits are modulated using QPSK. Moreover, the modulated symbols are mapped from the frequency domain to the time domain using the IDFT block. The time domain symbols pass through a parallel to serial (P/S) block and consecutively the CP is added. The OFDM symbols are transmitted over the Rayleigh fading channel using the multiple antennas that exist at the PU. At the SU, there are also multiple antennas. The CP of the received OFDM symbols are removed and then the DFT of the OFDM symbols are calculated. As we will show in Section 4.4, 25 training samples are not sufficient and the detection accuracy of our introduced algorithm significantly drops. Therefore, we use these 25 received symbols to train a cGAN to synthesize more training samples. In order to evaluate the quality of our synthesized samples, we consider different situations.Train on synthetic and test on real (TSTR), train in real and test on synthetic (TRTS), train on real and test on real (TRTR), and train on synthetic and test on synthetic (TSTS) are studied and it is shown that they all show similar performances which implies that the quality of synthesized data is very high and similar to the real data.

4.4 Simulation Results

4.4.1 Latency VS ISI

To evaluate the performance of our introduced method, we adopt receiver operating characteristic (ROC) curve. A method which achieves the highest area under curve (AUC) in the ROC curve, is the best MIMO-OFDM spectrum sensing technique.In detection problems, the threshold of the activation function is chosen according to the ROC curve. The optimal threshold is a point where $TP = 1 - FP$. TP and FP correspond to the true positive and false positive values. We introduced latency and ISI as two temporal encoding schemes of the spikes. In this part, we compare them in terms of AUC and identify the optimal encoding approach. In Table 4.1, the the AUC of the SDFR for latency and ISI is presented. As it can be seen, ISI achieves higher AUC than latency encoding. This is mainly due to two reasons. The first reason is that in latency encoding the information is encoded with respect to the onset of input and this requires an external temporal reference. On the other hand, in ISI encoding the information is encoded in terms of the relative temporal distance between multiple spikes. Therefore, a temporal reference with respect to the input onset is not required in ISI encoding. The temporal reference has to be chosen very precisely, otherwise it can cause performance drops. The second reason is that, in several studies ISI encoding has shown to convey more information than latency encoding [180]. Our simulation results confirm this superiority too. In the rest of this chapter, we will use ISI.

Table 4.1: TTFS vs ISI, Tx = 2 , Rx = 2

Method	SNR(db)	AUC
SDFR-Latency	-10dB	0.92
SDFR-Latency	-20dB	0.7
SDFR-Latency	-30dB	0.51
SDFR-ISI	-10dB	**0.97**
SDFR-ISI	-20dB	**0.76**
SDFR-ISI	-30dB	**0.54**

4.4.2 Comparison With Other Methods

We perform our simulations for different SNRs(dB), and number of antennas and the simulation results show that the introduced hybrid SDFR and MLP training algorithm outperform the other approaches. First, we perform our experiment at SNR(dB)=0dB, and all the methods regardless of the number of antennas achieve 100% of AUC. Then, we compare the performance of hybrid SDFR+MLP against deep long-short term memory (LSTM) which is a well-known RNN, support vector machines (SVMs) with radial basis function (RBF) kernel that has already shown to be effective for spectrum sensing, and square-law combining (SLC) which is a traditional energy detection based method at lower SNRs(dB). In this part, we assume that there is only one SU and one PU in the DSS environment. In this section, we show a single layer SDFR's capability to extract the temporal correlation between the received signals and use that information to identify available spectrum holes at low SNRs(dB). In Figure 4.5, the ROC curve of our simulation results at SNR(dB) = -20dB for different numbers of transmit and receive antennas are presented.

As it can be seen in Figure 4.5 for all the scenarios, the hybrid SDFR+MLP training approach outperforms the SLC and SVM because SDFR+MLP covers more AUC compared with the other two approaches. We assume that the channel is a fading Rayleigh channel. In Table 4.2, the AUC for other SNRs(dB) are presented as well. Table 4.2 shows that for SNR(dB)=-10dB regardless of the number of transmit and receive antennas the hybrid SDFR+MLP and the SLC energy detection based approach achieve equal performances. This observation implies that for -10dB$\leq$ SNR(dB), there is no need to use learning based techniques for MIMI-OFDM spectrum sensing. However, as the SNR(dB) goes below -10dB the performance of the energy detection drops significantly and at these low SNRs(dB) the learning based techniques achieve much higher AUCs compared with classical energy detection approach. It can be observed that at low SNRs(dB) even SVM which is not able to extract the temporal

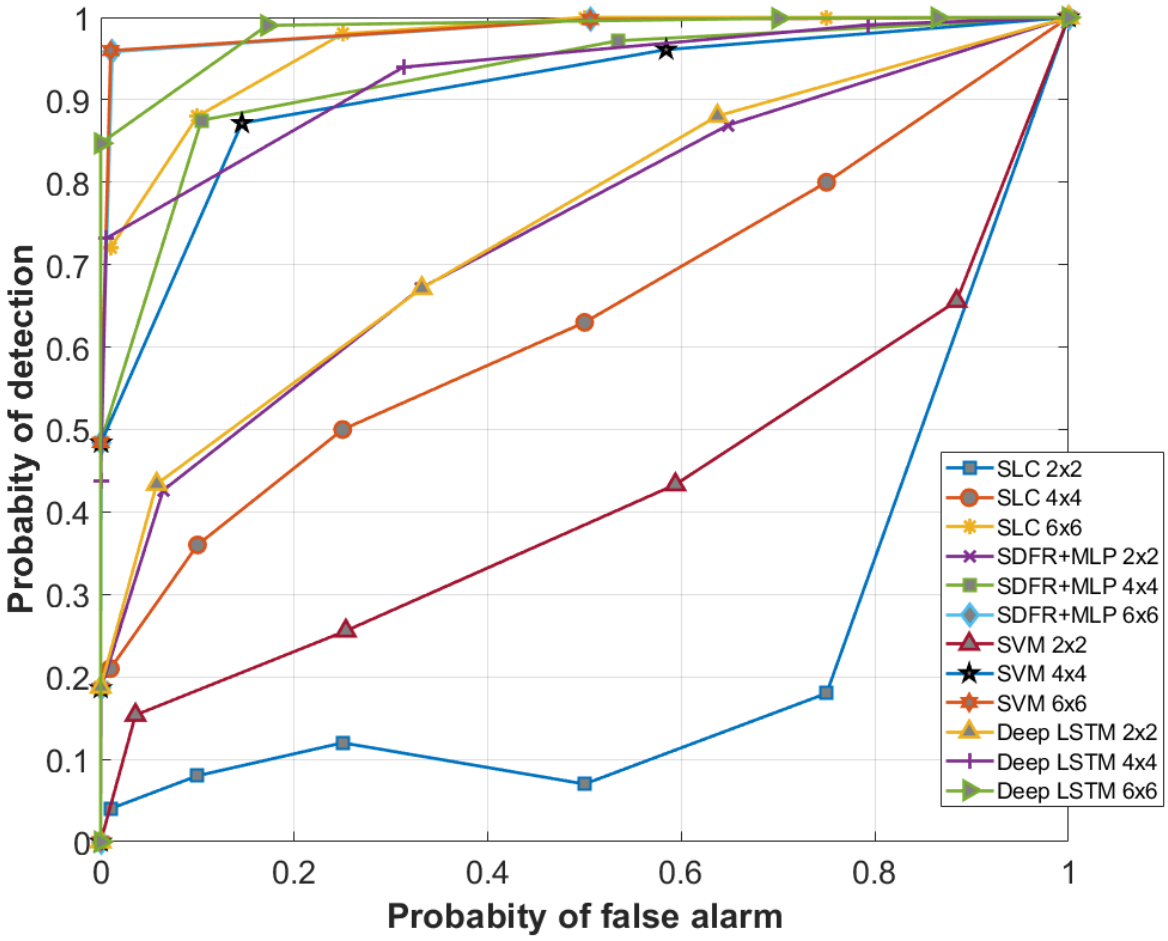

Figure 4.5. ROC curves for different sensing approaches and different number of antennas at SNR(dB) = -20dB .

Table 4.2: AUC of Different MIMO-OFDM Spectrum Sensing Methods at Different SNR(dB)

Method	SNR(dB)	Tx antenna	Rx antenna	AUC
SLC	-10dB	2	2	0.98
SLC	-10dB	4	4	1
SLC	-10dB	6	6	1
SVM	-10dB	2	2	0.95
SVM	-10dB	4	4	1
SVM	-10dB	6	6	1
Deep LSTM	-10dB	2	2	0.98
Deep LSTM	-10dB	4	4	1
Deep LSTM	-10dB	6	6	1
SDFR+MLP	-10dB	2	2	0.98
SDFR+MLP	-10dB	4	4	1
SDFR+MLP	-10dB	6	6	1
SLC	-20dB	2	2	0.11
SLC	-20dB	4	4	0.7
SLC	-20dB	6	6	0.97
SVM	-20dB	2	2	0.71
SVM	-20dB	4	4	0.92
SVM	-20dB	6	6	0.98
Deep LSTM	-20dB	2	2	**0.75**
Deep LSTM	-20dB	4	4	**0.95**
Deep LSTM	-20dB	6	6	**0.99**
SDFR+MLP	-20dB	2	2	**0.75**
SDFR+MLP	-20dB	4	4	**0.95**
SDFR+MLP	-20dB	6	6	**0.99**
SLC	-30dB	2	2	0
SLC	-30dB	4	4	0
SLC	-30dB	6	6	0
SVM	-30dB	2	2	0.46
SVM	-30dB	4	4	0.57
SVM	-30dB	6	6	0.62
Deep LSTM	-30dB	2	2	**0.55**
Deep LSTM	-30dB	4	4	**0.62**
Deep LSTM	-30dB	6	6	**0.71**
SDFR+MLP	-30dB	2	2	**0.55**
SDFR+MLP	-30dB	4	4	**0.62**
SDFR+MLP	-30dB	6	6	**0.71**

correlation, outperforms the energy detection based method. However, the learning based techniques undergo a training process which adds up some overhead compared with the classical approaches. Therefore, at high SNRs(dB) it is more efficient to use the classical methods, and use the learning based techniques only at low SNRs(dB). The average AUCs of SLC, SVM, deep LSTM, and SDFR+MLP for all the SNRs(dB) and all the transmit and receive antennas configurations are equal to **0.53**, **0.8**, **0.84**, and **0.84** respectively. The deep LSTM and hybrid SDFR+MLP achieve the same AUC in this scenario. However, in the deep LSTM 30 neurons are adopted in the recurrent layer which makes the deep LSTM model inefficient both in terms of computational complexity and the power consumption of hardware implementation. Our introduced SDFR model achieves the same performance of deep LSTM while the number of the neurons are much less. There is only one spiking neuron in the SDFR which is intrinsically energy efficient. To further evaluate the effect of larger number of antennas at low SNRs(dB), we increase the number of Tx, and Rx to 14. However, SLC still fails (AUC=0). This observation implies that at very low SNRs(dB) increasing the number of antennas cannot help traditional methods like SLC. Therefore, it is required to adopt the techniques which rely on temporal information.

4.4.3 SSDFR Performance for Cooperative Spectrum Sensing

In Section 4.2, we introduced SSDFR as an approach to extract the spatial and temporal correlations simultaneously while there are multiple SUs in a DSS environment. We assume that there are totally three SUs and one PU in the given DSS environment. The SUs are located in positions that sense MIMO-OFDM signals with -20dB, -25dB, and -30dB SNRs(dB), respectively. We compare the performance of SSDFR against a SVM with RBF kernel, deep LSTM, and deep CNN in terms of AUC and the training overhead. The CNNs have shown to be very effective in image classification and recently have been introduced to

be used in spectrum sensing as well [64]. Figure 4.6, shows the AUCs of the aforementioned

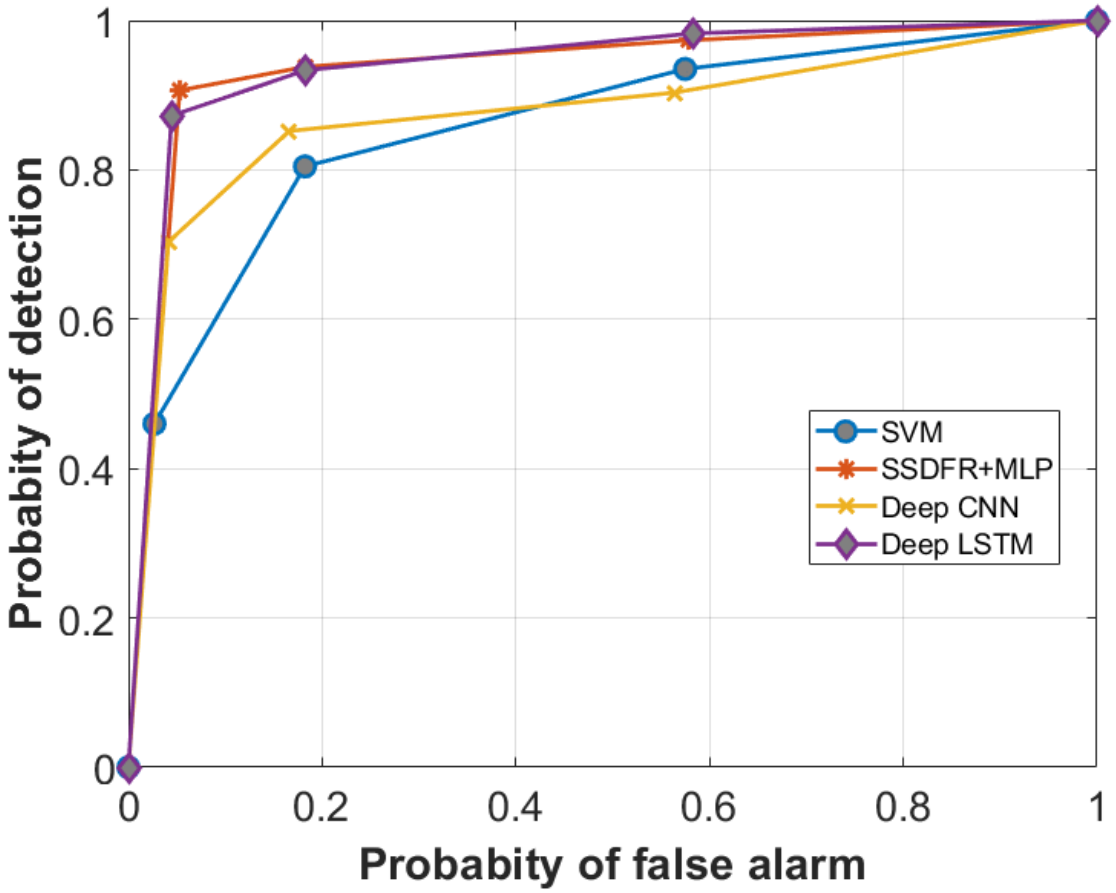

Figure 4.6. Average ROC curves of different methods in cooperative DSS environment.

methods. SSDFR+MLP, deep LSTM, deep CNN, and SVM achieve 0.96, 0.96, 0.89, and 0.88 of average AUCs, respectively. In the cooperative DSS the performance of all methods improve. However, the improvement that is brought by the hybrid SSDFR+MLP is more significant. The performance of hybrid SSDFR+MLP is improved by **12**%. On the other hand the performance of SVM is improved by 8%. This because the SVMs cannot exploit the spatio-temporal correlation as much as SSDFR+MLP and deep LSTM. In Section 4.4.2, the LSTM could achieve its maximum performance when the number of recurrent units was 30. However, in the cooperative scenario with this number of recurrent units the deep LSTM cannot achieve its maximum performance. In this part, we perform hyperparameter tuning

to identify the optimal number of recurrent units such that the deep LSTM can achieve its maximum performance. To achieve the best performance by the deep LSTM, 100 recurrent neurons are required. In the hybrid SSDFR+MLP, only three stacked layers are adopted while we achieve equal performance with deep LSTM. This observation shows the significant computational complexity superiority of SSDFR over deep LSTM. In Section 4.4.4, we will make a comprehensive comparison between the computational complexities of different spectrum sensing approaches. The CNN achieves slightly better performance than SVM due to its capability to extract the spatial correlations. However, the temporal correlations are completely underutilized while CNN is leveraged for spectrum sensing.

4.4.4 Computational Complexity Analysis

The computational complexity of the SLC is the lowest, as it does not require any training. However, it suffers from very low performances at low SNRs(dB). Therefore, in this section we compare the computational complexities of SSDFR+MLP, deep LSTM, deep CNN and SVM with RBF kernel in the cooperative spectrum sensing scenario. In the learning based approaches, the computational complexity corresponds to the training time [69]. All our simulations are run on an Intel(R), Core (TM)-i5, @3.4 GHz.

Table 4.3: Computational Complexity Analysis

Algorithm	Training Time
deep LSTM	189s
hybrid SSDFR+MLP	62
deep CNN	57s
SVM	1s

As it can be seen in Table 4.3, the training time of the hybrid SSDFR+MLP method is much lower than the deep LSTM while they achieve the same performance in terms of AUC. This is mostly because the unsupervised feature extraction in SSDFR does not require any

training. On the other hand, the end-to- end system of the deep LSTM is supervised and all the weights undergo training. Another factor that causes this meaningful difference between the computational complexities of SSDFR+MLP and deep LSTM, is the number of the neurons that are adopted in the feature extraction layers of each model. There are only three spiking neurons in the feature extraction layer of SSDFR, but in order to achieve a similar performance we need to adopt 100 neurons in the feature extraction layer of the deep LSTM which makes the training time of the deep LSTM very long. The computational complexity of the hybrid SSDFR+MLP is mostly caused by the spike to analog current block of the feature extraction layer and training the weights of the output MLP. SVM and deep CNN have shorter training times, but their detection accuracy is significantly lower.

4.4.5 Effect of Delay on Performance

The delay value of the reservoir layer is an important hyperparameter that requires tuning. It has been shown in [181] that the delay feedback loops can form short-term memory and show high-dimensional behavior only if their delay value is tuned somehow that they can operate at the edge of chaos. The dynamic behavior of the delay loop shifts from periodic to edge of chaos and to completely chaotic. There are several evidences that the neurons of our brain also operate at the edge of chaos [182]. Therefore, it is essential to tune the delay value of our introduced model such that it can operate at the edge of chaos. Plotting the phase portraits is one common method to analyze the dynamic behavior of delay feedback loop systems.

Solving the DDE equation of a delay system helps us to further investigate the dynamic behavior. In fact, by using the phase portraits we can visualize the dynamic behavior of a delay feedback system in long term run. Figure 4.7 demonstrates an example of a phase portrait corresponding to a delay feedback system. As it can be seen, by increasing the delay

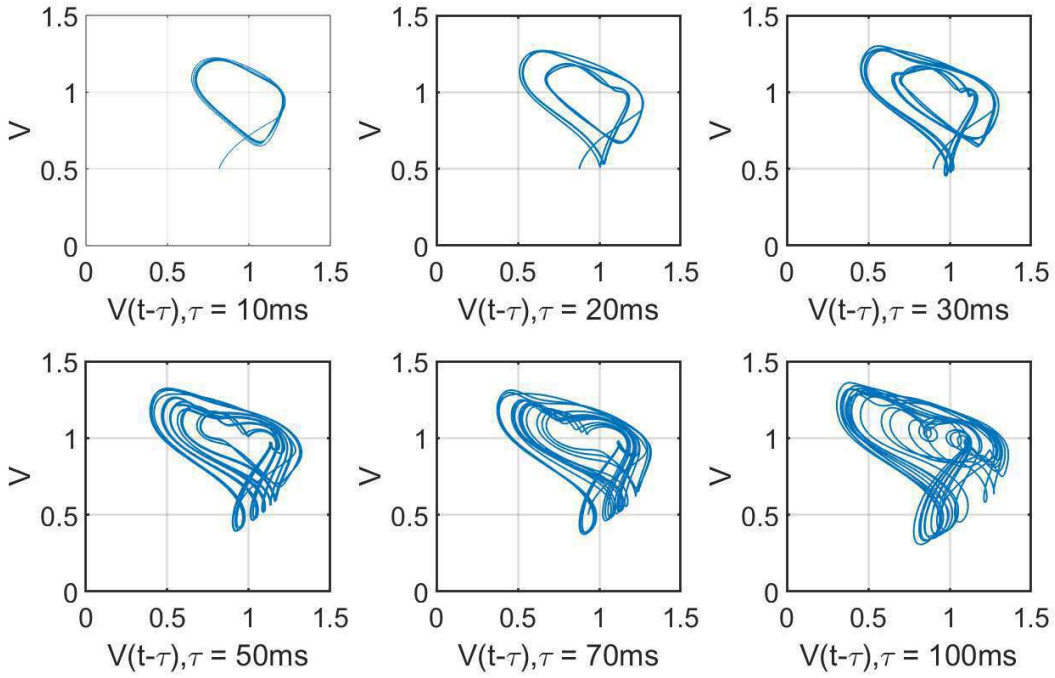

Figure 4.7. Phase Portrait.

value the dynamic behavior of a delay feedback system can shift from ordered to edge of chaos, and completely chaotic.

We investigate the effect of the delay value on SDFR and identify the optimal delay value in this section. Figure 4.8 shows the AUC of SDFR for different values of the delay. AUC is very low for small values of the delay, however, it improves as the delay increases. We can see that our introduced model achieves the highest AUC for $\tau = 30ms$. Furthermore, when $\tau > 30ms$, the AUC starts to reduce. This observation along with the phase portrait imply that $\tau = 30ms$ is the optimal delay value of our introduced model to analyze the MIMO-OFDM DSS spectrum occupancy time-series. Therefore, we can conclude that for $\tau = 30ms$ SDFR operates at the edge of chaos and can form a short-term memory and show high-dimensional behavior. Accordingly, we set $\tau = 30ms$ in this chapter [183].

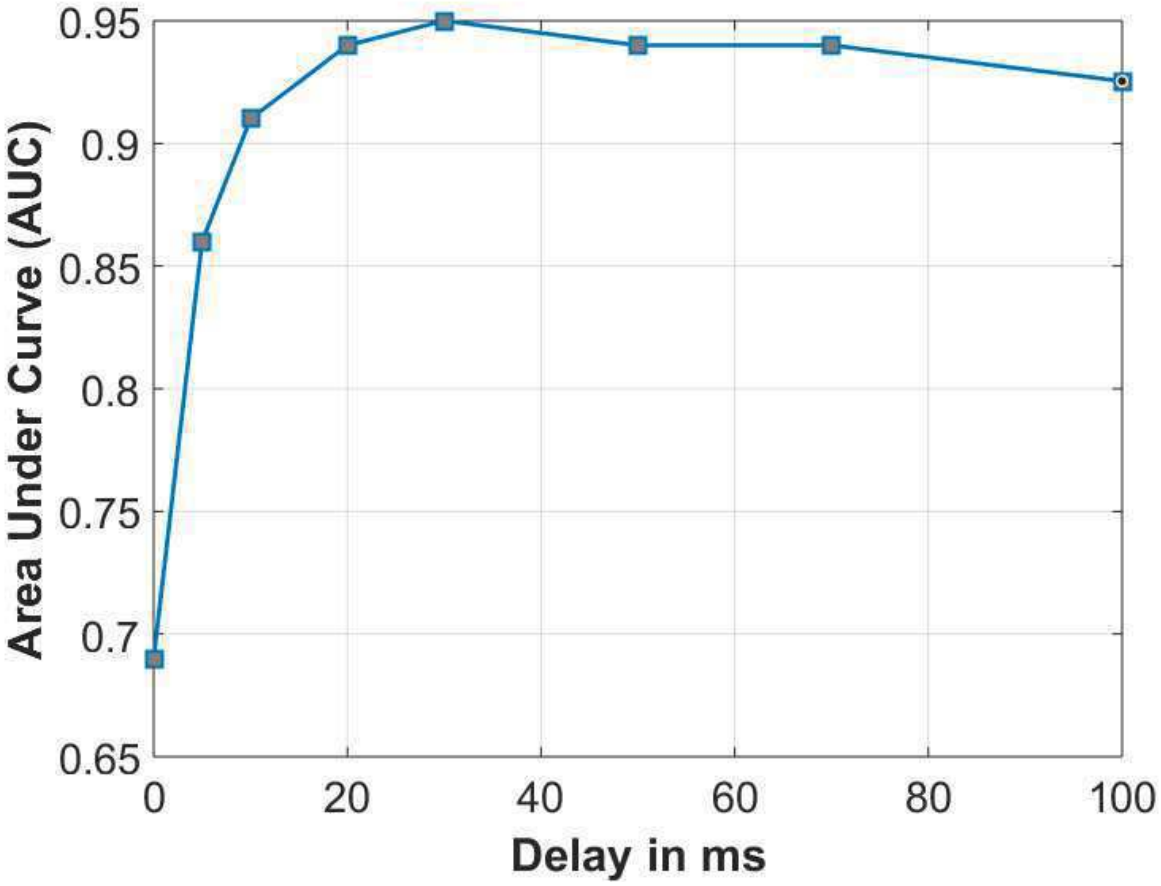

Figure 4.8. Delay effects: SNR(dB)=$-20dB$, Tx=4, Rx=4.

4.4.6 MIMO-OFDM Symbols Augmentation Using cGAN

Table 4.4 represents the AUC of our introduced approach while it is trained by 25 training samples and the test dataset is unchanged. For SNR(dB)=-10dB while we use 25 training samples the AUC does not drop. However, as the SNR(dB) goes below -10dB or the number of transmit and receive antennas is reduced the impact of a smaller training dataset is more significant on reducing AUC. In this section we introduce to use cGAN to synthesize more training data for the situations that suffer more from a small training dataset. The cGAN structure consists of two competing DNNs. The DNN corresponding to the generator network consists of two hidden layers with 20 neurons at each layer. The activation function that we

Table 4.4: AUC of hybrid SDFR+MLP spectrum sensing of 25 training samples

SNR(dB)	Tx antenna	Rx antenna	AUC
-10dB	2	2	0.97
-10dB	4	4	1
-10dB	6	6	1
-20dB	2	2	0.61
-20dB	4	4	0.85
-20dB	6	6	0.99
-30dB	2	2	0.48
-30dB	4	4	0.46
-30dB	6	6	0.7

use for all the layers of the generator is a leaky-rectified linear unit (leaky-RELU) function. The leaky-RELU function is a mapping as $f(x) = max(\alpha x, x)$ where α is leaky factor and in this chapter is set to 0.2. The DNN corresponding to the discriminator has also two hidden layers. However, there are 200 neurons at each hidden layer. In the discriminator, leaky-RELU is used in all the hidden layers, but sigmoid function is used at the output layer. The synthetic MIMO-OFDM symbols are generated in Tensorflow. Figure 4.9 demonstrates the loss functions of the discriminator and generator. The loss functions of discriminator and generator play a min-max game together. A min-max game is a game where the two parties of the game have opposite interests. In the loss function plot of the discriminator and generator, it can be observed that at the beginning of the training when the loss function of one party is maximized the loss function of other party is minimized. However, as the training continues the generator learns how to fool the discriminator and that is where we call it the steady state of the game. At the steady state the loss functions of neither of the parties change significantly. This is because that at this point the generator synthesizes some data that the discriminator is not able to specify whether it is real or fake. Figure 4.9 shows that after 1000 epochs of training the loss function of the discriminator is very close to 0.5 with not much variations.

After the cGAN is successfully trained, we use the generator to synthesize more training samples to enlarge the size of the training samples. The combination of the synthetic and

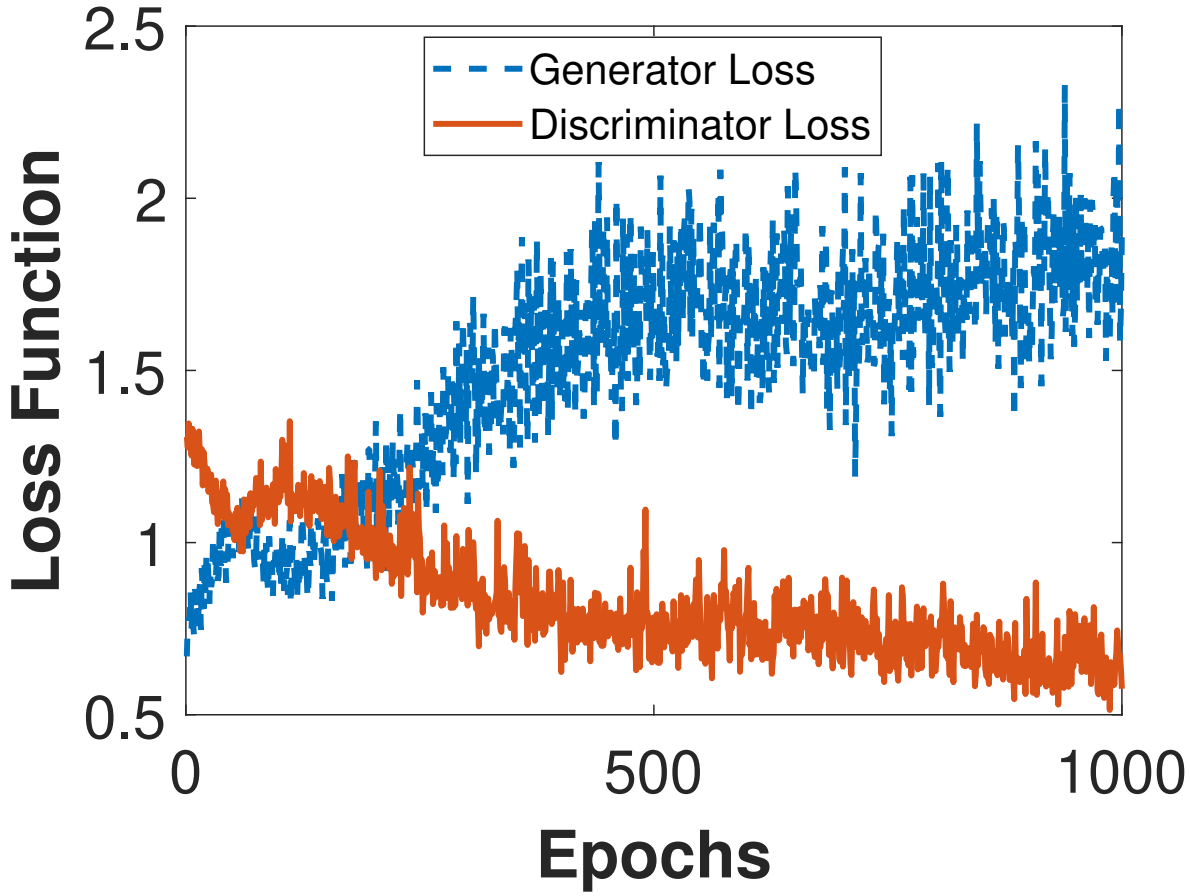

Figure 4.9. Loss Functions of generator and discriminator.

real samples is used to train our scheme. We perform the test on 5102 real samples, and the results are demonstrated in Table 4.5. We generate synthetic samples for three of the cases that are more severely affected by the limited training dataset. The AUCs of our model after retraining by the combination of the real and synthetic samples are presented in Table 4.5.

Table 4.5: AUC of hybrid SDFR+MLP spectrum sensing using synthetic and real data for training

SNR(dB)	Tx antenna	Rx antenna	AUC
-20dB	2	2	0.74
-20dB	4	4	0.93
-30dB	4	4	0.6

We can observe that after retraining our scheme the AUC is significantly improved compared with the case that the training data was very limited. In order to better evaluate the quality and expressiveness of the synthetic training samples, we compare the ROC curves of our scheme while we use real training samples with the case that the combination of real and

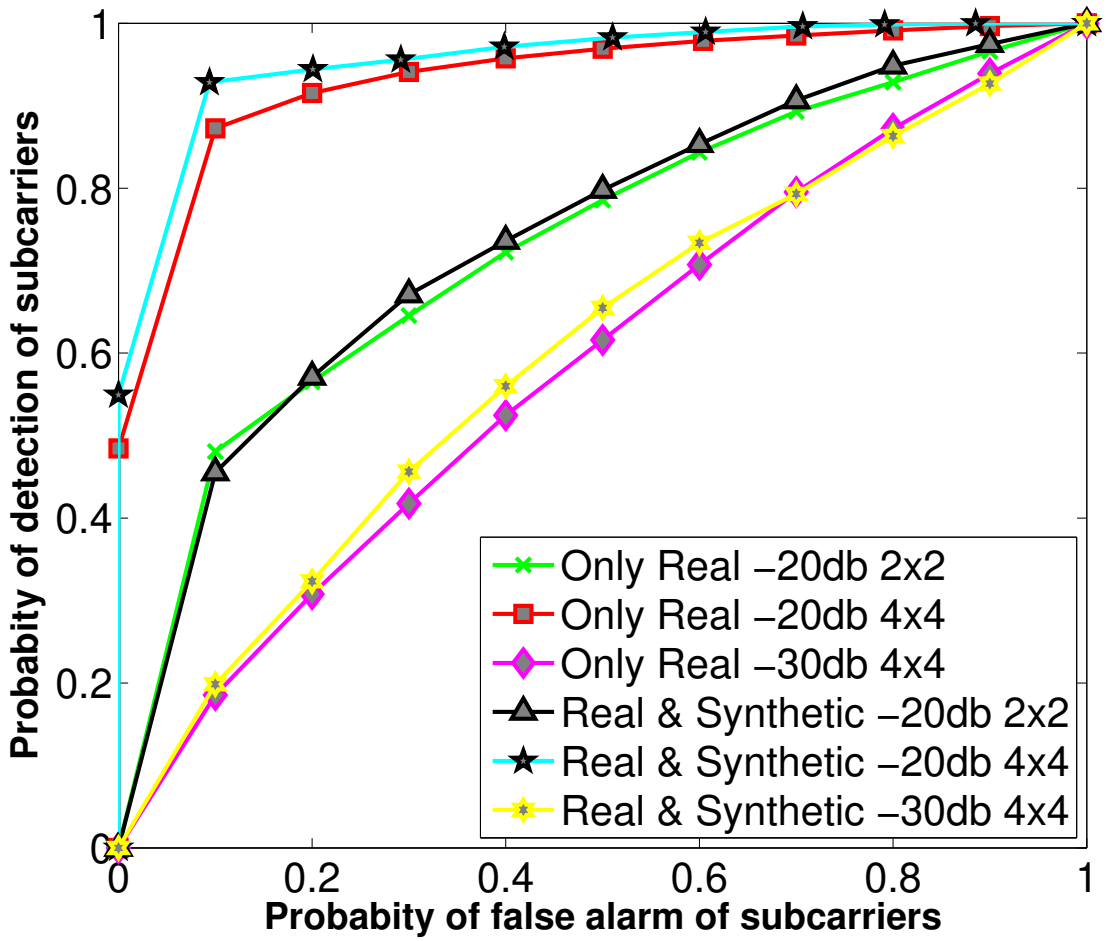

Figure 4.10. ROC curves of real samples VS combined real and fake.

The ROC curves of our introduced hybrid SDFR+MLP spectrum sensing approach demonstrate very similar performances while we use real samples and combination of real and fake samples respectively. This observation implies that the synthetic samples are very similar to the real samples and they are as expressive as the real dataset. In order to perform further analysis on the expressiveness of the synthetic dataset, we train and test our scheme in different scenarios. These scenarios are TSTR, TRTS, TRTR, and TSTS as they were described in Section 4.3. The presented results in Table 4.6 imply that the synthesized samples are very expressive and robust as in different train and test scenarios the AUCs of our introduced hybrid spectrum sensing scheme are very close together. The high quality and

Table 4.6: AUC of hybrid SDFR+MLP spectrum sensing in different scenarios using synthetic and real data for training

SNR(dB)	Tx ant.	Rx ant.	AUC (TRTR)	AUC (TSTR)	AUC (TRTS)	AUC (TSTS)
-20dB	2	2	0.75	0.73	0.74	0.74
-20dB	4	4	0.95	0.93	0.96	0.96

expressiveness of the synthesized training samples is fueled by the power of the SSDFR in capturing the spatio-temporal correlations even from a very small data. In similar works that the GANs are introduced to synthesize more training samples for spectrum sensing, it is assumed that at least 100 training samples are available to train the GANs. However, in this chapter we train cGAN to synthesize highly expressive training data while we have only 25 available samples. We have shown in this chapter that by combining SDFRs and cGANs we can reduce the number of real training samples compared against the state-of-the-art which is really important to reduce the costs of collecting training data.

4.5 Conclusion

In this chapter, we first introduced an energy efficient and easy to train spiking delayed feedback reservoir (SDFR) model to detect the spectrum availability in a MIMO-OFDM DSS environment. We showed that the introduced algorithm can effectively capture the temporal correlations of the received signals in a single SU environment. The introduced model is specially effective at low SNRs(dB) and outperforms the state-of-the-art spectrum sensing techniques in terms of the detection accuracy and computational complexity. Then, we extended the introduced model to a deep stacked SDFR (SSDFR) where multiple layers of SDFRs are developed in the space domain to capture the spatial and temporal correlations simultaneously while there are multiple SUs in a DSS environment. The superiority of the introduced algorithm over other methods is verified through extensive simulations. Two temporal neural spike encoding schemes (ISI and latency) were mathematically formulated

and we observed that ISI encoding is the optimal encoding scheme in terms of detection accuracy. The training data scarcity problem was also successfully tackled through the combination of SDFR and cGANs.

Chapter 5

Spiking Recurrent Neural Network with Novel Encoding and Defense Mechanisms

5.1 Introduction

Artificial spiking neural networks (SNNs) offer several advantages over other, non-spiking, artificial neural networks (ANNs). First, SNNs are brain-inspired mathematical representations of the spike signals which are adopted in the human brains [172, 184, 185]. This is due to: 1) the way information propagates between SNNs' units; and 2) the advantage that SNNs are intrinsically sensitive to the temporal characteristics of information transmission that occurs in the biological neural systems [186]. Second, SNNs are more power efficient. In fact, SNNs have been the main choice for neuromorphic hardware implementations, such as Loihi, TruNorth, and SpiNNaker [187–195]. Due to these advantages, SNNs have become the focus of a number of recent applications [196–199].

Despite the advantages of SNNs, their performance suffers from three major drawbacks. First, the current recurrent structures of SNNs are very difficult if not impossible to train [172, 200], making them impractical for data that has temporal and/or spatial correlations. Second, the performance of SNNs depends crucially on how the neural spikes get encoded, i.e., the encoding scheme. Two main categories have been introduced in the literature, namely, rate and temporal encoding. Although the two encoding schemes have been shown to be effective in different scenarios and applications [157, 201, 202], several questions remain unanswered. For example, is there an optimal encoding scheme? Can we combine two or more schemes to encode the spikes? Third, although the vulnerability of SNNs against adversarial attacks has been studied in the literature [203–205], an effective defense mechanism for SNNs against adversarial attacks is lacking.

This chapter aims at solving the three major drawbacks of SNNs mentioned earlier. More precisely, we provide three main contributions as follows:

- We present a recurrent structure of SNNs, which is inspired by delayed feedback reservoirs (DFRs) [114, 147, 206], to capture the temporal correlation of time-series data. DFRs are a subcategory of reservoir computing (RC), which is a new class of recurrent neural networks (RNNs) and is easier to train [20, 207]. In the reservoir layer of DFRs, only one neuron is required and the recurrence is modeled using a delay layer. In this chapter, we develop a spiking DFR (SDFR) model and show its effectiveness (in terms of classification accuracy) in capturing the temporal correlation of time-series data.

- For the first time, we introduce the idea of multiplexing two encoding schemes in artificial SNNs. More specifically, inspired by recent studies in cognitive neuroscience [157], we multiplex the phase and temporal encoding of the neuron spikes. Under our SDFR model, we show that our encoding scheme achieves higher classification accuracy, is more robust to noise, and is more robust to adversarial attacks than existing encoding

schemes.

- For the first time, we introduce an effective defense mechanism for SNN against adversarial attacks. We show that using our defense mechanism, SDFR is robust against adversarial attacks.

In order to evaluate the effectiveness of our SDFR model, encoding scheme, and defense mechanism, we adopt them in two applications, namely, video-based face recognition and bad data detection in smart grids (a new generation of power systems which are developed to provide more efficient, reliable, and intelligent power transmission and distribution) [40, 41]. For video-based face recognition, we use the VidTIMIT dataset [208] and the DeepFake-TIMIT dataset [209], which is a version of VidTIMIT that has been adversarially attacked by generative adversarial networks (GANs). For smart grids, we use MATPOWER, which is a MATLAB software package, to simulate the smart grids and generate training data and testing data.

The rest of the chapter is organized as follows: Section 5.2 describes our introduced recurrent structure of SNNs, the new multiplexing encoding technique, and the defense mechanism against adversarial attacks. Section 5.3 presents the simulation results and Section 5.4 concludes this chapter.

5.2 Approach

5.2.1 Spiking Delayed Feedback Reservoir (SDFR)

Recurrent neural networks (RNNs) are the state-of-the-art artificial neural networks to process time-series data. To train RNNs easier and faster, reservoir computing (RC) was introduced [20, 207]. In RC, the network is divided into an input layer, a reservoir layer, and

an output layer, where the reservoir layer's parameters are kept fixed and not trained, and hence, making training easier. It has been shown in [4, 210] that RC networks can outperform traditional RNNs in many cases. To further ease the training of the network, delayed feedback reservoirs (DFRs) were introduced [147, 206]. DFRs are subcategory of RC system where only one neuron is required, and the recurrence is modeled using a delay layer.

Motivated by DFRs, in this chapter we introduce the spiking delayed feedback reservoir (SDFR), where the neuron required in the reservoir layer is a spiking neuron. In this chapter, we use the leaky-integrate and fire (LIF) spiking neuron [211, 212]. The SDFR network consists of three layer: input, reservoir, and output layers. The input and reservoir layers do not have trainable parameters, and are used only for temporal feature extraction. Figure 5.1 illustrates the structure of SDFR.

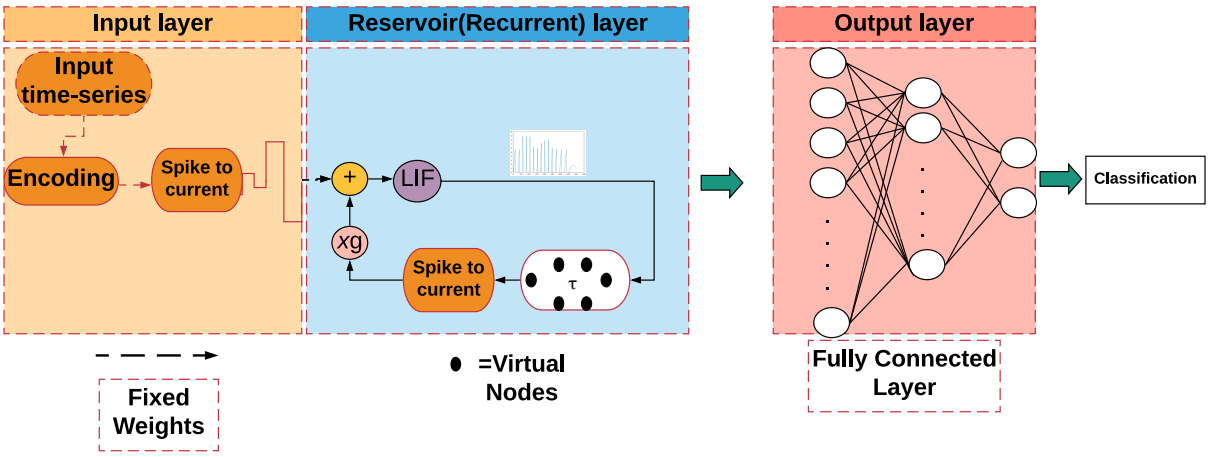

Figure 5.1. Structure of the SDFR network.

The input layer of SDFR is composed of two blocks used to pre-process the input time-series data. First, each input time sample is encoded into a train of spikes using a given encoding scheme (state-of-the-art schemes will be introduced in Section 5.2.2). The time instants at

which the spikes are fired, called $\{s\}_i$, are then used to generate an analog current:

$$I_{PSAC}(t) = \sum_{s_i} K(t - s_i)h(t - s_i), \tag{5.1}$$

where $I_{PSAC}(t)$ is the postsynaptic current of the encoding neuron, s_i is the time in which the ith spike is fired, $h(t)$ is the Heaviside function, and $K(t - s_i)$ is an exponential kernel defined as:

$$K\left(t - t_i\right) = N_0 \times \left(\exp\left(-\frac{t - t_i}{\kappa_s}\right) - \exp\left(-\frac{t - t_i}{\kappa_f}\right)\right), \tag{5.2}$$

where N_0, κ_s, and κ_f are normalization , slow decay, and fast decay constants, respectively. The ratio of $\frac{\kappa_s}{\kappa_f}$ is fixed at 4 in this chapter. For each sample of the time-series data, the input layer outputs an analog current using Equation (5.1).

The analog current generated for each time sample of the time-series data is used as an input to the LIF neuron in the reservoir layer. The LIF neuron converts the current into membrane voltage according to [211]:

$$T_{RC}\frac{dv(t)}{dt} = -(v(t) - E) + (I_{\text{noise}} + I(t))R, \tag{5.3}$$

where T_{RC} is a time constant, $v(t)$ is the membrane voltage, E is the resting potential, R is the membrane resistance, I_{noise} is the background noise, and $I(t)$ is the input current (output of the input layer) to the LIF neuron. The LIF neuron generates a spike as soon as the membrane voltage, $v(t)$, exceeds a certain threshold. The time instances at which the spikes (output of LIF neuron) are fired will be used as the features to train the output layer. In order to mimic a recurrent structure, the train of spikes fired by the LIF neuron for a given input time sample is: 1) shifted in time (delayed) by value of τ[1]; 2) converted to an analog current using Equation (5.1); 3) multiplied by a feedback gain (g); and 4) added to

[1] τ is a hyper-parameter to be fine-tuned.

the current of the next time sample of the time-series data. Each unit of delay is represented by a virtual neuron in the reservoir layer, see Figure 5.1. This process is repeated until the corresponding train of spikes of all the time-series samples are generated. We note that this feature extraction process *does not require training*.

The train of spikes (output of the LIF neuron) corresponding to each time sample in the time-series data is used an the input feature to train the output layer. In this chapter, the output layer is a fully connected multi-layer perceptron (MLP), which is trained via standard back-propagation algorithm .

5.2.2 Multiplexing Encoding Scheme

As mentioned in Section 5.2.2, in the input layer, each input sample in the time-series data is encoded into a train of spikes. In this section, we first present the state-of-the-art encoding schemes. Then, we present our novel scheme.

5.2.2.1 State-of-the-Art Encoding Schemes

The encoding process can be defined as converting the input into a train of spikes, where the number of spikes and the time instances at which these spikes are fired depend on the value of the input as well as the encoding scheme. Two main categories of encoding have been introduced in the literature, namely, rate and temporal encoding [186, 213–216]. In rate encoding, all the information about the input is encoded in the number of fired spikes. In temporal encoding, the information is encoded not only in terms of the spikes number but also the time instances at which these spikes are fired. Thus, temporal encoding has been shown to achieve better representation of the input. Therefore, we focus on temporal encoding. More precisely, we focus on the latency, inter-spike interval (ISI), and phase

encoding schemes (sub-categories of temporal encoding).

Latency Encoding: In latency encoding, the input is encoded by the time of the first spike with respect to the time at which the input onsets [157, 217].

ISI Encoding: In ISI encoding, the information about the input is encoded in terms of the temporal distance between the time instances at which output spikes are fired. If the ISI encoder has N encoding neurons, the number of output spikes, N_s, is defined as [218]:

$$N_s = 2^{N-1}. \tag{5.4}$$

The temporal distance between the output spike fired at time t_j and the output spike fired at time t_{j-1}, where $j \in \{1, \cdots, N_s - 1\}$, is named D_j and can be calculated as:

$$D_{j-1} = t_j - t_{j-1}. \tag{5.5}$$

Figure 5.2 demonstrates an example of ISI encoding with $N = 3$. Several studies have shown that ISI achieves better representation of the input compared with latency encoding [36, 219].

Phase Encoding One of the major drawbacks of latency encoding is its dependency on the accuracy of the measurement of the time difference between the input onset and the occurrence of the first spike. Inspired by cognitive science, the phase encoding resolves this issue by relying on an intrinsic internal clock of the neuron, and hence, measuring the accurate onset of the input and the occurrence of the first spike is not required [219]. In the cognitive science literature, this intrinsic clock is known as the sub-threshold membrane

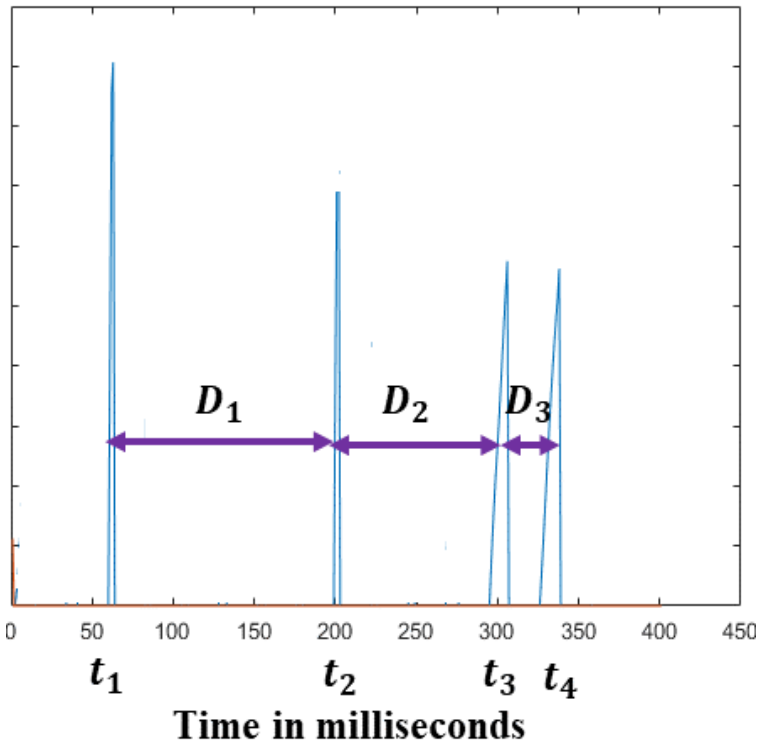

Figure 5.2. An example of ISI encoding.

oscillations (SMOs), which can be modeled as [219]:

$$\text{SMO}_i = A\cos\left(\omega t + \phi_i\right), \tag{5.6}$$

where $i \in \{1, \cdots, N\}$ for N being the dimension of the input, A is the magnitude of the oscillations, ω is the phase angular velocity, and ϕ_i is the phase whose value depends on the value of the current input sample i. Moreover, ϕ_i is defined as [219]:

$$\phi_i = \phi_0 + (i-1)\Delta\phi, \tag{5.7}$$

where ϕ_o is the reference initial phase, and $\Delta\phi = \frac{2\pi}{N}$.

The time at which a spike is fired is the time instance at which the value of SMO_i crosses

a given threshold. The value of the threshold is dependent on the sign of the input [136]. Figure 5.3 describes phase encoding.

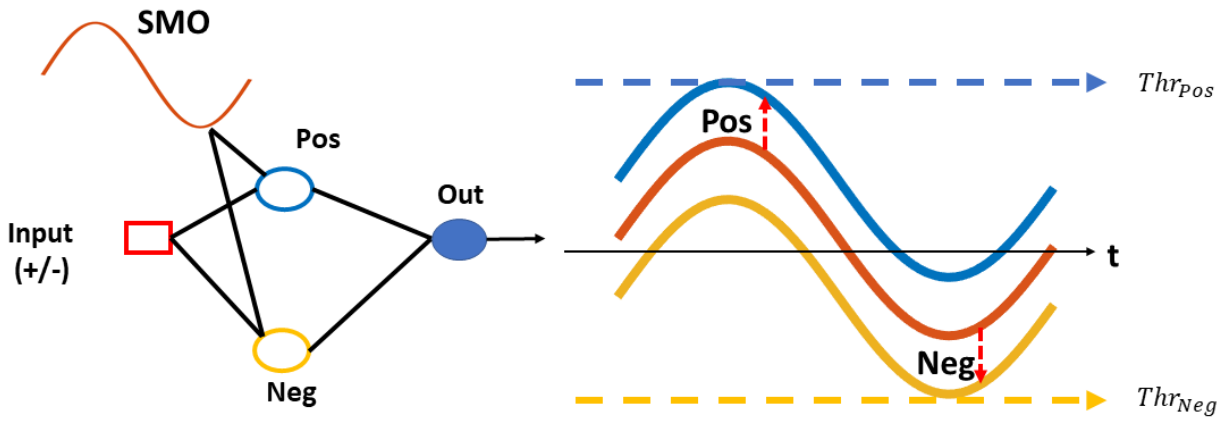

Figure 5.3. Phase Encoding.In a phase encoding unit, a positive neuron (Pos), a negative neuron (Neg), and output neuron exist. The Pos is used for encoding the stimulus values that are positive and the Neg conversely. The output neuron fires a spike at the times which the corresponding SMO of Pos or Neg crosses its threshold.

5.2.2.2 Multiplexing Phase and ISI Encoding

Despite the different encoding schemes introduced in the literature, several questions remain unanswered. For example, is there an optimal encoding scheme? Can we combine two or more schemes to encode the spikes? We introduce a multiplexing encoding scheme to answer the latter question.

Multiplexing is a process where multiple encoding schemes are combined together to increase the information which is conveyed about the input. In [202], an experiment was conducted where the rats used their whiskers to identify the texture of the surface of their surrounding environment. It was observed that the multiplexed rate and temporal codes conveyed more information about the input stimulus and the texture of the surface compared with each

individual code. The process of multiplexing was named Gamma alignment [219].

Inspired by these findings, we introduce, for the first time, a multiplexing scheme in which we multiplex the phase and ISI codes. In our multiplexing scheme, the spikes fired by the ISI code get aligned (shifted) according to the SMO of the phase code. More specifically, first, the input stimulus (e.g., a pixel of an image) is encoded using ISI encoding. The resulting output is a train of spikes fired at certain time instances. Second, the SMO of the input stimulus is generated using phase encoding. Finally, the train of spikes generated via ISI encoding are shifted to the closest points of the SMO which cross a certain threshold. The final output is the shifted time instance of the train of spikes. Figure 5.4 demonstrates an example of the multiplexing process. In this figure, ISI encoding fires three spikes at time instances t_1, t_2, and t_3, respectively. Then, these time instances are shifted to $t_1^{'}$, $t_2^{'}$, and $t_3^{'}$, according to the closest time instances at which the SMO crosses a given threshold. After multiplexing, $t_1^{'}$, $t_2^{'}$, and $t_3^{'}$ are used to encode the given input stimulus conveying information using of ISI and phase.

5.2.3 Defending SDFR Against Adversarial Attacks

Deep learning models have been shown to be vulnerable to what is called adversarial attacks. In other words, it has been shown that one can synthesize small and imperceptible perturbations of the input data, called adversarial examples or attacks, and cause the model to make highly-confident but erroneous predictions [220, 221]. This problem of adversarial attacks has garnered significant attention recently, resulting in many approaches to defend deep learning models against these attacks [220, 222, 223].

The vulnerability of SNNs against adversarial attacks has been studied in [203,204], showing that 1) SNN is not an exception of deep learning models, i.e., it is vulnerable to adversarial

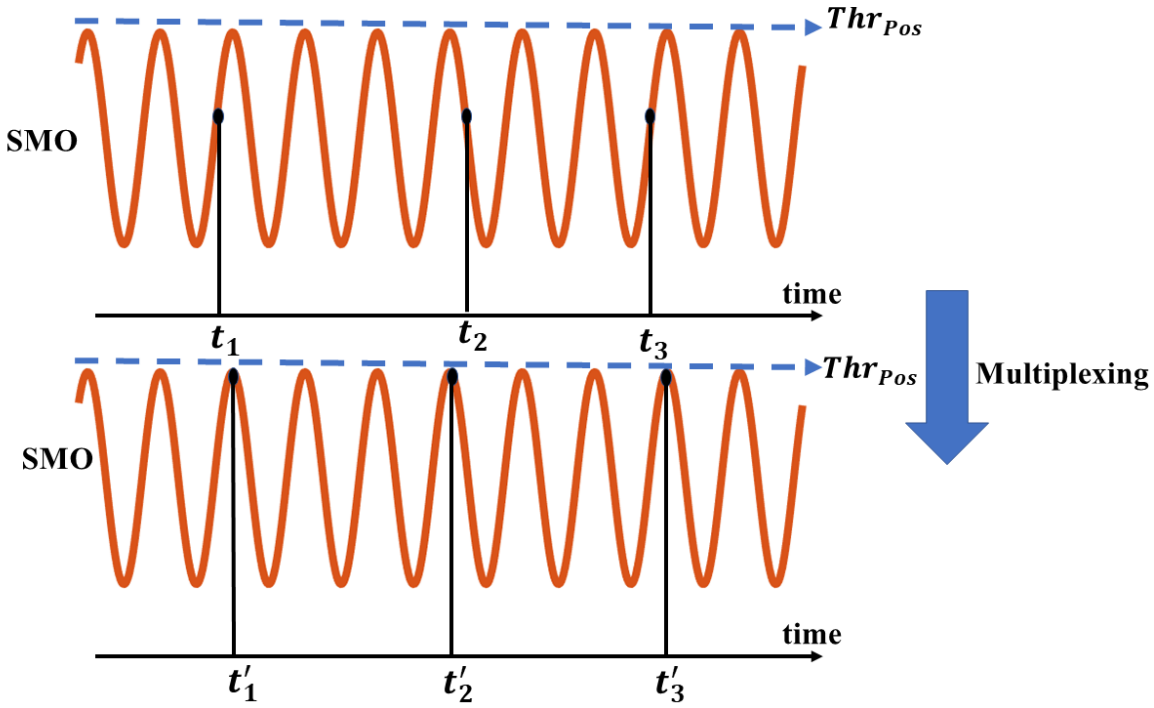

Figure 5.4. Multiplexing Phase and ISI codes.

attacks; and 2) SNNs are more resilient than DNNs against adversarial attacks. Although the latter observation supports the resilience of SNNs against adversarial attacks, without a defense mechanism, SNNs are still vulnerable. In this chapter, we introduce for the first time a defense mechanism for spiking neurons against adversarial attacks.

Generally, to make a deep learning model more robust against the adversarial attacks, its sensitivity to the variations of input should be decreased [220], [221]. The absolute value of the Jacobian of a model is a measure usually used to quantify the sensitivity of that model with respect to its input, and is defined as:

$$J = \left| \frac{\partial F(X)}{\partial X} \right|, \tag{5.8}$$

where J denotes the absolute of the Jacobian, F represents the deep learning model, X is

the input to deep model, and $F(X)$ is the output of the deep model F with respect to the input X. Thus, in order to increase the resilience of a model against adversarial attacks, its Jacobian has to be decreased.

To decrease the Jacobian of our introduced SDFR, we focus on Equation (5.3), which represents a LIF spiking neuron. The output and input of the LIF neuron are $v(t)$, $I(t)$, respectively. Therefore, the Jacobian of a spiking LIF neuron is equal to $\frac{\partial v(t)}{\partial I(t)}$. Our objective is to reduce the Jacobian of the spiking LIF neuron and that procedure is expressed as:

$$\begin{aligned}
\left|\frac{\partial v(t)}{\partial I(t)}\right| &= \left|\frac{\frac{dv(t)}{dt}}{\frac{dI(t)}{dt}}\right| \\
&= \left|\frac{\frac{-(v(t)-E)+(I_{noise}+I(t))R}{T_{RC}}}{\frac{dI(t)}{dt}}\right| \\
&= \left|\frac{\frac{-(v(t)-E)+(I_{noise}+I(t))R}{RC}}{\frac{dI(t)}{dt}}\right| \qquad (5.9) \\
&= \left|\frac{\frac{-(v(t)-E)}{RC}+\frac{(I_{noise}+I(t))}{C}}{\frac{dI(t)}{dt}}\right|, \qquad (5.10)
\end{aligned}$$

where in Equation (5.9) we used the fact that $T_{RC} = RC$ [211]. Equation (5.10) shows that to decrease $\left|\frac{\partial v(t)}{\partial I(t)}\right|$, C (the membrane capacitance of the LIF neuron) should be increased. This analysis shows that setting the membrane capacitance (C) of the LIF neuron to a large value at the training time can decrease its sensitivity to small variations of the input and consequently act as a defense mechanism to protect SNNs against adversarial attacks. We will demonstrate in Section 5.3 that this technique is effective in detecting the adversarial

attacks against SDFR.

Remark 1 *Note that our defense mechanism could be applied to any SNN using LIF neurons.*

5.3 Results and Analysis

To evaluate the performance and effectiveness of our introduced model and techniques, we apply them on two applications: 1) video-based face recognition; and 2) false data injection (FDI) detection in a cyber-physical system (smart grid). In the face recognition application, we use the VidTIMIT database which is comprised of the video recordings of subjects rotating their heads in different directions. For the FDI detection, we use MATPOWER software package of MATLAB to simulate a smart grid. The following subsections are organized as follows: first, we show the performance of our introduced SDFR network under the state-of-the-art encoding schemes; second, we show the effect of our multiplexing encoding schemes on the performance of SDFR in terms of robustness against noise and adversarial attacks; finally, we show the effectiveness of our defense mechanism. In all our simulations, the number of spikes, N_s, is set to 1 for phase codes, and 3 for both ISI and multiplexing codes. We set the values of E, and R to 0.5 Volts, and 1 Mega Ohms, respectively. We run our simulations on an Intel(R), Core (TM)-i5, @3.4 GHz using MATLAB R2018B.

5.3.1 Performance of the SDFR Network

The VidTIMIT Database: We apply SDFR on videos of 10 subjects rotating their heads from 0^o to 90^o. As an example, a few frames are shown in Figure 5.5 . The original size of the video frames are 512×384 pixels and we down-sample them to 77×58. We use the

state-of-the-art encoding scheme, i.e., ISI, to encode the pixels of each frame. The output spikes of all pixels are then used to generate one analog current according to Equation (5.1). For each subject, we use the first $\frac{2}{3}$ of the frames for training and validation, and the last $\frac{1}{3}$ for testing. To evaluate the performance of SDFR on video frames, the ***accuracy*** metric is used which is defined as,

$$\text{Accuracy} = \frac{\text{NF}}{\text{TF}}, \tag{5.11}$$

where NF is the number of frames that are classified correctly, and TF is the total number of frames.

Figure 5.5. A few frames of VidTIMIT.

Table 5.1 summarizes the accuracy of SDFR in comparison to two baseline MLPs, i.e., MLP1 and MLP2. MLP1 is a fully connected neural network with one hidden layer with 30 neurons. It is worth mentioning that MLP1 is the same as the MLP used in the output layer of SDFR. MLP2 is a fully connected neural network with two hidden layers with 80

Table 5.1: Classification accuracy of SDFR for face recognition.

Method	Accuracy
SDFR	$\approx$ **97%**
MLP1	$\approx$ 50%
MLP2	$\approx$ 93%

and 40 neurons, respectively. It can be observed in Table 5.1 that SDFR outperforms MLP1 by 47%, and hence, this shows the capability of SDFR in capturing temporal correlations. The performance of MLP2 is much higher than MLP1, yet still outperformed by SDFR by 4%. Note that the number of trainable parameters in SDFR is much less than MLP1 and MLP2. This is because the size of the input layer of MLP1 and MLP2 is equal to the number of pixels, i.e., 4466, while the size of the input layer of the MLP in SDFR is equal to the number of features extracted by the reservoir layer which is 20. In this experiment, we set $C = 10\mu$ Farads (μF), and fine tune the value of τ.

False Data Injection (FDI) Detection in Smart grids: Before we present our results on smart grids, a brief introduction to the system model of smart grids is introduced. Smart grids are new infrastructure that integrate energy with many different technologies, such as telecommunication, internet, and electronic devices [41, 224]. One of the main challenges for smart grids is the vulnerability to cyber-attacks [41], specially False data injection (FDI) attacks [225]. In a smart grid system, a control center observes a measurement vector $\mathbf{z}$ defined by [226]:

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{e}, \tag{5.12}$$

where $\mathbf{x}$ is the state vector which in this case is the voltage phase of the smart grid's buses [226], $\mathbf{H}$ is a matrix transforming the state vector $\mathbf{x}$ into measurement $\mathbf{z}$, and $\mathbf{e}$ is the environment noise.

In a smart grid system, an attacker can compromise the state estimation by injecting false data $\mathbf{a}$, in which case, the compromised measurements are:

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{a} + \mathbf{e}, \tag{5.13}$$

FDI attacks can be categorized into four categories, namely, direct-single-period, direct-dynamic, hidden-single-period, and hidden-dynamic attacks. The most challenging of them to detect is the latter [227], which is the category considered in this chapter. In hidden-dynamic attacks, the attack vector is defined as $\mathbf{a} = \mathbf{H}\mathbf{c}$, where $\mathbf{c}$ is a vector whose elements c_i vary with time according to:

$$c_i(t) = M n_i \cos\left(2\pi f_c t\right), \tag{5.14}$$

where M is the magnitude of the attack, f_c is the frequency of the attack[2], n_i is drawn from standard normal distribution, i.e., $\mathcal{N}(0, 1)$.

Now we present the performance of SDFR for FDI detection in smart grids. We generate 6000 training smart grid measurements $\mathbf{z}$, each of size 33, using MATPOWER [110] (a package in MATLAB). We divide the 6000 measurements into two sets, a safe (not attacked) set of size 5000 and a compromised (attacked) set of size 1000. The attacks are performed based on Equations (5.13) and (5.14). For each measurement $\mathbf{z}$, we assume the attacker could compromise as many of its 33 elements depending on the level of access it has to the system. For testing, we also generate another 6000 measurements and divide them into safe and compromised similarly as the training set. We train the SDFR to distinguish the safe measurements from the compromised ones. We use the F1 measure as the evaluation metrics

[2]In this subsection, $M = 1$ and $f_c = 1$.

of this application which is defined as,

$$\text{F1} = 2\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{5.15}$$

where Precision $= \frac{\text{TP}}{\text{TP+FP}}$, Recall $= \frac{\text{TP}}{\text{TP+FN}}$, and TP, FP, and FN correspond to true positive, false positive, and false negative values, respectively. Figure 5.6 demonstrates the

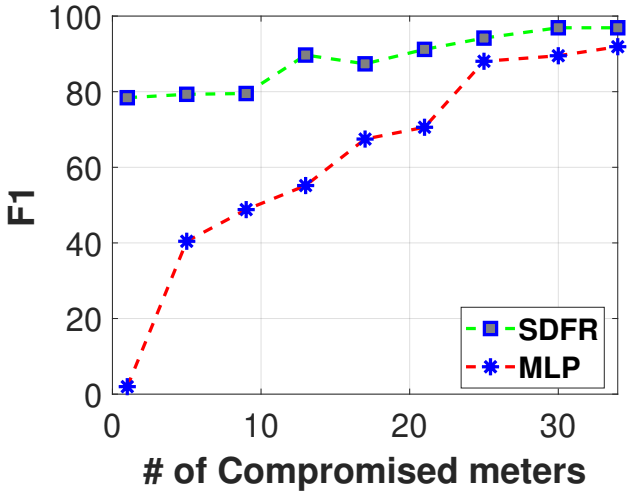

Figure 5.6. F1 value of attack detection.

comparison between the performance of SDFR and an MLP with one hidden layer with 40 neurons, which is the same MLP used in SDFR in this task. The figure shows that SDFR significantly outperforms MLP, and hence, demonstrating its capability to capture temporal correlation, despite having only one spiking neuron. In this experiment, we set $C = 10\mu$ Farads (μF), and fine tune the value of τ.

5.3.2 Effect of Multiplexing

In this section, we show that our multiplexing scheme is more robust to noise and adversarial attacks compared to state-of-the-art encoding schemes.

5.3.2.1 Robustness Against Noise:

The VidTIMIT Database: Different levels of salt & pepper noise is added to the frames of each subject. The videos of subjects without noise is used for training and the noisy frames are used for testing while the subjects rotate their heads from 0^o to 90^o. The accuracy of SDFR for each encoding mechanism with respect to different levels of noise is shown in Figure 5.7. The figure shows that the accuracy of SDFR under multiplexing is much more robust to noise compared to the accuracy of ISI or phase encoding. Our result

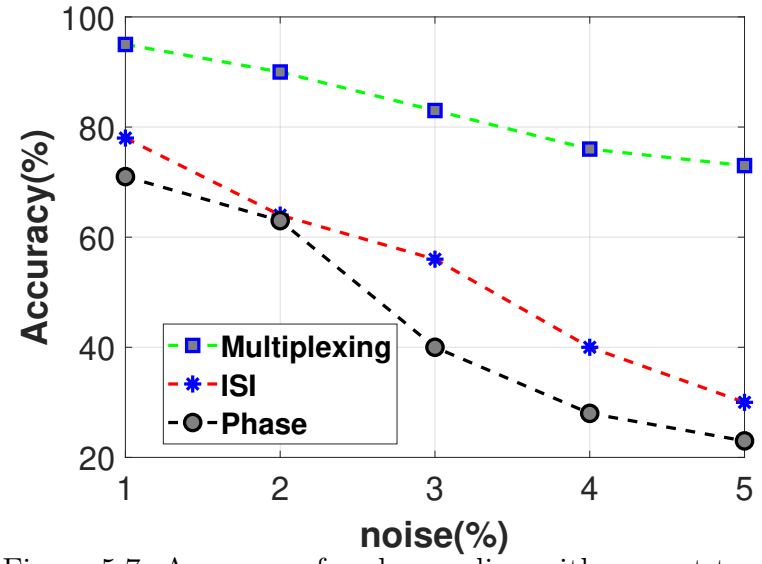

Figure 5.7. Accuracy of each encoding with respect to noise.

Table 5.2: F1 values of smart grid attack detection of SDFR for different encoding schemes.

Encoding	Variance of noise	F1
Multiplexing	1	**≈ 94.36%**
ISI	1	≈ 88.35%
Phase	1	≈ 91.49%
Multiplexing	2	**≈ 88.17%**
ISI	2	≈ 81%
Phase	2	≈ 86%
Multiplexing	10	**≈ 60%**
ISI	10	≈ 41%
Phase	10	≈ 41%

verifies the findings in the cognitive science research that the information that the biological neurons convey with respect to an input stimulus is increased through multiplexing. In this experiment, we set $C = 10\mu$ Farads (μF), and fine tune the value of τ.

False Data Injection (FDI) Detection in Smart grids: The value of noise is controlled by the variance of e. We show that as the magnitude of the variance of e becomes larger, the $F1$ value of attack detection drops. However, SDFR with multiplexing codes shows more robustness against more noise.

Table 5.2 shows the average $F1$ values of smart grid attack detection of SDFR for different encoding schemes. Note that the magnitude of attack, M, used to generate the training and testing sets is the same. Table 5.2 shows that as the variance of e increases, the $F1$ value for all the encoding schemes decreases. However, the performance of SDFR with multiplexing is improved. In this experiment, we set $C = 10\mu$ Farads (μF), and fine tune the value of τ.

5.3.2.2 Robustness Against Adversarial Attacks:

The DeepfakeTIMIT Database: Figure 5.8 shows two examples of the subjects in the DeepfakeTIMIT database, which is a version of the VidTIMIT database, which has been adversarially attacked by GANs. We train the SDFR model on the VidTIMIT database,

and test it on the DeepfakeTIMIT database. To evaluate the performance of SDFR against adversarial attacks, we use the attack success rate as the evaluation metric. The attack success rate is defined as the number of the frames that are misclassified by SDFR.

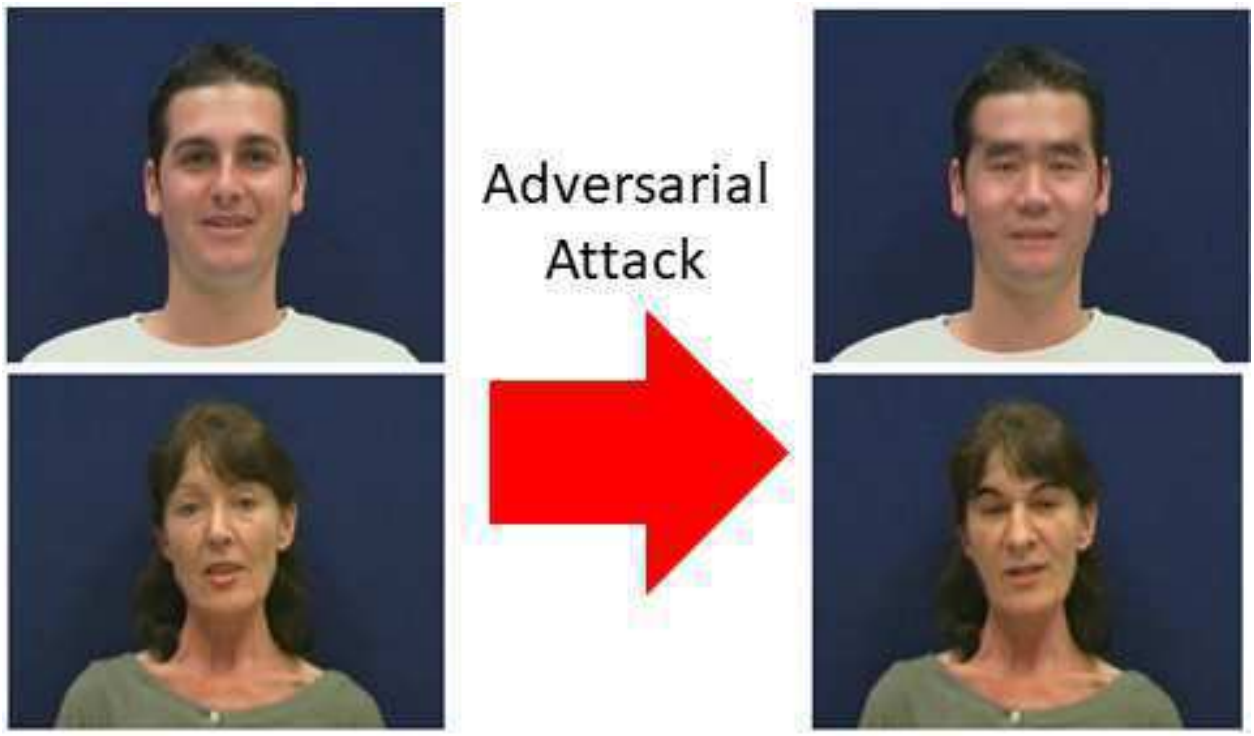

Figure 5.8. Adversarial attacks on video frames.

The performance of SDFR against adversarial attacks under different encoding schemes is shown in Table 5.3. The table shows that the adversary achieves the lowest success rate when multiplexing is adopted as the encoding scheme of SDFR. In this experiment, we set $C = 10\mu$ Farads (μF), and fine tune the value of τ.

False Data Injection (FDI) Detection in Smart grids: To perform adversarial attacks in smart grids, we assume that the magnitude of the attack ,M, in the testing data is significantly smaller than the magnitude of the attack in the training data. In other words,

Table 5.3: Success rate of the attacker while performing adversarial attacks on video frames.

Encoding	Attack success rate %
Multiplexing	**≈ 20%**
ISI	≈ 25%
Phase	≈ 34%

Table 5.4: F1 values of smart grid attack detection of SDFR for different encoding schemes.

Encoding	Attack success rate %
Multiplexing	**≈ 44%**
ISI	≈ 75%
Phase	≈ 76%

we assume the training takes into consideration the possibility of attacks, but with a certain magnitude. However, the attacker uses much smaller values, making detection of those attacks hard. More specifically, we set the magnitudes of train and test equal to $M = 10$, and $M = 1$, respectively.

The results in Table 5.4 verify the robustness of multiplexing codes over the state-of-the-art codes against adversarial attacks. In this experiment, we set $C = 10\mu$ Farads (μF), and fine tune the value of τ.

5.3.3 Defending SDFR against Adversarial Attacks

In this section, we verify the effectiveness of the introduced defense mechanism in SDFR. We showed in Equation (5.10) that the sensitivity of the Jacobian of the spiking LIF neuron can be decreased through increasing the membrane capacitance (C). In Figure 5.9, the effect of increasing the membrane capacitance of the LIF neuron (C) on defending SDFR against adversarial attacks is shown for the VidTIMIT database and smart grids.

The figure shows an interesting behavior. It can be seen that increasing C can reduce the success rate of the attacker up to a certain point. For VidTIMIT, the success rate of the attacker drops from **48%** to **12%** as we increase the capacitance from $0.1\mu F$ to $50\mu F$, then the success rate increases again. The same behavior can be observed for smart grid as well, where the success of the attacker drops from **100%** to **44%** as the capacitance is increased from $0.1\mu F$ to $10\mu F$, then the success rate increases again. This suggests a trade-off between power efficiency (decreasing C) and robustness of spiking neurons against adversarial attacks (increasing C) has to be made.

The reason why the attacker success rate increases after a certain point is due to the fact that increasing C to very large values, decays the current, i.e., $\frac{I(t)}{C} \approx 0$ according to Equation (5.3), and hence, decreasing the information conveyed through the current, which decays the representation power of the SDFR model.

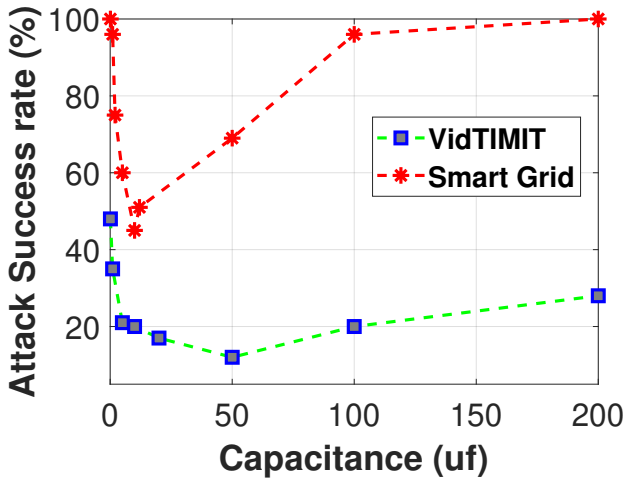

Figure 5.9. Effect of increasing the capacitance on adversarial attacks.

It can also be observed that larger capacitance values work better for VidTIMIT compared to the smart grid data. The reason is that the $I(t)$ of video frames are constituted of 4466 pixels, but the smart grid data has only 33 measurements which make the corresponding $I(t)$ of the smart grid smaller than the $I(t)$ of video frames. Therefore, a larger C is required to decay the $I(t)$ of video frames, and hence, larger C values are more effective on VidTIMIT than smart grid.

5.4 Conclusion and Future work

In this chapter, inspired by delayed feedback reservoirs, we introduced a spiking recurrent neural network, named SDFR. Our simulation results show the effectiveness of SDFR in capturing temporal correlations, compared with multi-layer perceptrons. We also demonstrated that multiplexing the state-of-the-art encoding schemes leads to more robustness against noise and adversarial attacks. Finally, for the first time, we showed that increasing the membrane capacitance of a spiking neuron (up to a certain point) can strengthen the resilience of SDFR against adversarial attacks. However, the growth of the membrane capacitance also increases power consumption, which requires a trade-off to balance.

For the future work, we will focus on: 1) verifying our proposed encoding and defense mechanisms in other existing structures of SNNs; and 2) designing more effective defense mechanisms for SNNs that do not increase the underlying power consumption.

Chapter 6

Conclusion and Open Problems

In this dissertation inspired by delayed feedback reservoirs, spiking recurrent neural network, named SDFR was introduced which was both easy to train and energy efficient. The effectiveness of this model was evaluated in solving different problems, namely, attacks detection in smart grids, MIMO-OFDM spectrum sensing, and video-based face recognition. Moreover, the model was extended in space to capture the spatial correlation as well. For the first time, cGANs were combined with SDFR to solve the data scarcity problem of AI enabled MIMO-OFDM spectrum sensing. Multiplexing different neural codes which is inspired from rats was formulated and implemented which leads to more robustness against noise and adversarial attacks. Finally, for the first time, it was shown that increasing the membrane capacitance of a spiking neuron (up to a certain point) can strengthen the resilience of SDFR against adversarial attacks.

6.1 Open Problems

- Defending SNNs against adversarial attacks without increasing the value of the membranes capaciatance of LIF neuron.
- Multiplexing several codes together, i.e., time, rate, and phase.
- Implementing an end-to-end SDFR model on FPGA.
- Implementing SDFR on multi-class tasks.
- Extending a deep SDFR model in time domain through multiple layers of SDFR.

Bibliography

[1] J. Li, C. Zhao, K. Hamedani, and Y. Yi, "Analog hardware implementation of spike-based delayed feedback reservoir computing system," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 3439–3446.

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

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

[4] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," *Neural Networks*, 2019.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[6] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.

[7] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[8] A. L. Blum and R. L. Rivest, "Training a 3-node neural network is np-complete," *Neural Networks*, vol. 5, no. 1, pp. 117–127, 1992.

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

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

[11] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

[12] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[13] I. Sutskever, J. Martens, and G. E. Hinton, "Generating text with recurrent neural networks," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, 2011, pp. 1017–1024.

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

[15] Y. Yi, Y. Liao, B. Wang, X. Fu, F. Shen, H. Hou, and L. Liu, "Fpga based spike-time dependent encoder and reservoir design in neuromorphic computing processors," *Microprocessors and Microsystems*, vol. 46, pp. 175–183, 2016.

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

[17] H. An, Z. Zhou, and Y. Yi, "Memristor-based 3d neuromorphic computing system and its application to associative memory learning," in *2017 IEEE 17th International Conference on Nanotechnology (IEEE-NANO)*. IEEE, 2017, pp. 555–560.

[18] M. A. Ehsan, Z. Zhou, and Y. Yi, "Neuromorphic 3d integrated circuit: A hybrid, reliable and energy efficient approach for next generation computing," in *Proceedings of the on Great Lakes Symposium on VLSI 2017*, 2017, pp. 221–226.

[19] H. An, Z. Zhou, and Y. Yi, "Opportunities and challenges on nanoscale 3d neuromorphic computing system," in *2017 IEEE International Symposium on Electromagnetic Compatibility & Signal/Power Integrity (EMCSI)*. IEEE, 2017, pp. 416–421.

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

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

[22] H. An, M. A. Ehsan, Z. Zhou, and Y. Yi, "Electrical modeling and analysis of 3d neuromorphic ic with monolithic inter-tier vias," in *2016 IEEE 25th Conference on*

Electrical Performance Of Electronic Packaging And Systems (EPEPS). IEEE, 2016, pp. 87–90.

[23] H. An, M. A. Ehsan, Z. Zhou, F. Shen, and Y. Yi, "Monolithic 3d neuromorphic computing system with hybrid cmos and memristor-based synapses and neurons," *Integration*, vol. 65, pp. 273–281, 2019.

[24] K. Hamedani, L. Liu, R. Atat, J. Wu, and Y. Yi, "Reservoir computing meets smart grids: attack detection using delayed feedback networks," *IEEE Trans. Ind. Informat.*, vol. 14, no. 2, pp. 734–743, 2018.

[25] K. Hamedani, L. Liu, S. Hu, J. Ashdown, J. Wu, and Y. Yi, "Detecting dynamic attacks in smart grids using reservoir computing: A spiking delayed feedback reservoir based approach," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2019.

[26] K. Hamedani, Z. Zhou, K. Bai, and L. Liu, "The novel applications of deep reservoir computing in cyber-security and wireless communication," in *Intelligent System and Computing*. IntechOpen, 2019.

[27] T. Akiyama and G. Tanaka, "Analysis on characteristics of multi-step learning echo state networks for nonlinear time series prediction," in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[28] C. Gallicchio, A. Micheli, and L. Pedrelli, "Deep reservoir computing: A critical experimental analysis," *Neurocomputing*, vol. 268, pp. 87–99, 2017.

[29] A. Katumba, M. Freiberger, P. Bienstman, and J. Dambre, "A multiple-input strategy to efficient integrated photonic reservoir computing," *Cognitive Computation*, vol. 9, no. 3, pp. 307–314, 2017.

[30] K. Bai, S. Liu, and Y. Yi, "High speed and energy efficient deep neural network for edge computing," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019, pp. 347–349.

[31] K. Bai, Q. An, L. Liu, and Y. Yi, "A training-efficient hybrid-structured deep neural network with reconfigurable memristive synapses," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2019.

[32] C. Zhao, Q. An, K. Bai, B. Wysocki, C. Thiem, L. Liu, and Y. Yi, "Energy efficient temporal spatial information processing circuits based on stdp and spike iteration," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2019.

[33] C. Zhao, L. Liu, and Y. Yi, "Design and analysis of real time spiking neural network decoder for neuromorphic chips," in *Proceedings of the International Conference on Neuromorphic Systems*, 2019, pp. 1–4.

[34] H. An, Q. An, and Y. Yi, "Realizing behavior level associative memory learning through three-dimensional memristor-based neuromorphic circuits," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2019.

[35] S. Liu and Y. Yi, "Quantized neural networks and neuromorphic computing for embedded systems," in *Intelligent System and Computing*. IntechOpen, 2020.

[36] C. Zhao, K. Hamedani, J. Li, and Y. Yi, "Analog Spike-Timing-Dependent Resistive Crossbar Design for Brain Inspired Computing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 1, pp. 38–50, 2017.

[37] J. Li, K. Bai, L. Liu, and Y. Yi, "A deep learning based approach for analog hardware implementation of delayed feedback reservoir computing system," in *2018 19th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2018, pp. 308–313.

[38] N. D. Haynes, M. C. Soriano, D. P. Rosin, I. Fischer, and D. J. Gauthier, "Reservoir computing with a single time-delay autonomous boolean node," *Physical Review E*, vol. 91, no. 2, p. 020801, 2015.

[39] R. Atat, L. Liu, J. Wu, G. Li, C. Ye, and Y. Yang, "Big data meet cyber-physical systems: A panoramic survey," *IEEE Access*, vol. 6, pp. 73 603–73 636, 2018.

[40] G. Liang, L. Zhao, F. Luo, S. Weller, and Z.Dong, "A review of false data injection attacks against modern power systems," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1630–1638, Jul. 2017.

[41] M. Ozay, I. Esnaola, F. Vural, S. Kulkarni, and H. Poor, "Machine learning methods for attack detection in the smart grid," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 48, pp. 1773–1786, Aug. 2016.

[42] M. Esmalifalak, L. Liu, N. Nguyen, R. Zheng, and Z. Han, "Detecting stealthy false data injection using machine learning in smart grid," *IEEE Systems Journal*, vol. 11, no. 3, pp. 1644–1652, 2017.

[43] J. Yan, B. Tang, N. Nguyen, and H. He, "Detection of false data attacks in smart grid with supervised learning," *In International Joint Conference on Neural Networks (IJCNN)*, pp. 1395–1402, 2016.

[44] M. Mohammadpourfard, A. Sami, and Y. Weng, "Identification of false data injection attacks with considering the impact of wind generation and topology reconfigurations," *IEEE Trans. Sustain. Energy*, vol. 9, no. 3, pp. 1349–1364, Jul 2018.

[45] J. Zhao, G. Zhang, M. L. Scala, Z. Dong, C. Chen, and J. Wang, "Short-term state forecasting-aided method for detection of smart grid general false data injection attacks," *IEEE Trans. Smart Grid*, vol. 8, no. 4, pp. 1580–1590, Jul 2017.

[46] A. M. R. Moslemi and J. Velni, "A fast, decentralized covariance selection-based approach to detect cyber attacks in smart grids," *IEEE Trans. Smart Grid*, vol. 6, no. 5, pp. 14 930–4941, Sep 2018.

[47] T. Xiong, Y.-D. Yao, Y. Ren, and Z. Li, "Multiband spectrum sensing in cognitive radio networks with secondary user hardware limitation: Random and adaptive spectrum sensing strategies," *IEEE Trans. Wireless Commun*, vol. 17, no. 5, pp. 3018–3029, 2018.

[48] A. Kumar and P. NandhaKumar, "Ofdm system with cyclostationary feature detection spectrum sensing," *ICT Express*, vol. 5, no. 1, pp. 21–25, 2019.

[49] P.-R. Lin, Y.-Z. Chen, P.-H. Chang, and S.-S. Jeng, "Cooperative spectrum sensing and optimization on multi-antenna energy detection in rayleigh fading channel," in *2018 27th Wireless and Optical Communication Conference (WOCC)*. IEEE, 2018, pp. 1–5.

[50] C. Jiang, H. Zhang, Y. Ren, Z. Han, K.-C. Chen, and L. Hanzo, "Machine learning paradigms for next-generation wireless networks," *IEEE Trans. Wireless Commun.*, vol. 24, no. 2, pp. 98–105, 2017.

[51] K. M. Thilina, K. W. Choi, N. Saquib, and E. Hossain, "Machine learning techniques for cooperative spectrum sensing in cognitive radio networks," *IEEE Journal on selected areas in communications*, vol. 31, no. 11, pp. 2209–2221, 2013.

[52] L. Li, L. Liu, J. Bai, H.-H. Chang, H. Chen, J. D. Ashdown, J. Zhang, and Y. Yi, "Accelerating model free reinforcement learning with imperfect model knowledge in dynamic spectrum access," *IEEE Internet of Things Journal*, 2020.

[53] Z. Zhou, L. Liu, S. Jere, Y. Yi *et al.*, "Rcnet: Incorporating structural information into deep rnn for mimo-ofdm symbol detection with limited training," *arXiv preprint arXiv:2003.06923*, 2020.

[54] H. Jiang, L. Li, H. He, and L. Liu, "Evolutionary search for energy-efficient distributed cooperative spectrum sensing," in *2020 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2020, pp. 567–571.

[55] H. Jiang, H. He, and L. Liu, "Dynamic spectrum access for femtocell networks: A graph neural network based learning approach," in *2020 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2020, pp. 927–931.

[56] H. Song, J. Bai, Y. Yi, J. Wu, and L. Liu, "Artificial intelligence enabled internet of things: Network architecture and spectrum access," *IEEE Computational Intelligence Magazine*, vol. 15, no. 1, pp. 44–51, 2020.

[57] Z. Zhou, L. Liu, V. Chandrasekhar, J. Zhang, and Y. Yi, "Deep reservoir computing meets 5g mimo-ofdm systems in symbol detection," in *Proceedings of the 34th AAAI Conf on Artificial Intell. AAAI Press*, 2020.

[58] L. Li, H. Chen, H.-H. Chang, and L. Liu, "Deep residual learning meets ofdm channel estimation," *IEEE Wireless Communications Letters*, 2019.

[59] Z. Zhou, L. Liu, and H.-H. Chang, "Learning for detection: Mimo-ofdm symbol detection through downlink pilots," *IEEE Transactions on Wireless Communications*, 2020.

[60] B. Shang and L. Liu, "Machine learning meets point process: Spatial spectrum sensing in user-centric networks," *IEEE Wireless Communications Letters*, vol. 9, no. 1, pp. 34–37, 2019.

[61] R. Shafin, L. Liu, V. Chandrasekhar, H. Chen, J. Reed, and J. C. Zhang, "Artificial intelligence-enabled cellular networks: A critical path to beyond-5g and 6g," *IEEE Wireless Communications*, 2020.

[62] C. Clancy, J. Hecker, E. Stuntebeck, and T. O'Shea, "Applications of machine learning to cognitive radio networks," *IEEE Wireless Communications*, vol. 14, no. 4, 2007.

[63] W. Lee, M. Kim, and D.-H. Cho, "Deep cooperative sensing: Cooperative spectrum sensing based on convolutional neural networks," *IEEE Trans. Veh. Technol*, vol. 68, no. 3, pp. 3005–3009, 2019.

[64] D. Han, G. C. Sobabe, C. Zhang, X. Bai, Z. Wang, S. Liu, and B. Guo, "Spectrum sensing for cognitive radio based on convolution neural network," in *2017 10th Int. Congr. Image Signal Process., Biomed. Eng. Inform (CISP-BMEI)*. IEEE, 2017, pp. 1–6.

[65] Q. Cheng, Z. Shi, D. Nguyen, and D. Erik, "Sensing ofdm signal: A deep learning approach," *IEEE Trans. Commun*, vol. 67, no. 11, pp. 7785–7798, Nov 2019.

[66] R. Shafin, H. Chen, Y. H. Nam, S. Hur, J. Park, J. Zhang, J. Reed, and L. Liu, "Self-tuning sectorization: Deep reinforcement learning meets broadcast beam optimization," *IEEE Transactions on Wireless Communications*, 2020.

[67] H. Chen, L. Liu, H. S. Dhillon, and Y. Yi, "Qos-aware d2d cellular networks with spatial spectrum sensing: A stochastic geometry view," *IEEE Transactions on Communications*, vol. 67, no. 5, pp. 3651–3664, 2018.

[68] H.-H. Chang, H. Song, Y. Yi, J. Zhang, H. He, and L. Liu, "Distributive dynamic spectrum access through deep reinforcement learning: A reservoir computing based approach," *IEEE Internet of Things Journal*, 2018.

[69] S. Mosleh, L. Liu, C. Sahin, Y. R. Zheng, and Y. Yi, "Brain-inspired wireless communications: Where reservoir computing meets mimo-ofdm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4694 – 4708, 2017.

[70] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Commun. Surveys Tuts*, vol. 20, no. 4, pp. 2595–2621, 2018.

[71] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman *et al.*, "Opportunities and obstacles for deep learning in biology and medicine," *J. Roy. Soc. Interface*, vol. 15, no. 141, p. 20170387, 2018.

[72] H. Jiang, H. He, L. Liu, and Y. Yi, "Q-learning for non-cooperative channel access game of cognitive radio networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–7.

[73] R. Atat, L. Liu, H. Chen, J. Wu, H. Li, and Y. Yi, "Enabling cyber-physical communication in 5g cellular networks: challenges, spatial spectrum sensing, and cybersecurity," *IET Cyber-Physical Systems: Theory & Applications*, vol. 2, no. 1, pp. 49–54, 2017.

[74] H. Chen, L. Liu, T. Novlan, J. D. Matyjas, B. L. Ng, and J. Zhang, "Spatial spectrum sensing-based device-to-device cellular networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 11, pp. 7299–7313, 2016.

[75] L. Liu, R. Chen, S. Geirhofer, K. Sayana, Z. Shi, and Y. Zhou, "Downlink mimo in lte-advanced: Su-mimo vs. mu-mimo," *IEEE Communications Magazine*, vol. 50, no. 2, pp. 140–147, 2012.

[76] L. Cheng, Y.-C. Wu, J. Zhang, and L. Liu, "Subspace identification for doa estimation in massive/full-dimension mimo systems: Bad data mitigation and automatic source

enumeration," *IEEE Transactions on Signal Processing*, vol. 63, no. 22, pp. 5897–5909, 2015.

[77] L. Liu, Y. Li, and J. Zhang, "Doa estimation and achievable rate analysis for 3d millimeter wave massive mimo systems," in *2014 IEEE 15th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2014, pp. 6–10.

[78] L. Liu, G. Miao, and J. Zhang, "Energy-efficient scheduling for downlink multi-user mimo," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 4394–4394.

[79] L. Liu, Y. Yi, J.-F. Chamberland, and J. Zhang, "Energy-efficient power allocation for delay-sensitive multimedia traffic over wireless systems," *IEEE transactions on vehicular technology*, vol. 63, no. 5, pp. 2038–2047, 2014.

[80] B. Kerrigan, F. L. Pour, and D. S. Ha, "System design of a high-temperature downhole transceiver: Part i–receiver," in *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2019, pp. 818–821.

[81] Y. Mao, Y. Luo, J. Zhang, and K. B. Letaief, "Energy harvesting small cell networks: feasibility, deployment, and operation," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 94–101, 2015.

[82] E. F. Camacho, T. Samad, M. Garcia-Sanz, and I. Hiskens, "Control for renewable energy and smart grids," *The Impact of Control Technology, Control Systems Society*, pp. 69–88, 2011.

[83] P. He and L. Zhao, "Noncommutative composite water-filling for energy harvesting and smart power grid hybrid system with peak power constraints," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 4, pp. 2026–2037, 2016.

[84] J. Wu, S. Guo, J. Li, and D. Zeng, "Big data meet green challenges: Greening big data," *IEEE Systems Journal*, vol. 10, no. 3, pp. 873–887, 2016.

[85] S. Soter and R. Wegener, "Development of induction machines in wind power technology," in *Electric Machines & Drives Conference, 2007. IEMDC'07. IEEE International*, vol. 2. IEEE, 2007, pp. 1490–1495.

[86] B. Kerrigan, F. L. Pour, and D. S. Ha, "System design of a high-temperature downhole transceiver: Part ii–transmitter," in *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2019, pp. 822–825.

[87] 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.

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

[89] S. Mosleht, C. Sahint, L. Liut, R. Zheng, and Y. Yit, "An energy efficient decoding scheme for nonlinear mimo-ofdm network using reservoir computing," in *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016, pp. 1166–1173.

[90] S. Mosleh, L. Liu, C. Sahin, Y. R. Zheng, and Y. Yi, "Brain-inspired wireless communications: Where reservoir computing meets mimo-ofdm," *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

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

[92] B. Schrauwen, D. Verstraeten, and J. Van Campenhout, "An overview of reservoir computing: theory, applications and implementations," in *Proceedings of the 15th European Symposium on Artificial Neural Networks. p. 471-482 2007*, 2007, pp. 471–482.

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

[94] L. Appeltant *et al.*, "Reservoir computing based on delay-dynamical systems," Ph.D. dissertation, Universitat de les Illes Balears, 2012.

[95] 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.

[96] D. S. Reich, F. Mechler, K. P. Purpura, and J. D. Victor, "Interspike intervals, receptive fields, and information encoding in primary visual cortex," *Journal of Neuroscience*, vol. 20, no. 5, pp. 1964–1974, 2000.

[97] S. M. Chase and E. D. Young, "First-spike latency information in single neurons increases when referenced to population onset," *Proceedings of the National Academy of Sciences*, vol. 104, no. 12, pp. 5175–5180, 2007.

[98] J. Lisman, "The theta/gamma discrete phase code occuring during the hippocampal phase precession may be a more general brain coding scheme," *Hippocampus*, vol. 15, no. 7, pp. 913–922, 2005.

[99] R. FitzHugh, "Impulses and physiological states in theoretical models of nerve membrane," *Biophysical journal*, vol. 1, no. 6, pp. 445–466, 1961.

[100] 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.

[101] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, p. 13, 2011.

[102] R. Xu, R. Wang, Z. Guan, L. Wu, J. Wu, and X. Du, "Achieving efficient detection against false data injection attacks in smart grid," *IEEE Access*, vol. 5, pp. 13 787–13 798, 2017.

[103] S. Tan, D. De, W.-Z. Song, J. Yang, and S. K. Das, "Survey of security advances in smart grid: A data driven approach," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 397–422, 2017.

[104] ——, "Survey of security advances in smart grid: A data driven approach," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 1, pp. 397–422, 2017.

[105] R. Deng, G. Xiao, and R. Lu, "Defending against false data injection attacks on power system state estimation," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 1, pp. 198–207, 2017.

[106] S. Bi and Y. J. Zhang, "Graphical methods for defense against false-data injection attacks on power system state estimation," *IEEE Transactions on Smart Grid*, vol. 5, no. 3, pp. 1216–1227, 2014.

[107] M. Cramer, P. Goergens, and A. Schnettler, "Bad data detection and handling in distribution grid state estimation using artificial neural networks," in *PowerTech, 2015 IEEE Eindhoven*. IEEE, 2015, pp. 1–6.

[108] A. Anwar, A. N. Mahmood, and Z. Tari, "Identification of vulnerable node clusters against false data injection attack in an ami based smart grid," *Information Systems*, vol. 53, pp. 201–212, 2015.

[109] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, "Information processing using a single dynamical node as complex system," *Nature communications*, vol. 2, p. 468, 2011.

[110] R. D. Zimmerman, C. E. Murillo-Sánchez, R. J. Thomas *et al.*, "Matpower: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Trans. Power Syst.*, vol. 26, no. 1, pp. 12–19, 2011.

[111] Y.-H. Liu and X.-J. Wang, "Spike-frequency adaptation of a generalized leaky integrate-and-fire model neuron," *Journal of computational neuroscience*, vol. 10, no. 1, pp. 25–45, 2001.

[112] J. Li, L. Liu, C. Zhao, K. Hamedani, R. Atat, and Y. Yi, "Enabling sustainable cyber physical security systems through neuromorphic computing," *IEEE Transactions on Sustainable Computing*, vol. 3, no. 2, pp. 112–125, 2018.

[113] J. Li, C. Zhao, and Y. Yi, "Energy efficient and compact analog integrated circuit design for delay-dynamical reservoir computing system," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, 2017.

[114] K. Bai and Y. Yi, "Dfr: An Energy-efficient Analog Delay Feedback Reservoir Computing System for Brain-inspired Computing," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 14, no. 4, pp. 1–22, 2018.

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

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

[117] A. Basu and P. E. Hasler, "Nullcline-based design of a silicon neuron," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 11, pp. 2938–2947, 2010.

[118] K. Ramanaiah and S. Sridhar, "Hardware implementation of artificial neural networks," *i-Manager's Journal on Embedded Systems*, vol. 3, no. 4, p. 31, 2014.

[119] M. A. Ehsan, H. An, Z. Zhou, and Y. Yi, "Design challenges and methodologies in 3d integration for neuromorphic computing systems," in *2016 17th International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2016, pp. 24–28.

[120] ——, "A novel approach for using tsvs as membrane capacitance in neuromorphic 3-d ic," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1640–1653, 2017.

[121] H. An, J. Li, Y. Li, X. Fu, and Y. Yi, "Three dimensional memristor-based neuromorphic computing system and its application to cloud robotics," *Computers & Electrical Engineering*, vol. 63, pp. 99–113, 2017.

[122] M. A. Ehsan, Z. Zhou, and Y. Yi, "Hybrid three-dimensional integrated circuits: A viable solution for high efficiency neuromorphic computing," in *2017 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. IEEE, 2017, pp. 1–2.

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

[124] M. A. Ehsan, Z. Zhou, and Y. Yi, "Modeling and analysis of neuronal membrane electrical activities in 3d neuromorphic computing system," in *2017 IEEE International Symposium on Electromagnetic Compatibility & Signal/Power Integrity (EMCSI)*. IEEE, 2017, pp. 745–750.

[125] ——, "Modeling and optimization of tsv for crosstalk mitigation in 3d neuromorphic system," in *2016 IEEE International Symposium on Electromagnetic Compatibility (EMC)*. IEEE, 2016, pp. 621–626.

[126] C. Zhao, J. Li, and Y. Yi, "Making neural encoding robust and energy efficient: an advanced analog temporal encoder for brain-inspired computing systems," in *Proceedings of the 35th International Conference on Computer-Aided Design*, 2016, pp. 1–6.

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

[128] M. Tateno and A. Uchida, "Nonlinear dynamics and chaos synchronization in mackey-glass electronic circuits with multiple time-delayed feedback," *Nonlinear Theory and Its Applications, IEICE*, vol. 3, no. 2, pp. 155–164, 2012.

[129] R. N. Anderson, A. Boulanger, W. B. Powell, and W. Scott, "Adaptive stochastic control for the smart grid," *Proceedings of the IEEE*, vol. 99, no. 6, pp. 1098–1115, 2011.

[130] O. Kosut, L. Jia, R. J. Thomas, and L. Tong, "Malicious data attacks on the smart grid," *IEEE Transactions on Smart Grid*, vol. 2, no. 4, pp. 645–658, 2011.

[131] J. Yan, B. Tang, and H. He, "Detection of false data attacks in smart grid with supervised learning," in *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016, pp. 1395–1402.

[132] Y. Kumar, "Study of power and renewable systems modeling and simulation tools," 2015.

[133] O. Bekri, M. Fellah, and M. Benkhoris, "Impact of the wind generator on the power flow in the electric grid," in *Environmental Friendly Energies and Applications (EFEA), 2014 3rd International Symposium on.* IEEE, 2014, pp. 1–6.

[134] W. T. P. Calculations, "Rwe npower renewables," *Mechanical and Electrical Engineering Power Industry, The Royal Academy of Engineering*, 2012.

[135] A. Patrascu and V.-V. Patriciu, "Cyber protection of critical infrastructures using supervised learning," in *Control Systems and Computer Science (CSCS), 2015 20th International Conference on.* IEEE, 2015, pp. 461–468.

[136] 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.

[137] K. Hamedani, S. A. Seyyedsalehi, and R. Ahamdi, "Video-based face recognition and image synthesis from rotating head frames using nonlinear manifold learning by neural networks," *Neural Computing and Applications*, vol. 27, no. 6, pp. 1761–1769, 2016.

[138] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge & Data Engineering*, no. 9, pp. 1263–1284, 2008.

[139] F. Wu, X.-Y. Jing, S. Shan, W. Zuo, and J.-Y. Yang, "Multiset feature learning for highly imbalanced data classification." in *AAAI*, 2017, pp. 1583–1589.

[140] K. Yang, J. Ren, Y. Zhu, and W. Zhang, "Active learning for wireless iot intrusion detection," *IEEE Wireless Commun.*, vol. 25, no. 6, pp. 19–25, December 2018.

[141] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, p. 13, 2011.

[142] J. Kim, L. Tong, and R. J. Thomas, "Dynamic attacks on power systems economic dispatch," in *Signals, Systs and Computers (ACSSC), 2014 48th Asilomar Conf on.* IEEE, 2014, pp. 345–349.

[143] S. Cui, Z. Han, S. Kar, T. T. Kim, H. V. Poor, and A. Tajer, "Coordinated data-injection attack and detection in the smart grid: A detailed look at enriching detection solutions," *IEEE Signal Process. Mag.*, vol. 29, no. 5, pp. 106–115, 2012.

[144] M. Ozay, I. Esnaola, F. T. Y. Vural, S. R. Kulkarni, and H. V. Poor, "Machine learning methods for attack detection in the smart grid," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 8, pp. 1773–1786, 2016.

[145] J. Yan, B. Tang, and H. He, "Detection of false data attacks in smart grid with supervised learning," in *Neural Nets (IJCNN), 2016 Intl Joint Conf on.* IEEE, 2016, pp. 1395–1402.

[146] Y. He, G. J. Mendis, and J. Wei, "Real-time detection of false data injection attacks in smart grid: A deep learning-based intelligent mechanism," *IEEE Trans. Smart Grid*, vol. 8, no. 5, pp. 2505–2516, 2017.

[147] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, "Information processing using a single dynamical node as complex system," *Nature communications*, vol. 2, p. 468, 2011.

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

[149] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proceedings of the National Academy of Sciences (NAS)*, vol. 113, no. 41, pp. 11 441–11 446, 2016.

[150] C. Zhao, B. T. Wysocki, Y. Liu, C. D. Thiem, N. R. McDonald, and Y. Yi, "Spike-time-dependent encoding for neuromorphic processors," *J. Emerg. Technol. Comput. Syst.*, vol. 12, no. 3, pp. 23:1–23:21, Sep. 2015.

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

[152] K. Ramezanpour, P. Ampadu, and W. Diehl, "A statistical fault analysis methodology for the ascon authenticated cipher," in *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE, 2019, pp. 41–50.

[153] ——, "Fima: Fault intensity map analysis," in *International Workshop on Constructive Side-Channel Analysis and Secure Design*. Springer, 2019, pp. 63–79.

[154] ——, "Fault intensity map analysis with neural network key distinguisher," in *Proceedings of the 3rd ACM Workshop on Attacks and Solutions in Hardware Security Workshop*, 2019, pp. 33–42.

[155] M. Esmalifalak, G. Shi, Z. Han, and L. Song, "Bad data injection attack and defense in electricity market using game theory study," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 160–169, 2013.

[156] 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.

[157] 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.

[158] 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.

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

[160] C. L. Giles and T. Maxwell, "Learning, invariance, and generalization in high-order neural networks," *Applied optics*, vol. 26, no. 23, pp. 4972–4978, 1987.

[161] R. Hegger, M. J. Bünner, H. Kantz, and A. Giaquinta, "Identifying and modeling delay feedback systems," *Physical review letters*, vol. 81, no. 3, p. 558, 1998.

[162] T. J. Gawne, T. W. Kjaer, and B. J. Richmond, "Latency: another potential code for feature binding in striate cortex," *Journal of neurophysiology*, vol. 76, no. 2, pp. 1356–1360, 1996.

[163] Z. Nadasdy, "Information encoding and reconstruction from the phase of action potentials," *Frontiers in systems neuroscience*, vol. 3, p. 6, 2009.

[164] C. Zhao, W. Danesh, B. T. Wysocki, and Y. Yi, "Neuromorphic encoding system design with chaos based cmos analog neuron," in *2015 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*. IEEE, 2015, pp. 1–6.

[165] R. Shafin, L. Liu, J. Zhang, and Y.-C. Wu, "Doa estimation and capacity analysis for 3-d millimeter wave massive-mimo/fd-mimo ofdm systems," *IEEE Transactions on Wireless Communications*, vol. 15, no. 10, pp. 6963–6978, 2016.

[166] Y. Li, P. Fan, A. Leukhin, and L. Liu, "On the spectral and energy efficiency of full-duplex small-cell wireless systems with massive mimo," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 3, pp. 2339–2353, 2016.

[167] R. Shafin and L. Liu, "Superimposed pilot for multi-cell multi-user massive fd-mimo systems," *IEEE Transactions on Wireless Communications*, 2020.

[168] ——, "Multi-cell multi-user massive fd-mimo: Downlink precoding and throughput analysis," *IEEE Transactions on Wireless Communications*, vol. 18, no. 1, pp. 487–502, 2018.

[169] F. E. Mahmood, E. S. Perrins, and L. Liu, "Energy consumption vs. bit rate analysis toward massive mimo systems," in *2018 IEEE International Smart Cities Conference (ISC2)*. IEEE, 2018, pp. 1–7.

[170] W. Xu, W. Xiang, M. Elkashlan, and H. Mehrpouyan, "Spectrum sensing of ofdm signals in the presence of carrier frequency offset," *IEEE Trans. Veh. Techno*, vol. 65, no. 8, pp. 6798–6803, 2015.

[171] T. J. O'Shea, S. Hitefield, and J. Corgan, "End-to-end radio traffic sequence recognition with recurrent neural networks," in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2016, pp. 277–281.

[172] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct Training for Spiking Neural Networks: Faster, Farger, Better," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1311–1318.

[173] K. Davaslioglu and Y. E. Sagduyu, "Generative adversarial learning for spectrum sensing," *IEEE International Conference on Communications (ICC)*, 2018.

[174] Y. Bengio, I. J. Goodfellow, and A. Courville, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[175] M. Wellens, A. de Baynast, and P. Mahonen, "Exploiting historical spectrum occupancy information for adaptive spectrum sensing," in *2008 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2008, pp. 717–722.

[176] H. Song, X. Fang, and Y. Fang, "Unlicensed spectra fusion and interference coordination for lte systems," *IEEE Trans. Mobile Comput.*, vol. 15, no. 12, pp. 3171–3184, 2016.

[177] Y.-C. Liang, Y. Zeng, E. C. Peh, and A. T. Hoang, "Sensing-throughput tradeoff for cognitive radio networks," *IEEE transactions on Wireless Communications*, vol. 7, no. 4, pp. 1326–1337, 2008.

[178] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[179] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[180] R. Brasselet, S. Panzeri, N. K. Logothetis, and C. Kayser, "Neurons with stereotyped and rapid responses provide a reference frame for relative temporal coding in primate auditory cortex," *J. of Neuroscience*, vol. 32, no. 9, pp. 2998–3008, 2012.

[181] M. Bertram, C. Beta, M. Pollmann, A. S. Mikhailov, H. H. Rotermund, and G. Ertl, "Pattern formation on the edge of chaos: Experiments with co oxidation on a pt (110)

surface under global delayed feedback," *Physical Review E*, vol. 67, no. 3, p. 036208, 2003.

[182] L. Chua, V. Sbitnev, and H. Kim, "Neurons are poised near the edge of chaos," *Intl J. of Bifurcation and Chaos*, vol. 22, no. 04, p. 1250098, 2012.

[183] K. Hamedani, L. Liu, S. Liu, H. He, and Y. Yi, "Deep spiking delayed feedback reservoirs and its application in spectrum sensing of mimo-ofdm dynamic spectrum sharing," in *Proceedings of the 34th AAAI Conf on Artificial Intell. AAAI Press*, 2020.

[184] A. Tavanaei, M. Ghodrati, S. R. Kheradpisheh, T. Masquelier, and A. Maida, "Deep Learning in Spiking Neural Networks," *Neural Networks*, vol. 111, pp. 47–63, 2019.

[185] A. Gilra and W. Gerstner, "Non-linear Motor Control by Local Learning in Spiking Neural Networks," in *Proceedings of the 35th International Conference on Machine Learning*, vol. 80. PMLR, 10–15 Jul 2018, pp. 1773–1782.

[186] H. Mostafa, "Supervised Learning Based on Temporal Coding in Spiking Neural Networks," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 7, pp. 3227–3235, 2017.

[187] M. M. Khan, D. R. Lester, L. A. Plana, A. Rast, X. Jin, E. Painkras, and S. B. Furber, "Spinnaker: Mapping Neural Networks onto a Massively-Parallel Chip Multiprocessor," in *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. Ieee, 2008, pp. 2849–2856.

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

[189] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A Neuromorphic Many core Processor with On-Chip Learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[190] H. An, Z. Zhou, and Y. Yi, "3d memristor-based adjustable deep recurrent neural network with programmable attention mechanism," in *Proceedings of the Neuromorphic Computing Symposium*, 2017, pp. 1–6.

[191] H. An, M. S. Al-Mamun, M. K. Orlowski, and Y. Yi, "Learning accuracy analysis of memristor-based nonlinear computing module on long short-term memory," in *Proceedings of the International Conference on Neuromorphic Systems*, 2018, pp. 1–7.

[192] Y. Yi, "Neuron design in neuromorphic computing systems and its application in wireless communications," The University of Kansas Center for Research, Inc. Lawrence, Tech. Rep., 2017.

[193] M. A. Ehsan, Z. Zhou, and Y. Yi, "Three dimensional integration technology applied to neuromorphic hardware implementation," in *2015 IEEE International Symposium on Nanoelectronic and Information Systems*. IEEE, 2015, pp. 203–206.

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

[195] Y. Yi, "Analog integrated circuit design for spike time dependent encoder and reservoir in reservoir computing processors," University of Kansas Center for Research, Inc. Lawrence United States, Tech. Rep., 2018.

[196] S. G. Wysoski, L. Benuskova, and N. Kasabov, "Evolving Spiking Neural Networks for Audiovisual Information Processing," *Neural Networks*, vol. 23, no. 7, pp. 819–835, 2010.

[197] B. Meftah, O. Lezoray, and A. Benyettou, "Segmentation and Edge Detection Based on Spiking Neural Network Model," *Neural Processing Letters*, vol. 32, no. 2, pp. 131–146, 2010.

[198] A. Tavanaei and A. Maida, "Bio-Inspired Multi-Layer Siking Neural Network Extracts Discriminative Features from Speech Signals," in *International conference on neural information processing*. Springer, 2017, pp. 899–908.

[199] S. Loiselle, J. Rouat, D. Pressnitzer, and S. Thorpe, "Exploration of Rank Order Coding with Spiking Neural Networks for Speech Recognition," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4. IEEE, 2005, pp. 2076–2080.

[200] W. Zhang and P. Li, "Spike-Train Level Backpropagation for Training Deep Recurrent Spiking Neural Networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 7800–7811.

[201] R. H. Fujii and K. Oozeki, "Temporal Data Encoding and Sequence Learning with Spiking Neural Networks," in *International Conference on Artificial Neural Networks*. Springer, 2006, pp. 780–789.

[202] Y. Zuo, H. Safaai, G. Notaro, A. Mazzoni, S. Panzeri, and M. E. Diamond, "Complementary Contributions of Spike Timing and Spike Rate to Perceptual Decisions in Rat S1 and S2 Cortex," *Current Biology*, vol. 25, no. 3, pp. 357–363, 2015.

[203] A. Marchisio, G. Nanfa, F. Khalid, M. A. Hanif, M. Martina, and M. Shafique, "Snn Under Attack: Are Spiking Deep Belief Networks Vulnerable to Adversarial Examples?" *arXiv preprint arXiv:1902.01147*, 2019.

[204] L. Liang, X. Hu, L. Deng, Y. Wu, G. Li, Y. Ding, P. Li, and Y. Xie, "Exploring Adversarial Attack in Spiking Neural Networks with Spike-Compatible Gradient," *arXiv preprint arXiv:2001.01587*, 2020.

[205] A. Bagheri, O. Simeone, and B. Rajendran, "Adversarial Training for Probabilistic Spiking Neural Networks," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. IEEE, 2018, pp. 1–5.

[206] G. Dion, S. Mejaouri, and J. Sylvestre, "Reservoir Computing with a Single Delay-coupled Non-linear Mechanical Oscillator," *Journal of Applied Physics*, vol. 124, no. 15, p. 152132, 2018.

[207] C. Du, F. Cai, M. A. Zidan, W. Ma, S. H. Lee, and W. D. Lu, "Reservoir Computing Using Dynamic Memristors for Temporal Information Processing," *Nature communications*, vol. 8, no. 1, p. 2204, 2017.

[208] C. Sanderson and B. C. Lovell, "Multi-Region Probabilistic Histograms for Robust and Scalable Identity Inference," in *International conference on biometrics*. Springer, 2009, pp. 199–208.

[209] P. Korshunov and S. Marcel, "Deepfakes: A New Threat to Face Recognition? Assessment and Detection," *arXiv preprint arXiv:1812.08685*, 2018.

[210] L. Larger, A. Baylón-Fuentes, R. Martinenghi, V. S. Udaltsov, Y. K. Chembo, and M. Jacquot, "High-speed Photonic Reservoir Computing Using a Time-delay-based Architecture: Million Words per Second Classification," *Physical Review X*, vol. 7, no. 1, p. 011015, 2017.

[211] A. N. Burkitt, "A Review of the Integrate-and-Fire Neuron Model: I. Homogeneous Synaptic Input," *Biological cybernetics*, vol. 95, no. 1, pp. 1–19, 2006.

[212] W. Nicola and C. Clopath, "Supervised Learning in Spiking Neural Networks with FORCE Training," *Nature communications*, vol. 8, no. 1, pp. 1–15, 2017.

[213] 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.

[214] M. J. Tovee, E. T. Rolls, A. Treves, and R. P. Bellis, "Information Encoding and the Responses of Single Neurons in the Primate Temporal Visual Cortex," *Journal of Neurophysiology*, vol. 70, no. 2, pp. 640–654, 1993.

[215] W. Mau, D. W. Sullivan, N. R. Kinsky, M. E. Hasselmo, M. W. Howard, and H. Eichenbaum, "The Same Hippocampal Ca1 Population Simultaneously Codes Temporal Information Over Multiple Timescales," *Current Biology*, vol. 28, no. 10, pp. 1499–1508, 2018.

[216] Y. Zheng, S. Li, R. Yan, H. Tang, and K. C. Tan, "Sparse Temporal Encoding of Visual Features for Robust Object Recognition by Spiking Neurons," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 12, pp. 5823–5833, 2018.

[217] Q. Yu, H. Tang, J. Hu, and K. C. Tan, "A Spike-Timing Based Integrated Model for Pattern Recognition," in *Neuromorphic Cognitive Systems*. Springer, 2017, pp. 43–63.

[218] C. Zhao, B. T. Wysocki, C. D. Thiem, N. R. McDonald, J. Li, L. Liu, and Y. Yi, "Energy Efficient Spiking Temporal Encoder Design for Neuromorphic Computing Systems," *IEEE Transactions on Multi-Scale Computing Systems*, vol. 2, no. 4, pp. 265–276, 2016.

[219] M. Tatsuno, *Analysis and Modeling of Coordinated Multi-Neuronal Activity*. Springer, 2015.

[220] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation As a Defense to Adversarial Perturbations Against Deep Neural Networks," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 582–597.

[221] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The Limitations of Deep Learning in Adversarial Settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.

[222] G. Goswami, N. Ratha, A. Agarwal, R. Singh, and M. Vatsa, "Unravelling robustness of deep learning based face recognition against adversarial attacks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[223] N. Akhtar and A. Mian, "Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey," *IEEE Access*, vol. 6, pp. 14 410–14 430, 2018.

[224] G. Dalal, E. Gilboa, and S. Mannor, "Hierarchical Decision Making in Electricity Grid Management," in *International Conference on Machine Learning*, 2016, pp. 2197–2206.

[225] Y. Liu, P. Ning, and M. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, vol. 14, no. 1, pp. 21–32, 2011.

[226] A. Sanjab and W. Saad, "Data Injection Attacks on Smart Grids with Multiple Adversaries: A Game-Theoretic Perspective," *IEEE Transactions on Smart Grid*, vol. 7, no. 4, pp. 2038–2049, 2016.

[227] J. Kim, L. Tong, and R. J. Thomas, "Dynamic Attacks on Power Systems Economic Dispatch," in *2014 48th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2014, pp. 345–349.